

Optimization-based mesh correction with volume and convexity constraints

Marta D’Elia, Denis Ridzal

*Optimization and Uncertainty Quantification,
Sandia National Laboratories¹, MS-1320, Albuquerque, NM 87185-1320, USA*

Kara J. Peterson, Pavel Bochev

*Computational Mathematics,
Sandia National Laboratories MS-1320, Albuquerque, NM 87123, USA*

Mikhail Shashkov

*X-Computational Physics, XCP-4
Los Alamos National Laboratory, Los Alamos, NM 87545, USA*

Abstract

We consider the problem of finding a mesh such that 1) it is the closest, with respect to a suitable metric, to a given source mesh having the same connectivity, and 2) the volumes of its cells match a set of prescribed positive values that are not necessarily equal to the cell volumes in the source mesh. This volume correction problem arises in important simulation contexts, such as satisfying a discrete geometric conservation law and solving transport equations by incremental remapping or similar semi-Lagrangian transport schemes. In this paper we formulate volume correction as a constrained optimization problem. Specifically, the distance to the source mesh defines an optimization objective, while the prescribed cell volumes, mesh validity and/or cell convexity specify the constraints. We solve the optimization problem numerically using a sequential quadratic programming (SQP) method whose performance scales with the mesh size. To achieve scalable performance we develop a specialized multigrid-based preconditioner for optimality systems that arise in the application of the SQP method to the volume correction problem. Numerical examples illustrate the importance of volume correction, and showcase the accuracy, robustness and scalability of our approach.

Keywords: Lagrangian motion, incremental remap, semi-Lagrangian transport, departure volume correction, volume fraction, passive tracer transport

1. Introduction, motivation and background

Mesh motion is at the core of many numerical methods for time dependent flow and structure problems. Examples include Lagrangian hydrodynamics methods [1], in which the mesh evolves with time in order to track deformations of the problem domain, and the related Arbitrary Lagrangian-Eulerian [2] methods, which incorporate a mesh rezoning step to mitigate excessive mesh deformations and mesh tangling. Another important example are semi-Lagrangian methods for advection problems [3, 4], including schemes utilizing incremental remapping [5, 6] as a transport algorithm. Such methods perform a single Lagrangian step and then remap the resulting solution back to a fixed Eulerian mesh.

Email addresses: mdelia@sandia.gov (Marta D’Elia), dridzal@sandia.gov (Denis Ridzal), kjpeter@sandia.gov (Kara J. Peterson), pbbochev@sandia.gov (Pavel Bochev), shashkov@lanl.gov (Mikhail Shashkov)

¹Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy’s National Nuclear Security Administration under contract DE-AC04-94AL85000.

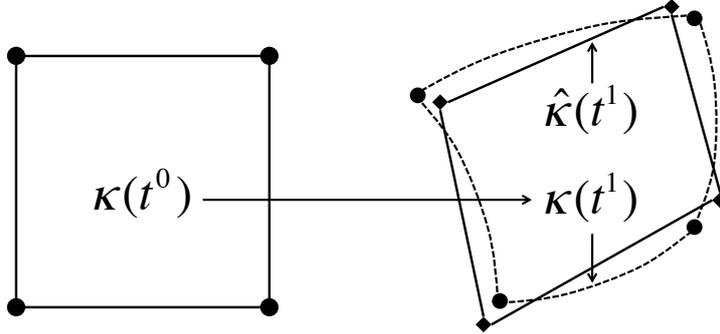


Figure 1: A non-divergent velocity field deforms an initial cell $\kappa(t^0)$ into a cell $\kappa(t^1)$ having the same volume. Approximation of vertex positions by numerical time integration (diamonds) and their subsequent linking by straight lines yields an approximate deformed cell $\hat{\kappa}(t^1)$ with a different volume.

Despite individual differences all such methods require computation of the mesh motion under a given velocity field \mathbf{u} . Typically, most methods approximate the deformed Lagrangian mesh by 1) solving numerically the kinematic relation $d\mathbf{x}/dt = \mathbf{u}$ to estimate the new cell vertex positions, and 2) connecting them by straight lines. Consequently, the cells in the deformed mesh incur two types of errors: one due to the approximation of the vertex trajectories and the second due to the approximation of the deformed cell edges by straight lines. The first kind of error depends on the time stepping and can be viewed as a time discretization error, while the second type can be interpreted as a spatial error.

The temporal and spatial errors combine to produce approximate Lagrangian cells whose volumes do not match the volumes of the exact Lagrangian cells; see Figure 1 for an illustration. A notable feature of numerical methods relying on mesh motion is that volume approximation errors may have an outsized influence on their accuracy. In particular, even if the temporal and spatial errors in the approximation of the Lagrangian cell are better than first-order accurate, the resulting scheme may fail to preserve constant solutions of the system.

To understand this phenomenon let us examine the Lagrangian motion of a single mesh cell $\kappa(t)$, representing a fluid parcel with material density ρ , under a non-divergent velocity field \mathbf{u} . Let $\kappa(t^0)$ denote the initial position of the fluid parcel and $\kappa(t)$ be the exact Lagrangian parcel at time $t > t^0$, and

$$|\kappa(t)| = \int_{\kappa(t)} dV \quad \text{and} \quad M_{\kappa}(t) = \int_{\kappa(t)} \rho dV.$$

Because non-divergent Lagrangian flows preserve volume and mass there holds

$$\frac{d}{dt}|\kappa(t)| = 0 \quad \text{and} \quad \frac{d}{dt}M_{\kappa}(t) = 0. \quad (1)$$

Suppose now that $\rho = \text{const}$ and $\kappa(t)$ moves from its initial position at time t^0 to a new position at time $t^1 = t^0 + \Delta t$. Owing to (1)

$$\rho(t^1) = \frac{M_{\kappa}(t^1)}{|\kappa(t^1)|} = \frac{M_{\kappa}(t^0)}{|\kappa(t^0)|} = \rho(t^0) = \text{const}.$$

On the other hand, since $\kappa(t^1)$ is not known exactly, except for some trivial velocity fields, a numerical scheme uses an approximation $\hat{\kappa}(t^1)$ for which $|\hat{\kappa}(t^1)| \neq |\kappa(t^1)|$; see Figure 1. As a result, the material state $\hat{\rho}$ in the approximate Lagrangian fluid parcel $\hat{\kappa}(t^1)$ is given by

$$\hat{\rho}(t^1) = \frac{M_{\kappa}(t^1)}{|\hat{\kappa}(t^1)|} \neq \frac{M_{\kappa}(t^0)}{|\kappa(t^0)|} = \rho(t^0). \quad (2)$$

In other words, the constant density is not preserved. Yet, many Lagrangian and semi-Lagrangian methods rely on relationships such as (2) to approximate densities, tracer mixing ratios, material volume fractions

and other quantities of interest. This effectively renders such methods unable to represent exactly *constant* solutions despite the fact that $\widehat{\kappa}(t^1)$ can be an accurate approximation of the exact fluid parcel $\kappa(t^1)$.

This example illustrates the consequences of violating the broader notion of a *Discrete Geometric Conservation Law* [7] (D-GCL) for a given numerical scheme. D-GCL stipulates that computation of geometric quantities in the deformed mesh configuration must be done in a way that allows the associated numerical scheme to reproduce constant states for any mesh motion.

One possible pathway for achieving a D-GCL for (2) is to enforce equality of deformed cell volumes at all time steps to their initial volumes in the undeformed configuration. This can be done by further deforming $\widehat{\kappa}(t^n)$ to a *corrected* cell $\kappa^*(t^n)$ such that

$$|\kappa^*(t^n)| = |\kappa(t^0)| \quad \forall n. \quad (3)$$

The corrected cell must be valid and can be subject to further mesh quality conditions such as convexity. In addition, we require that $\kappa^*(t^n)$ remain close to $\widehat{\kappa}(t^n)$ at all time steps, because the latter serves as a proxy for the unknown exact Lagrangian cell $\kappa(t^n)$. We refer to the task of finding a volume-corrected cell $\kappa^*(t^n)$ for every cell $\widehat{\kappa}(t^n)$ in a given deformed mesh as the *volume correction* problem.

This paper solves the volume correction problem by couching it in a constrained optimization formulation. The idea is to seek a mesh whose cells $\kappa^*(t^n)$ minimize the distance to the cells $\widehat{\kappa}(t^n)$ of the source mesh, while satisfying constraints expressing the volume condition (3) and appropriate mesh quality conditions. The optimization variables are the mesh vertices, i.e., the formulation does not require additional mesh points as some of the approaches reviewed below. The resulting optimization formulation of the volume correction problem is a nonlinear programming problem (NLP) with an equality constraint corresponding to (3) and possible additional inequality and/or equality constraints expressing cell quality conditions. To solve the NLP efficiently we develop a problem-specific preconditioner for the inexact trust-region sequential quadratic programming (SQP) method of [8]. The preconditioner is based on a reduction of the optimality systems arising in trust-region SQP to a matrix equation with the structure of a discrete Laplacian, which can be solved using algebraic multigrid methods.

Existing work on discrete geometric conservation laws can be broadly divided into three categories. The first category comprises methods which alleviate but do not fully eliminate the violation of the relevant D-GCL. Such methods typically use additional Lagrangian points to compute more accurate approximations of the deformed cells. For example, the semi-Lagrangian CSLAM method in [9] can evolve up to three additional points per quadrilateral side. CSLAM connects the additional points by straight line segments to obtain a better polygonal approximation of the exact boundary of the true Lagrangian cell.

The second category includes works that view the D-GCL as a constraint on the numerical scheme. These approaches formulate a set of conditions that must be satisfied by the equations computing the vertex positions, and various surface and volume integrals; see, e.g., [10] and [11, 12]. For instance, the D-GCL in [11] requires the change in cell volumes between two time steps to equal the volumes of the areas swept by the cell boundaries. This leads to a particular choice of integration rules for calculating force integrals and updating vertex positions.

The third category of methods [13, 14, 15, 16] views the D-GCL as a constraint on the mesh, i.e., they solve some form of the volume correction problem stated earlier. For instance, [13] address this problem for potential flows by using a heuristic adjustment procedure to perturb the mesh cells until they satisfy an appropriate volume condition. This is done in three stages by first adjusting regions of elements until the entire region has the correct volume, then repeating the adjustment for “rings” of elements, and finally proceeding to adjust the volumes of the individual cells in each ring. Cell adjustment is performed by sliding cell vertices along the characteristics, which prevents introducing bias in the direction of the flow. For non-divergent flows [14] reduces this complex heuristic to a simple element-by-element volume adjustment, leaving the vertices unperturbed and correcting only element midpoints. For logically Cartesian meshes [14] recommends adjusting the elements in a row-wise, column-wise or staircase-like pattern. However, there are no theoretical assurances that the adjustment will terminate and/or yield a valid mesh in general configurations.

The Monge-Ampere (MA) trajectory correction [17, 18] is another approach that solves a version of the volume correction problem for incompressible fluids. The idea of MA correction is to enforce volume equality

implicitly by enforcing the condition $J^{-1} = 1$ where J is the flow Jacobian [17, p. 180]. This is accomplished by correcting the numerical path-mean velocities with a gradient of a scalar potential. The scalar potential satisfies a form of the Monge-Ampere equation (MAE), thus the term Monge-Ampere trajectory correction. We note that the MA trajectory correction recovers the volume constraint (3) exactly only when closed-form analytic solutions of the MAE are available, as in [18]. In practice this is rarely the case, and the MA trajectory correction approach trades volume error for the spatial discretization error incurred in the solution of the MAE, as illustrated in [17, Table 1].

Our optimization-based solution of the volume correction problem falls in the third category of methods aiming to achieve the D-GCL. As such, it is less intrusive than the methods belonging to the second category, which require special problem discretizations. Specifically, our method can be viewed as “postprocessing” of the mesh motion and is easily incorporated into existing implementations of Lagrangian schemes.

To the best of our knowledge our method is also the first systematic application of NLP formulations and large-scale optimization algorithms in the context of discrete geometric conservation laws. Casting the volume correction problem as an optimization problem offers valuable theoretical and computational advantages. First, provided the problem has a solution, the computed NLP solution *provably satisfies the volume constraint* (3) in exact arithmetic, and to near machine precision in practice. This is an important advantage over the methods from the first category and the trajectory correction scheme based on numerically solving the MAE, [17].

Second, the NLP solution is *locally optimal*; if the original deformed mesh is chosen as the initial guess for the NLP solver (in addition to being part of the objective function definition), then the NLP solution represents a constraint-consistent mesh that is in some sense closest to the original constraint-inconsistent mesh. The latter carries important information about the mesh motion, and so the local solution of the NLP provides good balance between maintaining accurate cell volumes, which we view as a hard constraint, and maintaining the overall trajectories of cell vertices. In fact, the numerical results indicate that, as a side effect, our method *greatly improves the accuracy of vertex trajectories* for Lagrangian mesh motion, with respect to the exact trajectories.

Third, our approach is based on a *globally convergent* optimization algorithm that exhibits fast Newton convergence near the solution and admits *scalable* linear solvers. The numerical tests demonstrate that the optimization algorithm converges in only a handful of iterations, while its computational cost scales linearly with the mesh size owing to the devised multigrid-based preconditioner.

Fourth, the optimization approach enables a *robust and systematic implementation* of volume correction without the need to trace back the flow field point by point (or region by region). In contrast, the methods based on heuristic cell update rules, [13, 15, 16], may encounter intricate update scenarios, for example when multiple trace-back layers intersect, resulting in loss of automation [13, p. 2010]. Additionally, the implementation of such methods can be challenging, especially in three dimensions [13, p. 2003].

Finally, we point out that the optimization framework enables the *enforcement of additional constraints*, such as cell validity and cell convexity, and may include additional mesh-quality metrics, all of which would be difficult to incorporate rigorously in any of the previously discussed schemes. In fact, to the best of our knowledge, our approach is the first to guarantee cell convexity in the context of volume correction.

We consider two representative simulation settings to illustrate the potential of the new optimization-based volume correction algorithm. In the first one we use the algorithm to augment a standard semi-Lagrangian incremental remap transport scheme [6] to a *volume-corrected* scheme. Numerical solution of a passive tracer transport problem with multiple tracers reveals that the corrected scheme yields tracer volumes accurate to a machine precision, which is a significant improvement over the original uncorrected scheme. Besides the accuracy gains, this setting illustrates the ease with which the optimization-based volume correction can be incorporated in an existing scheme.

The second setting uses Lagrangian mesh motion to show that in addition to enforcing the desired cell volumes, our algorithm also brings about significant ancillary enhancements in the mesh geometry. Aside from the aforementioned improvement in the vertex trajectories we observe a very close alignment between the barycenters of the true and corrected meshes, and recovery of convex mesh cells from non-convex source meshes. These results suggest that our algorithm provides a flexible platform to address multiple mesh quality tasks extending beyond only correcting cell volumes.

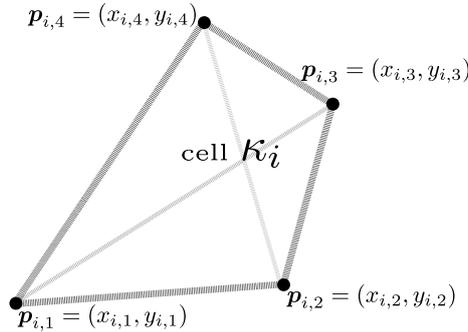


Figure 2: Quadrilateral cell κ_i , points $P(\kappa_i)$ and decomposition in two pairs of triangles.

We have arranged the paper as follows. Section 2 introduces the relevant notation, gives a formal statement of the volume correction problem and transforms it into a constrained optimization problem. Section 3 develops an optimization algorithm specifically tailored to the volume correction problem. Section 4 examines two different application contexts for the volume correction algorithm and Section 5 presents the corresponding numerical results.

2. The volume correction problem

This section gives a formal statement of the volume correction problem and then presents its transformation into a constrained optimization problem. We start by reviewing notation and mesh nomenclature. To avoid notational and technical complications that would diminish the clarity of the presentation, we focus on two space dimensions and meshes comprising quadrilateral cells with straight sides. Section 2.4 briefly discusses the volume correction problem and its reformulation into an optimization problem for general meshes.

2.1. Notation

We assume that $\Omega \subset \mathbb{R}^2$ is a simply connected domain with polygonal boundary $\Gamma = \partial\Omega$. Bold face fonts denote vector quantities and standard fonts are scalar quantities. Depending on the context the symbol $|\cdot|$ can be the Euclidean length of a vector in \mathbb{R}^k , i.e., $|\mathbf{q}|^2 = \sum_{i=1}^k \mathbf{q}_i^2$ for $\mathbf{q} \in \mathbb{R}^k$, the cardinality of a finite set, or the oriented measure of some bounded region.

Some mesh nomenclature germane to this paper follows. Except when noted otherwise, $K(\Omega)$ denotes a conforming partition [19] of Ω into quadrilateral cells κ_i , $i = 1, \dots, m$, where $m = |K(\Omega)|$. A cell κ_i is *valid* if it is star-shaped and a mesh $K(\Omega)$ is valid if all of its cells are valid. Note that valid cells have nonzero oriented volumes, i.e., $|\kappa_i| \neq 0$. The symbol $P(\Omega)$ stands for the set of all mesh vertices $\mathbf{p}_j = (x_j, y_j)$, $j = 1, \dots, p$, where $p = |P(\Omega)|$. The vector of all vertex coordinates is $\mathbf{p} \in \mathbb{R}^{2p}$. The set of all mesh vertices belonging to the boundary Γ is $P(\Gamma)$, and $P(\kappa_i) = \{\mathbf{p}_{i,1}, \dots, \mathbf{p}_{i,4}\}$ are the vertices belonging to a quadrilateral $\kappa_i \in K(\Omega)$; see Figure 2. Without loss of generality we can assume that $P(\kappa_i)$ are ordered counterclockwise and so every cell in a valid mesh has positive oriented volume, i.e., $|\kappa_i| > 0$ for $i = 1, \dots, m$.

We also need the cell volume function $\mathbf{c} : K(\Omega) \rightarrow \mathbb{R}^m$ such that its i th component corresponds to the volume of κ_i , i.e.

$$\mathbf{c}_i(K(\Omega)) = |\kappa_i|, \quad i = 1, \dots, m, \quad (4)$$

and the convexity indicator function $\mathcal{I} : K(\Omega) \rightarrow \mathbb{R}^m$ such that

$$\mathcal{I}_i(K(\Omega)) = \begin{cases} 1 & \text{if } \kappa_i \text{ is convex} \\ 0 & \text{if } \kappa_i \text{ is nonconvex.} \end{cases} \quad (5)$$

We recall that for quadrilateral cells

$$\mathbf{c}_i(K(\Omega)) = \frac{1}{2}((x_{i,1} - x_{i,3})(y_{i,2} - y_{i,4}) + (x_{i,2} - x_{i,4})(y_{i,3} - y_{i,1})).$$

A convenient way to express validity and define a convexity indicator for a quadrilateral cell κ_i is by splitting it into two pairs of triangles obtained by connecting the pairs of its opposing vertices; see Figure 2. For $r = 1, \dots, 4$ the triangle $\tau_{i,r}$ has vertices $(\mathbf{p}_{a_r}, \mathbf{p}_{b_r}, \mathbf{p}_{c_r})$ where

$$(a_r, b_r, c_r) = \begin{cases} (1, 2, 4) & r = 1 \\ (2, 3, 4) & r = 2 \\ (1, 3, 4) & r = 3 \\ (1, 2, 3) & r = 4. \end{cases} \quad (6)$$

For $r = 1, \dots, 4$ let $\mathbf{t}^r : K(\Omega) \rightarrow \mathbb{R}^m$ be the function that gives the oriented volume of the r th triangle in cell κ_i , that is,

$$\mathbf{t}_i^r(K(\Omega)) = |\tau_{i,r}|. \quad (7)$$

In terms of cell vertices these functions are given by

$$\mathbf{t}_i^r(K(\Omega)) = \frac{1}{2}(x_{i,a_r}(y_{i,c_r} - y_{i,b_r}) - x_{i,b_r}(y_{i,a_r} - y_{i,c_r}) - x_{i,c_r}(y_{i,b_r} - y_{i,a_r})). \quad (8)$$

The following conditions for validity and convexity of quadrilateral cells are straightforward.

Validity. A cell $\kappa_i \in K(\Omega)$ is valid if the oriented volumes of at least one pair of its triangles are positive, specifically

$$\forall i = 1, \dots, m \quad \mathbf{t}_i^r(K(\Omega)) > 0 \quad \text{for } r = 1, 2 \text{ or } r = 3, 4. \quad (9)$$

Convexity. A cell $\kappa_i \in K(\Omega)$ is convex if the oriented volumes of all its triangles are positive, i.e.,

$$\forall i = 1, \dots, m \quad \mathbf{t}_i^r(K(\Omega)) > 0 \quad \text{for } r = 1, \dots, 4. \quad (10)$$

Thus, for quadrilateral cells we can set

$$\mathcal{I}_i(K(\Omega)) = \begin{cases} 1 & \text{if } \mathbf{t}_i^r(K(\Omega)) > 0 \quad \text{for } r = 1, \dots, 4 \\ 0 & \text{if } \mathbf{t}_i^r(K(\Omega)) \leq 0 \quad \text{for some } 1 \leq r \leq 4. \end{cases}$$

Suppose that $\tilde{K}(\Omega)$ is another cell partition of Ω with the same connectivity as $K(\Omega)$. We distinguish the entities associated with this mesh by using the tilde accent, i.e., $\tilde{\kappa}_i$ are the cells in $\tilde{K}(\Omega)$, $\tilde{\mathbf{p}}_j$ are the vertices in this mesh, $\tilde{P}(\Gamma)$ are the vertices on the boundary and so on. Without loss of generality we may assume that the cells and the mesh vertices in $\tilde{K}(\Omega)$ are numbered in the same order as in $K(\Omega)$. In other words, if $\kappa_i \in K(\Omega)$ has vertices $\{\mathbf{p}_{i_1}, \mathbf{p}_{i_2}, \mathbf{p}_{i_3}, \mathbf{p}_{i_4}\}$, then $\tilde{\kappa}_i \in \tilde{K}(\Omega)$ has vertices $\{\tilde{\mathbf{p}}_{i_1}, \tilde{\mathbf{p}}_{i_2}, \tilde{\mathbf{p}}_{i_3}, \tilde{\mathbf{p}}_{i_4}\}$. With this assumption, define the distance between $K(\Omega)$ and $\tilde{K}(\Omega)$ according to

$$d(K(\Omega), \tilde{K}(\Omega)) = \left(\sum_{j=1}^p |\mathbf{p}_j - \tilde{\mathbf{p}}_j|^2 \right)^{\frac{1}{2}} = |\mathbf{p} - \tilde{\mathbf{p}}|. \quad (11)$$

2.2. Statement of the volume correction problem

Consider a pair $\{\tilde{K}(\Omega), \mathbf{c}_0\}$ where $\tilde{K}(\Omega)$ is a given mesh and $\mathbf{c}_0 \in \mathbb{R}^m$ is such that

$$\sum_{i=1}^m c_{0,i} = |\Omega| \quad \text{and} \quad c_{0,i} > 0 \quad \forall i = 1, \dots, m. \quad (12)$$

The volume correction problem seeks a mesh $K(\Omega)$, which has the same connectivity as $\tilde{K}(\Omega)$ and is the closest, with respect to (11), mesh to $\tilde{K}(\Omega)$ satisfying the following conditions:

- (a) the volumes of its cells match the values prescribed in \mathbf{c}_0 ; and
- (b) every cell $\kappa_i \in K(\Omega)$ is valid; or
- (c) every cell $\kappa_i \in K(\Omega)$ is convex; and
- (d) boundary points in $K(\Omega)$ correspond to boundary points in $\tilde{K}(\Omega)$.

In what follows we refer to $\tilde{K}(\Omega)$ as the *source* mesh, to the set \mathbf{c}_0 as the vector of *desired cell volumes*, and to $K(\Omega)$ as the *volume compliant mesh*.

Depending on a number of factors the volume correction problem may or may not have a solution. One important setting in which solutions always exist arises when the source mesh $\tilde{K}(\Omega)$ is a transformation of another mesh $\check{K}(\Omega)$, having convex cells, and $c_{0,i} = |\check{\kappa}_i|$. In this case $K(\Omega) = \check{K}(\Omega)$ is a trivial solution of the volume correction problem. Precise qualification of sufficient existence conditions for solutions of the volume correction problem is beyond the scope of this paper and so in the following we always assume that this problem has a solution.

2.3. Volume correction as an optimization problem

The functions \mathbf{c} , \mathcal{I} , and \mathbf{t}^r depend on the mesh connectivity and the mesh vertices \mathbf{p} . Assuming that the mesh connectivity is fixed, it is convenient to view these functions as mappings $\mathbb{R}^{2p} \rightarrow \mathbb{R}^m$. Thus, in what follows we use the notation $\mathbf{c}(\mathbf{p})$, $\mathcal{I}(\mathbf{p})$, and $\mathbf{t}^r(\mathbf{p})$.

In order to formulate the volume correction problem as a constrained optimization problem we consider the distance (11) between $K(\Omega)$ and $\tilde{K}(\Omega)$ as an *optimization objective*, while conditions (a)–(d) on the volume compliant mesh provide the *optimization constraints*.

Condition (a) defines an *equality* constraint

$$\mathbf{c}(\mathbf{p}) = \mathbf{c}_0, \quad (13)$$

which can be expressed as

$$\mathbf{C}(\mathbf{p}) = \mathbf{0}, \quad (14)$$

where $\mathbf{C}(\mathbf{p}) : \mathbb{R}^{2p} \rightarrow \mathbb{R}^m$ is the function

$$\mathbf{C}(\mathbf{p}) = \mathbf{c}(\mathbf{p}) - \mathbf{c}_0. \quad (15)$$

On the other hand, (9) and (10) imply that (b) and (c) can be enforced through *inequality* constraints. Since convexity implies validity, to streamline the presentation we focus on enforcing only (10). To express the last condition as an optimization constraint note that (d) is equivalent to

$$\tilde{\mathbf{p}}_j \in \tilde{P}(\Gamma) \Rightarrow \mathbf{p}_j \in P(\Gamma). \quad (16)$$

Let $\gamma : \mathbb{R}^2 \rightarrow \mathbb{R}$ be such that $\gamma(\tilde{\mathbf{p}}_j) = 0$ for $\tilde{\mathbf{p}}_j \in \tilde{P}(\Gamma)$ and define the set

$$\mathcal{S}_\Gamma = \{j : \tilde{\mathbf{p}}_j \in \tilde{P}(\Gamma)\}$$

containing the indices of all boundary points. Because we assume that the volume compliant mesh and the source mesh have the same connectivity, it follows that (16) is equivalent to the equality constraint

$$\gamma(\mathbf{p}_j) = 0 \quad \forall j \in \mathcal{S}_\Gamma. \quad (17)$$

Finally, we define a cost functional $J_0(\mathbf{p}) : \mathbb{R}^{2p} \rightarrow \mathbb{R}$ as

$$J_0(\mathbf{p}) = \frac{1}{2}d(K(\Omega), \tilde{K}(\Omega))^2 = \frac{1}{2}|\mathbf{p} - \tilde{\mathbf{p}}|^2. \quad (18)$$

The following optimization formulation then solves the volume correction problem: given a pair $\{\tilde{K}(\Omega), \mathbf{c}_0\}$ find $K^*(\Omega)$ with vertices \mathbf{p}^* such that

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \{J_0(\mathbf{p}) \text{ subject to (14), (10), and (17)}\}. \quad (19)$$

A simplified optimization formulation. We consider several modifications to the general formulation (19) that can be used to improve its efficiency. First, note that in many practical use cases the condition (17) can be subsumed in the equality constraint (14) in a structure-preserving manner. One such use case is that of rectangular domains Ω , where (17) can be enforced by eliminating the x -coordinates of the vertical-boundary variables from the optimization formulation, and likewise eliminating the y -coordinates of the horizontal-boundary variables. Second, we can enforce the convexity condition (10) *weakly* by using a logarithmic barrier function to penalize the objective (18). These considerations allow us to drop all but the equality constraint (14) and replace (19) by the following simplified equality constrained formulation:

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \{ J(\mathbf{p}) \text{ subject to } \mathbf{C}(\mathbf{p}) = \mathbf{0} \}, \quad (20)$$

where

$$J(\mathbf{p}) = J_0(\mathbf{p}) - \beta \sum_{i=1}^m \sum_{r=1}^4 \log t_i^r(\mathbf{p}), \quad (21)$$

and $\beta > 0$ is a mesh-dependent penalization parameter. The use of the logarithmic barrier penalty significantly reduces the occurrence of negative triangle areas. Moreover, for several examples it is possible to obtain fully convex meshes at optimality, i.e., meshes that rigorously satisfy condition (10).

2.4. Formulation for other cell types

The statement of the volume correction problem in Section 2.2 is not limited to quadrilateral meshes. It applies to general source meshes $\tilde{K}(\Omega)$ in two and three-dimensions comprising different types of cells, including, e.g., polygonal and polyhedral cells. Application of the optimization formulation (19) to solve the corresponding volume correction problem then boils down to proper definition of the constraints expressing convexity and/or validity of the particular cell type. For general polygons and polyhedra one may invoke some of the ideas found in e.g., [20]. A thorough discussion of all possible mesh configurations is clearly beyond the scope of this paper.

However, specialization of (19) to simplicial cell partitions in two and three-dimensions is particularly straightforward and we briefly explain the formulation. Indeed, suppose that $K(\Omega)$ is a conforming partition of Ω into simplicial cells τ_i . Since a valid triangle or a tetrahedron is always convex, i.e., (b) implies (c), it follows that we only need to formulate a constraint expressing (b). Recall that a triangle or a tetrahedron is valid if and only if its volume is positive, and so the inequality constraint

$$\mathbf{c}(\mathbf{p}) > \mathbf{0}, \quad (22)$$

is sufficient to enforce both (b) and (c). On the other hand, the desired cell volumes specified in \mathbf{c}_0 are already positive and so, the equality constraint $\mathbf{C}(\mathbf{p}) = \mathbf{0}$ implies $\mathbf{c}(\mathbf{p}) = \mathbf{c}_0 > \mathbf{0}$, i.e., it ensures cell validity independently of (22), thereby rendering it redundant. It follows that for triangular and tetrahedral meshes the optimization formulation of the volume correction problem (19) specializes to

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \{ J_0(\mathbf{p}) \text{ subject to } \mathbf{C}(\mathbf{p}) = \mathbf{0} \}. \quad (23)$$

For a triangle τ_i with vertices $P(\tau_i) = (\mathbf{p}_{i,1}, \mathbf{p}_{i,2}, \mathbf{p}_{i,3})$ the function $\mathbf{c}(\mathbf{p})$ is simply the function from (8):

$$\mathbf{c}_i(K(\Omega)) = \frac{1}{2} (x_{i,1}(y_{i,3} - y_{i,2}) - x_{i,2}(y_{i,1} - y_{i,3}) - x_{i,3}(y_{i,2} - y_{i,1})).$$

For a tetrahedron τ_i with vertices $P(\tau_i) = \{\mathbf{p}_{i,1}, \dots, \mathbf{p}_{i,4}\}$ and edges $E(\tau_i) = \{\mathbf{e}_{i,1}, \dots, \mathbf{e}_{i,4}\}$, assume that the vertices and the edges are numbered as in Figure 3. Then, the triple product of any three edges sharing a vertex gives the volume $|\tau_i|$. For example, using the edges sharing $\mathbf{p}_{i,1}$ we have the following definition of the cell volume function:

$$\mathbf{c}_i(K(\Omega)) = \frac{1}{6} |\mathbf{e}_{i,1} \cdot (\mathbf{e}_{i,3} \times \mathbf{e}_{i,4})|, \quad i = 1, \dots, m. \quad (24)$$

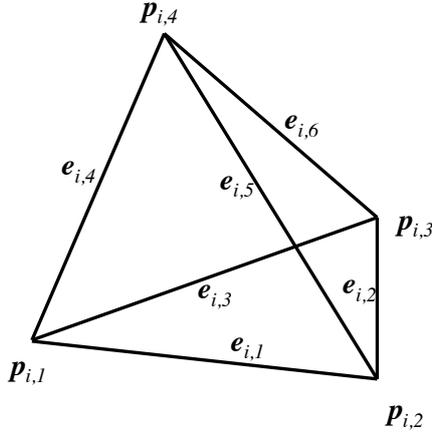


Figure 3: Tetrahedron τ_i , points $P(\tau_i)$ and edges $E(\tau_i)$.

3. Description of the algorithm

In this section we present an optimization algorithm for the numerical computation of

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} \{J(\mathbf{p}) \text{ subject to } \mathbf{C}(\mathbf{p}) = \mathbf{0}\}.$$

Based on the problem characteristics given in Sections 1 and 2, the general design requirements for the algorithm are as follows:

- The algorithm must handle nonlinear objective functions J and nonlinear constraints² \mathbf{C} ;
- It must apply to large computational meshes, where $|K(\Omega)| \approx \mathcal{O}(10^6)$ in two dimensions and $|K(\Omega)| \approx \mathcal{O}(10^9)$ in three dimensions; and
- Its computational performance must scale with $|K(\Omega)|$; specifically, the algorithm must accommodate scalable iterative linear system solvers.

We use the inexact trust-region sequential quadratic programming (SQP) method of Heinkenschloss and Ridzal [8] as the starting point. Alternatives include inexact SQP methods described in [21, 22]. The two key properties of inexact SQP methods are the fast local convergence, based on their relationship to Newton’s method, and the flexibility to use iterative (‘inexact’) linear systems solvers, enabling an efficient solution of very large nonlinear optimization problems. It is important to note that the algorithms described in [8, 21, 22] only provide a general optimization framework; the design of efficient linear system solvers is left to the user. In the following we describe the structure of linear systems arising in [8] and propose a scalable method for their solution.

Let $\mathcal{L} : \mathbb{R}^{2p} \times \mathbb{R}^m \rightarrow \mathbb{R}$, $\mathcal{L}(\mathbf{p}, \boldsymbol{\lambda}) = J(\mathbf{p}) + \boldsymbol{\lambda}^T \mathbf{C}(\mathbf{p})$, be the Lagrangian for (20). Below we use the symbol ∇ to denote gradients and Jacobians with respect to the variable \mathbf{p} . Let \mathbf{p}^k be the k -th SQP iterate and $\boldsymbol{\lambda}^k$ the Lagrange multiplier estimate at \mathbf{p}^k . Let $H^k = H(\mathbf{p}^k, \boldsymbol{\lambda}^k)$ be a symmetric approximation of the Hessian $\nabla^2 \mathcal{L}(\mathbf{p}^k, \boldsymbol{\lambda}^k)$ of the Lagrangian. Trust-region SQP methods solve (20) by solving a sequence of convex and

²In general, the equality constraints in (20) are polynomial functions in terms of the cell vertex coordinates, expressing the cell areas. For quadrilateral cells these constraints are bilinear functions of the coordinates. Among other things this implies that using, e.g., finite difference of the Jacobian to approximate the Hessian yields the exact Hessian operator. However, besides this fact we are not aware of any algorithms specialized for such kind of constraints and so we approach the algorithm design as we would for general nonlinear constraints.

non-convex subproblems derived from

$$\underset{\mathbf{s}}{\text{minimize}} \quad \frac{1}{2} \mathbf{s}^T H^k \mathbf{s} + \mathbf{s}^T \nabla \mathcal{L}(\mathbf{p}^k, \boldsymbol{\lambda}^k) + \mathcal{L}(\mathbf{p}^k, \boldsymbol{\lambda}^k) \quad (25a)$$

$$\text{subject to} \quad \nabla \mathbf{C}(\mathbf{p}^k) \mathbf{s} + \mathbf{C}(\mathbf{p}^k) = 0, \quad \|\mathbf{s}\| \leq \Delta^k. \quad (25b)$$

To solve (25) the method in [8] uses a Byrd-Omojokun composite-step approach. The trial step \mathbf{s}^k (the approximate solution of (25)) is computed as the sum of a quasi-normal step \mathbf{n}^k and a tangential step \mathbf{t}^k . The quasi-normal step \mathbf{n}^k reduces linear infeasibility by approximately solving

$$\mathbf{n}^k = \underset{\mathbf{n}}{\text{argmin}} \{ \|\nabla \mathbf{C}(\mathbf{p}^k) \mathbf{n} + \mathbf{C}(\mathbf{p}^k)\|^2 \quad \text{subject to} \quad \|\mathbf{n}\| \leq \zeta \Delta^k \}, \quad (26)$$

where $\zeta \in (0, 1)$ is a fixed constant, using a modified Powell's dogleg method. Subsequently, the tangential step \mathbf{t}^k is computed as an approximate solution of the subproblem

$$\mathbf{t}^k = \underset{\mathbf{t}}{\text{argmin}} \left\{ \begin{array}{l} \frac{1}{2} (\mathbf{t} + \mathbf{n}^k)^T H^k (\mathbf{t} + \mathbf{n}^k) + (\mathbf{t} + \mathbf{n}^k)^T \nabla \mathcal{L}(\mathbf{p}^k, \boldsymbol{\lambda}^k) + \mathcal{L}(\mathbf{p}^k, \boldsymbol{\lambda}^k) \\ \text{subject to} \quad \nabla \mathbf{C}(\mathbf{p}^k) \mathbf{t} = 0, \quad \|\mathbf{t} + \mathbf{n}^k\| \leq \Delta^k \end{array} \right\}, \quad (27)$$

using a variant of the Steihaug-Toint projected conjugate gradient (CG) iteration. The Lagrange multiplier $\boldsymbol{\lambda}^k$ is estimated by approximately solving

$$\boldsymbol{\lambda}^k = \underset{\boldsymbol{\lambda}}{\text{argmin}} \quad \|\nabla J(\mathbf{p}^k) + \nabla \mathbf{C}(\mathbf{p}^k)^T \boldsymbol{\lambda}\|. \quad (28)$$

Each of the steps (26), (27) and (28) involves the solution of special saddle-point systems, called *augmented systems*. They are used to compute: 1) a Newton direction for the quasi-normal step; 2) the constraint null-space projections in the projected Steihaug-Toint CG method; and 3) the Lagrange multiplier. For details, see [8]. Given an optimization iterate \mathbf{p}^k all systems are of the form

$$\begin{pmatrix} I & \nabla \mathbf{C}(\mathbf{p}^k)^T \\ \nabla \mathbf{C}(\mathbf{p}^k) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{v}^1 \\ \mathbf{v}^2 \end{pmatrix} = \begin{pmatrix} \mathbf{b}^1 \\ \mathbf{b}^2 \end{pmatrix}, \quad (29)$$

where $I : \mathbb{R}^{2p} \rightarrow \mathbb{R}^{2p}$ is the identity matrix. The systems are solved iteratively using the generalized minimal residual method (GMRES). The stopping tolerances are managed dynamically by the SQP algorithm. The key to the efficiency and scalability of the entire optimization framework [8] is in the fast solution of systems (29), which require a good preconditioner. We use the preconditioner

$$\pi^k = \begin{pmatrix} I & 0 \\ 0 & (\nabla \mathbf{C}(\mathbf{p}^k) \nabla \mathbf{C}(\mathbf{p}^k)^T + \varepsilon I)^{-1} \end{pmatrix}, \quad (30)$$

where $\varepsilon > 0$ is a small constant that depends on the mesh size h , e.g., $10^{-8}h$. The matrix in the 2×2 block, $(\nabla \mathbf{C}(\mathbf{p}^k) \nabla \mathbf{C}(\mathbf{p}^k)^T + \varepsilon I)$, is formed explicitly, and its inverse is applied using ML (Trilinos) [23, 24], a smoothed aggregation algebraic multigrid solver/preconditioner. We note that the design of the preconditioner π^k is motivated by a number of important considerations, such as sparsity of $\nabla \mathbf{C}(\mathbf{p}^k)$, possible rank deficiency of $\nabla \mathbf{C}(\mathbf{p}^k)$, iteration complexity of the preconditioned augmented system solve, iteration complexity of the overall SQP scheme, and scalability. These considerations are studied in some detail in Section 5.1.2, which examines the computational performance of the optimization algorithm using the application settings introduced below.

Remark 1. *It is often desirable to enforce the constraint $\mathbf{C}(\mathbf{p}) = \mathbf{0}$ as close to machine precision as possible. However, the SQP algorithm may terminate three or four digits away from machine precision. To recover two to three digits of accuracy in the constraint, we postprocess the SQP solution by solving another instance of the quasi-normal subproblem (26). This heuristic may result in a slight change of the objective function value, which is a reasonable trade-off.*

4. Applications

This section describes two different applications of the optimization-based volume correction algorithm. These applications are representative of the type of settings where failure to enforce correct cell volumes in deforming meshes may adversely impact the accuracy of the numerical solutions. Besides demonstrating accuracy improvements we also aim to assess the computational performance of the optimization algorithm and confirm its efficacy and suitability for practical computations.

Both applications involve meshes deforming under a non-divergent velocity field. We recall that trajectories of Lagrangian particles $\mathbf{p}(t)$ moving in a flow field with velocity $\mathbf{u}(\mathbf{p}(t), t)$ satisfy the kinematic relation

$$\frac{d\mathbf{p}}{dt} = \mathbf{u}(\mathbf{p}(t), t). \quad (31)$$

The first application setting models the evolution of the computational mesh under a non-divergent flow field and we refer to it as the *Lagrangian mesh motion*. Starting from a given initial mesh we solve (31) numerically to advance the mesh to its final position using a fixed time step. We apply optimization-based volume correction at each time step. The purpose of this example is to study how volume correction impacts the quality of the mesh and the accuracy of the vertex trajectories.

The second application combines the optimization-based volume correction algorithm with a semi-Lagrangian (SL) transport scheme to obtain a volume corrected SL method. This setting allows us to examine the importance of volume correction in the solution of transport problems and multi-material simulations.

4.1. Lagrangian mesh motion

Since the meshes in this setting evolve in time we label them and their entities by the time step, i.e., $K(t^n)$ is a mesh at time $t = t^n$, $P(t^n)$ is the set of its vertices $\mathbf{p}(t^n)$ and so on. We consider the motion of an initial mesh $K(t^0)$ under a non-divergent flow field (31) until a final time t^N and denote the *exact* deformed mesh at any time t by $K(t)$. In general, the exact mesh is not known, but since non-divergent flows preserve cell volumes $K(t^n)$ has the property $\mathbf{c}(K(t^n)) = \mathbf{c}(K(t^0)) =: \mathbf{c}_0$ at all time steps t^n .

To advance the mesh in time we solve (31) numerically using the mesh vertex positions at the previous time step as initial values. This gives a sequence of *uncorrected* mesh vertices $\{\hat{P}(t^n)\}_{n=1}^N$ such that

$$\hat{\mathbf{p}}(t^{n+1}) = \hat{\mathbf{p}}(t^n) + F(\mathbf{u}, t^n, \Delta t^n) \quad (32)$$

where $t^{n+1} = t^n + \Delta t^n$, $F(\mathbf{u}, t^n, \Delta t^n)$ is an explicit time stepping scheme, and Δt^n is a suitable time step. The new mesh vertices are connected by straight line segments following the mesh topology of the initial mesh. The meshes in the resulting *uncorrected* sequence $\{\hat{K}(t^n)\}_{n=1}^N$ have the same connectivity as $K(t^0)$, but their cells are not guaranteed to be valid, nor do these cells necessarily match the initial cell volumes in $K(t^0)$. The sequence of uncorrected meshes provides a benchmark for the volume correction algorithm.

We use this algorithm to construct another sequence $\{K^*(t^n)\}_{n=1}^N$ of *volume corrected* meshes as follows. Suppose that $K(t^0)$ is a given valid mesh and that a valid volume corrected mesh $K^*(t^n)$ at time step $t = t^n$ has been constructed. We advance this mesh to a valid volume corrected mesh $K^*(t^{n+1})$ in two stages. The first stage solves (31) numerically to advance the vertices $P^*(t^n)$ to their uncorrected positions $\tilde{P}(t^{n+1})$ at time $t = t^{n+1}$, i.e., we set

$$\tilde{\mathbf{p}}(t^{n+1}) = \mathbf{p}^*(t^n) + F(\mathbf{u}, t^n, \Delta t^n). \quad (33)$$

Reconnection of $\tilde{P}(t^{n+1})$ according to the topology of the initial mesh yields an intermediate *source* mesh $\tilde{K}(t^{n+1})$. Although $K^*(t^n)$ is by assumption valid, the source mesh may contain invalid cells and in general $\mathbf{c}(\tilde{K}(t^{n+1})) \neq \mathbf{c}_0$.

The second stage solves the volume correction problem for the pair $\{\tilde{K}(t^{n+1}), \mathbf{c}_0\}$ to obtain the corrected mesh vertices $P^*(t^{n+1})$, i.e.,

$$\mathbf{p}^*(t^{n+1}) = \underset{\mathbf{p}}{\operatorname{argmin}} \{J(\mathbf{p}) \quad \text{subject to} \quad \mathbf{C}(\mathbf{p}) = \mathbf{0}\}.$$

Linking of $P^*(t^{n+1})$ according to the initial mesh topology yields the volume corrected mesh $K^*(t^{n+1})$.

4.2. Volume corrected semi-Lagrangian transport scheme

The second application of the volume correction algorithm combines it with a semi-Lagrangian method (SL) for the solution of the related transport equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad \text{and} \quad \frac{\partial \rho T_s}{\partial t} + \nabla \cdot (T_s \rho \mathbf{u}) = 0, \quad s = 1, \dots, S, \quad (34)$$

where \mathbf{u} is a non-divergent velocity field. Equations (34) describe the evolution of a density ρ and a collection of passive tracers $\{T_s\}_{s=1}^S$ in a velocity field \mathbf{u} .

The tracers T_s can also be interpreted as mixing ratios or volume fractions of materials in a multi-material flow. As a result, besides testing volume correction for transport problems, this application allows us to examine its impact in the important case of multi-material flow simulations.

We choose the incremental remap approach [6] as a base SL scheme for this application. The SL scheme exploits conservation of mass (M) and tracer mass (Q_s) in Lagrangian cells $\kappa(t)$ analogous to (1), i.e.,

$$\frac{dM(t)}{dt} = 0 \quad \text{and} \quad \frac{dQ_s(t)}{dt} = 0 \quad (35)$$

where

$$M(t) = \int_{\kappa(t)} \rho dV \quad \text{and} \quad Q_s(t) = \int_{\kappa(t)} \rho T_s dV.$$

Consider the movement of the Lagrangian cell $\kappa(t)$ by one time step from $\kappa(t^n)$ to $\kappa(t^{n+1})$ where $t^{n+1} = t^n + \Delta t$. The conservation equation (35) implies a discrete (in time) conservation statement,

$$M(t^{n+1}) = M(t^n), \quad \text{and} \quad Q_s(t^{n+1}) = Q_s(t^n). \quad (36)$$

The volume averaged density associated with $\kappa(t)$ is

$$\langle \rho \rangle(t) = \frac{\int_{\kappa(t)} \rho dV}{\int_{\kappa(t)} dV} = \frac{M(t)}{|\kappa(t)|}, \quad (37)$$

and a density weighted volume averaged tracer for T_s is

$$\langle T_s \rangle(t) = \frac{\int_{\kappa(t)} \rho T_s dV}{\int_{\kappa(t)} \rho dV} = \frac{Q_s(t)}{M(t)}. \quad (38)$$

Together with (37)–(38) the discrete conservation (36) implies that the average density and tracer at time $t = t^{n+1}$ are given by

$$\langle \rho \rangle(t^{n+1}) = \frac{\int_{\kappa(t^{n+1})} \rho dV}{\int_{\kappa(t^{n+1})} dV} = \frac{M(t^{n+1})}{|\kappa(t^{n+1})|} = \frac{M(t^n)}{|\kappa(t^{n+1})|}, \quad (39)$$

and

$$\langle T_s \rangle(t^{n+1}) = \frac{\int_{\kappa(t^{n+1})} \rho T_s dV}{\int_{\kappa(t^{n+1})} \rho dV} = \frac{Q_s(t^{n+1})}{M(t^{n+1})} = \frac{Q_s(t^n)}{M(t^n)}, \quad (40)$$

respectively. The incremental remap SL scheme uses (39)–(40) in conjunction with a forward or a backward Lagrangian step to advect the average density and tracers to the next time step. For brevity we describe the backward SL scheme, the forward version involves the same phases but in different order.

SL schemes use a fixed Eulerian mesh \bar{K} to represent the approximate solution of (34) as it advances in time. Assume that the approximate solution at time $t = t^n$ is known on \bar{K} , i.e., for every cell $\bar{\kappa}_i \in \bar{K}$ we are given the approximate mass \bar{M}_i^n , mean density $\bar{\rho}_i^n$, tracer mass $\bar{Q}_{s,i}^n$ and mean tracer $\bar{T}_{s,i}^n$ such that

$$\begin{aligned} \bar{\rho}_i^n &\approx \frac{\int_{\bar{\kappa}_i} \rho(\mathbf{x}, t^n) dV}{\int_{\bar{\kappa}_i} dV}, & \bar{T}_{s,i}^n &\approx \frac{\int_{\bar{\kappa}_i} \rho(\mathbf{x}, t^n) T_s(\mathbf{x}, t^n) dV}{\int_{\bar{\kappa}_i} \rho(\mathbf{x}, t^n) dV}, \\ \bar{M}_i^n &= \bar{\rho}_i^n |\bar{\kappa}_i|, & \bar{Q}_{s,i}^n &= \bar{T}_{s,i}^n \bar{M}_i^n. \end{aligned}$$

To advance this solution to a solution \bar{M}_i^{n+1} , $\bar{\rho}_i^{n+1}$, $\bar{Q}_{s,i}^{n+1}$ and $\bar{T}_{s,i}^{n+1}$ at $t = t^{n+1}$ on the Eulerian mesh we identify \bar{K} with a Lagrangian mesh at this future time step, i.e., we set $K(t^{n+1}) := \bar{K}$. The mesh $K(t^{n+1})$ is referred to as the *arrival* mesh. The backward SL scheme then performs the following sequence of steps:

1. *Traceback*: Solve numerically (31) backward in time to compute an approximation $\tilde{K}(t^n)$ of the *departure* Lagrangian mesh $K(t^n)$. Using, e.g., an explicit Euler method, the equation

$$\tilde{\mathbf{p}}^n = \mathbf{p}^{n+1} + (-\Delta t)\mathbf{u}(\mathbf{p}^{n+1}, t^{n+1})$$

yields the vertices $\tilde{P}(t^n)$ of the approximate departure mesh.

2. *Remap*: Conservatively remap quantities from the Eulerian mesh \bar{K} onto the approximate departure mesh $\tilde{K}(t^n)$, i.e., for every cell $\tilde{\kappa}_i(t^n) \in \tilde{K}(t^n)$ find \tilde{M}_i^n , $\tilde{\rho}_i^n$, $\tilde{Q}_{s,i}^n$ and $\tilde{T}_{s,i}^n$, such that

$$\begin{aligned} \sum_{i=1}^m \tilde{M}_i^n &= \sum_{i=1}^m \bar{M}_i^n, & \sum_{i=1}^m \tilde{Q}_{s,i}^n &= \sum_{i=1}^m \bar{Q}_{s,i}^n, \\ \tilde{M}_i^n &= \tilde{\rho}_i^n |\tilde{\kappa}_i(t^n)|, & \text{and} & \quad \tilde{Q}_{s,i}^n = \tilde{T}_{s,i}^n \tilde{M}_i^n. \end{aligned} \quad (41)$$

We perform this step following [6]. Van Leer limiting yields the monotone reconstructions of the density and the tracers necessary for the implementation of the algorithms in this paper.

3. *Transfer to Lagrangian arrival mesh*: Compute mass and tracer mass on the arrival mesh according to (36): for all $\kappa_i(t^{n+1}) \in K(t^{n+1})$ set

$$M_i^{n+1} := \tilde{M}_i^n \quad \text{and} \quad Q_{s,i}^{n+1} := \tilde{Q}_{s,i}^n.$$

4. *Transfer to Eulerian fixed mesh*: For all $\bar{K} \ni \bar{\kappa}_i \equiv \kappa_i(t^{n+1}) \in K(t^{n+1})$ set

$$\bar{M}_i^{n+1} := M_i^{n+1} \quad \text{and} \quad \bar{Q}_{s,i}^{n+1} := Q_{s,i}^{n+1}$$

and compute average cell density and tracer according to (39) and (40):

$$\bar{\rho}_i^{n+1} := \frac{\bar{M}_i^{n+1}}{|\bar{\kappa}_i|} \quad \text{and} \quad \bar{T}_{s,i}^{n+1} := \frac{\bar{Q}_{s,i}^{n+1}}{\bar{M}_i^{n+1}}. \quad (42)$$

Assuming that at $t = t^n$ the density is constant on the Eulerian mesh, i.e., $\rho_i^n = c$ for some $c > 0$, let us examine the base SL solution at $t = t^{n+1}$. Non-divergent flows preserve Lagrangian volumes and so,

$$|\kappa_i(t^n)| = |\kappa_i(t^{n+1})| = |\bar{\kappa}_i|.$$

On the other hand, because $\tilde{K}(t^n)$ is an approximation of the true departure mesh, $|\tilde{\kappa}_i(t^n)| \neq |\kappa_i(t^n)|$, which implies that

$$|\tilde{\kappa}_i(t^n)| \neq |\bar{\kappa}_i|. \quad (43)$$

Since remap preserves constants, the second phase of the base SL scheme yields $\tilde{\rho}_i^n = c$ for all $\tilde{\kappa}_i(t^n) \in \tilde{K}(t^n)$. As a result,

$$\bar{\rho}_i^{n+1} = \frac{\bar{M}_i^{n+1}}{|\bar{\kappa}_i|} = \frac{M_i^{n+1}}{|\bar{\kappa}_i|} = \frac{\tilde{M}_i^n}{|\bar{\kappa}_i|} = \tilde{\rho}_i^n \frac{|\tilde{\kappa}_i(t^n)|}{|\bar{\kappa}_i|} \neq c. \quad (44)$$

Thus, (43) prevents the base SL scheme from preserving a constant density, and under some circumstances it can also lead to violation of global solution bounds. Indeed, if $|\tilde{\kappa}_i(t^n)| > |\bar{\kappa}_i|$ equation (44) implies that

$$\bar{\rho}_i^{n+1} = \tilde{\rho}_i^n \frac{|\tilde{\kappa}_i(t^n)|}{|\bar{\kappa}_i|} > c. \quad (45)$$

To mitigate these problems we propose to solve a volume correction problem for the approximate departure mesh by adding the following step to the base SL scheme:

1a. *Volume correction*: Solve the volume correction problem $\{\tilde{K}(t^n), \mathbf{c}_0\}$ with $\mathbf{c}_0 = \mathbf{c}(K(t^{n+1})) \equiv \mathbf{c}(\bar{K})$.

Inclusion of this step yields the volume corrected version of the base SL scheme. The vertices $P^*(t^n)$ of the volume corrected departure mesh $K^*(t^n)$ satisfy

$$\mathbf{p}^*(t^n) = \underset{\mathbf{p}}{\operatorname{argmin}} \{ J(\mathbf{p}) \quad \text{subject to} \quad \mathbf{C}(\mathbf{p}) = \mathbf{0} \}.$$

4.3. Multi-material incompressible flow

In the multi-material setting $\Omega(t) = \cup_{s=1}^S \Omega_s(t)$, where $\Omega_s(t)$ is a sub-domain occupied by a material s at time t . The intersection $\kappa_{s,i}(t) := \Omega_s(t) \cap \kappa_i(t)$ gives the part of material s in a Lagrangian cell $\kappa_i(t) \in K(t)$, and $\alpha_{s,i}(t) = |\kappa_{s,i}(t)|/|\kappa_i(t)|$ is the volume fraction of that material in $\kappa_i(t)$. Because $\kappa_{s,i}(t)$ is a Lagrangian cell on its own, for non-divergent flows its volume is preserved, $|\kappa_{s,i}(t)| = |\kappa_{s,i}(t^0)|$, which implies that

$$|\Omega_s(t)| = \sum_{i=1}^m |\kappa_{s,i}(t)| = \sum_{i=1}^m |\kappa_{s,i}(t^0)| = |\Omega_s(t^0)|, \quad (46)$$

that is, the total volume of each material is also preserved. Physically consistent simulations of multi-material flows require accurate approximations of volume fractions and conservation of material volumes.

By choosing $\rho = 1$ in (34) we can use these equations to study the importance of volume correction in the multi-material setting, such as the impact of inexact Lagrangian volumes on conservation of material volumes. Indeed, if $\rho = 1$ then one can identify the variable $T_s(\mathbf{x}, t)$ in (34) with the color (indicator or characteristic) function for $\Omega_s(t)$, i.e.,

$$T_s(\mathbf{x}, t) = \begin{cases} 1, & \text{if } \mathbf{x} \in \Omega_s(t) \\ 0, & \text{otherwise} \end{cases}, \quad (47)$$

while its mean cell value

$$\langle T_s \rangle(t) = \frac{\int_{\kappa_i(t)} T_s dV}{\int_{\kappa_i(t)} dV} = \frac{|\kappa_{s,i}(t)|}{|\kappa_i(t)|} = \alpha_{s,i}(t), \quad (48)$$

gives the volume fraction of material s in a Lagrangian cell $\kappa_i(t)$. Finally, since

$$\sum_{i=1}^m \langle T_s \rangle(t) |\kappa_i(t)| = \sum_{i=1}^m \alpha_{s,i}(t) |\kappa_i(t)| = |\Omega_s(t)|, \quad (49)$$

the total tracer volume can serve as a proxy for the total volume $|\Omega_s(t)|$ of material s .

Consider now a solution of this problem by the base SL scheme and assume that at time $t = t^n$ the average cell density $\bar{\rho}_i^n = 1$ on the Eulerian mesh. Let us first show that the remap phase of the algorithm preserves the total volume of material s , that is, $|\tilde{\Omega}_s^n| = |\bar{\Omega}_s^n|$. To this end we use the equivalency between the total tracer mass and the total tracer volume (which, according to (49), is a proxy for the total material volume $|\Omega_s(t)|$), and show that for non-divergent flows the remap step preserves the latter.

Because $\bar{\rho}_i^n = 1$ it follows that on the Eulerian mesh $\bar{M}_i^n = |\bar{\kappa}_i|$, and the total volume of material s on this mesh is

$$|\bar{\Omega}_s^n| = \sum_{i=1}^m \bar{T}_{s,i}^n |\bar{\kappa}_i| = \sum_{i=1}^m \bar{T}_{s,i}^n \bar{M}_i^n = \sum_{i=1}^m \bar{Q}_{s,i}^n.$$

On the other hand, since the remap step preserves constant density, after remapping $\bar{\rho}_i^n$ to the departure mesh $\tilde{K}(t^n)$ we have that $\tilde{\rho}_i^n = 1$ and $\tilde{M}_i^n = |\tilde{\kappa}_i(t^n)|$. Summing over all departure mesh cells shows that

$$|\tilde{\Omega}_s^n| = \sum_{i=1}^m \tilde{T}_{s,i}^n |\tilde{\kappa}_i(t^n)| = \sum_{i=1}^m \tilde{T}_{s,i}^n \tilde{M}_i^n = \sum_{i=1}^m \tilde{Q}_{s,i}^n.$$

Conservation of the total material volume $|\tilde{\Omega}_s^n| = |\bar{\Omega}_s^n|$ now follows from the second equation in (41)

$$\sum_{i=1}^m \tilde{Q}_{s,i}^n = \sum_{i=1}^m \bar{Q}_{s,i}^n,$$

which expresses conservation of the total tracer mass at the remap phase.

Let us examine the impact of volume errors (43) in the departure mesh on the accuracy of the base SL solution \bar{M}_i^{n+1} , $\bar{\rho}_i^{n+1}$, $\bar{Q}_{s,i}^{n+1}$ and $\bar{T}_{s,i}^{n+1}$ in the multi-material context. Besides the inability to recover a constant density, (44), the base SL scheme also yields erroneous estimates of the total volume of material s . Indeed, recalling that $\bar{T}_{s,i}^{n+1} = \tilde{T}_{s,i}^n$, we see that if $|\tilde{\kappa}_i(t^n)| \neq |\bar{\kappa}_i|$, then

$$|\bar{\Omega}_s^{n+1}| = \sum_{i=1}^m \bar{T}_{s,i}^{n+1} |\bar{\kappa}_i| = \sum_{i=1}^m \tilde{T}_{s,i}^n |\bar{\kappa}_i| \neq \sum_{i=1}^m \tilde{T}_{s,i}^n |\tilde{\kappa}_i(t^n)| = |\tilde{\Omega}_s^n| = |\bar{\Omega}_s^n|.$$

The errors in total material volumes correspond to artificial losses or gains of materials, i.e., they can lead to unphysical simulation results. Section 5.2 presents numerical results with the volume corrected SL scheme, which show that volume correction helps conserve total material volumes to machine precision.

5. Numerical results

We use the application settings from Section 4 to assess the performance of the volume correction algorithm as well as the qualitative and quantitative improvements in the accuracy of Lagrangian mesh motion and semi-Lagrangian transport schemes. In all numerical examples the initial guess for the optimization algorithm is a valid convex mesh.

All simulations are performed on an Intel(R) Xeon(R), CPU E5-1620, 3.60 GHz, and the optimization algorithm is implemented in Matlab (2013a) using vectorized operations. For the application of the algebraic multigrid preconditioner we rely on the Trilinos package ML [23, 24]. In the solution of all the augmented systems, see Section 3, we use a tolerance of 5.00e-01; for the ML preconditioner we set the maximum number of iterations to 20.

5.1. Lagrangian mesh motion

We consider the Lagrangian mesh motion setting, introduced in Section 4, with the velocity field [25]

$$\mathbf{u}_1(\mathbf{p}, t) = \begin{pmatrix} \cos\left(\frac{t\pi}{\mathcal{T}}\right) \sin(\pi x)^2 \sin(2\pi y) \\ -\cos\left(\frac{t\pi}{\mathcal{T}}\right) \sin(\pi y)^2 \sin(2\pi x) \end{pmatrix}, \quad (50)$$

where \mathcal{T} is the period, and the rotational velocity field

$$\mathbf{u}_2(\mathbf{p}) = \begin{pmatrix} 1 - 2y \\ 2x - 1 \end{pmatrix}. \quad (51)$$

Both fields are non-divergent. The computational domain $\Omega = [0, 1]^2$ is endowed with a uniform initial mesh $K(t^0)$ comprising $m = \ell^2$ square elements. The mesh size is $h = 1/\ell$ and $P(t^0)$ contains $(\ell + 1)^2$ points.

We solve (31) using the forward Euler method with a uniform time step Δt . Thus, the vertices of the uncorrected mesh $\hat{K}(t^{n+1})$ and the source mesh $\tilde{K}(t^{n+1})$ are given by

$$\hat{\mathbf{p}}(t^{n+1}) = \hat{\mathbf{p}}(t^n) + \Delta t \mathbf{u}(\hat{\mathbf{p}}(t^n), t^n), \quad (52)$$

and

$$\tilde{\mathbf{p}}(t^{n+1}) = \mathbf{p}^*(t^n) + \Delta t \mathbf{u}(\mathbf{p}^*(t^n), t^n),$$

respectively, where $\mathbf{p}^*(t^n)$ are the vertices of a volume corrected mesh $K^*(t^n)$ at time t^n . Finally, we compute a sequence of approximations to the ‘‘exact’’ Lagrangian meshes $\{K(t^n)\}_{n=1}^N$ by using a very fine initial uniform mesh and a refined time step. For notational simplicity in the discussion we denote the exact, uncorrected and corrected meshes after N time steps by K , \hat{K} and K^* , respectively, i.e.,

$$K = K(t^N); \quad \hat{K} = \hat{K}(t^N) \quad \text{and} \quad K^* = K^*(t^N). \quad (53)$$

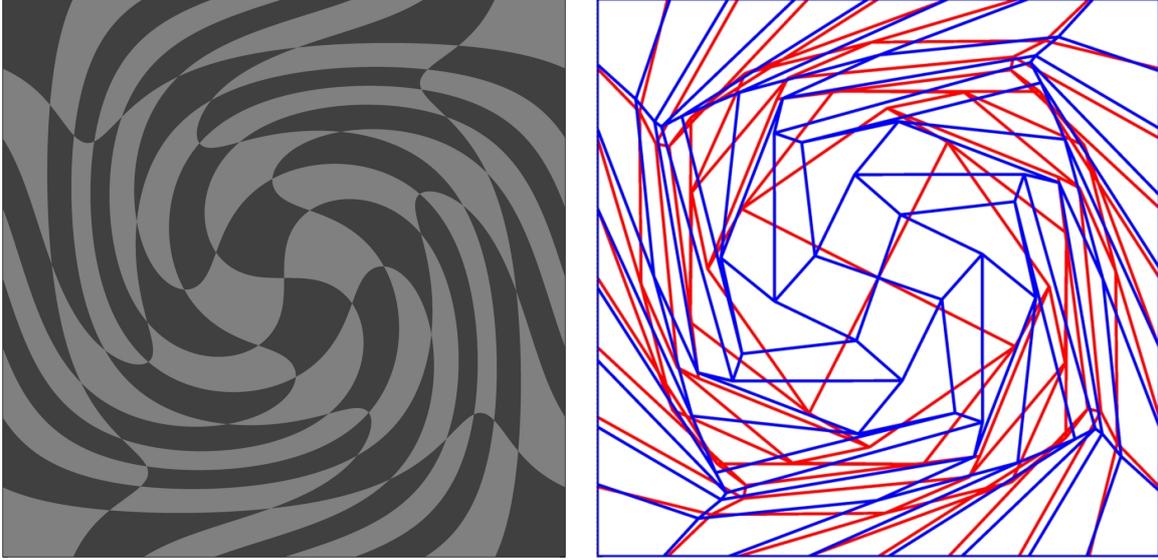


Figure 4: The exact mesh K (the different shades of grey serve to distinguish the mesh cells), on the left, and a superimposition of the uncorrected mesh \widehat{K} (red) and the corrected mesh K^* (blue) on the right. There are substantial differences between \widehat{K} and K^* , with the latter appearing closer to the exact mesh.

5.1.1. Qualitative tests

The tests in this section use the above Lagrangian mesh motion setting to examine improvements in the mesh geometry brought about by the volume correction algorithm. We start with a test that demonstrates how volume correction can produce meshes that are geometrically³ very close to the exact Lagrangian meshes.

Test 1. We set $\ell = 8$, $N = 4$ and compare \widehat{K} and K^* with K defined as in (53). The latter is approximated numerically by moving a very fine uniform mesh until time t^{N_e} with size $dx \ll \Delta x$, time step $dt \ll \Delta t$, $\ell = 256$ and such that $N_e dt = t^N$. Figures 4-7 summarize the numerical results; in all figures the exact cells from K are shaded. The first two figures compare the shapes of the exact cells with the shapes of the uncorrected and corrected cells. Figure 4 shows the exact mesh K on the left (the different shades of grey serve to distinguish the mesh cells) and a superimposition of \widehat{K} and K^* , on the right. We see substantial differences between the uncorrected mesh \widehat{K} and the corrected mesh K^* , with the latter appearing closer to the exact mesh. To assess the improvement in the accuracy of the corrected mesh, in Figure 5 we superimpose the exact mesh with \widehat{K} and K^* . The right plot in this figure clearly shows that in the “eye-ball” norm the corrected mesh is in much closer agreement with the exact mesh than the uncorrected mesh.

To give further credence to this observation in Figure 6 we compare the barycenters of the cells in the three meshes. The green circles are the barycenters of the exact cells, whereas the red squares and the blue stars show the barycenters of the uncorrected and the corrected meshes, respectively. The left plot reveals significant errors in barycenter positions on the uncorrected meshes, including many cases of barycenters located outside their corresponding exact cells. In contrast, the right plot shows that the barycenters of the corrected cells are very close to the exact ones and, with just few exceptions, inside the exact cells.

To appreciate the dramatic improvements in the mesh geometry afforded by the volume correction algorithm we examine two of the cells from the uncorrected mesh whose barycenters are grossly incorrect and do not belong to their respective exact cells. Figure 7 compares the shapes and barycenters of these two uncorrected cells with the exact cell shapes and the corrected cell shapes. The figure clearly shows the much improved accuracy of the corrected cells and the nearly perfect match of their barycenters with the

³By “geometrically close” we mean that the cell barycenters and the cell shapes are close.

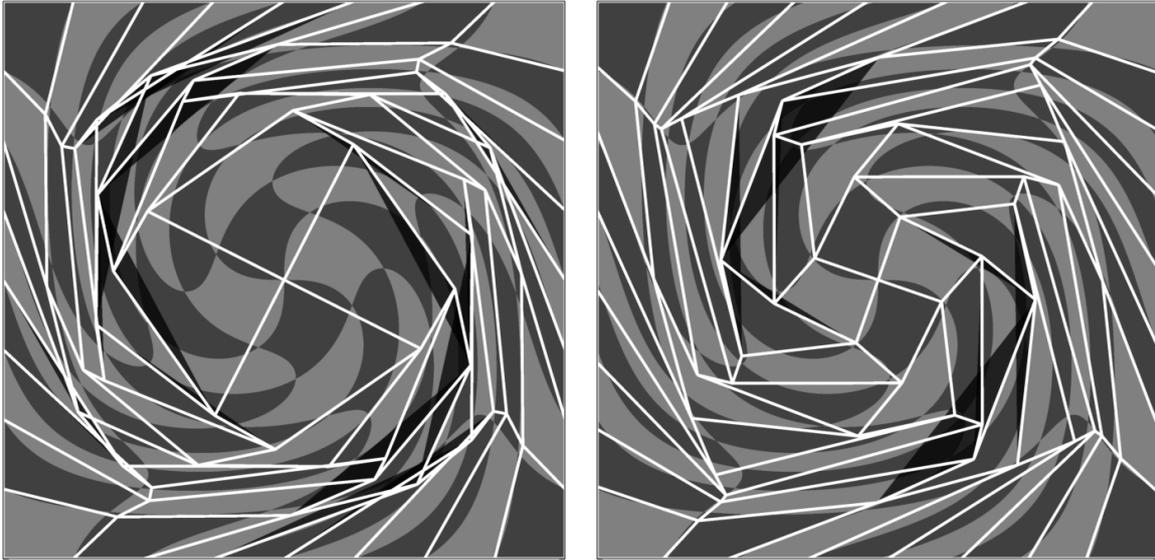


Figure 5: Superimposition of the exact mesh K with the uncorrected mesh \widehat{K} (left) and the corrected mesh K^* (right); the latter is in much closer agreement with the exact mesh than the uncorrected mesh.

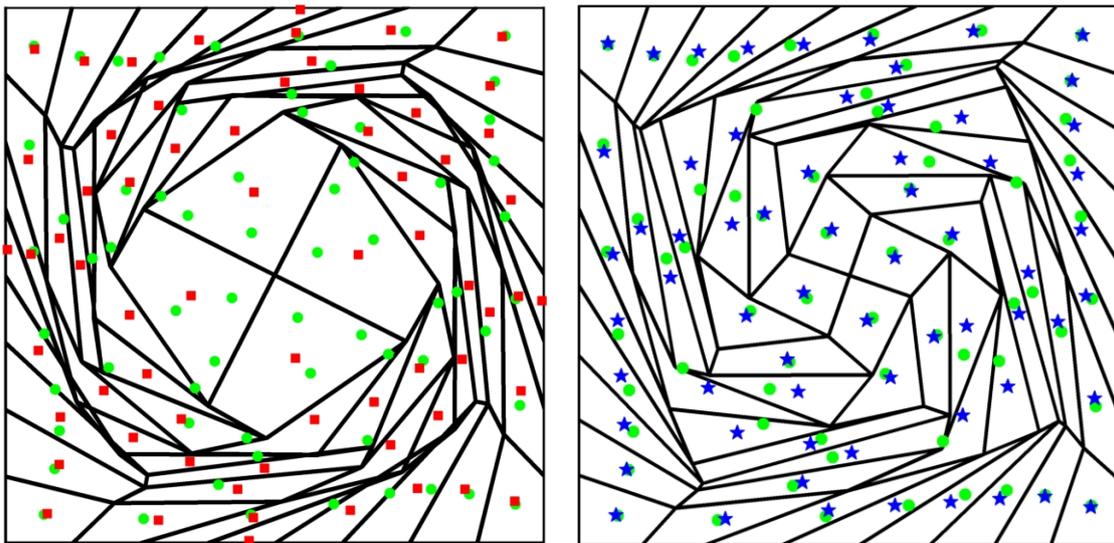


Figure 6: Comparison of the exact barycenters (green circles) with those of the uncorrected mesh \widehat{K} (red squares), on the left, and the corrected mesh K^* (blue stars), on the right. In the left plot there are significant errors in barycenter positions; in the right plot the barycenters of the corrected cells are very close to the exact ones and almost always inside the exact cells.

exact ones.

Comparison of the trajectories of the points $\{P(t^n)\}_{n=1}^N$ in the exact mesh with the trajectories of the points in $\{\widehat{P}(t^n)\}_{n=1}^N$ and $\{P^*(t^n)\}_{n=1}^N$ provides another useful diagnostic for assessing the geometric improvements in the corrected mesh. Figure 8 shows the trajectories of a single point from $\{P(t^n)\}_{n=1}^N$ and the corresponding points from $\{\widehat{P}(t^n)\}_{n=1}^N$ and $\{P^*(t^n)\}_{n=1}^N$ for \mathbf{u}_1 with $N = 5$ and \mathbf{u}_2 with $N = 100$. In both cases the uncorrected point trajectories deviate significantly from the exact trajectory, whereas the corrected point tracks the exact trajectory almost exactly.

These results show that volume correction leads to corrected meshes K^* whose cells have shapes, positions

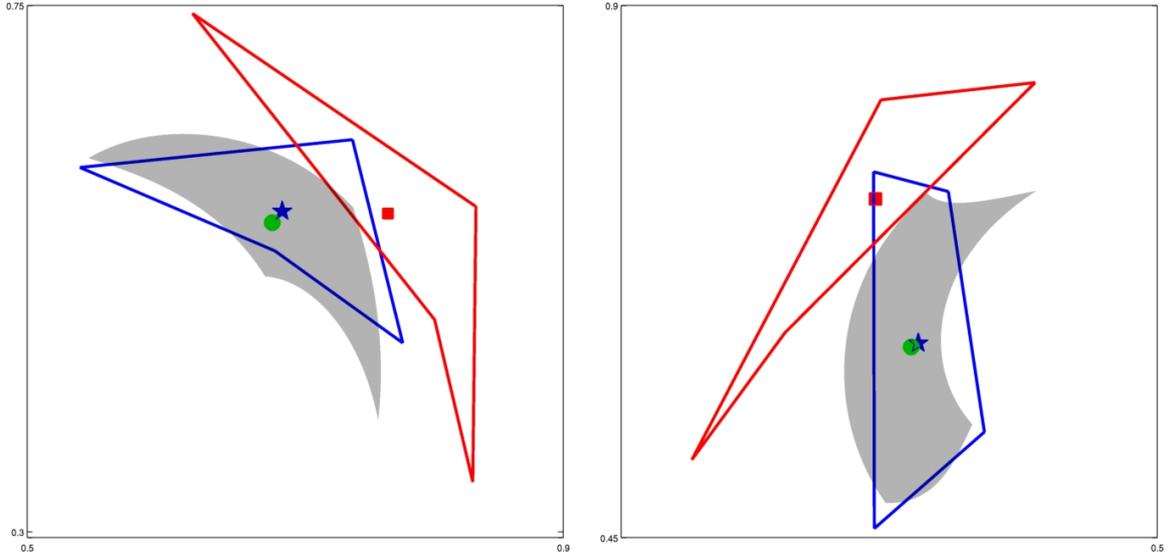


Figure 7: The grey shape and the green circle correspond to an exact cell $\kappa \in K$ and its barycenter, respectively. The red lines and the red square show the shape and the barycenter of corresponding *uncorrected* cell $\hat{\kappa} \in \hat{K}$. The blue lines and the blue star are the shape and the barycenter of the *corrected* cell $\kappa^* \in K^*$. The corrected cells feature a much improved accuracy and a nearly perfect match of their barycenters with the exact ones.

ℓ	N	β	Uncorrected mesh		Corrected mesh	
			invalid	non-convex	invalid	non-convex
8	4	5.0e-06	12	20	0	0
16	10	5.0e-06	8	88	0	0
32	19	1.0e-10	0	44	0	0
64	50	1.0e-10	0	80	0	0

Table 1: Numerical results of Test 2; ℓ is the number of elements in each direction, N is the number of time steps, β is the penalization parameter, ‘invalid’ and ‘non-convex’ are the numbers of invalid and non-convex cells in the uncorrected mesh \hat{K} and the corrected mesh K^* . These results show that, for empirically chosen values of β , the optimal meshes are always convex.

and barycenters very close to those of the exact Lagrangian mesh K , and whose points track the exact point trajectories very closely. Among other things, the latter suggests that application of the volume correction in conjunction with a transport scheme may allow for bigger time steps in the simulation of advection problems such as (34). The results in Section 5.2 corroborate this conjecture.

Test 2. This test examines the volume correction in the context of mesh quality improvement. To this end, we consider an invalid and/or non-convex uncorrected mesh \hat{K} and compute the corrected mesh K^* by solving the modified optimization problem (20). Table 1 reports ℓ , N , β , and the number of invalid and non-convex cells in \hat{K} and K^* . This test uses the field \mathbf{u}_1 in (50). The results in Table 1 demonstrate that, for empirically chosen values of β , the optimal meshes are always convex. In Figure 9 we report the uncorrected mesh \hat{K} (left) and the corrected mesh K^* (right) for the first row of Table 1. For the same row, in Figure 10 we report \hat{K} highlighting in dark brown the invalid cells and in light brown the non-convex ones. Though these results show that the logarithmic barrier yields convex meshes, the convexity of K^* highly depends on the choice of β .

5.1.2. Performance tests.

In this section we modify the Lagrangian mesh motion setting to study the computational cost of the optimization algorithm, in terms of iteration complexity and runtime. Since the focus is on the performance

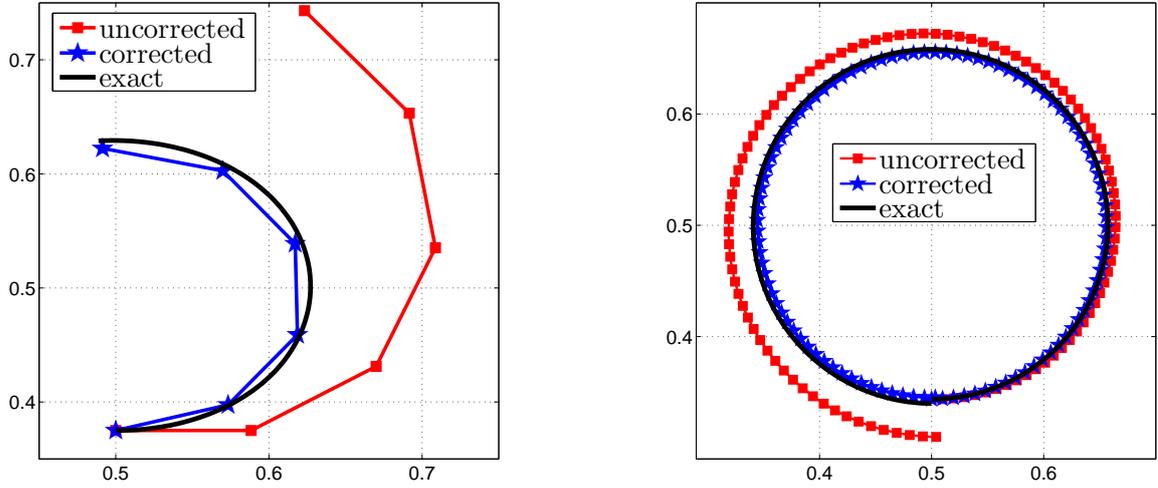


Figure 8: Comparison of the exact trajectory of a single point of $\{P(t^n)\}_{n=1}^N$ (black line) with the trajectories of the corresponding uncorrected (red squares) and corrected (blue stars) points from $\{\hat{P}(t^n)\}_{n=1}^N$ and $\{P^*(t^n)\}_{n=1}^N$, respectively. Left plot: u_1 and $N = 5$. Right plot: u_2 and $N = 100$. The uncorrected point deviates significantly from the exact trajectory, whereas the corrected point tracks the exact trajectory almost exactly.

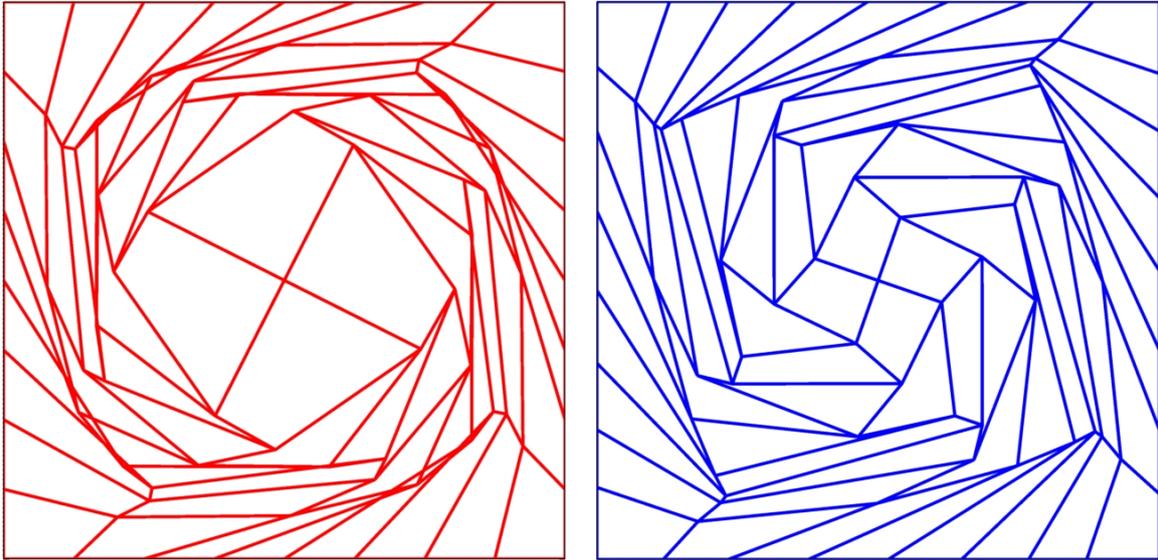


Figure 9: Comparison of the uncorrected mesh \hat{K} (left), corresponding to the data in the first row of Table 1, and the resulting volume corrected mesh K^* (right).

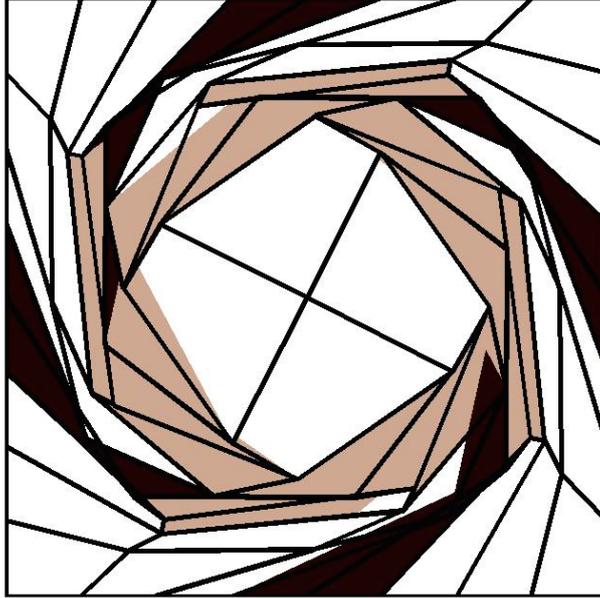


Figure 10: Uncorrected mesh for the first row in Table 1. The dark and light shading marks the invalid and the non-convex cells, respectively, in \tilde{K} .

of the algorithm we restrict attention to a single application of the mesh correction procedure (19) to a given source mesh \tilde{K} and a vector of desired mesh volumes \mathbf{c}_0 , and report the iteration complexity and the computational costs. To challenge the algorithm we set $\tilde{K} = \hat{K}$, i.e., the source mesh coincides with the *uncorrected* mesh at time t^N . We remind that the latter is generated by using (52) to solve numerically (31) until time t^N using the points of the uniform mesh as initial condition. Therefore, the vector \mathbf{c}_0 corresponds to the volumes of the cells in the initial uniform mesh. Note that for larger N the degree of deformation of \tilde{K} increases.

The performance studies involve two approximations H^k of the Hessian of the Lagrangian: 1) ‘Analytic Hessian’ refers to combining the analytic action of the objective function Hessian, $\nabla^2 J(\mathbf{p})$, on some vector \mathbf{v} , with finite differencing⁴ the action of the constraint Jacobian transpose, $\nabla C(\mathbf{p})^T$, on the Lagrange multiplier vector $\boldsymbol{\lambda}$ in the direction \mathbf{v} ; 2) ‘Gauss-Newton Hessian’ refers to using only the analytic action of the Hessian of the objective function, $\nabla^2 J(\mathbf{p})$, on some vector \mathbf{v} . We conduct performance tests using the velocity field \mathbf{u}_1 in (50). Tables 2 and 3 summarize the results: for increasing values of ℓ we double (Table 2) or quadruple (Table 3) N in order to obtain various degrees of deformation.

We report the number of SQP, CG, and GMRES (total and average) iterations, the computational time and the percentage of time spent in the multigrid solver ML. Overall, as we increase the number of mesh cells we observe little to no increase in iteration complexity, evidenced by the columns ‘SQP’, ‘CG’ and ‘GMRES tot.’; an approximately linear increase in total runtime, column ‘CPU’, and the dominating computational cost of the multigrid preconditioner, column ‘% ML time’. In essence, the computational complexity of our approach is determined fully by the computational complexity of the multigrid preconditioner, and the numerical evidence points to scalability at the large scale. We also note that the differences between the two approximations of the Hessian of the Lagrangian are negligible, so the Gauss-Newton approximation may be preferred in practice due to ease of implementation.

A non-convex source mesh provides an opportunity to test both the performance of the algorithm and the ability of the logarithmic barrier penalty to enforce cell convexity in the volume compliant mesh. To this end we repeat the above tests using a source mesh \tilde{K} obtained by evolving the initial uniform mesh

⁴The bilinearity of the constraints guarantees that this approximation is exact.

Analytic Hessian						
ℓ	SQP	CG	GMRES tot.	GMRES av.	CPU	% ML time
64	3	2	34	2.3	0.962	59
128	3	2	42	2.8	3.551	75
256	2	1	30	3.0	10.544	82
512	3	1	49	3.5	87.076	88

Gauss-Newton Hessian						
ℓ	SQP	CG	GMRES tot.	GMRES av.	CPU	% ML time
64	3	2	28	1.9	0.860	65
128	3	2	36	2.4	3.115	81
256	2	1	25	2.5	8.787	85
512	3	1	43	3.1	73.775	90

Table 2: Performance results where the source mesh \tilde{K} is an uncorrected mesh obtained by solving (31) until time t^N using a uniform mesh for the initial condition and for the vector of desired volumes \mathbf{c}_0 . We use $N = 10$ for $\ell = 64$ and we *double* N as we refine the mesh. We observe nearly constant iteration complexity and a linear increase in runtime. Note that approximately 90% of CPU time is spent in the ML preconditioner on the finest mesh. This points to the potential for scalability for large problems.

until the deformation is such that some cells become non-convex. For different ℓ we report in Table 4 the values of N and the number of non-convex cells in \tilde{K} (‘non-convex’ in the table). The parameter N_{guess} is the number of time steps used to generate the initial guess for the SQP algorithm, this mesh is obtained by solving (31) until time $t^{N_{guess}}$ with the same parameters used to compute \tilde{K} .

Note that for all ℓ the volume compliant mesh K^* is convex, i.e., the logarithmic barrier penalty is sufficient to enforce convexity constraints. In Table 5 we observe roughly constant iteration complexity and a linear increase in CPU time as we refine the mesh. The absolute CPU time is slightly higher compared to Tables 2 and 3. This is expected due to the additional nonlinear term in the objective function.

We now zoom in on a detailed study of the preconditioner π^k developed in Section 3. As stated earlier, this preconditioner is key for the efficiency of the algorithm. We discuss the results of our performance study by calling out several aspects that influence the design of π^k .

(a) *Sparsity*: The Jacobian matrix $\nabla \mathbf{C}(\mathbf{p}^k)$ may be very large, but it is also very sparse. This is due to the fact that cell volume computations only involve mesh points that are shared by a small number of immediate neighbors. Therefore, performing a sparse matrix-matrix multiplication to form the matrix $(\nabla \mathbf{C}(\mathbf{p}^k) \nabla \mathbf{C}(\mathbf{p}^k)^T + \varepsilon I)$ is efficient in this context. Moreover, the cost of forming the matrix is amortized over many augmented system solves in the SQP algorithm, because the matrix is recomputed only when \mathbf{p}^k changes. Some evidence of the amortization can be found in the dominating computational cost of the ML preconditioner, column ‘% ML time’ in Tables 2, 3 and 5.

(b) *Degeneracy*: Depending on the mesh connectivity and the boundary constraints, it is possible that the matrix $\nabla \mathbf{C}(\mathbf{p}^k)$ is rank-deficient. Our implementations of the SQP algorithm and the GMRES solver allow for such scenarios. However, the preconditioner π^k is only well-posed with the addition of the εI term. As a whole, π^k is positive definite. In our performance examples $\nabla \mathbf{C}(\mathbf{p}^k)$ is mildly rank-deficient, yet Tables 2, 3 and 5 exhibit no adverse effects on the solvability of augmented systems or the convergence of the SQP algorithm.

(c) *Optimal iteration complexity*: When $\nabla \mathbf{C}(\mathbf{p}^k)$ is full-rank, and $\varepsilon = 0$, we can prove that GMRES applied to (29) with the π^k preconditioner converges in *at most three iterations*, see [26]. Overall, we observe very small, and nearly constant, numbers of GMRES iterations per augmented system solve in all numerical examples, including those with rank-deficient Jacobian matrices. In Tables 2, 3 and 5 the average GMRES iteration numbers range from two to five, columns ‘GMRES av.’. The slight deviation from the theoretical bound of three iterations is due primarily to the inexact application of the ML preconditioner and the nonzero ε , $\varepsilon = 10^{-8}h$.

Analytic Hessian						
ℓ	SQP	CG	GMRES tot.	GMRES av.	CPU	% ML time
64	3	2	34	2.3	0.962	59
128	3	2	42	2.8	3.668	78
256	3	2	52	3.5	17.915	84
512	6	3	114	4.1	193.779	88

Gauss-Newton Hessian						
ℓ	SQP	CG	GMRES tot.	GMRES av.	CPU	% ML time
64	3	2	28	1.9	0.860	65
128	3	2	36	2.4	3.159	82
256	3	2	42	2.8	14.426	87
512	6	3	96	3.4	163.074	91

Table 3: Performance results where the source mesh \tilde{K} is an uncorrected mesh obtained by solving (31) until time t^N using a uniform mesh for the initial condition and for the vector of desired volumes \mathbf{c}_0 . We use $N = 10$ for $\ell = 64$ and we *quadruple* N as we refine the mesh. As before, we observe nearly constant iteration complexity and a linear increase in runtime, i.e., our approach is robust to added mesh deformation.

ℓ	N	non-convex	N_{guess}
64	66	52	56
128	165	64	155
256	425	64	415
512	1127	112	1117

Table 4: Parameters used for performance tests in the non-convex case. Results are reported in Table 5. The source mesh \tilde{K} is computed solving (31) until time t^N using a uniform mesh for the initial condition and for the vector of desired volumes \mathbf{c}_0 . N is chosen such that \tilde{K} has some non-convex cells (see the column ‘non-convex’). N_{guess} is the number of time steps used to generate the initial guess for the SQP algorithm, this is obtained by solving (31) with the same parameters used to compute \tilde{K} .

Analytic Hessian						
ℓ	SQP	CG	GMRES tot.	GMRES av.	CPU	% ML time
64	5	15	101	2.8	2.475	66
128	4	9	106	4.1	8.799	78
256	5	5	130	5.0	45.733	83
512	6	1	100	3.8	184.446	83

Gauss-Newton Hessian						
ℓ	SQP	CG	GMRES tot.	GMRES av.	CPU	% ML time
64	5	5	64	2.5	1.666	63
128	4	4	79	3.8	6.466	82
256	5	5	126	4.8	43.241	86
512	6	6	100	3.8	183.697	86

Table 5: Performance results in the non-convex case. The parameters used to generate \tilde{K} are reported in Table 4. As for the convex meshes, we observe nearly constant iteration complexity and, overall, scalable performance.

(d) *Applicability of multigrid*: A key consideration in formulating π^k is rooted in the structure of the Jacobian matrix. In the context of volume constraints (4) the graph of the matrix $\nabla C(\mathbf{p}^k)$ resembles the *graph divergence*. Therefore, the graph of $\nabla C(\mathbf{p}^k)\nabla C(\mathbf{p}^k)^T$ resembles the *graph div-grad* operator, i.e., the *graph Laplacian*. This is why algebraic multigrid is a good choice for the application of our preconditioner. The linear increase in CPU time per SQP iteration (obtained by dividing ‘CPU’ column values by ‘SQP’ column values in Tables 2, 3 and 5) as the mesh is refined is evidence of the preconditioner’s suitability for the volume correction problem.

(e) *Scalability*: As is apparent from our numerical examples, the SQP algorithm requires a very small, mesh-independent number of iterations to achieve near machine accuracy. Similarly, the inner projected CG algorithm converges in a handful of iterations. Finally, as mentioned above, the number of GMRES iterations per augmented system solve is also modest. All three iteration numbers are bounded by a small, mesh-independent constant. Here, this constant is 6, based on the results from Tables 2, 3 and 5. Therefore, our algorithm inherits its scalability directly from the algebraic multigrid solver.

5.2. Semi-Lagrangian transport

This section compares the base semi-Lagrangian transport scheme from Section 4.2 with its volume corrected version. We use the two schemes to solve (34) on the square domain $\Omega = [0, 1]^2$ with the velocity field (50). The initial densities and tracers are drawn from the following set of constant and piecewise-constant functions on the unit square:

$$\begin{aligned} \rho_{cyl,0}(x, y) &= \begin{cases} 2 & (x, y) \in B_{0.15}(0.5, 0.75) \\ 1 & (x, y) \notin B_{0.15}(0.5, 0.75), \end{cases} \\ \rho_{cyl,1}(x, y) &= \begin{cases} 0 & (x, y) \notin B_{0.15}(0.6, 0.75) \text{ and} \\ & (x, y) \in [0.575, 0.625] \times [0.60, 0.85] \\ 0.5 & \text{otherwise,} \end{cases} \\ \rho_{cyl,2}(x, y) &= \begin{cases} 0 & (x, y) \notin B_{0.15}(0.4, 0.65) \text{ and} \\ & (x, y) \in [0.375, 0.425] \times [0.50, 0.75] \\ 0.5 & \text{otherwise,} \end{cases} \\ \rho_{cyl,3}(x, y) &= 1 - \rho_{cyl,1}(x, y) - \rho_{cyl,2}(x, y), \quad \text{and} \quad \rho_{const}(x, y) = 2. \end{aligned}$$

where $B_r(x_c, y_c)$ is the ball of radius r centered in (x_c, y_c) .

Test 1. The purpose of this test is to illustrate the importance of volume correction in multi-material computations. As explained in Section 4.3 solution of (34) with $\rho = 1$ and S tracers simulates a multi-material setting comprising S different materials. In this case, the total tracer volume of T_s serves as a proxy for the total volume $|\Omega_s(t)|$ of material s .

In the test we consider the advection of $S = 3$ tracers T_1, T_2 and T_3 with initial distributions $T_1^0 = \rho_{cyl,1}$, $T_2^0 = \rho_{cyl,2}$, and $T_3^0 = \rho_{cyl,3}$. We set the period \mathcal{T} of the velocity field (50) equal to 1 and run the simulation for a total of 10 periods. Thus, the final time is $t^N = 10$, where $N = 1630$. We compute the difference between the initial tracer volumes and the volumes at each time step. We report the maximum value of such difference over the 10 periods in Tables 6 (without correction) and 7 (with correction). These results show that the volume correction gives a volume error close to machine epsilon. On the other hand, without the volume correction the errors in tracer volumes are not insignificant. Further evidence is provided in Figure 11 where, for $\ell = 64$, we report the graphs of the tracer volume errors for $t \in (0, t^N)$ with and without the volume correction.

Test 2. The purpose of this test is three-fold. First, it demonstrates the importance of volume correction for the preservation of constant densities in semi-Lagrangian transport schemes, i.e., for the mitigation of (44). Second, the test illustrates how (45) can lead to potentially significant violations of global solution bounds by an uncorrected scheme. Finally, the test confirms that volume correction completely eliminates violation of solution bounds, thereby ensuring physically meaningful solutions.

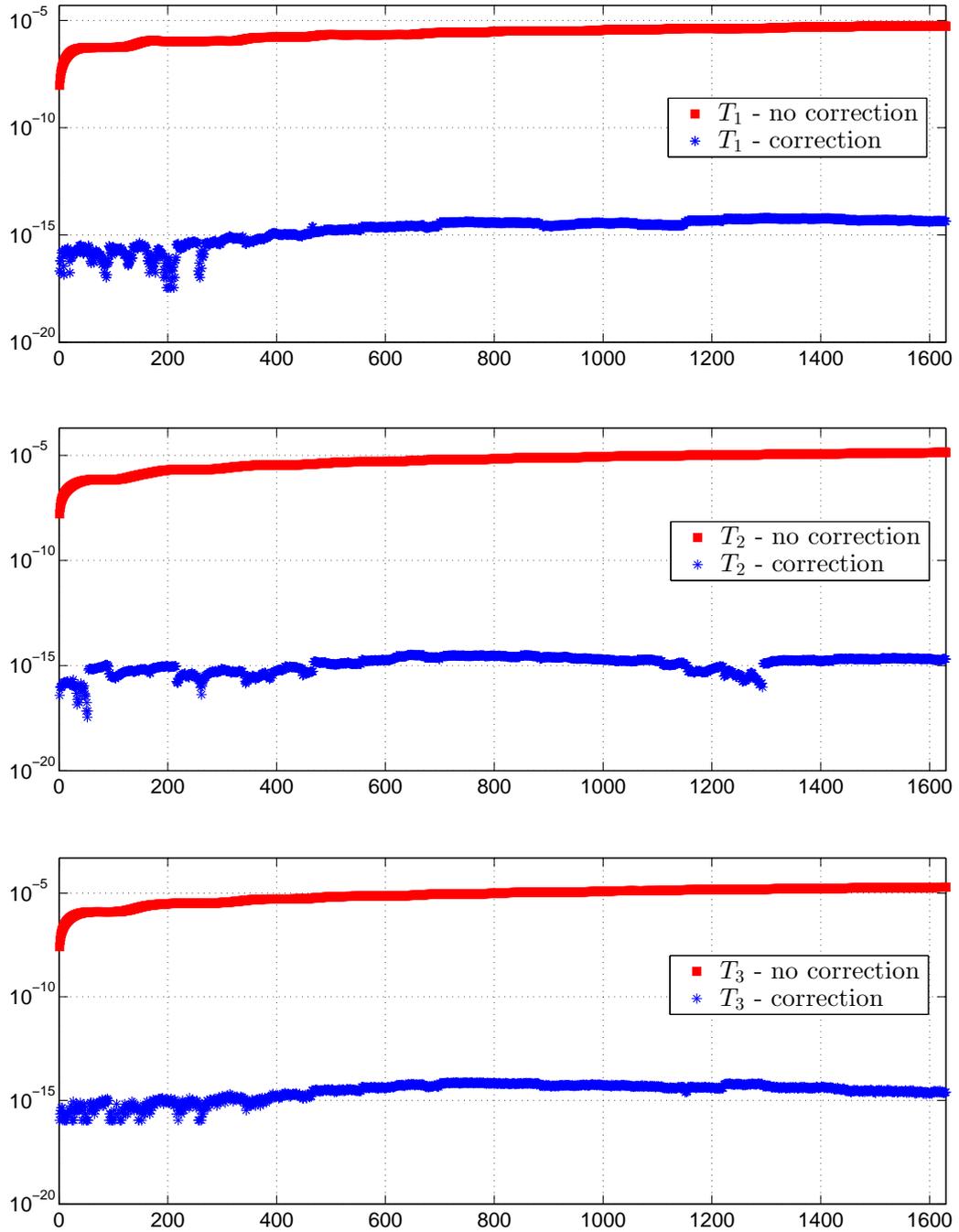


Figure 11: For Test 1, tracer volume errors with $\ell = 64$, with (blue) and without (red) correction. From top to bottom: tracer 1, 2, and 3; on the horizontal axis, the number of time steps. These results show that the volume correction gives a volume error close to machine epsilon.

ℓ	error T_1	error T_2	error T_3
8	7.8238e-04	1.2839e-03	2.2198e-03
16	2.3068e-04	5.1011e-04	7.7943e-04
32	3.5385e-05	9.1752e-05	1.3297e-04
64	5.3857e-06	1.3853e-05	1.9531e-05
128	7.2083e-07	1.7872e-06	2.5131e-06

Table 6: For Test 1, tracer volume errors *without* volume correction.

ℓ	error T_1	error T_2	error T_3
8	7.6328e-17	1.5266e-16	4.4409e-16
16	3.8858e-16	6.3144e-16	9.9920e-16
32	2.2100e-15	4.2119e-15	5.9952e-15
64	6.0854e-15	3.2439e-15	7.5495e-15
128	7.8514e-15	4.4444e-15	1.0769e-14

Table 7: For Test 1, tracer volume errors *with* volume correction. These results show that the volume correction gives a volume error close to machine epsilon.

In this test we solve the density equation from (34) with two different initial density distributions, ρ_{const} and $\rho_{cyl,0}$, respectively, using the uncorrected and corrected semi-Lagrangian schemes from Section 4.2. Time stepping is by Runge-Kutta 4 and the Forward Euler time integration schemes with two constant time step sizes, $\Delta t = 0.006$ and $\Delta t = 0.0001$, respectively. The choice of the time steps ensures that spatial and temporal errors are of comparable sizes. In the first case the number of time steps is $N = 250$ and in the second case is $N = 15000$. We start by examining the preservation of constant densities with and without the volume correction. Specifically, using the initial density ρ_{const} we compute, at time $t^N = 1.5$ the L^1 , L^∞ and L^2 -norms of the difference between the initial and the final densities. For $\Delta t = 0.006$ the first rows in Table 8 show that with both Runge-Kutta 4 and Forward Euler the uncorrected semi-Lagrangian scheme does not preserve the initial constant density. On the other hand, the third rows in the same table reveal that the volume-corrected transport scheme preserves ρ_{const} to a machine precision. The second rows in Table 8 show that without volume correction the errors in the constant density approximation persists despite decreasing the time step size to $\Delta t = 0.0001$.

The next suite of results focusses on the violation of physical solution bounds with and without the volume correction. For the same parameters and for both ρ_{const} and $\rho_{cyl,0}$ we report in Tables 9 and 10 respectively, the maximum and minimum values of the density over the time interval $(0, t^N)$; the results in the first and third row show that only with volume correction the bounds are preserved. The results in the second row show that, without volume correction, even decreasing the time step size to $\Delta t = 0.0001$, bounds are violated. In Figure 12 we illustrate these results reporting a slice of the density distribution; here, the red line corresponds to ρ_{const} (left) and $\rho_{cyl,0}$ (right). The green and blue lines correspond to the densities obtained with and without correction using the Forward Euler scheme with $\Delta t = 0.006$. Figures 13 and 14 illustrate the same result, for ρ_{const} and $\rho_{cyl,0}$ respectively, displaying the density configuration without (left) and with (right) volume correction.

6. Conclusion

We developed and studied numerically a new optimization-based method for volume preserving mesh correction, and an efficient optimization algorithm for the solution of the associated optimization problem. Our work formulates a new approach for improving the accuracy and physical fidelity of numerical schemes that rely on Lagrangian mesh motion, while the availability of a fast optimization algorithm makes the approach a practical alternative to existing volume correction methods.

Besides demonstrating the computational efficiency of the approach, numerical tests also reveal significant geometric improvements in the corrected meshes, which make the shapes and locations of their cells closer

Runge-Kutta 4				
Correction	Δt	L^1	L^∞	L^2
no	0.006	1.961e-05	5.235e-05	2.285e-05
no	0.0001	5.265e-07	1.378e-06	6.154e-07
yes	0.006	1.100e-12	4.555e-11	2.408e-12

Forward Euler				
Correction	Δt	L^1	L^∞	L^2
no	0.006	5.057e-02	3.191e-01	7.928e-02
no	0.0001	1.331e-03	9.134e-03	2.135e-03
yes	0.006	1.276e-12	3.804e-11	2.563e-12

Table 8: For Test 2, norms of the difference between the initial and the final density with ρ_{const} and $\ell = 128$. These results demonstrate that for the same Δt only the volume correction can eliminate the error. The test case with time step $\Delta t = 0.0001$ (second row) shows that, even reducing the time step, the error persists without volume correction.

Runge-Kutta 4			
Correction	Δt	min	max
no	0.006	1.999958147187	2.000052352773
no	0.0001	1.999998877779	2.000001378110
yes	0.006	1.999999999960	2.000000000046

Forward Euler			
Correction	Δt	min	max
no	0.006	1.680896653394	2.077796635104
no	0.0001	1.990865550875	2.002007661565
yes	0.006	1.999999999962	2.000000000038

Table 9: For Test 2, minimum and maximum values of ρ_{const} over the time interval $(0, t^N)$, with $\ell = 128$. These results demonstrate that for the same Δt only the volume correction preserves the bounds. The test case with time step $\Delta t = 0.0001$ (second row) shows that, even reducing the time step, the bounds cannot be preserved without volume correction.

Runge-Kutta 4			
Correction	Δt	min	max
no	0.006	0.9999790735933	2.000008983643
no	0.0001	0.9999994388897	1.999997888211
yes	0.006	0.9999999999783	1.999999999265

Forward Euler			
Correction	Δt	min	max
no	0.006	0.8404478260526	2.067996323565
no	0.0001	0.9954327396943	2.001650970717
yes	0.006	0.9999999999731	1.999998996500

Table 10: For Test 2, minimum and maximum values of $\rho_{cyl,0}$ over the time interval $(0, t^N)$, with $\ell = 128$. These results demonstrate that for the same Δt only the volume correction preserves the bounds. The test case with time step $\Delta t = 0.0001$ (second row) shows that, even reducing the time step, the bounds cannot be preserved without volume correction.

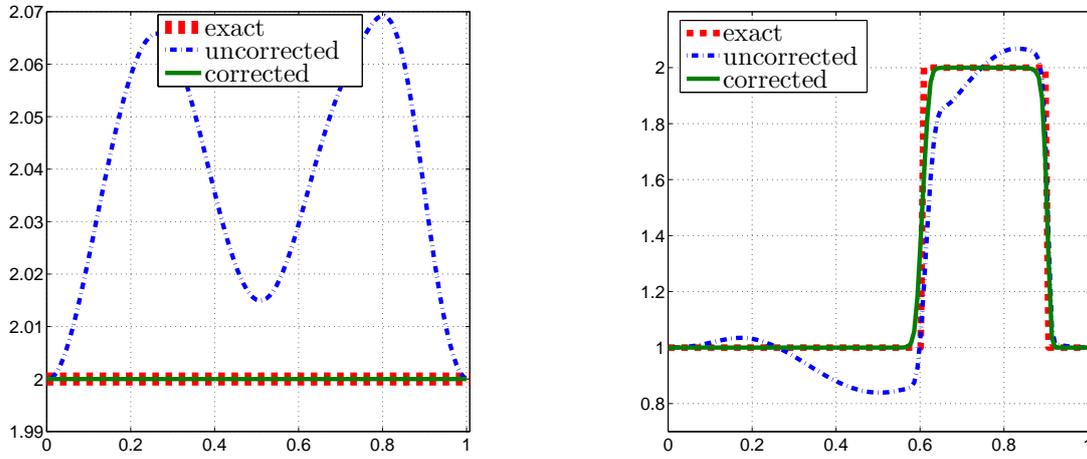


Figure 12: Plots of slice of the density at time $t^N = 1.5$ for Forward Euler simulations with $\Delta t = 0.006$ with and without correction. The exact densities (in red) correspond to ρ_{const} (left) and $\rho_{cyl,0}$ (right). These plots show that the volume correction preserves bounds.

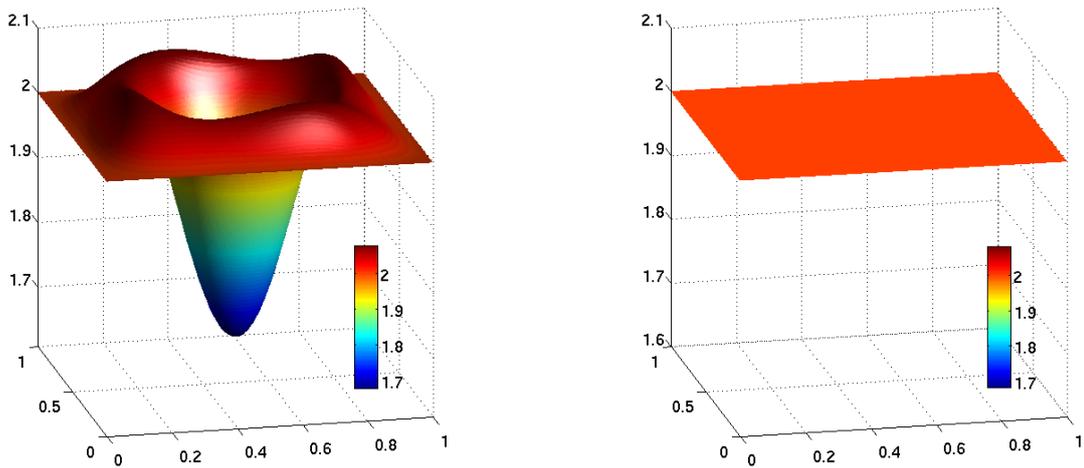


Figure 13: Plots of the density at time $t^N = 1.5$ for Forward Euler simulations with $\Delta t = 0.006$ without (left) and with (right) correction with initial density ρ_{const} . These plots show that the volume correction preserves bounds.

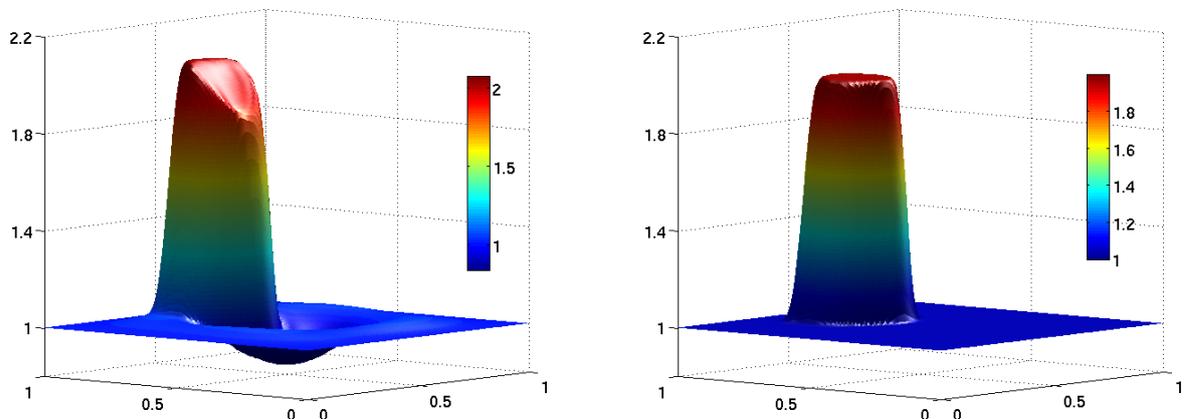


Figure 14: Plots of the density at time $t^N = 1.5$ for Forward Euler simulations with $\Delta t = 0.006$ without (left) and with (right) correction with initial density $\rho_{cyl,0}$. These plots show that the volume correction preserves bounds.

to those of the exact Lagrangian mesh. In particular, the trajectories of the corrected mesh vertices are very close to their true Lagrangian trajectories. Further numerical tests confirm that the geometric mesh improvements enable larger time steps in the context of, e.g., semi-Lagrangian transport schemes.

In addition to improvements in mesh geometry, the algorithm is also capable of recovering a valid mesh with convex cells from a mesh containing multiple invalid and/or non-convex cells, which suggest potential utility of the approach in a mesh quality improvement context.

In the future work we will address the development of optimization algorithms that robustly and scalably handle additional mesh-quality constraints, expressed as nonlinear inequalities, based on interior-point methods. We also plan to investigate the rigorous enforcement of mesh validity. Finally, we wish to apply our methods to tetrahedral meshes, and study extensions to other types of cells in three dimensions.

Acknowledgements

All the authors acknowledge the support of the US Department of Energy Office of Science Advanced Scientific Computing Research (ASCR) Program in Applied Mathematics Research. The work of M. Shashkov was performed under the auspices of the National Nuclear Security Administration of the US Department of Energy at Los Alamos National Laboratory under Contract No. DE-AC52-06NA25396. M. Shashkov also gratefully acknowledges the partial support of the US Department of Energy National Nuclear Security Administration Advanced Simulation and Computing (ASC) Program.

M. Shashkov would like to thank H.-T. Ahn and S. Li who participated in an initial work on volume correction ideas in a context of interface reconstruction and Dr. P. K. Smolarkiewicz for sharing early conference version of reference [17].

References

- [1] J. Campbell, M. Shashkov, A compatible lagrangian hydrodynamics algorithm for unstructured grids, *Selcuk Journal of Applied Mathematics* 4 (2) (2003) 53–70.
- [2] C. Hirt, A. Amsden, J. Cook, An arbitrary Lagrangian-Eulerian computing method for all flow speeds 14 (1974) 227–253.
- [3] D. Xiu, G. E. Karniadakis, A semi-lagrangian high-order method for navier–stokes equations, *Journal of Computational Physics* 172 (2) (2001) 658 – 684. doi:<http://dx.doi.org/10.1006/jcph.2001.6847>.
- [4] M. Zerroukat, N. Wood, A. Staniforth, Slice: A semi-lagrangian inherently conserving and efficient scheme for transport problems, *Quarterly Journal of the Royal Meteorological Society* 128 (586) (2002) 2801–2820. doi:[10.1256/qj.02.69](https://doi.org/10.1256/qj.02.69).

- [5] M. Rancic, An efficient, conservative, monotonic remapping for semi-lagrangian transport algorithms, *Monthly Weather Review* 123 (4) (1995) 1213–1217. doi:doi: 10.1175/1520-0493(1995)123<1213:AECMRF>2.0.CO;2.
- [6] J. K. Dukowicz, J. R. Baumgardner, Incremental remapping as a transport/advection algorithm, *Journal of Computational Physics* 160 (1) (2000) 318 – 335. doi:DOI: 10.1006/jcph.2000.6465.
- [7] P. D. Thomas, C. K. Lombard, Geometric conservation law and its application to flow computations on moving grids, *AIAA Journal* 17 (10) (1979) 1030–1037. doi:10.2514/3.61273.
- [8] M. Heinkenschloss, D. Ridzal, A Matrix-Free Trust-Region SQP Method for Equality Constrained Optimization, *SIAM Journal on Optimization* 24 (3) (2014) 1507–1541. doi:10.1137/130921738.
- [9] P. H. Lauritzen, R. D. Nair, P. A. Ullrich, A conservative semi-lagrangian multi-tracer transport scheme (cslam) on the cubed-sphere grid, *Journal of Computational Physics* 229 (5) (2010) 1401 – 1424. doi:10.1016/j.jcp.2009.10.036.
- [10] B. Nkonga, H. Guillard, Godunov type method on non-structured meshes for three-dimensional moving boundary problems, *Computer Methods in Applied Mechanics and Engineering* 113 (1–2) (1994) 183 – 204. doi:http://dx.doi.org/10.1016/0045-7825(94)90218-6.
- [11] M. Lesoinne, C. Farhat, Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations, *Computer Methods in Applied Mechanics and Engineering* 134 (1–2) (1996) 71 – 90. doi:http://dx.doi.org/10.1016/0045-7825(96)01028-6.
- [12] H. Guillard, C. Farhat, On the significance of the geometric conservation law for flow computations on moving meshes, *Computer Methods in Applied Mechanics and Engineering* 190 (11–12) (2000) 1467 – 1482. doi:http://dx.doi.org/10.1016/S0045-7825(00)00173-0.
- [13] T. Arbogast, C.-S. Huang, A fully mass and volume conserving implementation of a characteristic method for transport problems, *SIAM Journal on Scientific Computing* 28 (6) (2006) 2001–2022. doi:10.1137/040621077.
- [14] T. Arbogast, C.-S. Huang, A fully conservative eulerian–lagrangian method for a convection–diffusion problem in a solenoidal field, *Journal of Computational Physics* 229 (9) (2010) 3415 – 3427. doi:http://dx.doi.org/10.1016/j.jcp.2010.01.009.
- [15] T. Arbogast, W. Wang, Convergence of a fully conservative volume corrected characteristic method for transport problems, *SIAM Journal on Numerical Analysis* 48 (3) (2010) 797–823. arXiv:http://dx.doi.org/10.1137/09077415X, doi:10.1137/09077415X.
- [16] J. Douglas, Jr., C.-S. Huang, F. Pereira, The modified method of characteristics with adjusted advection, *Numerische Mathematik* 83 (3) (1999) 353–369. doi:10.1007/s002110050453.
- [17] J.-F. Cossette, P. K. Smolarkiewicz, A monge–ampère enhancement for semi-lagrangian methods, *Computers & Fluids* 46 (1) (2011) 180 – 185, 10th {ICFD} Conference Series on Numerical Methods for Fluid Dynamics (ICFD 2010). doi:http://dx.doi.org/10.1016/j.compfluid.2011.01.029.
- [18] J.-F. Cossette, P. K. Smolarkiewicz, P. Charbonneau, The monge–ampère trajectory correction for semi-lagrangian schemes, *Journal of Computational Physics* (0) (2014) –. doi:http://dx.doi.org/10.1016/j.jcp.2014.05.016.
- [19] P. Ciarlet, *The Finite Element Method for Elliptic Problems*, SIAM Classics in Applied Mathematics, SIAM, Philadelphia, 2002.
- [20] J. Matoušek, M. Sharir, E. Welzl, A subexponential bound for linear programming, *Algorithmica* 16 (4-5) (1996) 498–516. doi:10.1007/BF01940877.
- [21] F. Curtis, J. Nocedal, A. Wächter, A Matrix-Free Algorithm for Equality Constrained Optimization Problems with Rank-Deficient Jacobians, *SIAM Journal on Optimization* 20 (3) (2010) 1224–1249. doi:10.1137/08072471X.
- [22] R. Byrd, F. Curtis, J. Nocedal, An Inexact SQP Method for Equality Constrained Optimization, *SIAM Journal on Optimization* 19 (1) (2008) 351–369. doi:10.1137/060674004.
- [23] M. Gee, C. Siefert, J. Hu, R. Tuminaro, M. Sala, ML 5.0 Smoothed Aggregation User’s Guide, Tech. Rep. SAND2006-2649, Sandia National Laboratories (2006).
- [24] M. A. Heroux, J. M. Willenbring, A New Overview of the Trilinos Project, *Sci. Program.* 20 (2) (2012) 83–88. doi:10.3233/SPR-2012-0355.
- [25] R. J. LeVeque, High-resolution conservative algorithms for advection in incompressible flow, *SIAM Journal on Numerical Analysis* 33 (2) (1996) 627–665. doi:10.1137/0733033.
- [26] M. Murphy, G. Golub, A. Wathen, A Note on Preconditioning for Indefinite Linear Systems, *SIAM Journal on Scientific Computing* 21 (6) (2000) 1969–1972. doi:10.1137/S1064827599355153.