Industrial scale large eddy simulations (LES) with adaptive octree meshes using immersogeometric analysis

Kumar Saurabh[a,1], Boshun Gao[a,1], Milinda Fernando[c], Songzhe Xu[c], Makrand A. Khanwale[a,b], Biswajit Khara[a], Ming-Chen Hsu[a], Adarsh Krishnamurthy[a], Hari Sundar[c], Baskar Ganapathysubramanian[a,*]

[a]*Department of Mechanical Engineering, Iowa State University, Ames, Iowa 50010, USA*
[b]*Department of Mathematics, Iowa State University, Ames, Iowa 50010, USA*
[c]*School of Computing, The University of Utah, Salt Lake City, Utah 84112, USA*

Abstract

We present a variant of the immersed boundary method integrated with octree meshes for highly efficient and accurate Large-Eddy Simulations (LES) of flows around complex geometries. We demonstrate the scalability of the proposed method up to $O(32K)$ processors. This is achieved by (a) rapid in-out tests; (b) adaptive quadrature for an accurate evaluation of forces; (c) tensorized evaluation during matrix assembly. We showcase this method on two non-trivial applications: accurately computing the drag coefficient of a sphere across Reynolds numbers $1 - 10^6$ encompassing the *drag crisis* regime; simulating flow features across a semi-truck for investigating the effect of platooning on efficiency.

Keywords: Weak boundary conditions, Immersogeometric Analysis, octrees, Continuous Galerkin method, drag crisis, distributed parallel computing.

Highlights

- Deployed VMS with weak BCs on massively parallel octree-based adaptive meshes.
- Developed octree parallelization, efficient matrix assembly, and rapid in-out tests.
- Demonstrated the ability to capture drag crisis without any wall treatment.
- Demonstrated scalability of the framework up to $O(32K)$ processors.
- Deployed for industrial scale study of the platooning effect of semi-trucks.

1. Introduction

There has been a significant demand for the development of fast, scalable numerical methods that can run industrial scale Large Eddy Simulations (LES). An ambitious goal of the community is to perform LES over a complex geometry overnight [1]. One bottleneck to this goal is creating an analysis-suitable, body-fitted mesh with appropriate refinement for complex geometry, which is time consuming and usually a labor intensive process. This becomes even more challenging for moving bodies, as deforming meshes or remeshing is required at every time step. The main motivation behind immersed boundary methods (IBMs) is to alleviate these time consuming and laborious process. The origin of IBM dates back to 1972 when Peskin [2] utilized it to solve a cardiac blood flow problem on a Cartesian grid. This highlights the major advantage of IBM to perform the complete simulation on a structured grid, and thus avoids the requirement for the grid to conform to the complex shape. Unfortunately, this also made the application of kinematic boundary conditions such as no-slip on the surface of immersed boundaries non-trivial. There have been several developments over the past two decades on the application of correct boundary condition for the immersed cells [3, 4, 5, 6, 7, 8, 9]. Interested readers are referred to the review by Mittal and Iaccarino [10].

In this work, we consider a variant of IBM, known as Immersogeometric Analysis (IMGA), used in the context of Finite Element (FE) and Isogeometric [11] simulations. In IMGA, the surface representation of the body in the form of boundary representation (B-rep,

*Corresponding author
Email addresses: maksbh@iastate.edu (Kumar Saurabh), boshun@iastate.edu (Boshun Gao), milinda@cs.utah.edu (Milinda Fernando), songzhex@sci.utah.edu (Songzhe Xu), khanwale@iastate.edu (Makrand A. Khanwale), bkhara@iastate.edu (Biswajit Khara), jmchsu@iastate.edu (Ming-Chen Hsu), adarsh@iastate.edu (Adarsh Krishnamurthy), hari@cs.utah.edu (Hari Sundar), baskarg@iastate.edu (Baskar Ganapathysubramanian)
[1]These authors contributed equally

NURBS or .stl) is immersed into a non-body-fitted spatial discretization, thereby preserving the exact representation of the immersed geometry. Additionally, the Dirichlet boundary conditions on the immersed geometry surfaces are enforced weakly using Nitsche's method [12]. This variational weakening of the no-slip condition into a Neumann type condition provides a consistent and robust way of enforcing kinematic conditions, especially in the context of FE analysis [13].

IMGA has been deployed by several research groups for a variety of fluid-structure interaction (FSI) simulations, including complex biomedical applications with NURBS [14, 15, 16, 17], design optimization [18], external aerodynamics simulations with tetrahedral meshes [19, 20, 21, 22], and moving objects [23, 24]. The weak imposition of the no-slip condition has been demonstrated to be numerically very advantageous, especially for flow past complex geometries where steep gradients are produced [25, 26]. However, challenges remain to practical deployment of IMGA, especially towards the goal of overnight LES. In this work, we identify and resolve the following technical bottlenecks to deploying IMGA for large scale simulations:

- *Simulations on massively parallel adaptive meshes*. In contrast to using unstructured background meshes, we utilize octree-based, parallel adaptive meshes resulting in improved scalability at extreme scales, as well as the ability to efficiently remesh.
- *Matrix assembly*: The performance of linear algebra solvers has substantially improved. Several robust optimized libraries have been developed to perform fast numerical linear algebra calculations [27, 28, 29, 30]; this has made the other parts of the code, specifically matrix assembly a major bottleneck. An analysis of our IMGA solver suggests that up to 70% of the time is spent in matrix assembly. Substantial improvements are possible by optimizing matrix assembly, which we accomplish using tensorized operations. This is a step towards our intent to transition to matrix-free methods.
- *Load imbalance*: The enforcement of kinematic constraints on Intercepted elements (see Fig. 1) requires an additional surface and volume integration in those elements. The volume integration must be performed accurately on *only that fraction* of each Intercepted element that is outside the object. This can be a large fraction of assembly time for complex geometries. We deploy a weighted, dynamic partitioning to ensure load balancing, even with adaptive quadrature.
- In - Out *test*: Classifying the location of a point in the background mesh with respect to the object (as inside or outside the object) is a quintessential IBM ingredient. We go beyond conventional ray-tracing approaches [26, 19] (which are computationally expensive) to a more efficient normal based test.
- *Reliable numerical methods*: The utility of industrial CFD (Computational Fluid Dynamics) simulations is limited by the choice of appropriate modeling strategies that can accurately predict the region of separation especially for the turbulent cases. Currently, the most widely methods used like Reynolds Averaged Navier–Stokes (RANS) or Detached Eddy Simulation (DES) rely on additional wall treatment to achieve this. Such application specific strategies limit reliability of industrial scale CFD. In this work we demonstrate the ability to predict the drag crisis phenomena *without any special treatment*, with the same method working across six orders of magnitude variation in Reynolds number.

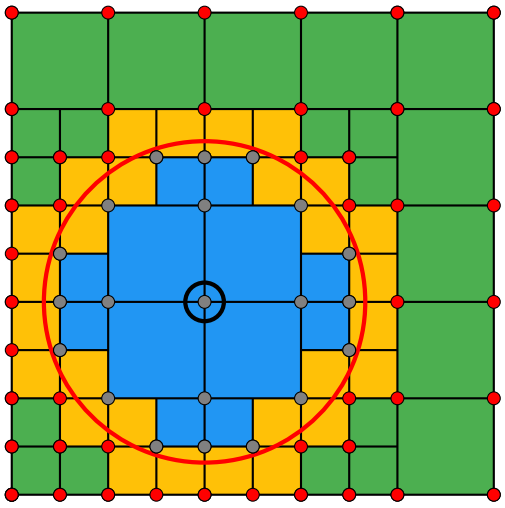


Fig. 1. Sketch illustrating the IMGA on octree based adaptive mesh. The solid red line represents the boundary of the immersed object. The elements are classified into 3 types :a) Out (■) : all nodes are outside the body; b) In (■): all nodes are inside the body; c) Intercepted (■) : some of the nodes are inside and some are outside. CG (Continuous Galerkin) nodes are divided into In (●) and Out (●) nodes. Hanging nodes are neither classified In or Out as they do not correspond to independent degrees of freedom. The nodes which neither belong to the Intercepted nor Out elements and are also not the parent of hanging nodes (marked by ○) are not solved for in IMGA simulations.

This work is motivated by challenges articulated in the *NASA CFD 2030* [31] vision towards the goal of performing overnight LES: a) *"Mesh generation and adaptivity continue to be significant bottlenecks in the CFD workflow.."*; b) *"The use of CFD ... is severely limited by the inability to accurately and reliably predict turbulent flows with significant regions of separation"*. To the best of our knowledge, this is the first time that IMGA simulations in conjunction with VMS and weak BC on a massively parallel octree-based adaptive mesh has been performed. Fig. 1 shows the representative diagram of IMGA computation on octrees. In a nutshell, our main contributions include:

1. Deploy the variational formulation of Navier–Stokes with weak enforcement of Dirichlet boundary

conditions on adaptive octree based mesh at high Reynolds numbers.

2. Deploy adaptive quadrature based schemes for accurate integration of INTERCEPTED elements.
3. Demonstrate the ability of the numerical method to capture the drag crisis without any special wall treatment.
4. Develop a fast normal based in - out test to accurately determine the points in and out of the boundary.
5. Perform near optimal load balancing by accurately estimating the weight per element (using an empirical model) to account for imbalance in the load arising due to IMGA.
6. Show scaling of our framework to $O(32K)$ processors.
7. Illustrate framework on a complex engineering problem with implications to autonomous vehicles.

The rest of the paper is organized as follows: we begin by giving a brief overview of IMGA and weak imposition of boundary condition in Section 2, the key algorithmic developments for mesh generation, matrix assembly, weighted partitioning and adaptive quadrature are outlined in Section 3; numerical and scaling benchmark results are presented and discussed in Section 4; the framework is then deployed to study the platooning effect on a semi-trailer truck in Section 5; and concluding remarks and future outlook are made in Section 6.

2. Mathematical preliminaries: Variational treatment of IMGA

The fundamentals of immersogeometric fluid-flow analysis consist of three main components. The flow physics is formulated using a variational multiscale (VMS) method for incompressible flows [32, 33]. To capture the flow domain geometry accurately, adaptively refined quadrature rules are used in the INTERCEPTED elements, without modifying the background mesh [34, 19]. Finally, the Dirichlet boundary conditions on the surface of the immersed objects are enforced weakly in the sense of Nitsche's method [12, 25]. We briefly discuss each of these components next.

2.1. Variational Multiscale Formulation (VMS)

The VMS approach has been successfully utilized to model flow physics across a range of applications [35, 23, 36, 37, 38]. It is considered to be an LES type approach, and uses variational projections in place of the traditional filtered equations in LES. The method is derived completely from the incompressible Navier–Stokes and does not employ any eddy viscosity. See [33] for a detailed derivation of the VMS method.

The VMS discretization of the Navier–Stokes equations is stated as: Find fluid velocity $\mathbf{u}^h$ and pressure p^h such that for all test functions $\mathbf{w}^h$ and q^h (defined in appropriate function spaces):

$$B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) = 0\,, \tag{1}$$

where,

$$\begin{aligned}
&B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) = \\
&\int_\Omega \mathbf{w}^h \cdot \rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega \\
&+ \int_\Omega \boldsymbol{\varepsilon}(\mathbf{w}^h) : \boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right) d\Omega + \int_\Omega q^h\, \boldsymbol{\nabla}\cdot\mathbf{u}^h d\Omega \\
&- \sum_e \int_{\Omega^e\cap\Omega} \left(\mathbf{u}^h \cdot \boldsymbol{\nabla}\mathbf{w}^h + \frac{\boldsymbol{\nabla} q^h}{\rho}\right)\cdot \mathbf{u}' d\Omega \\
&- \sum_e \int_{\Omega^e\cap\Omega} p'\, \boldsymbol{\nabla}\cdot\mathbf{w}^h d\Omega \\
&+ \sum_e \int_{\Omega^e\cap\Omega} \mathbf{w}^h \cdot (\mathbf{u}'\cdot\boldsymbol{\nabla}\mathbf{u}^h) d\Omega \\
&- \sum_e \int_{\Omega^e\cap\Omega} \frac{\boldsymbol{\nabla}\mathbf{w}^h}{\rho} : (\mathbf{u}'\otimes\mathbf{u}')\, d\Omega \\
&+ \sum_e \int_{\Omega^e\cap\Omega} \left(\mathbf{u}'\cdot\boldsymbol{\nabla}\mathbf{w}^h\right)\overline{\tau}\cdot\left(\mathbf{u}'\cdot\boldsymbol{\nabla}\mathbf{u}^h\right) d\Omega\,,
\end{aligned} \tag{2}$$

and F^{VMS} is the force term. The fine scale velocity, $\mathbf{u}'$, and pressure, p', are defined as

$$\mathbf{u}' = -\tau_{\text{M}}\left(\rho\left(\frac{\partial \mathbf{u}^h}{\partial t} + \mathbf{u}^h\cdot\boldsymbol{\nabla}\mathbf{u}^h - \mathbf{f}\right) - \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}\left(\mathbf{u}^h, p^h\right)\right), \tag{3}$$

and

$$p' = -\rho\,\tau_{\text{C}}\,\boldsymbol{\nabla}\cdot\mathbf{u}^h\,. \tag{4}$$

In the above equations, Ω^e represents the disjoint elements, such that $\Omega \subset \cup_e \Omega^e$, ρ is the density of the fluid, and $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ are the stress and strain-rate tensors, respectively. The terms integrated over element interiors may be interpreted both as stabilization and as a turbulence model [39, 40, 41, 42, 33, 43]. $\tau_{\text{M}}, \tau_{\text{C}}$ and $\overline{\tau}$ are the stabilization parameters. Their detailed expression used in this work can be found in Xu et al. [19].

2.2. Adaptive Quadrature

The geometric boundary of immersed object creates complex, discontinuous integration in the INTERCEPTED elements. In order to ensure the geometrically accurate integration of the volume integrals, we use a sub-cell based quadrature scheme [34, 15]. The basic idea is to increase the number of quadrature points in the INTERCEPTED elements and perform accurate evaluation by considering the quadrature points that are only outside the body. We discuss an efficient approach to do this later in Section 3.4.

2.3. Weak Enforcement of Boundary Conditions

The standard way of imposing Dirichlet boundary conditions is to enforce them strongly by ensuring that they are satisfied by all trial solution functions, which is not feasible in immersed methods. Instead, the strong enforcement is replaced by weakly enforced Dirichlet boundary conditions [25, 44, 45]. The semi-discrete problem can be stated as follows: Find fluid velocity $\mathbf{u}^h$ and pressure p^h such that for all test functions $\mathbf{w}^h$ and q^h:

$$
\begin{aligned}
&B^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}, \{\mathbf{u}^h, p^h\}\right) - F^{\text{VMS}}\left(\{\mathbf{w}^h, q^h\}\right) \\
&\quad - \int_{\Gamma^D} \mathbf{w}^h \cdot \left(-p^h\,\mathbf{n} + 2\mu\,\boldsymbol{\varepsilon}(\mathbf{u}^h)\,\mathbf{n}\right) d\Gamma \\
&\quad - \int_{\Gamma^D} \left(q^h\,\mathbf{n} + 2\mu\,\boldsymbol{\varepsilon}(\mathbf{w}^h)\,\mathbf{n}\right) \cdot \left(\mathbf{u}^h - \mathbf{g}\right) d\Gamma \\
&\quad - \int_{\Gamma^{D,-}} \mathbf{w}^h \cdot \rho\left(\mathbf{u}^h \cdot \mathbf{n}\right)\left(\mathbf{u}^h - \mathbf{g}\right) d\Gamma \\
&\quad + \int_{\Gamma^D} \tau^B_{\text{TAN}} \left(\mathbf{w}^h - \left(\mathbf{w}^h \cdot \mathbf{n}\right)\mathbf{n}\right) \\
&\qquad\qquad \cdot \left(\left(\mathbf{u}^h - \mathbf{g}\right) - \left(\left(\mathbf{u}^h - \mathbf{g}\right) \cdot \mathbf{n}\right)\mathbf{n}\right) d\Gamma \\
&\quad + \int_{\Gamma^D} \tau^B_{\text{NOR}} \left(\mathbf{w}^h \cdot \mathbf{n}\right)\left(\left(\mathbf{u}^h - \mathbf{g}\right) \cdot \mathbf{n}\right) d\Gamma \;=\; 0\,,
\end{aligned}
\tag{5}
$$

where $\mathbf{n}$ is the outward unit normal, μ is the dynamic viscosity, Γ^D is the Dirichlet boundary that may cut through element interiors, $\Gamma^{D,-}$ is the *inflow* part of Γ^D, on which $\mathbf{u}^h \cdot \mathbf{n} < 0$, $\mathbf{g}$ is the prescribed velocity on Γ^D, and τ^B_{TAN} and τ^B_{NOR} are stabilization parameters.

Remark. *The stabilization parameter for the weak imposition of the boundary condition is chosen such that* $\tau^B_{TAN} = \tau^B_{NOR} = \frac{C_b}{Reh_b} > 4$, *where* h_b *is computed as the maximum projected distance from the outside node*(s) *(i.e. fluid node) of the background* Intercepted *elements to the triangulated surface. Complex surface geometries can sometimes result in 'sliver cut' elements, which exhibit* $h_b \to 0$. *This can lead to arbitrarily large values of the stabilization parameter, resulting in accuracy and convergence issues* [46]. *To circumvent this issue, the lower limit on* h_b *is set to* $0.01h$, *where* h *is the grid spacing of background element.*

The weak enforcement of the boundary condition in IMGA is particularly attractive as this alleviates the need for a body-fitted mesh. The additional Nitsche terms (the third to last terms on the left-hand side of Eq. (5)) are formulated independently of the mesh. The only additional overhead is a separate discretization of the domain boundary whose position in the Intercepted elements can be determined.

Remark. *We use a fully implicit Crank Nicholson time stepping scheme. All results are reported using linear basis functions, unless explicitly stated otherwise.*

3. Algorithmic developments

In this section, we highlight some of the key algorithmic developments to accelerate IMGA computations and deploy it on massively parallel computers. Algorithm 1 gives an overview of the key steps in IMGA. We start by constructing an octree-based adaptive mesh with a constraint on the relative size of neighboring elements (2:1 balancing). The mesh is refined near the object boundary which is important to capture boundary layer effects. A brief overview of our mesh generation algorithm is given in Section 3.1. Based on the mesh boundaries, the *.stl* triangles are distributed across the processors, based on its overlap with elements that are part of the background mesh. This step is important to perform fast In - Out tests described in Section 3.3. The elements are labeled as In , Out or Intercepted based on the position of the *.stl* geometry. To avoid repeated tests these labels are stored as element markers. Elements have different routines, and computational costs, based on the element marker. For In elements, no integration is performed and Dirichlet conditions are set on all CG nodes corresponding to In elements. For Out elements, integration is performed by looping through all quadrature points. For Intercepted elements, we perform adaptive quadrature Section 3.4 and perform the integration only over Out Gauss points. The matrix assembly is an important part here and is optimized for achieving significant speedup and is discussed in Section 3.2.

3.1. Octree based Mesh Generation

Octrees are widely used in computational sciences [47, 48, 49, 50, 51, 52], due to its conceptual simplicity and ability to scale across large number of processors. Proceeding in a top-down fashion, an octant is refined based on a user-specified criteria. The refinement criteria is specified by a user-specified function that takes the coordinates of the octant, and returns `true` or `false`. Since the refinement happens in an element-local fashion, this step is embarrassingly parallel. In distributed memory, the initial top-down tree construction, also enables an efficient partitioning of the domain across an arbitrary number of processes. All processes start at the root node (i.e., the cubic bounding box for the entire domain). In order to avoid communication during the refinement stage, we opt to perform redundant computations on all processes. Starting from the root node, all processes refine (similar to the sequential code) until at least $O(p)$ octants requiring further refinement are produced. Then using a weighted space-filling-curve (SFC) based partitioning, we partition the octants across all processes. Note that we do not communicate the octants as every process has a copy of the octants, and all that needs to be done at each process is to retain a subset of the current octants corresponding to its sub-domain. This allows us to have excellent scalability, as all processes perform (roughly) the same amount of work without requiring any communication.

Algorithm 1 IMMERSOGEOMETRIC_ALGORITHM: Brief Overview

Require: The octree and *.stl* file
for *elements* ∈ *leaf* **do**
  compute element Markers ▷ Algorithm 5
  compute background triangles ($\mathcal{B}_\mathcal{T}$)
for time < T_{final} **do** ▷ Loop over time
  for *elements* ∈ *leaf* **do**
    if IN **then**
      continue ▷ Skip IN elements
    if OUT **then**
      Loop over Gauss Points
      perform Matrix and Vector Assembly
    if INTERCEPTED **then**
      Fill the elements with Gauss points
      Check if the Gauss point $\mathcal{P}$ is IN or OUT ▷ Algorithm 6
      if ($\mathcal{P}$ is OUT) **then**
        perform Matrix and Vector Assembly
  for *element* ∈ *leaf* **do**
    if INTERCEPTED **then**
      for $t \in \mathcal{B}_\mathcal{T}$ **do**
        Assemble weak BC contribution to *element*
  Solve system of equations

Upon octree generation we enforce the 2:1 balanced constraint which ensures that neighboring octants differ by at most one level. Such a 2:1 balanced constraint simplifies mesh generation, and eliminates ill-conditioning introduced due to extreme scaling of neighbor elements. Following balancing, meshing is performed which generates the required data-structures to perform FE computations, intergrid transfers, spatial queries and membership tests. Additional details on our octree-based FEM framework can be found in Fernando et al. [50], Sundar et al. [53] and Fernando and Sundar [54].

3.2. Matrix Assembly on Distributed Octrees

In FEM, the differential operator $\mathcal{L}$, after weakening and discretization becomes a matrix (stiffness matrix) $K_{ij} = (\phi_i, \mathcal{L}\phi_j)$ where $(u, v) = \int_\Omega uvd\Omega$, and ϕ_i is the basis function defined at i^{th} node. In a distributed octree, each partition τ_k of Ω will compute the K_{ij} restricted to τ_k denoted by $K_{ij|\tau_k}$. The octants in τ_k are further decomposed into two disjoint octant sets – independent τ_I, and dependent τ_D such that, $\tau_k = \tau_I \cup \tau_D$. An octant $e \in \tau_I$ ensures that all the nodal information related to e is local while τ_D is $\tau_k \setminus \tau_I$. The notion of independent and dependent octants is used to overlap computation with communication during the matrix assembly (see Fig. 2).

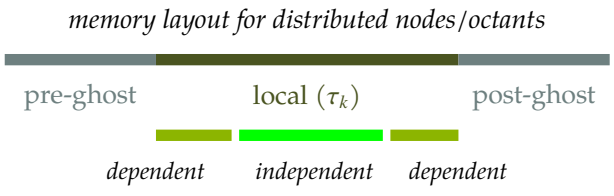


Fig. 2. Figure depicting the memory layout for the distributed nodes/octants. The distributed octree T is partitioned across p processors, where each partitioned τ_k tree (local octants) has pre (from $t\tau_m m < k$) and post (from $t\tau_m m > k$) ghost octant/nodal information. The local partition τ_k is further logically decomposed in to independent τ_I, and dependent τ_D disjoint octant sets such that $\tau_k = \tau_I \cup \tau_D$. This notion of independent and dependent is used in overlapping computation with communication in FEM matrix assembly.

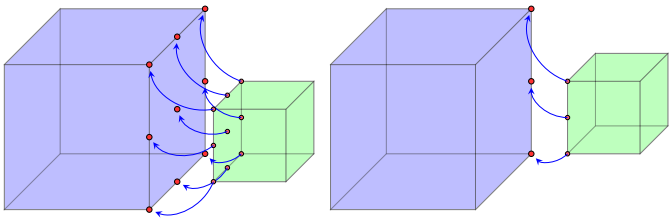


Fig. 3. An example of a hanging face and a hanging edge where in both cases octant (■) has a hanging face (left figure) and a hanging edge (right figure) with octant (■). Nodes on the hanging face/edge are mapped to the larger octant and the hanging nodal values are obtained via interpolation. Note that for illustrative purposes, the two octants are drawn separately, but are contiguous.

Algorithm 2 MATRIX_ASSEMBLY

Require: Octree T, distributed across p processors, (τ_k local partition)
Ensure: global assembled matrix K_{ij}
$K \leftarrow 0$ matrix
for $e \in \tau_D$ **do** ▷ local dependent elements
  $K_e \leftarrow compute\_ele\_matrix()$ ▷ Algorithm 4
  $M_e \leftarrow compute\_hanging\_corrections()$ ▷ Algorithm 3
  $k_e \leftarrow M_e^T K_e M_e$
start_assembly_comm() ▷ start communication
for $e \in \tau_I$ **do** ▷ local independent elements
  $k_e \leftarrow compute\_ele\_matrix()$ ▷ Algorithm 4
  $M_e \leftarrow compute\_hanging\_corrections(\tau_k, e)$ ▷ Algorithm 3
  $K_e \leftarrow M_e^T k_e M_e$
  $K \leftarrow K +$ O2N(K_e)
end_assembly_comm() ▷ wait till communication ends
for $e \in \tau_D$ **do**
  $K \leftarrow K +$ O2N(K_e)

Hanging nodes: Adaptivity in octree meshes causes non-conformity. In our framework, additional degrees of freedom on shared faces between elements of unequal sizes—the so called hanging nodes—do not represent independent degrees of freedom, and are not stored and instead are represented as a linear combination of the basis functions corresponding to the larger face (see Fig. 3). We implement this using a correction operator [49, 55, 56] during the overall matrix assembly (see Algorithm 2 and Algorithm 3).

Elemental matrix assembly: We accelerate the assembly process[2] by viewing each weakened differential FEM operator (FEM kernel) of the form (MAT_OP1, GP$_i$MAT_OP2) as an outer product and making

[2] Although we illustrate some promising results with a matrix-free approach [57], further optimization is needed—specifically in terms of tailored pre-conditioners—to make the matrix-free approach competitive.

Algorithm 3 COMPUTE_HANGING_CORRECTIONS

Require: Octree T, distributed across p processors, (τ_k local partition), local element $e \in \tau_k$
Ensure: Hanging node correction matrix M_e
1: $M_e \leftarrow I$ ▷ I is the identity matrix
2: $I2d_{ij} \leftarrow I1d_i \bigotimes I1d_j$ ▷ $I1d$ is 1d interpolation matrices
3: **for** $f \in Faces(e)$ **do**
4: **if** $is\_hanging(\tau_k, e, f)$ **then**
5: $M_e[f] \leftarrow I2d_{ij}$
6: **for** $edge \in Edges(e)$ **do**
7: **if** $is\_hanging(\tau_k, e, edge)$ **then**
8: $M_e[edge] \leftarrow I1d_i$
return M_e

use of optimized matrix-matrix multiplication libraries. Each FEM kernel comprises of two operators (MAT_OP1 and MAT_OP2) obtained as a result of weakening, and GP_i denotes the value of interpolated variables at the Gauss points. Algorithm 4 describes the procedure along with the associated complexity of each step. In this work, we represent MAT_OP as a matrix of appropriate size. For better performance, the matrix corresponding to MAT_OP can be precomputed and cached (Section Appendix A). Step 1-4 denotes the computation of second term in the kernel (GP_i MAT_OP2). W_i denotes the weight of the quadrature points. Step 5 donates the action of MAT_OP1. The scalar factor SCALE is the resultant of mapping from the reference frame to the physical frame and is operator dependent [3]. For example, the scale factor for the stiffness matrix is $\Delta x/2$ and for the mass matrix is $\Delta x^3/8$ for a reference element $\in [-1, 1]^3$ in 3D. Clearly, this last step is computationally the most expensive. For a detailed analysis, interested readers are referred to Deville et al. [58].

Our approach is different from the general approach in open source libraries (such as deal.ii [59], for instance) that loop over individual Gauss points in each direction. Although the overall theoretical complexity remains the same ($O(\text{bf}+1)^{3d}$, where bf is the basis function order and d is the number of spatial dimensions) for both implementations, implementing the operator as a matrix-matrix multiplication allows a natural way to leverage the power of GEMM kernel [60] that is available as vendor optimized libraries (such as INTEL MKL). These libraries are fine tuned to exploit assembly-code-level parallelism and are extremely fast on modern cache-based and pipelined architectures. As evident from the complexity analysis, optimized matrix assembly becomes especially important when using higher order basis functions. This is a careful middle ground that ensures portability across various computing platforms, while not as efficient as explicit vectorization.

It has been shown that the theoretical lower bound on the computational complexity for elemental matrix

[3]This makes use of the fact that the octree elements, in general, are cubes with equal aspect ratio ($\Delta x = \Delta y = \Delta z$)

Algorithm 4 COMPUTE_ELE_MATRIX

Require: FEM operators (MAT_OP1 and MAT_OP2), Quadrature Values (GP), SCALE, Size of matrix(n × n): n = $(\text{bf}+1)^d$
Ensure: The computed elemental matrix EMAT
1: GP ← $GP_i \times W_i$ ▷ O(n)
2: **for** i ← 1 to n **do**
3: **for** j ← 1 to n **do**
4: $M_{ij} \leftarrow MAT\_OP2_{ij} \times GP_j$ ▷ $O(\text{n})^2$
5: GEMM:EMAT ← SCALE(MAT_OP1^T M) ▷ $O(\text{n})^3$
return EMAT

Algorithm 5 ELEMENT_MARKERS : IN / OUT or INTERCEPTED

Require: The element $\mathcal{E}$ and list of background triangles $\mathcal{B_T}$ associated with it.
Ensure: The marker for $\mathcal{E}$ ▷ IN , OUT or INTERCEPTED
1: $counts \leftarrow 0$
2: **for** $n \in n_{nodes}$ **do**
3: compute $\mathcal{P}$ ▷ Global position of the node
4: **if** ISIN($\mathcal{P}$,$\mathcal{B_T}$) **then** ▷ Algorithm 6
5: increment count
6: **if** count == n_{nodes} **then** ▷ All nodes are IN
7: **return** IN
8: **if** count == 0 **then** ▷ All nodes are OUT
9: **return** OUT
return INTERCEPTED

assembly is $O(\text{bf}+1)^{2d}$, since the elemental matrix has many non-zero entries [58]. There has been substantial effort to achieve this lower bound [61, 62, 58]. However, most of the proposed analysis and implementation have been limited to stiffness and mass matrix. Deploying these algorithms for complicated non-linear kernels arising as a result of weakening the Navier–Stokes equation seems non-trivial and needs further analysis. This is left as future work.

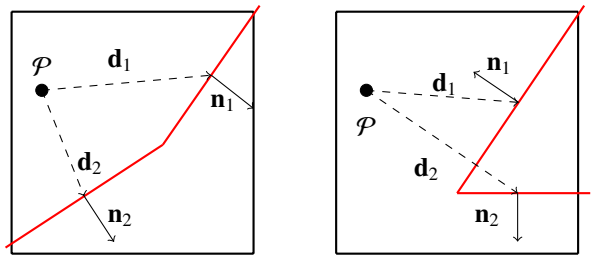
3.3. IN - OUT *Test*

In this work, we propose a new normal based identification of IN - OUT test, as opposed to ray-tracing that is conventionally use in immersed boundary simulations [63, 64, 65, 66]. Algorithm 6 describes the procedure to identify whether a given quadrature point $\mathcal{P}$ in an INTERCEPTED element is IN or OUT . Given a point $\mathcal{P}$ in an element $\mathcal{E}$, we identify the background triangles (that is stored in the data structure during partition of triangles) associated with the element. The dot product of the position vector (i.e. vector from $\mathcal{P}$ to the triangle centroid) with the normal of all the background triangles $\mathcal{B_T}$ is evaluated. If the result of the dot product with all the triangles is greater than 0, then the point is outside the geometry and vice versa. The normal based test comes at no additional cost in terms of memory requirements and involves series of simple dot product evaluation between the point and background triangles. This makes it very cost efficient as compared to ray-tracing. In case of conflict, as in case of sharp corners (see Fig. 4), we revert back to ray-tracing.

Algorithm 6 isIn : Normal based In - Out test

Require: The global position $\mathcal{P}$ and list of background triangles $\mathcal{B}_\mathcal{T}$ associated with each octant.
Ensure: Point $\mathcal{P}$ is In or Out of the given geometry

```
count ← 0
for t ∈ B_T do
    Compute d = P - t                    ▷ distance vector from P to t
    if d · n_t ≤ 0 then
        increment count
if count == |B_T| then
    return Out ▷ All background triangles indicate that point is Out
if count == 0 then
    return In  ▷ All background triangles indicate that point is In
perform Ray - Tracing                    ▷ Normal based test fails
```



(**a**) No sharp corners (**b**) Sharp corners

Fig. 4. Figure illustrating the normal based In - Out test. Fig. 4a shows the case where there is no sharp corners. In this case, the dot product of $\mathbf{d}_1 \cdot \mathbf{n}_1$ and $\mathbf{d}_2 \cdot \mathbf{n}_2$ both are greater than 0 stating that point is In . But in case of sharp corners, as shown in Fig. 4b the background triangles can give conflicting decisions. $\mathbf{d}_1 \cdot \mathbf{n}_1 < 0$ and $\mathbf{d}_2 \cdot \mathbf{n}_2 > 0$ In these cases, we resort to the ray-tracing algorithm.

3.4. Adaptive Quadrature

There has been recent advances in the development of accurate and efficient quadrature rules to account for the cut cells that arise during IMGA computations [67, 68, 69, 70, 71, 72, 73, 74, 19, 75]. In this work, to accurately account for the effect of fractional content of fluids in the Intercepted cells, we use adaptive quadrature by recursive cell subdivision. Intercepted elements are further subdivided to include additional Gauss points. This ensures the accurate integration of the fluid domain for intersected elements. The splitting of the element is done *only* to add Gauss points and does not introduce any new degrees of freedom. The Intercepted elements are recursively sub divided to fill in Gauss points till the smallest cells are completely In (or Out); or a max split level is reached. Fig. 5 demonstrates the case with maximum split level set to 2.

3.5. Dynamic Load Balancing

We use a Space Filling Curve (SFC) – specifically the Hilbert curve – to partition our octree mesh and the associated data across all processes. In the case of IMGA, different elements can have different computational loads depending on the In , Out , and Intercepted status. Intercepted elements perform the following additional computations compared to In and Out elements: a) identify whether a particular Gauss point is in or out , b) loop over additional Gauss points (due to recursive quadrature), and c) perform surface integral over the surface element for accumulation of the contribution from the weak boundary condition. We ensure load-balance across all processes by using a weighted SFC partition, wherein Intercepted elements are assigned a higher weight. The weight associated with Intercepted elements is proportional to the ratio of computational effort of an Intercepted elements to an Out element.

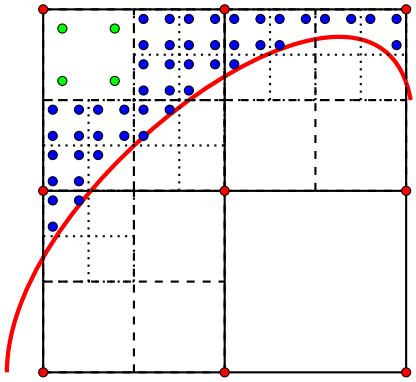


Fig. 5. Sketch demonstrating adaptive quadrature. Boundary is represented by the red line. Octant shared nodes are represented by (●). (●) represents the new Gauss points generated as the result of 1^{st} level of splitting and (●) represents the Gauss point of 2^{nd} level of splitting. It is to be noted that the splitting happens only for the Intercepted elements. Once all the Gauss points in the splitted elements are either in or out the elements is not further sub-divided.

3.5.1. Estimation of Weight for the Intercepted Elements

Let T_v be the computational cost per volumetric quadrature point and T_s be the cost per surface quadrature point. The relative weight for each element, e, can be estimated as:

$$W(e) = \frac{n_{\text{Vgp}}(e)}{n_{\text{Tgp}}} + \frac{T_s}{T_v}\frac{n_{\text{Sgp}}(e)}{n_{\text{Tgp}}} \tag{6}$$

where: n_{Vgp} is the number of Gauss point that are outside the geometry, n_{Sgp} is the total number of surface Gauss points belonging to the Intercepted elements and n_{Tgp} represents the total number of Gauss point in the volumetric elements which scales as $(\text{bf} + 1)^{\text{nsd}}$, where bf is the order of basis function. For a completely Out element and a completely In element, Eq. (6) reduces to 1 and 0, respectively.

4. Benchmark results

In this section, we present the benchmark results for our solver in terms of accuracy and speed by considering a suite of appropriate test cases. We begin by validating our framework using a canonical problem of flow past a sphere across a wide range of Reynolds numbers, encompassing the laminar-transition-turbulent regimes (Section 4.1). Next, we show the impact of tensorization

on the matrix assembly and solve time using a benchmark lid driven cavity problem (Section 4.2). We next quantify the advantage of our proposed In - Out test for significantly complex standard geometries in Section 4.3. In order to articulate the advantage of using adaptive quadrature and weighted partitioning on IMGA simulations, we revisit flow past a sphere in Section 4.5. Finally, we present the scaling results of our solver for different mesh resolutions (Section 4.6).

4.1. Validation: Flow Past a Sphere

As the first step, we validate our numerical setup by computing the non-dimensional drag coefficient C_d on sphere in a range of Reynolds number from 1 to 1,000,000. Fig. 6 shows the schematic of the computational domain used to perform the simulation for flow past the sphere. The computational domain consists of length $10d$ with the center of the sphere of diameter $d = 1$ placed at distance $3d$ from inlet (at $(3d, 5d, 5d)$). All the walls of the cubic domain, except the outlet have constant non-dimensional freestream velocity of (1, 0, 0) and zero pressure gradient. At the outlet, the pressure is set to 0 and zero gradient velocity boundary condition is applied at the wall. The no-slip boundary condition (zero Dirichlet) for velocity is weakly imposed on the surface of sphere. The octree and surface mesh resolution was varied by successively refining the mesh depending on the Reynolds number according to the Taylor micro length scale (R_{bdy} was varied from 5 to 11 with increasing Re). At the finest resolution for the case of $Re = 1$ million, we have the equivalent of around 500 elements across the diameter of sphere. The spatial domain was discretized using linear basis function. The Crank Nicholson scheme was used for time discretization.

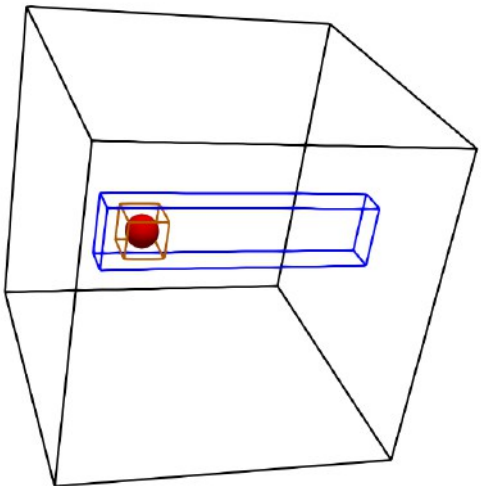


Fig. 6. Computational domain for sphere simulations. The computation domain consists of a cubic domain with dimension $10d \times 10d \times 10d$. Overall there are three different levels of refinement: R_{bkg} for the background mesh, R_{wake} within the blue box to capture the wake, and R_{bdy} near the body within the brown box. The refinement level inside the body is equal to R_{bkg}. The sphere of diameter $d = 1$ is positioned at $(3d, 5d, 5d)$. A refinement level of i would corresponding to the resolution of $\frac{10d}{2^i}$ in each direction.

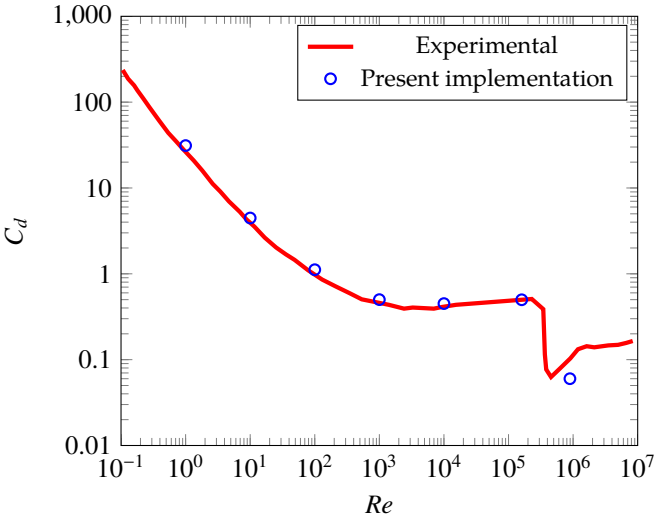
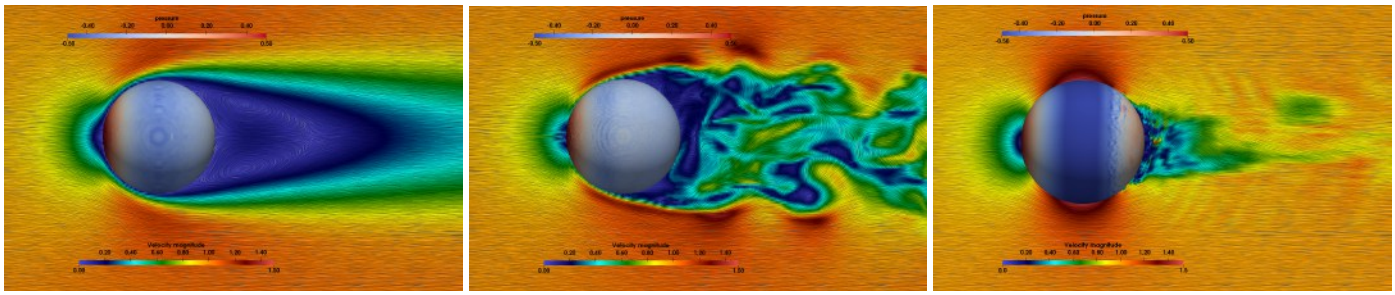


Fig. 7. *Drag crisis:* Variation of drag coefficient C_d with Re for flow past a sphere compared with experimental data [76, 77]. Notice that the x-axis is log(Re) representing variation for a wide range of Re number spanning multiple regimes (from laminar to fully developed turbulence) each with distinct flow physics. The framework demonstrates good comparison with experimental data in all these regimes without any special adjustments to the numerical scheme. Moving towards the right on a log scale as Re is increased becomes increasingly computationally expensive, e.g adding another point on the right of the last point would increase the Reynolds number ten-folds and would require a decrease in finest element size by 3 times.

Remark. *The finest resolution for $Re = 1$ million was chosen based on the work by Geier et al. [78]. This resolution has been shown to be sufficient to capture the drag crisis. Further refinement was not considered in this work due to the high computational cost.*

We evaluated the non-dimensional drag coefficient across this range of Reynolds numbers, which is plotted in Fig. 7. We can see that the IMGA results are in excellent agreement with experimental data. Note that we are able to accurately capture the drag crisis phenomena, where a sudden drop in drag coefficient is observed. The prediction of drag crisis is of significant interest to the CFD community, with important engineering implications in aerodynamics and vehicular dynamics. We emphasize that no special numerical/modelling treatment is needed in this framework to capture the drag crisis; this is in contrast to existing work where typically a wall treatment [79, 80] is required to predict the drag crisis.

Fig. 8 show the flow structures in the wake behind the sphere for increasing Re. At $Re = 100$, an attached stable ring eddy is formed behind the surface of the sphere (Fig. 8a). At Re in the range of 1000, the ring eddy becomes unstable and starts to shed. Further increase in Re results in a fully turbulent wake (Fig. 8b and Fig. 8c). The drag crisis is evident by noticing difference in the



(a) $Re = 100$ (b) $Re = 160{,}000$ (c) $Re = 1{,}000{,}000$

Fig. 8. The wake structures and pressure distribution on sphere at different Reynolds number. At low Re, the wake remains axisymmetric. As Re increases, it starts shedding and boundary layer becomes turbulent. The picture gives the phenomenological description of the drag crisis. The drag crisis is evident by noticing the wake structure as it changes from being divergent at $Re = 160,000$ (high drag state) to being convergent at $Re = 10^6$ (low drag state). At the same time, we observe a high pressure region being developed behind the sphere. The development of this high pressure zone is attributed to the low drag state. Below the drag crisis, the flow separates at an angle smaller than 90^0 in the hemisphere facing the flow but above the drag crisis, the separation angle is pushed backward to the hemisphere pointing away from the flow.

wake structure between Fig. 8b and Fig. 8c. In the subcritical regime, the wake behind the sphere tends to diverge as it moves away from the sphere. Above the critical Re, the wake tends to converge and become narrower as it moves behind the sphere. Subsequently, we can observe a high pressure zone being developed behind the sphere. The main reason for the drag crisis can be attributed to the development of this high pressure zone behind the sphere that tends to push the sphere in the inflow direction. These results are in agreement with the simulations by Geier et al. [78] using Lattice Boltzmann method to capture the drag crisis.

Remark. *This shows the ability of our framework to capature the phenomenological description of the drag crisis observed as Re is increased from* $160,000$ *to* 10^6*. A more detailed analysis of drag crisis is required to answer some of the key questions from the flow physics perspective. This includes identification of features that trigger drag crisis, and characterization of transition from high drag (subcritical) to low drag (supercritical) state (into sudden vs continuous transition). These questions are beyond the scope of the current methods paper, but we anticipate addressing these intriguing questions in subsequent work.*

4.2. Matrix Assembly

We report on how tensorization can speed up matrix assembly as described in Section 3.2 using linear and quadratic basis function. In order to compare the performance, we considered a canonical lid driven cavity problem in a unit cubic domain at $Re = 100$ on a uniform mesh of $16 \times 16 \times 16$. Table 1 shows the comparison of the time for matrix assembly and subsequent solve time on TACC Stampede2 SKX and KNL node and Frontera CLX processors. We observe a significant speedup in the matrix assembly process across all these platforms, with more substantial speedups for higher order basis functions.

		Assembly Time			Total Time		
		GP (s)	MM (s)	Speedup	GP (s)	MM (s)	Speedup
Linear	SKX	1.0617	0.7754	1.37	4.2598	3.9201	1.10
	KNL	5.4996	3.775	1.46	7.83192	6.33589	1.24
	CLX	0.80538	0.562	1.43	10.33978	10.17841	1.02
Quadratic	SKX	71.132	10.245	6.94	234.202	175.03	1.34
	KNL	385.59	40.409	9.54	664.05	346.15	1.92
	CLX	59.356	8.4647	7.01	212.77	164.46	1.3

Table 1: Comparison of time on different computing environment for matrix assembly and subsequent solve time for linear and quadratic basis function on Stampede2 SKX and KNL nodes, and Frontera CLX nodes. GP indicates the matrix assembly by looping over the individual Gauss points whereas MM indicates the assembly by imposing FEM operator as matrix–matrix multiplication.

4.3. In - Out *Test*

Here, we show the performance gain by using the normal based In - Out test as compared to the ray-tracing for three different complex geometries: Stanford bunny, Stanford dragon (available at [81]) and truck (our target problem, discussed later in Section 5). These examples represent fairly complex geometries creating a diverse set of Intercepted elements. In order to mimic our simulation scenario, we generated 5^3 Gauss quadrature points per Intercepted elements. Table 2 shows the comparison between the normal based In - Out and ray-tracing. Since both of these algorithms are embarrassingly parallel, the total time reported is the cumulative sum of the time spent by each processor. We make several observations: (a) on average, normal based In - Out test is $1000\times$ faster than ray-tracing; (b) as the mesh resolution improves, the fraction of octants needing ray-tracing decreases. This is because any sharp corners (where our normal In - Out test fails) now reside within fewer elements; and (c) by combining ray-tracing with the normal based In - Out we see an overall speedup by at least $6\times$ (for the finest resolution) without compromising on accuracy.

	Grid Size	Total points	Proposed scheme							Ray–tracing	Speedup
			Normal–based			Ray–tracing			Total time (s)		
			P_s	Time (s)	T_{avg}(s)	P_f	Time (s)	T_{avg}(s)		Total time (s)	
Bunny	64^3	1,193,640	0.79	0.996	1E-6	0.21	697.012	0.003	698.008	3247.6	4.66
	128^3	4,149,957	0.87	2.205	6E-7	0.13	1496.81	0.003	1499.01	11264.5	7.52
	256^3	10,317,097	0.91	4.234	4E-7	0.09	2555.78	0.003	2560.02	28156.6	11
Dragon	64^3	837,263	0.56	0.558	5E-7	0.44	1086.05	0.003	1086.32	2213.46	2.03
	128^3	3,351,110	0.74	0.624	2.53E-7	0.26	2615.81	0.003	2616.43	8885.58	3.4
	256^3	12,120,591	0.85	1.382	1.3E-7	0.15	6096.96	0.003	6098.34	35392.4	5.81
Truck	64^3	383,474	0.64	0.99	4E-6	0.36	621.469	0.004	622.459	1638.21	2.63
	128^3	1,602,839	0.80	1.84	1E-7	0.20	1724.55	0.005	1726.39	7468.15	4.33
	256^3	6,205,213	0.89	3.535	6.3E-7	0.11	3816.79	0.005	3820.33	34675.8	9.08

Table 2: Comparison of the performance of the proposed In - Out test with the conventional ray-tracing for three different complicated geometries. P_s represents the success fraction of normal based test and P_f represents the failure fraction for which we resort to ray-tracing. T_{avg} represents the average time per evaluation of the normal based and ray-tracing.

4.4. Adaptive Quadrature

In order to test the impact of adaptive quadrature on the flow physics, we consider a benchmark problem of flow past a sphere at $Re = 100$. The simulation is performed on a fixed mesh ($R_{bkg} = 5, R_{wake} = 5, R_{bdy} = 6$), but with increasing levels of adaptive quadrature of Intercepted elements. Fig. 9 shows the convergence of drag coefficient and associated percentage error to its reference value with increasing number of quadrature points in the Intercepted elements. This indicates that adaptive quadrature is essential to accurately model Intercepted elements. Our numerical experiments suggest that two levels of quadrature splitting (percentage error ~ 1%) produces converged results, and further increase in quadrature level does not give any significant improvement in quantitative prediction of aerodynamic coefficient, which was also reported by Xu et al. [19] on unstructured meshes. With increase in the quadrature levels, the associated cost with increased Gauss points in Intercepted elements leads to a load imbalance, significantly increasing the time-to-solve. This suggests the need for weighted partitioning.

4.5. Weighted Partitioning

Here, we demonstrate the advantage of using weighted partitioning using flow past a sphere problem (at $Re = 100$) as a benchmark. Eq. (6) provides a theoretical estimate of the weights. This requires identification of the ratio, $\frac{T_V}{T_S}$, which can in principle be evaluated in an architecture specific way. Alternatively, simple Monte-Carlo sampling of volumetric and surface quadrature points provides a good estimate of this fraction. With this fraction, we accurately identify the weight associated with each octant using Eq. (6). Table 3 shows the estimated cost by running the simulation on Stampede2 SKX and KNL nodes. The experiments reveal the relative cost for matrix assembly to be ≈ 3.3.

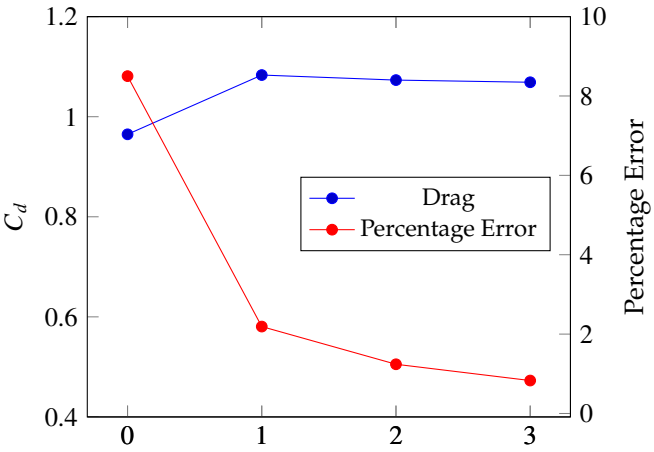


Fig. 9. Drag and percentage error for different levels of adaptive quadrature. We see that with increase in the number of quadrature point for Intercepted nodes, C_d converges to the experimental observed value of 1.06-1.096 at $Re = 100$ [19]. Reference value of 1.06 [82] was chosen to compute percent error.

	Matrix Assembly		
	Tv	Ts	Ratio
SKX	2.72E-5±7.5E-6	8.4E-6 ±1.85E-6	3.23
KNL	1.6E-4 ± 2E-5	4.83E-5 ± 1.51E-5	3.32

Table 3: The ratio $\frac{T_v}{T_s}$ estimated by running the simulation on Stampede2 SKX and KNL nodes.

Fig. 10 compares the predicted weight (using Eq. (6)) with the experimental weight, averaged over 10 runs, for two different meshes and surface discretization. We can see that the predicted weight is in good agreement with the actual weight. The estimated ratio of $\frac{T_v}{T_s}$, on a specific architecture only depends on the type of PDE being solved (which governs the number of FLOPS and data movement across memory hierarchy required per

volume and surface quadrature points) and is independent of the number of mesh elements, the nature of surface discretization, as well as the parameters (like Reynolds numbers). Further, we utilize this model to show the impact of correct load balancing on the actual solve time.

	Equal partition	Weighted Partition	Speedup
Matrix Assembly	72.672 s	28.01 s	2.60
Solve Time	127.4 s	47.67 s	2.67

Table 4: Comparison of matrix assembly and solve time for equal and weighted partitioning

Fig. 11 shows the element and weight distribution across different processors. In case of equal partition, the number of elements are equally distributed across different processors. This results in load imbalance which is circumvented by making use of weighted partitioning. The weighted partitioning ensures that the elements are distributed such that each processor receives equal amount of work. Table 4 shows the overall speedup achieved in the assembly time and total solve time. We observe that the correct distribution of work can help to achieve a substantial speedup. Eq. (6) generalizes to moving meshes, where after each remeshing step, re-partitioning is performed. The enumeration of the viable Gauss points and surface elements in each INTERCEPTED element is performed in linear time, involving a single pass over the elements.

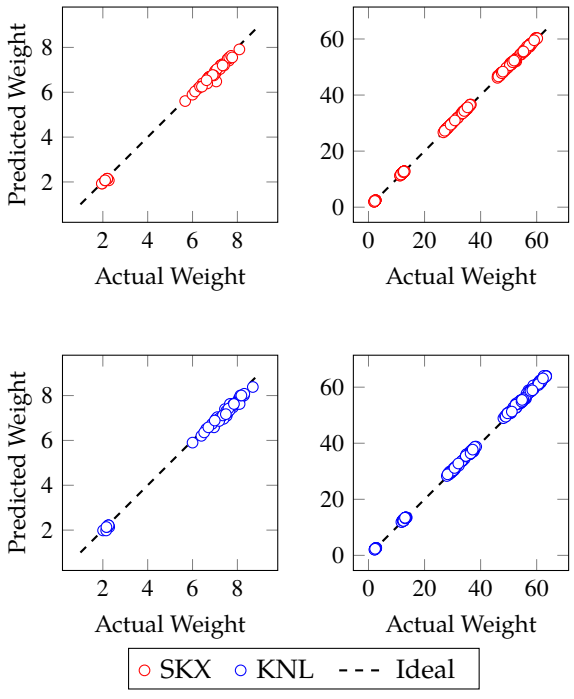


Fig. 10. Comparison of predicted weight and actual weight for INTERCEPTED elements for two different spatial and surface mesh on Stampede2 SKX and KNL nodes. The left panel corresponds to the 16^3 uniform grid with surface discretization comprising of 3000 triangles and right panel correspond to the 32^3 uniform grid and surface discretization with 82000 triangles.

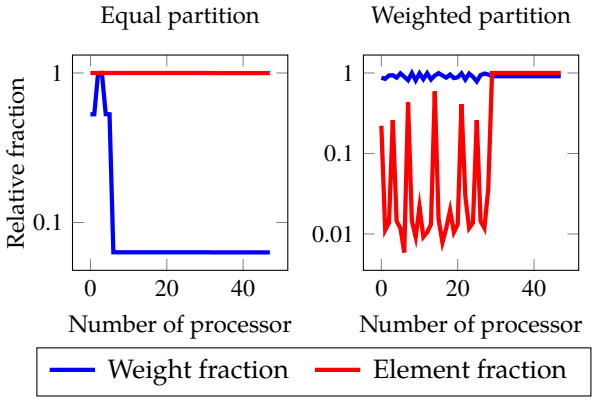


Fig. 11. Impact of weighted partition: Figure compares how elements and weights are distributed for the standard (left) and weighted (right) partitioning scheme. In equal partition, each processor receives the equal number of element whereas in case of weighted partitioning, the elements are partitioned in such a way that each processor receives nearly identical weight. The fraction are computed by normalizing by the maximum number of elements and weight across all processors.

4.6. *Scaling Studies*

Finally, in this section, we report on the scaling behaviour of our solver by simulating the flow across a sphere case discussed in detail for validating the numerical method in Section 4.1.

4.6.1. ***Strong Scaling***

For strong scaling, we consider four different adaptive meshes: M1 consisting of 76,868 elements, M2 consisting of 519,800 elements, M3 consisting of around 4 million elements and M4 consisting of around 15 million elements. The maximum refinement region in all the meshes was near the sphere region. Table 5 shows the refinement level in different regions for the scaling studies for different mesh. The sphere surface discretization for carrying out the integration of weak boundary conditions was kept constant and comprises of around 0.3 million triangle elements. Surface discretization of the sphere was chosen to ensure that we have at least 3-4 Gauss quadrature points per INTERCEPTED elements at the finest octree mesh resolution (Mesh M4). We used linear basis function for spatial discretization.

Fig. 12 shows the strong scaling result for the different meshes on TACC's Frontera.[4] We use a

[4]We have shown the scaling to the maximum allocation that we had on Frontera supercomputer.

	R_{bkg}	R_{wake}	R_{bdy}
M1	5	7	8
M2	6	8	9
M3	7	9	10
M4	8	10	11

Table 5: The level of refinement in the various region of the domain for scaling studies.

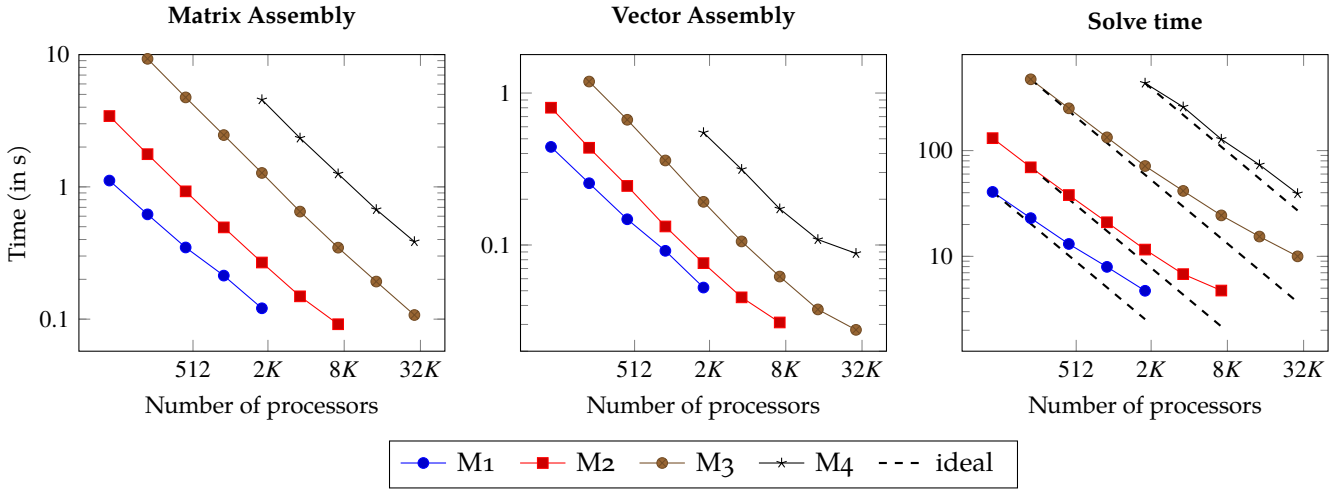


Fig. 12. Strong scaling result on TACC Frontera on 4 different meshes. The plot shows good scaling until the grain size per processor value of ≈ 320 (5000 dof).

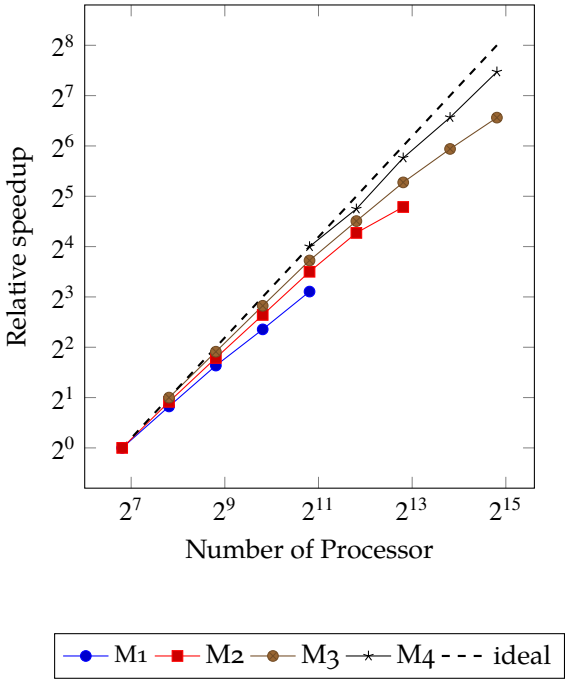


Fig. 13. Relative speedup for solve time as a function of number of processor for different problem size.

bi-conjugate gradient solver with additive Schwartz preconditioner. The simulation was carried out for 10 time steps with Δt of 0.01. In total there were 21 rounds of non-linear solve, which resulted in 21 rounds of matrix assembly and 31 rounds of vector assembly. The average per iteration matrix and vector assembly time is reported. The same results are also presented in Fig. 13 as the relative speedup for different problem sizes. The results reveal near ideal scaling till the grain size (the number of octants per processor) is around 320. Since we are solving for 4 degrees of freedom per node, this translates to roughly 5000 degrees of freedom per processor. At smaller grain sizes, the amount of data being exchanged with neighboring processor increases compared to the amount of work being done. This makes it difficult to effectively overlap communication and computation, leading to a loss in scalability. We also observed that the number of linear solve iterations increases with the number of processors, primarily due to block preconditioning, which additionally contributes to the deviation from the ideal scaling. Finally, at small grain sizes, load balancing for the intercepted elements becomes challenging, and contributes to the loss of scalability. Note that for most practical problems, we will be operating with much larger grain sizes avoiding these challenges. These extreme strong scaling results are presented to motivate additional research into efficient preconditioners and load-balancing techniques for IMGA.

4.6.2. *Weak Scaling*

For weak scaling, we considered 3 different adaptive meshes with number of elements per processor varying from ~ 500 to ~ 2000. Similar to the strong scaling case, the discretization for the object is kept constant. Fig. 14 plots the results of our weak scaling experiments. We compare the weak scaling result for the matrix assembly, vector assembly, the time taken for each KSP iteration[5] iteration and total solve time. We achieve excellent weak scaling performance, with similar trends as the strong scaling. Overall, scaling is better at larger grain sizes, especially for matrix and vector assembly. We achieve a weak scaling efficiency ~ 0.5 for total solve time while increasing the number of processor and problem size by a factor of 16. The overall time to solve suffers because of degradation of the preconditioner and poor conditioning of the matrix, which results in increase in number of KSP iteration with problem size.

[5] KSP is the Krylov subspace context used in PETSc for solving linear systems.

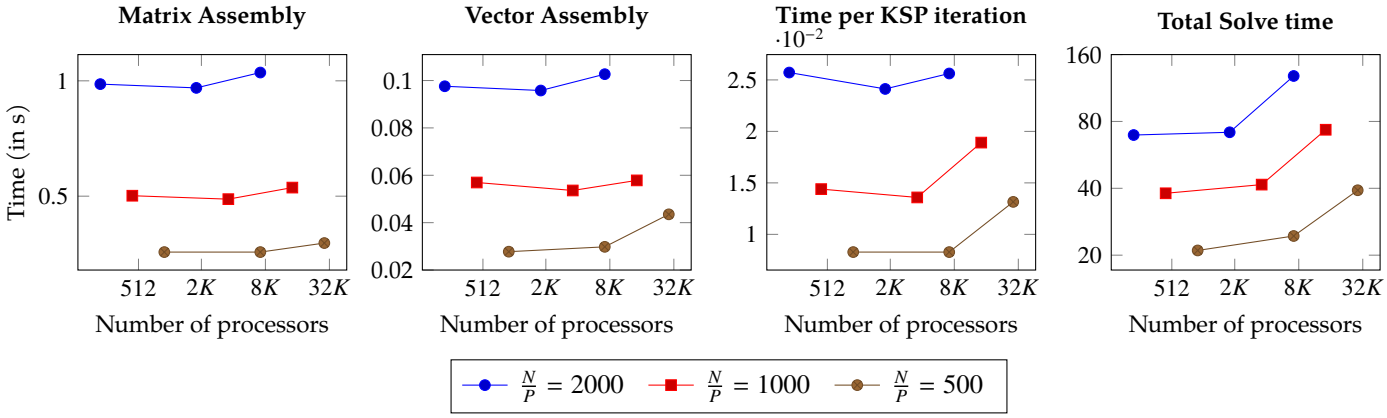


Fig. 14. Weak scaling result on TACC Frontera. We considered 3 different adaptive meshes with the number of elements per processor varying from 500 to 2000.

Remark. *We hypothesize that a multilevel preconditioner can improve the scalability (both strong and weak) as the number of iteration count remains approximately constant with increase in the number of processor and the problem size. However, in our experiments using Algebraic Multigrid (AMG) via PETSc interface, while we observed a constant number of iteration with increase in the number of processor as well as mesh independence (number of iterations remaining approximately constant with increase problem size), the setup costs for AMG are high and exhibit poor scalability for large number of processor counts (especially > 128 Frontera nodes ~ 7000 processors) (Section Appendix C.1). Currently, for our target problems, we achieved a better time to solve using a single level Additive Schwarz preconditioner. The use of Geometric multigrid (GMG) should address the high setup costs associated with AMG, as well as its scalability. This is left as future work.*

5. Flow past a complex geometry: Semi-trailer truck

In this section, we illustrate the utility of our framework on a practical application problem. We explore the flow physics across a realistic semi-truck geometry travelling at 65 MPH (a Reynolds number of 30×10^6). Then, we quantify the advantage of platooning multiple semi-trucks which is one of the compelling fuel saving features of next-generation autonomous vehicles. Compared with modern motor vehicles, commercial vehicles are aerodynamically inefficient due to their bulky design. Any reduction of drag can lower the cost of fuel consumption and thus possess potential economic and environment benefit [83]. We demonstrate the capability of our framework to investigate the effect of individual parts of a semi-trailer truck on the drag as well as the platooning effect of two trucks.

5.1. Geometry of the Truck

The geometry of semi-trailer consists of several parts such as tractor, trailer, tanks, tires and axis. Each part is verified to be a water-tight triangulated manifold. The non-dimensional length of the truck is 1 (normalized by the truck length, which is 15 m). The Reynolds number of the problem, estimated based on the truck length and cruising at 65 mph, is around 30×10^6. We solve in a moving reference frame, where the truck is stationary and air is moving past at 65 mph.

5.2. Computational Domain and Boundary Conditions

The computational domain has a dimension of $16 \times 2 \times 2$, with inlet velocity set at 1 and the outlet pressure set at 0. The surrounding walls are no-slip walls moving at the same speed as the incoming flow. The truck is positioned 5 unit lengths behind the inlet. No-slip boundary condition is weakly imposed on the surface of the truck. We model the rotating wheels by enforcing a no-slip condition corresponding to the wheels rotating with an angular speed of $1/r$, producing a linear speed of 1.0. Additional details are presented in Appendix C.2.

5.3. Mesh Generation

The mesh is refined adaptively using multiple refine regions around the truck and in the wake region. The base level of refinement is set to 8 (i.e. 2^8 divisions along a dimension) and the smallest element around the truck has refinement level of 12 (2^{12} divisions along a dimension). This level of refinement is chosen according to the Taylor length scales at this Reynolds number, $Re = 30 \times 10^6$. This resulted in around 3.1 million elements. A time step (Δt) of 0.00125 is chosen for the simulation, resulting in CFL number ranging from 0.02 to 0.32 from the largest element to the smallest element. The simulation is started with a lower Re, larger Δt and a less refined mesh. As the solution converges, we ramp up the Re, decrease Δt and further refine mesh near the region of interest. This is done to remove the initial transient quickly. The removal of transient is a major bottleneck in high fidelity CFD simulations; with transient removal taking several days of simulation time before any statistically reliable data can be collected. The

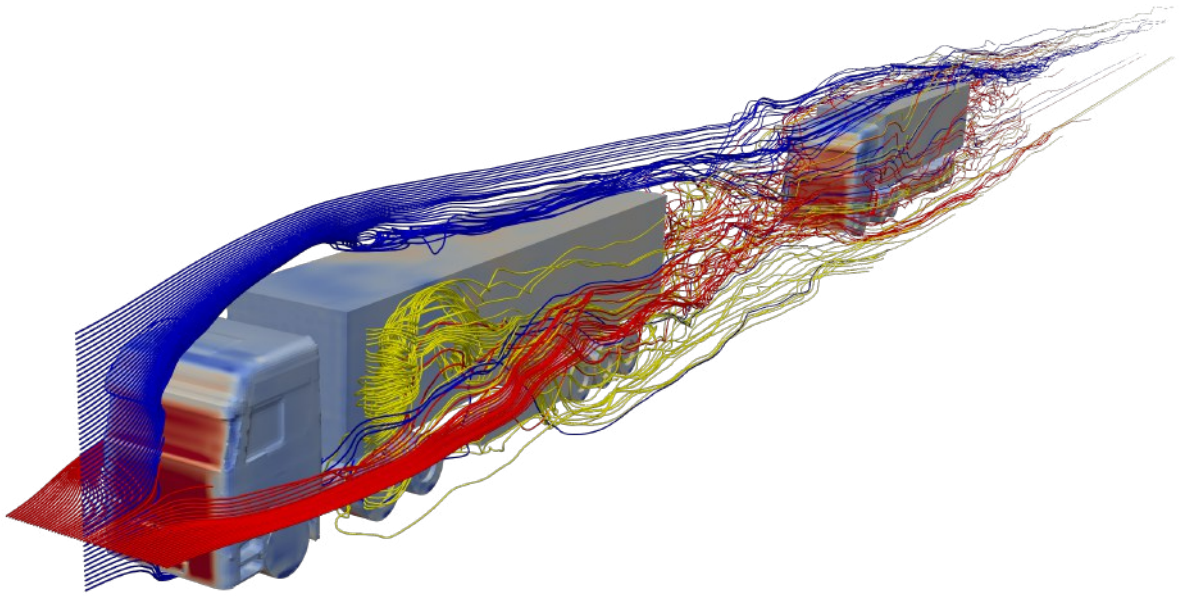


Fig. 15. Flow streamlines demonstrating the platooning effect with two trucks travelling at 65 MPH, corresponding to a Reynolds number, Re of 30×10^6.

octree based framework provides a principled approach to remove these transients by performing simulation starting from a relatively coarse mesh, and successively refining the mesh. Here, to remove transients, we used Stampede2 SKX and KNL processors with number of processors ranging from 192 to 2176. Once the initial transient was removed, the simulation was carried out on 64 Frontera nodes with 3584 processors. In approximately 4 hours, we were able to collect the statistics for about 3.5 seconds.

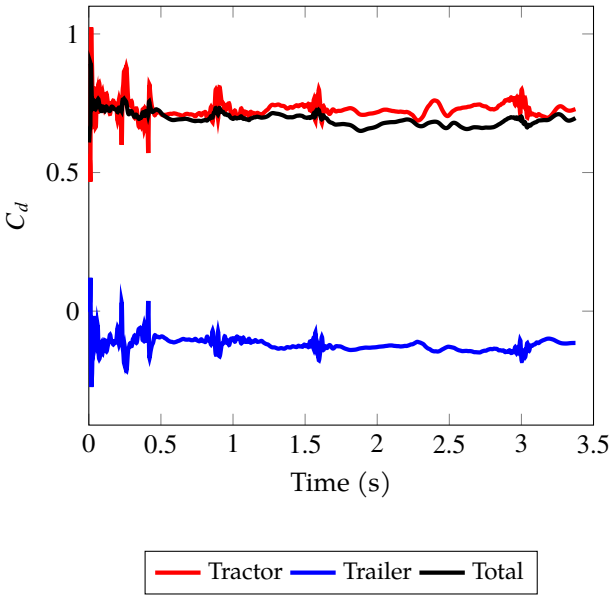


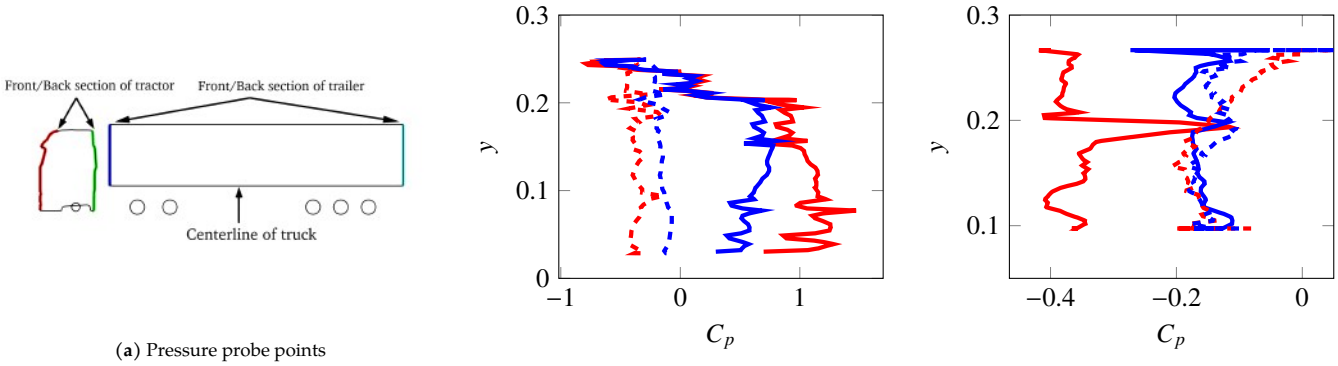
Fig. 16. Time evolution of drag on tractor, trailer and total drag on the semi - truck.

5.4. *Flow Quantities of Interest*

Fig. 16 shows the (coefficient of) drag as a function of time. Note that we are plotting for time after removal of the initial transients, when statistically consistent results are produced. The truck head contributes the most to the drag force, whereas the trailer actually contributes a small negative quantity. This is mainly due to the pressure difference at the front and back of the trailer surface. The average non-dimensional drag coefficient C_d comes around to be 0.695. The reference area A is chosen to be the projected frontal area of the truck. This result is comparable to the previous conducted experimental studies [84, 85, 86] which had reported C_d in the range of 0.6–0.9 for heavy vehicles. The previous numerical result reported the drag coefficient to be about 0.57 for RANS (Reynolds Averaged Navier–Stokes) simulation and 0.62 for DES (Detached Eddy Simulation) simulations for the truck travelling at the speed of 55 MPH. [87]. It must be noted that numerical results based on RANS and DES are susceptible to the proper choice of parameters for wall treatment; while our framework does not rely on any additional treatments. This makes our approach significantly more robust than existing state-of-art approaches.

5.5. *Flow Past Multiple Complex Objects: Platooning of Semi-trailer Trucks*

Studies have shown that by platooning large, blunt vehicles, the overall fuel saving can be significant. This is especially significant for autonomous vehicles, where a platoon of trucks can operate safely and efficiently (with nearly 25% improved efficiency [88, 87]). We explore this concept using the detailed LES simulations afforded by



(a) Pressure probe points

(b) Tractor

(c) Trailer

Fig. 17. *Platooning effect:* Time-averaged non-dimensional pressure coefficient (C_p) at the center-line of the truck; Front of vehicles 1 and 2 are represented by thick red and blue line and back is represented by dashed red and blue lines. Fig. 17a shows the pressure probe points.

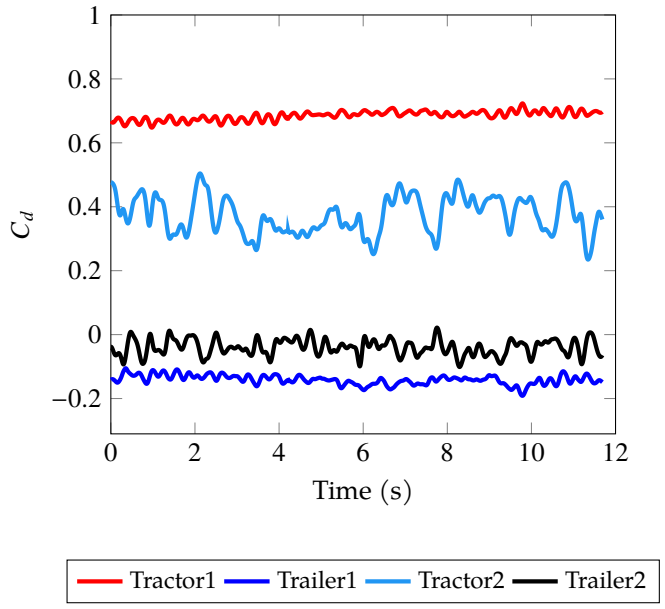


Fig. 18. Time evolution of drag on tractor and trailer demonstrating the platooning effect. The drag on second truck is significantly lower than the first one.

this framework. We placed two identical semi-trailer trucks one truck-length D apart, with the computation domain and boundary conditions identical to previous simulation. We can see that the The full scale two-truck simulation resulted in a mesh with about 6 million elements. The complete simulation was carried out on 128 Frontera CLX nodes (7168 processor) for about 16 hours [6] to collect the data over 12 seconds, after removal of transients. Leveraging the ability to refine and coarsen the grid, we used a checkpoint solution from the one-truck simulation as the initial guess, which resulted in substantial reduction in overhead for transient removal during the two truck simulations. Fig. 18 shows the C_d history for major components of two trucks. The averaged C_d for the second truck is 0.475, showing a 32% decrease in total drag. This reduction in drag is comparable with the ones reported in literature where a reduction of 30% is observed using DES simulation [87], as well as a full scale experimental study [89]. Fig. 15 shows the streamlines for the flow over two trucks. We can see the streamlines of the first truck is effecting the flow around the second truck. It is clear from Fig. 17b that the main source of drag reduction is due to the lowered drag of the tractor head. The figure plots the non–dimensional pressure coefficient C_p at the centerline of the two tractors. The front section of the trailing truck has a lower pressure than the leading truck, with larger difference around the lower half of the truck. On the surface of the trailer, shown in Fig. 17c, the platooning truck shows similar pressure magnitude with both the front and back surface while the leading truck clearly have higher pressure at the front surface of the trailer. Additional analysis of the flow physics is reported in Appendix C.2

[6] This time includes the time taken to remove the transients for two truck simulation.

6. Conclusion

We present a highly scalable, adaptive IMGA framework to solve industrial scale LES problem. We highlighted some of the key algorithmic challenges and improvements over the existing state-of-the-art IMGA methods. We have demonstrated excellent scaling results for our framework on current supercomputers, and shown preliminary application to practical problems. We believe that this will serve as a step towards achieving the goal of conducting overnight large scale LES simulations. Some future direction that can be explored in the context of IMGA are:

- Carefully designed PDE and method specific preconditioners [90, 91, 92] that can exploit the matrix-free method and accelerate the convergence.
- Extension to higher-order finite element spaces. Higher order methods are more difficult to converge due to poor conditioning of matrices [93], and therefore careful design of the preconditioners is of utmost importance to achieve faster solve time.

- Designing fast(er) algorithms for assembly of FEM kernels (both matrix and matrix-free) that can achieve the lower bound on complexity.
- Efficient integration schemes for the INTERCEPTED elements.
- Scalable Multigrid methods [94, 95, 96] in the context of IMGA.

Acknowledgment

KS, Boshun G, BK, MAK, Baskar G were partly supported by NSF 1855902 and NSF 1935255. AK was partly supported by NSF 1644441 and NSF 1750865. HS was partly funded by NSF OAC-1808652. Computing allocation through Directors discretionary award (Frontera), XSEDE CTS110007 (Stampede2) and Iowa State University (Nova) are gratefully acknowledged

References

[1] R. Lohner, C. Othmer, M. Mrosek, A. Figueroa, A. Degro, Overnight industrial les for external aerodynamics, in: AIAA Scitech 2020 Forum, 2020, p. 2031.
[2] C. S. Peskin, Flow patterns around heart valves: a numerical method, Journal of computational physics 10 (1972) 252–271.
[3] R. Mittal, H. Dong, M. Bozkurttas, F. Najjar, A. Vargas, A. Von Loebbecke, A versatile sharp interface immersed boundary method for incompressible flows with complex boundaries, Journal of computational physics 227 (2008) 4825–4852.
[4] E. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, Journal of computational physics 161 (2000) 35–60.
[5] Y.-H. Tseng, J. H. Ferziger, A ghost-cell immersed boundary method for flow in complex geometry, Journal of computational physics 192 (2003) 593–623.
[6] J. Kim, D. Kim, H. Choi, An immersed-boundary finite-volume method for simulations of flow in complex geometries, Journal of computational physics 171 (2001) 132–150.
[7] K. Taira, T. Colonius, The immersed boundary method: a projection approach, Journal of Computational Physics 225 (2007) 2118–2137.
[8] A. Pinelli, I. Naqavi, U. Piomelli, J. Favier, Immersed-boundary methods for general finite-difference and finite-volume navier–stokes solvers, Journal of Computational Physics 229 (2010) 9073–9091.
[9] X. Yang, X. Zhang, Z. Li, G.-W. He, A smoothing technique for discrete delta functions with application to immersed boundary method in moving boundary simulations, Journal of Computational Physics 228 (2009) 7821–7836.
[10] R. Mittal, G. Iaccarino, Immersed boundary methods, Annu. Rev. Fluid Mech. 37 (2005) 239–261.
[11] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (2005) 4135–4195.
[12] J. A. Nitsche, Über ein Variationsprinzip zur Lösung von Dirichlet-Problemen bei Verwendung von Teilräumen, die keinen Randbedingungen unterworfen sind, Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg 36 (1970) 9–15.
[13] A. Embar, J. Dolbow, I. Harari, Imposing dirichlet boundary conditions with nitsche's method and spline-based finite elements, International journal for numerical methods in engineering 83 (2010) 877–898.
[14] M.-C. Hsu, D. Kamensky, Y. Bazilevs, M. S. Sacks, T. J. R. Hughes, Fluid–structure interaction analysis of bioprosthetic heart valves: significance of arterial wall deformation, Computational Mechanics 54 (2014) 1055–1071.
[15] D. Kamensky, M.-C. Hsu, D. Schillinger, J. A. Evans, A. Aggarwal, Y. Bazilevs, M. S. Sacks, T. J. R. Hughes, An immersogeometric variational framework for fluid–structure interaction: Application to bioprosthetic heart valves, Computer Methods in Applied Mechanics and Engineering 284 (2015) 1005–1053.
[16] F. Xu, S. Morganti, R. Zakerzadeh, D. Kamensky, F. Auricchio, A. Reali, T. J. R. Hughes, M. S. Sacks, M.-C. Hsu, A framework for designing patient-specific bioprosthetic heart valves using immersogeometric fluid–structure interaction analysis, International Journal for Numerical Methods in Biomedical Engineering 34 (2018) e2938.
[17] M. C. H. Wu, H. M. Muchowski, E. L. Johnson, M. R. Rajanna, M.-C. Hsu, Immersogeometric fluid–structure interaction modeling and simulation of transcatheter aortic valve replacement, Computer Methods in Applied Mechanics and Engineering 357 (2019) 112556.
[18] M. C. Wu, D. Kamensky, C. Wang, A. J. Herrema, F. Xu, M. S. Pigazzini, A. Verma, A. L. Marsden, Y. Bazilevs, M.-C. Hsu, Optimizing fluid–structure interaction systems with immersogeometric analysis and surrogate modeling: Application to a hydraulic arresting gear, Computer Methods in Applied Mechanics and Engineering 316 (2017) 668–693.
[19] F. Xu, D. Schillinger, D. Kamensky, V. Varduhn, C. Wang, M.-C. Hsu, The tetrahedral finite cell method for fluids: Immersogeometric analysis of turbulent flow around complex geometries, Computers & Fluids 141 (2016) 135–154.
[20] M.-C. Hsu, C. Wang, F. Xu, A. J. Herrema, A. Krishnamurthy, Direct immersogeometric fluid flow analysis using B-rep CAD models, Computer Aided Geometric Design 43 (2016) 143–158.
[21] C. Wang, F. Xu, M.-C. Hsu, A. Krishnamurthy, Rapid b-rep model preprocessing for immersogeometric analysis using analytic surfaces, Computer aided geometric design 52 (2017) 190–204.
[22] F. Xu, Y. Bazilevs, M.-C. Hsu, Immersogeometric analysis of compressible flows with application to aerodynamic simulation of rotorcraft, Mathematical Models and Methods in Applied Sciences 29 (2019) 905–938.
[23] S. Xu, F. Xu, A. Kommajosula, M.-C. Hsu, B. Ganapathysubramanian, Immersogeometric analysis of moving objects in incompressible flows, Computers & Fluids 189 (2019) 24–33.
[24] D. Kamensky, Open-source immersogeometric analysis of fluid–structure interaction using FEniCS and tIGAr, Computers and Mathematics with Applications (2020). URL: https://doi.org/10.1016/j.camwa.2020.01.023.
[25] Y. Bazilevs, T. J. R. Hughes, Weak imposition of Dirichlet boundary conditions in fluid mechanics, Computers & Fluids 36 (2007) 12–26.
[26] Q. Zhu, F. Xu, S. Xu, M.-C. Hsu, J. Yan, An immersogeometric formulation for free-surface flows with application to marine engineering problems, Computer Methods in Applied Mechanics and Engineering 361 (2020) 112748.
[27] S. Balay, S. Abhyankar, M. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, A. Dener, V. Eijkhout, W. Gropp, et al., Petsc users manual, 2019.
[28] M. A. Heroux, J. M. Willenbring, Trilinos users guide, 2003.
[29] S. Tomov, J. Dongarra, V. Volkov, J. Demmel, Magma library, Univ. of Tennessee and Univ. of California, Knoxville, TN, and Berkeley, CA (2009).
[30] W. Deconinck, P. Bauer, M. Diamantakis, M. Hamrud, C. Kühnlein, P. Maciel, G. Mengaldo, T. Quintino, B. Raoult, P. K. Smolarkiewicz, et al., Atlas: A library for numerical weather prediction and climate modelling, 2017.
[31] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp,

E. Lurie, D. Mavriplis, Cfd vision 2030 study: a path to revolutionary computational aerosciences, 2014.
[32] T. J. R. Hughes, L. Mazzei, K. E. Jansen, Large eddy simulation and the variational multiscale method, Computing and Visualization in Science 3 (2000) 47–59.
[33] Y. Bazilevs, V. M. Calo, J. A. Cottrel, T. J. R. Hughes, A. Reali, G. Scovazzi, Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows, Computer Methods in Applied Mechanics and Engineering 197 (2007) 173–201.
[34] A. Düster, J. Parvizian, Z. Yang, E. Rank, The finite cell method for three-dimensional problems of solid mechanics, Computer Methods in Applied Mechanics and Engineering 197 (2008) 3768–3782.
[35] V. Gravemeier, S. Lenz, W. A. Wall, Variational multiscale methods for incompressible flows, International Journal of Computing Science and Mathematics 1 (2007) 444.
[36] Y. Bazilevs, I. Akkerman, Large eddy simulation of turbulent taylor–couette flow using isogeometric analysis and the residual-based variational multiscale method, Journal of Computational Physics 229 (2010) 3402–3414.
[37] B. Koobus, C. Farhat, A variational multiscale method for the large eddy simulation of compressible turbulent flows on unstructured meshes—-application to vortex shedding, Computer Methods in Applied Mechanics and Engineering 193 (2004) 1367–1383.
[38] J. Loewe, G. Lube, A projection-based variational multiscale method for large-eddy simulation with application to non-isothermal free convection problems, Mathematical Models and Methods in Applied Sciences 22 (2012) 1150011.
[39] A. N. Brooks, T. J. R. Hughes, Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier–Stokes equations, Computer Methods in Applied Mechanics and Engineering 32 (1982) 199–259.
[40] T. E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Advances in Applied Mechanics 28 (1992) 1–44.
[41] T. E. Tezduyar, Y. Osawa, Finite element stabilization parameters computed from element matrices and vectors, Computer Methods in Applied Mechanics and Engineering 190 (2000) 411–430.
[42] T. J. R. Hughes, L. Mazzei, A. A. Oberai, A. Wray, The multiscale formulation of large eddy simulation: Decay of homogeneous isotropic turbulence, Physics of Fluids 13 (2001) 505–512.
[43] M.-C. Hsu, Y. Bazilevs, V. M. Calo, T. E. Tezduyar, T. J. R. Hughes, Improving stability of stabilized and multiscale formulations in flow simulations at small time steps, Computer Methods in Applied Mechanics and Engineering 199 (2010) 828–840.
[44] Y. Bazilevs, C. Michler, V. M. Calo, T. J. R. Hughes, Weak Dirichlet boundary conditions for wall-bounded turbulent flows, Computer Methods in Applied Mechanics and Engineering 196 (2007) 4853–4862.
[45] Y. Bazilevs, C. Michler, V. Carlo, T. J. R. Hughes, Isogeometric variational multiscale modeling of wall-bounded turbulent flows with weakly enforced boundary conditions on unstretched meshes, Computer Methods in Applied Mechanics and Engineering 199 (2010) 780–790.
[46] F. de Prenter, C. Lehrenfeld, A. Massing, A note on the stability parameter in nitsche's method for unfitted boundary value problems, Computers & Mathematics with Applications 75 (2018) 4322–4336.
[47] H. Sundar, R. Sampath, G. Biros, Bottom-up construction and 2:1 balance refinement of linear octrees in parallel, SIAM Journal on Scientific Computing 30 (2008) 2675–2708. doi:10.1137/070681727.
[48] C. Burstedde, L. C. Wilcox, O. Ghattas, p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees, SIAM Journal on Scientific Computing 33 (2011) 1103–1133. doi:10.1137/100791634.
[49] M. Fernando, D. Neilsen, H. Lim, E. Hirschmann, H. Sundar, Massively parallel simulations of binary black hole intermediate-mass-ratio inspirals, SIAM Journal on Scientific Computing 41 (2019) C97–C138. URL: https://doi.org/10.1137/18M1196972. doi:10.1137/18M1196972. arXiv:https://doi.org/10.1137/18M1196972.
[50] M. Fernando, D. Duplyakin, H. Sundar, Machine and application aware partitioning for adaptive mesh refinement applications, in: Proceedings of the 26th International Symposium on High-Performance Parallel and Distributed Computing, HPDC '17, ACM, New York, NY, USA, 2017, pp. 231–242. URL: http://doi.acm.org/10.1145/3078597.3078610. doi:10.1145/3078597.3078610.
[51] M. A. Khanwale, A. D. Lofquist, H. Sundar, J. A. Rossmanith, B. Ganapathysubramanian, Simulating two-phase flows with thermodynamically consistent energy stable cahn-hilliard navier-stokes equations on parallel adaptive octree based meshes, Journal of Computational Physics (2020) 109674.
[52] S. Xu, B. Gao, A. Lofquist, M. Fernando, M.-C. Hsu, H. Sundar, B. Ganapathysubramanian, An octree-based immersogeometric approach for modeling inertial migration of particles in channels, Computers & Fluids 214 (2021) 104764.
[53] H. Sundar, R. S. Sampath, G. Biros, Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel, SIAM J. Sci. Comput. 30 (2008) 2675–2708. doi:10.1137/070681727.
[54] M. S. Fernando, H. Sundar, paralab/Dendro-5.01: Local timestepping on octree grids, 2020. URL: https://doi.org/10.5281/zenodo.3879315. doi:10.5281/zenodo.3879315.
[55] P. Kus, P. Solin, D. Andrs, Arbitrary-level hanging nodes for adaptive hp-fem approximations in 3d, Journal of Computational and Applied Mathematics 270 (2014) 121–133.
[56] H. Sundar, R. S. Sampath, G. Biros, Bottom-up construction and 2: 1 balance refinement of linear octrees in parallel, SIAM Journal on Scientific Computing 30 (2008) 2675–2708.
[57] M. Ishii, M. Fernando, K. Saurabh, B. Khara, B. Ganapathysubramanian, H. Sundar, Solving pdes in space-time: 4d tree-based adaptivity, mesh-free and matrix-free approaches, in: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, 2019, pp. 1–61.
[58] M. O. Deville, P. F. Fischer, P. F. Fischer, E. Mund, et al., High-order methods for incompressible fluid flow, volume 9, Cambridge university press, 2002.
[59] G. Alzetta, D. Arndt, W. Bangerth, V. Boddu, B. Brands, D. Davydov, R. Gassmoeller, T. Heister, L. Heltai, K. Kormann, M. Kronbichler, M. Maier, J.-P. Pelteret, B. Turcksin, D. Wells, The deal.II library, version 9.0, Journal of Numerical Mathematics 26 (2018) 173–183. doi:10.1515/jnma-2018-0054.
[60] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, D. Sorensen, LAPACK Users' guide, volume 9, Siam, 1999.
[61] A. Bressan, S. Takacs, Sum factorization techniques in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 352 (2019) 437–460.
[62] J. M. Melenk, K. Gerdes, C. Schwab, Fully discrete hp-finite elements: Fast quadrature, Computer methods in applied mechanics and engineering 190 (2001) 4339–4364.
[63] B. Khalighi, S. Jindal, J. Johnson, K. Chen, G. Iaccarino, Validation of the immersed boundary cfd approach for complex aerodynamic flows, in: The Aerodynamics of Heavy Vehicles II: Trucks, Buses, and Trains, Springer, 2009, pp. 21–38.
[64] I. Borazjani, L. Ge, F. Sotiropoulos, Curvilinear immersed boundary method for simulating fluid structure interaction with complex 3d rigid bodies, Journal of Computational physics 227 (2008) 7587–7620.
[65] M. De Tullio, A. Christallo, E. Balaras, G. Pascazio, G. Iaccarino, M. Napolitano, Recent advances in the immersed boundary method, European Conference on Computational Fluid Dynamics (2006).
[66] G. Iaccarino, R. Verzicco, Immersed boundary technique for turbulent flow simulations, Appl. Mech. Rev. 56 (2003) 331–347.
[67] V. Thiagarajan, V. Shapiro, Adaptively weighted numerical integration over arbitrary domains, Computers & Mathematics with Applications 67 (2014) 1682–1702.

[68] V. Thiagarajan, V. Shapiro, Adaptively weighted numerical integration in the finite cell method, Computer Methods in Applied Mechanics and Engineering 311 (2016) 250–279.
[69] S. Duczek, U. Gabbert, Efficient integration method for fictitious domain approaches, Computational Mechanics 56 (2015) 725–738.
[70] D. Schillinger, Q. Cai, R.-P. Mundani, E. Rank, A review of the finite cell method for nonlinear structural analysis of complex cad and image-based geometric models, in: Advanced Computing, Springer, 2013, pp. 1–23.
[71] P. J. Barendrecht, M. Bartoň, J. Kosinka, Efficient quadrature rules for subdivision surfaces in isogeometric analysis, Computer Methods in Applied Mechanics and Engineering 340 (2018) 1–23.
[72] A. Stavrev, L. H. Nguyen, R. Shen, V. Varduhn, M. Behr, S. Elgeti, D. Schillinger, Geometrically accurate, efficient, and flexible quadrature techniques for the tetrahedral finite cell method, Computer Methods in Applied Mechanics and Engineering 310 (2016) 646–673.
[73] D. Schillinger, M. Ruess, The finite cell method: A review in the context of higher-order structural analysis of cad and image-based geometric models, Archives of Computational Methods in Engineering 22 (2015) 391–455.
[74] L. Kudela, N. Zander, T. Bog, S. Kollmannsberger, E. Rank, Efficient and accurate numerical quadrature for immersed boundary methods, Advanced Modeling and Simulation in Engineering Sciences 2 (2015) 10.
[75] S. C. Divi, C. V. Verhoosel, F. Auricchio, A. Reali, E. H. van Brummelen, Error-estimate-based adaptive integration for immersed isogeometric analysis, Computers & Mathematics with Applications 80 (2020) 2481–2516.
[76] M. Namburi, S. Krithivasan, S. Ansumali, Crystallographic lattice boltzmann method, Scientific reports 6 (2016) 27172.
[77] E. Achenbach, Experiments on the flow past spheres at very high reynolds numbers, Journal of Fluid Mechanics 54 (1972) 565–575.
[78] M. Geier, A. Pasquali, M. Schönherr, Parametrization of the cumulant lattice boltzmann method for fourth order accurate diffusion part ii: Application to flow around a sphere at drag crisis, Journal of Computational Physics 348 (2017) 889–898.
[79] J. Hoffman, Simulation of turbulent flow past bluff bodies on coarse meshes using general galerkin methods: drag crisis and turbulent euler solutions, Computational Mechanics 38 (2006) 390–402.
[80] G. Constantinescu, K. Squires, Numerical investigations of flow over a sphere in the subcritical and supercritical regimes, Physics of fluids 16 (2004) 1449–1466.
[81] M. Levoy, J. Gerth, B. Curless, K. Pull, The stanford 3d scanning repository, URL http://www-graphics. stanford. edu/data/3dscanrep 5 (2005).
[82] S. Marella, S. Krishnan, H. Liu, H. Udaykumar, Sharp interface cartesian grid method i: an easily implemented technique for 3d moving boundary computations, Journal of Computational Physics 210 (2005) 1–31.
[83] K. Takizawa, T. E. Tezduyar, T. Kuraishi, Multiscale space–time methods for thermo-fluid analysis of a ground vehicle and its tires 25 (2015) 2227–2255.
[84] H. Götz, G. Mayr, Commercial vehicles, in: W. Hucho (Ed.), Aerodynamics of Road Vehicles: From fluid mechanics to vehicle engineering, 4th Edition, Society of Automotive Engineers, 1998, pp. 415–488.
[85] H. Chowdhury, H. Moria, A. Ali, I. Khan, F. Alam, S. Watkins, A study on aerodynamic drag of a semi-trailer truck, Procedia Engineering 56 (2013) 201–205.
[86] R. J. Englar, Advanced Aerodynamic Devices to Improve the Performance, Economics, Handling and Safety of Heavy Vehicles, SAE Technical Paper 2001-01-2072, SAE International, 2001.
[87] V. Viswanathan, V. Shankar, S. Sripad, Platooning for Improved Safety and Efficiency of Semi-trucks PISES, Technical Report, Carnegie Mellon University, 2019.
[88] M. Guttenberg, S. Sripad, V. Viswanathan, Evaluating the potential of platooning in lowering the required performance metrics of li-ion batteries to enable practical electric semi-trucks, ACS Energy Letters 2 (2017) 2642–2646.
[89] S. Torabi, M. Wahde, Fuel-efficient driving strategies for heavy-duty vehicles: a platooning approach based on speed profile optimization, Journal of Advanced Transportation 2018 (2018).
[90] M. Esmaily-Moghadam, Y. Bazilevs, A. L. Marsden, A bi-partitioned iterative algorithm for solving linear systems arising from incompressible flow problems, Computer Methods in Applied Mechanics and Engineering 286 (2015) 40–62.
[91] F. de Prenter, C. V. Verhoosel, G. J. van Zwieten, E. H. van Brummelen, Condition number analysis and preconditioning of the finite cell method, Computer Methods in Applied Mechanics and Engineering 316 (2017) 297–327.
[92] J. N. Jomo, F. de Prenter, M. Elhaddad, D. D'Angella, C. V. Verhoosel, S. Kollmannsberger, J. S. Kirschke, V. Nübel, E. van Brummelen, E. Rank, Robust and parallel scalable iterative solutions for large-scale finite cell analyses, Finite Elements in Analysis and Design 163 (2019) 14–30.
[93] K. P. Gahalaut, S. K. Tomar, C. Douglas, et al., Condition number estimates for matrices arising in nurbs based isogeometric discretizations of elliptic partial differential equations, arXiv preprint arXiv:1406.6808 (2014).
[94] S. Saberi, A. Vogel, G. Meschke, Parallel finite cell method with adaptive geometric multigrid, in: European Conference on Parallel Processing, Springer, 2020, pp. 578–593.
[95] J. Jomo, O. Oztoprak, F. de Prenter, N. Zander, S. Kollmannsberger, E. Rank, Hierarchical multigrid approaches for the finite cell method on uniform and multi-level hp-refined grids, arXiv preprint arXiv:2010.00881 (2020).
[96] F. de Prenter, C. V. Verhoosel, E. van Brummelen, J. Evans, C. Messe, J. Benzaken, K. Maute, Multigrid solvers for immersed finite element methods and immersed isogeometric analysis, Computational Mechanics 65 (2020) 807–838.
[97] H. Sundar, G. Biros, C. Burstedde, J. Rudi, O. Ghattas, G. Stadler, Parallel geometric-algebraic multigrid on unstructured forests of octrees, in: SC'12: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, IEEE, 2012, pp. 1–11.

Appendix A. Precomputation of operators for FEM

A careful analysis of the FEM kernels resulting from the weakening of the Navier–Stokes equations reveals that the evaluation of these four basis function values $(\frac{\partial\phi}{\partial x}; \frac{\partial\phi}{\partial y}; \frac{\partial\phi}{\partial z}; \phi)$ at the Gauss points is repeatedly performed. Thus, the values corresponding to them, over a reference element at the Gauss quadrature points can be pre-computed and cached in a matrix form. These matrices can be explicitly represented as:

$$\underline{\underline{[\nabla\phi]}}^k = \left[\frac{\partial\phi}{\partial k}\right]_{ij} \quad \text{for} \quad i \in 1\cdots\text{nbf}; j \in 1\cdots\text{ngp}; k \in (x,y,z)$$
$$\underline{\underline{\phi}} = [\phi]_{ij} \quad \text{for} \quad i \in 1\cdots\text{nbf}; j \in 1\cdots\text{ngp};$$

where nbf donates the number of basis functions, ngp donates the number of Gauss points, $[\frac{\partial\phi}{\partial k}]_{ij}$ denotes the value of the k^{th} direction derivative of the i^{th} basis function at j^{th} Gauss point. Similarly ϕ_{ij} denotes the value of the i^{th} basis function at Gauss point j. Using standard Gauss - Legendre quadrature rule, each of these forms are matrices of size $(\text{nbf}+1)^d \times (\text{nbf}+1)^d$, where d is the spatial dimension $(d = 3)$. Note that the derivative along each direction k is stored separately as a matrix.

Using these precomputed forms, the various contributing expressions can be efficiently computed.

For example, the stiffness matrix, can be computed as:

$$\underline{\underline{\mathbf{K}}} = \text{COMPUTE\_ELE\_MATRIX}(\text{MAT\_OP1} = \underline{\underline{[\nabla\phi]^x}}, \text{MAT\_OP2} = \underline{\underline{[\nabla\phi]^x}}) +$$
$$\text{COMPUTE\_ELE\_MATRIX}(\text{MAT\_OP1} = \underline{\underline{[\nabla\phi]^y}}, \text{MAT\_OP2} = \underline{\underline{[\nabla\phi]^y}}) +$$
$$\text{COMPUTE\_ELE\_MATRIX}(\text{MAT\_OP1} = \underline{\underline{[\nabla\phi]^z}}, \text{MAT\_OP2} = \underline{\underline{[\nabla\phi]^z}})$$

Other operators for a complicated FEM kernel, for instance $(\mathbf{u}\cdot\nabla\phi_i, \mathbf{u}\cdot\nabla\phi_j)$, can be evaluated as a two step process: first interpolating nodal values (here, $\mathbf{u}$) onto Gauss points (Complexity: $O(d(\text{bf}+1)^{d+1})$ [58]) and then performing the element-wise matrix scalar multiplication. (Complexity: $O(\text{bf}+1)^{2d}$). Thus, the overall complexity of matrix assembly remain bounded by the complexity of matrix-multiplication for all FEM kernels.

Appendix B. Details of solver selection

PETSc was used to solve all the linear algebra problems. In particular, bi-conjugate gradient descent (`-ksp_type bcgs`) solver was used in conjunction with Additive - Schwartz (`-pc_type asm`) preconditioner to solve the linear system of equations. The NEWTONLS class by PETSc, that implements a Newton Line Search method, was used for the nonlinear problems. Both the relative residual tolerance and the absolute residual tolerance for linear and non - linear solve are set to 10^{-6} in all numerical results.

The correctness of the code has been validated by solving the Navier–Stokes with a known analytical solution. The validation case for flow past a sphere performed in 4.1 gives us confidence in the correctness of the framework. The timings reported are measured through PETSc `log_view` routine. Below we provide additional simulation details for the semi - truck simulations.

Appendix C. Additional Simulation Details

Appendix C.1. Scaling studies with AMG preconditioners

Fig. C.19 shows the strong scaling result with AMG. We see that with increasing number of processors, the time for initial AMG setup increases. This result is consistent with what is observed previously in the simulations by Sundar et al. [97]. This limits our ability to deploy AMG on a large number of processor. We are currently working on GMG to avoid this setup cost.

Appendix C.2. Truck Simulations

Fig. C.20 shows the computational domain of the truck and Fig. C.21 shows the details of the CAD model of the truck. Fig. C.22 shows the streamlines starting from four different locations passing the truck. The blue streamline, starting from a vertical line in front of the tractor shows that the flow stagnates at 40% height of the

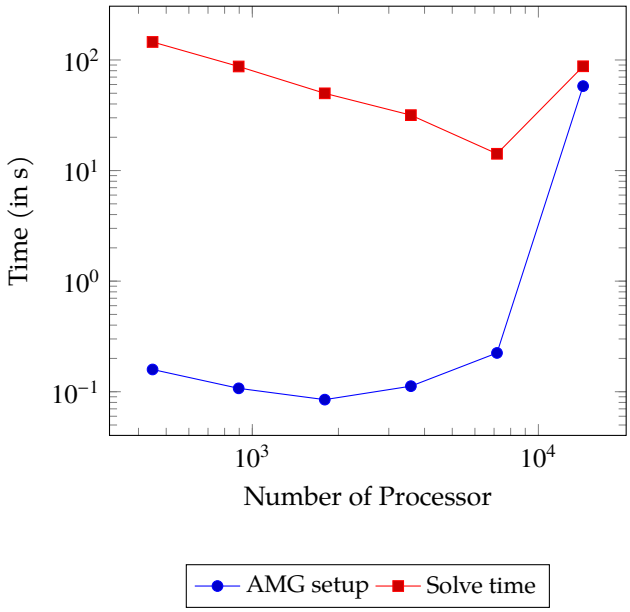


Fig. C.19. The time for AMG setup and complete solve time for mesh M3 with increasing number of processor.

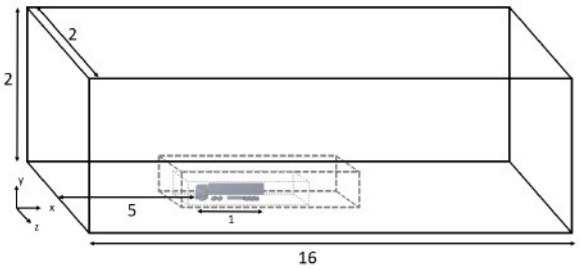


Fig. C.20. Computational domain used for the flow simulation over a semi-trailer truck.

truck. The lower portion gets pushed towards the undercarriage of the vehicle and interacts with the rotating wheels. The green streamlines in Fig. C.22c shows the flow at the top of the tractor does not pass the top surface of the trailer smoothly, some portion of the flow is blocked by the extra height of the trailer and enters the gap between the tractor and the trailer, therefore creating extra turbulence downstream.

Fig. C.24a shows the vertical mid-plane flow structure around the truck. We observed flow re-circulation at the leading edge of the top surface of trailer followed by flow re-attachment downstream. Fig. C.23 shows the affect of rotating wheel in the simulation. In case of stationary wheels, (Fig. C.23a), flow passes through the gap between tires and trailer without obstruction and no re-circulation near the ground between tires is observed, whereas in case of rotating wheel (Fig. C.23b), the clear vortex structures are seen near the tires.

Fig. C.24 shows the slices at different y location at 0.05, 0.1, 0.15 and 0.2, it can be seen that the rotating tires causes flow separation at the leading edge of the side surface of the trailer, which maybe reduced by a side

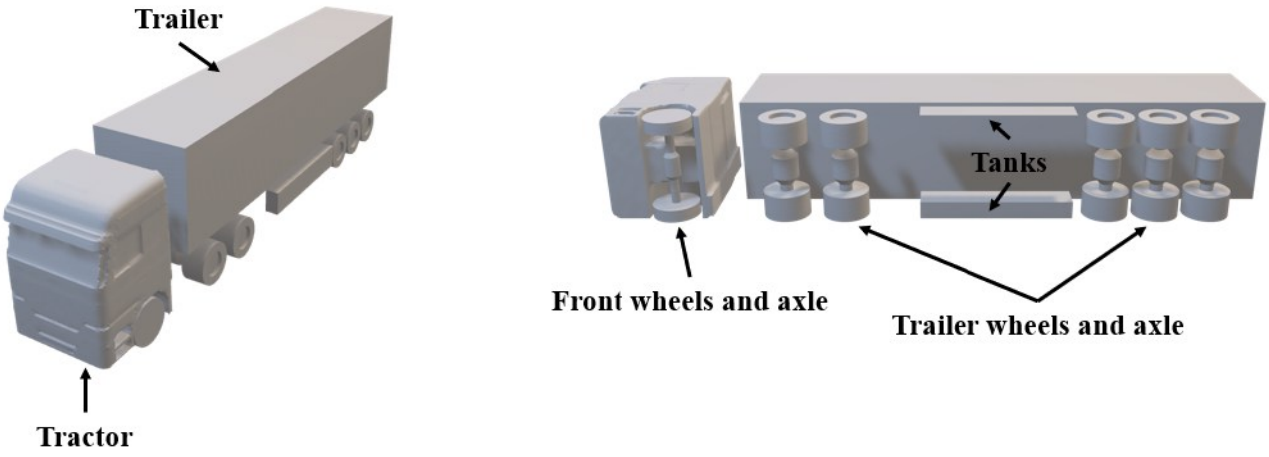
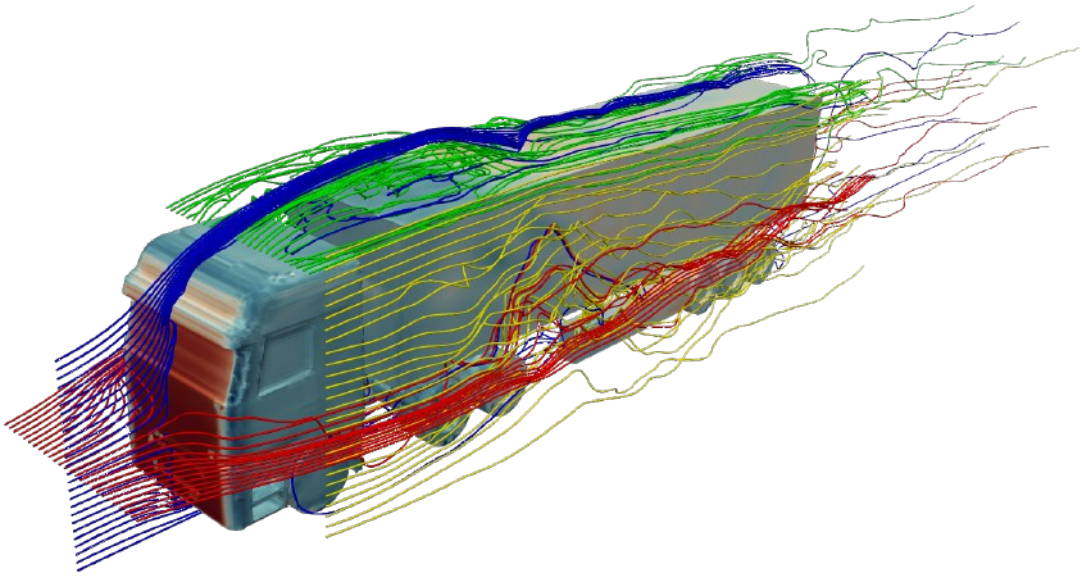
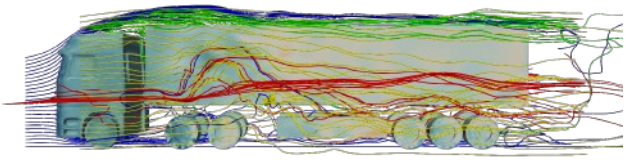


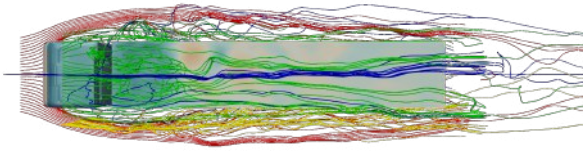
Fig. C.21. Geometry and parts of the simulated semi-trailer truck.



(**a**) Overview of the flow streamlines passing the truck



(**b**) Side view of the flow streamlines passing the truck

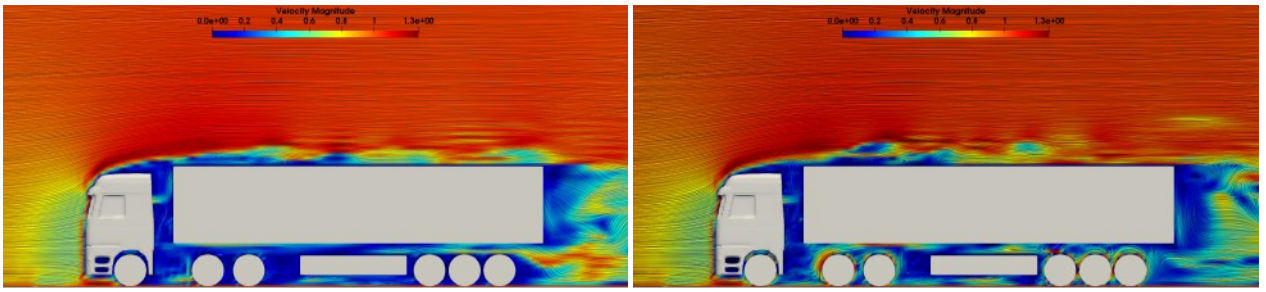


(**c**) Top view of the flow streamlines passing the truck

Fig. C.22. Flow streamlines passing the truck at different locations.

skirt device. Finally, in Fig. C.24b, we show the flow structure at different x slices. This flow pattern resembles the vortices coming off the wing-tip of an airplane, which contributes to additional drag.

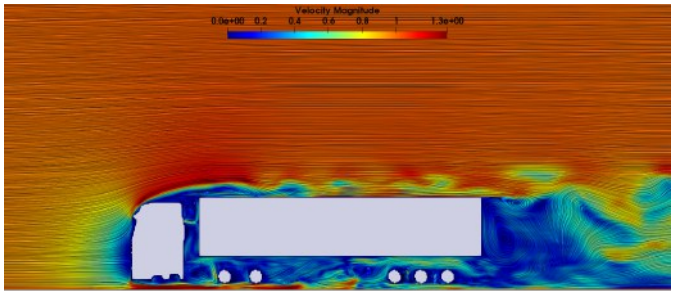
Fig. C.25 shows the comparison in flow structure of the leading truck and the trailing truck in top-down view. The platooning truck is in the turbulent wake of the leading truck. The asymmetric incoming flow



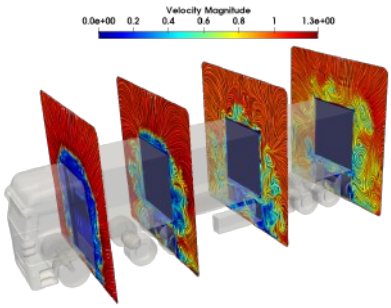
(**a**) Snapshot of flow structures around a stationary wheels.

(**b**) Snapshot of flow structures around a rotating wheels.

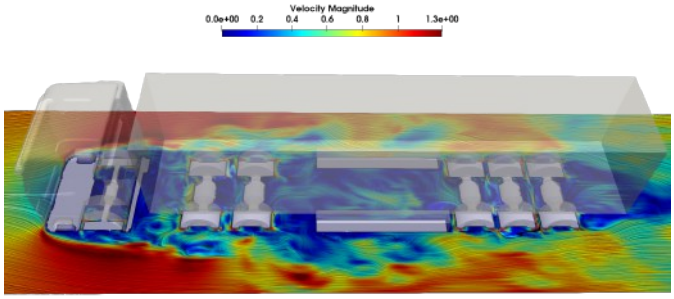
Fig. C.23. Comparison of flow structures of stationary vs. rotating wheels.



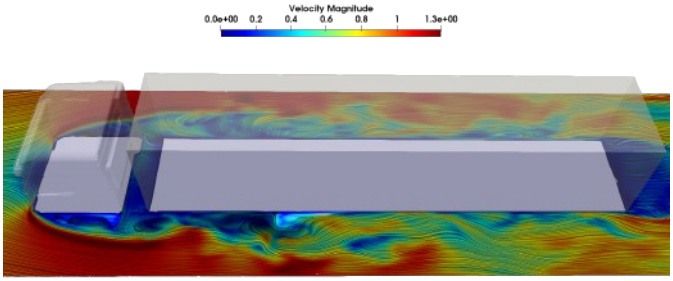
(**a**) Flow structures at vertical middle plane.



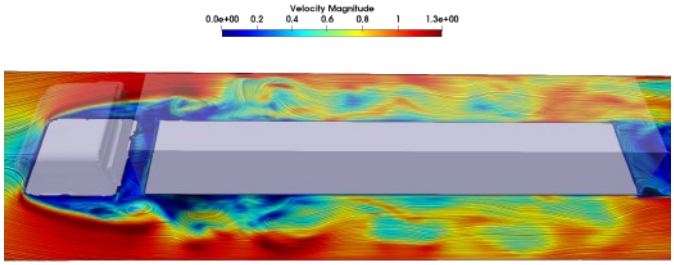
(**b**) Flow structures at different slices in the x direction.



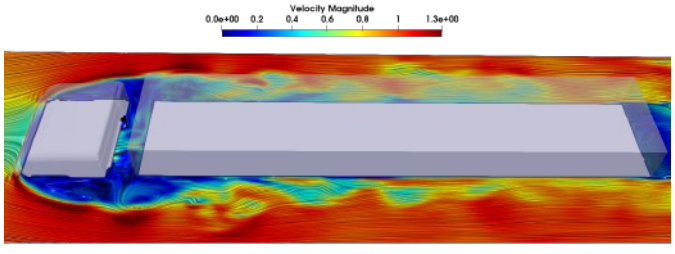
(**c**) Flow structures near the truck at $y = 0.05$



(**d**) Flow structures near the truck at $y = 0.10$



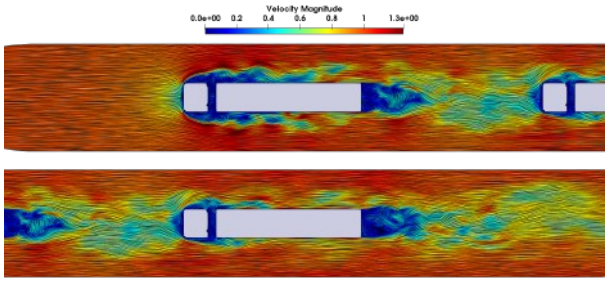
(**e**) Flow structures near the truck at $y = 0.15$



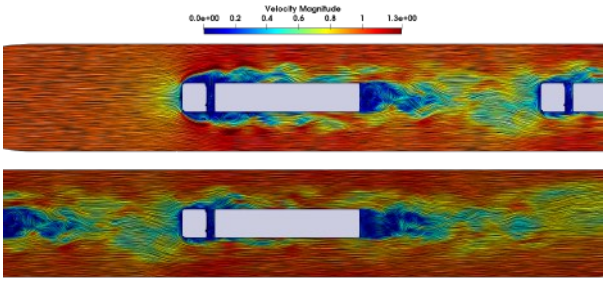
(**f**) Flow structures near the truck at $y = 0.20$

Fig. C.24. Flow structures around the truck at different locations.

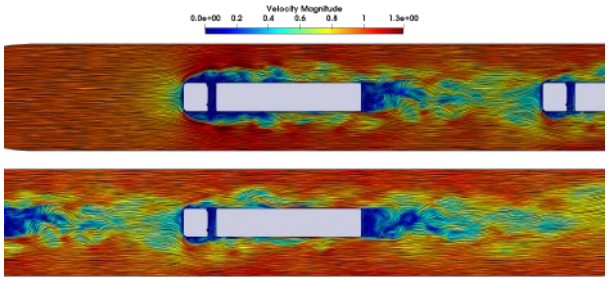
triggers the separation at the side of the tractor. The flow structure shows reduction in the re-circulation region at both sides of the tractor and earlier reattachment onto the trailer surface.



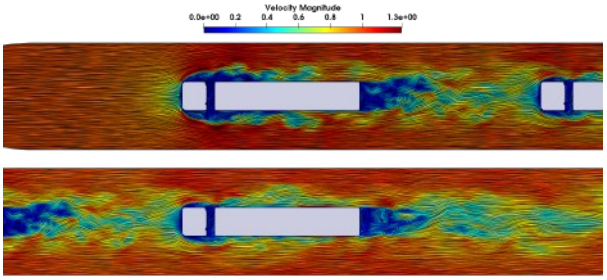
(**a**) Flow structures at $T = 0$



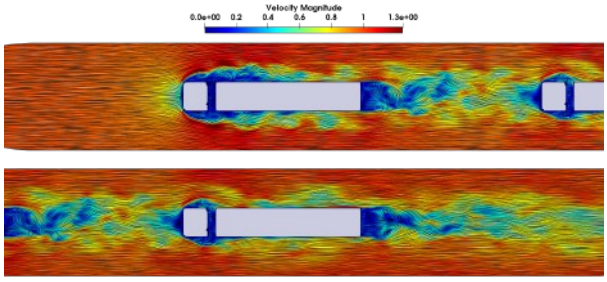
(**b**) Flow structures at $T = 0.25$



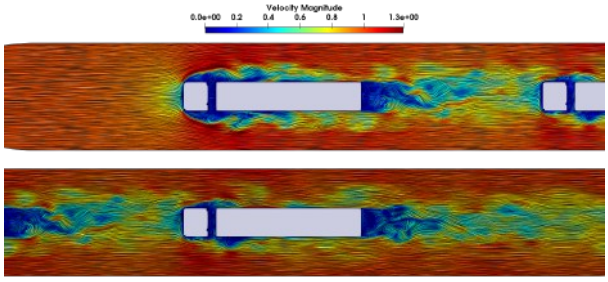
(**c**) Flow structures at $T = 0.5$



(**d**) Flow structures at $T = 0.75$



(**e**) Flow structures at $T = 1.0$



(**f**) Flow structures at $T = 1.25$

Fig. C.25. Flow structures for two platooning trucks in top-down view.