Accelerating Vortex Particle Methods by Downsampling the Vorticity Field Representation

Siemaszko, Jakub; Pasolari, Rention; van Zuijlen, Alexander

DOI
10.1002/fld.70002

Publication date
2025

Document Version
Final published version

Published in
International Journal for Numerical Methods in Fluids

Citation (APA)
Siemaszko, J., Pasolari, R., & van Zuijlen, A. (2025). Accelerating Vortex Particle Methods by Downsampling the Vorticity Field Representation. *International Journal for Numerical Methods in Fluids*, *97*(11), 1410-1425. https://doi.org/10.1002/fld.70002

International Journal for Numerical Methods in Fluids

WILEY

RESEARCH ARTICLE OPEN ACCESS

Accelerating Vortex Particle Methods by Downsampling the Vorticity Field Representation

Jakub Siemaszko | Rention Pasolari | Alexander van Zuijlen

Delft University of Technology, Delft, the Netherlands

Correspondence: Jakub Siemaszko (j.d.siemaszko@student.tudelft.nl)

Received: 24 September 2024 | **Revised:** 23 May 2025 | **Accepted:** 18 June 2025

Funding: The authors received no specific funding for this work.

Keywords: compression | downsampling | merging computational elements | remeshing | vortex particle methods

ABSTRACT

Computational efficiency of vortex particle methods (VPMs) is hindered by the particle count increasing in simulation time. To reduce the number of computational elements, two algorithms are presented that downsample the discretized vorticity field representation in two-dimensional variable-core-size VPMs. The two methods are based on existing schemes of particle merging and regridding, and are adapted to follow a compression parameter set a priori. The effectiveness of the schemes is demonstrated on two benchmark cases of external flow: A stationary Lamb-Oseen vortex and an advecting vortex dipole. In both cases, compression is associated with a drastic reduction in particle count and computation time at a cost of diffusive errors in the vorticity field. Crucially, for gentle compression steps applied at appropriate intervals, the immediate errors in the vorticity field are comparable to reference cases despite great improvements in computational time. To examine the long-term impact of compression on accuracy and performance, it is recommended that repeated compressive steps be tested on more complex cases of bluff-body wakes, with a focus on the impact of downsampling on surface forces.

1 | Introduction

Lagrangian methods in computational fluid dynamics (CFD) serve as an alternative approach to the more conventional Eulerian solvers such as finite volume (FVM) or finite element (FEM). In the Lagrangian framework, the behavior of fluid is described in the inertial reference frame of a fluid element as it moves and deforms in space and time. Accordingly, Lagrangian CFD methods model fluid flow as a number of advecting volumes or particles associated with some solution quantity. In the case of incompressible viscous flows, Lagrangian vortex particle methods (VPMs) represent the vorticity field as a number of discrete vortex particles, each carrying some finite circulation.

The most significant advantages of VPMs stand out as the shortcomings of Eulerian CFD methods, and include simple description of advection, low numerical dissipation, and trivial handling of boundary conditions at infinity [1]. Additionally, by nature of vortex elements, VPMs only resolve regions of sufficient vorticity, eliminating the need to model computationally irrelevant domains. These features make VPMs particularly fit for the description of external aerodynamics and modeling of wakes, and have historically been used extensively in that context [2–4].

Combining the strengths of both Eulerian and Lagrangian approaches, hybrid methods [5–7] have emerged as highly efficient and versatile tools for flow modeling. These methods

Abbreviations: AMR, adaptive mesh refinement; (A)VRM, (adaptive) vortex redistribution method; CFD, computational fluid dynamics; CR, compression ratio; FEM, finite element method; FVM, finite volume method; VPM, vortex particle method(s).

approach CFD by splitting the computational domain into Eulerian and Lagrangian sub-domains, with an interface between the two regions. The coupling of the two methods, motivated by their complementary nature, aims to utilize each model at its most efficient, which usually entails using an Eulerian solver near solid boundaries and particle methods in the wake.

In both pure-Lagrangian and hybrid methods, the particle count is often a factor limiting the computational efficiency of the solver. Modeling the particles' evolution in time requires the computation of pair-wise influence between all vortices, which results in a drastic increase in complexity with the number of particles N, most commonly scaling as $O(N^2)$ when using direct pair-wise computations, or as $O(N \log N)$, using multipole methods [8, 9]. This poor scaling is a significant limitation of VPMs, as the number of particles N can increase during a simulation run due to diffusion [10] and vorticity generation at solid boundaries [1]. As a result, the computational complexity of VPMs tends to steadily increase in simulation time, making long simulation runs expensive to resolve. This limitation is a particular concern in hybrid methods, where the Lagrangian domain can hold back an otherwise highly efficient solver.

To combat this issue, the current contribution aims to introduce and verify a number of algorithms that reduce the particle count in VPM simulations (here referred to as *compression* or *downsampling* methods), with a focus on downsampling vortices in the far wakes of solid bodies. This is achieved by analyzing a number of existing approaches and making adjustments to fit the current problem. As a secondary goal, the compression step is set out to act independently from other parts of the solver, such as advection or diffusion functions, so as to assure compatibility with various implementations of VPM. As a benchmark for verification, two test cases are considered: A pure-diffusion analytical case of the Lamb-Oseen Vortex [11] and an advecting vortex dipole, both chosen as simplified representations of wake structures.

In the past, various attempts have been made to address the growing particle count in VPMs. Rossi [12] presented a merging function, where multiple close-by particles are combined into one equivalent particle that conserves the 0th, 1st, and 2nd moments of vorticity. The moment-preserving approach is particularly suitable for flow problems focusing on computing surface forces induced by the vortex structures, such as the (aeroelastic) cylinder flow problem [13–15]. Indeed, conserving low vorticity moments assures that the global contribution of a downsampled vortex structure remains well-approximated, provided the structures are far enough away from the domain of interest [16].

This merging scheme was later improved by Stock [17], who incorporated particle merging as part of a solution-responsive diffusion scheme. This adaptive diffusion can in itself be treated as a downsampling approach, combining the concepts of core-spreading [18] and redistribution methods [10] to achieve local resolution coarsening near regions of low vorticity. However, the adaptive diffusion scheme does not answer the current problem as is; it does not, for instance, allow for artificial, forced downsampling in regions of high vorticity.

Aside from particle merging, remeshing methods, such as the one presented by Palha et al. [7], appear to be a promising prospect for compression schemes, allowing for easy manipulation of particle count via the mesh spacing parameter. However, the need to resolve the circulation of the new, meshed particles presents a challenge and a possible source for inaccuracies for the scheme in the context of downsampling.

Finally, various implementations of mesh-adaptivity have been historically proven efficient and reliable at limiting the resolution of irrelevant computational regions [8]. These include analogues to Eulerian r-adaptivity schemes [19] and adaptive mesh refinement (AMR) [20] that utilize various features of the vorticity and velocity fields as refinement conditions [21–26]. These methods, while efficient, are inherently tied to the advection process, as the particles can advect over boundaries of mesh resolution. Resolving such boundaries requires the use of a separate solver as part of the advection time step, which makes the schemes unfit for independent compression functions. Without such a boundary solver, adjacent regions of varying resolution cannot easily represent a smooth vorticity field, as the vastly different particle cores do not overlap sufficiently at the boundary. Nevertheless, various concepts of mesh-adaptivity, such as the conditions for resolution adjustment, can be noted.

The remainder of the paper is structured as follows. Firstly, the Lagrangian approach to flow modeling is introduced in Section 2, including the vorticity formulation of the Navier-Stokes equations and the description of the vorticity field discretization. Afterwards, the proposed compression methods are introduced and examined in Sections 3–5. The methods are then tested and their performance verified with the help of two benchmark cases in Sections 6 and 7. Finally, the conclusions are drawn in Section 8, focusing on the potential, limitations, and possible improvements to the examined schemes.

2 | Vortex Particle Methods

The Navier-Stokes (N-S) equations governing fluid flow can be expressed in terms of vorticity $\boldsymbol{\omega}$, defined as the curl of velocity. In the case of incompressible flow, the N-S equation(s) in vorticity form can be expressed as follows:

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u} + \frac{1}{\mathrm{Re}}\boldsymbol{\nabla}^2\boldsymbol{\omega} \tag{1}$$

A fluid flow problem is then modeled with additional equations representing, in order: Flow continuity, vorticity-velocity relation, initial conditions, and boundary conditions:

$$\boldsymbol{\nabla} \cdot \boldsymbol{u} = 0 \tag{2}$$

$$\boldsymbol{\nabla} \times \boldsymbol{u} = \boldsymbol{\omega} \tag{3}$$

$$\boldsymbol{\omega}(\boldsymbol{x}, 0) = \boldsymbol{\omega}_0 \tag{4}$$

$$\lim_{|\boldsymbol{x}| \to \infty} \boldsymbol{u}(\boldsymbol{x}) = \boldsymbol{u}_\infty \tag{5}$$

where $\boldsymbol{\omega}_0$ represents some initial vorticity field, and $\boldsymbol{u}_\infty$ is the free-stream velocity away from any perturbations.

In the case of two-dimensional VPMs, a number of simplifications can be applied to the flow equations. In 2D, the vorticity vector will always be perpendicular to the computational domain, which allows it to be modeled as a scalar according to: $\boldsymbol{\omega} = \omega \cdot \hat{\boldsymbol{z}}$. Additionally, the term $(\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\boldsymbol{u}$, corresponding to vortex stretching [1], reduces to zero in 2D. The simplified N-S equation takes the following form:

$$\frac{\partial \omega}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\omega = \nu \boldsymbol{\nabla}^2 \omega \tag{6}$$

Vortex particle methods approach solving Equation (6) by discretizing the vorticity field into a number of mollified particles, such that:

$$\omega(\boldsymbol{x}, t) \approx \omega_h(\boldsymbol{x}, t) = \sum_{p=0}^{N-1} g_p \cdot \zeta_{\sigma_p}(\boldsymbol{x} - \boldsymbol{x}_p(t)) \tag{7}$$

where g_p and σ_p represent, in order, the circulation ("strength") and core size of vortex p, and ζ_σ is a mollified, homogeneous kernel function, satisfying the following conditions:

$$\zeta_\sigma(\boldsymbol{x}) = \frac{1}{\sigma^n} \zeta_1\left(\frac{\boldsymbol{x}}{\sigma}\right) \tag{8}$$

$$\int_\Omega \zeta_\sigma(\boldsymbol{x}) d\Omega = 1 \tag{9}$$

for $n = 2, 3$ being the dimension of the examined problem.

The velocity induced by each vortex particle can be derived by solving the Poisson equation relating the vorticity and velocity field:

$$-\boldsymbol{\nabla}^2 \boldsymbol{u} = \boldsymbol{\nabla}\omega \tag{10}$$

resulting in a general solution in the form:

$$\boldsymbol{u} = \boldsymbol{u}_\infty + \boldsymbol{K} * \omega \tag{11}$$

$$\boldsymbol{K}(\boldsymbol{x}) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \frac{\boldsymbol{x}}{2\pi |\boldsymbol{x}|^2} \tag{12}$$

where $*$ represents the convolution operation.

For a discretized vorticity field with variable-core-size particles, the velocity formulation can then be expressed in the following form:

$$\boldsymbol{u} \approx \boldsymbol{u}_h = \boldsymbol{u}_\infty + \sum_p g_p \boldsymbol{K}_{\sigma_p}(\boldsymbol{x} - \boldsymbol{x}_p) \tag{13}$$

where $\boldsymbol{K}_\sigma = \boldsymbol{K} * \zeta_\sigma$ is the velocity kernel associated with the mollified vorticity kernel ζ_σ. In the context of this paper, ζ_σ is the Gaussian kernel in 2D:

$$\zeta_\sigma(\boldsymbol{x}) = \frac{1}{2\pi\sigma^2} \exp\{\frac{-|\boldsymbol{x}|^2}{2\sigma^2}\} \tag{14}$$

with the associated velocity kernel in the form:

$$K_\sigma = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \frac{\boldsymbol{x}}{|\boldsymbol{x}|^2} \left(1 - \exp\{\frac{|\boldsymbol{x}|^2)}{2\sigma^2}\}\right) \tag{15}$$

Note the arbitrary core size convention σ, which can differ from related literature [12, 17]. The kernel ζ_σ and (the magnitude of) its associated velocity kernel K_σ are presented in Figure 1.

The evolution of each particle in time follows from Equation (6). It is common to separate the advection and diffusion processes into their own sub-problems [8]. This is achieved by splitting Equation (6) into pure-advection and pure-diffusion problems as such:

$$\frac{\partial \omega}{\partial t} + (\boldsymbol{u} \cdot \boldsymbol{\nabla})\omega = 0 \qquad \text{(advection)} \tag{16}$$

$$\frac{\partial \omega}{\partial t} = \nu \boldsymbol{\nabla}^2 \omega, \qquad \text{(diffusion)} \tag{17}$$

Each of the sub-problems can then be rewritten for the discretized vorticity field in accordance with Equations (7) and (13).

Advection step:

$$\begin{aligned} \frac{d\boldsymbol{x}_p}{dt} &= \boldsymbol{u}(\boldsymbol{x}_p) \\ \frac{dg_p}{dt} &= 0 \end{aligned} \tag{18}$$

As per Equation (18), particles move according to the total velocity induced at their location, which is a sum of contributions from all other particles and the free-stream. This operation is commonly resolved either directly in $O(N^2)$ computations, or using a Fast Multipole Method (FMM) in $O(N \log N)$ computations [9]. Throughout this paper, direct computations are used with a fourth-order Runge-Kutta scheme to resolve the time derivative.

Diffusion step:

$$\begin{aligned} \frac{d\boldsymbol{x}_p}{dt} &= 0 \\ \frac{d\omega(\boldsymbol{x}_p)}{dt} &= \nu \boldsymbol{\nabla}^2 \omega(\boldsymbol{x}_p) \end{aligned} \tag{19}$$

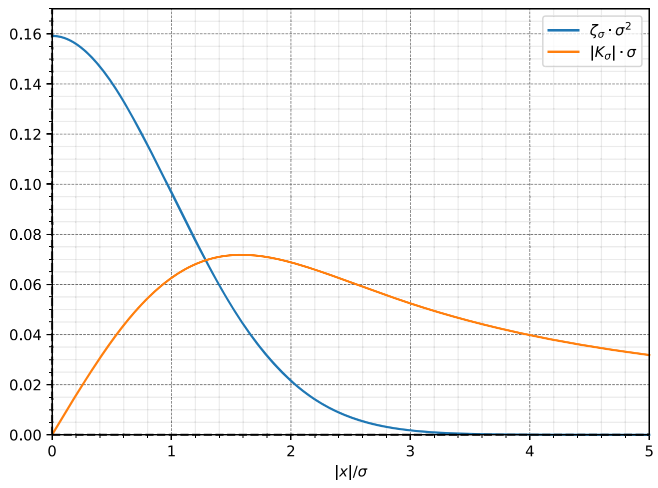


FIGURE 1 | The Gaussian vorticity kernel ζ_σ (in blue) and (the magnitude of) its associated velocity kernel $|\boldsymbol{K}_\sigma|$ (in orange). Both functions are non-dimensionalized by multiplying by σ^2 and σ, respectively. [Colour figure can be viewed at wileyonlinelibrary.com]

The diffusion of vorticity can be modeled in a variety of methods, including probabilistic random-walk schemes [27], particle core-spreading [18], and redistribution methods [10, 28]. In this paper, a combination of deterministic methods is considered, and, in particular, Stock's adaptive vortex redistribution method (AVRM) [17], which combines the concepts of core-spreading and vortex redistribution with a particle merging scheme [12] for a solution-adaptive diffusion method with uniformly varying particle cores.

For implementation purposes, the pHyFlow solver [7, 29], developed by Palha et al., is used throughout the remainder of the paper. PHyFlow is a hybrid-method solver combining the Eulerian Finite Element Method with Lagrangian Vortex Particles. The Lagrangian portion of the solver is based on Python 3.9 (although originally developed in Python 2), with efficiency-critical code written in lower-level languages, such as C++, C, Cython, and CUDA. The solver incorporates all relevant methods examined throughout this section, and additionally includes various acceleration methods, such as GPU parallelization for pair-wise operations.

3 | Methods

The main objective of any considered compression function is to reduce the number of vortex particles representing the vorticity field. Naturally, each application of such a process will lead to increased representation errors, turning any implementation of compression into an optimization problem with a goal of minimizing vorticity errors subject to some desired particle count, or, oppositely, minimizing the number of vortex particles subject to some error bounds.

In this paper, the former option is considered. That is, the number of particles desired after the compression step is specified a priori via the compression ratio parameter CR, defined in Equation (20):

$$CR := \frac{N_1}{N_0} \tag{20}$$

where N_0 and N_1 represent, in order, the particle counts before and after the compression step.

Importantly, in order for the new particle distribution to approximate the same vorticity field, the post-compression particles have to be, on average, wider than the pre-compression ones. The growth of particle cores makes any compressive step inherently diffusive in a process analogous to core-spreading [18].

The introduction of spatial variation in particle cores is yet another source of representation errors, as the overlap between any two particles will vary from the expected value of $\lambda := \frac{\sigma}{h}$ (in the range of 0.5–2.0). To assure smoothness of the vorticity field, the variation of σ is limited in space by a lapse rate parameter [17]:

$$\frac{\sigma_1 - \sigma_2}{|\boldsymbol{x}_1 - \boldsymbol{x}_2|} \leq |\nabla\sigma|_{\max} \tag{21}$$

where indices 1, 2 represent any two particles. The considered lapse ratio is in the order of 0.001–0.1m/m.

Additional concerns arise when considering the repeated application of downsampling, as any additional step will likely amplify the vorticity errors and lead to further, potentially excessive, particle count reductions. Thus, in case of repeated downsampling, the compression period should be carefully chosen in unison with the compression ratio, so as not to induce excessive resolution loss. One could also consider a responsive implementation of downsampling, with each compression event triggering only after the particle count reaches a certain threshold $N \geq N_{\mathrm{TH}}$. Within the scope of this paper, only the simple, periodic implementation is considered, with a compression period based on problem size growth of the base case, which is assumed to be known a priori.

If the compression is performed far away from any relevant simulation regions, as is set out in this paper, the most significant influence of the downsampled domain can be approximated by its lowest-order features [16], quantified as vorticity moments $m_{k,i}$:

$$m_{k,i}(\omega, x_m, y_m) := \int_\Omega \omega(x,y)(x - x_m)^i (y - y_m)^{k-i} d\Omega \tag{22}$$

where $k \in \{0, 1, 2, 3, \dots\}$, $i \in \{0, 1, \dots, k\}$ are double-indices of the moment, and (x_m, y_m) represents the evaluation point, usually taken as the center of mass of the particle distribution.

In order for the post-compression vorticity and velocity fields to approximate the previous ones in the global sense, the solution should conserve (or approximate) the lowest moments of vorticity (here chosen as $k = 0, 1, 2$), such that the most significant features of the vorticity field are preserved. This is also particularly important if surface forces are of interest, as the lowest-order features will have the highest impact on the surface integrals, provided the physical distance between the surface and vortex structure is large [30–32]. This can be seen, for instance, by expanding the (2D) Fourier series of $\omega(\boldsymbol{x})$ in terms of its moments [16]:

$$\begin{aligned}
\widehat{\omega(\boldsymbol{x})}(\boldsymbol{\xi}) &= \int_{\mathbb{R}^2} \omega(\boldsymbol{x}) \exp(-i\boldsymbol{\xi}\cdot\boldsymbol{x}) d\boldsymbol{x} \\
&= \sum_n^\infty \sum_m^\infty \frac{(-i\xi_x)^n(-i\xi_y)^m}{n!m!} \int_{\mathbb{R}^2} \omega(\boldsymbol{x}) x^n y^m dx dy \\
&= \sum_n^\infty \sum_m^\infty \frac{(-i\xi_x)^n(-i\xi_y)^m}{n!m!} m_{n+m,n}(\omega, 0, 0)
\end{aligned}$$

It is clear that for high-scale (low frequency) modes $\xi_x, \xi_y << 1$, the Fourier transform of ω is well-approximated by the lowest powers of ξ_x, ξ_y, corresponding to the contributions of low vorticity moments.

For a discretized vorticity field ω_h, the relevant moments simplify to a sum of moments of individual vortex particles:

$$\begin{aligned}
m_{0,0} &= \sum_p g_p \\
m_{1,1} &= \sum_p g_p(x_p - x_m) \\
m_{1,0} &= \sum_p g_p(y_p - y_m) \\
m_{2,2} &= \sum_p g_p(x_p - x_m)^2 + g_p\sigma_p^2
\end{aligned}$$

$$m_{2,1} = \sum_p g_p (x_p - x_m)(y_p - y_m)$$
$$m_{2,0} = \sum_p g_p (y_p - y_m)^2 + g_p \sigma_p^2 \tag{23}$$

where $g_p \sigma_p^2$ is the second moment of the particle core in one direction x or y (mind the convention used for the core size σ).

The compression methods examined in this paper are inherently based on the conservation (or, at least, approximation) of the lowest-order features as presented in Equation (23). In an analogous way, additional moment conservation restrictions can be added to produce a higher-order compression scheme.

With the implementation of conservation laws presented in Equation (23), a compression algorithm aims to achieve a particle distribution matching a desired CR. The following sections present contrasting approaches to this problem.

4 | Particle Merging

The merging scheme introduced by Rossi [12] attempts to downsample the vorticity field by replacing a number of excessively overlapping particles by a single, *merged* particle. The position, strength, and core size of the merged particle can be easily determined by the moment conservation equations, similar to Equation (23):

$$\overline{g} = \sum_i g_i \tag{24}$$

$$\overline{g} \cdot \overline{\boldsymbol{x}} = \sum_i g_i \cdot \boldsymbol{x}_i \tag{25}$$

$$2\overline{g} \cdot \overline{\sigma}^2 = \sum_i g_i \left(2\sigma_i^2 + |\boldsymbol{x}_i - \overline{\boldsymbol{x}}|^2\right) \tag{26}$$

where the indices i denote all particles participating in the merging event, and $(\overline{\boldsymbol{x}}, \overline{g}, \overline{\sigma})$ are the position, strength, and core size of the resulting merged particle.

The merging problem thus reduces to finding a set of potential *merge candidates* which, when merged, do not induce excessive error in the vorticity field. In Rossi's approach, a number of candidate sets C are constructed by searching for neighbors of each particle p within some radius R; each neighbor is then accepted or rejected from the candidate set based on an estimate of error induced by the potential merging event. The process is then repeated until all merge possibilities have been exhausted. Crucially, the merging step only considers particles that overlap excessively (in Rossi's case, $R \leq 0.1\frac{\sigma}{\lambda}$). Indeed, the merge-induced vorticity error will increase drastically for higher values of R, up to a maximum of $\approx \frac{g}{2\pi\sigma^2}$ when merging two far-away particles.

An altered implementation of the merging scheme was introduced by Stock [17], who incorporated merging to remove excess particles during a VRM diffusion step. This implementation replaces the core-size computation of Equation (26) by a simplified relation presented in Equation (27), which no longer guarantees moment conservation.

$$\overline{\sigma} = \sqrt{\frac{|g_1|}{|g_1| + |g_2|}\sigma_1^2 + \frac{|g_2|}{|g_1| + |g_2|}\sigma_2^2} \tag{27}$$

Crucially, in this implementation, the merging step is treated as a (repeated) pair-wise computation between two participating particles, negating the need to pre-compute the candidate set C. Additional advantages of this scheme include better stability for extreme values and the ability to handle oppositely-signed particles. For these reasons, Stock's approach to merging is preferred in the context of this paper.

For the purposes of this paper, neither of the two implementations is compatible as a compression function as is, as a top-down compression step has to be able to include particles that are not overlapping excessively. This raises two critical concerns for the current implementation: The strict radius bound R, and the error estimation argument.

To address the former, we propose preceding the merging step by artificial coarsening (core growth) of all participating particles. This step is designed to increase the particle density in the core-size-normalized neighborhoods of all vortices, resulting in excessive overlap of particles throughout the domain. The coarsening is achieved by scaling the particle size by a constant factor dependent on the desired compression ratio:

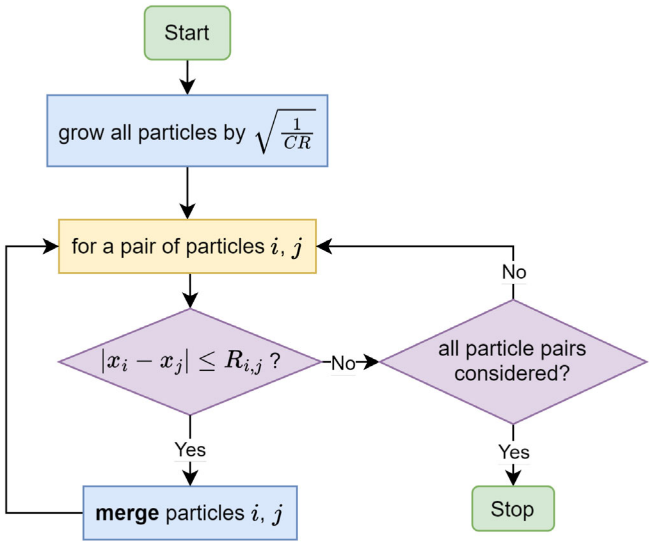
$$\sigma_{i,1} = \sigma_{i,0} \cdot \sqrt{\frac{1}{CR}} \tag{28}$$

This pre-coarsening step is designed to force an adequate number of merging events to achieve the desired particle count of $N_1 = \text{CR} \cdot N_0$. By design of the methods of Rossi and Stock, merging events will cease when the coarsened particle distribution is no longer excessively overlapping, or, equivalently, when the post-merging particles occupy (approximately) the same area as the pre-merge particles, accounting for overlap:

$$A_0 \approx \frac{1}{\lambda^2}\sum^{N_0}\sigma_{p,0}^2 = N_0 \frac{\overline{\sigma_0^2}}{\lambda^2} = \frac{N_1}{\text{CR} \cdot \lambda^2}\overline{\sigma_0^2}$$
$$\approx N_1 \frac{\overline{\sigma_1^2}}{\lambda^2} = \frac{1}{\lambda^2}\sum^{N_1}\sigma_{p,1}^2 \approx A_1 \tag{29}$$

The area equivalency presented above is approximate in nature and will likely result in deviations from the expected compression ratio, which is deemed acceptable for the purpose of this paper. Nevertheless, the intermediate steps of Equation (29) present opportunities for a more elaborate core-scaling approach, or even a solution-responsive scheme for localized compression. In the context of this paper, only the simple scaling of Equation (28) is considered.

The pre-coarsening step is yet to address the issue of error estimation, which is likely to influence the efficiency of the scheme. Indeed, the maximum error of a single merging event will likely be comparable to the particle's vorticity peak of $\frac{g}{2\pi\sigma}$, while the combined total error of the procedure is expected to be much lower (assuming that the post-compression vorticity field is a good approximation of the original). Basing the merging step on an error estimation is thus likely to reject valid merging events, drastically reducing the efficiency of the scheme. The current



Start
grow all particles by $\sqrt{\frac{1}{CR}}$
for a pair of particles i, j
$|x_i - x_j| \leq R_{i,j}$?
No
all particle pairs considered?
No
Yes
Yes
merge particles i, j
Stop

FIGURE 2 | Flowchart of the merge compression algorithm. [Colour figure can be viewed at wileyonlinelibrary.com]

implementation neglects any error estimation and instead bases merging solely on the (core-normalized) distance between particles. Here, we use an arbitrary distance bound based on Stock's implementation:

$$R_{i,j} = 0.5 \cdot \frac{\min(\sigma_{i,0}, \sigma_{j,0})}{\lambda} \cdot \sqrt{\frac{1}{\mathrm{CR}}} \tag{30}$$

The algorithmic description of the current implementation of merging is presented in Figure 2, with all merging events performed according to Equations (24, 25), and (27).

5 | Remeshing

Aside from particle merging, existing remeshing schemes are a natural candidate for simple and efficient compression methods, as in this case, controlling the compression level is trivialized by varying the mesh spacing h. In addition, the core size of all new particles can be conveniently assigned based on the mesh spacing and the required overlap criterion λ:

$$\sigma_{1,p} = \lambda \cdot h \tag{31}$$

The mesh spacing h is to be chosen such that the desired compression ratio CR is followed. Similarly to the previous case, it is assumed that the pre-compression particle distribution satisfies the overlap criterion (at least, approximately), in which case, the required post-compression spacing is approximated in the same manner as in Equation (29) by way of area equivalency:

$$A_1 = \frac{1}{\lambda^2}\sum_p \sigma_{p,1}^2 = N_1 h^2 \approx \frac{1}{\lambda^2}\sum_p \sigma_{p,0}^2 \approx A_0$$
$$\Rightarrow h \approx \sqrt{\frac{A_0}{\mathrm{CR}\cdot N_0}} \approx \frac{1}{\lambda}\sqrt{\frac{\sum_p \sigma_{p,0}^2}{\mathrm{CR}\cdot N_0}} \tag{32}$$

The above approximation is exact if the initial particle distribution satisfies the overlap criterion (e.g., if the initial distribution is also a regular mesh with overlap λ). In any other situation, the post-compression particle count will likely vary from its expected value of $N_1 = \mathrm{CR}\cdot N_0$. Other approximations for h can be considered to assure stricter adherence to the desired compression ratio. For instance, the area A_0 occupied by the pre-compression particles can be estimated empirically using cell-counting methods.

The final step of remeshing is to assign the circulation of each new particle. A method presented by Palha et al., relying on the calculation of induced vorticity, assigns the circulation of each particle as:

$$g_p = \omega(x_p, y_p)\cdot h^2 \tag{33}$$

This scheme can be shown to approximate the low moments of vorticity and converge to the original vorticity field as $h \to 0$. It is also simple to implement and parallelize, and has been proven reliable in the context of the pHyFlow solver. However, the need to compute the induced vorticity at each particle location makes it inefficient in a variable-core-size setting. The circulation assignment can also be shown to have a relatively poor $O(h^2)$ scaling in the pointwise representation error of the vorticity field.

A natural alternative to the induced-vorticity assignment would be to directly use the moment conservation equations, in a manner similar to Stock [17]. In a constant-core-size setting, moment conservation (up to an arbitrary order) can be enforced with the use of interpolation kernels [8], such as the commonly used M_4' kernel. However, in the current case of variable-core particles, these kernels no longer guarantee moment conservation. Despite this, the error introduced by remeshing with the use of an interpolation kernel is expected to be small, as the spatial variation of particle core sizes is limited by Equation (21).

Throughout the remainder of this paper, the remeshing scheme is utilized with the M_4' kernel for particle circulation assignment. The kernel is defined as follows:

$$M_4'(z) = \begin{cases} 1 - \frac{5}{2}|z|^2 + \frac{3}{2}|z|^3 & \text{if} \quad |z| \leq 1 \\ \frac{1}{2}(1-|z|)(2-|z|)^2 & \text{if} \quad 1 < |z| \leq 2 \\ 0 & \text{if} \quad |z| > 2 \end{cases} \tag{34}$$

The circulation of each particle is then assigned as a tensor product of two kernel functions, taking into account the contributions of all surrounding pre-compression particles:

$$g_j = \sum_i M_4'\left(\frac{|x_j - x_i|}{h}\right)\cdot M_4'\left(\frac{|y_j - y_i|}{h}\right)\cdot g_i \tag{35}$$

where i, j are the indices of the pre- and post-compression particles respectively. The overview of the entire meshing process is presented in Figure 3.

Unlike the merging method examined in Section 4, remeshing has no way of adapting to the pre-compression solution. In particular, the remeshing step results in a constant-core-size particle patch, even if the pre-compression field already introduced core-size variation. In that case, simple remeshing would result in averaging the core sizes of the particles over the entire compression domain, possibly causing excessive or insufficient compression in some regions.

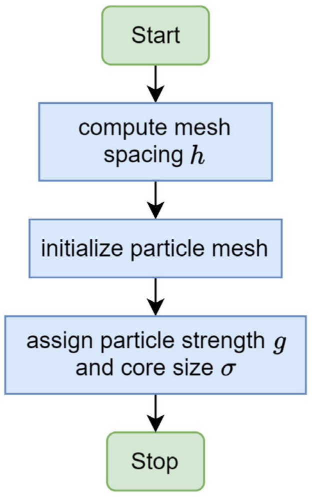
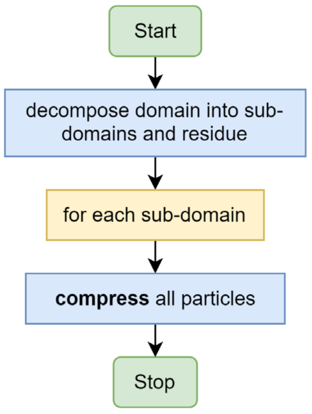


FIGURE 3 | Flowchart of the remesh compression algorithm. [Colour figure can be viewed at wileyonlinelibrary.com]



Start
decompose domain into sub-domains and residue
for each sub-domain
compress all particles
Stop

FIGURE 4 | Flowchart of a domain decomposition wrapper with an arbitrary compression function. [Colour figure can be viewed at wileyonlinelibrary.com]

To address this issue, the remeshing scheme is to be utilized locally on some subdomains of the computational domain: $\Omega' \subseteq \Omega$, each subject to its own remeshing scheme (possibly even with compression ratio varying per region). In the design case of flow around a cylinder, the far wake can be coarsened in strips along the downstream direction, assuring a gradually decreasing particle count away from the solid boundary.

This procedure can be achieved by preceding the compression step by decomposing the computational domain Ω into smaller subdomains Ω'_i, fed into the compression function, and the residue set Ω_{res}—the particles exempt from compression. Depending on the modeled problem, a number of domain decomposition functions can be considered, including directional, annular, or even cluster-based splitting. In such an implementation, the domain decomposition is compatible with an arbitrary compression function, including both methods examined in this paper. The general description of domain decomposition is presented in Figure 4.

6 | Results

The methods examined in Sections 4 and 5 have been implemented in the Lagrangian solver of the pHyFlow software, serving as a base for verification with the use of two benchmark cases: A stationary diffusing point vortex and an advecting vortex dipole. Crucially, the design case of a cylinder wake problem is not considered in this paper due to the limitations of the current solver implementation. This means that the impact of compression on large-scale wake structures, and the subsequent impact on unsteady surface forces are not studied here, and are instead referred to as potential directions for future work.

In both test cases, the simulations were performed with the use of hardware of the DelftBlue supercomputer [33], with each run using a single GPU type-A node comprising of four NVidia Tesla A100 32GB GPU's, two AMD EPYC 7402 24C CPU's, and a total of 256GB of available CPU memory. However, the hardware was limited to only 16 CPU cores and 64GB of CPU memory during each run. The inputs, description, and immediate results of each test case are presented in the following subsections.

6.1 | Choice of Parameters

Recalling the design case of a cylinder flow problem, the verification cases are designed to act as a representative simplification of the wake of a bluff body. Accordingly, the compression parameters are chosen to resemble the application of compression on a more complex problem of bluff-body wakes.

As reasoned in Section 3, the main parameter influencing the impact of compression is the ratio CR, which will have a drastic effect on the accuracy of the post-compression vorticity field, with representation errors likely to increase drastically for extreme downsampling cases of CR $< 20\%$. Of additional concern is the compression period, which could lead to further error amplification.

Based on the design case, focus is put on gentle compression steps (CR $\geq 80\%$), with a compression interval allowing for overall gradual reduction of particle count in simulation time. More extreme cases of CR $= 30\%, 50\%$ are also considered for comparison. The compression interval is tuned on a per-case basis, dependent on the problem growth rate of the base case.

In the context of the two (relatively simple) benchmark cases, no domain decomposition is used, resulting in the compression encapsulating the entire computational domain. The cases are treated as representing a single sub-domain of a larger wake problem.

6.2 | Stationary Lamb-Oseen Vortex

The Lamb-Oseen Vortex is an idealized case of a stationary diffusing point vortex [11] with a known analytical solution. For a vortex of strength Γ modeled as a δ-function at time $t = 0$, the vorticity field at time t will be the Gaussian distribution:

$$\omega(r,t) = \frac{\Gamma}{4\pi\nu t}\exp\left(\frac{-r^2}{4\nu t}\right) \tag{36}$$

The numerical solver was initialized at time $t = 4\Delta t$ as a single particle at $(0,0)$ of strength 1.0 and size $\sqrt{8\nu\Delta t}$ to account for the diffusion of the δ-function. The simulation was run for a total of 20 s, in increments of 0.02 for a total of 1000 time steps. Additional simulation parameters are presented in Table 1.

Both of the examined methods were tested, with a single compression step at time $t = 10$s (time step 500). As a reference, the base case of AVRM diffusion with no compression is considered. Additional diffusion implementations are also tested for comparison, including scattered and mesh-based redistribution methods (VRM [10]/Tutty [28]). All cases utilize direct advection calculations with a fourth-order Runge-Kutta scheme to resolve Equation (18).

Qualitatively, the results of the compression runs match their reference counterparts, with the vortex diffusing at more or less the same pace. The contours of an example compression case can be seen on Figure 5 compared to a reference case of remeshing redistribution (Tutty). Immediate impact of compression can also be observed in the particle distribution directly after the compression step at time $t = 10s$, presented in Figure 6.

6.3 | Vortex Dipole

The case of an advecting vortex dipole presents an opportunity to compare the performance of the compression methods with previous results of the pHyFlow solver [7, 29] and test their performance against advection errors.

The initial vorticity field in this case is set as a sum of two diffused point vortices:

$$\omega(\boldsymbol{x},0) = \omega_0\left(1 - \frac{r_1^2}{R^2}\right)e^{-(r_1^2/R^2)} - \omega_0\left(1 - \frac{r_2^2}{R^2}\right)e^{-(r_2^2/R^2)} \tag{37}$$

TABLE 1 | Simulation parameters for the Lamb-Oseen vortex case.

Parameter	Symbol	Value	Dimension
Diffusion and advection time step	$\Delta t_a = \Delta t_d$	$2 \cdot 10^{-2}$	s
Kinematic viscosity	ν	10^{-3}	m^2/s
Overlap ratio	λ	1.41	—
AVRM thresholds (Ignore/Adapt/Merge)	$(\Gamma_I, \Gamma_A, \Gamma_M)$	$(10^{-6}, 10^{-3}, 10^{-3})$	m^2/s
AVRM Lapse Rate	$\lvert\nabla\sigma\rvert_{\max}$	0.1	—
Population control thresholds (Local/Global)	$(\Gamma_{loc}, \Gamma_{glob})$	$(10^{-12}, 10^{-12})$	m^2/s
Compression time step	Δt_{comp}	10.0	s

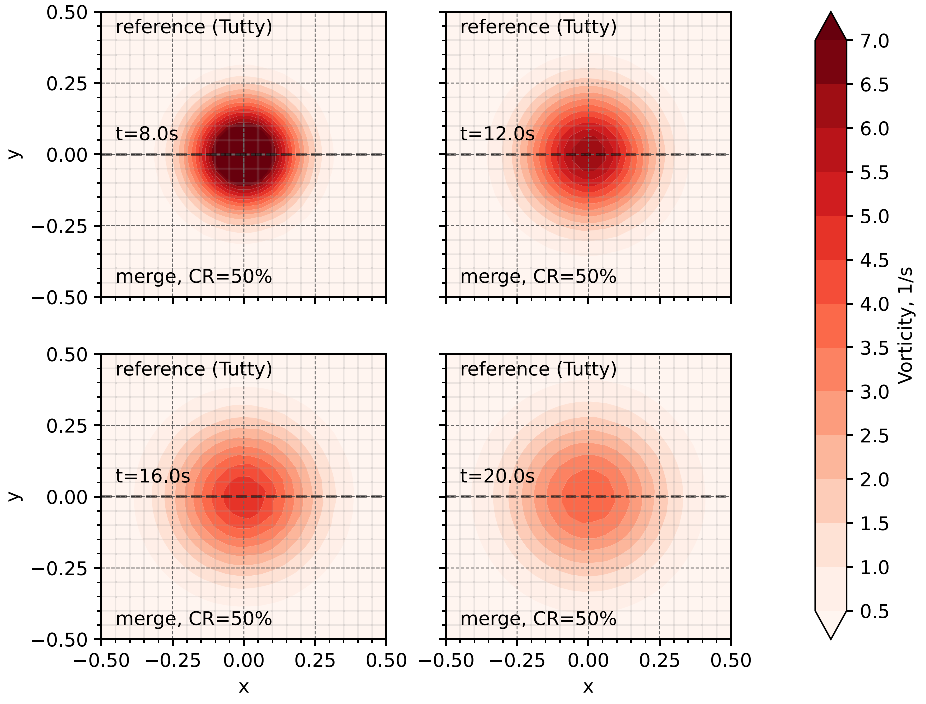


FIGURE 5 | The evolution of the vorticity field of the Lamb-Oseen vortex at time $t = [8.0, 12.0, 16.0, 20.0]s$ for the reference case (upper half) and a merge compression run with CR = 50% (lower half). The contours are in accordance with the analytical reference [11]. [Colour figure can be viewed at wileyonlinelibrary.com]

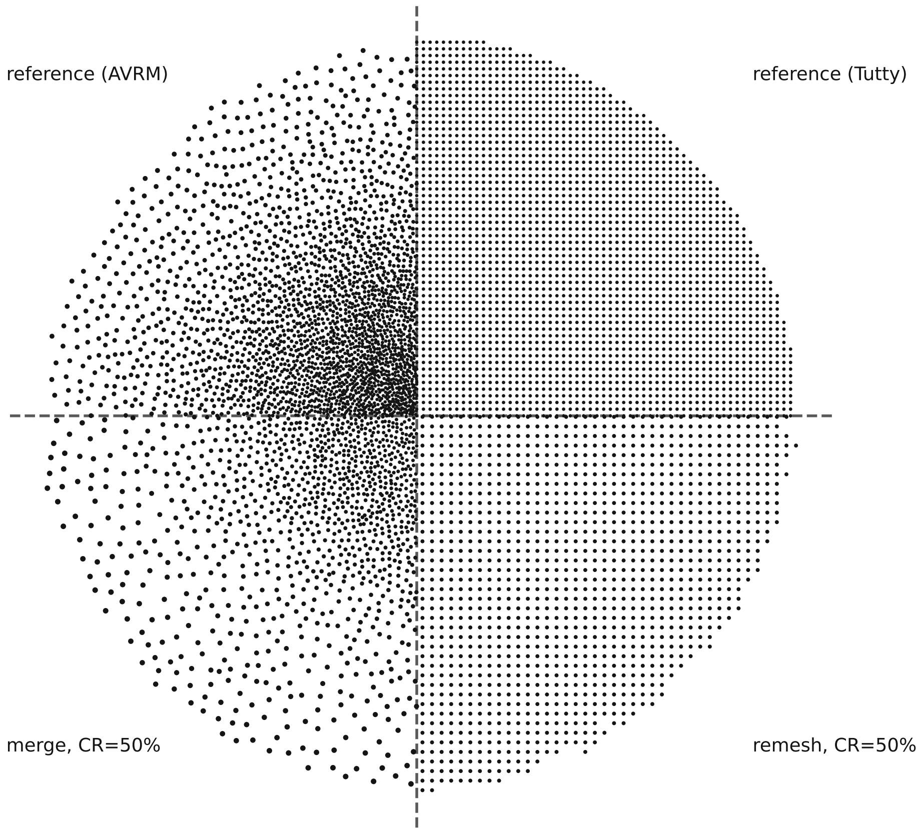


FIGURE 6 | Particle distribution of the Lamb-Oseen vortex case immediately after a compression step at time step 500 for different compression methods and reference cases. The total particle counts at this time step are: 9, 504 (AVRM), 10, 093 (Tutty), 4, 796 (remesh, CR = 50%), 4, 066 (merge, CR = 50%).

TABLE 2 | Simulation parameters for the vortex dipole case.

Parameter	Symbol	Value	Dimension
Diffusion and advection time step	$\Delta t_a = \Delta t_d$	$2.5 \cdot 10^{-4}$	s
Kinematic viscosity	ν	$1.6 \cdot 10^{-3}$	m^2/s
Overlap ratio	λ	1.0	—
AVRM thresholds (Ignore/Adapt/Merge)	$(\Gamma_I, \Gamma_A, \Gamma_M)$	$(10^{-6}, 10^{-3}, 10^{-3})$	m^2/s
AVRM Lapse Rate	$\lvert\boldsymbol{\nabla}\sigma\rvert_{\max}$	0.1	—
Population control thresholds (Local/Global)	$(\Gamma_{loc}, \Gamma_{glob})$	$(10^{-8}, 10^{-8})$	m^2/s
Compression time step	Δt_{comp}	0.25	s

where $R = 0.1$ is the radius of each vortex, $\omega_0 = 299.528385375 s^{-1}$ is the characteristic vorticity of the dipole [29], and r_1, r_2 are the distances to the center of each vortex:

$$r_i^2 = (x - x_i)^2 + (y - y_i)^2 \tag{38}$$

The initial vortex positions are set to: $(x_1, y_1) = (-1.0, 0.1)$, and $(x_2, y_2) = (-1.0, -0.1)$.

The solver was initialized by creating a mesh of particles in a rectangle bounded by: $-1.5 \leq x < 0.5$, $-0.6 \leq y < 0.6$, with an initial spacing of $h_0 = \sigma_0/\lambda = \sqrt{12\nu\Delta t_d}$. The circulation of each particle was initialized with the use of Equation (37), such that: $g_{p,0} = \omega(\boldsymbol{x}_p, 0) \cdot h_0^2$. Any particles with negligible circulation $(g_p < \Gamma_{glob})$ were subsequently removed during the first time step using a population control function [7]. The simulation was run for a total of 1 s, in 4000 steps of size $\Delta t_a = \Delta t_d = 2.5 \cdot 10^{-4}$. Additional simulation parameters are presented in Table 2.

Similarly to the previous case, both merging and remeshing compression methods were considered with compression ratios between 30% and 100%. In this case, multiple compression steps were performed during the simulation run, with a compression interval of 0.25s, or 1000 time steps, chosen to induce overall particle reduction in simulation time. The same set of reference cases is considered, including implementations of vorticity redistribution (VRM, Tutty) and Stock's AVRM, all with direct advection.

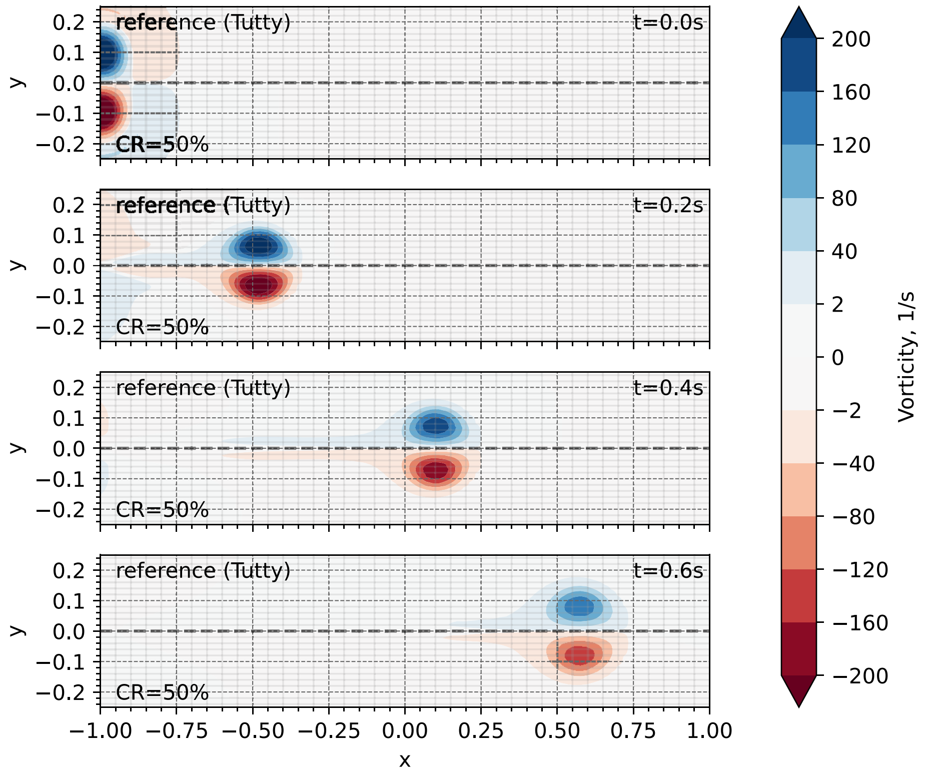


FIGURE 7 | The evolution of the vorticity field of the vortex dipole at $t = [0.0, 0.2, 0.4, 0.6]s$ for the reference case (upper half), and a merge compression run with CR = 50% (lower half). The contours are in accordance with the corresponding plots in the work of Palha & Pasolari [7, 29]. [Colour figure can be viewed at wileyonlinelibrary.com]

The immediate simulation result shows adequate behavior of the compression functions, with even a highly compressed case of CR = 30% not inducing noticeable advection errors. The vorticity contour plots of an example compression case can be seen on Figure 7 compared to a reference case.

7 | Discussion

The two tested methods have been found to, predictably, have a significant impact on the total particle count and computation time at a cost of accuracy in the vorticity field. The impact of compression on various aspects of simulation performance is examined in detail in the following subsections.

7.1 | Output Noise Filtering

Preceding the discussion of results, a note on data processing is due. The AVRM diffusion scheme used throughout the compression runs has been observed to suffer from high-amplitude high-frequency errors in the vorticity field, likely introduced by a non-negative least-squares solver used in the vorticity redistribution step [17]. To ease the analysis of simulation results, data ranges affected by noise are treated with a second-order Savitzky-Golay filter [34] with a window size of 201 samples/time steps. Examples of data filtering can be seen on Figures 8,9 and 11.

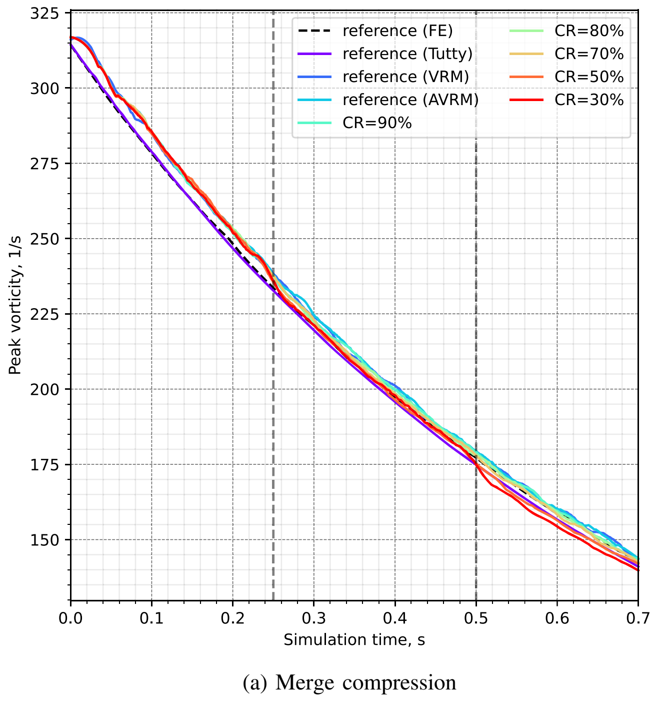
7.2 | Impact on Diffusion

As introduced in Section 3, the coarsening of vortex particles during compression can be treated as artificial diffusion via core-spreading [18]. The examined benchmark cases show that compressing the vorticity distribution indeed results in accelerated diffusion, with each compression step resulting in a step error in the vorticity field. The introduced error naturally varies per compression level, with the highest errors present in the high-compression cases.

The diffusive effect can be seen, for instance, in the study of peak vorticity of the vortex dipole case (Figure 8). Comparing the evolution of peak vorticity with a high-resolution Eulerian reference, taken from Palha et al. [7], a clear step-wise shift can be seen immediately after each compression step. To highlight the error magnitude, Figure 9 presents the relative error of the shift compared to the Eulerian reference. The error presented in the figure is defined as follows:

$$\epsilon_{rel} = \frac{\omega_{max} - \omega_{ref,max}}{\omega_{ref,max}} \tag{39}$$

It is clear that both methods introduce measurable errors in peak vorticity, with the high-compression cases of CR ≤ 50% resulting in peak vorticity shift in the order of 5% of the true peak. Crucially, a number of gentle compression runs of CR = 80%, 90% can be observed to not induce significant diffusive errors when



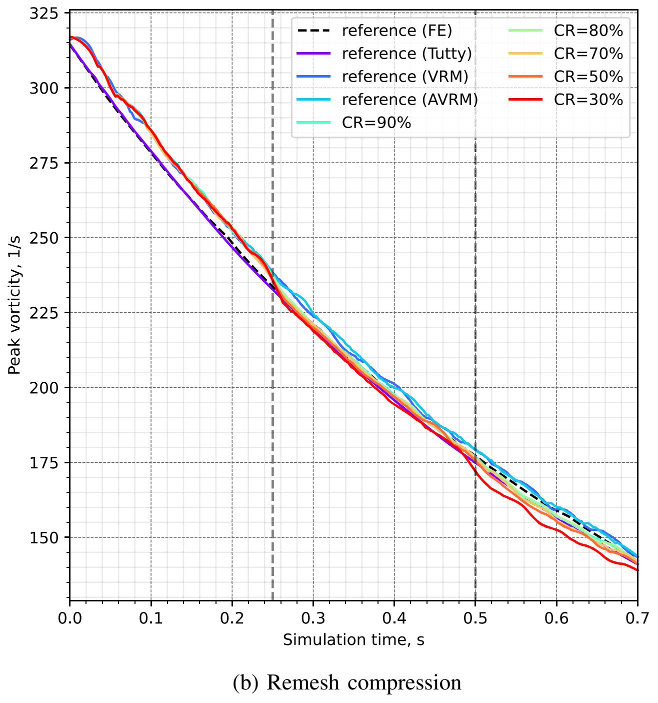
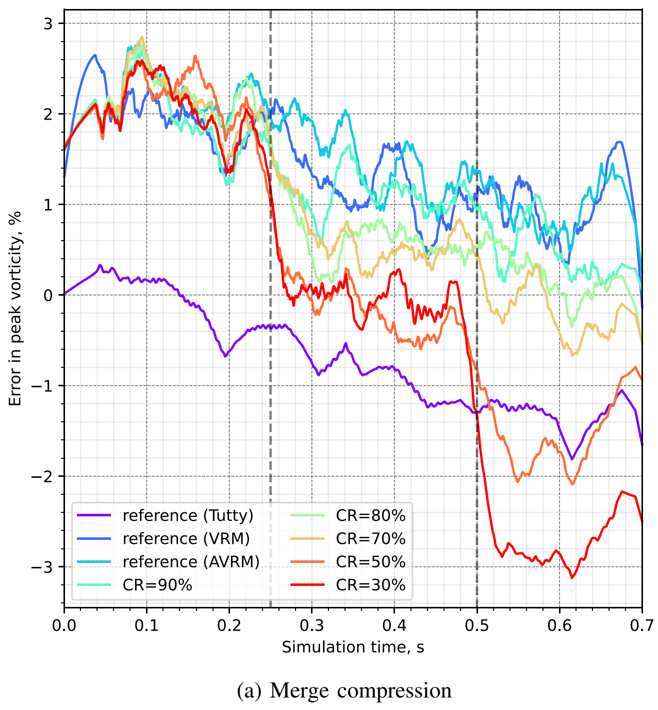


FIGURE 8 | The evolution of the peak vorticity of the vortex dipole case for different compression levels, compared to base cases and a high-resolution Eulerian reference [7]. All data ranges are subject to a Savitzky-Golay filter [34].



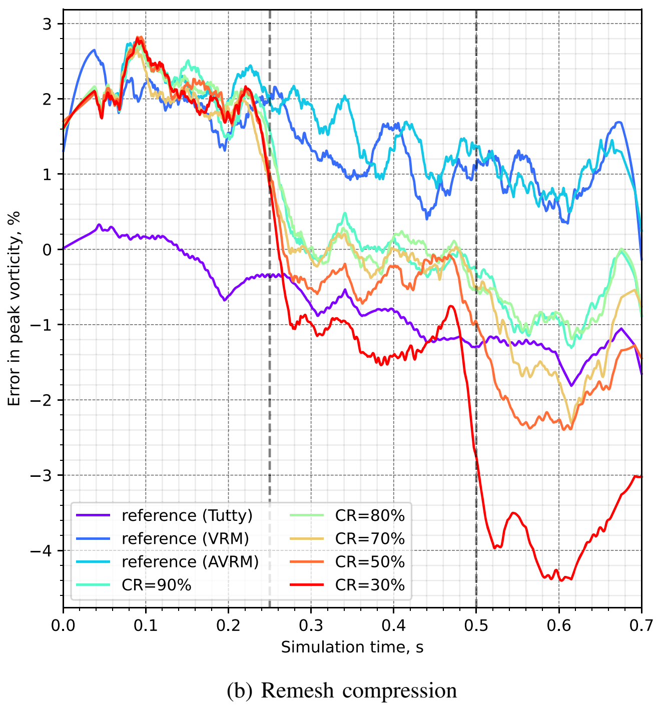


FIGURE 9 | The error in peak vorticity of the vortex dipole case relative to a high-resolution Eulerian reference [7]. All data ranges are subject to a Savitzky-Golay filter [34].

compared to the base case. Note also the non-zero initialization error at time $t = 0$ for all simulation runs utilizing VRM/AVRM diffusion. The "noisy" errors of scatter-based VRM methods are also visible when comparing the different diffusion functions (AVRM/VRM/Tutty).

The error study highlights the differences in performance between merge and remesh compression methods. In general, merging tends to result in lower diffusive shifts for a given compression target, with gentle compression cases not inducing errors beyond the noise amplitude of AVRM. On the other hand, all examined remesh compression levels introduce significant diffusive effects. The differences in performance of the two methods can be attributed to several factors. As noted in Section 5, the remeshing method has no way of adapting to the solution, resulting in a constant-core-size particle patch, as seen in the Lamb-Oseen case in Figure 6. This top-down approach to downsampling results in a much larger loss of resolution than in the case of merging, especially in regions of high vorticity, where the particle cores are much smaller than the domain average. Additionally, differences in performance likely stem from the deviations in the overall particle count (N_1), which can vary from the expected compression target ($N_0 \cdot \mathrm{CR}$). This point is examined further in a later subsection.

In the Lamb-Oseen vortex case, the artificial diffusion can be examined on the entirety of the vorticity field by comparison to the analytical reference. The vorticity field error can be observed to form a rough (noisy) annular pattern indicative of excessive diffusion. Example contours of relative point-wise error for the case of 70% compression can be seen on Figure 10 compared to a reference case. The relative point-wise error of the figure is normalized as follows:

$$\epsilon_{rel}(\boldsymbol{x}) = \frac{\omega(\boldsymbol{x}) - \omega_{\text{ref}}(\boldsymbol{x})}{\max(\omega_{\text{ref}})} \tag{40}$$

The vorticity field errors can also be measured in the global sense with the use of the L^2 integral norm, here taken on a uniform Cartesian grid of spacing h as [17]:

$$\epsilon_{L^2} = h^2 \sqrt{\sum_i |\omega(\boldsymbol{x}_i) - \omega_{ref}(\boldsymbol{x}_i)|^2} \tag{41}$$

where the sum is taken over all nodes of the grid. Note that the unit of the result is m^2/s, representing a circulation error.

The L^2 error was computed in a lattice of $40,000 = 200^2$ points in $-8\sqrt{4\nu t} \leq x, y \leq 8\sqrt{4\nu t}$ using Equations (7) and (36) to resolve the vorticity components. Note the variation of lattice size in simulation time, matching the rate of diffusion of the point vortex. The plot of the L^2 error in simulation time for an example case of $\mathrm{CR} = 70\%$ can be seen on Figure 11 and is generally in agreement with [17]. It is clear that the L^2-error experiences a sharp increase immediately after a compression step and stabilizes shortly after the jump. The integral error in both cases is comparable to the numerical reference counterparts, although some error propagation is visible at $t = 20$s, suggesting a possible long-term influence of compression on accuracy.

7.3 | Adherence to Compression Target

Both of the examined methods rely on estimating the post-compression core size to adhere to the desired compression ratio CR, which predictably results in some deviation from the target. The performance of the two methods can be examined by analyzing the particle count over simulation time. Presented on Figure 12 is the comparison of problem size growth between merging and remeshing compression applied at the same (expected) compression ratio of $\mathrm{CR} = 80\%$. It is clear that both methods tend to excessively compress the vorticity field, with remeshing compression being generally much less accurate than merging, with as much as 20% error in the effective compression ratio, compared to merging's < 3% error.

The inaccuracy of the remeshing scheme can be traced back to Equation (32), which introduced the estimation for the post-compression mesh spacing h. As seen in Figure 12, the approximated spacing is too large, which is likely the result of the particle overlap assumption. As has been reported in the

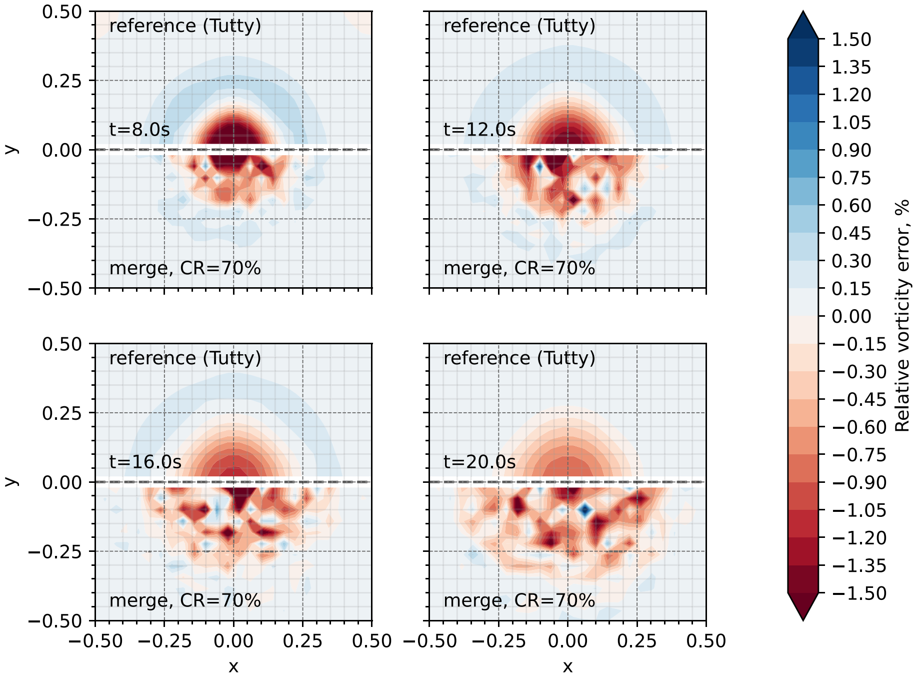


FIGURE 10 | Relative error contours of the Lamb-Oseen vortex case at times $t = [4.0, 8.0, 12.0, 16.0]s$ with a single merge compression step of $CR = 70\%$ at time $t = 10s$ compared to a reference case. [Colour figure can be viewed at wileyonlinelibrary.com]

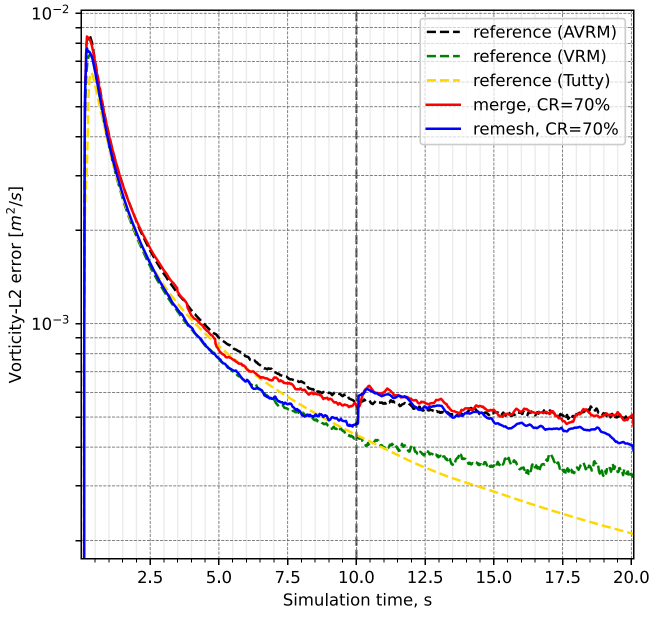


FIGURE 11 | The L^2-measure error of the vorticity field of the Lamb-Oseen vortex compressed to CR = 70% at $t = 10s$ compared to reference cases.

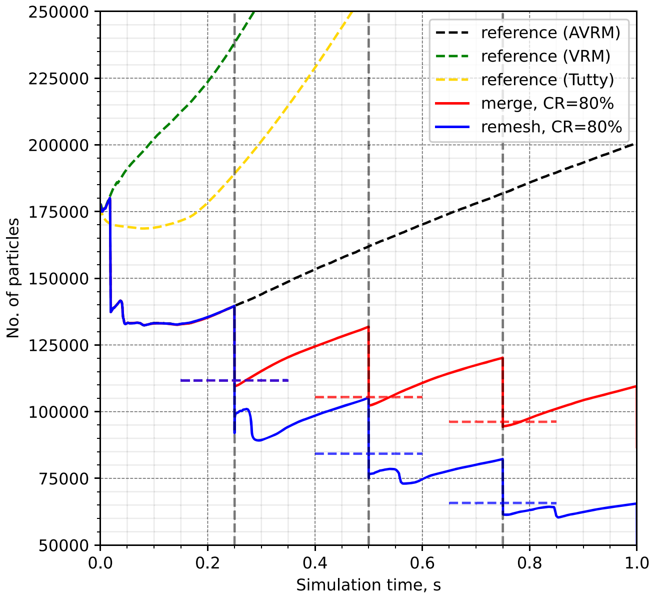


FIGURE 12 | Particle count in simulation time for the vortex dipole case compressed to CR = 80% at times $t = 0.25, 0.50, 0.75,\ 1.00$ s. Dashed horizontal lines represent the expected particle count immediately after the compression steps, color-coded per case.

simulation results, the scattered vortex particles in the AVRM diffusion scheme seldom maintain the desired overlap ratio, and instead can overlap excessively up to a certain threshold. In the current implementation of AVRM, particles are allowed to overlap as much as $\frac{\sigma}{h} = 2.0$ before a merging function is introduced to reduce the overlap. As a result, a substantial amount of particles is allowed excessive overlap, making the approximation from Equation (32) inaccurate. For better compressive accuracy, the mesh spacing equation should be adjusted with a correction factor in the range 0.5–1.0. Crucially, the same inaccuracy does not affect the merge case, despite using a similar approximation in Equation (29). In this case, the original overlap properties are preserved by the nature of the merging function using the same overlap limits as AVRM.

The inaccuracy of the compression level also explains the vast difference in vorticity-representation accuracy between merging and remeshing schemes. Indeed, the effective compression level of remeshing is, in general, around 20% lower than merging. Within the scope of this paper, the examined errors are compared based on the expected compression ratio, which can serve as an a priori prediction for the expected accuracy. Nevertheless, a more representative assessment could be achieved by analyzing cases with comparable *effective* compression.

7.4 | Impact on Computation Time

As designed, the examined downsampling methods achieved significant particle count reductions in both benchmark cases, which, in turn, drastically reduce the total computation time of the simulation runs. Of particular interest is the case of merging with gentle compression CR ≥ 80%, which has been shown not to induce significant errors in the vorticity field in the short term, despite an immense reduction in particle count. Significant results can be seen in the vortex dipole case, where an 80% merge-based compression applied at an interval of 1000 time steps was able to nearly halve the total number of particles after just four applications of compression, with appropriate reductions in total computation time of around 40%. A summary of the results of the vortex dipole case is presented in Table 3, including the absolute error in the zeroth vorticity moment (total circulation) at the final time step.

The impact on computation time is presented in Figures 13 and 14, showing, respectively, the evolution time of a single timestep and the cumulative computation time as functions of simulation time. The effect of compression schemes can be seen clearly in the evolution time, where the typical behavior of gradually increasing time per timestep is reversed, and a significant reduction is visible compared to all reference cases. The time taken by the compression step can also be clearly visible as spikes in evolution time, reaching as high as 40s per timestep on the current hardware. Despite the lack of optimization measures in the current implementation of compression, the time taken by the downsampling routine is small when compared to the cumulative computation time (in the cases of $CR = 80\%$, compression accounted for < 3% of the cumulative computation time). Further attempts at memory optimization and parallelization of the scheme would likely make the computational cost of the compression step negligible when compared to other solver routines.

In the cumulative sense, the compression can also be seen to significantly reduce the scaling of total computation time in Figure 14. Compared to the reference cases, showing a scaling with simulation time T clearly higher than $O(T)$, repeated compression allows for an arbitrarily strict scaling, here clearly lower than $O(T)$. This improvement naturally comes at a cost of increasingly coarse vorticity distribution and representation errors. An

TABLE 3 | Results of the vortex dipole case for different compression methods and reference cases.

Diffusion, Advection	Compression, CR	T_{tot} [s]	N_{final}	$\Delta m_{0,0}$ [m^2/s]
VRM, Direct	N/A, 100%	36,192	549,292	-6.0428×10^{-9}
Tutty, Direct	N/A, 100%	23,077	432,778	-3.1086×10^{-8}
AVRM, Direct	N/A, 100%	9,265	200,636	1.1343×10^{-7}
AVRM, Direct	merge, 90%	6,341	120,856	2.1545×10^{-7}
	—, 80%	5,496	86,699	2.6187×10^{-7}
	—, 70%	4,369	58,669	-1.5211×10^{-6}
	—, 50%	3,089	21,258	8.5583×10^{-8}
	—, 30%	2,339	4,285	-9.4309×10^{-6}
AVRM, Direct	remesh, 90%	4,759	72,564	-4.0154×10^{-6}
	—, 80%	3,902	50,224	-3.0548×10^{-7}
	—, 70%	3,304	32,862	-6.1200×10^{-7}
	—, 50%	2,546	11,226	2.6676×10^{-7}
	—, 30%	2,120	2,043	1.3910×10^{-4}

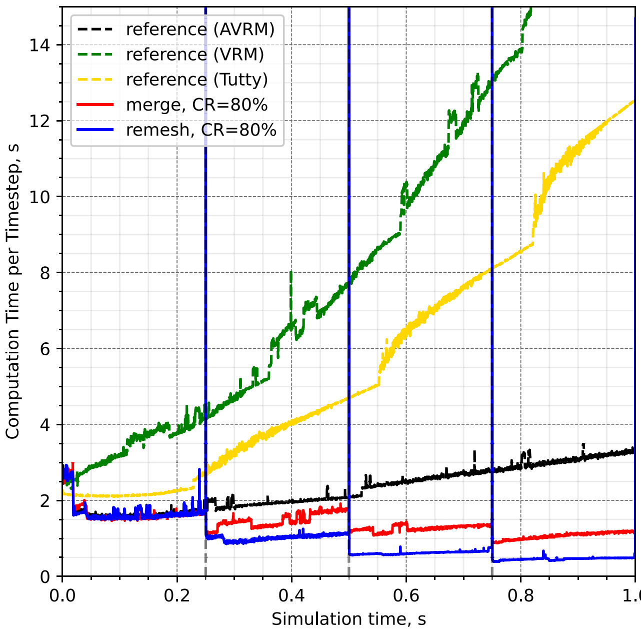


FIGURE 13 | Computation time (per timestep) in simulation time for the vortex dipole case compressed to CR = 80% at times $t = 0.25, 0.5, 0.75$s compared to reference cases.

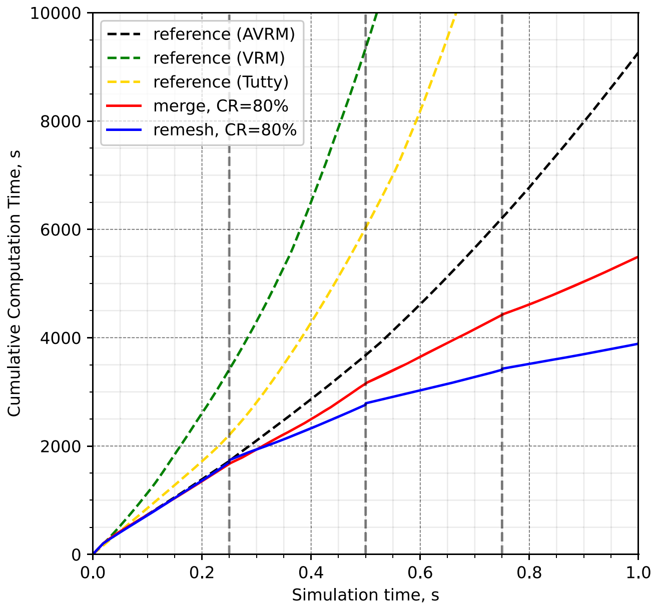


FIGURE 14 | Cumulative computation time in simulation time for the vortex dipole case compressed to CR = 80% at times $t = 0.25, 0.5, 0.75$s compared to reference cases.

example of this effect can be seen immediately following each compression step, where the curve stabilizes at a slope lower than before the compression timestep. This matches the previous observations of persistent effects of downsampling on the solution, as the problem growth rate is reduced following each application of compression.

The impact on performance can be extrapolated to more complex cases, where the effect of compression would likely be even more pronounced due to high particle counts. In particular, similar methods applied to cases of bluff-body wakes show the potential for great performance yields, with periodic compression in the far wake drastically reducing the particle count, and possibly imposing a limit to problem growth altogether. Provided that compression is introduced only in the far wake, the vorticity field errors introduced by these schemes will have a limited effect on the upstream region, allowing for simulation of wake problems with accurately resolved near-body regions, medium-to-low resolution wakes, and more attractive problem-size scaling in simulation time.

8 | Conclusions

Based on VPMs' inherent difficulty in handling a growing problem size, two compression methods have been proposed to downsample the discrete representation of the vorticity field.

The two methods, inspired by existing schemes of regridding and particle merging, are designed with a focus on preserving the lowest-order features of the vorticity field while achieving a desired compression level, here tuned via a parameter CR.

The examined methods have been tested with the use of two benchmark cases of a stationary Lamb-Oseen vortex and an advecting vortex dipole. Compression has been demonstrated to drastically reduce the particle count of both VPM simulations, at a cost of diffusive errors in the vorticity field, with errors in peak vorticity in the range of $\leq 5\%$ for even highly-compressed cases. Of particular interest is compression via merging with compression ratios of at least 80%, in which case the error induced by the merging step is negligible in the short term when compared to other error sources. On the other hand, the remeshing method has been observed to suffer from comparably higher errors and inaccuracies in the effective compression level.

A number of improvements to the examined methods are recommended for effective implementation. Importantly, compression examined in this paper has yet to employ any solution-adaptivity, which could drastically improve the accuracy of the post-compression solution representation. A detailed error study is recommended to relate the problem resolution to the expected representation errors.

Finally, both of the examined methods are yet to be tested in a practical setting, including their design case of a cylinder flow problem. Further study into the use of compression functions in the context of bluff-body wakes and their impact on surface forces is advisable to examine any long-term effects of repeated application of downsampling.

Nomenclature

Symbols:

A	Area
C	Merging candidate set
g	Circulation
h	Mesh spacing, particle nominal separation
m	Moment (of vorticity)
N	No. of particles
Re	Reynolds number
t	Time
u	Velocity
x	Position
$\hat{\mathbf{z}}$	Unit vector in the z-direction
Γ	Circulation
λ	Particle overlap ratio
ν	Kinematic viscosity
σ	Particle core size
$\boldsymbol{\omega}$	Vorticity
$\hat{\cdot}$	The Fourier transform

Subscripts:

p	particle
h	discrete, discretized
∞	free-stream

Acknowledgments

The authors would like to acknowledge the DelftBlue supercomputer facility of Delft University of Technology for providing the computing resources required to carry out the numerical simulations presented in the paper.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The authors have nothing to report.

References

1. G.-H. Cottet and P. D. Koumoutsakos, *Vortex Methods: Theory and Practice* (Cambridge University Press, 2000).

2. J. Pan, C. Ferreira, and A. van Zuijlen, "Estimation of Power Performances and Flow Characteristics for a Savonius Rotor by Vortex Particle Method," *Wind Energy* 26, no. 1 (2023): 76–97.

3. M. Stock, A. Gharakhani, and C. Stone, "Modeling Rotor Wakes With a Hybrid OVERFLOW-Vortex Method on a GPU Cluster," 2010.

4. D.-G. Caprace, P. Chatelain, and G. Winckelmans, "Wakes of Rotorcraft in Advancing Flight: A Large-Eddy Simulation Study," *Physics of Fluids* 32, no. 8 (2020): 087107.

5. J.-L. Guermond, S. Huberson, and W.-Z. Shen, "Simulation of 2D External Viscous Flows by Means of a Domain Decomposition Method," *Journal of Computational Physics* 108, no. 2 (1993): 343–352.

6. G. Wang, Y. Qu, and Y. Li, "A Hybrid Multi-Scale/Finite Element Method in Arbitrary Lagrangian-Eulerian Framework for Predicting Nonlinear Structural-Acoustic Responses of a Large-Deformable Beam in Fluid," *Journal of Sound and Vibration* 577 (2024): 118333.

7. A. Palha, L. Manickathan, C. S. Ferreira, and G. van Bussel, "A Hybrid Eulerian-Lagrangian Flow Solver," 2015.

8. C. Mimeau and I. Mortazavi, "A Review of Vortex Methods and Their Applications: From Creation to Recent Advances," *Fluids* 6, no. 2 (2021): 68.

9. R. Yokota, T. Narumi, R. Sakamaki, S. Kameoka, S. Obi, and K. Yasuoka, "Fast Multipole Methods on a Cluster of GPUs for the Meshless Simulation of Turbulence," *Computer Physics Communications* 180, no. 11 (2009): 2066–2078.

10. S. Shankar and L. Dommelen, "A New Diffusion Procedure for Vortex Methods," *Journal of Computational Physics* 127, no. 1 (1996): 88–109.

11. P. Saffman, *Cambridge Monographs on Mechanics: Vortex Dynamics* (Cambridge University Press, 1995).

12. L. F. Rossi, "Merging Computational Elements in Vortex Simulations," *SIAM Journal on Scientific Computing* 18, no. 4 (1997): 1014–1027.

13. F. M. Besem, J. P. Thomas, R. E. Kielb, and E. H. Dowell, "An Aeroelastic Model for Vortex-Induced Vibrating Cylinders Subject to Frequency Lock-In," *Journal of Fluids and Structures* 61 (2016): 42–59.

14. J. M. Pérez, "Stability Analysis of a Uniform Viscous Transonic Flow Past a Rotating Circular Cylinder," *Results in Engineering* 22 (2024): 102137.

15. S. B. H. Shah and X. Yun Lu, “Numerical Simulation of an Oscillating Flow Past a Circular Cylinder in the Vicinity of a Plane Wall,” *Journal of Hydrodynamics, Series B* 20, no. 5 (2008): 547–552.

16. J. A. Shohat and J. D. Tamarkin, *The Problem of Moments* (American Mathematical Society, 1950).

17. M. J. Stock and A. Gharakhani, “Solution-Responsive Particle Size Adaptivity in Lagrangian Vortex Particle Methods,” in *Proceedings of the ASME 2021 Fluids Engineering Division Summer Meeting (FEDSM2021)*, vol. 1 (2021), V001T02A033.

18. C. Greengard, “The Core Spreading Vortex Method Approximates the Wrong Equation,” *Journal of Computational Physics* 61, no. 2 (1985): 345–348.

19. O. Giannopoulou, A. Colagrossi, A. Di Mascio, and C. Mascia, “Chorin’s Approaches Revisited: Vortex Particle Method vs Finite Volume Method,” *Engineering Analysis With Boundary Elements* 106 (2019): 371–388.

20. M. J. Berger and J. Oliger, “Adaptive Mesh Refinement for Hyperbolic Partial Differential Equations,” *Journal of Computational Physics* 53, no. 3 (1984): 484–512.

21. M. Bergdorf and P. Koumoutsakos, “A Lagrangian Particle-Wavelet Method,” *Multiscale Modeling & Simulation* 5, no. 3 (2006): 980–995.

22. D. Rossinelli, B. Hejazialhosseini, W. van Rees, M. Gazzola, M. Bergdorf, and P. Koumoutsakos, “MRAG-I2D: Multi-Resolution Adapted Grids for Remeshed Vortex Methods on Multicore Architectures,” *Journal of Computational Physics* 288 (2015): 1–18.

23. M. Bergdorf, G.-H. Cottet, and P. Koumoutsakos, “Multilevel Adaptive Particle Methods for Convection-Diffusion Equations,” *Multiscale Modeling & Simulation* 4, no. 1 (2005): 328–357.

24. M. El Ossmani and P. Poncet, “Efficiency of Multiscale Hybrid Grid-Particle Vortex Methods,” *Multiscale Modeling & Simulation* 8, no. 5 (2010): 1671–1690.

25. J. T. Rasmussen, G.-H. Cottet, and J. H. Walther, “A Multiresolution Remeshed Vortex-In-Cell Algorithm Using Patches,” *Journal of Computational Physics* 230, no. 17 (2011): 6742–6755.

26. E. Rossi, D. Durante, S. Marrone, and A. Colagrossi, “A Novel Multi-Resolution Technique for Solving Complex Vorticity Patterns in Planar Viscous Flows Past Bodies Through the DVH Method,” *Computer Methods in Applied Mechanics and Engineering* 396 (2022): 115082.

27. A. J. Chorin, “Numerical Study of Slightly Viscous Flow,” *Journal of Fluid Mechanics* 57, no. 4 (1973): 785–796.

28. O. R. Tutty, “A Simple Redistribution Vortex Method (With Accurate Body Forces),” 2010.

29. R. Pasolari, C. Ferreira, and A. van Zuijlen, “Coupling of OpenFOAM With a Lagrangian Vortex Method for External Aerodynamic Simulations,” *Physics of Fluids* 35 (2023): 107115.

30. J. C. Wu, “Theory for Aerodynamic Force and Moment in Viscous Flows,” *AIAA Journal* 19, no. 4 (1981): 432–441.

31. P. Gehlert, I. Andreu Angulo, and H. Babinsky, “Vortex Force Decomposition-Forces Associated With Individual Elements of a Vorticity Field,” *Experiments in Fluids* 64 (2023): 6.

32. National Research Council, “Vorticity Fields due to Rolling Bodies in a Free Surface–Experiment and Theory,” in *Twenty-First Symposium on Naval Hydrodynamics* (National Academies Press, 1997).

33. Delft High Performance Computing Centre (DHPC), “DelftBlue Supercomputer (Phase 1),” (2022), https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase1.

34. A. Savitzky and M. J. E. Golay, “Smoothing and Differentiation of Data by Simplified Least Squares Procedures,” *Analytical Chemistry* 36, no. 8 (1964): 1627–1639.