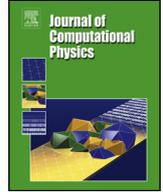




Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

A fast dynamic smooth adaptive meshing scheme with applications to compressible flow

Raaghav Ramani*, Steve Shkoller

Department of Mathematics, University of California, Davis, CA 95616, USA



ARTICLE INFO

Article history:

Received 13 June 2022
 Received in revised form 24 March 2023
 Accepted 5 June 2023
 Available online 14 June 2023

Keywords:

Mesh adaptation
 Moving meshes
 Monge-Ampère
 Compressible flows
 WENO
 Rayleigh-Taylor

ABSTRACT

We develop a fast-running smooth adaptive meshing (SAM) algorithm for dynamic curvilinear mesh generation, which is based on a fast solution strategy of the time-dependent Monge-Ampère (MA) equation, $\det \nabla \psi(x, t) = G \circ \psi(x, t)$. The novelty of our approach is a new so-called *perturbation formulation* of MA, which constructs the solution map ψ via composition of a sequence of near-identity deformations of a reference mesh. Then, we formulate a new version of the deformation method [21] that results in a simple, fast, and high-order accurate numerical scheme and a dynamic SAM algorithm that is of optimal complexity when applied to time-dependent mesh generation for solutions to hyperbolic systems such as the Euler equations of gas dynamics. We perform a series of challenging 2D and 3D mesh generation experiments for grids with large deformations, and demonstrate that SAM is able to produce smooth meshes comparable to state-of-the-art solvers [22,18], while running approximately 200 times faster. The SAM algorithm is then coupled to a simple Arbitrary Lagrangian Eulerian (ALE) scheme for 2D gas dynamics. Specifically, we implement the C-method [64,65] and develop a new ALE interface tracking algorithm for contact discontinuities. We perform numerical experiments for both the Noh implosion problem as well as a classical Rayleigh-Taylor instability problem. Results confirm that low-resolution simulations using our SAM-ALE algorithm compare favorably with high-resolution uniform mesh runs.

© 2023 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (<http://creativecommons.org/licenses/by/4.0/>).

Contents

1.	Introduction	2
1.1.	Mesh refinement for multi- D gas dynamics	3
1.2.	Fast smooth adaptive meshing	4
1.3.	Application to ALE gas dynamics	4
1.4.	Outline	5
2.	Preliminaries	5
2.1.	Domains, meshes, and mappings	5
2.2.	Eulerian and ALE variables	6
2.3.	Derivatives and important geometric quantities	6
2.4.	Computational platform and code optimization	7
3.	Fast static adaptive meshing	7

* Corresponding author.

E-mail addresses: rmani@math.ucdavis.edu (R. Ramani), shkoller@math.ucdavis.edu (S. Shkoller).

3.1.	Mathematical formulation of static mesh generation	7
3.2.	The basic mesh generation procedure	7
3.3.	Numerical implementation details	9
3.4.	High-order accuracy and a benchmark computation	10
4.	Fast dynamic adaptive meshing	12
4.1.	Dynamic formulation	12
4.2.	Reformulation in terms of near-identity maps	12
4.3.	Restarted dynamic mesh generation	14
5.	Dynamic mesh generation experiments	14
5.1.	Static mesh with large zoom-in factor	14
5.2.	Propagating circular front	16
5.3.	Uniformly rotating patch	17
5.4.	Differential rotation with small scales	18
5.5.	3D swirling flow	20
6.	SAM-ALE scheme for gas dynamics	21
6.1.	The 2D ALE-Euler system	21
6.2.	The C-method for 2D ALE-Euler	24
6.3.	Coupled SAM-ALE algorithm	26
7.	SAM-ALE simulations of gas dynamics	26
7.1.	Noh implosion	26
7.2.	Rayleigh-Taylor instability	27
8.	Concluding remarks	31
	CRediT authorship contribution statement	31
	Declaration of competing interest	31
	Data availability	31
	Acknowledgements	31
	Appendix A. The C-method for 2D ALE-Euler	32
	Appendix B. Boundary smoothing for non-Neumann functions	33
	Appendix C. The MK scheme	33
	C.1. Machine comparison	33
	References	34

1. Introduction

The efficiency of smooth moving-mesh methods for numerical simulations of gas dynamics¹ and related systems has been investigated in recent years [3,85,35,36,62,58,25,49,26]; however, to the best of our knowledge, compelling evidence of the gain in efficiency relative to fixed uniform-mesh simulations in multiple space dimensions has rarely been provided. In a recent result [49], the authors demonstrate that low-resolution adaptive simulations are roughly 2–6 times faster than high-resolution uniform simulations of comparable quality; most results in this area focus on novel solution methodologies but not on the ultimate speed-up that may be gained by the algorithms that they produce. The papers cited above focus on one half of the moving-mesh methodology, namely, the numerical discretization of the physical PDEs. They develop state-of-the-art high-resolution shock-capturing techniques, but use well-established and somewhat standard meshing algorithms. Our point-of-view is that it is essential to simultaneously develop both numerical methods for hyperbolic systems (for discontinuous solutions) as well as novel meshing strategies.²

Herein, we propose a novel and fast³ Smooth Adaptive Meshing (SAM) algorithm for multi- D simulations requiring mesh adaptivity. We present adaptive-simulation speed-up results for two classical but extremely challenging gas dynamic problems: the Noh shock implosion, and the (highly unstable) Rayleigh-Taylor (RT) test. For the Noh problem, our adaptive simulations are free of the numerical anomalies that are present in almost all reported results, while running approximately 6 times faster than a comparable uniform-mesh simulation. The ten-fold speed-up provided by SAM for the RT problem is, to the best of our knowledge, the first of its kind.⁴

¹ Moving-mesh simulations are often referred to as *adaptive simulations* and we shall use this terminology herein.

² This philosophy is in agreement with [62], in which the authors state that the main obstacle in their moving-mesh simulations is the lack of a simple, robust, and efficient algorithm for dynamic and smooth adaptive mesh generation, particularly in 3D geometries, and for multi-phase flows with unstable interfaces.

³ We will demonstrate that our SAM algorithm is the first to be able to solve classical Rayleigh-Taylor problems on coarse, but adaptive, grids faster than simulations on uniform grids.

⁴ Most attempts at using moving-mesh adaptivity to numerically simulate the RT instability result in runs that prematurely blow-up due to mesh tangling, meaning that those algorithms are not sufficiently stable to provide a competitive speed-up factor. Recent papers [57,4] instead focus on novel and sophisticated meshing techniques with the goal of simply simulating the RT instability until the final simulation time without the code crashing; however, these meshing algorithms are currently too expensive to provide speed-up over uniform-mesh simulations.

1.1. Mesh refinement for multi-D gas dynamics

It is by now well-known that static uniform meshes are both inaccurate and inefficient at representing the dynamically evolving and interacting small-scale structures that appear in solutions to nonlinear conservation laws in multiple space dimensions. Adaptive mesh refinement (AMR) via h -adaptivity is the most well-developed refinement technique and is used in many commercial codes [70,10,29,30]. However, the dyadic refinement at the heart of AMR schemes results in an artificially discontinuous transition from coarse-scale to fine-scale representation of numerical solutions on AMR meshes. Several theoretical and numerical studies [5,78,56] have demonstrated the spurious reflection, refraction, and scattering of waves that propagate across discontinuously refined grids. Many problems in gas dynamics, such as strong blast waves, self-similar implosions, and unstable contact discontinuities are extremely sensitive to small perturbations; spurious wave reflections produce corrupted numerical solutions, with the anomalies persisting, or even worsening, as the AMR mesh is globally refined [29,76].

On the other hand, Lagrangian-type schemes are well-known to produce highly distorted or tangled meshes i.e. some cells in the grid are non-convex or have folded over, at which point the simulation breaks down. Arbitrary Lagrangian Eulerian (ALE) methods aim to mitigate the problem of mesh tangling. *Indirect* ALE methods are somewhat *ad hoc*, and current rezoning strategies are heuristic in nature [46,57]. In this work, we consider the *direct* ALE approach, in which an adaptive mesh is generated directly without any initial Lagrangian phase or subsequent mesh rezoning.

1.1.1. Adaptive mesh redistribution

Our SAM scheme falls under the category of r -refinement schemes, or adaptive mesh redistribution methods. In contrast to Lagrangian-rezone methods, a grid is generated via a user-prescribed monitor function which determines the grid size and orientation. High-resolution representation of numerical solutions is obtained by defining the monitor function appropriately, e.g., using solution derivatives. Moreover, the adaptive grids can be generated to align with the geometry of evolving fronts [39], and to naturally capture self-similar dynamics or scale-invariant structures [13,11].

Historically, the first r -refinement methods were based on the variational approach, examples of which include the equipotential [82], variable diffusion [83], cost function [6], and harmonic mapping [27] methods. The variational approach also currently appears to be the method of choice for use in direct ALE schemes, several of which employ the popular MMPDE framework [40,47]. These variational methods, however, require the accurate numerical solution of a coupled set of d complicated nonlinear auxiliary PDEs in \mathbb{R}^d , for which simple, fast, and accurate algorithms are in general not available. For these reasons, among others, r -refinement methods have yet to become incorporated into large scale established hydrodynamics codes. See, for example, [41,14,22,18] and the references therein for thorough reviews of r -adaptive methods and their associated difficulties.

1.1.2. Prescribing the Jacobian determinant

The fundamental guiding principle for smooth adaptive mesh generation is control of the local cell volume of the adaptive grid. In the time-dependent multi- D setting, we assume that we have a given smooth positive *target Jacobian function* $G(y, t)$ describing the size of the cells in the moving target adaptive mesh. We then seek to construct a diffeomorphism $\psi(x, t)$ mapping a fixed reference mesh to the target mesh by requiring that $\det \nabla \psi(x, t) = G(\psi(x, t), t)$. A semi-discretization in time $t = t_k$, where k is the time-index, yields a sequence of nonlinear elliptic equations of Monge-Ampère (MA) type

$$\det \nabla \psi_k(x) = G_k(\psi_k(x)), \quad (1)$$

where each G_k is again a given positive target Jacobian function.

Solutions to the MA equation are unique in 1D. For dimension $d \geq 2$, however, the single scalar MA equation is insufficient to uniquely determine ψ . The question then becomes how to choose a particular solution ψ that is in some sense optimal. One such choice that has received a great deal of attention in recent years is the Monge-Kantorovich (MK) formulation based on optimal transport, in which a map ψ is (uniquely [8,16]) constructed to minimize the L^2 displacement $\|\psi(x) - x\|_{L^2}$. This is attractive from a numerical perspective, since smaller grid velocities can reduce interpolation and other numerical errors [48].

On the other hand, the MK formulation results in a fully nonlinear second order elliptic equation, whose numerical solution is difficult to obtain. One approach is to consider a parabolized formulation by introducing an artificial time variable τ then iterating until a steady state is reached [72,9,59,81]. In this case, the Jacobian constraint is only satisfied in the asymptotic limit $\tau \rightarrow \infty$, and many iterations may be required to obtain a sufficiently accurate solution, particularly for target meshes with large deformations. An alternative, fully nonlinear approach using preconditioned Newton-Krylov solvers is designed in [22,18], leading to a robust, scalable algorithm that is, to the best of our knowledge, the state-of-the-art in the field (see also the recent papers [12,15]). However, the Newton-Krylov iterative approach is still relatively slow for our ultimate goal of efficient adaptive gas dynamics simulations; specifically, its implementation in our ALE scheme (to be described below) leads to adaptive mesh simulations with computational runtimes greater than would otherwise be

obtained with a uniformly high-resolution mesh, thereby defeating the purpose of using an adaptive meshing scheme in the first place.

1.2. Fast smooth adaptive meshing

In contrast to the MK approach, we construct a map ψ_k satisfying (1) with the aim of optimizing for the efficiency of the resulting numerical algorithm, which we refer to as SAM. The key to our fast SAM algorithm is a new *perturbation formulation* of (1) along with a new formulation and implementation of the deformation method [21].

Specifically, the perturbation formulation constructs each map ψ_{k+1} as the image of the map ψ_k acting on a *near identity deformation* $\delta\psi_{k+1} \approx \text{id}$ of a fixed reference mesh Ω_{ref} . The formulation on Ω_{ref} is crucial, since it enables the use of, at each time-step t_k , the same numerical solvers for the mesh PDEs.⁵ This, in turn, produces a code with a simple modular structure so that the basic mesh redistribution procedure is developed entirely in the static setting on Ω_{ref} , then “bootstrapped” to form a dynamic scheme. The same principle also yields an algorithm for efficiently generating smooth meshes with very large zoom-in factors, which allows us to obtain high-resolution representation of small-scale structures with few total number of mesh points.

The mesh redistribution algorithm we propose is a new version of the deformation method [50,54,51], which constructs a solution to the nonlinear Monge-Ampère equation via a single elliptic solve for a linear Poisson problem, along with the solution of a system of transport equations for a flowmap $\eta(x, \tau)$ between pseudo-time $\tau = 0$ and $\tau = 1$. There are at least two advantages of this new deformation method: the first is that the algorithm can be made fully automated with no user-prescribed parameters; the second is that costly and often complicated interpolation procedures are not required. We design a simple, fast, stable, and high-order accurate method using an efficient spectral solver with boundary smoothing for the Poisson equation, and standard RK4 time integration with high-order linear upwind differencing for the transport equations. A key implication of our numerical design choices is a consistency between the stability conditions for the transport problem in SAM and the physical time-step in an ALE gas dynamics simulation. As we shall demonstrate, this consistency results in a dynamic SAM algorithm with optimal complexity for hyperbolic systems.

Our SAM algorithm is approximately 200 times faster than the MK nonlinear solvers [22,18], and the computed numerical solutions exhibit both higher accuracy as well as better convergence rates under global mesh refinement. We perform a number of challenging mesh generation experiments designed to replicate flows with high vorticity and large deformations, and demonstrate that the meshes produced with our dynamic SAM scheme are smooth and accurate. For example, we are able to generate smooth moving meshes that resolve around a complex 3D swirling helical-type curve at 256^3 resolution with only a serial implementation on a laptop computer and without any specific and sophisticated algorithmic optimizations (see Section 5.5).

1.3. Application to ALE gas dynamics

To demonstrate the efficacy of our SAM scheme in practical applications, we formulate a simple coupled SAM-ALE method for 2D gas dynamics. Several moving-mesh methods for the 2D Euler system have been developed based on the MMPDE approach and finite volume (FV) and finite element (FE) methods [73,74]. A formulation on smooth tensor product meshes enables the use of finite difference (FD) methods, which are both simpler and more efficient than FV and FE methods,⁶ and have been investigated in several recent papers [60,45,49]. In this work, we further develop the C-method [64,65], a simplified WENO-based solver with space-time smooth nonlinear artificial viscosity and explicit tracking of material interfaces.

Special care is given to the so-called *geometric conservation law* (GCL), and we show that our nonlinear WENO reconstruction procedure respects the free-stream preservation property on adaptive meshes. The C-method dynamically tracks the location and geometry of evolving fronts, and is used to add both directionally isotropic and anisotropic artificial viscosity to shocks and contacts. Herein, we implement the C-method in the ALE context and introduce a new ALE front-tracking algorithm for contact discontinuities, which we subsequently use to construct suitable target Jacobian functions for SAM. Previous studies have mainly investigated target Jacobian functions constructed based on interpolation errors [42,39], or weighted combinations of solution gradient estimates [77], which sometimes fail to capture small scale vortical structures [73]. Our simple ALE front-tracking algorithm allows us to generate smooth adaptive meshes that capture small scale Kelvin-Helmholtz roll-up zones in unstable RT problems. We apply our coupled SAM-ALE scheme to two challenging test problems, namely the Noh implosion and RT instability. For the Noh problem, we find that the 50×50 SAM-ALE solution is more accurate than the 200×200 uniform solution, while running approximately 6 times faster. Moreover, the SAM-ALE solution is completely free of spurious numerical anomalies, such as lack of symmetry, unphysical oscillations, and wall-heating. For the RT problem, we find that the 64×128 SAM-ALE solution is comparable to the 256×512 uniform solution, while running 10 times faster.

⁵ This is in contrast with other methods [67,32] which require finite-element solvers with costly recalculation (at each time-step of a dynamic simulation) of the mass and stiffness matrices, as well as complicated interpolation procedures.

⁶ FV schemes are 4 times more expensive than FD schemes in 2D, and 9 times more expensive in 3D [80].

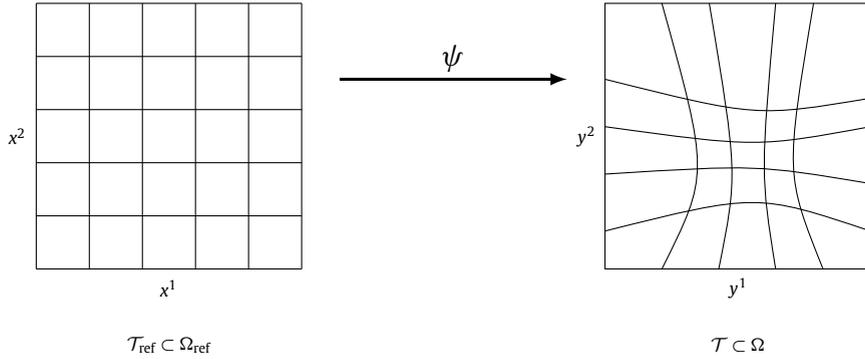


Fig. 1. The uniform $m \times n$ mesh \mathcal{T}_{ref} and the adaptive $m \times n$ mesh $\mathcal{T} = \psi(\mathcal{T}_{\text{ref}})$.

1.4. Outline

Section 2 introduces notation and definitions that will be used throughout the paper. In Section 3, we develop the basic SAM algorithm for static mesh generation, upon which we shall build our dynamic scheme. We show that our scheme is high-order accurate and benchmark the algorithm against the MK scheme. In Section 4, we consider dynamic mesh generation and introduce the perturbation formulation of the MA system. We then perform, in Section 5, a series of challenging mesh generation experiments to demonstrate the capabilities of the scheme. In Section 6, we formulate a simple coupled SAM-ALE scheme for the 2D compressible Euler system, and describe some aspects of our numerical method. In Section 7, we apply SAM-ALE to the Noh and RT test problems and compare the results with low-resolution and high-resolution uniform solutions. Finally, in Section 8, we provide some brief concluding remarks. Three sections are included in the Appendices: the first concerns the C-method regularization for the 2D ALE-Euler system, the second describes a simple boundary smoothing technique, and the third provides a machine comparison test for the purposes of benchmarking our SAM algorithm.

2. Preliminaries

2.1. Domains, meshes, and mappings

The focus of this work is mesh adaptation on 2D rectangles and we provide the mathematical formulation and numerical implementation details of our mesh adaptation strategy in this setting. However, all of our meshing algorithms can be extended to 3D cuboids,⁷ and we show in Section 5.5 results from a mesh generation experiment modeling three-dimensional swirling flow.

Let $\Omega_{\text{ref}} \subset \mathbb{R}^2$ be a reference domain with coordinates $x = (x^1, x^2) \in \Omega_{\text{ref}}$, and given explicitly by the rectangle $\Omega_{\text{ref}} = (x_{\text{min}}^1, x_{\text{max}}^1) \times (x_{\text{min}}^2, x_{\text{max}}^2)$. The outward pointing unit normal vector to the boundary $\partial\Omega_{\text{ref}}$ is defined everywhere on $\partial\Omega_{\text{ref}}$, except at the four corners, and is denoted by ν . The domain Ω_{ref} is also sometimes referred to in the literature as the *logical* or *computational* domain, and in the context of ALE gas dynamics, the *ALE* domain.

We denote by $\Omega \subset \mathbb{R}^2$ the *physical* or *Eulerian* domain, with coordinates $y = (y^1, y^2) \in \Omega$ and boundary $\partial\Omega$. We assume that Ω_{ref} and Ω represent the same mathematical domain i.e. $\Omega_{\text{ref}} = \Omega$. The purpose of using the different notations Ω_{ref} and Ω is to clearly distinguish between functions defined on each of these domains, as we shall explain in the next subsection. We let $\text{id} : \Omega_{\text{ref}} \rightarrow \Omega$ denote the identity map, i.e. $\text{id}(x) = x$.

We discretize Ω_{ref} and Ω with $m + 1$ nodes in the horizontal direction, and $n + 1$ nodes in the vertical direction, and denote by \mathcal{T}_{ref} and \mathcal{T} the grids (or meshes) on each of these domains. Each of these meshes contains $N = m \times n$ cells. The domain Ω_{ref} is discretized uniformly, and we refer to \mathcal{T}_{ref} as the *reference* or *uniform* mesh. The *physical* or *adaptive* mesh \mathcal{T} is *a priori* unknown and will be generated through a meshing scheme. The mesh \mathcal{T} is not assumed to be uniform, but contains the same number of cells and retains the same mesh connectivity structure as the uniform mesh \mathcal{T}_{ref} cf. Fig. 1. The fixed uniform mesh spacing is denoted by $\Delta x = (\Delta x^1, \Delta x^2)$.

The physical domain Ω can also be discretized uniformly with a uniform mesh \mathcal{U} . Since $\Omega_{\text{ref}} = \Omega$, the meshes \mathcal{U} and \mathcal{T}_{ref} are identical. We stress, however, that functions defined on each of these meshes are very different.

⁷ In fact, our algorithms can also be applied in arbitrary complex geometry (see Fig. 19 for a preliminary result), though their numerical implementations are more involved.

The mesh \mathcal{T} will be the image of \mathcal{T}_{ref} under the action of a suitable map $\psi : \Omega_{\text{ref}} \rightarrow \Omega$. The map ψ is bijective, continuously differentiable, and has a continuously differentiable inverse $\psi^{-1} : \Omega \rightarrow \Omega_{\text{ref}}$ i.e. ψ is a smooth diffeomorphism. Our SAM scheme solves for the map ψ by prescribing its Jacobian determinant, as we shall explain in Sections 3 and 4. Nodes in \mathcal{T} on the boundary $\partial\Omega$ will be allowed to move tangential to the boundary, with the exception of the four nodes at the corners of Ω , which must remain fixed.

In the dynamic setting, we consider ψ to be a time-dependent map $\psi : \Omega_{\text{ref}} \times [0, T] \rightarrow \Omega$ where, for each $t \in [0, T]$, the map $\psi(\cdot, t) : \Omega_{\text{ref}} \rightarrow \Omega$ is a smooth diffeomorphism with prescribed Jacobian cf. Fig. 3.

2.2. Eulerian and ALE variables

A *physical* or *Eulerian* function (scalar, vector-valued, or tensor) is defined on Ω and denoted with the upright mathematical font $f : \Omega \rightarrow \mathbb{R}^k$. For a time-dependent function $f : \Omega \times [0, T] \rightarrow \mathbb{R}^k$ we shall write $f(y, t)$.

Since ψ maps Ω_{ref} to Ω , we write $y = \psi(x, t)$ for $(x, t) \in \Omega_{\text{ref}} \times [0, T]$. Given an Eulerian variable $f : \Omega \rightarrow \mathbb{R}^k$, we define its *computational* or *ALE* counterpart $f : \Omega_{\text{ref}} \times [0, T] \rightarrow \mathbb{R}^k$ by

$$f(x, t) = [f \circ \psi](x, t) = f(\psi(x, t), t), \quad \forall (x, t) \in \Omega_{\text{ref}} \times [0, T]. \tag{2}$$

We shall also denote the function composition in (2) by $f \circ \psi$. When there is no confusion, we omit the function arguments and write f or f .

In the discrete setting, computational variables are defined at the nodal points of the uniform reference mesh \mathcal{T}_{ref} . Physical/Eulerian variables, on the other hand, can be defined on either the adaptive mesh $\mathcal{T}(t)$ or the uniform mesh \mathcal{U} on Ω .

2.3. Derivatives and important geometric quantities

We denote spatial derivatives on Ω_{ref} and Ω by

$$\partial_i = \frac{\partial}{\partial x_i} \quad \text{and} \quad D_i = \frac{\partial}{\partial y_i},$$

respectively. Higher order derivatives are then denoted in the standard fashion, e.g. $\partial_{ij} = \partial_i \partial_j$. We use the notation $\nabla = (\partial_1, \partial_2)^T$ and $D = (D_1, D_2)^T$ for the gradient operators with respect to x and y coordinates, respectively. The Laplacian operator on Ω_{ref} is $\Delta = (\partial_1^2 + \partial_2^2)$. The operator Δ should not be confused with the discrete uniform mesh spacing Δx^i .

The time derivative of a function f is written as $\partial_t f$, or sometimes with the subscript notation f_t . Throughout, we shall use Einstein's summation convention wherein a repeated index in the same term indicates summation over all values of that index. We shall also use the standard Kronecker delta symbol δ_j^i .

We now introduce the following important geometric quantities, all defined on $\Omega_{\text{ref}} \times [0, T]$:

$$\mathcal{A} = [\nabla \psi]^{-1} \quad (\text{inverse of the deformation tensor}), \tag{3a}$$

$$\mathcal{J} = \det \nabla \psi \quad (\text{Jacobian determinant}), \tag{3b}$$

$$a = \mathcal{J} \mathcal{A} \quad (\text{cofactor matrix of the deformation tensor}). \tag{3c}$$

We assume that there exists $\varepsilon > 0$ such that

$$\mathcal{J}(x, t) \geq \varepsilon > 0, \quad \text{for every } (x, t) \in \Omega_{\text{ref}} \times [0, T].$$

Thus, the Jacobian determinant in 2D reads

$$\mathcal{J}(x, t) = \partial_1 \psi^1 \partial_2 \psi^2 - \partial_1 \psi^2 \partial_2 \psi^1.$$

For a matrix $M = (M_i^j)$, the subscript i indexes the columns of M , while the superscript j indexes the rows.

By explicit computation, we can verify the so-called Piola identity, which states that the columns of the cofactor matrix are divergence-free:

$$\partial_j a_i^j = 0, \quad \text{for } i = 1, 2. \tag{4}$$

Given an Eulerian variable $f(y, t)$ and its ALE counterpart $f(x, t)$, we use the chain rule to compute

$$D_i f(y, t) = \frac{1}{\mathcal{J}(x, t)} a_i^j(x, t) \partial_j f(x, t) = \frac{1}{\mathcal{J}} \partial_j (a_i^j f), \tag{5}$$

where we have used the Piola identity (4) in the second equality. Using (5) and the chain rule again, we have that

$$\partial_t f(y, t) = \partial_t f - \frac{1}{\mathcal{J}} a_i^j \psi_t^i \partial_j f, \tag{6}$$

where $\psi_t(x, t) \equiv \partial_t \psi(x, t)$ is the mesh velocity.

2.4. Computational platform and code optimization

All of the algorithms in this work were coded in Fortran90, and all of the numerical simulations performed were run on a Macbook Pro laptop with an Apple M1 pro processor and 32 GB of RAM. The operating system is macOS Ventura 13.1, and the gfortran compiler is used. The codes for the numerical methods described in the paper are implemented in the same programming framework, but are not otherwise specially optimized, apart from specific calculations described in the paper. The same input, output, and timing routines are used in all of the codes. This consistency allows for a reliable comparison of the different algorithms and their associated imposed computational burdens.

3. Fast static adaptive meshing

3.1. Mathematical formulation of static mesh generation

We construct an adaptive mesh \mathcal{T} as the image of the uniform mesh \mathcal{T}_{ref} under the action of a suitable smooth diffeomorphism $\psi : \Omega_{\text{ref}} \rightarrow \Omega$ cf. Fig. 1. Our objective is to compute the map ψ by prescribing its Jacobian determinant $\mathcal{J}(x) = \det \nabla \psi(x)$. Specifically, given a strictly positive target Jacobian function $G : \Omega \rightarrow \mathbb{R}^+$, the map ψ is found as a solution to the following nonlinear nonlocal Monge-Ampère (MA) equation

$$\begin{cases} \det \nabla \psi(x) = G \circ \psi(x), & x \in \Omega_{\text{ref}} & \text{(a)} \\ \psi(x) \cdot \nu = x \cdot \nu, & x \in \partial \Omega_{\text{ref}} & \text{(b)} \end{cases} \tag{7}$$

with ν the unit outward normal to the boundary $\partial \Omega_{\text{ref}}$.

The function G is a user prescribed or constructed function that compresses the mesh in regions where G is small, and expands the mesh in regions where G is large. Note that G is a physical target Jacobian function defined on the physical domain Ω . Assuming that a map ψ satisfying (7) is found, the function G then describes the size of the cells in \mathcal{T} . Let \mathcal{V} denote a cell in \mathcal{T} , and $\mathcal{V}_{\text{ref}} = \psi^{-1}(\mathcal{V})$ the uniform cell in \mathcal{T}_{ref} mapped to \mathcal{V} by ψ . If G is sufficiently smooth, a Taylor series argument shows that

$$|\mathcal{V}| := \int_{\mathcal{V}} dy = \int_{\mathcal{V}_{\text{ref}}} \det \nabla \psi(x) dx = |\mathcal{V}_{\text{ref}}| \cdot G(\psi(x_c)) + \mathcal{O}(|\Delta x|^2),$$

where x_c denotes the cell center of \mathcal{V}_{ref} . Thus, the value of G in \mathcal{V} is a scaling factor that scales the uniform cell volume $|\mathcal{V}_{\text{ref}}| = \Delta x^1 \Delta x^2$ to the volume $|\mathcal{V}|$, up to some spatially fixed constant of order $\mathcal{O}(|\Delta x|)$.

It is convenient to formulate the problem for the inverse map $\phi = \psi^{-1}$, which is found as a solution to

$$\begin{cases} \det D\phi(y) = \frac{1}{G(y)}, & y \in \Omega & \text{(a)} \\ \phi(y) \cdot \nu = y \cdot \nu, & y \in \partial \Omega. & \text{(b)} \end{cases} \tag{8}$$

For a solution to exist for (7), the function G is required to satisfy the solvability condition

$$\int_{\Omega} \frac{1}{G(y)} dy = \int_{\Omega_{\text{ref}}} \frac{\det \nabla \psi(x)}{G \circ \psi(x)} dx = |\Omega_{\text{ref}}| = |\Omega|. \tag{9}$$

If (9) holds, then the system (7) admits an infintude of solutions. The question then becomes how to construct a solution ψ that is in some sense optimal. Our primary concern in this work is the development of a fast-running algorithm that can be easily implemented within an ALE framework for hydrodynamics simulations. We next describe a simple and efficient procedure for constructing a solution to (7).

3.2. The basic mesh generation procedure

The key to our fast-running algorithm is the reduction of the nonlinear equation (7) to a simple linear Poisson solve and transport equation solve. Our approach is motivated, as in [50,54,31], by the deformation method of Dacorogna and Moser [21]. Specifically, a solution to (7) is obtained by the five step construction provided in Algorithm 1. We refer to this algorithm as SAM or, in the context of time-dependent meshing, static SAM.

Algorithm 1 STATIC SAM.

Step 1: Assume that the physical target Jacobian function $G : \Omega \rightarrow \mathbb{R}^+$ is given and satisfies the solvability condition

$$\int_{\Omega} \frac{1}{G(y)} dy = |\Omega|, \tag{10}$$

and let $F(y) = 1/G(y)$. In practice, we are usually given an auxiliary target Jacobian function $\bar{G} : \Omega \rightarrow \mathbb{R}^+$ that *does not* satisfy (10), and we define G and F by the following normalization procedure:

$$\bar{F}(y) = \frac{1}{\bar{G}(y)} \longrightarrow F(y) = |\Omega| \frac{\bar{F}(y)}{\int_{\Omega} \bar{F}(y) dy} \longrightarrow G(y) = \frac{1}{F(y)}.$$

Step 2: Solve the following linear Poisson equation with homogeneous Neumann boundary conditions for the *potential* $\Phi : \Omega_{\text{ref}} \rightarrow \mathbb{R}$

$$\begin{cases} \Delta \Phi(x) = F \circ \text{id}(x) - 1, & x \in \Omega_{\text{ref}} & \text{(a)} \\ \nabla \Phi(x) \cdot \nu = 0, & x \in \partial \Omega_{\text{ref}} & \text{(b)} \end{cases} \tag{11}$$

Step 3: Define the velocity $\bar{w} : \overline{\Omega_{\text{ref}}} \rightarrow \mathbb{R}^2$ as

$$\bar{w}(x) = \nabla \Phi(x). \tag{12}$$

Step 4: Solve the following system of transport equations for the *flowmap* $\eta : \overline{\Omega_{\text{ref}}} \times [0, 1] \rightarrow \overline{\Omega}$

$$\begin{cases} \partial_{\tau} \eta + w \cdot \nabla \eta = 0, & x \in \overline{\Omega_{\text{ref}}} \text{ and } 0 < \tau \leq 1 & \text{(a)} \\ \eta(x, 0) = x, & x \in \overline{\Omega_{\text{ref}}} \text{ and } \tau = 0 & \text{(b)} \end{cases} \tag{13}$$

where the *transport velocity* $w : \overline{\Omega_{\text{ref}}} \times [0, 1] \rightarrow \mathbb{R}^2$ is defined as

$$w(x, \tau) = \frac{\bar{w}(x)}{\tau + (1 - \tau)F \circ \text{id}(x)}. \tag{14}$$

Step 5: Define $\psi(x) := \eta(x, 1)$. Then ψ solves (7).

3.2.1. *Validity of construction*

The proof that the map ψ constructed according to Algorithm 1 satisfies (7) proceeds as follows. Define the back-to-labels map $\xi : \Omega \times [0, 1] \rightarrow \Omega_{\text{ref}}$ by $\xi(y, \tau) = \eta^{-1}(y, \tau)$. The Eulerian transport equation for η is transformed into a Lagrangian advection equation for ξ :

$$\begin{cases} \partial_{\tau} \xi(y, \tau) = w \circ \xi(y, \tau), & y \in \overline{\Omega} \text{ and } 0 < \tau \leq 1 & \text{(a)} \\ \xi(y, 0) = y, & y \in \overline{\Omega} \text{ and } \tau = 0. & \text{(b)} \end{cases} \tag{15}$$

Note that $\xi|_{\tau=1} = \eta^{-1}|_{\tau=1} = \psi^{-1} = \phi$.

Next, define the quantity

$$\mathcal{R}(y, \tau) = J(y, \tau) [\tau + (1 - \tau)F] \circ \xi(y, \tau),$$

where $J(y, \tau) = \det D\xi(y, \tau)$ and $F = F \circ \text{id}$. We compute

$$\partial_{\tau} \mathcal{R} = \partial_{\tau} J [\tau + (1 - \tau)F] \circ \xi + J [1 - F] \circ \xi + J(1 - \tau) \partial_{\tau} (F \circ \xi).$$

We recall Euler's lemma, which states that $J(y, \tau)$ evolves according to $\partial_{\tau} J = J \text{div } w \circ \xi$. Using (11)(a) and (12), we calculate

$$\text{div } w = \frac{\text{div } \bar{w}}{\tau + (1 - \tau)F} - \frac{(1 - \tau)\bar{w} \cdot \nabla F}{[\tau + (1 - \tau)F]^2} = \frac{F - 1 - (1 - \tau)w \cdot \nabla F}{\tau + (1 - \tau)F},$$

so that

$$\partial_{\tau} J [\tau + (1 - \tau)F] \circ \xi = J [F - 1 - (1 - \tau)w \cdot \nabla F] \circ \xi.$$

Next, we have that

$$\partial_{\tau} (F \circ \xi) = \partial_{\tau} \xi \cdot \nabla F \circ \xi = [w \cdot \nabla F] \circ \xi,$$

where we have used equation (15)(a).

Using the two formulae above, we find that $\partial_{\tau} \mathcal{R} = 0$, so that $F(y) = \mathcal{R}(y, 0) = \mathcal{R}(y, 1) = \det D\phi(y)$ and thus ϕ satisfies (8)(a), which is in turn equivalent to (7)(a). The condition $w(x, \tau) \cdot \nu = 0$ for every $x \in \partial \Omega_{\text{ref}}$ and $0 \leq \tau \leq 1$ ensures that (7)(b) is satisfied. \square

3.2.2. Discussion

The first numerical implementation of the deformation method [50] utilized the no slip boundary conditions $\psi(x) = x$, $\forall x \in \partial\Omega_{\text{ref}}$, rather than the no penetration boundary conditions (7)(b) which permit tangential motion of boundary nodes. The method of proof in the original paper of [21], which includes an analysis of the Poisson problem (11), requires the domain Ω_{ref} to have smooth boundary $\partial\Omega_{\text{ref}}$ and so is not valid for the rectangular domains we consider in this work. A modified method, which avoids the use of the Poisson problem (11) via a direct construction of the deformation velocity field, is provided in [50], but the resulting numerical implementation yields poor quality grids with high levels of distortion [22]. On the other hand, as we shall demonstrate in our numerical experiments, the use of the Poisson equation (11) together with the slip boundary conditions (7)(b) produces smooth grids. Moreover, the arguments in [21] can be modified with the help of elliptic estimates on polygonal domains [33] to show that the procedure outlined in Algorithm 1 yields existence of a solution to (7).

The basic mesh generation scheme Algorithm 1 differs from other deformation methods in the literature, e.g. [54,31], in both its formulation and numerical implementation. Specifically, the use of the transport system (13) avoids costly interpolation procedures required for the solution of the Lagrangian advection equations in other deformation methods, which results in an order of magnitude speed-up. Moreover, our numerical algorithm produces solutions that converge with high-order accuracy, in contrast to other methods which only yield second-order accurate solutions, at best. In the following subsection, we describe in detail the two main steps of Algorithm 1, namely the Poisson solve in Step 2, and the transport equation solve in Step 4.

3.3. Numerical implementation details

3.3.1. FFT-based elliptic solve for Φ

The Poisson problem (11) is solved in frequency space using the Fast Fourier Transform (FFT). The solvability condition (10) is enforced by the normalization procedure described in Algorithm 1 with trapezoidal integration to compute integrals. The RHS of (11)(a) then has zero mean, and a (non-unique) solution to (11) exists. We choose a unique solution Φ with zero mean, enforced in spectral space by zeroing out the first frequency component. The use of FFT requires the forcing G to be periodic; we periodize the problem by doubling the size of the domain in each direction and extending G symmetrically to the extended domain.⁸ In Step 3, the velocity \bar{w} is also computed via FFT.

3.3.2. Boundary conditions and order of convergence

Solutions to the Poisson problem (11) in general have limited regularity due to the presence of corner singularities in the domain, unless the function G satisfies certain compatibility conditions [37]. In this work, we shall assume the stronger Neumann condition $DG(y) \cdot \nu = 0$ for $y \in \partial\Omega$ to ensure high-order convergence of the numerical solution ψ in the limit of zero mesh size. If $DG(y) \cdot \nu \neq 0$, then the symmetric extension of G is not differentiable on the boundary $\partial\Omega$ and is only Lipschitz continuous. In this case, the potential Φ , velocity \bar{w} , and solution ψ all converge with 2nd order accuracy, but the convergence rate of the Jacobian determinant $\mathcal{J}(x)$ and cofactor matrix $a(x)$ is only 1.5.

On the other hand, if the function G does satisfy the Neumann condition $DG(y) \cdot \nu = 0$ for $y \in \partial\Omega$, then the symmetric extension of G is at least twice continuously differentiable, and the quantities $\psi(x)$, $\mathcal{J}(x)$, and $a(x)$ all converge with (at least) 4th order accuracy. We confirm this high order convergence with a numerical example in Section 3.4.2.

For most of the problems we consider in this work, the function G does indeed satisfy the Neumann condition. However, even if the Neumann condition is not satisfied, the errors in the numerical solution are localized to the boundary, and the meshes produced are still of high accuracy and quality. Additionally, boundary smoothing techniques [2,28] can be applied to obtain high order convergence. We implement a simplified version of this technique in Section 3.4.3 and demonstrate that the quantity $\mathcal{J}(x)$ converges with 2nd order accuracy. The details of this boundary smoothing technique are provided in Appendix B.

3.3.3. Numerical solution of the transport equations

The solution η to the transport equations (13) is smooth, and we shall therefore utilize the simple 5th order linear upwind scheme to compute derivatives, with the upwind direction in the r -th coordinate determined based on the sign of w^r . For instance, if $w_{i,j}^1 \geq 0$, then we approximate the derivative $\partial_1 f$ of a function f by

$$[\partial_1 f]_{i,j} = \frac{-2f_{i-3,j} + 15f_{i-2,j} - 60f_{i-1,j} + 20f_{i,j} + 30f_{i+1,j} - 3f_{i+2,j}}{60\Delta x^1} + \mathcal{O}(|\Delta x|^5).$$

For time-integration, we utilize the standard explicit RK4 scheme, which has the associated stability condition

$$\text{CFL}_\tau = \Delta \tau \left(\frac{\|w^1\|_\infty}{\Delta x^1} + \frac{\|w^2\|_\infty}{\Delta x^2} \right) \leq C. \quad (16)$$

⁸ An alternative implementation with the discrete cosine transform can also be used.

Table 1
Jacobian errors \mathcal{E}_2 demonstrating high order convergence of SAM solutions for the circular target Jacobian function (19).

Scheme		Cells					
		32×32	64×64	128×128	256×256	512×512	1024×1024
SAM	Error	2.85×10^{-2}	5.10×10^{-3}	5.96×10^{-4}	3.73×10^{-5}	1.87×10^{-6}	9.89×10^{-8}
	Order	-	2.5	3.1	4.0	4.3	4.2

Accordingly, the adaptive time-step $\Delta\tau$ is chosen via

$$CFL_\tau = \frac{\Delta\tau}{\tau + \frac{(1-\tau)}{\|\mathbf{G}\|_\infty}} \left(\frac{\|\bar{w}^1\|_\infty}{\Delta x^1} + \frac{\|\bar{w}^2\|_\infty}{\Delta x^2} \right). \tag{17}$$

Our numerical experiments have shown that $CFL_\tau = 2$ is sufficient for a stable scheme.

3.4. High-order accuracy and a benchmark computation

3.4.1. Jacobian error metric

Assessing the accuracy and convergence behavior of the numerical solutions ψ produced with SAM requires an error metric. Since the exact solution ψ_{exact} to the scheme described in Algorithm 1 is not known, we shall instead use the L^2 Jacobian error, defined as

$$\mathcal{E}_2 := \|\mathcal{J}(x) - \mathbf{G} \circ \psi(x)\|_{L^2}. \tag{18}$$

We use bicubic interpolation to compute the composition $\mathbf{G} \circ \psi$ in (18) and trapezoidal integration to compute the L^2 integral norm.

3.4.2. High order convergence of solutions

To demonstrate the high order convergence of numerical solutions computed with SAM, we perform a mesh generation experiment on $\Omega = [0, 1]^2$ for the circular target Jacobian function

$$\bar{\mathbf{G}}(y) = 1 - \delta \exp \left\{ - \left| \sigma \left((y^1 - 0.5)^2 + (y^2 - 0.5)^2 - r^2 \right) \right|^2 \right\}, \tag{19}$$

which forces the mesh to resolve in an annular region containing the circle of radius r centered at $(0.5, 0.5)$. The parameters δ and σ control the smallest cell-size and width of the resolving region, respectively. We choose $\delta = 0.75$, $\sigma = 64$, and $r = 0.2$. See Fig. 5 for the meshes associated with a time-dependent version of (19).

We generate a sequence of meshes using SAM for cell resolutions $N = 32^2$ up to $N = 1024^2$. We compute the Jacobian errors \mathcal{E}_2 given by (18), with the Jacobian determinant \mathcal{J} approximated using 4th order central differencing (CD4). The errors provided in Table 1 show that SAM solutions exhibit the expected 4th order accuracy. We note that, to the best of our knowledge, all other grid generation schemes are at best 2nd order accurate.

3.4.3. Benchmarking against the MK mesh generation scheme

Now, we perform a numerical experiment to benchmark SAM against the MK mesh generation scheme [22], a brief description of which is provided in Appendix C. The test problem [22] we consider is as follows: the domain is $\Omega = [0, 1]^2$, and the target Jacobian function is

$$\bar{\mathbf{G}}(y) = 2 + \cos(8\pi r), \tag{20}$$

where $r = \sqrt{(y^1 - 0.5)^2 + (y^2 - 0.5)^2}$ is the radial coordinate.

We compute a sequence of meshes for $N = 16^2$ up to $N = 256^2$ using SAM, and calculate the L^2 Jacobian errors \mathcal{E}_2 . For the purposes of consistency with [22], we use a slightly different formula to compute \mathcal{E}_2 (see equations (46)-(52) in [22]). In particular, 2nd-order differencing is used to calculate the Jacobian and, as such, we expect only 2nd order convergence of the errors \mathcal{E}_2 . Consequently, we instead use the 3rd order linear upwind scheme for the transport equation solve. The pseudo-time step is set according to (17) with $CFL_\tau = 8$.

The errors are listed in Table 2, along with the errors for the MK scheme obtained from [22]. The function $\bar{\mathbf{G}}$ is radially symmetric, and consequently does not satisfy the Neumann condition $D\mathbf{G} \cdot \nu \neq 0$. As such, the resulting solutions computed with SAM do not display 2nd order accuracy in the limit $N \rightarrow \infty$, though, as shown in Table 2, the order of convergence only degrades to approximately 1.75 for the resolutions considered.

Nonetheless, we shall additionally consider a modified version of this test problem in which the function $\bar{\mathbf{G}}(y)$ in (20) is replaced by the function $\bar{\mathbf{G}}^*(y)$, where $\bar{\mathbf{G}}^*(y)$ is such that $D\bar{\mathbf{G}}^* \cdot \nu = 0$ on $\partial\Omega$. The function $\bar{\mathbf{G}}^*$ is equal to $\bar{\mathbf{G}}$ in the interior of Ω , but is mollified with an appropriate cut-off function in a small region near the boundary $\partial\Omega$ to enforce the Neumann

Table 2

Comparison of L^2 Jacobian errors and convergence rates for the MK and SAM schemes applied to (20). The data for the MK scheme is taken from Table 1 of [22].

Scheme		Cells				
		16×16	32×32	64×64	128×128	256×256
MK	Error	9.64×10^{-2}	2.80×10^{-2}	5.78×10^{-3}	1.46×10^{-3}	3.67×10^{-4}
	Order	–	1.78	2.28	1.99	1.99
SAM with \bar{G}	Error	6.54×10^{-2}	2.05×10^{-2}	7.82×10^{-3}	2.00×10^{-3}	5.96×10^{-4}
	Order	–	1.68	1.39	1.96	1.75
SAM with \bar{G}^*	Error	2.30×10^{-2}	1.44×10^{-2}	5.46×10^{-3}	1.25×10^{-3}	3.25×10^{-4}
	Order	–	0.68	1.40	2.12	1.94

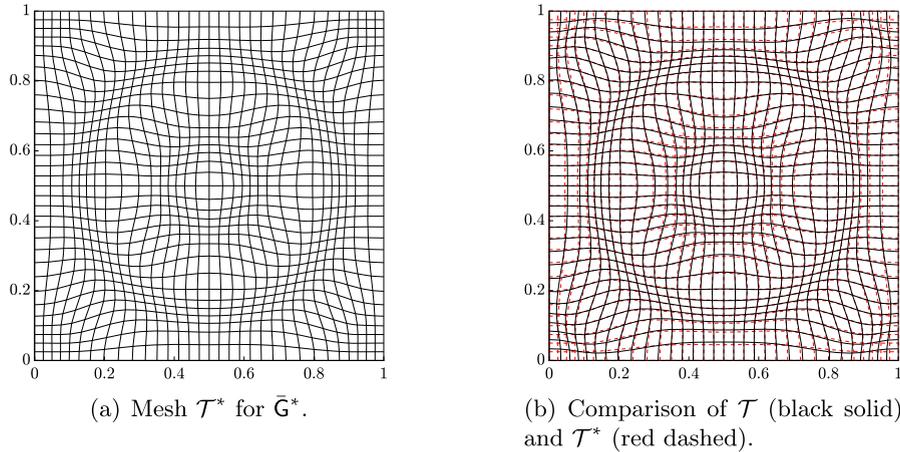


Fig. 2. SAM algorithm with boundary smoothing for the radial sinusoidal target function (20). Figure (a) is the 32×32 cell mesh \mathcal{T}^* produced for the modified target Jacobian \bar{G}^* , and Figure (b) is a comparison of the mesh \mathcal{T} without boundary smoothing for the target Jacobian \bar{G} (black solid), and the mesh \mathcal{T}^* for \bar{G}^* (red dashed). (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 3

CPU runtimes for the MK scheme vs the SAM scheme. The results for the MK scheme are taken from Table 1 of [22] then divided by 2.2 to account for machine difference.

Scheme		Cells				
		16×16	32×32	64×64	128×128	256×256
MK	T_{CPU}	0.045	0.182	0.591	2.227	8.636
	T_{CPU}	0.0012	0.0017	0.004	0.013	0.042
SAM	speed-up factor	47	131	160	174	190

condition (see Appendix B for further details). The mesh \mathcal{T}^* produced using SAM with \bar{G}^* is shown in Fig. 2(a), and a comparison with the mesh \mathcal{T} for \bar{G} is shown in Fig. 2(b), from which it can be seen that the two meshes are very similar: they are nearly identical in the interior, with small differences near the boundary. While the solutions for \bar{G} do not attain the full 2^{nd} order accuracy, the solutions for \bar{G}^* do. The SAM solutions for \bar{G}^* have smaller errors than MK across all the resolutions considered. Moreover, the SAM solutions for \bar{G}^* display 2^{nd} order accuracy as N increases.⁹ We also find that the L^2 mesh displacement $\|\psi(x) - x\|_{L^2} \approx 0.0178$ is comparable to the value of 0.0174 for MK reported in Table 1 of [22].

Next, we benchmark the computational efficiency of our SAM scheme against the MK scheme. We list in Table 3 the CPU runtimes for the MK scheme and the SAM scheme, where the MK runtimes are taken from Table 1 of [22]. To account for the different machines on which the MK and SAM schemes were run on, we divide the MK runtimes by 2.2, where the factor of 2.2 is determined from a machine comparison experiment, the details of which are provided in Appendix C. We then list the speed-up factor of the SAM scheme over the MK scheme in the final row of Table 3. We find that our SAM scheme is almost 200 times faster than the MK scheme at $N = 256^2$ cell resolution. We also note that the CPU times reported in [22] do not include the cost of an interpolation call, which is non-negligible at high resolutions.

⁹ We have verified this up to $N = 2048^2$ but for brevity do not show the results here.

4. Fast dynamic adaptive meshing

4.1. Dynamic formulation

Given a time interval $t \in [0, T]$, we seek to construct a time dependent diffeomorphism $\psi : \Omega_{\text{ref}} \times [0, T] \rightarrow \Omega$. We denote the time-dependent mesh on Ω by $\mathcal{T}(t)$, which will be found as the image of \mathcal{T}_{ref} under the action of $\psi(\cdot, t)$. The time-dependent map $\psi : \Omega_{\text{ref}} \times [0, T] \rightarrow \Omega$ is constructed by prescribing, for each $t \in [0, T]$, its Jacobian determinant. Let $G : \Omega \times [0, T] \rightarrow \mathbb{R}^+$ denote a given time-dependent (physical) target Jacobian function. Then ψ satisfies

$$\begin{cases} \det \nabla \psi(x, t) = G \circ \psi(x, t), & (x, t) \in \Omega_{\text{ref}} \times [0, T] & \text{(a)} \\ \psi(x, t) \cdot \nu = x \cdot \nu, & (x, t) \in \partial \Omega_{\text{ref}} \times [0, T] & \text{(b)} \end{cases} \quad (21)$$

The target Jacobian function G must satisfy, for each $t \in [0, T]$, the following integral constraint to ensure that (21) has a solution:

$$\int_{\Omega} \frac{1}{G(y, t)} dy = \int_{\Omega_{\text{ref}}} \frac{\det \nabla \psi(x, t)}{G \circ \psi(x, t)} dx = |\Omega|. \quad (22)$$

4.1.1. Temporal discretization

We uniformly discretize the time domain $[0, T]$ into K intervals of length Δt and set $t_k = k\Delta t$ for $k = 0, 1, \dots, K$. Denote $G_k := G(\cdot, t_k)$, $\psi_k := \psi(\cdot, t_k)$, and $\mathcal{T}_k = \mathcal{T}(t_k)$. Then each $\psi_k : \Omega_{\text{ref}} \rightarrow \Omega$ is a diffeomorphism satisfying

$$\begin{cases} \det \nabla \psi_k(x) = G_k \circ \psi_k(x), & x \in \Omega_{\text{ref}} & \text{(a)} \\ \psi_k(x) \cdot \nu = x \cdot \nu, & x \in \partial \Omega_{\text{ref}} & \text{(b)} \end{cases} \quad (23)$$

with each target Jacobian function $G_k : \Omega \rightarrow \mathbb{R}^+$ satisfying the integral constraint

$$\int_{\Omega} \frac{1}{G_k(y)} dy = \int_{\Omega_{\text{ref}}} \frac{\det \nabla \psi_k(x)}{G_k \circ \psi_k(x)} dx = |\Omega|. \quad (24)$$

4.1.2. Algorithmic complexity

The simplest possible strategy for (23) is to compute each map ψ_k using static SAM. Our numerical experiments indicate that static SAM is highly efficient for a single mesh generation call. However, for unsteady fluids simulations which require dynamic meshing at every time-step, the use of static SAM can be expensive at high resolutions due to the computational bottleneck in the transport equation solve stage.

Specifically, suppose that the target Jacobian function has large deviation from the identity i.e. $\|1/G - 1\|_{L^\infty} \gg 1$. Then the associated potential Φ solving (11) has large gradients, and the flowmap velocity (14) will therefore be large in magnitude. Consequently, many pseudo-time steps will be required in the transport equation solve to preserve stability and accuracy of the computed numerical solution for $\eta(x, \tau)$. In particular, the stability condition (16) forces the pseudo-time step to decay like $\mathcal{O}(N^{-1/2})$, so that the overall complexity of the SAM algorithm is $\mathcal{O}(N^{3/2})$. For large N , this can become prohibitively computationally expensive. In the next section, we resolve this issue via a novel reformulation of (23).

4.2. Reformulation in terms of near-identity maps

4.2.1. The perturbation formulation

Assume that we are given the map ψ_k and the target Jacobian functions G_k and G_{k+1} , and suppose that we wish to compute the map ψ_{k+1} . Rather than computing the map ψ_{k+1} directly by solving (23), we instead solve for the *perturbation map* $\delta\psi_{k+1} : \Omega_{\text{ref}} \rightarrow \Omega_{\text{ref}}$ defined implicitly by

$$\psi_{k+1}(x) = \psi_k \circ \delta\psi_{k+1}(x). \quad (25)$$

That is, we suppose that the map ψ_{k+1} can be found as the image of ψ_k acting on a near-identity transformation $\delta\psi_{k+1}$ on the reference domain Ω_{ref} (see Fig. 3).

By the chain rule and inverse function theorem, we find that $\delta\psi_{k+1}$ satisfies

$$\begin{cases} \det \nabla \delta\psi_{k+1}(x) = P_{k+1} \circ \delta\psi_{k+1}(x), & x \in \Omega_{\text{ref}} & \text{(a)} \\ \delta\psi_{k+1}(x) \cdot \nu = x \cdot \nu, & x \in \partial \Omega_{\text{ref}} & \text{(b)} \end{cases} \quad (26)$$

with the function $P_{k+1} : \Omega_{\text{ref}} \rightarrow \mathbb{R}^+$ defined as

$$P_{k+1}(x) = \left(\frac{G_{k+1}}{G_k} \right) \circ \psi_k(x). \quad (27)$$

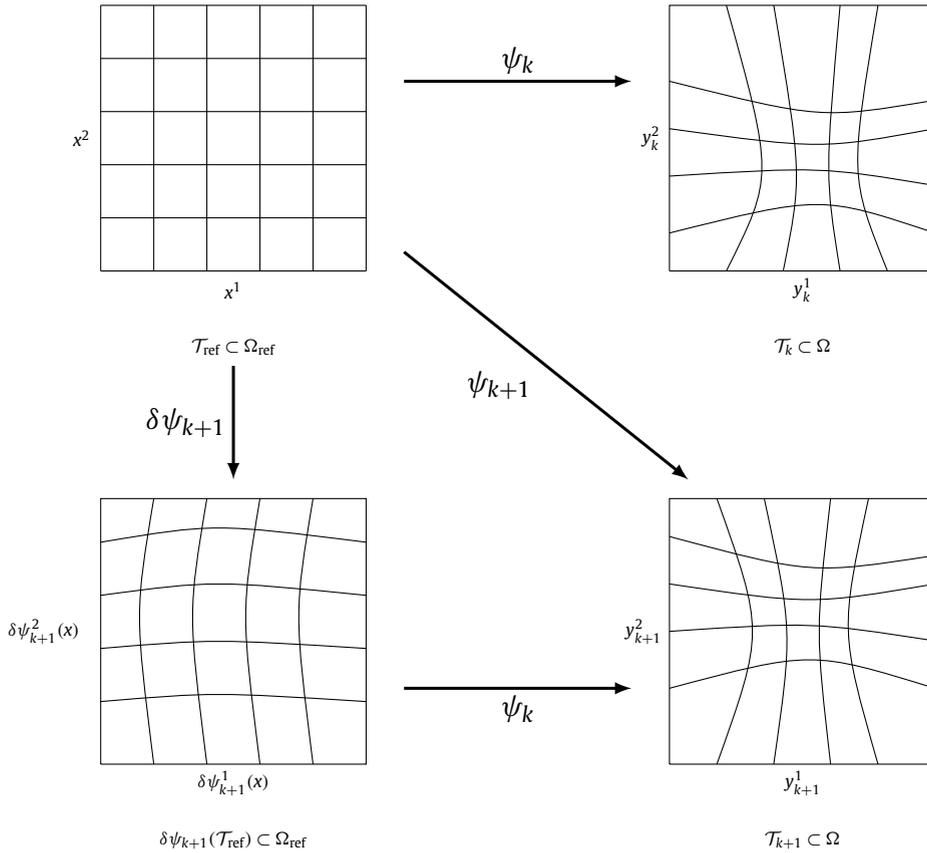


Fig. 3. Schematic of the meshes, and the maps between them, for dynamic mesh generation.

The system (26) is of exactly the same form as (7), and the identical solution procedure described in Section 3.2 for static mesh generation can therefore be used to find the solution $\delta\psi_{k+1}$. Then ψ_{k+1} is computed according to (25). A complete description of the dynamic SAM scheme is provided in Algorithm 2.

Algorithm 2 DYNAMIC SAM.

Step 1: Set $t = 0$. Given an initial target Jacobian function $G_0 : \Omega \rightarrow \mathbb{R}^+$, compute the initial diffeomorphism ψ_0 according to the static solution scheme from Section 3.2.

Step 2: For $t = t_{k+1}$, assume that we are given the following: the map ψ_k and target Jacobian function $G_k : \Omega \rightarrow \mathbb{R}^+$, both from the previous time step $t = t_k$, and the target Jacobian function $G_{k+1} : \Omega \rightarrow \mathbb{R}^+$ at the current time level. Define $P_{k+1} : \Omega_{\text{ref}} \rightarrow \mathbb{R}^+$ by (27), and assume that it satisfies the solvability condition

$$\int_{\Omega_{\text{ref}}} \frac{1}{P_{k+1}(x)} dx = \int_{\Omega_{\text{ref}}} \left(\frac{G_k}{G_{k+1}} \right) \circ \psi_k(x) dx = |\Omega_{\text{ref}}|. \tag{28}$$

In general, we will be given target Jacobian functions \bar{G}_k and \bar{G}_{k+1} such that the corresponding $\bar{P}_{k+1} = (\bar{G}_{k+1}/\bar{G}_k) \circ \psi_k$ does not satisfy (28). In this case, we define P_{k+1} according to the following normalization procedure:

$$\bar{Q}_{k+1}(x) = \frac{1}{\bar{P}_{k+1}(x)} \rightarrow Q_{k+1}(x) = |\Omega| \frac{\bar{Q}_{k+1}(x)}{\int_{\Omega_{\text{ref}}} \bar{Q}_{k+1}(x) dx} \rightarrow P_{k+1}(x) = \frac{1}{Q_{k+1}(x)}.$$

Step 3: Solve (26) for the perturbation map $\delta\psi_{k+1} : \Omega_{\text{ref}} \rightarrow \Omega_{\text{ref}}$ using the solution procedure in Section 3.2.

Step 4: Define $\psi_{k+1} : \Omega_{\text{ref}} \rightarrow \Omega$ by (25). If $t_{k+1} = T$, then stop; otherwise, set $t = t_{k+2}$, and return to **Step 2**.

4.2.2. Discussion

The key to the efficiency of dynamic SAM is the reformulation in terms of the perturbation map $\delta\psi_{k+1}$ satisfying (26). Specifically, while it may be that both G_k and G_{k+1} have large deviation from 1 i.e. $\|1/G_k - 1\|_{L^\infty} \gg 1$ and $\|1/G_{k+1} - 1\|_{L^\infty} \gg 1$, we may nonetheless have that $\|1/P_{k+1} - 1\|_{L^\infty} \ll 1$. Indeed, this is the case in ALE simulations, which are naturally constrained by a CFL condition that limits the evolution of the numerical solution over a single time-step.

More precisely, the usual stability condition for the *physical* time step Δt in an (Eulerian) simulation forces the time step Δt to decay like $\Delta t \sim 1/\sqrt{N}$ as $N \rightarrow \infty$, which is a constraint of exactly the same form as the condition (16) on $\Delta \tau$. A Taylor series argument shows that

$$1 - \frac{1}{P_{k+1}(x)} = \Delta t \cdot \frac{\partial_t \mathbf{G}(\psi(x, t_k), t_k)}{\mathbf{G}(\psi(x, t_k), t_k)} + \mathcal{O}(\Delta t^2)$$

$$\implies \|1 - 1/P(\cdot, t)\|_{L^\infty} = \mathcal{O}(\Delta t) = \mathcal{O}(1/\sqrt{N}), \tag{29}$$

since $\partial_t \mathbf{G} \sim \mathcal{O}(1)$. Since the stability condition for the pseudo-time step $\Delta \tau$ scales according to (16), and $\|w\|_{L^\infty} = \mathcal{O}(1/\sqrt{N})$ by (29), we see that $\Delta \tau = \mathcal{O}(1)$ i.e. the pseudo-time step $\Delta \tau$ can be kept fixed across resolutions N , resulting in a dynamic SAM algorithm with optimal complexity.

We emphasize here that our perturbation formulation (26) differs from the methods considered in [67,32] in an important way. In particular, our perturbation map $\delta\psi_{k+1}$ is a near identity transformation of the *uniform reference domain* Ω_{ref} , whereas the schemes in [67,32] define a perturbation map via $\bar{\psi}_{k+1} = \delta\bar{\psi}_{k+1} \circ \bar{\psi}_k$, rather than through (25). This means that the $\bar{\psi}_k$ solutions in [67,32] are different from our ψ_k SAM solutions. Moreover, the equation for $\delta\bar{\psi}_{k+1}$ is posed on the deformed mesh \mathcal{T}_k ; this means that the solvers for the Poisson problem and transport equation must be appropriately modified at each time-step t_k , requiring, for example, the costly recalculation of the stiffness and mass matrices. Consequently, SAM is simpler, faster, and more accurate than the schemes in [67,32]. For instance, the scheme in [32] has order of accuracy 1.5, whereas our dynamic SAM solutions converge with 4th order accuracy if the data is sufficiently smooth. Additionally, the interpolation routine in [32] requires $\mathcal{O}(N^{1.5})$ grid searching on deformed grids, in contrast to our $\mathcal{O}(N)$ SAM algorithm.

4.3. Restarted dynamic mesh generation

The perturbation formulation (26), by design, follows the time history of $\psi(x, t)$. That is to say, the solution $\psi(x, t)$ at time $t = t_k$ depends upon the solution for all $t < t_k$. As such, numerical solutions to (26) are susceptible to increasing grid distortion and mesh tangling, a common ailment of Lagrangian-type methods. To mitigate this issue, we can periodically *restart* the dynamic mesh generation by computing at time $t = t_k$ the map ψ_k directly with static SAM, rather than with dynamic SAM. In this way, the greater efficiency of dynamic SAM is utilized, while grid distortion errors are controlled with the use of static SAM, thereby preventing mesh tangling. The restarting criterion is chosen as $\lambda_k > \Lambda \lambda_{\text{ref}}$, where λ_k is the L^1 grid distortion at time step t_k , λ_{ref} is a “reference” grid distortion (defined in Algorithm 3), and Λ is a user prescribed parameter. A description of our restarted dynamic SAM scheme is provided in Algorithm 3.

Algorithm 3 RESTARTED DYNAMIC SAM.

- Step 0:** Choose the maximum grid distortion parameter $\Lambda > 1$.
- Step 1:** Set $t = 0$. Given an initial target Jacobian function $G_0 : \Omega \rightarrow \mathbb{R}^+$, compute the initial diffeomorphism ψ_0 according to the static solution scheme from Section 3.2. Let λ_{ref} be the (reference) L^1 grid distortion of the adaptive mesh \mathcal{T}_0 , computed according to (30).
- Step 2:** For $t = t_{k+1} > 0$, compute the average grid distortion of the map ψ_k

$$\lambda_k := \left\| \frac{1}{2} \text{Tr} \left(\nabla \psi_k \nabla \psi_k^T \right) \right\|_{L^1}. \tag{30}$$

- Step 3:** If $\lambda_k > \Lambda \lambda_{\text{ref}}$, then compute the map ψ_{k+1} using static SAM Algorithm 1 and recalculate λ_{ref} according to (30). Otherwise, compute ψ_{k+1} using dynamic SAM Algorithm 2. If $t_{k+1} = T$, then stop; otherwise, set $t = t_{k+2}$, and return to **Step 2**.
-

5. Dynamic mesh generation experiments

In this section, we present and discuss the results of several dynamic mesh generation experiments conducted with the static, dynamic, and restarted SAM algorithms. Unless otherwise stated, all experiments are conducted on the unit square $\Omega = [0, 1]^2$ with an equal number of cells in the horizontal and vertical directions $m = n = \sqrt{N}$.

5.1. Static mesh with large zoom-in factor

This static test problem demonstrates the ability of dynamic SAM to generate smooth meshes with large zoom-in factors which, in practical applications, can be used to track very small scale structures with only a few total number of cells. On the other hand, when the target function G has large gradients (as is the case for large zoom-in meshes), numerical errors in the Poisson solve often lead to poor quality grids containing non-convex elements [22,32].

As an example, consider the circular target Jacobian function $G_\delta(y)$ given by (19) with $\sigma = 64$, $r = 0.2$, and $\delta \in [0, 1)$. More generally, we have a family of target functions $\{G_\delta(y)\}_{0 \leq \delta < 1}$ parameterized by δ , with each such G_δ forcing the mesh to resolve around some given curve (see equation (59)). The zoom-in parameter δ determines the zoom-in factor $\Upsilon = 1/\min \mathcal{J}$ of the adaptive mesh \mathcal{T} , and thus the smallest scales that can be represented on \mathcal{T} . When $\delta = 0$ (uniform mesh) we have

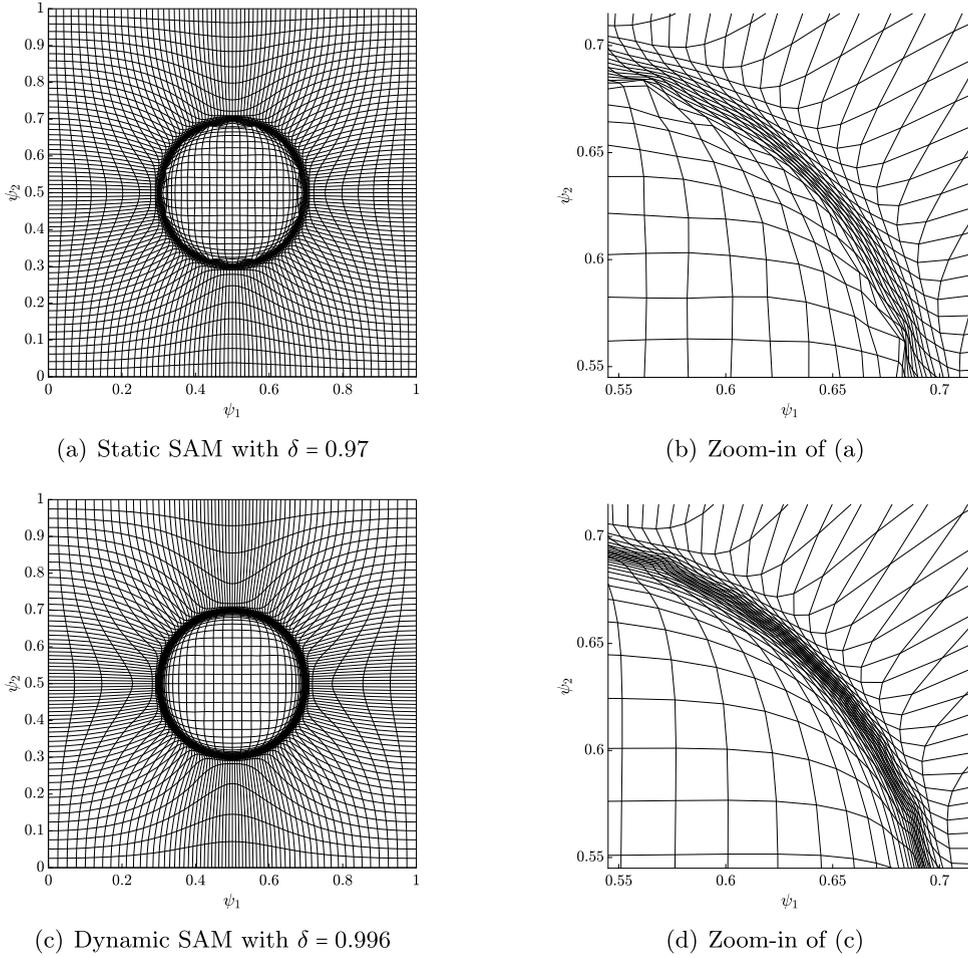


Fig. 4. Test Problem 5.1 demonstrating smooth large zoom-in meshing using dynamic SAM. Shown are the 64^2 cell meshes with large zoom-in parameter δ for the circular target Jacobian function (19). Figure (a) is the poor quality mesh containing non-convex elements produced with static SAM with $\delta = 0.97$, and (b) is a zoom-in of (a) near the refining region. Figure (c) is the smooth large zoom-in mesh produced with dynamic SAM with $\delta = 0.996$ and with smallest cell 100 times smaller than a uniform cell, and (d) is a zoom-in.

$\Upsilon = 1$. As δ increases, so does Υ , with smaller and smaller scales captured on \mathcal{T} . As $\delta \rightarrow 1$, $\Upsilon \rightarrow \infty$ and the mapping is degenerate at $\delta = 1$. From the point of view of efficiency, we would like to have Υ large since we then require few total number of grid points. Moreover, for unstable RT problems that have evolving interfaces with large curvature, it is essential that we construct adaptive meshes with large enough Υ such that they can capture the small-scale vortical structures of the flow. As such, we often want to choose $\delta \approx 1$, and refer to the associated meshes as “large zoom-in” meshes.

While the continuous mapping ψ is non-degenerate for all $0 \leq \delta < 1$, in practice numerical errors in static SAM will produce folded grids if δ is sufficiently close to 1. That is, for each N , there exists a corresponding δ_{\max} such that the grids produced with static SAM for $\delta > \delta_{\max}$ contain non-convex elements. An example of such a grid is shown for $N = 64^2$ and $\delta = 0.97$ in Figs. 4(a) and 4(b). The function G_δ is such that $\|1 - 1/G_\delta\|_{L^\infty} \approx \frac{1}{1-\delta} \rightarrow \infty$ as $\delta \rightarrow 1$. When $\delta \approx 1$, large errors in the numerical solution of the Poisson problem lead to grids with non-convex elements.

Dynamic SAM provides a simple method for producing smooth grids with $\delta \approx 1$. We define the time-dependent function

$$G(y, s) = (1 - s) + sG_\delta(y). \tag{31}$$

Then (31) linearly interpolates between 1 at $s = 0$ and G_δ at $s = 1$, and applying dynamic SAM with sufficiently many time steps Δs yields smooth grids with no non-convex elements. As an example, we set $\Delta s = 0.05$ and construct a 64^2 cell mesh using Algorithm 2 with $\delta = 0.996$ in (19). The resulting grid, shown in Figs. 4(c) and 4(d), is smooth with $\Upsilon \approx 100$. In Section 7, we consider large zoom-in meshing for the more complicated Rayleigh-Taylor test.

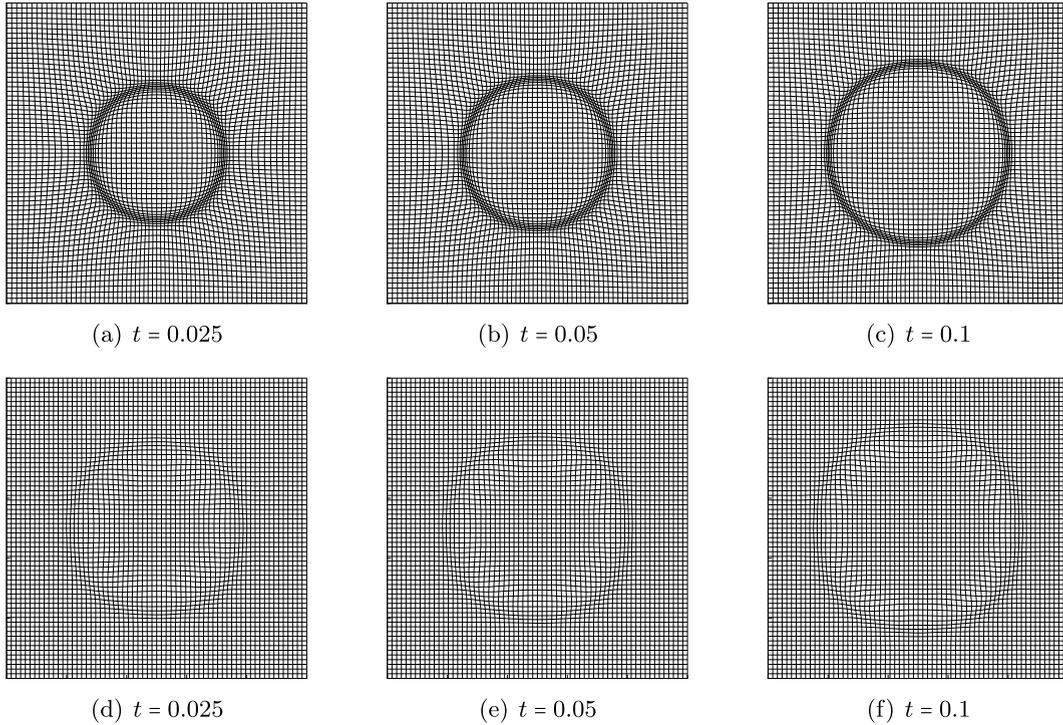


Fig. 5. Test Problem 5.2: tracking a propagating circular front modeling a shock wave. Shown are the 64^2 cell meshes produced using dynamic SAM for the circular target Jacobian function (32). The top row shows the adaptive meshes $\mathcal{T}(t)$, while the bottom row shows the corresponding “perturbation meshes” $\delta\psi_k(\mathcal{T}_{\text{ref}})$.

5.2. Propagating circular front

5.2.1. Problem description

Our first dynamic mesh generation experiment tracks a circular front propagating radially outwards with radial velocity 1. The time-dependent target Jacobian function is defined as

$$\bar{G}(y, t) = 1 - \delta \exp \left\{ - \left| \sigma \left[(y^1 - 0.5)^2 + (y^2 - 0.5)^2 - r(t)^2 \right] \right|^2 \right\}. \quad (32)$$

The parameters are chosen as $\delta = 0.75$, $\sigma = 64$, and the radius is $r(t) = 0.2 + t$. We generate a sequence of meshes for $0 \leq t \leq 0.1$.

The choice of time step Δt depends upon N as

$$\Delta t = \frac{0.64}{2\sqrt{N}}. \quad (33)$$

This choice of scaling for Δt is motivated by the CFL condition. Since the radial velocity of the propagating front is 1, we can estimate that the CFL number associated with (33) is 0.64.

5.2.2. Results

The 64^2 cell adaptive meshes $\mathcal{T}(t)$ for (32) are shown in the top row of Fig. 5 at various times t . The computed meshes \mathcal{T}_k are smooth and are correctly resolved around the evolving circular front. The meshes $\delta\psi_k(\mathcal{T}_{\text{ref}})$ are shown at the same times in the bottom row of Fig. 5; from these figures, it is clear that $\delta\psi_k(\mathcal{T}_{\text{ref}})$ is a near-identity transformation of the uniform mesh \mathcal{T}_{ref} . For this problem, the function G is such that $\|1 - 1/G(\cdot, t)\|_{L^\infty} \approx 2.43$, whereas the perturbation density P is such that $\|1 - 1/P(\cdot, t)\|_{L^\infty} \approx 0.3$.

5.2.3. Comparison with static SAM

Next, we conduct a grid resolution study with N ranging from $N = 32^2$ to $N = 512^2$, and compare the results of dynamic SAM with those of static SAM. The Jacobian errors \mathcal{E}_2 at the final time $t = 0.1$ are shown in Table 4. Both schemes exhibit 4th order accuracy, as expected, but the dynamic SAM solutions have smaller errors. This is due to the higher accuracy of the Poisson solve in the dynamic method vs the static method.

Table 4

Test Problem 5.2: tracking a propagating circular front. We list the L^2 Jacobian errors \mathcal{E}_2 at $t = 0.1$, convergence rates, and total CPU runtimes for static and dynamic SAM. The results confirm that dynamic SAM produces high order accurate solutions and is of optimal complexity.

Scheme		Cells					
		32×32	64×64	128×128	256×256	512×512	1024×1024
Static SAM	\mathcal{E}_2	4.23×10^{-2}	1.15×10^{-2}	1.22×10^{-3}	9.29×10^{-5}	4.98×10^{-6}	2.65×10^{-7}
	Order	–	1.9	3.2	3.7	4.2	4.2
	T_{CPU} (sec)	0.006	0.049	0.59	7.26	112.8	1663
Dynamic SAM	\mathcal{E}_2	4.05×10^{-2}	6.66×10^{-3}	6.18×10^{-4}	4.00×10^{-5}	2.23×10^{-6}	1.33×10^{-7}
	Order	–	2.6	3.4	3.9	4.2	4.1
	T_{CPU} (sec)	0.017	0.101	0.869	7.31	65.2	582
	speed-up factor	0.33	0.48	0.68	0.99	1.73	2.86

At low resolutions, the dynamic SAM runtimes are greater than those for static SAM. This is due to the interpolation required in the dynamic SAM algorithm. On the other hand, static SAM is of complexity $\mathcal{O}(N^{3/2}/\Delta t) = \mathcal{O}(N^2)$, whereas dynamic SAM is of optimal complexity $\mathcal{O}(N/\Delta t) = \mathcal{O}(N^{3/2})$. For this test, dynamic SAM becomes more efficient than static SAM at $N = 256^2$.

5.3. Uniformly rotating patch

5.3.1. Problem description

Our next mesh generation experiment assesses the performance of SAM for target Jacobian functions of the form

$$\bar{G}(y, t) = \frac{1}{1 + M \exp \left\{ - \left(\sigma \left[(y^1 - 0.5 - r \cos(2\pi t))^2 + (y^2 - 0.5 - r \sin(2\pi t))^2 - R^2 \right] \right)^2 \right\}}. \tag{34}$$

Equation (34) forces the mesh to concentrate nodes within a uniformly rotating (with angular velocity $\omega = 2\pi$) circular patch of radius $R > 0$, whose center is a distance $r \geq 0$ from $(0.5, 0.5)$. The constant $M \geq 0$ determines the zoom-in factor, and σ controls the width of the transition region from fine to coarse scale of the mesh.

5.3.2. Comparison with the schemes in [71]

The case $M = 5$, $\sigma = 50$, $r = 0.25$, and $R = 0.1$ in (34) corresponds to a test problem from [71]. Therein, the authors compare four different mesh generation methods and conclude that the so-called Parabolic Monge-Kantorovich method (PMKP) is the best method among the four for (34), both in terms of accuracy as well as efficiency. The PMKP method is similar to the MK scheme, but replaces the nonlinear Newton-Krylov solver in MK with a parabolization (in pseudo-time τ) and time-stepping until a steady state is reached. The solution in PMKP is only found in the asymptotic limit $\tau \rightarrow \infty$, whereas the SAM solution is computed at pseudo-time $\tau = 1$. Moreover, the explicit integration of the parabolic PDE requires that the pseudo-time step scales like $\Delta \tau \sim \frac{1}{N}$ to ensure 2nd order convergence. In contrast, static SAM requires only that $\Delta \tau \sim \frac{1}{\sqrt{N}}$, while for dynamic SAM we can keep $\Delta \tau = \mathcal{O}(1)$.

We set $N = 40^2$, $\Delta t = 0.01$, and generate a sequence of meshes for $0 \leq t \leq 1$. The adaptive meshes generated with static, dynamic, and restarted SAM are shown in Fig. 6 at various times t . The Jacobian errors, mean grid distortion, and cumulative simulation runtimes are provided in Table 5. For the purposes of comparison with [71], we also provide the *mesh fidelity measure* $\hat{\mathcal{E}}_2$, defined by

$$\hat{\mathcal{E}}_2 := \left| \left\| \mathcal{J}(\cdot, t) / G \circ \psi(\cdot, t) \right\|_{L^2} - 1 \right|. \tag{35}$$

The superior accuracy of SAM produces fidelity measures $\hat{\mathcal{E}}_2$ that are an order of magnitude smaller than those produced with the PMKP method (see Tables 6 and 7 in [71]). Moreover, the SAM runtimes are more than two orders of magnitude smaller than the PMKP runtimes provided in [71] e.g. 0.179 sec vs 75 sec for static SAM vs PMKP.

Since static SAM constructs the map ψ directly from the uniform mesh, the meshes at $t = 0.25, 0.5, 0.75, 1.0$ are simply rotated versions of the initial grid. This is confirmed in Table 5, which shows that static SAM produces grids with identical grid quality metrics at these times. Dynamic SAM, on the other hand, necessarily tracks the history of the simulation, and the rotating target Jacobian produces grids with increasing levels of distortion. The Jacobian errors of dynamic SAM are smaller than static SAM for $t = 0.25$ and $t = 0.5$ due to the high accuracy with which the Poisson problem is solved for in the perturbation formulation. For $t > 0.6$, however, grid distortion errors outweigh the improved accuracy for the Poisson solve, and dynamic SAM errors become larger than static SAM errors. The restart criterion parameter in restarted SAM is set as $\Lambda = 1.01$. The mesh restarting controls the grid distortion errors, which in turn prevents the Jacobian errors from

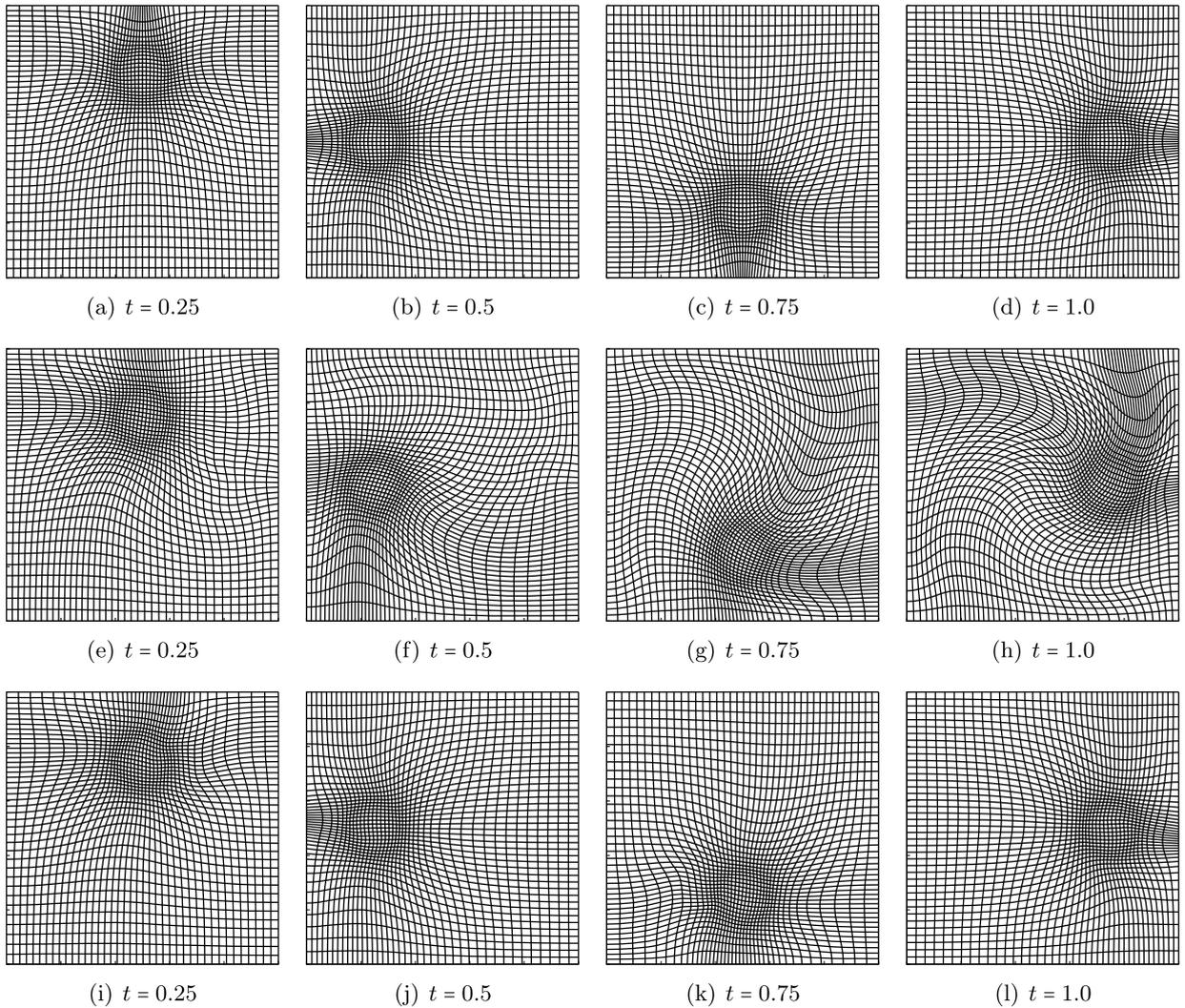


Fig. 6. Test Problem 5.3: tracking a uniformly rotating patch with target function (34). Shown are plots of the 40^2 cell adaptive meshes at various times t . The meshes are produced using static SAM (top), dynamic SAM (middle), and restarted SAM (bottom). Restarted SAM removes the grid distortion errors associated with Lagrangian methods.

growing. As shown in the bottom row of Fig. 6, and confirmed in Table 5, restarted SAM grids are of comparable accuracy and smoothness to static SAM grids

At this low resolution, dynamic SAM is actually slower than the static algorithm. For higher resolutions, however, dynamic SAM is much more efficient than static SAM. To demonstrate this, we repeat the above experiment with $N = 400^2$ cells. For brevity, we do not report the Jacobian errors or L^1 distortion, since the conclusions are similar to the $N = 40^2$ case. The computational runtimes, however, are very different: 1414 sec for static SAM vs 184 sec for dynamic SAM, and 194 sec for restarted SAM. We thus see that restarted SAM combines the best aspects of static and dynamic SAM i.e. smoothness and efficiency, respectively.

5.4. Differential rotation with small scales

5.4.1. Problem description

This dynamic mesh generation test models a Gaussian “blob” deforming under a rotating flow in which the angular velocity is dependent upon the distance from the center of the blob [18]. The time-dependent target Jacobian function is defined as

$$\bar{G}(y, t) = \frac{1}{1 + 4 \exp \left[-r(y)^2 \left(\frac{\cos^2 \theta_0(y, t)}{\sigma_1} + \frac{\sin^2 \theta_0(y, t)}{\sigma_2} \right) \right]}, \tag{36}$$

Table 5

Test Problem 5.3: tracking a uniformly rotating patch. We provide the mesh fidelity measure $\hat{\mathcal{E}}_2$, L^2 Jacobian error \mathcal{E}_2 , L^1 distortion λ , and cumulative CPU runtime T_{CPU} at various times t for the static, dynamic, and restarted SAM schemes. The mesh fidelity measures $\hat{\mathcal{E}}_2$ of SAM solutions are an order of magnitude smaller than those provided in [71]. Additionally, static SAM is more than 400 times faster than the schemes in [71].

Scheme		Time				
		$t = 0$	$t = 0.25$	$t = 0.5$	$t = 0.75$	$t = 1.0$
Static SAM	$\hat{\mathcal{E}}_2$	3.86×10^{-3}				
	\mathcal{E}_2	3.79×10^{-2}				
	λ	1.251	1.251	1.251	1.251	1.251
	T_{CPU} (sec)	0.003	0.048	0.091	0.134	0.179
Dynamic SAM	$\hat{\mathcal{E}}_2$	3.86×10^{-3}	1.13×10^{-3}	1.13×10^{-3}	1.49×10^{-3}	6.65×10^{-3}
	\mathcal{E}_2	3.79×10^{-2}	3.28×10^{-2}	3.49×10^{-2}	4.79×10^{-2}	6.31×10^{-2}
	λ	1.251	1.328	1.544	1.877	2.311
	T_{CPU} (sec)	0.003	0.049	0.094	0.139	0.185
Restarted SAM	$\hat{\mathcal{E}}_2$	3.86×10^{-3}	2.85×10^{-3}	1.18×10^{-3}	3.86×10^{-3}	2.85×10^{-3}
	\mathcal{E}_2	3.79×10^{-2}	3.05×10^{-2}	2.15×10^{-2}	3.79×10^{-2}	3.05×10^{-2}
	λ	1.251	1.252	1.259	1.251	1.252
	T_{CPU} (sec)	0.003	0.048	0.093	0.138	0.183

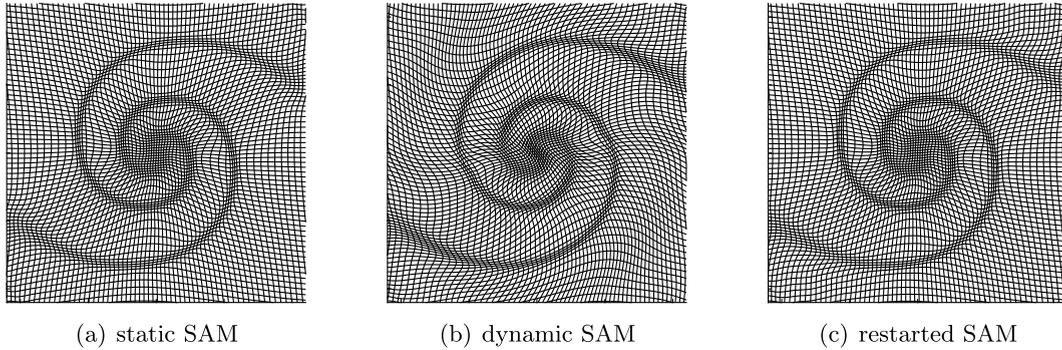


Fig. 7. Test Problem 5.4: tracking small scale vortical structures in flows with differential rotation using (36). Shown are zoomed in plots of the 128^2 cell adaptive meshes at $t = 90$. SAM produces smooth meshes without the grid distortion errors associated with Lagrangian-type schemes.

where $r(y) = |y - 0.5|$ is the radial coordinate, $\theta_0(y, t) = \theta(y) + \omega(r)t$, and $\theta(y) = \arctan\left(\frac{y^2 - 0.5}{y^1 - 0.5}\right)$. The parameters σ_1 and σ_2 control the aspect ratio of the blob, while $\omega(r)$ is the angular velocity. As in [18], we set $\sigma_1 = 0.05$, $\sigma_2 = 0.001$, and

$$\omega(r) = 1.6 \max[(0.5 - r)r, 0].$$

The function (36) describes the evolution of an initially smooth Gaussian blob $\bar{G}(y, 0)$ advected by an incompressible velocity field $V = (V_r, V_\theta) = (0, r\omega(r))$, where V_r and V_θ are the velocity components in the r and θ directions, respectively. The initial blob is smooth but will develop arbitrarily small scales for $t > 0$ due to the radial dependence of the angular velocity. As in [18], we set the grid resolution at $N = 128^2$, the time-step as $\Delta t = 1$, and generate a sequence of meshes for $0 \leq t \leq 90$.

5.4.2. Comparison of static, dynamic, and restarted SAM

The results of static, dynamic, and restarted SAM simulations are provided in Fig. 7, which shows zoomed-in plots of the meshes near $(0.5, 0.5)$ at the final time $t = 90$. All three schemes produce grids that are untangled, but while static SAM grids are smooth, the dynamic SAM grids contain more distorted elements. This is confirmed in Fig. 8, which provides plots of the time history of the L^2 Jacobian error (18) and L^1 distortion (30). As with the rotating patch problem, the distorted dynamic SAM grids are still more accurate than the static SAM grids, though the Jacobian errors are roughly comparable.

For restarted SAM, the restart criterion $\lambda_k > \Lambda \lambda_{\text{ref}}$, with $\Lambda = 1.01$, forces the grid to reset 2 times during the simulation. As shown in Fig. 8 and in the final row of Fig. 7, the restarted SAM grids are smooth and comparable to static SAM grids, and are almost as accurate as the dynamic SAM grids. A comparison of Fig. 8 with Figure 10 in [18] shows that static and restarted SAM grids are of similar quality to the MK grids.

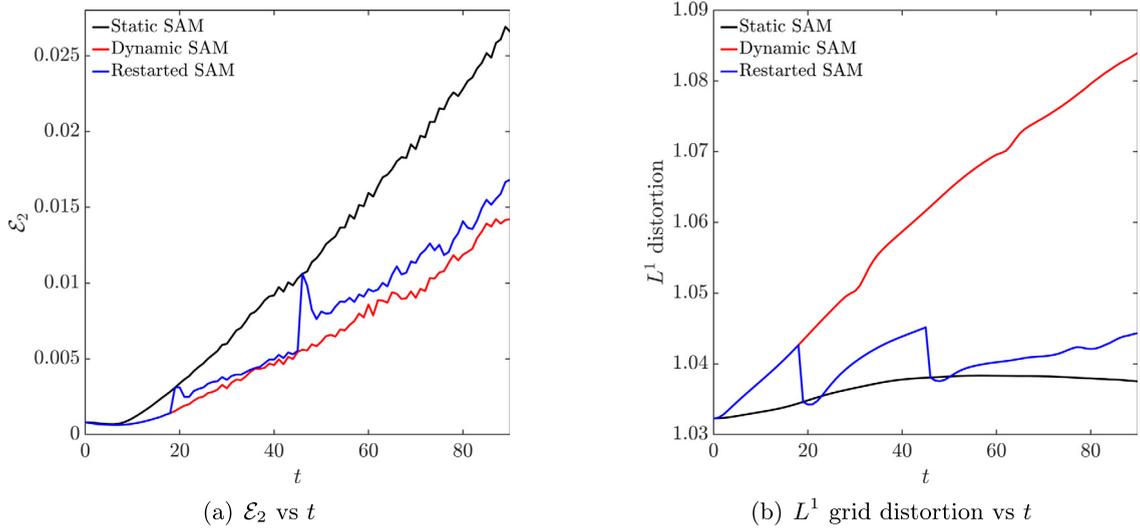


Fig. 8. Test Problem 5.4: tracking small scale vortical structures in flows with differential rotation using (36). Shown are (a) L^2 Jacobian error \mathcal{E}_2 and (b) L^1 grid distortion history of the grids produced using static, dynamic, and restarted SAM. The errors are comparable to those provided in [18], and restarted SAM controls the grid distortion associated with Lagrangian-type schemes.

5.5. 3D swirling flow

5.5.1. Problem description

Our final experiment is a 3D dynamic version of the test in [9]. The domain is $\Omega = [0, 1]^3$, the time interval is $0 \leq t \leq 1$, and the target Jacobian function is given by

$$\bar{G}(y^1, y^2, y^3, t) = \frac{1}{1 + 5e^{-36(y^3 - \frac{1}{2})^2} \exp(-\omega_1 R(y^1, y^2, y^3, t))}, \quad (37)$$

with

$$R(y^1, y^2, y^3, t) = \left(y^1 - \frac{1}{2} - \omega_2 \cos(4\pi(y^3 - t/4)) \right)^2 + \left(y^2 - \frac{1}{2} - \omega_2 \sin(4\pi(y^3 - t/4)) \right)^2,$$

and $\omega_1 = 100$ and $\omega_2 = 0.25$. As discussed in [9], the target function (37) describes a complex 3D helical surface and poses a major challenge for mesh generation algorithms since it leads to highly non-uniform and twisted meshes. See Fig. 9 for plots of the target function and (a portion of) the associated mesh at $t = 1$ and at $N = 128^3$ cell resolution. In Fig. 10, we provide plots of $\psi(P)$, where $P \subset \mathcal{T}_{\text{ref}}$ is some planar subset (lying in either the x^1x^2 -, x^2x^3 -, or x^1x^3 -planes) of the reference mesh.

We generate a sequence of meshes starting with $N = 32^3$ resolution and doubling in each direction until $N = 256^3$. The time-step Δt depends on N according to the CFL scaling and is set as $\Delta t = \frac{2}{\sqrt[3]{N}}$. It is straightforward to adapt the 2D numerical scheme described in Section 3.3 to the 3D setting, and for brevity we omit the details.

5.5.2. $N = 128^3$ simulations using static, dynamic, and restarted SAM

Plots of the time-history of the Jacobian errors \mathcal{E}_2 and L^1 distortion at $N = 128^3$ are shown in Fig. 11. For $0 \leq t \leq 0.5$, dynamic SAM produces the smallest errors, due to the greater accuracy with which the Poisson and transport problems are solved. As expected, dynamic SAM meshes exhibit increasing grid distortion, which causes growth of the Jacobian error. The restart criterion in restarted SAM forces the mesh to reset two times during the simulation, which controls the growth of both the mesh distortion as well as the Jacobian error; for this example, $\Lambda = 1.003$.

5.5.3. Resolution study

Next, we provide in Fig. 12 plots (as a function of the resolution N) of the Jacobian error, L^1 distortion, and CPU runtime at $t = 1$. Fig. 12(a) shows that restarted SAM produces grids with the smallest Jacobian errors, but the errors for the various schemes are comparable for all the resolutions considered; as expected, we observe 4th order convergence for all the schemes. Fig. 12(b) shows that the L^1 distortion for both static and dynamic SAM is consistent across resolutions, with the grid distortion for restarted SAM bounded between the two. Finally, Fig. 12(c) shows that, while static SAM is of complexity $\mathcal{O}(N \cdot N^{1/3}/\Delta t) = \mathcal{O}(N^{5/3})$, both dynamic and restarted SAM are of optimal complexity $\mathcal{O}(N/\Delta t) = \mathcal{O}(N^{4/3})$. Based on this, we can estimate that, for this test, restarted SAM becomes more efficient than static SAM for $N > 735^3$.

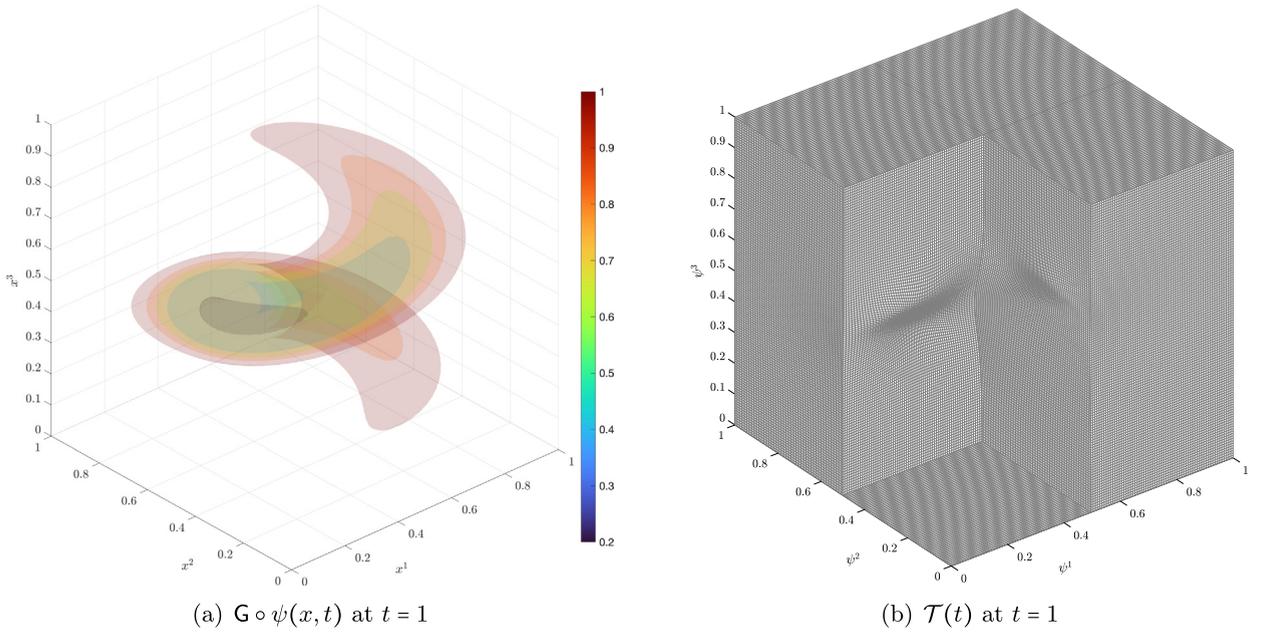


Fig. 9. Test Problem 5.5: 3D swirling flow with the helical target function (37). Fig. 9(a) shows isosurfaces of the target function G and Fig. 9(b) shows a portion of the corresponding mesh.

6. SAM-ALE scheme for gas dynamics

We next couple our SAM scheme to a very simple FD WENO-based ALE scheme. The purpose of this section is to demonstrate the ability of SAM-ALE to reproduce high-resolution uniform runs using fewer cells and less total CPU time. The numerical method for the ALE system of equations we use is highly simplified and not meant to be representative of the full class of ALE solvers. Nonetheless, even for the two very difficult test problems presented in Section 7, the highly simplified scheme performs remarkably well.

For the notation used in this section, we refer the reader to Section 2.

6.1. The 2D ALE-Euler system

6.1.1. Equations in Eulerian coordinates

The 2D compressible Euler system in Eulerian coordinates $y = (y^1, y^2) \in \Omega$ can be written in the following compact conservation-law form

$$\partial_t Q + D_i F^i(Q) = 0, \quad (y, t) \in \Omega \times (0, T), \tag{38a}$$

$$Q(y, 0) = Q_0(y), \quad (y, t) \in \Omega \times \{0\}. \tag{38b}$$

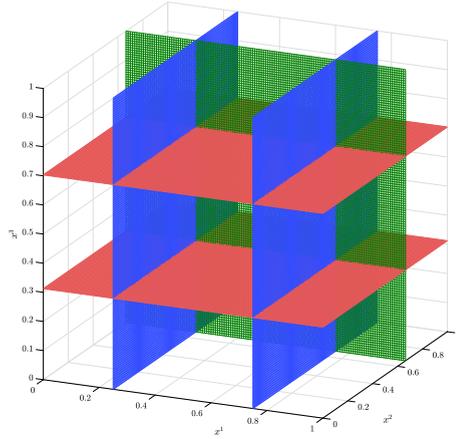
Here, Q is the vector of conserved variables, and $F^1(Q)$ and $F^2(Q)$ are the flux functions, defined as

$$Q = \begin{pmatrix} \rho \\ \rho u^1 \\ \rho u^2 \\ E \end{pmatrix} \quad \text{and} \quad F^i(Q) = \begin{pmatrix} \rho u^i \\ \rho u^1 u^i + \delta_1^i p \\ \rho u^2 u^i + \delta_2^i p \\ u^i (E + p) \end{pmatrix}. \tag{39}$$

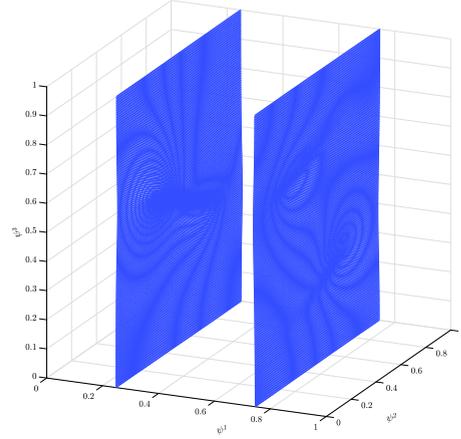
The velocity vector is $u = (u^1, u^2)$ with horizontal component u^1 and vertical component u^2 , $\rho > 0$ is the fluid density (assumed strictly positive), E denotes the energy, and p is the pressure defined by the ideal gas law,

$$p = (\gamma - 1) \left(E - \frac{1}{2} \rho |u|^2 \right), \tag{40}$$

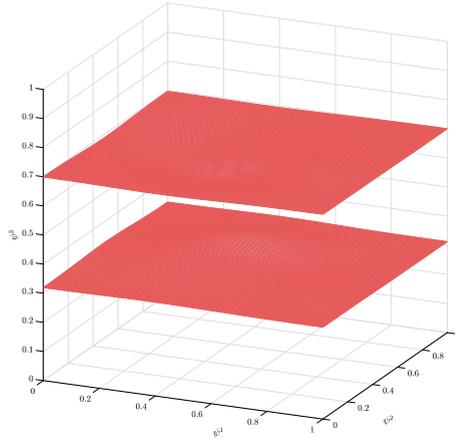
where γ is the adiabatic constant, which we will assume takes the value $\gamma = 1.4$, unless otherwise stated.



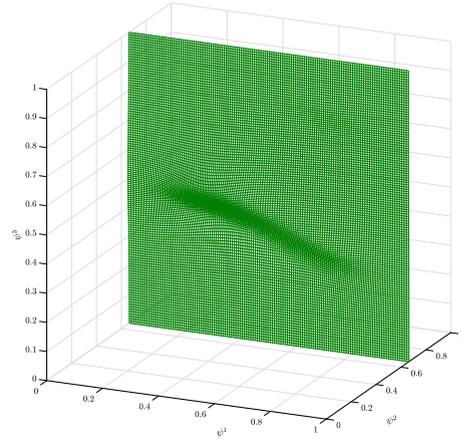
(a) Planes in \mathcal{T}_{ref} showing the pre-image of the meshes in Figures 10(b) to 10(d).



(b) Image of the planes $\{x^1 = 32/128\}$ and $\{x^1 = 96/128\}$ under the map ψ .



(c) Image of the planes $\{x^3 = 40/128\}$ and $\{x^3 = 90/128\}$ under the map ψ .



(d) Image of the planes $\{x^2 = 85/128\}$ under the map ψ .

Fig. 10. Test Problem 5.5: 3D swirling flow with the helical target function (37). Shown are plots of the images of various planes $P \subset \mathcal{T}_{\text{ref}}$ under the map ψ .

6.1.2. Equations in ALE coordinates

Let Ω_{ref} be the fixed reference domain with coordinates (x^1, x^2) , and assume that we have, for each $t \geq 0$, a smooth ALE map $\psi(\cdot, t) : \Omega_{\text{ref}} \rightarrow \Omega$. Denote by the regular font f the ALE counterpart to the Eulerian variable written with upright font f i.e. $f(x, t) = f \circ \psi(x, t)$. The 2D ALE-Euler system can then be written in conservation law form as

$$\partial_t Q + \partial_j F^j(Q) = 0, \quad (x, t) \in \Omega_{\text{ref}} \times (0, T), \tag{41a}$$

$$Q(x, 0) = Q_0(x), \quad (x, t) \in \Omega_{\text{ref}} \times \{0\}, \tag{41b}$$

where the conserved ALE variables Q and flux functions $F^j(Q)$ are given as

$$Q = \begin{pmatrix} \mathcal{J} \rho \\ \mathcal{J} \rho u^1 \\ \mathcal{J} \rho u^2 \\ \mathcal{J} E \end{pmatrix} \quad \text{and} \quad F^j(Q) = \begin{pmatrix} \rho a_i^j (u^i - \psi_t^i) \\ \rho u^1 a_i^j (u^i - \psi_t^i) + a_1^j p \\ \rho u^2 a_i^j (u^i - \psi_t^i) + a_2^j p \\ E a_i^j (u^i - \psi_t^i) + p a_i^j u^i \end{pmatrix}. \tag{42}$$

Here, a_i^j denotes the components of the cofactor matrix defined by (3), and ψ_t^i is the i^{th} component of the mesh velocity. It is also convenient to introduce the ALE transport velocity $v(x, t)$ with j^{th} component $v^j := \frac{1}{\mathcal{J}} a_i^j (u^i - \psi_t^i)$. The 2D ALE-Euler

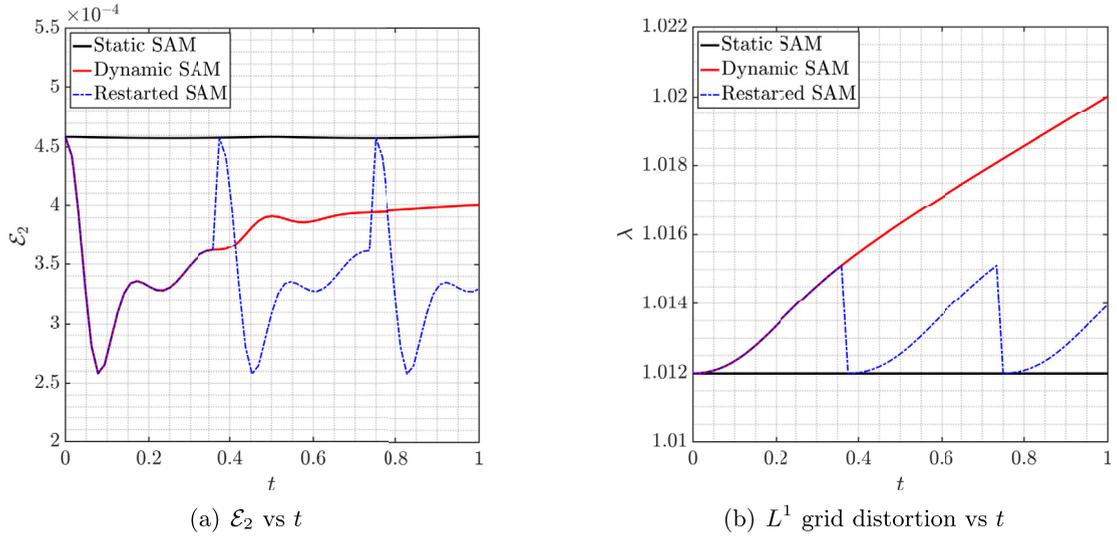


Fig. 11. Test Problem 5.5: 3D swirling flow with the helical target function (37). Shown are (a) L^2 Jacobian error \mathcal{E}_2 and (b) L^1 grid distortion history of the grids produced using static, dynamic, and restarted SAM at $N = 128^3$. Restarted SAM controls the grid distortion associated with Lagrangian-type schemes.

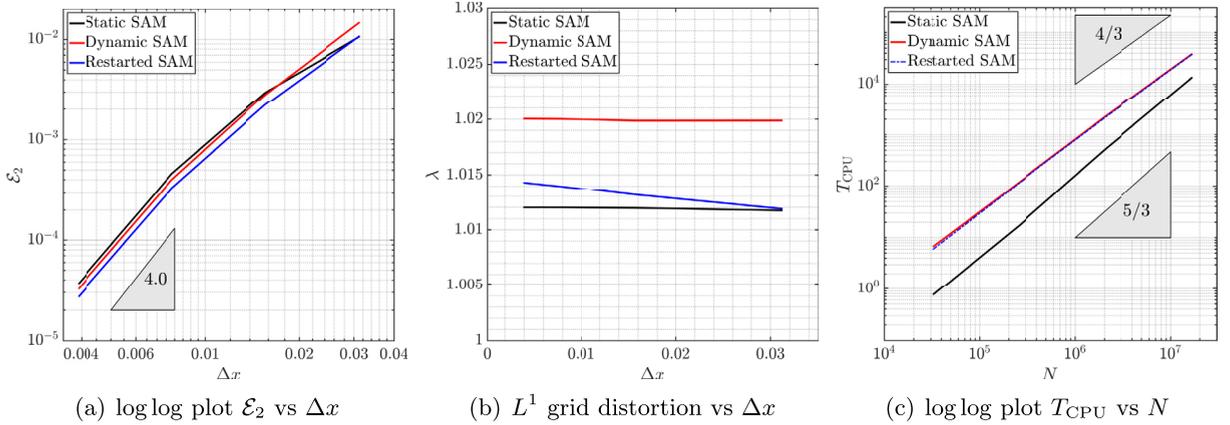


Fig. 12. Test Problem 5.5: 3D swirling flow with the helical target function (37). Shown are (a) loglog plots of L^2 Jacobian error \mathcal{E}_2 vs Δx , (b) L^1 grid distortion vs Δx , and (c) loglog plots of total runtime T_{CPU} vs N of the grids produced using static, dynamic, and restarted SAM for $N = 32^3, \dots, 256^3$.

system (41) is hyperbolic in the sense that each of $\nabla_Q F^j(Q)$ is diagonalizable with real eigenvalues (or wave speeds), which are given explicitly by

$$\lambda^{j,\pm} = \frac{1}{\mathcal{J}}(v^j \pm c) \quad \text{and} \quad \lambda^{j,0} = \frac{1}{\mathcal{J}}v^j \text{ (repeated)}, \tag{43}$$

with $c = \sqrt{\gamma p/\rho}$ the sound speed.

6.1.3. Geometric conservation law and free-stream preservation

An explicit computation shows that the Jacobian determinant $\mathcal{J}(x, t)$ satisfies the *geometric conservation law* (GCL) [75]

$$\partial_t \mathcal{J} - \partial_j (a_j^i \psi_t^i) = 0. \tag{44}$$

For (41), an equivalent property to the GCL is the *free-stream preservation property*, which states that an initially uniform flow (i.e. $Q_0 \equiv \text{constant}$) is preserved under evolution by (41a) i.e. $Q \equiv \text{constant}$ for every $t > 0$. Numerical schemes that fail to preserve the free-stream produce unacceptably large errors that corrupt small-scale vortical structures [38,79,17,60,45].

Finite difference schemes on static uniform meshes preserve the free-stream. On dynamic adaptive meshes, however, this is no longer a given, and indeed many standard schemes (including WENO [43]) fail to preserve the free-stream. As such, we design our numerical scheme to ensure free-stream preservation by explicitly incorporating (44) into the system of conservation laws to be solved [38,84]. Specifically, we append to (41) the equation (44) and consider the modified system

$$\partial_t \tilde{Q} + \partial_j \tilde{F}^j(\tilde{Q}) = 0, \quad (x, t) \in \Omega_{\text{ref}} \times (0, T), \tag{45a}$$

$$\tilde{Q}(x, 0) = \tilde{Q}_0(x), \quad (x, t) \in \Omega_{\text{ref}} \times \{0\}, \tag{45b}$$

with

$$\tilde{Q} = \begin{pmatrix} \mathcal{J}\rho \\ \mathcal{J}\rho u^1 \\ \mathcal{J}\rho u^2 \\ \mathcal{J}E \\ \mathcal{J} \end{pmatrix} \quad \text{and} \quad \tilde{F}^j(\tilde{Q}) = \begin{pmatrix} \mathcal{J}\rho v^j \\ \mathcal{J}\rho u^1 v^j + a_1^j p \\ \mathcal{J}\rho u^2 v^j + a_2^j p \\ \mathcal{J}E v^j + p a_i^j u^i \\ -a_i^j \psi_t^i \end{pmatrix}. \tag{46}$$

We emphasize that, while the cofactor matrix a_i^j is computed directly from the map ψ according to (3), the Jacobian determinant \mathcal{J} is computed (using the same numerical method used for the other equations in (45)) via (44) and not by the usual determinant formula $\mathcal{J} = \partial_1 \psi^1 \partial_2 \psi^2 - \partial_1 \psi^2 \partial_2 \psi^1$ except at the initial time $t = 0$.

6.2. The C-method for 2D ALE-Euler

Next, we describe some aspects of our numerical framework for solving (45). Specifically, we adapt the C-method, introduced in the Eulerian setting in [64,65], to the ALE setting. One of the key features of the C-method is space-time smooth tracking of shock/contact fronts and their geometries via so-called C-functions. The C-functions are space-time smoothed versions of localized solution gradients, and are found as the solutions to auxiliary scalar reaction-diffusion equations. These C-functions in turn allow us to implement both directionally isotropic (for shock stabilization) and anisotropic (for contact stabilization) artificial viscosity schemes. In particular, the C-method is a PDE-level modification of (45). Consequently, the methods developed in [64,65] can be implemented in the ALE context in a straightforward manner. For the purposes of brevity, we omit some of the details here and refer the reader to [65] and Appendix A.

6.2.1. WENO-type reconstruction and computation of a_i^j

We discretize the uniform mesh and index the nodes by $x_{r,s} = (x_r^1, x_s^2)$. At each $x_{r,s}$ we construct numerical flux functions $\hat{F}_{r+\frac{1}{2},s}^1$ and $\hat{F}_{r,s+\frac{1}{2}}^2$ that will be used to approximate the derivatives $\partial_1 \tilde{F}^1(\tilde{Q})|_{x_{r,s}}$ and $\partial_2 \tilde{F}^2(\tilde{Q})|_{x_{r,s}}$, respectively. We describe the procedure for $\hat{F}_{r+\frac{1}{2},s}^1$. For ease of notation, we drop the superscript 1 and let $\tilde{F}^1 \equiv \tilde{F}$. Decompose $\tilde{F} = \tilde{F}^v + \tilde{F}^p + \tilde{F}^E + \tilde{F}^{\mathcal{J}}$ with

$$\tilde{F}^v = \begin{pmatrix} \mathcal{J}\rho v^j \\ \mathcal{J}\rho u^1 v^j \\ \mathcal{J}\rho u^2 v^j \\ \mathcal{J}E v^j \\ 0 \end{pmatrix}, \quad \tilde{F}^p = \begin{pmatrix} 0 \\ a_1^j p \\ a_2^j p \\ 0 \\ 0 \end{pmatrix}, \quad \tilde{F}^E = \begin{pmatrix} 0 \\ 0 \\ 0 \\ p a_i^j u^i \\ 0 \end{pmatrix}, \quad \tilde{F}^{\mathcal{J}} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -a_i^j \psi_t^i \end{pmatrix}. \tag{47}$$

Each component of the advection term \tilde{F}^v is approximated at the half-point $x_{r+\frac{1}{2},s}$ as

$$\hat{F}_{r+\frac{1}{2},s}^v = \text{WENO}(q, \mathcal{J}v^j) := q_{r+\frac{1}{2},s}(\mathcal{J}v^j)_{r+\frac{1}{2},s}, \tag{48}$$

where q denotes one of the variables $q \in \{\rho, \rho u^1, \rho u^2, E\}$ and $q_{r+\frac{1}{2},s}$ is computed using a standard 5th order WENO reconstruction [68] of q with upwinding based on the sign of $(\mathcal{J}v^j)_{r+\frac{1}{2},s}$. The velocity $(\mathcal{J}v^j)_{r+\frac{1}{2},s}$ is computed according to the 4th order average

$$(w)_{r+\frac{1}{2},s} := \frac{-w_{r-1,s} + 7w_{r,s} + 7w_{r+1,s} - w_{r+2,s}}{12}. \tag{49}$$

The additional advection terms $\hat{F}_{r+\frac{1}{2},s}^E = \text{WENO}(p, a_i^j u^i)$ and $\hat{F}_{r+\frac{1}{2},s}^{\mathcal{J}} = \text{WENO}(1, -a_i^j \psi_t^i)$ can be approximated in a similar fashion to (48). The pressure term $\hat{F}_{r+\frac{1}{2},s}^p$ is approximated by the 4th order average (49). Finally, the total flux is given by the sum $\hat{F}_{r+\frac{1}{2},s} = \hat{F}_{r+\frac{1}{2},s}^v + \hat{F}_{r+\frac{1}{2},s}^p + \hat{F}_{r+\frac{1}{2},s}^E + \hat{F}_{r+\frac{1}{2},s}^{\mathcal{J}}$. The semi-discrete scheme for (45) then reads

$$\partial_t \tilde{Q}_{r,s} + \frac{\hat{F}_{r+\frac{1}{2},s}^1 - \hat{F}_{r-\frac{1}{2},s}^1}{\Delta x^1} + \frac{\hat{F}_{r,s+\frac{1}{2}}^2 - \hat{F}_{r,s-\frac{1}{2}}^2}{\Delta x^2} = 0. \tag{50}$$

For free-stream flows, we have that $q_{r,s} \equiv \text{constant}$ and the scheme becomes linear, due to the linear averaging (49). In particular, it is easy to verify that the free-stream is preserved, provided the components of the cofactor matrix a_i^j are computed by 4th order central differencing of the map ψ as

$$\left[\partial_1 \psi^j \right]_{r,s} = \frac{\psi_{r-2,s}^j - 8\psi_{r-1,s}^j + 8\psi_{r+1,s}^j - \psi_{r+2,s}^j}{12\Delta x^1}, \tag{51}$$

and similarly for $\left[\partial_2 \psi^j \right]_{r,s}$.

To confirm this, we perform a free-stream test on the 50×50 time-dependent moving-mesh defined by

$$\begin{cases} \psi^1(x^1, x^2, t) = x^1 + 0.4 \sin\left(\frac{3\pi t}{T}\right) \sin\left(\frac{3\pi}{8}(x^2 + 8)\right) & \text{(a)} \\ \psi^2(x^1, x^2, t) = x^2 + 0.8 \sin\left(\frac{3\pi t}{T}\right) \sin\left(\frac{3\pi}{8}(x^1 + 8)\right) & \text{(b)} \end{cases} \tag{52}$$

for $(x^1, x^2) \in [-8, +8]^2$ and $0 \leq t \leq T = 80$. The initial data is uniform $U_0 \equiv 1$ and we employ periodic boundary conditions. The magnitude of the density error at the final time $t = T$ is $\|\rho(\cdot, T) - 1\|_{L^\infty} = 9.10 \times 10^{-14}$ i.e. the scheme maintains free stream flows to machine precision.

For non-smooth problems with shocks or contacts, it is necessary to add an artificial viscosity term to the right-hand side of (45a), and the semi-discrete scheme (50) must be modified appropriately. The details of the particular form of artificial viscosity we use are provided in Appendix A.

Remark 1. The simplified WENO-type reconstruction procedure outlined above is similar in some respects to the WENO schemes based on the so-called *alternative flux formulation*, first introduced in [69] and explored extensively in several recent papers [44,45,61,19,55]. In particular, both schemes define the flux $\hat{F}_{r+\frac{1}{2},s}$ by first reconstructing the variables $q_{r+\frac{1}{2},s}$. On the other hand, the alternative flux formulation WENO schemes utilize characteristic decompositions and (exact or approximate) Riemann solvers. The resulting algorithms are more expensive but also more robust. Nonetheless, for simple problems, both the simplified WENO and alternative flux WENO schemes produce similar results [64,65]. For more challenging problems, the simplified WENO scheme produces oscillatory solutions; these oscillations can be suppressed with C-method artificial viscosity.

6.2.2. *Explicit interface tracking*

The C-method utilizes a simple method for tracking of contact discontinuities which we first describe in the Eulerian setting i.e. for the system (38). Let $z : \mathcal{I} \times [0, T] \rightarrow \Omega$ be a parametrization of the material interface with parameter $\alpha \in \mathcal{I} \subset \mathbb{R}$, and with components $z = (z^1, z^2)$. In many simulations, the contact discontinuity is a closed or periodic curve, and in this case we take $\mathcal{I} = [-\pi, \pi]$. Given an initial parametrization z_0 of the contact discontinuity, the interface $z(\alpha, t)$ is found as the solution to

$$\begin{cases} \partial_t z(\alpha, t) = \bar{u} \circ z(\alpha, t), & \alpha \in \mathcal{I} \text{ and } 0 < t \leq T & \text{(a)} \\ z(\alpha, 0) = z_0(\alpha), & \alpha \in \mathcal{I} \text{ and } t = 0 & \text{(b)} \end{cases} \tag{53}$$

Here, the velocity \bar{u} is defined as the average $\bar{u} = \frac{1}{2}(u^+ + u^-)$, with u^\pm denoting the fluid velocity on either side of the interface. In a numerical implementation, the average \bar{u} is approximated by bilinear interpolation of u onto z .

The ALE analog of the (Lagrangian) interface tracking algorithm described above can be derived by defining the ALE interface parametrization $z : \mathcal{I} \times [0, T]$ as the image of z under the action of the inverse ALE map $\psi^{-1} : \Omega \times [0, T] \rightarrow \Omega_{\text{ref}}$ i.e.

$$z(\alpha, t) = \psi^{-1} \circ z(\alpha, t).$$

If the map ψ resolves mesh points around z , then the ALE interface z represents a “zoomed-in” version of z that magnifies small scale structures cf. Fig. 17(d).

A chain rule computation shows that z is the solution to

$$\begin{cases} \partial_t z(\alpha, t) = \bar{v} \circ z(\alpha, t), & \alpha \in \mathcal{I} \text{ and } 0 < t \leq T & \text{(a)} \\ z(\alpha, 0) = z_0(\alpha), & \alpha \in \mathcal{I} \text{ and } t = 0 & \text{(b)} \end{cases} \tag{54}$$

where $\bar{v} = \frac{1}{2}(v^+ + v^-)$. The initial interface z_0 is defined by

$$z_0(\alpha) = \psi^{-1} \circ z(\alpha, 0). \tag{55}$$

In a numerical implementation, the initial ALE interface z can be computed as the roots of $\psi_0(z_0) = z_0$ using e.g. Newton’s method.

Algorithm 4 COUPLED SAM-ALE.

- Step 0: Initialization** $t = 0$.
- (a) Define the initial Eulerian data Q_0 on the uniform mesh $\mathcal{U} \subset \Omega$ and the initial interface parametrization $z_0(\alpha)$.
 - (b) Define the initial target Jacobian function G_0 on \mathcal{U} . Compute the initial ALE map $\psi_0 : \Omega_{\text{ref}} \rightarrow \Omega$ and adaptive mesh $\mathcal{T}_0 = \psi_0(\mathcal{T}_{\text{ref}}) \subset \Omega$ using static SAM Algorithm 1.
 - (c) Define the initial ALE data Q_0 . Compute the initial ALE interface z_0 using Newton's method.
- Step 1: Time-stepping** $t = t_k \geq 0$. Assume that we are given all quantities at $t = t_k$.
- (a) Define the target Jacobian function G_{k+1} and compute the map ψ_{k+1} and adaptive mesh \mathcal{T}_{k+1} according to restarted dynamic SAM Algorithm 3.
 - (b) Compute the cofactor matrix a_i^j using (51). Define the mesh velocity $\partial_t \psi_{k+1} = \frac{\psi_{k+1} - \psi_k}{\Delta t}$.
 - (c) Compute the ALE variables \tilde{Q}_{k+1} and z_{k+1} using the C-method and RK4 time-stepping. The mesh, cofactor matrix, and mesh velocity are kept fixed over the time step.
 - (d) Compute the interface $z_{k+1} = \psi_{k+1} \circ z_{k+1}$.
 - (e) If $t_{k+1} = T$, then stop; else, set $t = t_{k+1}$ and return to **Step 1(a)**.

6.3. Coupled SAM-ALE algorithm

Our SAM algorithm is coupled to the ALE C-method by defining an appropriate target Jacobian function G_k . In this work, for simplicity, we shall assume that G_k is explicitly defined, either by some particular formula (as in the Noh test), or via the interface z_k (for the RT test). Future work will investigate coupling of SAM-ALE by means of balanced monitoring of solution gradients [77]. In the case of RT instability, it is important to use the interface z to control adaptation since it allows high mesh concentration in KH roll up zones, in contrast to the balanced monitoring approach in which the magnitudes of solution gradients decrease in KH zones due to mixing [73].

The complete SAM-ALE algorithm is provided in Algorithm 4.

7. SAM-ALE simulations of gas dynamics

7.1. Noh implosion

The first test is the 2D Noh implosion: an initially cold gas is directed towards the origin with speed 1 and instantaneously implodes at the origin, resulting in a radially symmetric infinite strength shock propagating outwards with speed 1/3. This is an extremely difficult test problem and almost all codes report errors in the form of wall heating, lack of symmetry, incorrect shock speeds, or even failure to run [53]. This is the case for both Lagrangian-type codes with artificial viscosity [52,7,20], as well as AMR codes such as RAGE [30]. Extensive numerical testing in [76] showed that catastrophic anomalies occur in AMR solutions, with the anomalies persisting, or even worsening as the grid is refined. These anomalies occur due to spurious wave reflections on discontinuous grids [78,29].

7.1.1. Problem description

The domain as $\Omega = [0, 1]^2$, the adiabatic constant is $\gamma = 5/3$, and the initial data is

$$\begin{bmatrix} \rho_0 \\ (\rho u^1)_0 \\ (\rho u^2)_0 \\ E_0 \end{bmatrix} = \begin{bmatrix} 1 \\ -\cos(\theta) \\ -\sin(\theta) \\ 0.5 + 10^{-6}/(\gamma - 1) \end{bmatrix} \chi_{r>0} + \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0.5 + 10^{-6}/(\gamma - 1) \end{bmatrix} \chi_{r=0}, \tag{56}$$

where $r = |y|$ is the radial coordinate, $\theta \in [0, \frac{\pi}{2})$ is the polar angle, and χ_A is the indicator function on the set A . We employ reflecting boundary conditions on the left and bottom boundaries and use the exact solution to impose the boundary conditions at the top and right boundaries. The problem is run until the final time $T = 2$.

7.1.2. Uniform mesh simulations

We apply the C-method as described in [65] on 50×50 , 100×100 , and 200×200 meshes with time step Δt set so that $CFL \approx 0.2$. The C-method artificial viscosity coefficients in (65) are fixed as $\beta_u = 0.35$, $\beta_E = 2.5$, and $\mu = 0$. The scatter plots of density vs r in Fig. 13 show that the C-method produces stable non-oscillatory solutions that maintain radial symmetry. Moreover, the smooth artificial viscosity almost entirely removes the wall-heating error in the higher resolution runs.

7.1.3. SAM-ALE simulations

Next, we apply SAM-ALE on a 50×50 dynamic adaptive mesh. For simplicity, we choose a specially designed forcing function G for the mesh generation, defined as

$$C_\psi(y^1, y^2, t) = \exp \left[-400 \left(r^2 - t^2/9 \right) \right],$$

$$\tilde{G}(y, t) = \frac{1}{1 + \frac{\kappa}{1-\kappa} \frac{C_\psi(y, t)}{\int_\Omega C_\psi(y, t) dy}}. \tag{57}$$

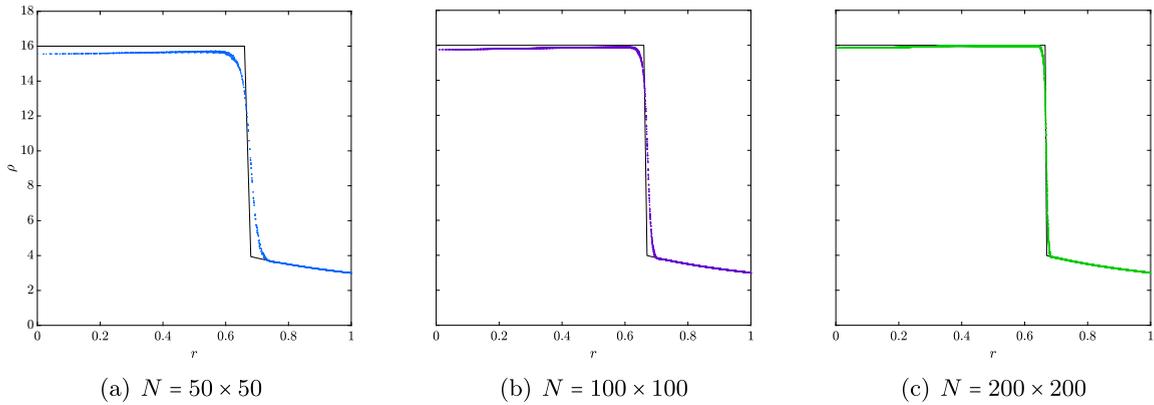


Fig. 13. Uniform mesh runs for the 2D Noh problem. Shown are the density scatter plots vs radial coordinate r . The black curve in each subfigure is the exact solution. The shock fronts are sharp and the solutions free of the spurious asymmetry, wall-heating, oscillation, and shock-racing errors associated with the majority of numerical methods for this test.

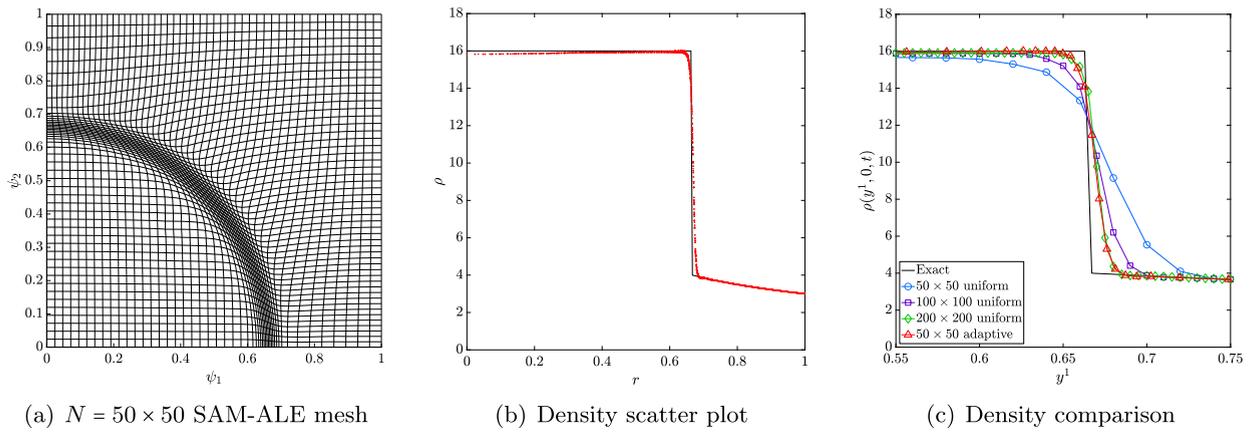


Fig. 14. SAM-ALE simulations of the Noh implosion. Shown are (a) adaptive mesh \mathcal{T} . (b) density scatter plot, and (c) comparison of uniform vs SAM-ALE density zoom-in at the shock. The smooth concentration and alignment of the mesh in the vicinity of the shock front allows for a sharp shock representation in the SAM-ALE solution, comparable to the high-resolution 200×200 uniform mesh solution.

This forcing function is designed, using the known analytical solution, to track the moving shock. In the future, a shock-tracking scheme analogous to the z-type advection (53) for contract tracking will be employed to define \bar{G} . The z-type advection can track the shock with high accuracy, and the resulting \bar{G} is almost exactly the same as (57). As such, for simplicity we use the specially designed function (57) in this work, with the understanding that similar results can be obtained when z-type shock tracking is used instead. The particular normalization used to define \bar{G} is motivated by the balanced monitoring method [77]. We set $\kappa = 0.3$ and the time-step as $\Delta t = 5 \times 10^{-4}$, which yields $CFL \approx 0.2$, and choose artificial viscosity parameters $\beta_u = 0.1$, $\beta_E = 0.7$, and $\mu = 0$.

The results are shown in Fig. 14. The shock front is sharp, the wall-heating error is very small, and solution symmetry is well preserved. The latter is a consequence of both C-method artificial viscosity as well as grid alignment with the shock front. The density cross sections $\rho(y^1, 0, t)$ along the y^1 -axis for the various simulations are shown in Fig. 14(c), which clearly shows that the 50×50 adaptive simulation outperforms the low-res and mid-res uniform simulations, and is comparable to the high-res uniform simulation. The wall heating error is smallest for the adaptive simulation, and the sharpness of the shock fronts for the 200×200 uniform and 50×50 adaptive simulations is comparable. As shown in Table 6, the adaptive mesh simulation produces the solution with the smallest L^2 error in the density. Moreover, the adaptive simulation is approximately 6 times faster than the high-res uniform simulation, and requires roughly the same amount of memory as the lowest-resolution uniform run.

7.2. Rayleigh-Taylor instability

Our second test problem is the classical RT instability. This test poses a huge challenge for Lagrangian and ALE methods due to the complex geometry of the evolving unstable interface. As such, limited RT ALE simulations are available in the literature (but see [85,57,24,34] for some examples). In fact, the RT problem is so challenging for ALE codes that very often

Table 6
 Comparison of simulation statistics for the uniform and adaptive mesh C-method simulations for the Noh problem. The low-res SAM-ALE simulation is more accurate than the high-res uniform simulation, while running 6 times faster and requiring only 18% as much memory.

Simulation	Simulation statistic		
	L^2 density error	CPU time (secs)	Memory usage (MBs)
50 × 50 uniform	1.019×10^0	4.3	7.5
100 × 100 uniform	6.917×10^{-1}	35.3	14.6
200 × 200 uniform	5.406×10^{-1}	289	43.7
50 × 50 adaptive	4.897×10^{-1}	45.6	7.8

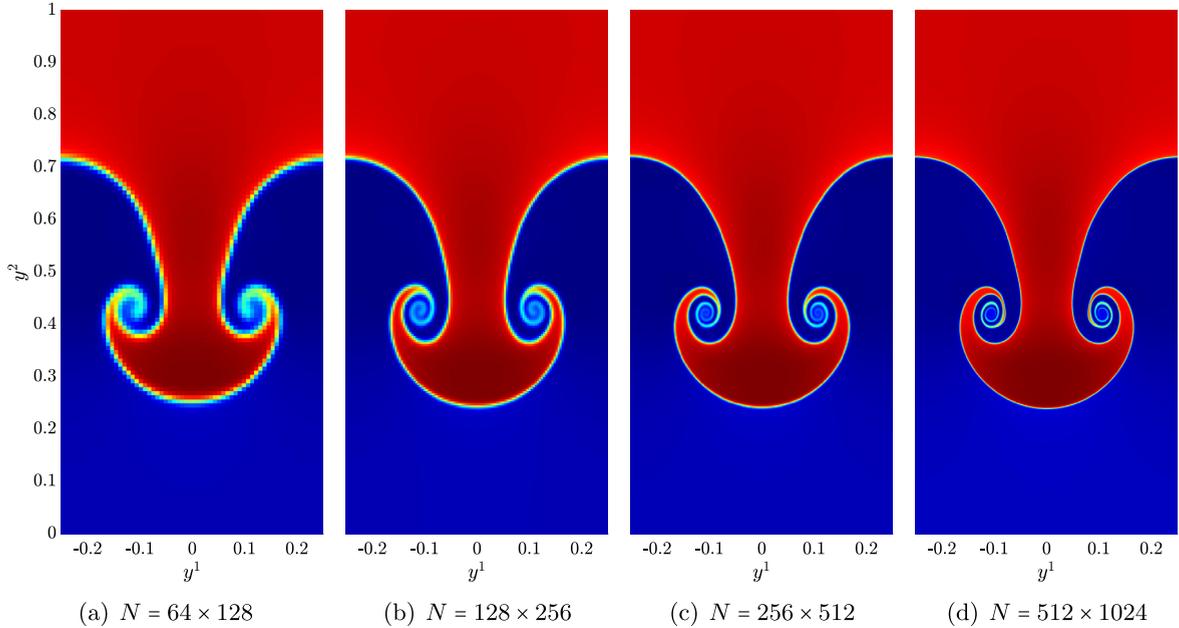


Fig. 15. Uniform mesh simulations of RT instability with sharper fronts and more small scale structure in the KH zone as the resolution increases.

the goal is simply to perform a simulation that runs until the final time without excessive mesh tangling, at which point the simulation breaks down [57,4].

7.2.1. Problem description

We add the source term $\tilde{S}(x, t) = (0, 0, -\mathcal{J}\rho g, -\mathcal{J}\rho g u^2, 0, 0)^T$ to the right-hand side of (45a). The domain is $\Omega = [-0.25, 0.25] \times [0, 1]$ and we apply periodic and free-flow conditions in the y^1 and y^2 directions [63]. The initial data is $u_0 = 0$, and

$$\rho_0 = \begin{cases} 5 - \rho^- g y^2 & , \text{ if } y^2 < 0.5 \\ 5 - 0.5\rho^- g - \rho^+ g(y^2 - 0.5) & , \text{ if } y^2 \geq 0.5 \end{cases} \tag{58a}$$

$$\rho_0(y^1, y^2) = \rho^- + \frac{\rho^+ - \rho^-}{2} \left[1 + \tanh\left(\frac{y^2 - \eta_0(y^1)}{h}\right) \right] \tag{58b}$$

where $\rho^+ = 2$ and $\rho^- = 1$, $\eta_0(y^1) = 0.5 - 0.01 \cos(4\pi y^1)$, $h = 0.005$, and $g = 1$. The problem is run until the final time $T = 2.5$.

7.2.2. Uniform mesh simulations

We compute a sequence of uniform mesh simulations for resolutions $N = 64 \times 128$ through $N = 512 \times 1024$ with $CFL \approx 0.45$. The artificial viscosity parameters are set as $\mu = 7.5 \times 10^{-4}$ and $\beta_u = \beta_E = 0$, and we show heatmap plots of the density in Fig. 15. As the resolution is increased, more small-scale structure can be seen in the main KH roll up region. The artificial viscosity term suppresses further secondary instabilities that usually occur with other dimensionally split numerical methods [53,1].

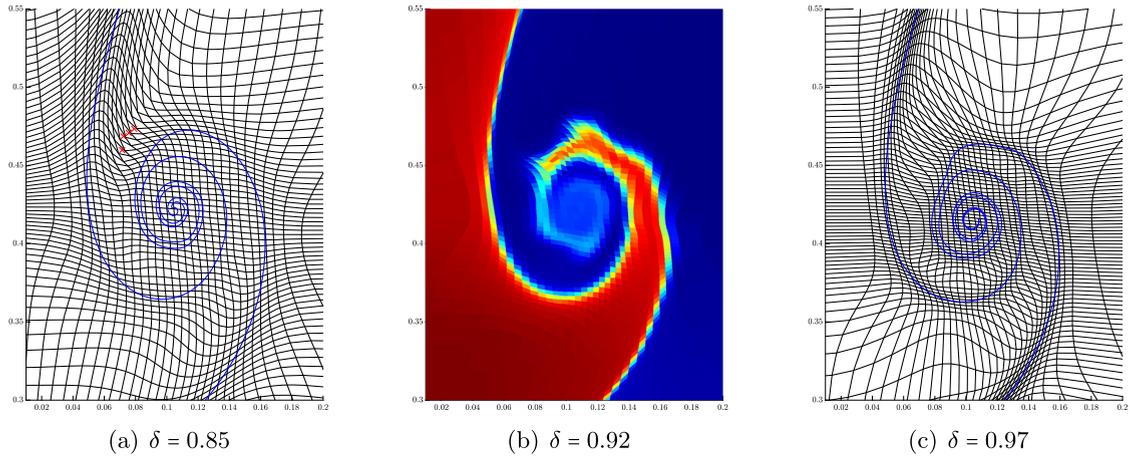


Fig. 16. 64×128 adaptive mesh simulations of RT with large zoom-in factor. Figure (a) is a zoom-in of the mesh computed with restarted SAM and $\delta = 0.85$. The interface z is shown as the blue curve, and the non-convex elements are indicated by red crosses. Figure (b) is a zoom-in of the density with $\delta = 0.92$. The non-convex elements cause spurious instabilities along the interface. Figure (c) shows the mesh computed with the large zoom-in algorithm; all the elements are convex and the mesh is smooth.

Table 7

Total CPU runtime for uniform and adaptive simulations of RT instability.

Runtime (sec)	Cells				
	64×128	128×256	256×512	512×1024	64×128 SAM-ALE
T_{CPU}	2.21×10^1	1.67×10^2	1.37×10^3	1.21×10^4	1.38×10^2

7.2.3. Mesh generation with large zoom-in factor

Next, we aim to produce a 64×128 adaptive mesh with large zoom-in factor that resolves around the material interface z and define a target Jacobian function as

$$G_\delta(y, t) = 1 - \delta \exp\left(-\left|\sigma \min_\alpha |y - z(\alpha, t)|\right|^2\right), \quad (59)$$

with $\sigma = 25$. For this resolution, the meshes produced with dynamic SAM contain non-convex elements for δ larger than approximately 0.85, as shown in Fig. 16(a). These non-convex elements arise due to a strong cusp-type flow in the region between the “stem” of the mushroom and the roll up region. The choice $\delta = 0.85$ produces a mesh with smallest cell size only approximately 3.8 times smaller than a uniform mesh cell. Increasing the value of δ further produces a mesh with more non-convex elements, which in turn causes spurious errors in the computed numerical solution as shown in Fig. 16(b).

A simple technique to resolve this issue is to use the large zoom-in algorithm described in Section 5.1. Specifically, we use the large zoom-in algorithm (with 25 sub time steps) in combination with restarted dynamic SAM. The 64×128 adaptive mesh with $\delta = 0.97$ is shown in Fig. 16(c), from which it can be seen that the mesh is smooth and all elements are convex. The smallest cell size in the mesh is approximately 13 times smaller than a uniform cell. The large zoom-in algorithm is applied only when the mesh resets, and the increase in CPU runtime is therefore negligible.

7.2.4. Comparison of adaptive and uniform simulations

We perform a 64×128 cell SAM-ALE simulation with zoom-in parameter $\delta = 0.97$ and $\Delta t = 1.5625 \times 10^{-4}$. Plots of the adaptive mesh and density heatmap are provided in Fig. 17(a) and Fig. 17(b), and we refer to Fig. 16(c) for the mesh zoom-in. A comparison with the uniform mesh simulations in Fig. 15 shows that the 64×128 SAM-ALE simulation has a much sharper interface and exhibits more small-scale roll-up than the 64×128 uniform simulation, and is roughly comparable to the $N = 256 \times 512$ simulation. However, some of the small-scale structure is not observed in the SAM-ALE density. Interestingly, this roll up is captured by the interface z , shown in Fig. 17(c). This suggests that a more robust ALE solver (e.g. WENO with alternative flux formulation) may produce improved results.¹⁰ The ALE interface z is shown in Fig. 17(d) and is clearly a zoomed-in version of z , with the small scale KH zones magnified and represented over a much larger region.

The CPU runtimes of the various simulations are provided in Table 7, from which we see that the SAM-ALE simulation is approximately 10 times and 88 times faster than the 256×512 and 512×1024 uniform runs, respectively. For this problem, the CPU time spent on mesh generation is roughly the same as the time spent on ALE calculations. Since SAM is

¹⁰ See also [77] for a comparison of Lax-Friedrichs vs low dissipation HLLC flux reconstruction in the FV framework.

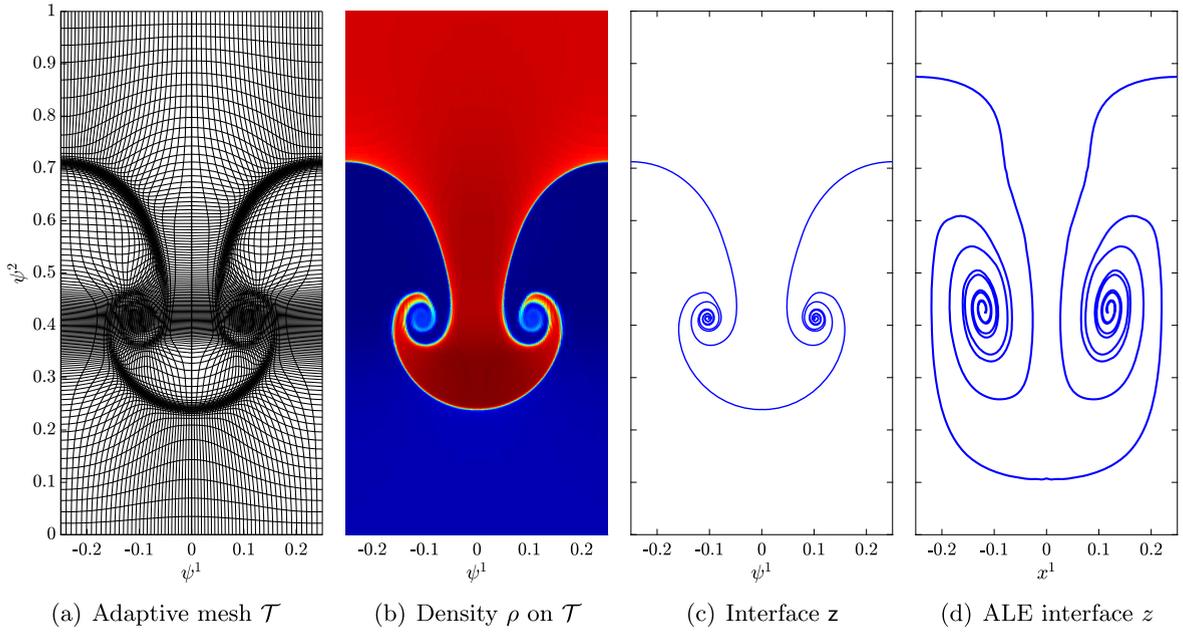


Fig. 17. 64×128 SAM-ALE simulation of RT instability with $\delta = 0.97$.

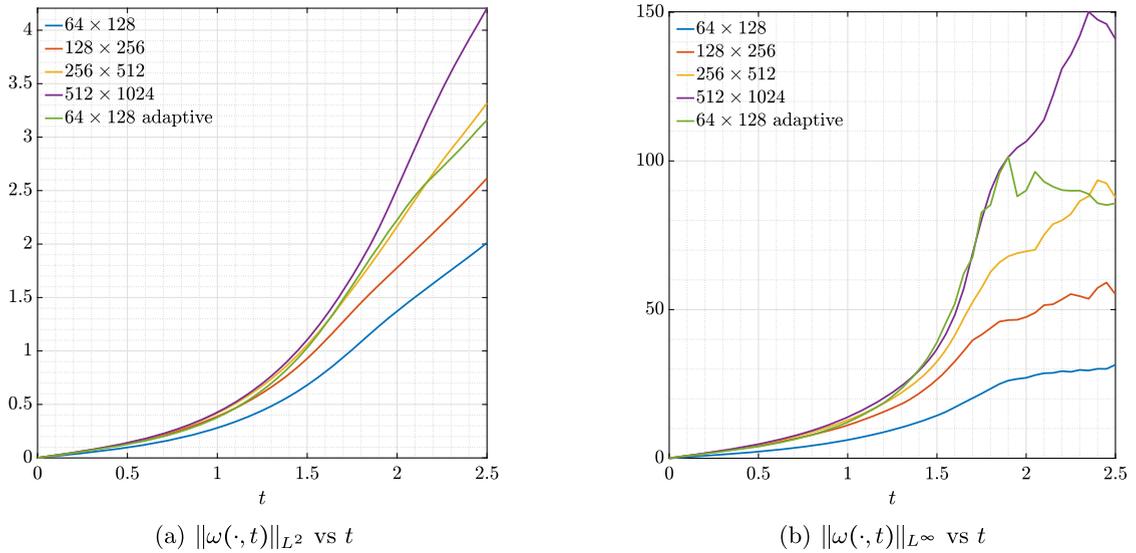


Fig. 18. Time history of the L^2 and L^∞ norms of the vorticity for uniform and adaptive mesh simulations of RT instability.

roughly 100-200 times faster than MK mesh generation, it is clear that an MK-ALE scheme cannot provide a speed-up over uniform mesh simulations. On the other hand, the use of a more robust ALE solver can only improve the relative efficiency of SAM-ALE, since the main computational expense will be the ALE calculations rather than mesh generation.

The time histories of the L^2 and L^∞ norms of the vorticity ω for the uniform and adaptive mesh simulations are shown in Fig. 18. These figures confirm that the 64×128 SAM-ALE run is comparable to the 256×512 uniform run. In fact, for $t \leq 1.75$, when the mesh zoom-in factor is approximately 20 times, the 64×128 SAM-ALE run closely approximates the 512×1024 uniform run. For $t > 1.75$, the mesh zoom-in factor decreases due to the stretching of the interface and the adaptive mesh is no longer able to capture the smallest scales that are present in the 512×1024 run. The decrease in the mesh zoom-in factor is a consequence of the fact that the number of cells in the mesh are fixed. So-called h - r adaptive mesh methods [23] are a way to overcome this issue; the simplicity of our algorithmic framework suggests that a dynamic h - r method based on SAM can be readily formulated and implemented, and this will be investigated in future work.

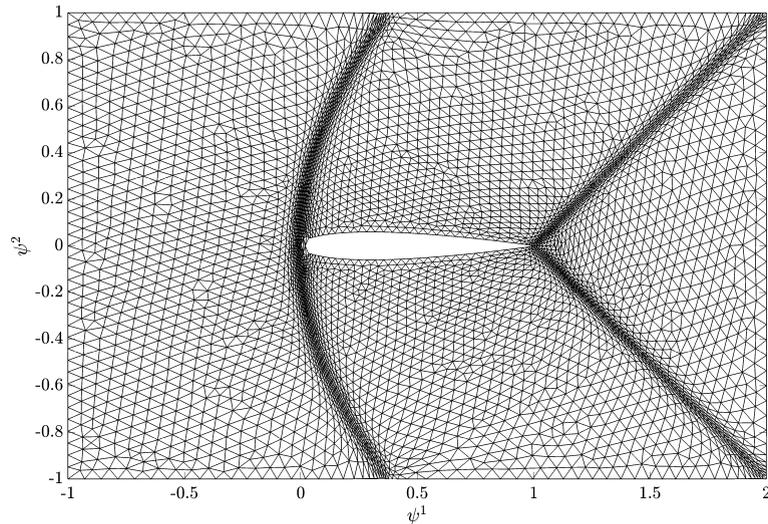


Fig. 19. Unstructured mesh modeling compressible flow past an airfoil. The mesh is constructed using (a preliminary version of) unstructured SAM in the Firedrake framework [66].

8. Concluding remarks

In this work, we developed a new Smooth Adaptive Meshing (SAM) algorithm based on a new perturbation formulation and implementation of the deformation method. The resulting numerical algorithm is simple, stable, automated, high-order accurate, and able to generate smooth and untangled meshes resolving around complex multi- D flows. We coupled SAM to a simple ALE scheme for gas dynamics and presented adaptive-simulation speed-up results for the challenging Noh and Rayleigh-Taylor problems.

Several aspects of our SAM formulation and algorithm require further investigation and improvement. As discussed in Section 7.2, we are interested in developing an h - r -refinement scheme based on SAM and, more generally, a dynamic SAM algorithm on general unstructured meshes. The numerical implementation of unstructured SAM is obviously more delicate than the simple uniform-mesh scheme presented in the current paper, and will be thoroughly investigated in future work. Nonetheless, we provide in Fig. 19 a preliminary result showing an unstructured SAM mesh that models compressible flow past an airfoil. This mesh was produced¹¹ within the finite-element based Firedrake code [66]. In the future, we will investigate the theoretical properties of SAM solutions on general domains, and their connections to the regularity of the discrete mesh \mathcal{T} .

CRedit authorship contribution statement

Raaghav Ramani: Conceptualization, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Steve Shkoller:** Conceptualization, Methodology, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Raaghav Ramani reports equipment, drugs, or supplies was provided by University of New Mexico Center for Advanced Research Computing.

Data availability

No data was used for the research described in the article.

Acknowledgements

Research reported in this publication was supported by NSF grant DMS-2007606 and DTRA grant HDTRA11810022. This research was also supported by Office of Defense Nuclear Nonproliferation, NA-22 and NA-24; we note that the views of the authors do not necessarily reflect the views of the USG. This work was also supported by the Laboratory Directed Research

¹¹ We express our gratitude to Dr. Mariana Clare for her assistance with writing the code and generating the result shown in Fig. 19.

and Development Program of the Los Alamos National Laboratory, which is under the auspices of the National Nuclear Security Administration of the U.S. Department of Energy under DOE Contracts W-7405-ENG-36 and LA-UR-10-04291.

We would like to thank the UNM Center for Advanced Research Computing, supported in part by the National Science Foundation, for providing high performance computing resources used in this work.

We would like to express our gratitude to the anonymous referees for their numerous suggestions that have greatly improved the manuscript.

Appendix A. The C-method for 2D ALE-Euler

We provide a brief review of the C-method for adding space-time smooth artificial viscosity to shocks and contacts [65]. The most important feature of the C-method is smooth tracking of shock/contact fronts and their geometries via so-called C-functions. The C-functions are space-time smoothed versions of localized solution gradients, and are found as the solutions to auxiliary scalar reaction-diffusion equations. Specifically, we use C to denote a smoothed shock tracking function, and τ to denote the vector-valued function $\tau = (\tau^1, \tau^2)$. The function $\tilde{\tau}$ is a smoothed version of the tangent vector to an evolving contact discontinuity. These C-functions allow us to implement both directionally isotropic (for shock stabilization) and anisotropic (for contact stabilization) artificial viscosity schemes.

To summarize the method, it is convenient to introduce advection, artificial viscosity, and C-equation operators as follows.

A.0.1. ALE advection operator

For a scalar function $Q : \Omega_{\text{ref}} \rightarrow \mathbb{R}$, and a vector-valued function $v : \Omega_{\text{ref}} \rightarrow \mathbb{R}^2$, define

$$\mathcal{A}[Q; v] := \partial_k (Q a_i^k v^i). \tag{60}$$

A.0.2. ALE isotropic artificial viscosity operator

For a scalar function $Q : \Omega_{\text{ref}} \rightarrow \mathbb{R}$, define

$$\mathcal{D}[Q; \beta] := \partial_k (\tilde{\beta} \rho C a_i^k a_i^l \partial_l Q), \tag{61}$$

with

$$\tilde{\beta} = \frac{|\Delta x|^2}{\max C} \beta.$$

The constant β is an isotropic artificial viscosity parameter for shock stabilization.

A.0.3. ALE anisotropic artificial viscosity operator

For a scalar function $Q : \Omega_{\text{ref}} \rightarrow \mathbb{R}$, we define

$$\mathcal{D}^\tau [Q; \mu] := \partial_k [\tilde{\mu} \rho \tau^i \tau^j a_i^k a_j^l \partial_l Q], \tag{62}$$

with

$$\tilde{\mu} = \frac{|\Delta x|^2}{\alpha^2} \mu. \tag{63}$$

Here, μ is the anisotropic artificial viscosity parameter for contact discontinuity stabilization and $\alpha = \max_x \{|\tau^1|, |\tau^2|\}$.

A.0.4. ALE C-equation operator

For a scalar function $H : \Omega_{\text{ref}} \rightarrow \mathbb{R}$ and scalar forcing function $Q : \Omega_{\text{ref}} \rightarrow \mathbb{R}$, let

$$\mathcal{L}[H; Q] := \frac{S}{\varepsilon |\Delta x|} (Q - H) + \kappa S |\Delta x| \Delta H. \tag{64}$$

A.0.5. The complete ALE-Euler-C system

Now, we can write the full ALE-Euler-C system as

$$\partial_t (\mathcal{J} \rho) + \mathcal{A}[\rho; u - \psi_t] = 0, \tag{65a}$$

$$\partial_t (\mathcal{J} \rho u^r) + \mathcal{A}[\rho u^r; u - \psi_t] = \mathcal{D}^\tau [u^r; \mu] + \mathcal{D}[u^r; \beta_u] - \partial_j (a_r^j p), \quad \text{for } r = 1, 2, \tag{65b}$$

$$\partial_t (\mathcal{J} E) + \mathcal{A}[E; u - \psi_t] + \mathcal{A}[p; u] = \mathcal{D}[E/\rho; \beta_E], \tag{65c}$$

$$\partial_t \mathcal{J} - \mathcal{A}[1; \psi_t] = 0, \tag{65d}$$

$$\partial_t C - \mathcal{L}[C; F] = 0, \tag{65e}$$

$$\partial_t \tau^r - \mathcal{L}[\tau^r; F^r] = 0, \quad \text{for } r = 1, 2. \tag{65f}$$

The forcing functions for (65e) and (65f) are defined as follows. The shock C forcing function is given by

$$\hat{F} = \frac{|\frac{1}{\mathcal{J}}a_i^j \partial_j \rho|}{\max |\frac{1}{\mathcal{J}}a_i^j \partial_j \rho|}, \tag{66}$$

while the components of the forcing to the contact tangent vector τ equations are defined by

$$F^1 = -\frac{1}{\mathcal{J}}a_2^j \partial_j \rho \quad \text{and} \quad F^2 = \frac{1}{\mathcal{J}}a_1^j \partial_j \rho. \tag{67}$$

The initial conditions for C and τ are defined by solving the time-independent versions of (65e) and (65f).

Appendix B. Boundary smoothing for non-Neumann functions

Herein, we describe a simple boundary smoothing technique for non-Neumann functions. Let $x_{\text{mid}}^r = \frac{1}{2}(x_{\text{min}}^r + x_{\text{max}}^r)$, for $r = 1, 2$. Define smooth cutoff functions

$$\begin{aligned} \phi^1(\xi) &= \frac{1}{2} \left[\tanh\left(\frac{\xi - (x_{\text{min}}^1 + d_1)}{\varepsilon}\right) - \tanh\left(\frac{\xi - (x_{\text{max}}^1 - d_1)}{\varepsilon}\right) \right], \\ \phi^2(\eta) &= \frac{1}{2} \left[\tanh\left(\frac{\eta - (x_{\text{min}}^2 + d_2)}{\varepsilon}\right) - \tanh\left(\frac{\eta - (x_{\text{max}}^2 - d_2)}{\varepsilon}\right) \right], \end{aligned}$$

where ε is a smoothing parameter, which we choose as $\varepsilon = 0.02$. The function ϕ^1 is equal to 1 in the interior of the domain, then smoothly decreases to 0 at a distance d_1 near the left and right boundaries. The function ϕ^2 behaves similarly. We set $d_r = 0.05(x_{\text{max}}^r - x_{\text{min}}^r)$.

Given a non-Neumann function G, we first compute the derivatives D_1G , D_2G , and $D_{12}G$. We then compute

$$\begin{aligned} \mathcal{I}^{(1)}(y^1) &= \int_{x_{\text{mid}}^1}^{y^1} \phi^1(\xi) D_1G(\xi, x_{\text{mid}}^2) d\xi, \\ \mathcal{I}^{(2)}(y^2) &= \int_{x_{\text{mid}}^2}^{y^2} \phi^2(\eta) D_2G(x_{\text{mid}}^1, \eta) d\eta, \\ \mathcal{I}^{(3)}(y^1, y^2) &= \int_{x_{\text{mid}}^2}^{y^2} \int_{x_{\text{mid}}^1}^{y^1} \phi^1(\xi) D_{12}G(\xi, \eta) d\xi d\eta, \end{aligned}$$

and define

$$G^*(y^1, y^2) := G(x_{\text{mid}}^1, x_{\text{mid}}^2) + \mathcal{I}^{(1)}(y^1) + \mathcal{I}^{(2)}(y^2) + \mathcal{I}^{(3)}(y^1, y^2).$$

The function G^* then satisfies $DG^* \cdot \nu = 0$ on $\partial\Omega$.

Appendix C. The MK scheme

The MK scheme solves for the unique [8,16] diffeomorphism ψ satisfying (7) that minimizes the L^2 displacement $\|\psi(x) - x\|_{L^2}$. The MK formulation is developed by writing $\psi = x + \nabla\Psi$, where Ψ is a scalar potential. The equation governing Ψ is found by minimizing a functional consisting of the L^2 displacement and a local Lagrange multiplier, where the latter is used to enforce the Jacobian constraint (7). The resulting equation for Ψ is fully nonlinear, and the MK scheme uses an iterative Newton-Krylov solver with multigrid preconditioning to find an approximation to the solution Ψ , within some error tolerance ϵ .

C.1. Machine comparison

To reliably compare the runtimes of our static SAM Algorithm 1 with the MK scheme as listed in [22], we need to account for the different machines on which these codes were run. Therefore, we perform the following machine comparison experiment. In [22], the authors also report the CPU runtimes for a deformation method of Liao and Anderson [50], whose description is provided in the Appendix of [22]. We coded a numerical implementation of this method, which we refer to

Table 8

CPU runtimes for the LA scheme on the machine from [22] and the LA scheme on our machine. The data for the LA scheme in the top row is taken from Table 3 of [22].

Scheme		Cells				
		16 × 16	32 × 32	64 × 64	128 × 128	256 × 256
LA on [22] machine	T_{CPU}	0.2	0.9	3.4	13.6	55.0
LA on our machine	T_{CPU}	0.12	0.41	1.53	6.22	24.16
	speed-up factor	1.7	2.2	2.2	2.2	2.3

as LA, and ran the numerical experiments from [22] on our machine. The runtimes for LA on our machine, along with the LA runtimes from Table 3 of [22], are shown in Table 8. These data show that our machine runs approximately 2.2 times faster than the machine on which the MK simulations in [22] were performed.

References

- [1] A.S. Almgren, V.E. Beckner, J.B. Bell, M.S. Day, L.H. Howell, C.C. Joggerst, M.J. Lijewski, A. Nonaka, M. Singer, M. Zingale, CASTRO: A new compressible astrophysical solver. I. Hydrodynamics and self-gravity, *Astrophys. J.* 715 (2) (may 2010) 1221–1238, <https://doi.org/10.1088/0004-637x/715/2/1221>.
- [2] A. Averbuch, M. Israeli, L. Vozovoi, A fast Poisson solver of arbitrary order accuracy in rectangular regions, *SIAM J. Sci. Comput.* 19 (3) (1998) 933–952, <https://doi.org/10.1137/S1064827595288589>.
- [3] B.N. Azarenok, S.A. Ivanenko, T. Tang, Adaptive mesh redistribution method based on Godunov's scheme, *Commun. Math. Sci.* 1 (1) (2003) 152–179.
- [4] A.J. Barlow, P.-H. Maire, W.J. Rider, R.N. Rieben, M.J. Shashkov, Arbitrary Lagrangian Eulerian methods for modeling high-speed compressible multimaterial flows, *J. Comput. Phys.* (ISSN 0021-9991) 322 (2016) 603–665, <https://doi.org/10.1016/j.jcp.2016.07.001>, <https://www.sciencedirect.com/science/article/pii/S0021999116302807>.
- [5] M. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, *J. Comput. Phys.* (ISSN 0021-9991) 82 (1) (1989) 64–84, [https://doi.org/10.1016/0021-9991\(89\)90035-1](https://doi.org/10.1016/0021-9991(89)90035-1), <https://www.sciencedirect.com/science/article/pii/0021999189900351>.
- [6] J. Brackbill, J. Saltzman, Adaptive zoning for singular problems in two dimensions, *J. Comput. Phys.* (ISSN 0021-9991) 46 (3) (1982) 342–368, [https://doi.org/10.1016/0021-9991\(82\)90020-1](https://doi.org/10.1016/0021-9991(82)90020-1), <https://www.sciencedirect.com/science/article/pii/0021999182900201>.
- [7] J. Breil, Numerical Methods for Lagrangian and Arbitrary-Lagrangian-Eulerian Hydrodynamic Contribution to the simulation of High-Energy-Density-Physics Problems, Habilitation à diriger des recherches, Université de Bordeaux, June 2016, <https://hal.archives-ouvertes.fr/tel-01467157>.
- [8] Y. Brenier, Polar factorization and monotone rearrangement of vector-valued functions, *Commun. Pure Appl. Math.* (ISSN 0010-3640) 44 (4) (1991) 375–417, <https://doi.org/10.1002/cpa.3160440402>.
- [9] P. Browne, C. Budd, C. Piccolo, M. Cullen, Fast three dimensional r-adaptive mesh redistribution, *J. Comput. Phys.* (ISSN 0021-9991) 275 (2014) 174–196, <https://doi.org/10.1016/j.jcp.2014.06.009>, <https://www.sciencedirect.com/science/article/pii/S0021999114004161>.
- [10] G.L. Bryan, M.L. Norman, B.W. O'hea, T. Abel, J.H. Wise, M.J. Turk, D.R. Reynolds, D.C. Collins, P. Wang, S.W. Skillman, B. Smith, R.P. Harkness, J. Bordner, J. hoon Kim, M. Kuhlen, H. Xu, N. Goldbaum, C. Hummels, A.G. Kritsuk, E. Tasker, S. Skory, C.M. Simpson, O. Hahn, J.S. Oishi, G.C. So, F. Zhao, R. Cen, Y. L. ENZO: an adaptive mesh refinement code for astrophysics, *Astrophys. J. Suppl. Ser.* 211 (2) (mar 2014) 19, <https://doi.org/10.1088/0067-0049/211/2/19>.
- [11] C. Budd, B. Leimkuhler, M. Piggott, Scaling invariance and adaptivity, *Appl. Numer. Math.* (ISSN 0168-9274) 39 (3) (2001) 261–288, [https://doi.org/10.1016/S0168-9274\(00\)00036-2](https://doi.org/10.1016/S0168-9274(00)00036-2), <https://www.sciencedirect.com/science/article/pii/S0168927400000362>.
- [12] C. Budd, R. Russell, E. Walsh, The geometry of r-adaptive meshes generated using optimal transport methods, *J. Comput. Phys.* (ISSN 0021-9991) 282 (2015) 113–137, <https://doi.org/10.1016/j.jcp.2014.11.007>, <https://www.sciencedirect.com/science/article/pii/S0021999114007591>.
- [13] C.J. Budd, W. Huang, R.D. Russell, Moving mesh methods for problems with blow-up, *SIAM J. Sci. Comput.* 17 (2) (1996) 305–327, <https://doi.org/10.1137/S1064827594272025>.
- [14] C.J. Budd, W. Huang, R.D. Russell, Adaptivity with moving grids, *Acta Numer.* 18 (2009) 111–241, <https://doi.org/10.1017/S0962492906400015>.
- [15] C.J. Budd, A.T. McRae, C.J. Cotter, The scaling and skewness of optimally transported meshes on the sphere, *J. Comput. Phys.* (ISSN 0021-9991) 375 (2018) 540–564, <https://doi.org/10.1016/j.jcp.2018.08.028>, <https://www.sciencedirect.com/science/article/pii/S0021999118305515>.
- [16] L.A. Caffarelli, Interior $W^{2,p}$ estimates for solutions of the Monge-Ampère equation, *Ann. Math.* (2) (ISSN 0003-486X) 131 (1) (1990) 135–150, <https://doi.org/10.2307/1971510>.
- [17] X. Cai, F. Ladeinde, Performance of weno scheme in generalized curvilinear coordinate systems, in: 46th AIAA Aerospace Sciences Meeting and Exhibit, 2008, p. 36.
- [18] L. Chacón, G. Delzanno, J. Finn, Robust, multidimensional mesh-motion based on Monge-Kantorovich equidistribution, *J. Comput. Phys.* (ISSN 0021-9991) 230 (1) (2011) 87–103, <https://doi.org/10.1016/j.jcp.2010.09.013>, <https://www.sciencedirect.com/science/article/pii/S0021999110005073>.
- [19] A.J. Christlieb, X. Peng, Y. Jiang, Q. Tang, A high-order finite difference WENO scheme for ideal magnetohydrodynamics on curvilinear meshes, *SIAM J. Sci. Comput.* 40 (4) (2018) A2631–A2666, <https://doi.org/10.1137/17M115757X>.
- [20] A.W. Cook, M.S. Ulitsky, D.S. Miller, Hyperviscosity for unstructured ale meshes, *Int. J. Comput. Fluid Dyn.* 27 (1) (2013) 32–50, <https://doi.org/10.1080/10618562.2012.756477>.
- [21] B. Dacorogna, J. Moser, On a partial differential equation involving the jacobian determinant, *Ann. Inst. Henri Poincaré, Anal. Non Linéaire* 7 (1) (1990) 1–26, <http://eudml.org/doc/78211>.
- [22] G. Delzanno, L. Chacón, J. Finn, Y. Chung, G. Lapenta, An optimal robust equidistribution method for two-dimensional grid adaptation based on Monge-Kantorovich optimization, *J. Comput. Phys.* (ISSN 0021-9991) 227 (23) (2008) 9841–9864, <https://doi.org/10.1016/j.jcp.2008.07.020>, <https://www.sciencedirect.com/science/article/pii/S0021999108004105>.
- [23] V. Dobrev, P. Knupp, T. Kolev, K. Mittal, V. Tomov, hr-adaptivity for nonconforming high-order meshes with the target matrix optimization paradigm, *Eng. Comput.* (2021) 1–17, <https://doi.org/10.1007/s00366-021-01407-6>.
- [24] V.A. Dobrev, T.V. Kolev, R.N. Rieben, High-order curvilinear finite element methods for lagrangian hydrodynamics, *SIAM J. Sci. Comput.* 34 (5) (2012) B606–B641, <https://doi.org/10.1137/120864672>.
- [25] J. Duan, H. Tang, Entropy stable adaptive moving mesh schemes for 2d and 3d special relativistic hydrodynamics, *J. Comput. Phys.* (ISSN 0021-9991) 426 (2021) 109949, <https://doi.org/10.1016/j.jcp.2020.109949>, <https://www.sciencedirect.com/science/article/pii/S0021999120307233>.
- [26] J. Duan, H. Tang, High-order accurate entropy stable adaptive moving mesh finite difference schemes for special relativistic (magneto)hydrodynamics, *J. Comput. Phys.* (ISSN 0021-9991) 456 (2022) 111038, <https://doi.org/10.1016/j.jcp.2022.111038>, <https://www.sciencedirect.com/science/article/pii/S0021999122001000>.
- [27] A.S. Dvinsky, Adaptive grid generation from harmonic maps on Riemannian manifolds, *J. Comput. Phys.* (ISSN 0021-9991) 95 (2) (1991) 450–476, [https://doi.org/10.1016/0021-9991\(91\)90285-S](https://doi.org/10.1016/0021-9991(91)90285-S), <https://www.sciencedirect.com/science/article/pii/0021999191902855>.

- [28] H. Feng, S. Zhao, FFT-based high order central difference schemes for three-dimensional Poisson's equation with various types of boundary conditions, *J. Comput. Phys.* 410 (June 2020) 109391, <https://doi.org/10.1016/j.jcp.2020.109391>.
- [29] B. Fryxell, K. Olson, P. Ricker, F.X. Timmes, M. Zingale, D.Q. Lamb, P. MacNeice, R. Rosner, J.W. Truran, H. Tufo, FLASH, An adaptive mesh hydrodynamics code for modeling astrophysical thermonuclear flashes, *Astrophys. J. Suppl. Ser.* 131 (1) (nov 2000) 273–334, <https://doi.org/10.1086/317361>.
- [30] M. Gittings, R. Weaver, M. Clover, T. Betlach, N. Byrne, R. Coker, E. Dendy, R. Hueckstaedt, K. New, W.R. Oakes, D. Ranta, R. Stefan, The RAGE radiation-hydrodynamic code, *Comput. Sci. Discov.* 1 (1) (nov 2008) 015005, <https://doi.org/10.1088/1749-4699/1/1/015005>.
- [31] M. Grajewski, M. Köster, S. Turek, Mathematical and numerical analysis of a robust and efficient grid deformation method in the finite element context, *SIAM J. Sci. Comput.* 31 (2) (2009) 1539–1557, <https://doi.org/10.1137/050639387>.
- [32] M. Grajewski, M. Köster, S. Turek, Numerical analysis and implementational aspects of a new multilevel grid deformation method, *Appl. Numer. Math.* (ISSN 0168-9274) 60 (8) (2010) 767–781, <https://doi.org/10.1016/j.apnum.2010.03.017>, <https://www.sciencedirect.com/science/article/pii/S0168927410000474>.
- [33] P. Grisvard, *Elliptic Problems in Nonsmooth Domains*, Society for Industrial and Applied Mathematics, 2011, <https://epubs.siam.org/doi/abs/10.1137/1.9781611972030>.
- [34] J.-L. Guermond, B. Popov, L. Saavedra, Second-order invariant domain preserving ALE approximation of hyperbolic systems, *J. Comput. Phys.* (ISSN 0021-9991) 401 (2020) 108927, <https://doi.org/10.1016/j.jcp.2019.108927>, <https://www.sciencedirect.com/science/article/pii/S0021999119306321>.
- [35] P. He, H. Tang, An adaptive moving mesh method for two-dimensional relativistic hydrodynamics, *Commun. Comput. Phys.* 11 (1) (2012) 114–146.
- [36] P. He, H. Tang, An adaptive moving mesh method for two-dimensional relativistic magnetohydrodynamics, *Comput. Fluids* 60 (2012) 1–20.
- [37] T. Hell, A. Ostermann, Compatibility conditions for Dirichlet and Neumann problems of Poisson's equation on a rectangle, *J. Math. Anal. Appl.* 420 (2014) 1005–1023, <https://doi.org/10.1016/j.jmaa.2014.06.034>.
- [38] R.G. Hindman, Generalized coordinate forms of governing fluid equations and associated geometrically induced errors, *AIAA J.* 20 (10) (1982) 1359–1367, <https://doi.org/10.2514/3.51196>.
- [39] W. Huang, Metric tensors for anisotropic mesh generation, *J. Comput. Phys.* (ISSN 0021-9991) 204 (2) (2005) 633–665, <https://doi.org/10.1016/j.jcp.2004.10.024>, <https://www.sciencedirect.com/science/article/pii/S0021999104004310>.
- [40] W. Huang, R.D. Russell, Moving mesh strategy based on a gradient flow equation for two-dimensional problems, *SIAM J. Sci. Comput.* 20 (3) (1998) 998–1015, <https://doi.org/10.1137/S1064827596315242>.
- [41] W. Huang, R.D. Russell, *Adaptive Moving Mesh Methods*, vol. 174, Springer Science & Business Media, 2010.
- [42] W. Huang, W. Sun, Variational mesh adaptation II: Error estimates and monitor functions, *J. Comput. Phys.* (ISSN 0021-9991) 184 (2) (2003) 619–648, [https://doi.org/10.1016/S0021-9991\(02\)00040-2](https://doi.org/10.1016/S0021-9991(02)00040-2), <https://www.sciencedirect.com/science/article/pii/S0021999102000402>.
- [43] G.-S. Jiang, C.-W. Shu, Efficient implementation of weighted ENO schemes, *J. Comput. Phys.* (ISSN 0021-9991) 126 (1) (1996) 202–228, <https://doi.org/10.1006/jcph.1996.0130>.
- [44] Y. Jiang, C.-W. Shu, M. Zhang, An alternative formulation of finite difference weighted ENO schemes with Lax–Wendroff time discretization for conservation laws, *SIAM J. Sci. Comput.* 35 (2) (2013) A1137–A1160, <https://doi.org/10.1137/120889885>.
- [45] Y. Jiang, C.-W. Shu, M. Zhang, Free-stream preserving finite difference schemes on curvilinear meshes, *Methods Appl. Anal.* 21 (1) (2014) 1–30, <https://doi.org/10.4310/MAA.2014.v21.n1.a1>.
- [46] P. Knupp, L.G. Margolin, M. Shashkov, Reference Jacobian optimization-based rezoning strategies for arbitrary Lagrangian Eulerian methods, *J. Comput. Phys.* 176 (1) (Feb. 2002) 93–128, <https://doi.org/10.1006/jcph.2001.6969>.
- [47] R. Li, T. Tang, P. Zhang, Moving mesh methods in multiple dimensions based on harmonic maps, *J. Comput. Phys.* (ISSN 0021-9991) 170 (2) (2001) 562–588, <https://doi.org/10.1006/jcph.2001.6749>, <https://www.sciencedirect.com/science/article/pii/S002199910196749X>.
- [48] S. Li, L. Petzold, Moving mesh methods with upwinding schemes for time-dependent pdes, *J. Comput. Phys.* (ISSN 0021-9991) 131 (2) (1997) 368–377, <https://doi.org/10.1006/jcph.1996.5611>, <https://www.sciencedirect.com/science/article/pii/S0021999196956119>.
- [49] S. Li, J. Duan, H. Tang, High-order accurate entropy stable adaptive moving mesh finite difference schemes for (multi-component) compressible Euler equations with the stiffened equation of state, <https://arxiv.org/abs/2202.07989>, 2022.
- [50] G. Liao, D. Anderson, A new approach to grid generation, *Appl. Anal.* 44 (3–4) (1992) 285–298, <https://doi.org/10.1080/00036819208840084>.
- [51] R. Liao, F. Liu, G.C. de la Pena, D. Peng, S. Osher, Level-set-based deformation methods for adaptive grids, *J. Comput. Phys.* (ISSN 0021-9991) 159 (1) (2000) 103–122, <https://doi.org/10.1006/jcph.2000.6432>, <https://www.sciencedirect.com/science/article/pii/S0021999100964325>.
- [52] K. Lipnikov, M. Shashkov, A framework for developing a mimetic tensor artificial viscosity for Lagrangian hydrocodes on arbitrary polygonal meshes, *J. Comput. Phys.* (ISSN 0021-9991) 229 (20) (2010) 7911–7941, <https://doi.org/10.1016/j.jcp.2010.06.045>, <https://www.sciencedirect.com/science/article/pii/S0021999110003694>.
- [53] R. Liska, B. Wendroff, Comparison of several difference schemes on 1D and 2D test problems for the Euler equations, *SIAM J. Sci. Comput.* (ISSN 1064-8275) 25 (3) (2003) 995–1017, <https://doi.org/10.1137/S1064827502402120>.
- [54] F. Liu, S. Ji, G. Liao, An adaptive grid method and its application to steady Euler flow calculations, *SIAM J. Sci. Comput.* 20 (3) (1998) 811–825, <https://doi.org/10.1137/S1064827596305738>.
- [55] Z. Liu, Y. Jiang, M. Zhang, Q. Liu, High order finite difference WENO methods for shallow water equations on curvilinear meshes, *Commun. Appl. Math. Comput.* (2022) 1–44, <https://doi.org/10.1007/s42967-021-00183-w>.
- [56] D. Long, J. Thuburn, Numerical wave propagation on non-uniform one-dimensional staggered grids, *J. Comput. Phys.* (ISSN 0021-9991) 230 (7) (apr 2011) 2643–2659, <https://doi.org/10.1016/j.jcp.2010.12.040>.
- [57] R. Loubère, P.-H. Maire, M. Shashkov, J. Breil, S. Galera, Reale: a reconnection-based arbitrary-Lagrangian-Eulerian method, *J. Comput. Phys.* (ISSN 0021-9991) 229 (12) (2010) 4724–4761, <https://doi.org/10.1016/j.jcp.2010.03.011>, <https://www.sciencedirect.com/science/article/pii/S002199911000121X>.
- [58] D. Luo, W. Huang, J. Qiu, A quasi-lagrangian moving mesh discontinuous Galerkin method for hyperbolic conservation laws, *J. Comput. Phys.* (ISSN 0021-9991) 396 (2019) 544–578, <https://doi.org/10.1016/j.jcp.2019.06.061>, <https://www.sciencedirect.com/science/article/pii/S0021999119304693>.
- [59] A.T.T. McRae, C.J. Cotter, C.J. Budd, Optimal-transport-based mesh adaptivity on the plane and sphere using finite elements, *SIAM J. Sci. Comput.* 40 (2) (2018) A1121–A1148, <https://doi.org/10.1137/16M1109515>.
- [60] T. Nonomura, N. Iizuka, K. Fujii, Freestream and vortex preservation properties of high-order WENO and WCNS on curvilinear grids, *Comput. Fluids* 39 (2) (2010) 197–214, <https://doi.org/10.1016/j.compfluid.2009.08.005>.
- [61] T. Nonomura, D. Terakado, Y. Abe, K. Fujii, A new technique for freestream preservation of finite-difference weno on curvilinear grid, *Comput. Fluids* (ISSN 0045-7930) 107 (2015) 242–255, <https://doi.org/10.1016/j.compfluid.2014.09.025>, <https://www.sciencedirect.com/science/article/pii/S0045793014003624>.
- [62] H.S. Pathak, R.K. Shukla, Adaptive finite-volume weno schemes on dynamically redistributed grids for compressible Euler equations, *J. Comput. Phys.* 319 (2016) 200–230.
- [63] R. Ramani, S. Shkoller, A multiscale model for Rayleigh-Taylor and Richtmyer-Meshkov instabilities, *J. Comput. Phys.* 405 (2020) 109177, <https://doi.org/10.1016/j.jcp.2019.109177>.
- [64] R. Ramani, J. Reisner, S. Shkoller, A space-time smooth artificial viscosity method with wavelet noise indicator and shock collision scheme, Part 1: The 1-D case, *J. Comput. Phys.* (ISSN 0021-9991) 387 (2019) 81–116, <https://doi.org/10.1016/j.jcp.2019.02.049>, <https://www.sciencedirect.com/science/article/pii/S0021999119301664>.

- [65] R. Ramani, J. Reisner, S. Shkoller, A space-time smooth artificial viscosity method with wavelet noise indicator and shock collision scheme, Part 2: The 2-D case, *J. Comput. Phys.* (ISSN 0021-9991) 387 (2019) 45–80, <https://doi.org/10.1016/j.jcp.2019.02.048>.
- [66] F. Rathgeber, D.A. Ham, L. Mitchell, M. Lange, F. Luporini, A.T.T. Mcrae, G.-T. Bercea, G.R. Markall, P.H.J. Kelly, Firedrake: automating the finite element method by composing abstractions, *ACM Trans. Math. Softw.* (ISSN 0098-3500) 43 (3) (dec 2016), <https://doi.org/10.1145/2998441>.
- [67] B. Semper, G. Liao, A moving grid finite-element method using grid deformation, *Numer. Methods Partial Differ. Equ.* 11 (1995) 603–615, <https://doi.org/10.1002/num.1690110606>.
- [68] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws*, in: *LNM*, vol. 1697, Springer, Berlin, Heidelberg, ISBN 978-3-540-49804-9, 1998, pp. 325–432.
- [69] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, *J. Comput. Phys.* 77 (2) (Aug. 1988) 439–471, [https://doi.org/10.1016/0021-9991\(88\)90177-5](https://doi.org/10.1016/0021-9991(88)90177-5).
- [70] J.M. Stone, T.A. Gardiner, P. Teuben, J.F. Hawley, J.B. Simon, Athena: a new code for astrophysical MHD, *Astrophys. J. Suppl. Ser.* 178 (1) (sep 2008) 137–177, <https://doi.org/10.1086/588755>.
- [71] M. Sulman, J. Williams, R. Russell, Optimal mass transport for higher dimensional adaptive grid generation, *J. Comput. Phys.* (ISSN 0021-9991) 230 (9) (2011) 3302–3330, <https://doi.org/10.1016/j.jcp.2011.01.025>, <https://www.sciencedirect.com/science/article/pii/S0021999111000507>.
- [72] M.M. Sulman, J. Williams, R.D. Russell, An efficient approach for the numerical solution of the Monge-Ampère equation, *Appl. Numer. Math.* (ISSN 0168-9274) 61 (3) (2011) 298–307, <https://doi.org/10.1016/j.apnum.2010.10.006>, <https://www.sciencedirect.com/science/article/pii/S0168927410001819>.
- [73] H. Tang, T. Tang, Adaptive mesh methods for one- and two-dimensional hyperbolic conservation laws, *SIAM J. Numer. Anal.* 41 (2) (2003) 487–515, <https://doi.org/10.1137/S003614290138437X>.
- [74] T. Tang, *Moving mesh methods for computational fluid dynamics*, in: *Recent Advances in Adaptive Computation*, in: *Contemp. Math.*, vol. 383, Amer. Math. Soc., Providence, RI, 2005, pp. 141–173.
- [75] P.D. Thomas, C.K. Lombard, Geometric conservation law and its application to flow computations on moving grids, *AIAA J.* 17 (10) (Oct. 1979) 1030–1037, <https://doi.org/10.2514/3.61273>.
- [76] F.X. Timmes, G. Gisler, G.M. Hrbek, *Automated analyses of the tri-lab verification test suite on uniform and adaptive grids for code project a*, 2005.
- [77] A. van Dam, P.A. Zegeling, et al., Balanced monitoring of flow phenomena in moving mesh methods, *Commun. Comput. Phys.* 7 (1) (2010) 138, <https://doi.org/10.4208/cicp.2009.09.033>.
- [78] R. Vichnevetsky, L. Turner, Spurious scattering from discontinuously stretching grids in computational fluid dynamics, *Appl. Numer. Math.* (ISSN 0168-9274) 8 (3) (1991) 289–299, [https://doi.org/10.1016/0168-9274\(91\)90058-8](https://doi.org/10.1016/0168-9274(91)90058-8), <https://www.sciencedirect.com/science/article/pii/0168927491900588>.
- [79] M.R. Visbal, D.V. Gaitonde, On the use of higher-order finite-difference schemes on curvilinear and deforming meshes, *J. Comput. Phys.* 181 (1) (2002) 155–185, <https://doi.org/10.1006/jcph.2002.7117>.
- [80] R. Wang, H. Feng, R.J. Spiteri, Observations on the fifth-order weno method with non-uniform meshes, *Appl. Math. Comput.* (ISSN 0096-3003) 196 (1) (2008) 433–447, <https://doi.org/10.1016/j.amc.2007.06.024>, <https://www.sciencedirect.com/science/article/pii/S0096300307006972>.
- [81] H. Weller, P. Browne, C. Budd, M. Cullen, Mesh adaptation on the sphere using optimal transport and the numerical solution of a Monge-Ampère type equation, *J. Comput. Phys.* (ISSN 0021-9991) 308 (2016) 102–123, <https://doi.org/10.1016/j.jcp.2015.12.018>, <https://www.sciencedirect.com/science/article/pii/S0021999115008372>.
- [82] A.M. Winslow, Numerical solution of the quasilinear Poisson equation in a nonuniform triangle mesh, *J. Comput. Phys.* (ISSN 0021-9991) 1 (2) (1966) 149–172, [https://doi.org/10.1016/0021-9991\(66\)90001-5](https://doi.org/10.1016/0021-9991(66)90001-5), <https://www.sciencedirect.com/science/article/pii/0021999166900015>.
- [83] A.M. Winslow, Adaptive-mesh zoning by the equipotential method, UCID-19062, Lawrence Livermore National Laboratory, 4 1981, <https://www.osti.gov/biblio/6227449>.
- [84] X. Yang, W. Huang, J. Qiu, A moving mesh weno method for one-dimensional conservation laws, *SIAM J. Sci. Comput.* 34 (A2317–A2343) (01 2012), <https://doi.org/10.1137/110856381>.
- [85] P.A. Zegeling, W.D. de Boer, H.Z. Tang, Robust and efficient adaptive moving mesh solution of the 2-D Euler equations, in: *Recent Advances in Adaptive Computation*, in: *Contemp. Math.*, vol. 383, Amer. Math. Soc., Providence, RI, 2005, pp. 375–386.