DISSERTATION

GEOMETRY CONSIDERATIONS FOR HIGH-ORDER FINITE-VOLUME METHODS ON STRUCTURED GRIDS WITH ADAPTIVE MESH REFINEMENT

Submitted by

Nathaniel D. Overton-Katz

Department of Mechanical Engineering

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Summer 2022

Doctoral Committee:

Advisor: Stephen Guzik
Co-Advisor: Xinfeng Gao

Chris Weinberger
Wolfgang Bangerth

ABSTRACT

GEOMETRY CONSIDERATIONS FOR HIGH-ORDER FINITE-VOLUME METHODS ON STRUCTURED GRIDS WITH ADAPTIVE MESH REFINEMENT

Computational fluid dynamics (CFD) is an invaluable tool for engineering design. Meshing complex geometries with accuracy and efficiency is vital to a CFD simulation. In particular, using structured grids with adaptive mesh refinement (AMR) will be invaluable to engineering optimization where automation is critical. For high-order (fourth-order and above) finite volume methods (FVMs), discrete representation of complex geometries adds extra challenges. High-order methods are not trivially extended to complex geometries of engineering interest. To accommodate geometric complexity with structured AMR in the context of high-order FVMs, this work aims to develop three new methods.

First, a robust method is developed for bounding high-order interpolations between grid levels when using AMR. High-order interpolation is prone to numerical oscillations which can result in unphysical solutions. To overcome this, localized interpolation bounds are enforced while maintaining solution conservation. This method provides great flexibility in how refinement may be used in engineering applications.

Second, a mapped multi-block technique is developed, capable of representing moderately complex geometries with structured grids. This method works with high-order FVMs while still enabling AMR and retaining strict solution conservation. This method interfaces with well-established engineering work flows for grid generation and interpolates generalized curvilinear coordinate transformations for each block. Solutions between blocks are then communicated by a generalized interpolation strategy while maintaining a single-valued flux.

Finally, an embedded-boundary technique is developed for high-order FVMs. This method is particularly attractive since it automates mesh generation of any complex geometry. However,

the algorithms on the resulting meshes require extra attention to achieve both stable and accurate results near boundaries. This is achieved by performing solution reconstructions using a weighted form of high-order interpolation that accounts for boundary geometry.

These methods are verified, validated, and tested by complex configurations such as reacting flows in a bluff-body combustor and Stokes flows with complicated geometries. Results demonstrate the new algorithms are effective for solving complex geometries at high-order accuracy with AMR. This study contributes to advance the geometric capability in CFD for efficient and effective engineering applications.

ACKNOWLEDGEMENTS

I would like to thank my advisors Dr. Guzik and Dr. Gao for their advice, support, and the opportunities for growth they have made available to me. I am gratefully for the patience and guidance they have given me as a student, and have become a better researcher and person thanks to the time spent working with them. From my committee I thank Dr. Weinberger for his feedback and suggestions, and Dr. Bangerth for continually pushing me to think critically. I especially want to thank Dr. Johansen and Dr. Antepara at LBNL for their insight and guidance with the embedded-boundary method and Chombo framework. It has been an honor to work with them, and their help has been invaluable. I thank my colleges in the CFD and Propulsion lab for their friendship, discussion, and collaboration. I would like to express my gratitude to my wife for her love, encouragement, and patience. Finally, I would like to express my appreciation to my parents, family, and friends for their support.

TABLE OF CONTENTS

LIST OF NOTATION

The notation is generally explained as it is introduced. Bold type is reserved for vectors dependent on the physical dimension (e.g. 3 dimensional), such as solution variables of a system of the governing equations. Matrices are named with capital upright letters, such as N, and the entries are named with subscripts, $\mathrm{N}_{i,j}$, where i and j are the indices for the row and the column, respectively. Some symbols are listed here for convenience.

D	the spatial dimension
$\mathcal{V}_{\boldsymbol{i}}$	the volume of cell $\boldsymbol{i}$
$\partial\mathcal{V}_{\boldsymbol{i}}$	the surface of cell $\boldsymbol{i}$
$\mathcal{A}_{\boldsymbol{f}}$	the area of face $\boldsymbol{f}$
$\hat{\mathbf{n}}_{\boldsymbol{f}}$	the normal vector to a face $\boldsymbol{f}$
$\mathbf{U}$	solution vector, e.g., $[\mathrm{U}_1, \cdots, \mathrm{U}_N]^\intercal$
$\vec{\mathbf{F}}$	flux dyad, e.g., $[\mathbf{F}_1, \cdots, \mathbf{F}_D]^\intercal$
$\mathbf{F}$	flux vector, e.g., $[\mathrm{F}_1, \cdots, \mathrm{F}_N]^\intercal$
F_j	the j^{th} component of vector $\mathbf{F}$
$\mathbf{S}$	source term
$\mathbb{I}$	the identity matrix
$\boldsymbol{i}$	grid cell indices, e.g., (i, j, k) in 3D
$\boldsymbol{i}$	grid face indices, e.g., (i, j, k) in 3D
$\boldsymbol{e}^d$	unit vector in direction d
$\boldsymbol{\xi}$	computational space, e.g., (ξ, η, ζ) in 3D
ξ_d	the d^{th} component of $\boldsymbol{\xi}$
$\boldsymbol{x}$	physical space, e.g., (x, y, z) in 3D
h	grid spacing
t	time

Δt	time step spacing
ℓ	grid refinement level
n_{ref}	grid refinement factor
$\mathsf{N}^\intercal$	the grid mapping transformation matrix
J	the grid metric Jacobian
Q	polynomial order
$\mathbf{q}$	polynomial multi-index
ρ	density
$\boldsymbol{u}$	velocity
p	pressure
e	internal energy

Operators

$O(-)$	order of accuracy
$\langle - \rangle$	cell-averaged or face-averaged quantity
$-^{-1}$	inverse of an operation or matrix
$-^\intercal$	matrix transpose
$-^\dagger$	matrix pseudoinverse
$- \cdot -$	the matrix inner product
∇_x	the differential operator in physical space
Δ	Laplacian operator
$\mathcal{C}$	grid coarsening operator
$\mathcal{C}^{-1}$	grid refinement operator
$\mathcal{I}(\boldsymbol{i})$	a neighborhood or collection of cells about the cell $\boldsymbol{i}$

Chapter 1

Introduction

1.1 Motivations

Computational fluid dynamics (CFD) has become an invaluable tool for engineering design, allowing for rapid analysis and design optimization [1]. However, the accuracy of CFD modeling of practical fluid dynamics is dependent on many factors, such as the physics models employed in the governing equations, space and time discretization schemes, computational configurations of initial and boundary conditions, and the representation of complex geometry.

When modeling compressible fluid dynamics, one of the significant challenges is managing discontinuities and shocks. Finite-volume methods (FVMs) are an especially good fit for solving flows that experience discontinuities because FVMs are inherently conservative. FVMs have been well-developed for low-order accuracy [2]. Modern research codes extend FVMs to high-order accuracy, that is $O(h^Q)$ with $Q \geq 3$ and h the spatial cell size. High-order FVMs decrease the numerical errors far more rapidly as the mesh is refined for smooth flows. Despite the improved accuracy and efficiency of high-order FVMs, they are less prevalent because they require more effort to understand and develop algorithms that are robust and efficient [3]. High-order FVMs do require more computation per cell, although they also better utilize modern computer architectures.

Applying a high-order FVM to structured grids greatly facilitates the solution process, due to the simple data layout, straightforward solution reconstruction, and well-understood error properties. An additional motivation for using structured grids is to permit efficient adaptive mesh refinement (AMR). AMR is a method of allowing refinement to be added in local regions of interest while the solution is running, which allows for higher solution accuracy with reduced computational expense. AMR at high-order is viable on structured grids due to a straightforward process of refining and coarsening [4]. Nevertheless, structured grids do have a significant drawback — representing complex geometries is difficult, especially at high-order. Therefore, this research

aims to develop efficient and accurate geometric capabilities to represent complex geometries for high-order FVMs on structured grids with AMR.

1.2 Objectives

The goal of this work is to enable high-ordered finite-volume methods to operate on structured grids with AMR for complex geometries of interest to practical engineering. The challenge is — how to effectively represent complicated geometries using a structured grid with AMR. To overcome the challenge and achieve this goal, three distinct methods are defined. First, we develop a strategy for more robust AMR, allowing for use in a larger variety of engineering applications. Second, we develop the mapped multi-block (MMB) method to represent complex geometries defined by discrete grids. The MMB method supports high-order accurate geometries with AMR but has geometric limitations, in addition to the fact that creating meshes requires both expertise and time. Third, the embedded-boundary (EB) method is developed for high-order FVMs as a promising new approach for highly complex geometry representation where the MMB method would encounter difficulties. EB methods are less accurate for the same resolution than the MMB method due to the cut cells on the boundaries. However, the EB method permits complete automation of the meshing process for any geometry. Development for each of these three methods not only stem from a common goal but also a shared set of mathematical tools.

Specifically, the primary objectives of this research are:

1. Develop a robust method for high-order AMR. This addresses the stability challenges faced by high-order interpolation across AMR levels and at multi-block boundaries. This is important for applications involving multi-scales and multi-physics such as turbulent combustion.

2. Establish a standard CFD workflow for mapped structured grids that permits high-order accuracy and AMR. This entails the use of grids that are generated by common mesh generation software where it is standard practice to produce grids defined by discrete points. In order to effectively apply AMR and high-order features, a mapping function describing the shape of the structured grid is required to be constructed from a discrete grid.

3. Enable the MMB method to represent general geometries of moderate complexity. This method should maintain fourth-order accuracy for smooth flows, strictly maintain conservation, and operate with AMR. The approach taken to achieve this is to extend an existing conforming mapped multi-block method from operating on smooth analytic geometries to general geometries coming from external mesh generation software.

4. Extend the EB method at fourth-order accuracy to solve the Stokes equations while accommodating highly-complex geometries. In the CFD community, development of EB methods is an active research area due to their ability to completely automate the mesh generation process even for challenging geometries.

5. Demonstrate that with the development of the new methods in this dissertation, complex fluid flows influenced by geometries are possible with the fourth-order FVM on structured grids with AMR.

1.3 Dissertation Organization

This dissertation is structured as follows. The mathematical and numerical framework supporting this research is presented in Chapter 2 for completeness and convenience. In Chapter 3, the new adaptive clipping-and-redistribution method is described and discussed, which improves the robustness of AMR by providing conservative bounds-preservation for multi-dimensional interpolation. When using AMR with a high-order FVM, high-order numerical interpolation between different levels of refinement is required. However, this interpolation sometimes leads to numerical issues when physical bounds are violated. The new method overcomes two major challenges of the high-order interpolation method. First, the new method is bound-preserving near extrema or discontinuities to prevent the emergence of unphysical oscillations while maintaining the fourth-order accuracy in smooth flows. Second, the new method satisfies the conservation requirement in multiple dimensions, particularly in the context of generalized curvilinear coordinate transformations. Additionally, the new method is designed to be localized and computationally inexpensive.

Chapter 4 presents the conforming MMB method for handling structured grids that are discretely defined for general complex geometries. The method effectively represents complex geometries by connecting multiple structured grids along block boundaries. It is designed for high-order FVMs with AMR and maintains strict solution conservation over block boundaries by extending the mapping into ghost cells and interpolating ghost cells between blocks. This dissertation extends the existing conforming mapped multi-block method for particular geometries that are analytically defined, to operate on arbitrary geometries defined by discrete data points. In Chapter 5 the fourth-order Cartesian EB method is presented. The method employs a weighted least-squares technique for spatial discretization to mitigate the "small cut cell" problem, without mesh modifications, cell merging, or redistribution. Finally, Chapter 6 concludes the dissertation by summarizing the original contributions and novelty of work, and proposes potential directions for future development.

Chapter 2

Mathematical and Numerical Frameworks

In this chapter, the mathematical and numerical frameworks supporting this dissertation are summarized. First, the FVM is described since it is the base CFD method for this dissertation. Next, discretization schemes for FVMs on structured grids and AMR are detailed, including the mapped multi-block, and embedded-boundary strategies. Furthermore, high-order solution reconstruction processes are explained. Mathematical models testing the proposed methods are over-viewed. Finally, the computational framework where this research is implemented, Chord, is described.

2.1 The Finite-Volume Method

This research employs FVMs to discretize partial differential equations (PDEs) in the spatial domain [5]. One of the advantages of FVMs is that solution conservation is satisfied naturally without special consideration of the numerical scheme. This conservative property is valuable for many applications such as the Navier-Stokes equations where conservation of mass, momentum, and energy, are fundamental physical constraints.

The divergence form of the conservation law governing fluid dynamics can be written as

$$\frac{\partial}{\partial t}\mathbf{U} + \nabla_{\boldsymbol{x}} \cdot \vec{\mathbf{F}} = \mathbf{S}\,, \tag{2.1}$$

where $\mathbf{U}$ is a vector of the solution quantity that is conserved, and varies in space $\boldsymbol{x}$ and time t. The tensor $\vec{\mathbf{F}}$ is the flux of $\mathbf{U}$, and the source term vector $\mathbf{S}$ is the production of $\mathbf{U}$ per unit time. To apply the FVM [2], the PDE system is converted to an integral form over finite volumes $\mathcal{V}_{\boldsymbol{i}}$ in space, and the divergence theorem is applied to the flux term yielding

$$\frac{\partial}{\partial t}\int_{\mathcal{V}_{\boldsymbol{i}}} \mathbf{U}\,\mathrm{d}\boldsymbol{x} + \int_{\partial\mathcal{V}_{\boldsymbol{i}}} \vec{\mathbf{F}} \cdot \hat{\mathbf{n}}\,\mathrm{d}\boldsymbol{x} = \int_{\mathcal{V}_{\boldsymbol{i}}} \mathbf{S}\,\mathrm{d}\boldsymbol{x}\,. \tag{2.2}$$

The flux term may be broken into separate regions over the volume surface $\partial\mathcal{V}_{\boldsymbol{i}}$ as

$$\int_{\partial\mathcal{V}_{\boldsymbol{i}}} \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \, \mathrm{d}\boldsymbol{x} = \sum_{\boldsymbol{f} \in \partial\mathcal{V}_{\boldsymbol{i}}} \int_{\mathcal{A}_{\boldsymbol{f}}} \vec{\mathbf{F}} \cdot \hat{\mathbf{n}}_{\boldsymbol{f}} \, \mathrm{d}\boldsymbol{x} \,,$$

where $\boldsymbol{f}$ indexes a particular face of the cell indexed by $\boldsymbol{i}$, and $\hat{\mathbf{n}}_{\boldsymbol{f}}$ is the corresponding outward normal vector. We define cell averaged and face averaged quantities as

$$\langle \mathbf{U} \rangle_{\boldsymbol{i}} = \frac{1}{\mathcal{V}_{\boldsymbol{i}}} \int_{\mathcal{V}_{\boldsymbol{i}}} \mathbf{U} \, \mathrm{d}\boldsymbol{x} \,, \qquad \langle \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}} = \frac{1}{\mathcal{A}_{\boldsymbol{f}}} \int_{\mathcal{A}_{\boldsymbol{f}}} \vec{\mathbf{F}} \cdot \hat{\mathbf{n}}_{\boldsymbol{f}} \, \mathrm{d}\boldsymbol{x} \,, \tag{2.3}$$

respectively. Using the averaged notation, the finite volume scheme is expressed in terms of averages as the semi-discrete form

$$\frac{\mathrm{d}}{\mathrm{d}t} \langle \mathbf{U} \rangle_{\boldsymbol{i}} + \sum_{\boldsymbol{f} \in \partial\mathcal{V}_{\boldsymbol{i}}} \frac{\mathcal{A}_{\boldsymbol{f}}}{\mathcal{V}_{\boldsymbol{i}}} \langle \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}} = \langle \mathbf{S} \rangle_{\boldsymbol{i}} \,. \tag{2.4}$$

This is an exact formulation, since no approximations have been made. In practice, the fluxes and time derivatives are not known exactly, so numerical approximations of each are required in order to solve the solution state. The order of the solution accuracy depends upon the methods used to evaluate fluxes and time derivatives. This dissertation specifically focuses on approximation methods that are fourth-order in space and time, with approaches further detailed by McCorquodale et al. [4] and in Section 2.3. The approximations themselves are highly-dependent upon the choice of discretization in space and time. This dissertation explores space discretization on structured grids which allow for simplifications for flux reconstruction.

2.2 Structured Grids

Structured grids are defined by a Cartesian space with integer points $\boldsymbol{i} \in \mathbb{Z}^D$ which mark cell centers, where D is the number of spatial dimensions [4, 6]. The problem domain to operate in is a subset of the integer lattice $\Gamma \subset \mathbb{Z}^D$. Each cell is defined by computational volume $V_{\boldsymbol{i}} = [(\boldsymbol{i} - \frac{1}{2}\boldsymbol{I})h, (\boldsymbol{i} + \frac{1}{2}\boldsymbol{I})h]$, where $\boldsymbol{I}$ is the vector with all components equal to one, and h is the cell

size. The faces of each cell are regularly spaced and aligned with the coordinate system. This allows faces $\boldsymbol{f}$ to be indexed by $\boldsymbol{i} \pm \frac{1}{2}\boldsymbol{e}^d$ where $\boldsymbol{e}$ is a unit vector in direction d. This notation for indexing structured grids is illustrated in Figure 2.1.

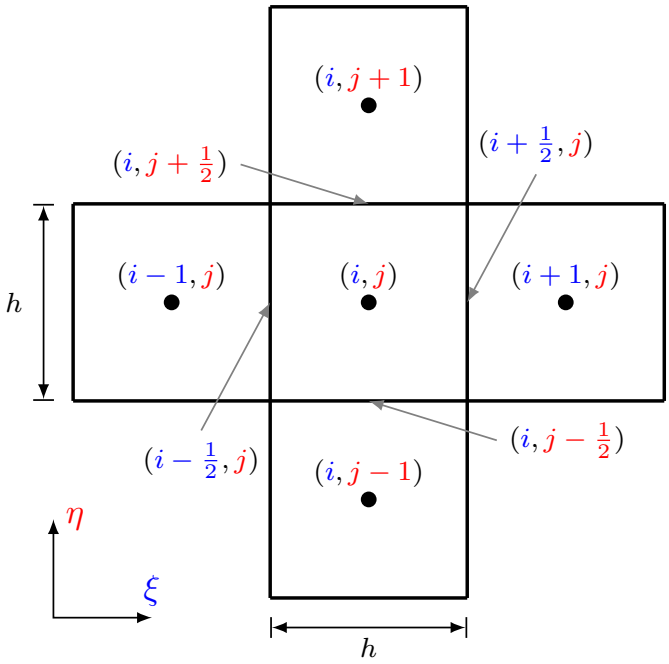


Figure 2.1: Grid indexing notation for a structured grid in two dimensions, where $\boldsymbol{\xi} = (i, j)$.

When the FVM is restricted to structured grids, the flux terms in Equation 2.4 can be simplified since faces are oriented with the coordinate system, and cells are regularly sized. Additionally, because all cells share the same computational volumes and face areas, the $\frac{\mathcal{A}_f}{\mathcal{V}_i}$ term reduces to $\frac{1}{h}$ for all cells. Accounting for the regularity of structured grids, Equation 2.4 can be simplified to

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle \mathbf{U} \rangle_{\boldsymbol{i}} + \frac{1}{h} \sum_{d=1}^{D} \left(\langle \mathbf{F}_d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \mathbf{F}_d \rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d} \right) = \langle \mathbf{S} \rangle_{\boldsymbol{i}} \,. \tag{2.5}$$

2.2.1 Mapped Grids

One approach to accommodate geometries using a structured grid is by applying a grid mapping, as illustrated in Figure 2.2. Geometries are represented by transforming the Cartesian grid in computational space, indexed by coordinates $\boldsymbol{\xi}$, into a physical space, denoted by coordinates $\boldsymbol{x}$.

The transformation between the two spaces is defined by the forward mapping $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi})$ and the inverse mapping $\boldsymbol{\xi} = \boldsymbol{\xi}(\boldsymbol{x})$, as illustrated in Figure 2.3. A valid mapping function and its inverse must both be one-to-one in the domain they are used. The forward mapping function must always be known and readily evaluated. On the other hand, the inverse mapping is generally not directly required or known. Throughout this dissertation *the mapping function* when referred to, and unless explicitly stated otherwise, indicates the forward mapping $\boldsymbol{x}(\boldsymbol{\xi})$.

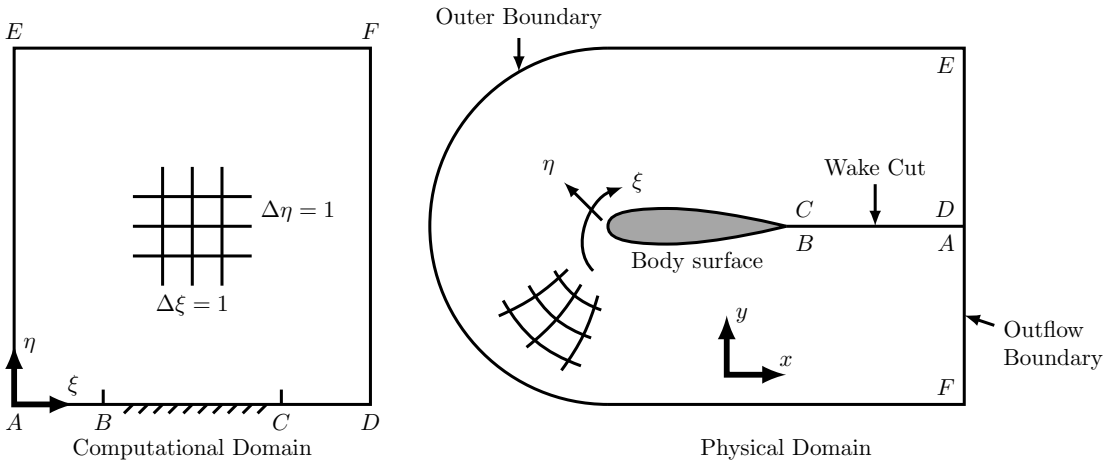


Figure 2.2: A representation of an airfoil geometry with a mapped structured grid.

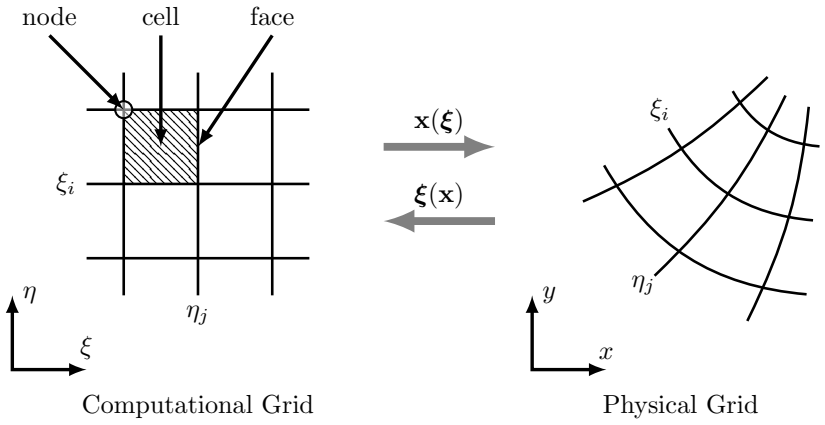


Figure 2.3: A representation of the transformation between computational and physical space.

This section considers the high-order FVM on single-block analytically-mapped grids. The effect of the mapping is summarized here, but a more detailed description is available for single-level mapped in Colella et al. [7] and for adaptively-refined mapped grids in Guzik et al. [6, 8].

On mapped grids, the control volumes in physical space are defined by $\mathcal{V}_{\boldsymbol{i}} = \mathbf{x}(V_{\boldsymbol{i}})$. The integral form of the PDE in Equation 2.2 is transformed from physical space into computation space, $\boldsymbol{\xi}$, by using grid metric terms such that

$$\frac{\partial}{\partial t}\int_{V_{\boldsymbol{i}}} J\mathbf{U}\,\mathrm{d}\boldsymbol{\xi} + \int_{V_{\boldsymbol{i}}} \nabla_{\boldsymbol{\xi}}\cdot(\mathrm{N}^{\intercal}\vec{\mathbf{F}})\,\mathrm{d}\boldsymbol{\xi} = \int_{V_{\boldsymbol{i}}} J\mathbf{S}\,\mathrm{d}\boldsymbol{\xi}\,,$$

where the transformation matrix, $\mathrm{N}^{\intercal} = J\nabla_{\boldsymbol{x}}\boldsymbol{\xi}$, describes the grid metrics and the metric Jacobian $J \equiv \det(\nabla_{\boldsymbol{\xi}}\boldsymbol{x})$. After applying the divergence theorem of Gauss, the integrals can be represented as cell averaged values on each face, yielding

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle J\mathbf{U}\rangle_{\boldsymbol{i}} + \frac{1}{h}\sum_{d=1}^{D}\left(\langle \mathrm{N}_d^{\intercal}\vec{\mathbf{F}}\rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} - \langle \mathrm{N}_d^{\intercal}\vec{\mathbf{F}}\rangle_{\boldsymbol{i}-\frac{1}{2}\boldsymbol{e}^d}\right) = \langle J\mathbf{S}\rangle_{\boldsymbol{i}}\,, \tag{2.6}$$

where the subscript d denotes the d^{th} row of $\mathrm{N}^{\intercal}$. Note that this is an exact formulation.

The algorithm for arriving at a high-order, free-stream preserving $\langle \mathrm{N}_d^{\intercal}\vec{\mathbf{F}}\rangle$ from $\mathrm{N}_d^{\intercal}\vec{\mathbf{F}}$ is detailed in work by Guzik et al. [8], which provides the framework for the present study. Solving these quantities requires the mapping $\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{\xi})$ and its derivatives on codimension two elements of the grid (e.g., vertices in 2-D and edges in 3-D). Additionally, this framework is compatible with adaptive mesh refinement, although the mapping function is required at any potential location inside the domain.

Grid Metrics

To define the mapping transformation terms J and $\mathrm{N}^{\intercal}$, grid metrics are required. The grid metrics are the set of directional derivatives, which define the grid mapping and are needed for calculation. The metrics and inverse metrics are defined respectively as

$$\nabla_{\boldsymbol{\xi}} \boldsymbol{x} \equiv \frac{\partial(x_1, x_2, x_3)}{\partial(\xi_1, \xi_2, \xi_3)} = \begin{bmatrix} \frac{\partial x_1}{\partial \xi_1} & \frac{\partial x_1}{\partial \xi_2} & \frac{\partial x_1}{\partial \xi_3} \\ \frac{\partial x_2}{\partial \xi_1} & \frac{\partial x_2}{\partial \xi_2} & \frac{\partial x_2}{\partial \xi_3} \\ \frac{\partial x_3}{\partial \xi_1} & \frac{\partial x_3}{\partial \xi_2} & \frac{\partial x_3}{\partial \xi_3} \end{bmatrix}, \tag{2.7}$$

$$\nabla_{\boldsymbol{x}} \boldsymbol{\xi} \equiv \frac{\partial(\xi_1, \xi_2, \xi_3)}{\partial(x_1, x_2, x_3)} = \begin{bmatrix} \frac{\partial \xi_1}{\partial x_1} & \frac{\partial \xi_1}{\partial x_2} & \frac{\partial \xi_1}{\partial x_3} \\ \frac{\partial \xi_2}{\partial x_1} & \frac{\partial \xi_2}{\partial x_2} & \frac{\partial \xi_2}{\partial x_3} \\ \frac{\partial \xi_3}{\partial x_1} & \frac{\partial \xi_3}{\partial x_2} & \frac{\partial \xi_3}{\partial x_3} \end{bmatrix}. \tag{2.8}$$

Through definition it is known that $\nabla_{\boldsymbol{x}} \boldsymbol{\xi} = (\nabla_{\boldsymbol{\xi}} \boldsymbol{x})^{-1}$.

While these quantities are known at any point, the average $\langle \mathrm{N}_d^{\intercal} \rangle$ on a face, as used in Equation 2.6, is determined using a more elaborate process to ensure freestream preservation [8]. Ultimately, $\langle \mathrm{N}_d^{\intercal} \rangle$ is computed on a face normal to direction d from a line integral of $\mathcal{N}_{s,(d,d')}, d' \neq d$ around the edges of the face. $\mathcal{N}_{s,(d,d')}$ is defined as

$$\mathcal{N}_{s,(d,d')} = \frac{1}{D-1} \det((\nabla_{\boldsymbol{\xi}} \boldsymbol{x})^{\intercal}(d|\boldsymbol{e}^s)(d'|\boldsymbol{x}))\,, \tag{2.9}$$

where $\mathrm{A}(p|\boldsymbol{v})$ denotes a modification of matrix A by replacing row p with vector $\boldsymbol{v}$. At interfaces of mesh resolution, $\mathcal{N}_s$ is not continuous across the face and a one-order loss of accuracy can be observed. For this reason, the metrics computed should be approximated to $O(h^{p+1})$ accuracy in order for $\langle \mathrm{N}_d^{\intercal} \rangle$ to be accurate to $O(h^p)$ everywhere [8].

2.2.2 Mapped Multi-Block Grids

Mapped grid approaches are useful to accommodate geometries on structured grids, but have substantial restrictions in their capabilities. For example, creating a mapped grid with multiple bodies or holes in the domain is not feasible in general. The mapped grid approach can be extended to represent a much greater range of geometries using the mapped multi-block, or MMB, method. In the MMB framework, the global domain is subdivided into a number of regions called blocks. Each block is defined by a block-domain in computational space, $\mathcal{D} \subset \mathbb{R}^D$, and a block-range in physical space, $\mathcal{R} \subset \mathbb{R}^D$, as illustrated in Figure 2.4. Each block has a unique block-domain, block-range, and mapping function. High-order methods with overlapping grids have shown great success [9], although at the loss of strict conservation at block intersections. This dissertation specifically builds upon the conforming MMB approach of McCorquodale et al [10]. The conforming multi-block scheme ensures strict conservation, high-order accuracy, and free-stream preservation.

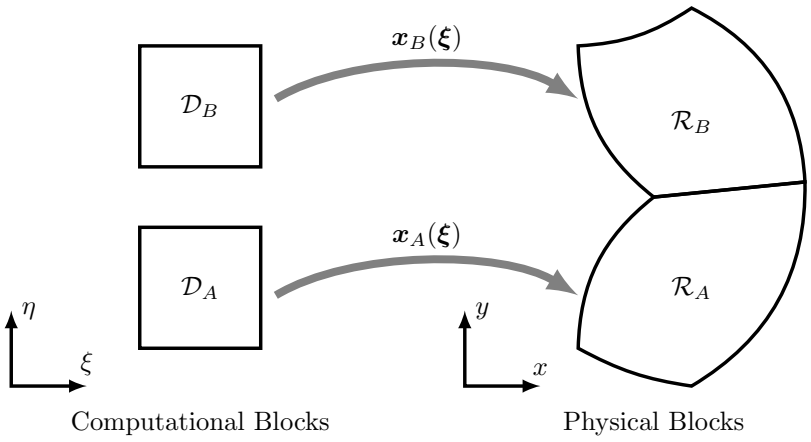
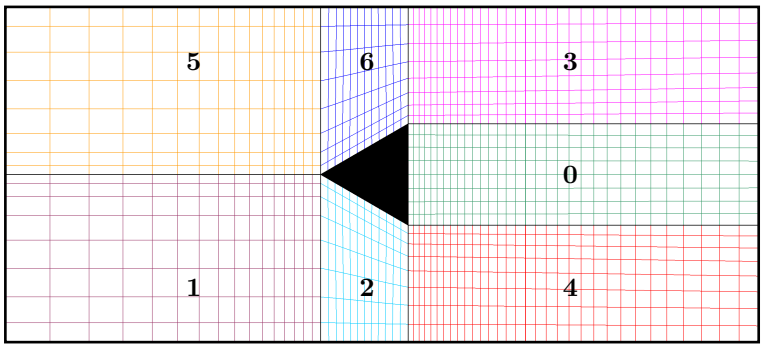


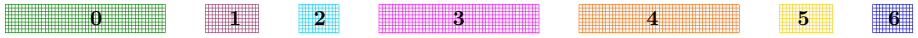
Figure 2.4: A sample multi-block system with two blocks.

Restriction to conforming mapped multi-blocks requires that the block-ranges connect exactly at the block-domain boundaries without overlapping. When two blocks are connected, i.e. their block-ranges intersect at a boundary, an additional restriction is enforced. That is, for connected blocks A and B it must be that $\boldsymbol{x}_A{}^{-1}(\mathcal{R}_A \cup \mathcal{R}_B)$ and $\boldsymbol{x}_B{}^{-1}(\mathcal{R}_A \cup \mathcal{R}_B)$, which are subsets of bound-

aries in $\mathcal{D}_A$ and $\mathcal{D}_B$ respectively, are isometries of one another [10]. In effect, this enforces that every cell has a single neighbor per face and can use a single flux value to maintain conservation. As an illustration, Figure 2.5 shows a conforming mapped multi-block grid; Figure 2.5a shows a global physical domain containing 7 blocks (i.e., block-domains) and Figure 2.5b shows the 7 blocks in the computational domain (i.e., block-ranges).



(a) A sample mapped multi-block grid in physical space.



(b) A sample mapped multi-block grid in computational space.

Figure 2.5: A sample conforming mapped multi-block grid. Seven blocks are shown in this example, each with a mapping function from its Cartesian computational grid to the curvilinear one in physical space.

For mapped multi-block grids, considerations for flux stencils next to block interfaces must be made. The present study applies the same centered scheme everywhere in order to avoid inappropriate bias at the multi-block boundaries. This is achieved by employing ghost cells, which are discussed in Chapter 4. Inertial fluxes are made single valued at block interfaces by solving a Riemann problem. Viscous fluxes are made single valued by averaging.

2.2.3 Embedded-Boundary Grids

Mapped multi-block methods are a powerful method for geometry representation using structured grid methods, but generating grids requires time and expertise for complex geometries. Geometries that are porous or have rough surfaces are exceptionally difficult, if feasible at all, to represent using the MMB method. The embedded-boundary, or EB, method takes a different ap-

proach to geometry representation on structured grids. EB grids are created by first generating a Cartesian grid independent of boundary geometry. Then, the boundary geometry of sufficient order [11] is overlain on the grid, and cells that intersect the boundaries are cut. The resulting grid is one that is structured everywhere away from the boundary, while near the boundary the grid is made up of partial, or "cut" cells. The resulting method retains the advantages of solving on structured grids on the interior of the domain, while having relatively little restriction in geometries that may be represented. Additionally, this grid generation process can be automated and performed quickly. In the EB method, cells fall into one of three categories; regular cells, irregular cells, and invalid cells. This distinction between cell types is illustrated in Figure 2.6. Regular cells are those that are full Cartesian cells. Irregular cells, or cut cells, are those which are partial cells because they intersect with the boundary geometry. Invalid cells, as the name indicates, are cells that fall outside the domain boundaries and are thus not in the solution domain. To denote the embedded-boundary regions, let Ω be the irregular domain and $V_{\boldsymbol{i}}$ be any regular cell, then a particular cell can be denoted by $\mathcal{V}_{\boldsymbol{i}} = V_{\boldsymbol{i}} \cap \Omega$.

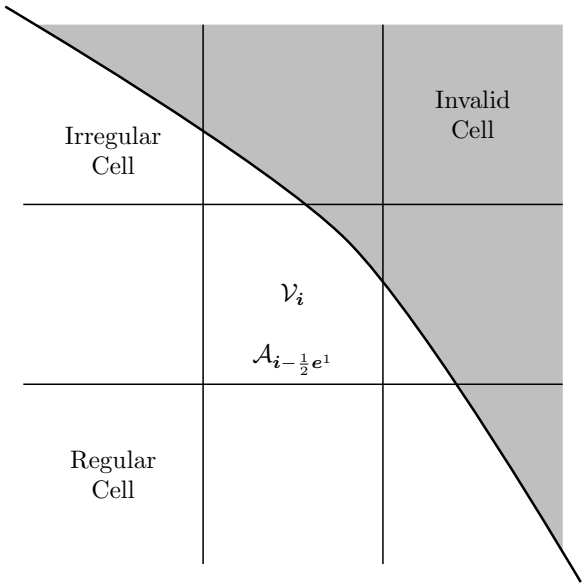


Figure 2.6: Illustration of notation used for embedded-boundary methods. The shaded region lies outside the problem domain of interest.

The challenge of the EB method comes from the grid being irregular at the boundaries. As a result, the convenient notation for the FVM as in Equation 2.5 can not be used in the presence of cut cells. To address this, the present study continues and extends the work by Devendran et al. [12], and the procedure is briefly summarized in this section. A cell fraction term is introduced as

$$\kappa = \frac{\mathcal{V}_{\boldsymbol{i}}}{h^D} \tag{2.10}$$

which varies between 0 for an invalid cell and 1 for a regular cell. When in the presence of cut cells, the approach for Equation 2.5 can no longer be used. Further, cells can become arbitrarily small and volume terms $\mathcal{V}_{\boldsymbol{i}}$ can approach zero. To prevent division by zero, a κ-weighted form for the Equation 2.4 is used as

$$\frac{\mathrm{d}}{\mathrm{d}t}\langle \kappa \mathbf{U} \rangle_{\boldsymbol{i}} + \frac{1}{h^D} \sum_{\boldsymbol{f} \in \partial \mathcal{V}_{\boldsymbol{i}}} \mathcal{A}_{\boldsymbol{f}} \langle \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}} = \langle \kappa \mathbf{S} \rangle_{\boldsymbol{i}} \,, \tag{2.11}$$

to avoid the small cell problem. The faces of each cell, $\boldsymbol{f}$, consist of up to $2D$ Cartesian faces on in interior, and a number of curved faces on the boundaries.

Another, difficulty resulting from the mesh being irregular at the boundaries is that flux terms $\langle \vec{\mathbf{F}} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}}$ can no longer be solved using the convenient form available for regular structured grids. On irregular regions, our approach is to create a weighted least-squares polynomial interpolation from the cell average values, and use that to evaluate the required flux terms.

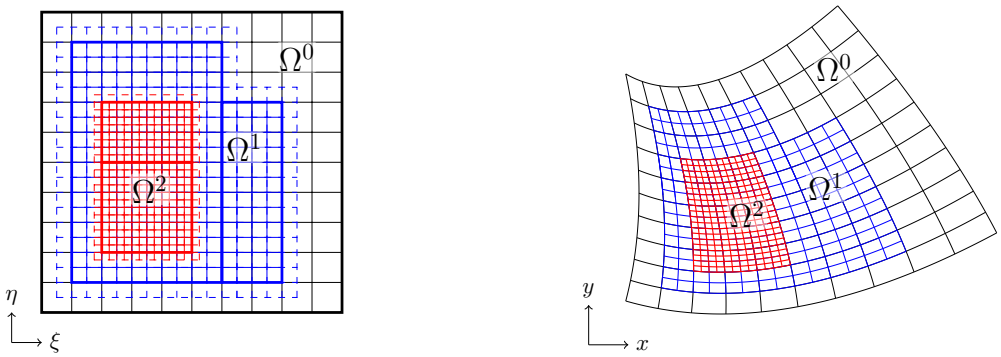
2.2.4 Adaptive Mesh Refinement

One of the advantages to operating on structured grids is that refining the grid is straightforward, and the ability to do so based on solutions is a key feature for the high-order FVM this dissertation expands on [13]. Adaptive mesh refinement, or AMR, allows for an efficient approach to solve multi-scale problems. As the name implies, AMR operates by changing the mesh resolution locally depending upon the solution state. Regions where a solution is under-resolved or are of particular interest may be solved on finer meshes, while regions that are well resolved and are

of little interest remain on coarser meshes. When discontinuities appear in a solution, high-order methods must locally revert to low-order methods around discontinuities. The decrease in order of the scheme is necessary and adversely impacts the solution accuracy. To remedy this, locally refining the grid near the discontinuities with AMR is the best option available. Traditionally, discontinuities are well-embedded inside the fine grid. Layers of buffer cells ensure that interpolations to fill ghost cells and the fine patches are kept away from any discontinuities. In some recent non-traditional approaches, such as embedded-DNS, discontinuities are allowed to cross AMR interfaces. The AMR approach used in this work is detailed by McCorquodale and Colella [4] and Guzik et al. [8], and relevant portions of the AMR algorithm are described here to facilitate understanding.

For AMR on structured grids, the idea of a domain on the integer lattice $\Gamma \subset \mathbb{Z}^D$ is extended to a hierarchy of grids $\Gamma^0 \ldots \Gamma^{\ell_{\max}}$, each with a different cell size and nested such that $\Gamma^\ell = \mathcal{C}_{n^\ell_{\text{ref}}}(\Gamma^{\ell+1})$. Figure 2.7 illustrates a 3-level AMR grid. The integer n^ℓ_{ref} is the refinement ratio between level ℓ and $\ell+1$ so that the Cartesian mesh spacing $h^\ell = n^\ell_{\text{ref}} h^{\ell+1}$. The operator $\mathcal{C}$ is a coarsening of the grid, and the inverse $\mathcal{C}^{-1}$ is the refinement. AMR calculations are performed on a hierarchy of nested meshes $\Omega^\ell \subset \Gamma^\ell$, with $\Omega^\ell \supset \mathcal{C}_{n^\ell_{\text{ref}}}(\Omega^{\ell+1})$. The grid levels are considered as overlapping rather than embedded. At level ℓ all cells inside Ω^ℓ are labeled valid and all cells outside Ω^ℓ are labeled invalid. Valid ghost cells are used to communicate data on a single level split over multiple computational units. Invalid ghost cells are those used to transfer information at the interface between a coarse and fine level. It is enforced that there are a sufficient number of cells on level ℓ separating the level $\ell+1$ from the level $\ell-1$ such that interpolations to fill invalid ghost cells on finer levels can be independently performed.

In addition to the refinement in space, AMR can make use of sub-cycling for refinement in time, as detailed by McCorquodale and Colella [4]. Briefly, sub-cycling updates each grid level using a different time step size to reduce computational costs. Solutions on level $\ell+1$ use a time step sized as $\Delta t_{\ell+1} = \Delta t_\ell / n^\ell_{\text{ref}}$. Levels $\ell+1$ and ℓ are synchronized at the beginning of each time step on ℓ, and operate independently between these time synchronization points. During this



(a) Computational Domain **(b)** Physical Domain

Figure 2.7: A three-level grid with $n_{\text{ref}} = 2$, with each level shown by a different color. Nesting sufficient for one cell to separate level $\ell + 1$ from level $\ell - 1$ exists. One layer of invalid ghost cells are shown on the computational domain using the dashed lines for each level.

synchronization, the solution values on level ℓ are updated to match the cell-averages from $\ell + 1$, and fluxes at coarse-fine interfaces are corrected to ensure conservation. Flux correction replaces the fluxes on the coarse grid with those from the fine grid at the interface between the two levels. At intermediate steps on the fine level $\ell + 1$, ghost cells are interpolated from the coarse level ℓ in space and time.

Conservative Interpolation Schemes for AMR

When using explicit time-marching on an AMR grid, each level is separately evolved with data transfer between levels occurring at only refinement boundaries. From coarse to fine, data is interpolated. There are two AMR operations where interpolation is used directly. When regridding, a new level $\ell + 1$ is created, and its valid region is interpolated from existing coarse data on level ℓ. This interpolation must be conservative to maintain the strict conservation property of finite-volume schemes. At AMR interfaces, a number of operations to communicate between levels ℓ and $\ell + 1$ rely on having invalid ghost cells of level $\ell + 1$ interpolated from level ℓ. An example grid with one layer of invalid ghost cells is show in Figure 2.7. This interpolation needs not be conservative because invalid ghost cells are only intermediate states used for flux reconstruction, although enforcing conservation may improve the quality of the solution [8]. These

two interpolations between levels have slightly different implementations, but both follow this procedure:

1. For each coarse cell $\boldsymbol{i}^\ell$, determine an interpolation stencil $\mathcal{I}(\boldsymbol{i}^\ell)$. The stencil must have sufficient degrees of freedom for the desired order of accuracy.

2. Construct a scheme-order multi-dimensional polynomial using the data in $\mathcal{I}(\boldsymbol{i}^\ell)$ while maintaining conservation of $\langle \mathbf{U} \rangle_{\boldsymbol{i}^\ell}$.

3. For each desired fine cell $\boldsymbol{i}^{\ell+1} \in \mathcal{C}^{-1}(\boldsymbol{i}^\ell)$, evaluate the cell-averaged value using the prior reconstruction.

The interpolation procedure for both newly refined cells and invalid ghost cells as described by McCorquodale and Colella [4] is a form of k-exact reconstruction [14]. This particular interpolation strategy is further detailed in the following Section 2.3.4

2.3 High-Order Reconstruction

One of the fundamental aspects of the FVM used in this dissertation is the high-order approach used to numerically solve Equation 2.4. Doing so is based upon the idea of interpolating data using a Taylor series expansion of desired accuracy, and evaluating the resulting function for required data such as face average values for fluxes. A common framework for this interpolation strategy is presented, which is later used for a number of specific applications through this dissertation.

2.3.1 Multi-Dimensional Polynomials

When interpolating a solution for operations such as evaluating the fluxes, the approach is to construct a piecewise interpolant than can be evaluated as required. For a scalar variable ϕ, the piecewise interpolation for cell $\boldsymbol{i}$ is defined by the function $\phi_{\boldsymbol{i}}(\boldsymbol{\xi})$. To achieve a specified order of accuracy, Q, a multi-dimensional Taylor expansion about the center of cell $\boldsymbol{i}$ is used. Multi-dimensional notation is defined using $\mathbf{q}$ as an integer vector of size D. Vector powers of $\mathbf{q}$ of

vector quantities $\boldsymbol{\xi}$ are expressed as

$$(\boldsymbol{\xi} - \boldsymbol{\xi_i})^{\mathbf{q}} = \prod_{d=1}^{D} (\boldsymbol{\xi}_d - \boldsymbol{\xi}_{\boldsymbol{i},d})^{\mathrm{q}_d} \,. \tag{2.12}$$

A multi-dimensional form of the Taylor series up to order Q, centered about the point $\bar{\boldsymbol{\xi}}$, is expressed as

$$\phi(\boldsymbol{\xi}) = \sum_{\|\mathbf{q}\|_1 < Q} \frac{1}{\mathbf{q}!} \phi^{(\mathbf{q})}(\bar{\boldsymbol{\xi}})(\boldsymbol{\xi} - \bar{\boldsymbol{\xi}})^{\mathbf{q}} + O(h^Q) \,, \tag{2.13}$$

where $\phi^{(\mathbf{q})}(\bar{\boldsymbol{\xi}})$ is the $\mathbf{q}^{\text{th}}$ derivative in multiple dimensions (e.g., $\frac{\partial^{(q_1+q_2+q_3)}\phi}{\partial \xi^{q_1} \partial \eta^{q_2} \partial \zeta^{q_3}}$), and h is the cell spacing.

The moments, or definite integrals, about the cells of the Taylor polynomial are defined as

$$m_{\boldsymbol{i}}^{\mathbf{q}}(\bar{\boldsymbol{\xi}}) = \int_{\mathcal{V}_{\boldsymbol{i}}} (\boldsymbol{\xi} - \bar{\boldsymbol{\xi}})^{\mathbf{q}} \, \mathrm{d}\boldsymbol{\xi} \,, \tag{2.14}$$

for volume moments, or similarly for face moments

$$m_{\boldsymbol{f}}^{\mathbf{q}}(\bar{\boldsymbol{\xi}}) = \int_{\mathcal{A}_{\boldsymbol{f}}} (\boldsymbol{\xi} - \bar{\boldsymbol{\xi}})^{\mathbf{q}} \, \mathrm{d}\boldsymbol{\xi} \,. \tag{2.15}$$

The moments depend only on the grid, and so may be cached for efficiency. Moments are computed analytically for structured grids. Near embedded boundaries moments generally can not be known analytically, so an approximation of required order is computed using the algorithm by Schwartz et al. [11]. Choosing $c_{\mathbf{q}} = \frac{1}{\mathbf{q}!}\phi^{(\mathbf{q})}(\bar{\boldsymbol{\xi}})$ as the unknowns to be solved, the Taylor polynomial in Equation 2.13 working with cell-averaged quantities, defined in Equation 2.3, can be written

compactly as

$$\langle \phi(\boldsymbol{\xi}) \rangle_{\boldsymbol{i}} = \frac{1}{\mathcal{V}_i} \int_{\mathcal{V}_{\boldsymbol{i}}} \left(\sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} (\boldsymbol{\xi} - \bar{\boldsymbol{\xi}})^{\mathbf{q}} + O(h^Q) \right) \mathrm{d}\boldsymbol{\xi} \tag{2.16}$$

$$= \sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} \langle (\boldsymbol{\xi} - \bar{\boldsymbol{\xi}})^{\mathbf{q}} \rangle_{\boldsymbol{i}} + O(h^Q) \tag{2.17}$$

$$= \frac{1}{\mathcal{V}_i} \sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} m_{\boldsymbol{i}}^{\mathbf{q}} (\bar{\boldsymbol{\xi}}) + O(h^Q) \,. \tag{2.18}$$

For a unique construction, $K = \binom{D+Q-1}{D}$ terms (binomial coefficients) are needed. Throughout this dissertation, we primarily target constructions of $Q = 4$.

2.3.2 Flux Reconstruction

Aside from grid structure, properties of different finite-volume methods generally arise from the specific algorithm used to evaluate the flux integrals in (2.4). Higher-order algorithms may use some form of quadrature to evaluate the flux integrals. Various interpolations may be devised to interpolate properties on the face, including solution-adaptive variants such as ENO [15, 16] and WENO [17] schemes. This dissertation focuses on fourth-order methods using the approach detailed by Guzik et al. [6]. The starting point for this approach is to use a Taylor series expansion as in Section 2.3.1 for the flux (2.4) about face $\boldsymbol{i} + \frac{1}{2}\boldsymbol{e}^d$, resulting in

$$\int_{\mathcal{A}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}} \mathbf{F}_d \mathrm{d}\boldsymbol{x} = \sum_{0 \le |\mathbf{q}| < Q} \frac{1}{\mathbf{q}!} \vec{\nabla}^{\mathbf{q}} \mathbf{F}_d \big|_{\boldsymbol{x} = \boldsymbol{x}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}} \int_{\mathcal{A}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}} (\boldsymbol{x} - \boldsymbol{x}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d})^{\mathbf{q}} \mathrm{d}\boldsymbol{x} + O(h^{Q+D-1}), \tag{2.19}$$

When $Q = 4$, on a regular structured grid we obtain

$$\langle \mathbf{F}_d \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d} = \frac{1}{h^{D-1}} \int_{\mathcal{A}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}} \mathbf{F}_d \mathrm{d}\boldsymbol{x} = \mathbf{F}_d(\boldsymbol{x}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}) + \frac{h^2}{24} \sum_{d' \neq d} \frac{\partial^2 \mathbf{F}_d}{\partial x_{d'}^2} + O(h^4). \tag{2.20}$$

If the derivatives are replaced by approximations of a suitable order (in this case second-order is sufficient) the resulting approximation of the average of the flux divergence over a cell is $O(h^4)$.

The process of obtaining the values for $\mathbf{F}_d$ itself also requires a reconstruction. This involves constructing point values of $\mathbf{U}_{\boldsymbol{i}}$ from $\langle \mathbf{U} \rangle_{\boldsymbol{i}}$, converting from the conservative to primitive state $\mathbf{W}_{\boldsymbol{i}} = \mathbf{W}(\mathbf{U}_{\boldsymbol{i}})$, constructing cell-averages $\langle \mathbf{W} \rangle_{\boldsymbol{i}}$ from $\mathbf{W}_{\boldsymbol{i}}$, interpolating face averages $\langle \mathbf{W} \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$, constructing point values of $\mathbf{W}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$ from $\langle \mathbf{W} \rangle_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d}$, and then solving for fluxes $\mathbf{F}_d(\mathbf{W}_{\boldsymbol{i}+\frac{1}{2}\boldsymbol{e}^d})$ which are finally used to solve the fourth-order face average fluxes in Equation 2.20. For full details of the process, readers are referred to the work by Guzik et al. [6].

The flux reconstruction approach presented here assumes smooth solutions to interpolate from. When the solution to reconstruct from has discontinuities, the resulting fluxes are prone to oscillations that negatively impact the solution. Traditional CFD approaches for solutions with discontinuities locally revert to low-order methods in their presence [18]. This dissertation specifically uses the PPM limiter [19], which is well known for its application in high-order schemes. This method operates by constructing high-order approximations at cell faces, while ensuring these constructions do not generate new extrema and are monotonic. Furthermore, solution smoothness is accounted for in order to prevent excessive limiting and reduction in the order of accuracy in regions of smooth extrema.

2.3.3 Solution Interpolation Using Least Squares Methods

Although there are some cases where the polynomial expansions Section 2.3.1 can be solved easily, such as in Section 2.3.2 for regular grids, that is often not the case. The general use of polynomial interpolation in the FVM is to reconstruct a solution quantity to order Q in cell $\boldsymbol{i}$ that depends on a stencil of neighborhood $\mathcal{I}(\boldsymbol{i})$. For many cases, there is no obvious choice of stencil neighborhood $\mathcal{I}(\boldsymbol{i})$ that contains exactly the number of known values to uniquely solve the corresponding polynomial.

As a preferred method, we instead choose a regularly shaped neighborhood of size $N > K$ so long as $\boldsymbol{i} \in \mathcal{I}(\boldsymbol{i})$. This allows a stencil choice that appears more regular and has flexibility near boundaries [4]. The polynomial required to fit the data in the chosen neighborhood is then fit *approximately* using the least squares method. This method of least squares interpolation is said

to be k-exact [14], that is polynomial solutions up to order Q are interpolated *exactly* using the approach. Any least squares remainder is of the desired order for smooth solutions. Using this approach, fourth-order accurate methods with AMR have been demonstrated [4, 8].

The least squares method [20] solves an over-determined linear system of equations $\mathrm{MC} = \mathrm{V}$ for the C in all possible $\tilde{\mathrm{C}}$ that minimizes the squared error. This is expressed as

$$\mathrm{C} = \arg\min_{\tilde{\mathrm{C}}} \left\| \mathrm{V} - \mathrm{M}\tilde{\mathrm{C}} \right\|_2 \tag{2.21}$$

where $\mathrm{C} \in \mathbb{R}^n$, $\mathrm{V} \in \mathbb{R}^m$, $\mathrm{M} \in \mathbb{R}^{m \times n}$, and $n < m$. Solving this system is done by using the pseudo-inverse [20], where the unknown vector in the least squares method is solved for as

$$\mathrm{C} = \mathrm{M}^{\dagger}\mathrm{V} \,. \tag{2.22}$$

This pseudo-inverse may be expanded as

$$\mathrm{M}^{\dagger} = \mathrm{M}^{\intercal}(\mathrm{M}\mathrm{M}^{\intercal})^{-1} \,. \tag{2.23}$$

In this dissertation, the primary application of the least squares method is for constructing interpolation polynomials. With this purpose in mind, the unknown vector C corresponds to the polynomial coefficients, the vector V the solution state to interpolate from, and M the matrix of moments.

For example, when $D = 2$ and $Q = 2$ the corresponding coefficient vector of length 6 is

$$\mathrm{C} = \begin{bmatrix} c^{(0,0)} & c^{(1,0)} & c^{(2,0)} & c^{(0,1)} & c^{(1,1)} & c^{(0,2)} \,. \end{bmatrix}^{\intercal}$$

The solution data of component d to interpolate from can be collected into a vector as $\mathrm{V} = [\langle \mathbf{U}_d \rangle_{\boldsymbol{j}} : \boldsymbol{j} \in \mathcal{I}(\boldsymbol{i})]$. The moment matrix is expressed using condensed notation where $m^{\mathbf{q}}_{\boldsymbol{j}_1} =$

$m_{\boldsymbol{j}}^{\mathbf{q}}(\boldsymbol{x}_{\boldsymbol{i}})$ for the first $\boldsymbol{j} \in \mathcal{I}(\boldsymbol{i})$, resulting in

$$\mathrm{M} = \begin{bmatrix} \frac{m_{\boldsymbol{j}_1}^{(0,0)}}{\mathcal{V}_{\boldsymbol{j}_1}} & \frac{m_{\boldsymbol{j}_1}^{(1,0)}}{\mathcal{V}_{\boldsymbol{j}_1}} & \frac{m_{\boldsymbol{j}_1}^{(2,0)}}{\mathcal{V}_{\boldsymbol{j}_1}} & \frac{m_{\boldsymbol{j}_1}^{(0,1)}}{\mathcal{V}_{\boldsymbol{j}_1}} & \frac{m_{\boldsymbol{j}_1}^{(1,1)}}{\mathcal{V}_{\boldsymbol{j}_1}} & \frac{m_{\boldsymbol{j}_1}^{(0,2)}}{\mathcal{V}_{\boldsymbol{j}_1}} \\ & \vdots & & & \vdots & \\ \frac{m_{\boldsymbol{j}_k}^{(0,0)}}{\mathcal{V}_{\boldsymbol{j}_k}} & \frac{m_{\boldsymbol{j}_k}^{(1,0)}}{\mathcal{V}_{\boldsymbol{j}_k}} & \frac{m_{\boldsymbol{j}_k}^{(2,0)}}{\mathcal{V}_{\boldsymbol{j}_k}} & \frac{m_{\boldsymbol{j}_k}^{(0,1)}}{\mathcal{V}_{\boldsymbol{j}_k}} & \frac{m_{\boldsymbol{j}_k}^{(1,1)}}{\mathcal{V}_{\boldsymbol{j}_k}} & \frac{m_{\boldsymbol{j}_k}^{(0,2)}}{\mathcal{V}_{\boldsymbol{j}_k}} \\ & \vdots & & & \vdots & \\ \frac{m_{\boldsymbol{j}_N}^{(0,0)}}{\mathcal{V}_{\boldsymbol{j}_N}} & \frac{m_{\boldsymbol{j}_N}^{(1,0)}}{\mathcal{V}_{\boldsymbol{j}_N}} & \frac{m_{\boldsymbol{j}_N}^{(2,0)}}{\mathcal{V}_{\boldsymbol{j}_N}} & \frac{m_{\boldsymbol{j}_N}^{(0,1)}}{\mathcal{V}_{\boldsymbol{j}_N}} & \frac{m_{\boldsymbol{j}_N}^{(1,1)}}{\mathcal{V}_{\boldsymbol{j}_N}} & \frac{m_{\boldsymbol{j}_N}^{(0,2)}}{\mathcal{V}_{\boldsymbol{j}_N}} \end{bmatrix} .$$

2.3.4 Conservative Interpolation for AMR on Mapped Grids

For interpolation between AMR levels, preserving scalar conservation is required. To fulfill this, a constrained least squares problem of the form

$$\mathrm{C} = \arg\min_{\tilde{\mathrm{C}}} \left\| \mathrm{V} - \mathrm{M}\tilde{\mathrm{C}} \right\|_2 \tag{2.24}$$

$$\text{subjet to } \mathrm{B}\tilde{\mathrm{C}} = \mathrm{D} \tag{2.25}$$

is formulated, where the unknowns C are a vector of the coefficients $c_{\mathbf{q}}$. The conservation constraint $\mathrm{BC} = \mathrm{D}$ is enforced using the equation

$$\sum_{\boldsymbol{i}^{\ell+1} \in \mathcal{C}^{-1}(\boldsymbol{i}^{\ell})} \sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} \langle J \hat{\boldsymbol{\xi}}^{\mathbf{q}} \rangle_{\boldsymbol{i}^{\ell+1}} = \langle J\mathbf{U} \rangle_{\boldsymbol{i}^{\ell}} , \tag{2.26}$$

while the objectives $\mathrm{MC} - \mathrm{V}$ to minimize are described by

$$\sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} \langle J \hat{\boldsymbol{\xi}}^{\mathbf{q}} \rangle_{\boldsymbol{j}^{\ell}} = \langle J\mathbf{U} \rangle_{\boldsymbol{j}^{\ell}} \quad \boldsymbol{j}^{\ell} \in \mathcal{I}(\boldsymbol{i}^{\ell}) . \tag{2.27}$$

Once the interpolating function has been constructed by solving coefficients $c_{\mathbf{q}}$, fine cells are evaluated as

$$\langle J\mathbf{U} \rangle_{\boldsymbol{i}^{\ell+1}} = \sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} \langle J \hat{\boldsymbol{\xi}}^{\mathbf{q}} \rangle_{\boldsymbol{i}^{\ell+1}} . \tag{2.28}$$

Due to the conservation constraint, it can be shown that

$$\langle J\mathbf{U}\rangle_{\boldsymbol{i}^\ell} = \sum_{\boldsymbol{i}^{\ell+1}\in\mathcal{C}^{-1}(\boldsymbol{i}^\ell)} \langle J\mathbf{U}\rangle_{\boldsymbol{i}^{\ell+1}}\,, \tag{2.29}$$

using the refinement operator $\mathcal{C}^{-1}$. The values of $\langle J\hat{\boldsymbol{\xi}}^{\mathbf{q}}\rangle$ are computed using a fourth-order product formula [8] and the interpolation step is shown to be freestream-preserving [8]. When filling invalid ghost cells (see Figure 2.7), interpolation of conservative values $\langle\mathbf{U}\rangle$ is sufficient, which is solved using constant values of J in space. For regridding operations where valid cells are filled, conservation of mapped quantities $\langle J\mathbf{U}\rangle$ is required.

In practice, we use interpolation stencils as shown by McCorquodale et al. [4]. For interior solutions, this stencil uses a radius of one cell from the interpolation center with an additional cell in each face-normal direction, resulting in 13 cells in 2D and 33 in 3D. A similar stencil is used near boundaries, but shifted towards the interior to maintain a sufficient number number of cells. (See [4] for specific stencil construction details.)

2.3.5 Weighted Least Squares Stencils

One of the difficulties when using the least squares method for high-order interpolation is that larger interpolation neighborhoods are prone to oscillations that can cause numerical instabilities. One approach to control interpolation of this nature is discussed in the work by Devendran et al. [12], where an additional weighting to the least squares method is found to be beneficial. The least squares formulation in Equation 2.21 is expanded to include weights which tune the importance of satisfying each of the linear equations. A simple but effective choice of weights, W, takes the form of a diagonal matrix. Equations with larger weights more heavily influence the solution, while those with smaller weights carry less influence. The diagonal matrix of weights is included into the least squares method in the form

$$\mathrm{WV} = \mathrm{WMC}\,. \tag{2.30}$$

Using the pseudo-inverse, the polynomial coefficients of the weighted least squares method are solved for as

$$\mathrm{C} = (\mathrm{WM})^{\dagger}\mathrm{WV}\,. \tag{2.31}$$

Weights corresponding to $(\boldsymbol{x_i} - \bar{\boldsymbol{x}})^{-(Q+1)}$ are reportedly effective [12] due to the fact that they offset the polynomial growth with distance. This leads to an interpolation that prioritizes fitting the data most local to the interpolation center. The k-exact property of the weighted system is the same as that of the non-weighted system, so long as sufficiently many weights are non-zero to maintain an over-determined system.

2.4 Time Marching Methods

In addition to discretization in space, solutions must also be discretized in time. When using a method of lines approach, such as in this dissertation, Equation 2.4 is a semi-discrete system of ordinary differential equations. Numerical methods for ordinary differential equations are well established, and in this work we use the four-stage, fourth-order classical Runge-Kutta scheme. For systems with stiff terms, that arise when solving systems with chemical reaction or in the embedded-boundary method, the ARK4 time marching method [21] is used.

2.5 Model Equations

In this dissertation, the applications of interest are modeling fluid flows in the continuum regime. Specifically two different realms of flow are considered. The first is that of fully-coupled, compressible, thermally-perfect, reacting flow governed by the Navier-Stokes equations. A set of species transport equations is coupled with the Navier-Stokes equations for reaction modeling when combustion is considered for the MMB method. The second is the incompressible single species Stokes flow for the EB method. The remainder of this section provides a review of the important elements for these governing equations.

2.5.1 Reacting Navier-Stokes Equations

The system of PDEs for a compressible gas consisting of the continuity, momentum, energy, and a set of species transport equations is given by

$$\frac{\partial}{\partial t}\rho + \nabla \cdot (\rho \boldsymbol{u}) = 0\,, \tag{2.32}$$

$$\frac{\partial}{\partial t}(\rho \boldsymbol{u}) + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u} + p\mathbb{I}) = \nabla \cdot \vec{\vec{\tau}} + \rho \vec{f}\,, \tag{2.33}$$

$$\frac{\partial}{\partial t}(\rho e) + \nabla \cdot \left(\rho \boldsymbol{u}\left(e + \frac{p}{\rho}\right)\right) = \nabla \cdot \left(\vec{\vec{\tau}} \cdot \boldsymbol{u}\right) - \nabla \cdot \vec{q}\,, \tag{2.34}$$

$$\frac{\partial}{\partial t}(\rho c_n) + \nabla \cdot (\rho c_n \boldsymbol{u}) = -\nabla \cdot \vec{\mathcal{J}}_n + \rho \dot{w}_n\,, \quad n = 1, \ldots, N_g\,, \tag{2.35}$$

where ρ is the density, $\boldsymbol{u}$ is the velocity vector, p is pressure, $\mathbb{I}$ is the identity tensor, and $e = |\boldsymbol{u}|^2/2 + \sum c_{\mathrm{n}} h_{\mathrm{n}} - p/\rho$ is the total specific energy with c_n and h_n being the mass fraction and the specific enthalpy of species n, respectively. A total of N_g species comprise the gaseous mixture, with N_g transport equations. The pressure is given by the ideal gas law $p = \sum \rho c_n R_n T$, where R_n is the universal gas constant of species n and T is the temperature. $\vec{q}$ is the molecular heat flux vector. The molecular fluid stress tensor, $\vec{\vec{\tau}}$, is defined as

$$\vec{\vec{\tau}} = 2\mu\left(\vec{\vec{S}} - \frac{1}{3}\mathbb{I}\nabla \cdot \boldsymbol{u}\right)\,, \tag{2.36}$$

where $\vec{\vec{S}}$ is the strain rate tensor and μ is the fluid molecular viscosity. $\vec{\mathcal{J}}_n$ and $\dot{w}_n$ are the species diffusive flux and the production/destruction rate of species n, respectively. The thermally-perfect reactive mixture of gases neglects Dufour, Soret, and radiative heat transfer effects in the present work. For non-reacting flows, equation Equation 2.35 is omitted.

For reacting flows, the finite-rate chemistry model is used for calculating the reacting source term, which is described in detail in works by Gao et al. [22–26], Owen et al. [27–29], and Wang [30]. In this dissertation, the combustion of two fuels — hydrogen (H_2) and propane (C_3H_8) are considered. The C_3H_8-air combustion is modeled by the chemical mechanism developed by Zettervall et al. [31] ("Z66"), including 25 species and 66 reactions.

2.5.2 Navier-Stokes Equations on Mapped Grids

After applying the mapped grid transformations from Section 2.2.1 to Equations 2.32 – 2.35, the governing equations on a mapped domain can be written into the vector form, given by

$$\frac{\partial}{\partial t}\langle J\mathbf{U}\rangle_{\boldsymbol{i}} + \nabla_{\boldsymbol{\xi}} \cdot \left(\mathrm{N}^{\intercal}\langle\vec{\mathbf{F}}\rangle - \mathrm{N}^{\intercal}\vec{\mathbf{G}}\right) = \langle J\mathbf{S}\rangle_{\boldsymbol{i}}\,, \tag{2.37}$$

where $\vec{\mathbf{F}}$ is the advection (hyperbolic) flux vector, $\vec{\mathbf{G}}$ is the viscous (elliptic) flux vector, and $\mathbf{S}$ is the reaction source vector. They are defined as

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho\boldsymbol{u} \\ \rho e \\ \rho c_n \end{pmatrix}, \quad \vec{\mathbf{F}} = \begin{pmatrix} \rho\boldsymbol{u} \\ \rho\boldsymbol{u}\boldsymbol{u} + p\mathbb{I} \\ \rho\boldsymbol{u}\left(e + \frac{p}{\rho}\right) \\ \rho c_n\boldsymbol{u} \end{pmatrix}, \quad \vec{\mathbf{G}} = \begin{pmatrix} 0 \\ \vec{\vec{\mathcal{T}}} \\ \vec{\vec{\mathcal{T}}} \cdot \boldsymbol{u} - \vec{\mathcal{Q}} \\ -\vec{\mathcal{J}}_n \end{pmatrix}, \quad \mathbf{S} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \rho\dot{w}_n \end{pmatrix}. \tag{2.38}$$

The mapped stress tensor, $\vec{\vec{\mathcal{T}}}$, is given by

$$\vec{\vec{\mathcal{T}}} = 2\mu\left(\vec{\vec{\mathcal{S}}} - \frac{1}{3}J^{-1}\mathbb{I}\nabla_{\boldsymbol{\xi}} \cdot (\mathrm{N}^{\intercal}\boldsymbol{u})\right), \tag{2.39}$$

where the mapped strain rate tensor, $\vec{\vec{\mathcal{S}}}$, is given by

$$\vec{\vec{\mathcal{S}}} = \frac{1}{2}\left((\nabla_{\boldsymbol{\xi}}\boldsymbol{u})\left(\frac{\mathrm{N}^{\intercal}}{J}\right) + \left(\frac{\mathrm{N}}{J}\right)^{\intercal}(\nabla_{\boldsymbol{\xi}}\boldsymbol{u})^{\intercal}\right). \tag{2.40}$$

The mapped molecular heat flux, $\vec{\mathcal{Q}}$, is modeled by

$$\vec{\mathcal{Q}} = -\left(\kappa\frac{\mathrm{N}}{J}\nabla_{\boldsymbol{\xi}}T - \sum_{n=1}^{N_g}\left(h_n\vec{\mathcal{J}}_n\right)\right), \tag{2.41}$$

where κ is the thermal conductivity coefficient and $\vec{\mathcal{J}}_n$ is the mapped mass diffusion of species n modeled by the Fourier's law, given by

$$\vec{\mathcal{J}}_n = -\rho D_n \frac{\mathrm{N}}{J} \nabla_{\boldsymbol{\xi}} \boldsymbol{u} \,. \tag{2.42}$$

In the equation, D_n is defined as the molecular diffusivity for species n.

2.5.3 Stokes Equation

For flows that are dominated by diffusive physics, where the Reynolds number Re $\ll 1$, a subset of the Navier-Stokes equations can be used that assumes the flow is incompressible, of constant density, and inertial effects can be neglected. The resulting unsteady Stokes equation is given by a single momentum equation

$$\frac{\partial}{\partial t} \boldsymbol{u} = -\nabla p + \nu \Delta \boldsymbol{u} \,, \tag{2.43}$$

$$\nabla \cdot \boldsymbol{u} = 0 \,, \tag{2.44}$$

where $\boldsymbol{u}$ is the flow velocity, p the pressure, and ν the kinematic viscosity, and Δ is the Laplacian operator. Boundary conditions for inflows prescribe a velocity $\boldsymbol{u} = \boldsymbol{u}_{\text{in}}$, while outflows are specified by $\nabla \boldsymbol{u} \cdot \hat{\mathbf{n}} = 0$. At walls, viscous boundary conditions prescribe a boundary velocity $\boldsymbol{u} = \boldsymbol{u}_{\text{wall}}$. In this work, solid walls are only permitted tangential velocities such that $\boldsymbol{u}_{\text{wall}} \cdot \hat{\mathbf{n}} = 0$

2.5.4 Finite Volume Projected Stokes Equation

A particular challenge for the Stokes equations is that solution methods must satisfy a divergence-free constraint. A Hodge projection operator $\mathbb{P}$ has been used in the finite volume litera-

ture [32, 33] to enforce a divergence-free velocity field. This projection operator is defined by

$$\mathbb{P}(\boldsymbol{w}) = \boldsymbol{v}\,, \tag{2.45}$$

$$\nabla \cdot \boldsymbol{v} = 0\,, \tag{2.46}$$

where ,

$$\mathbb{P}(\boldsymbol{w}) \equiv \left(\mathbb{I} - \nabla\Delta^{-1}\nabla\cdot\right)\boldsymbol{w}\,. \tag{2.47}$$

where $\boldsymbol{w}$ is an arbitrary velocity field, and the resulting $\boldsymbol{v}$ is the divergence free component of $\boldsymbol{w}$.

We choose discretizations for each of the spatial operators in Eqs. (2.43-2.44), and write the resulting discrete equations at grid locations $\boldsymbol{i}$ as

$$\frac{\partial}{\partial t}\boldsymbol{u}_{\boldsymbol{i}} = -(\mathbf{G}p)_{\boldsymbol{i}} + \nu(\mathbf{L}\boldsymbol{u})_{\boldsymbol{i}}\,, \tag{2.48}$$

$$(\mathbf{D}\boldsymbol{u})_{\boldsymbol{i}} = 0\,, \tag{2.49}$$

where $\mathbf{D}, \mathbf{G}$, and $\mathbf{L}$ are fourth-order finite volume approximations of divergence, gradient, and Laplacian terms, respectively. (From this point forward, we will drop the subscript $\boldsymbol{i}$ except for clarity.) The goal is to discretize these operators so that

$$\mathbf{D}\boldsymbol{u} = \nabla \cdot \boldsymbol{u} + O(h^4)\,, \tag{2.50}$$

$$\mathbf{G}\boldsymbol{u} = \nabla\boldsymbol{u} + O(h^4)\,, \tag{2.51}$$

$$\mathbf{L}\boldsymbol{u} = \nabla \cdot \nabla\boldsymbol{u} + O(h^4) \tag{2.52}$$

in the regular interior of the domain, with some potential loss of accuracy near boundaries and in cut cells.

We use co-located cell-average velocity and pressure, and so we take the approach of an *approximate* projection [34]. Instead of a strictly zero discrete divergence, we allow $\boldsymbol{u}$ to have a

divergence that is at the level of the discretization error. The equivalent discrete projection is

$$\mathbb{P}(\boldsymbol{w}) = \left(\mathbb{I} - \mathbf{G}\mathbf{L}^{-1}\mathbf{D}\right)\boldsymbol{w}\,, \tag{2.53}$$

which requires the inversion of the Laplacian operator over the whole domain. Using the projection operator, the incompressible flow equations can be approximated by

$$\frac{\partial}{\partial t}\boldsymbol{u} = \mathbb{P}\left(\nu\mathbf{L}\boldsymbol{u}\right)\,, \tag{2.54}$$

$$\mathbf{D}\boldsymbol{u} = O(h^4)\,. \tag{2.55}$$

The general procedure for solving this system separates into an intermediate update that evolves the viscous terms in time, and then projects that update using Equation 2.53, to maintain an approximately-divergence free solution for each time step. This is essentially a higher-order accurate version of the projection operator described by Trebotich et al. [35]

Projection Formulation for Open Boundaries

Boundary conditions for the projection operator must be modified for open domain boundaries, so that a given velocity, such as that resulting from the viscous terms, has three separate components: $\boldsymbol{w} = \mathbf{G}\psi + \boldsymbol{v} + \mathbf{G}\phi$. Here, ψ is the scalar potential flow solution which satisfies only the boundary conditions, and two parts with homogeneous boundary conditions: a pure gradient, ϕ, and $\boldsymbol{v}$, the divergence-free part. Each of these components must be considered in the projection:

$$\mathbf{L}\psi = \mathbf{D}\mathbf{G}\psi = 0, \quad \mathbf{G}\psi\cdot\hat{\mathbf{n}} = \boldsymbol{u}\cdot\hat{\mathbf{n}} \qquad \text{(potential flow with BCs)}, \tag{2.56}$$

$$\mathbf{D}\boldsymbol{v} = 0, \quad \boldsymbol{v}\cdot\hat{\mathbf{n}} = 0 \qquad \text{(divergence-free)}, \tag{2.57}$$

$$\mathbf{L}\phi = \mathbf{D}\mathbf{G}\phi = \mathbf{D}\boldsymbol{w}, \quad \mathbf{G}\phi\cdot\hat{\mathbf{n}} = 0 \qquad \text{(interior gradient)}. \tag{2.58}$$

In the end, our desired divergence-free velocity field, $\boldsymbol{u}$, that satisfies the correct boundary conditions is just the components $\boldsymbol{u} = \boldsymbol{v} + \mathbf{G}\psi = \boldsymbol{w} - \mathbf{G}\phi$.

When solving for the pure gradient component, ϕ, boundary conditions for inflow and walls are defined by $\nabla\phi \cdot \hat{\mathbf{n}} = 0$ while outflow boundaries prescribe $\phi = 0$. For the divergence operator on the right-hand side of Equation 2.58, velocity boundary conditions for $\boldsymbol{w}$ match those of the Stokes equations for inflow, $\boldsymbol{w} = \boldsymbol{u}_{\text{in}}$. Wall boundary conditions specify no normal-flow, $\boldsymbol{w} \cdot \hat{\mathbf{n}} = 0$, and at outflow boundaries the divergence is calculated with no boundary condition.

2.6 Computational Framework

The numerical frameworks utilized in this dissertation are Chord [8, 13, 26, 27, 29, 36–38] and the Chombo library for parallel AMR [39,40]. Chombo provides a set of tools for solving PDEs on block-structured adaptively refined grids on parallel platforms. Chord is a FVM that is fourth-order accurate in space and time for smooth flows, and solves the compressible Navier-Stokes equations with and without chemical reaction. It features AMR in space and time, and has been shown to scale to at least $1 \cdot 10^5$ cores with flat MPI. For non-smooth flow features (e.g. shocks or detonation waves), Chord uses the PPM [4, 19] limiter for stability. Mapped multi-block grids are used for the construction of complex geometry. Solution adaptivity is achieved by refining grids based on physics of interest, such as solution discontinuities, regions of high vorticity, or combustion flame fronts. Chord is particularly powerful in solving high-speed flows with turbulence, chemical reactions, and shocks present. Reynolds-averaged Navier-Stokes (RANS) and state-of-the-art large eddy simulation (LES) turbulence models are available in Chord [41–43]. Additionally, a range of fuels and their chemical kinetics are available for combustion, including H_2, CH_4, C_3H_8, and NH_3.

Embedded-boundary methods are also supported by Chombo [40], while still allowing for AMR and a high degree of parallelism. Existing work for lower-order methods solving the Navier-Stokes equations with this framework have been shown [35, 44]. Effort towards extending this framework to high-order has been made, with work towards geometry representations [45] and solvers for Poisson's equation from Devendran et al. [12].

Chapter 3

Adaptive Clipping-and-Redistribution for Bounded High-Order Interpolation

In this chapter, we devise a new high-order bound-preserving interpolation scheme, which improves the robustness of AMR for high-order FVMs when modeling combusting flows. The motivations for developing this method are first explained, and existing methods that have similar objectives discussed. Next, details of the newly developed HO-ACR algorithm are presented, including the solution procedures and simple verification tests. The new method is implemented in Chord, and the HO-ACR algorithm is verified by advection physics. Finally, results and discussions are provided, which demonstrate the effectiveness of the method for complex flows.

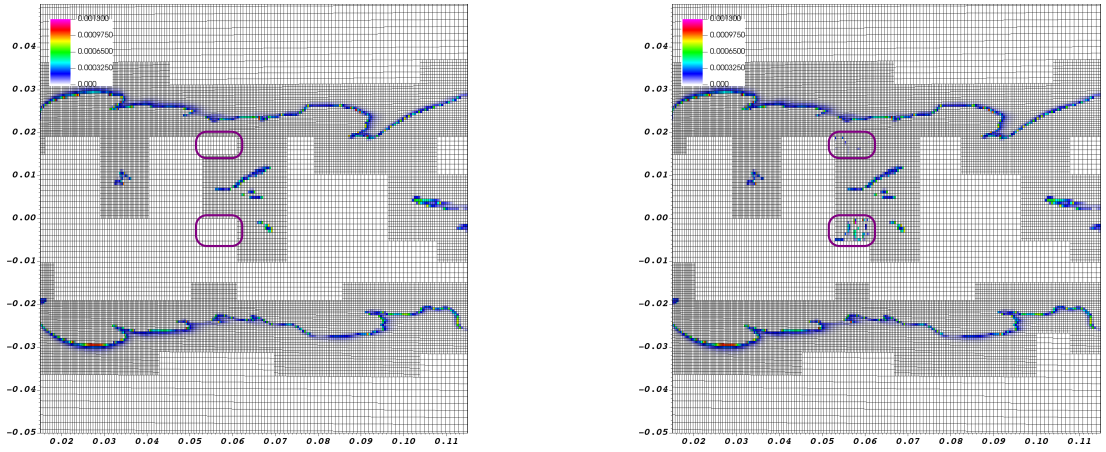
3.1 Motivations

The purpose of AMR (Section 2.2.4) is to better resolve regions of interest in a solution in order to reduce the discretization error. AMR requires interpolation in *both space and time* at the boundaries of grid refinements, to enable local time stepping at each resolution. When the underlying finite-volume method is high-order accurate for smooth flows, as in the present study, oscillations can appear near extrema or discontinuities due to the use of a *high-order conservative interpolation scheme*, as described in Section 2.3.4. These overshoots are particularly troublesome when they become unphysical, and ideally AMR would *tag* (mark locations for refinement) discontinuous features before they form. When solving compressible reactive flows, shock waves and flame fronts are features of interest. A high-order interpolant near or crossing these features often brings about oscillations, resulting in undesirable characteristics such as negative mass fractions, or such that the sum of all species' mass fractions are greater than unity in some localized region. Eventually, these unphysical phenomena can destroy the solution.

Two types of interpolation are required for the finite-volume AMR strategy discussed herein. Interpolation is used to fill ghost cells surrounding a patch of fine grid cells, and interpolation is used to define new fine grids as the solution evolves. For both cases, standard practice is to adjust the fine grids so that discontinuities always remain well away from refinement boundaries, and the interpolations are of smooth solution profiles. Having a discontinuous feature encroach on a grid refinement boundary is avoided. However, for reasons of computational cost, this is not always practical. In many simulation strategies, regions are refined based on criteria other than following a discontinuity. For example, if reproducing an experiment, one may add fixed levels of refinement that match a physical observation window to focus computational effort, and minimize error, in that window. Strategies such as embedded-LES and embedded-DNS (to be discussed shortly) have similar requirements. For some problems, such as combustion described by highly sensitive reaction mechanisms in Section 2.5.1, tiny overshoots and undershoots in bounded species mass fractions may be enough to corrupt the solution, in regions well away from the identified flame discontinuity. Although it is not the preferred way to apply AMR, there are strong motivations for robustly and accurately interpolating in the vicinity of discontinuities. The goal of the work in this chapter is to create high-order accurate AMR methods that are more robust to tagging and refinement variability, especially when AMR interfaces are near regions with strong gradients.

To demonstrate the problem specifically, an example of high-order interpolation is shown in Figure 3.1 where significant erroneous features appear due to mesh refinement in patches highlighted by violet bubbles. The figure shows the distribution of CH_2O species mass fraction from combustion of a C_3H_8-air mixture. Further details for this case are explained in Section 3.4.2, but here we mainly point out the oscillations produced by the high-order interpolation. These seemingly small noises, resulting from mesh refinement, have been shown to grow and interact with the flow dynamics, eventually leading to numerical instability. This is observed to be especially problematic for chemically reacting flows where small differences in mass factions are able to push the chemical reaction to unphysical states that lead to solution failure due to the stiff nature of the reactions. Therefore, for combustion with stiff chemistry, this necessitates a high-order interpola-

tion approach that is not only bound-preserving but also maintains conservation while providing a high-order accurate solution in smooth regions. By contrast, for calorically perfect gases, even at high Mach numbers, it is the authors' experience that bounded interpolation is not necessary as long as discontinuities are contained in fine grids.



(a) Before AMR regrid

(b) After AMR regrid

Figure 3.1: Flame front in the bluff-body combustion case, shown by mass fractions of species CH_2O. The grid is adaptively refined to capture the flame front, and regridded between the left and right images. The addition of new fine grid cells introduces overshoots which significantly disrupt the flow. The regions with overshoot are indicated by violet bubbles.

Another example is controlling oscillations that may appear from high-order interpolation when studying turbulent flow applications utilizing embedded-large-eddy-simulation (embedded-LES) and embedded-direct-numerical-simulation (embedded-DNS) methods [46]. In these approaches, only a small subset of the domain has sufficient resolution for a detailed query of the physics of interest using LES or DNS modeling of turbulence. Elsewhere, the domain can be considered as approximating the influence of realistic geometries and boundary conditions using far more affordable methods. With additional control of error, e.g., via data assimilation, it is a way forward when full LES or DNS is otherwise completely infeasible due to the computational expense. Since embedded-DNS and embedded-LES cases can only afford limited amounts of mesh refinement, discontinuous flow features often must span AMR interfaces. This is illustrated by Figure 3.2, where only a small window of a flame front is resolved using DNS. In this example,

the discontinuous flame front must cross multiple different levels of grid refinement. To reduce computational time, there is also interest in allowing the flame to only develop on the coarse level for an amount of time before refining and using it as an initial condition for the embedded-DNS region. For these applications, it is important to have an algorithm which is able to mitigate the oscillations present in high-order interpolation to prevent unwanted numerical instabilities due to abrupt AMR-driven changes in grid resolution.

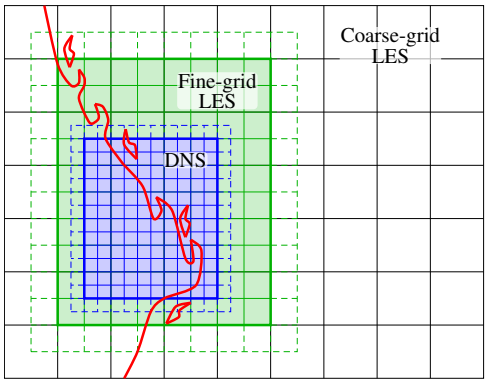


Figure 3.2: Illustration of embedded-DNS. Due to computational limitations, only a small window of DNS spanning a flame front is affordable. In order to develop the large scale flow features of a flame, multiple levels of grid refinement must be used. This results in discontinuous features spanning different refinement levels, which can destabilize the solution.

3.1.1 Existing Methods to Cope with Unphysical Numerical Solutions

Common methods for bounds preservation in CFD schemes exist in the form of limiters. The PPM limiter [19] is well known for its application in high-order schemes, and relative simplicity. This method operates by constructing high-order approximations at cell faces, while ensuring these constructions do not generate new extrema and are monotonic. Furthermore, solution smoothness is accounted for in order to prevent excessive limiting and reduction in the order of accuracy in regions of smooth extrema. Although well-developed, in general limiters are only designed for face reconstructions and most often applied in one dimension at a time. Approaches for limiters in multiple dimensions do exist [47], but are not easily modified to achieve strict bound-preserving cell-based interpolation.

In a similar spirit to limiters, the flux corrected transport (FCT) methods utilize low-order solutions to provide bounds for high-ordered flux reconstructions. Unlike limiters, FCT methods have been extended to operate in multiple dimensions. The FCT-like maximum preserving principle (MMP) in the work by Anderson et al. [48] enforces the flux correction by instead adjusting Discontinuous Galerkin elements. This adjustment requires satisfying flux constructions, while redistributing and preserving element mass. Despite the approach having many valuable properties, the aims are subtly different from ours and can not be directly translated into a FVM scheme.

One approach that is multi-dimensional, conservative, and solves the spatial interpolation problem we face is the CENO method described by Ivan et al. [49]. This approach formulates a smoothness measure to identify regions which are non-smooth. In non-smooth regions a low-order interpolation is used, while a high-order interpolation is employed elsewhere. Although it is an effective approach, it requires a tuned coefficient for smoothness detection. Additionally, large stencils of data can be required which necessitate expensive data communication, but further developments of this approach are able to reduce stencil sizes by more frequently moving smaller amounts of information [50]. Implementing this interpolation approach other than within the CENO scheme would be non-trivial since it requires careful consideration of limiter choice and significant changes to communication patterns.

Another approach for the multi-dimensional, conservative interpolation problem is the clipping-and-redistribution algorithm described by Hilditch et al. [51]. After a high-order interpolation, any solution outside physically imposed bounds is clipped off. The sum of clipped quantities is tracked and redistributed elsewhere inside the domain to maintain conservation. This approach is developed to prevent physically constrained overshoots based upon predetermined bounds. Additionally, the redistribution happens globally and thus requires solving an implicit system which is potentially expensive. Nevertheless, this dissertation is the inspiration for the algorithm that we present. We seek to adaptively enforce bounds near discontinuities and reduce the cost of redistribution. In addition, the present algorithm operates on mapped quantities since the underlying CFD infrastructure uses mapped structured grids to accommodate complex geometries.

None of the above approaches fully match our required criteria for refining complex combusting flows. Although the interpolation method in the CENO scheme is the nearest to our goals, it is not straightforward to incorporate into our existing code. In the present work, we develop the high-order adaptive clipping-and-redistribution (HO-ACR) method and successfully demonstrate the design properties of the interpolation: conservative, solution-dependent bounds-preservation, efficiency, interpolation without compromising accuracy in smooth regions. Additionally, the HO-ACR algorithm is designed to operate as a correction to a high-order interpolation. This makes it non-invasive and relatively simple to implement independent of the scheme it is used in.

3.2 Technical Approach

The adaptive clipping-and-redistribution, or ACR, method is designed as a post-processing operation independent of the high-order interpolation. This simplifies implementation since it requires no fundamental changes to the existing high-order interpolation algorithm, and can be adaptively enabled. The method maintains conservation, avoids significant data exchange, and does not require predetermined bounds. The procedure for ACR is outlined by the following steps:

1. Interpolate the fine data from the coarse data using a high-order conservative scheme as described in Section 2.2.4.

2. Establish minimum and maximum bounds for interpolation values of each fine cell. These bounds are created using local coarse cell information.

3. Clip each fine cell to fall within the established bounds and track the solution quantity change.

4. Redistribute the clipped excess to local regions that can accept it while remaining inside their bounds.

A few main features deserve emphasis and clarification. The primary objective of the ACR method is to prevent new extrema while maintaining solution conservation. Notably, this does not

strictly enforce monotonicity of the solution. It is important to properly establish the local clipping bounds since their selection directly impacts the resulting solution. Furthermore, the extension of the ACR to high-order is desirable for maximum accuracy. At the end of the section, we provide verification of the new method.

3.2.1 Establishing Local Clipping Bounds

Local clipping bounds provide a feasible range that the interpolation of each fine cell must lie within to prevent generation of new extrema. For each fine cell $\boldsymbol{i}$ on AMR level $\ell + 1$, an interval $\beta_{\boldsymbol{i}^{\ell+1}}$ containing the valid interpolated solution is determined. Bounds are created from the minimum and maximum values of a neighborhood $\mathcal{I}(\mathcal{C}(\boldsymbol{i}^{\ell+1}))$ of coarse data on level ℓ about the fine cell $\boldsymbol{i}^{\ell+1}$. In this work, the neighborhood $\mathcal{I}(\mathcal{C}(\boldsymbol{i}^{\ell+1}))$ is chosen to be the same as the neighborhood from the interpolation stencils. However, it need not be identical. Clipping bounds are determined as

$$\beta_{\boldsymbol{i}^{\ell+1}} = \left[\min\left(\langle \mathbf{U} \rangle_{\boldsymbol{j}^{\ell}}, \boldsymbol{j}^{\ell} \in \mathcal{I}(\mathcal{C}(\boldsymbol{i}^{\ell+1})) \right), \max\left(\langle \mathbf{U} \rangle_{\boldsymbol{j}^{\ell}}, \boldsymbol{j}^{\ell} \in \mathcal{I}(\mathcal{C}(\boldsymbol{i}^{\ell+1})) \right) \right]. \tag{3.1}$$

The ACR method will ensure the interpolated fine level respects the bounds from the existing coarse level. As a result, the ACR method does not require explicit bounds to maintain physically realizable solutions, so long as the coarse solution itself is physically realizable.

The only requirement on local neighborhood choice $\mathcal{I}(\mathcal{C}(\boldsymbol{i}^{\ell+1}))$ is that it must contain the coarse cell $\mathcal{C}(\boldsymbol{i}^{\ell+1})$; however other choices do impact the behavior of the ACR method. For example, the smallest clipping neighborhood may be set to include only the coarse cell $\mathcal{C}(\boldsymbol{i}^{\ell+1})$. This removes any flexibility in the interpolation, and reverts to a piecewise constant interpolation. In the other extreme, the clipping neighborhood is expanded to the entire domain, and the ACR method will only restrict the creation of new global extrema. Our compromise is that clipping stencils should have the same locality as interpolation stencils; smaller neighborhoods are overly restrictive, while larger neighborhoods are not restrictive enough and are computationally impractical.

3.2.2 High-Order Extension

In regions where data is smooth and in state variables with no strict physical bounds (such as velocity), high-order interpolation is expected to perform well without assistance from the ACR method. Reducing the interpolation order in these smooth regions can compromise physics that are correctly captured by the high-order scheme [19]. In such cases, interpolation of smooth data is expected to accurately generate subtle new extrema. This is illustrated in Figure 3.3 where high-order interpolation correctly captures a Gaussian profile, and enforcing bounds via the ACR method results in a poor quality result. For this purpose the ACR method is extended to high-order, hereafter called the HO-ACR method. The HO-ACR method is adaptive in that the operation is not engaged in smooth regions, while being engaged elsewhere. Regions are determined as smooth if all second derivative constructions share the same sign. For each direction of the computational domain, the second derivatives are computed for each coarse cell $\boldsymbol{i}^{\ell}$, and compared to each of the directly adjacent neighbors. If the entire set share the same sign, the region is determined as smooth and the HO-ACR for fine cells $\boldsymbol{i}^{\ell+1} \in \mathcal{C}^{-1}(\boldsymbol{i}^{\ell})$ is disabled. It is at the discretion of the researcher whether high-order detection is used for any of the state variables. HO-ACR is used for all results presented herein except for the bluff-body combustor in Section 3.4.2.

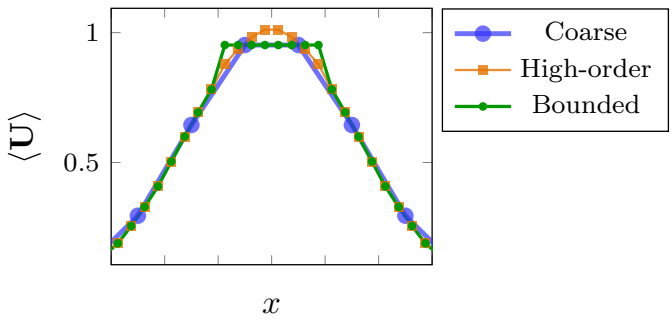


Figure 3.3: An example of the high-order interpolation correctly creating new extrema of a smooth Gaussian profile.

Computing the curvature using a 1^{st}-order accurate method is sufficient. Specifically, curvature is evaluated using a 2^{nd}-order centered scheme as given by

$$\frac{\partial^2 \mathbf{U}_{\boldsymbol{i}}}{\partial \xi_d^2} = \frac{\langle \mathbf{U} \rangle_{\boldsymbol{i}-\boldsymbol{e}^d} - 2\langle \mathbf{U} \rangle_{\boldsymbol{i}} + \langle \mathbf{U} \rangle_{\boldsymbol{i}+\boldsymbol{e}^d}}{h^2} + O(h^2)\,, \tag{3.2}$$

where h is the cell spacing. For locations near the physical boundaries, 1^{st}-order one-sided formulations such as

$$\frac{\partial^2 \mathbf{U}_{\boldsymbol{i}}}{\partial \xi_d^2} = \frac{\langle \mathbf{U} \rangle_{\boldsymbol{i}} - 2\langle \mathbf{U} \rangle_{\boldsymbol{i}+\boldsymbol{e}^d} + \langle \mathbf{U} \rangle_{\boldsymbol{i}+2\boldsymbol{e}^d}}{h^2} + O(h)\,, \tag{3.3}$$

are used as required.

3.2.3 Local Linear Redistribution

Once clipping is completed, properly redistributing the clipped quantities is necessary to maintain conservation. One of our design goals for the redistribution scheme is to be computationally inexpensive by avoiding data exchange. A redistribution region is chosen such that the redistribution can be solved explicitly without additional data exchange by choosing non-overlapping redistribution regions $\mathcal{R}$. The natural choice for such regions is the coarse cells where data is being interpolated from, such that $\mathcal{R} = \mathcal{C}(\boldsymbol{i}^{\ell+1})$.

Many CFD algorithms use a mapped grid technique to accommodate moderately complex geometries, meaning that a general curvilinear coordinate transformation is used to map the physical domain to the computational domain. Operations on the mapped quantities $\langle J\mathbf{U} \rangle$ require additional care to enforce bounds on the physical solution $\langle \mathbf{U} \rangle$. Bounds can only be enforced on conservative quantities $\langle \mathbf{U} \rangle$ because gradients in $\langle J\mathbf{U} \rangle$ may appear solely as the result of grid mapping. However, regridding requires conservation of the mapped variables as in Equation 2.26. The mapped HO-ACR algorithm must first solve for a 2^{nd}-order conservative variable

$$\overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} = \frac{\langle J\mathbf{U} \rangle_{\boldsymbol{i}}}{\langle J \rangle_{\boldsymbol{i}}} + O(h^2)\,. \tag{3.4}$$

On Cartesian grids, the same HO-ACR algorithm can be applied directly to conservative quantities $\langle \mathbf{U} \rangle$ by setting J to one, and Equation 3.4 then becomes exact.

The method for localized linear redistribution begins by determining the overshoot from each fine cell $\delta\langle \mathbf{U} \rangle_{\boldsymbol{i}}$ with

$$\delta\overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} = \begin{cases} \overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} - \beta_{\boldsymbol{i}}^{\max} & \overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} > \beta_{\boldsymbol{i}}^{\max} \\ \overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} - \beta_{\boldsymbol{i}}^{\min} & \overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} < \beta_{\boldsymbol{i}}^{\min} \\ 0 & \text{otherwise} \end{cases}, \tag{3.5}$$

where $\beta_{\boldsymbol{i}}^{\max}$ and $\beta_{\boldsymbol{i}}^{\min}$ are respective maximum and minimum of the previously determined clipping bounds $\beta_{\boldsymbol{i}}$ (e.g., an interval). The mapped quantities that are clipped off are computed by

$$\delta\overline{\langle J\mathbf{U} \rangle}_{\boldsymbol{i}} = \delta\overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} \langle J \rangle_{\boldsymbol{i}} + O(h^2)\,, \tag{3.6}$$

which again is a 2^{nd}-order operation. Using the clipping bounds, the extra mass to be redistributed in region $\mathcal{R}$ is given by

$$E_{\mathcal{R}} = \sum_{\boldsymbol{j} \in \mathcal{R}} \delta\overline{\langle J\mathbf{U} \rangle}_{\boldsymbol{j}}\,. \tag{3.7}$$

To determine where mass may be pushed to, each fine cell has a maximum allowable difference calculated as

$$\alpha_{\boldsymbol{i}} = \begin{cases} \beta_{\boldsymbol{i}}^{\max} - \overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} & E_{\mathcal{R}} > 0 \\ \overline{\langle \mathbf{U} \rangle}_{\boldsymbol{i}} - \beta_{\boldsymbol{i}}^{\min} & E_{\mathcal{R}} < 0 \\ 0 & \text{otherwise} \end{cases}. \tag{3.8}$$

Using this, each fine cell can be updated using the clipping-and-redistribution as

$$\widetilde{\langle J\mathbf{U} \rangle}_{\boldsymbol{i}} = \langle J\mathbf{U} \rangle_{\boldsymbol{i}} - \delta\overline{\langle J\mathbf{U} \rangle}_{\boldsymbol{i}} + \frac{\alpha_{\boldsymbol{i}} E_{\mathcal{R}}}{\sum_{\boldsymbol{j} \in \mathcal{R}} \alpha_{\boldsymbol{j}}}\,. \tag{3.9}$$

In effect, this drops the order of accuracy of the interpolation to 1^{st}-order in regions which are fully clipped. This mapping process only needs to be 2^{nd}-order, since the HO-ACR method effectively

reduces the interpolation order in non-smooth regions where clipping is engaged. Note that the smoothness check for the high-order extension must operate on $\langle \mathbf{U} \rangle$.

Conservation Proof

The ACR method can be shown to maintain conservation, by demonstrating

$$\sum_{i \in \mathcal{R}} \widetilde{\langle J\mathbf{U} \rangle}_i = \sum_{i \in \mathcal{R}} \langle J\mathbf{U} \rangle_i \,. \tag{3.10}$$

Assuming the prior equation holds, and substituting it into Equation 3.9, conservation can be shown by

$$\begin{aligned} \sum_{i \in \mathcal{R}} \delta \overline{\langle J\mathbf{U} \rangle}_i &= \sum_{i \in \mathcal{R}} \frac{\alpha_i E_{\mathcal{R}}}{\sum_{j \in \mathcal{R}} \alpha_j} \\ &= E_{\mathcal{R}} \,, \end{aligned}$$

which is true by the definition of Equation 3.7, and therefore

$$\sum_{i \in \mathcal{R}} \widetilde{\langle J\mathbf{U} \rangle}_i = \sum_{i \in \mathcal{R}} \langle J\mathbf{U} \rangle_i$$

so long as

$$\sum_{i \in \mathcal{R}} \beta_i^{\max} \geq \sum_{i \in \mathcal{R}} \overline{\langle \mathbf{U} \rangle}_i \geq \sum_{i \in \mathcal{R}} \beta_i^{\min} \,.$$

In general $\sum_{i \in \mathcal{R}} \widetilde{\langle \mathbf{U} \rangle}_i \neq \sum_{i \in \mathcal{R}} \langle \mathbf{U} \rangle_i$ except for the case of Cartesian grids in physical space when J is one.

1D Verification

The property of bound-preservation of the new method is verified using a range of 1D profiles from smooth to non-smooth, including 6 tests — (a) a step function, (b) a smooth step function, (c) a piecewise linear function, (d) a Gaussian profile, (e) a piecewise cubic function, and (f) a discrete delta function. Figure 3.4 shows the spatial profiles of solution variable $\langle \mathbf{U} \rangle$ and compares

the performance of the HO-ACR method for the 6 cases on a Cartesian grid (by setting $J = 1$ uniformly).

The conservative interpolation is 4^{th}-order accurate using a centered 5-point centered stencil. For all cases, the HO-ACR method successfully clips off the overshoots (e.g., cases a, b, c, e, f), while preserving the smooth extrema (e.g., case d). Additionally, all 1D cases maintain conservation to machine precision.

Despite bounding the interpolation near sharp features, there is still room for improvement in the solution. In cases (c) and (e), the high-order interpolation at one coarse-cell away from the discontinuities does not correctly capture the coarse polynomial solution. This is due to the interpolation stencil spanning the discontinuities, and as a result oscillations are present in the high-order conservative interpolation. While the HO-ACR method is not able to correct the high-frequency oscillations, it still does no worse than the existing high-order interpolation. It is important to note, the objective of the HO-ACR method is to prevent new extrema near discontinuous features and the method does not enforce monotonic interpolations of non-smooth coarse data. The emphasis of the HO-ACR method is to maintain solutions that are physically bounded, but leave the quality of these solutions up to the base interpolation scheme.

Using the same 6 cases as shown in Figure 3.4, we demonstrate the mapped HO-ACR performance on mapped grids. The setting of Figure 3.5 and Figure 3.6 is the same as that of Figure 3.4 — the sequence of cases (a-f), the line color and symbols, and the interpolation scheme. Cases in Figure 3.5 use a mapping function of $x = a\tan(b\xi)$ while in Figure 3.6 the mapping function is $x = \frac{1}{2}\xi^2$. The mapped HO-ACR performs exceptionally well for both cases. The clipped profiles are bound-preserving and smooth extrema-preserving, while maintaining conservation to machine precision. As in the prior example, the cases (c) and (e) do not correctly match the coarse polynomial solution due to the high-order interpolation. Although there is room for improvement in regions away from the discontinuity, the solution is still improved as a whole over the unbounded high-order interpolation. Notably these small regions of error do not violate any existing coarse bounds of the solution.

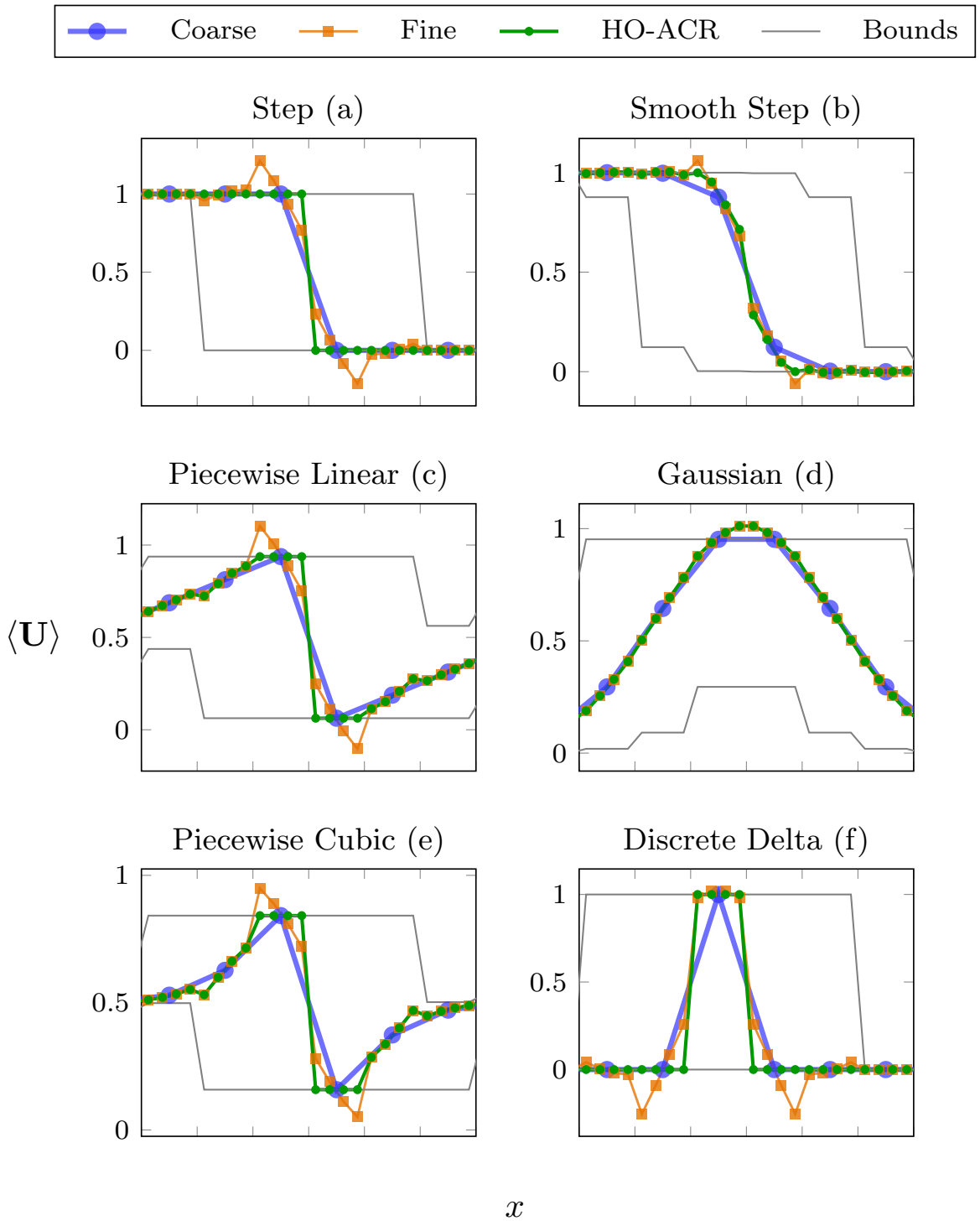


Figure 3.4: 1D interpolation showing the HO-ACR method on Cartesian grids. The interpolation is 4^{th}-order and conservative using a 5-point centered stencil. The x axis shows the coarse cells, and y axis the 4^{th}-order cell-averaged values of the solution. Blue lines indicate coarse solution values, orange lines the coarse to fine interpolation, and green lines the interpolation with addition of the HO-ACR method. Gray lines show the bounding envelope used by the HO-ACR method for non-smooth regions.

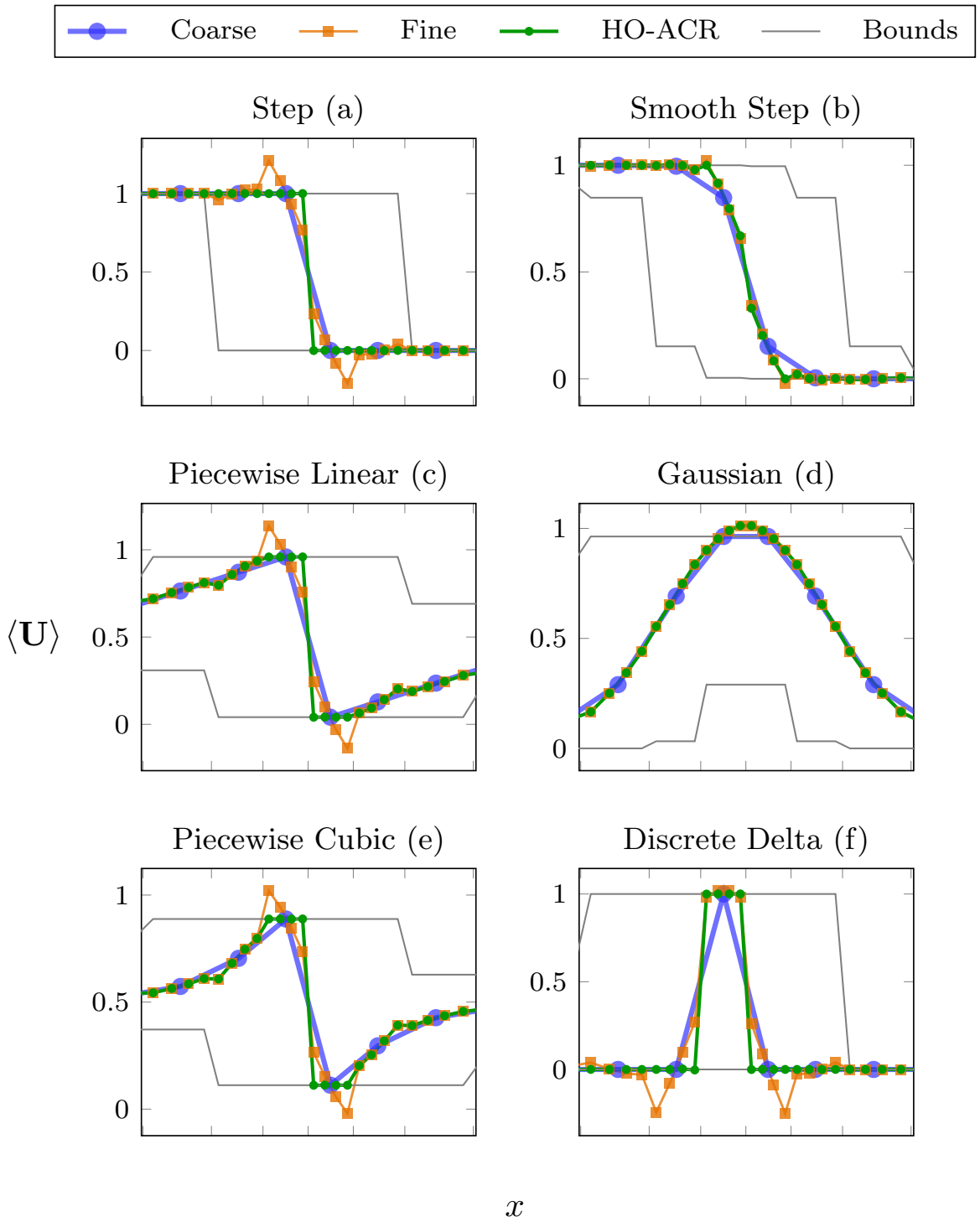


Figure 3.5: 1D interpolation using the HO-ACR method on a mapped grid defined by $x = a\tan(b\xi)$, where the scaling factors are $a = (2\cot(\frac{\pi}{8}))^{-1}$ and $b = \frac{3\pi}{4}$ for the domain centered about zero. The interpolation is 4^{th}-order and conservative using a 5-point centered stencil. The x axis shows the coarse cells, and y axis the 4^{th}-order cell-averaged values of the solution. The blue lines shows the values of coarse solutions, the orange lines the coarse to fine interpolation, and green lines the interpolation with addition of the HO-ACR method. Gray lines show the bounding envelope used by the HO-ACR method for non-smooth regions.

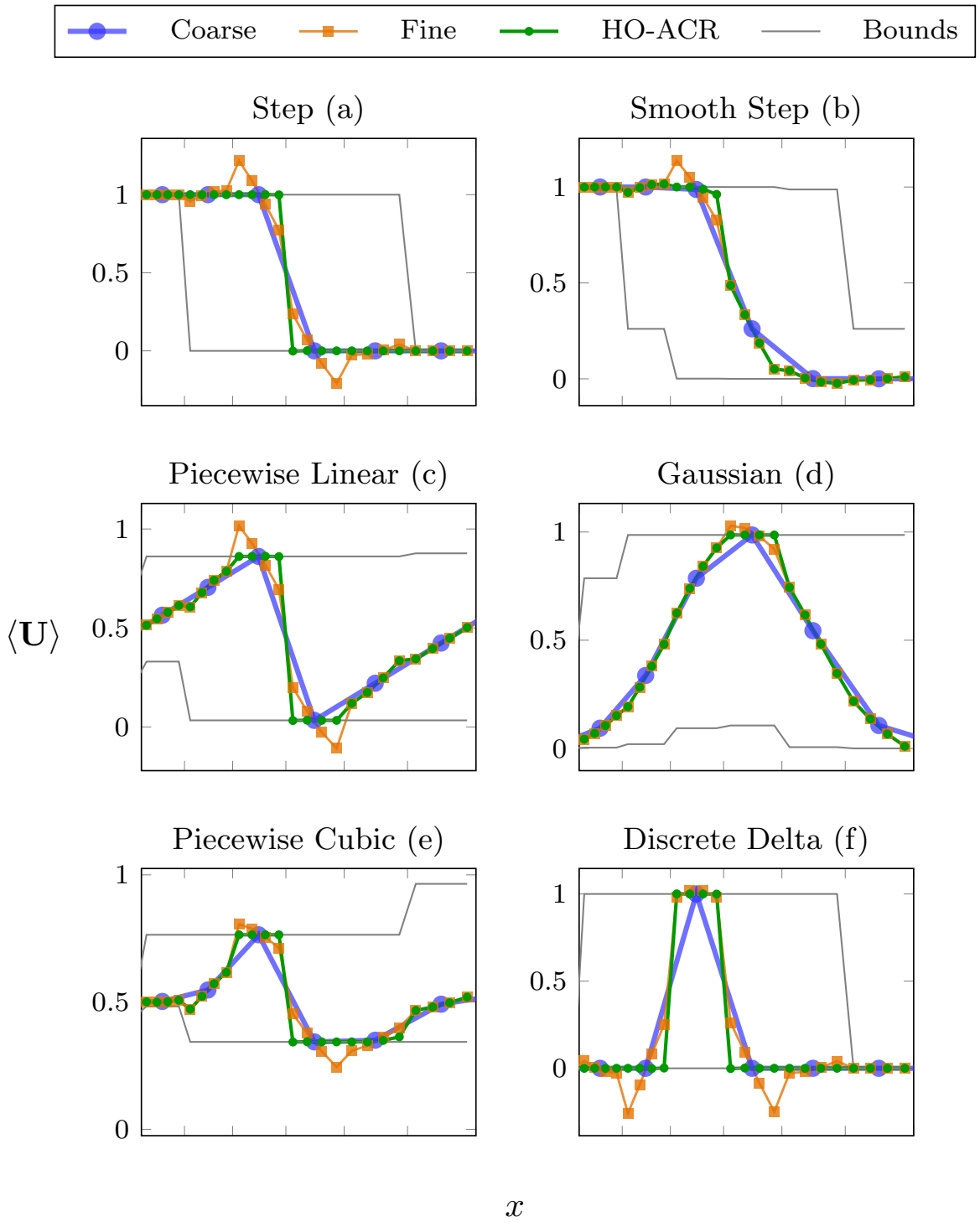


Figure 3.6: 1D interpolation using the HO-ACR method on a mapped grid defined by $x = \frac{1}{2}\xi^2$ for positive ξ. The interpolation is 4^{th}-order and conservative using a 5-point centered stencil. The x axis shows the coarse cells, and y axis the 4^{th}-order cell-averaged values of the solution. The blue lines shows the values of coarse solutions, the orange lines the coarse to fine interpolation, and green lines the interpolation with addition of the HO-ACR method. Gray lines show the bounding envelope used by the HO-ACR method for non-smooth regions.

Algorithm 1 presents the mapped HO-ACR for $\langle J\mathbf{U}\rangle$ of AMR level $\ell+1$. The same algorithm can be applied directly to conservative quantities $\langle\mathbf{U}\rangle$ on a Cartesian grid by setting J to be one.

Algorithm 1: Mapped HO-ACR for $\langle J\mathbf{U}\rangle$ of level $\ell+1$

// Determine the clipping bounds

for $\boldsymbol{i}^{\ell} \in \mathcal{C}(\Omega^{\ell+1})$ **do**

  for $\boldsymbol{i}^{\ell+1} \in \mathcal{C}^{-1}(\boldsymbol{i}^{\ell})$ **do**

    $\beta^{\min}_{\boldsymbol{i}^{\ell+1}} = \min\left(\langle\mathbf{U}\rangle_{\boldsymbol{j}^{\ell}}, \boldsymbol{j}^{\ell} \in \mathcal{I}(\boldsymbol{i}^{\ell})\right)$

    $\beta^{\max}_{\boldsymbol{i}^{\ell+1}} = \max\left(\langle\mathbf{U}\rangle_{\boldsymbol{j}^{\ell}}, \boldsymbol{j}^{\ell} \in \mathcal{I}(\boldsymbol{i}^{\ell})\right)$

  end for

end for

// Clip-and-redistribute

for $\boldsymbol{i}^{\ell} \in \mathcal{C}(\Omega^{\ell+1})$ **do**

  if $\text{sign}\left(\frac{\partial^2}{\partial\xi_d^2}\mathbf{U}_{\boldsymbol{i}^{\ell}}\right) \neq \text{sign}\left(\frac{\partial^2}{\partial\xi_d^2}\mathbf{U}_{\boldsymbol{i}^{\ell}\pm\boldsymbol{e}^b}\right)$ *for all* $b, d \in D$ **then**

    for $\boldsymbol{i}^{\ell+1} \in \mathcal{R}(\boldsymbol{i}^{\ell+1})$ **do**

      $\overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} = \frac{\langle J\mathbf{U}\rangle_{\boldsymbol{i}^{\ell+1}}}{\langle J\rangle_{\boldsymbol{i}^{\ell+1}}}$

      $\delta\overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} = \begin{cases} \overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} - \beta^{\max}_{\boldsymbol{i}^{\ell+1}} & \overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} > \beta^{\max}_{\boldsymbol{i}^{\ell+1}} \\ \overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} - \beta^{\min}_{\boldsymbol{i}^{\ell+1}} & \overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} < \beta^{\min}_{\boldsymbol{i}^{\ell+1}} \\ 0 & \text{otherwise} \end{cases}$

      $\delta\overline{\langle J\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} = \delta\overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}}\langle J\rangle_{\boldsymbol{i}^{\ell+1}}$

    end for

    $E_{\mathcal{R}} = \sum_{\boldsymbol{j}^{\ell+1}\in\mathcal{R}} \delta\overline{\langle J\mathbf{U}\rangle}_{\boldsymbol{j}^{\ell+1}}$

    for $\boldsymbol{i}^{\ell+1} \in \mathcal{R}(\boldsymbol{i}^{\ell+1})$ **do**

      $\alpha_{\boldsymbol{i}^{\ell+1}} = \begin{cases} \beta^{\max}_{\boldsymbol{i}^{\ell+1}} - \overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} & E_{\mathcal{R}} > 0 \\ \overline{\langle\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} - \beta^{\min}_{\boldsymbol{i}^{\ell+1}} & E_{\mathcal{R}} < 0 \\ 0 & \text{otherwise} \end{cases}$

      $\widetilde{\langle J\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} = \langle J\mathbf{U}\rangle_{\boldsymbol{i}^{\ell+1}} - \delta\overline{\langle J\mathbf{U}\rangle}_{\boldsymbol{i}^{\ell+1}} + \frac{\alpha_{\boldsymbol{i}^{\ell+1}} E_{\mathcal{R}}}{\sum_{\boldsymbol{j}^{\ell+1}\in\mathcal{R}} \alpha_{\boldsymbol{j}^{\ell+1}}}$

    end for

  end if

end for

3.3 Verification and Validation

The new HO-ACR algorithm is verified by a set of test cases involving the advection of smooth and sharp profiles on mapped grids. The features to be verified are: (1) ensuring high-order accu-

racy for smooth flow problems; (2) bound-preservation for non-smooth ones; and (3) maintaining conservation. Specifically the algorithm is examined for interpolation operations involving both the invalid ghost cells and the regridding process. The HO-ACR method is applied for invalid ghost cells and newly refined cells without fundamentally changing the base interpolation algorithm. The verification is first performed with the advection of a sharp profile in 1D, then advection of both a smooth profile and non-smooth profile in 2D. A more complicated case is explored with the solid body rotation of smooth and non-smooth profiles in 2D. All three features are demonstrated and assessed in each case.

3.3.1 Advection of a 1D sharp profile

To explore the effects of the HO-ACR method in Chord, a one dimensional scalar advection test using an Euler equation solver is explored. The initial condition top-hat profile is defined by

$$\phi = \begin{cases} 1 & \text{if } 0.6875 > x > 0.3125 \\ 0 & \text{otherwise} \end{cases}$$

on a one dimensional periodic Cartesian grid of length $L = 2$. This initial profile aligns the discontinuity with cell boundaries on the coarse grid with spacing $\Delta x = 1/16$, which preserves the sharp discontinuity. The profile is advanced to $t = 0.5$ with speed $u = 1$. A time step size of $\Delta t = 5 \cdot 10^{-3}$ is used on the coarse level, and finer levels are sub-cycled in time.

To investigate the HO-ACR method, three versions of "extreme" AMR are compared using Chord with and without the HO-ACR method. In each case, a single level of AMR with a refinement factor of 4 is used. Results are shown in figures 3.7, 3.8, and 3.9. To serve as references, solutions run entirely on the fine and coarse grids are also shown. Due to the nature of the 4^{th}-order finite volume method, small overshoots and undershoots appear around discontinuous features as they evolve in time, even on the coarse grid. Ideally the results in these test cases with "extreme" AMR should never be worse than the coarse solution, and good results should match the fine solution.

The first case (3.7) examines the effects of delaying regridding, by initializing on the coarse grid and first advancing for one coarse time step. Beginning at the second time step, the solution is either refined using the base interpolation algorithm of Chord or using the HO-ACR method. The solutions after the second time step are shown in Figure 3.7(a). A significant overshoot is observed in the base algorithm; the HO-ACR method effectively reduces these overshoots. We quantify the improvement in Table 3.1 by measuring the fine solution's overshoot relative to the analytic solution. The base algorithm has overshoots that are an order of magnitude larger than the reference fine solution, while the HO-ACR modification has slightly smaller errors than the fine solution. Although this represents a significant improvement over the base algorithm, the HO-ACR method does not fully remove the small oscillations near the discontinuity, which is reflected in the larger L_2 norm than the fine solution. To see the long-term impact, the solutions are advanced to 100 time steps, with results in Figure 3.7(b). During the advance, there are no further interpolations between levels, only the action of the limiter on a single-grid level. The overshoots from refining using the base algorithm are still present, while the solution using the HO-ACR solution closely matches the reference fine case, with extrema quantified in Table 3.2. The base algorithm has extrema that are nearly an order of magnitude larger than the fine solution, while the HO-ACR has extrema that almost match the fine solution. Comparing the difference in extrema demonstrates the improvement of HO-ACR, which is nearly two orders of magnitude smaller. To summarize, improper refinement of the initial condition by tagging one time step too late has significant and lasting consequences in the evolution of a solution. However, with the use of the HO-ACR method we are able to mitigate the effects of imperfect initial tagging. In more complex cases where discontinuous features appear spontaneously, this indicates that the HO-ACR method may be more accurate and robust with new flow features.

The second case examines the effects of ghost cell interpolation by specifying an initial profile with discontinuities on the coarse grid, shown in Figure 3.8. Directly ahead of the profile is a region of AMR fixed in space for $x > 0.75$. Figure 3.8(a) shows the solution as the leading edge enters the refinement region. The base algorithm produces a leading negative undershoot on the fine

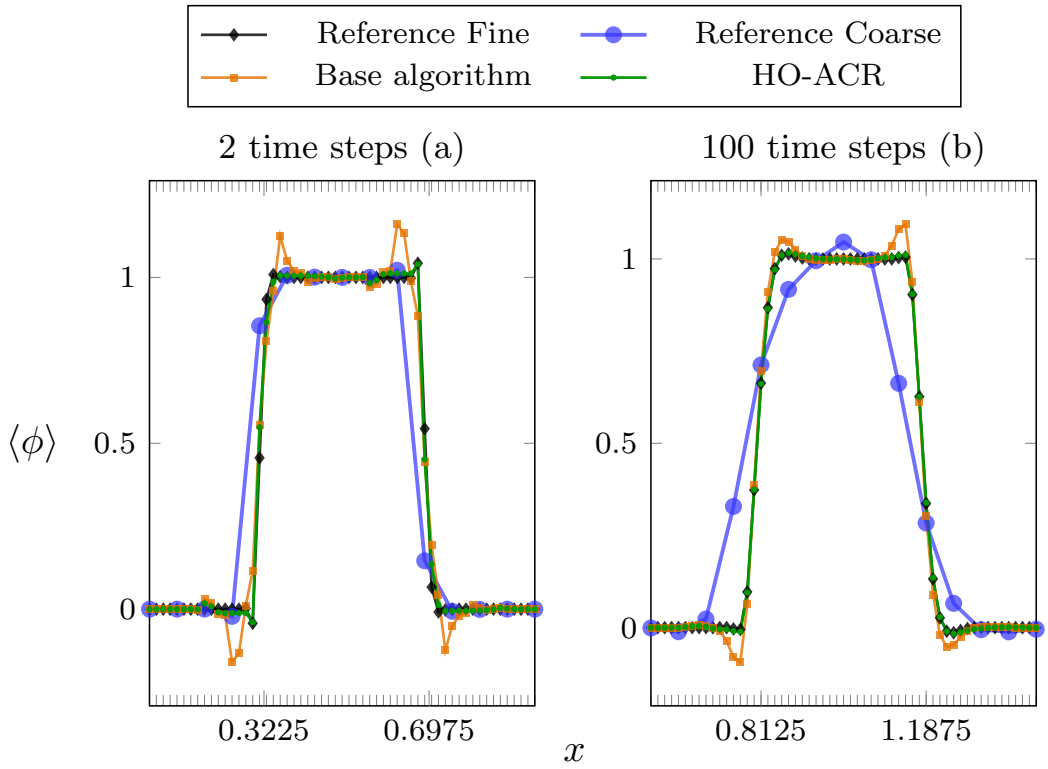


Figure 3.7: 1D advection with refinement spanning the entire domain at the beginning of the 2nd time step, with tick marks representing the finest grid spacing. The orange lines are the solution using the base algorithm of Chord, and green lines show the solution with addition of the HO-ACR method. The blue lines show the coarse solution values with no AMR, and black lines show the solution purely on the fine grid. The plot window follows the profile in time and the labeled x-axis locations indicate the analytic of solution discontinuities.

Table 3.1: Interpolation overshoots for 1D advection with AMR added at the beginning of the 2nd time step, measured at 2 time steps. Absolute overshoot is defined by solution values exceeding the bounds $[0, 1]$. The overshoot relative to fine data are calculated using solution values exceeding the fine solution peaks at the matching time. Since the HO-ACR method does not exceed the fine solution peaks, there are no relative errors to measure.

	Absolute Overshoot			Overshoot Relative to Fine	
	Base	HO-ACR	Fine	Base	HO-ACR
Max	$1.621 \cdot 10^{-1}$	$3.950 \cdot 10^{-2}$	$4.259 \cdot 10^{-2}$	$1.195 \cdot 10^{-1}$	—
Min	$-1.621 \cdot 10^{-1}$	$-3.950 \cdot 10^{-2}$	$-4.259 \cdot 10^{-2}$	$-1.195 \cdot 10^{-1}$	—
L_2	$8.757 \cdot 10^{-4}$	$1.690 \cdot 10^{-4}$	$1.511 \cdot 10^{-4}$	$5.969 \cdot 10^{-4}$	—
L_1	$1.189 \cdot 10^{-2}$	$7.144 \cdot 10^{-4}$	$8.014 \cdot 10^{-4}$	$1.047 \cdot 10^{-3}$	—

Table 3.2: Interpolation overshoots for 1D advection with AMR added at the beginning of the 2nd time step, measured at 100 time steps. Absolute overshoot is defined by solution values exceeding the bounds $[0, 1]$. Overshoot relative to the fine solution is calculated using values exceeding the minimum and maximum present in the fine solution at the same time.

	Absolute Overshoot			Overshoot Relative to Fine	
	Base	HO-ACR	Fine	Base	HO-ACR
Max	$9.623 \cdot 10^{-2}$	$1.635 \cdot 10^{-2}$	$1.374 \cdot 10^{-2}$	$8.249 \cdot 10^{-2}$	$2.611 \cdot 10^{-3}$
Min	$-9.623 \cdot 10^{-2}$	$-1.634 \cdot 10^{-2}$	$-1.375 \cdot 10^{-2}$	$-8.248 \cdot 10^{-2}$	$-2.597 \cdot 10^{-3}$
L_2	$2.793 \cdot 10^{-4}$	$5.335 \cdot 10^{-5}$	$6.872 \cdot 10^{-5}$	$2.211 \cdot 10^{-4}$	$4.744 \cdot 10^{-6}$
L_1	$5.503 \cdot 10^{-3}$	$1.230 \cdot 10^{-3}$	$1.187 \cdot 10^{-3}$	$3.842 \cdot 10^{-3}$	$3.794 \cdot 10^{-5}$

region, as well as an overshoot in the coarse region. The HO-ACR method prevents both of these new extrema, with minor oscillations from the 4^{th}-order evolution scheme. In Figure 3.8(b), the solution has been advanced until the profile has mostly entered the fine region. The extrema persist in the base algorithm while the HO-ACR method keeps them suppressed. Extrema at the end time are quantified in Table 3.3 relative to the analytic and fine solutions. The overshoots of the HO-ACR method are only marginally larger than the fine solution, while the base algorithm roughly doubles the overshoot. Surprisingly, the minimum values from the HO-ACR are even smaller than those in the fine solution. Because of the smoother solution from the HO-ACR method, the L_2 norm of its extrema is smaller than the fine solution's, and the L_1 norm nearly matches, while the base algorithm case is significantly worse. The differences in extrema between the base algorithm and the HO-ACR become more pronounced when compared relative to the fine solution. Relative to the extrema allowed by the fine solution, the HO-ACR method reduces the overshoot in L_2 and L_1 norms by over two orders of magnitude from the base algorithm. To summarize, this test demonstrates better robustness around sharp features and AMR interfaces when using HO-ACR. In realistic cases, this shows that discontinuous features can interact with AMR interfaces without causing excessive overshoots and undershoots.

The third case, shown in Figure 3.9, defines the initial profile with immediate refinement based on the solution gradients. The mesh is regridded at the beginning of each time step over the entire run. However, the refinement is applied minimally so that only one coarse cell on each side of the discontinuity is refined. This causes insufficient spacing between the discontinuity and the

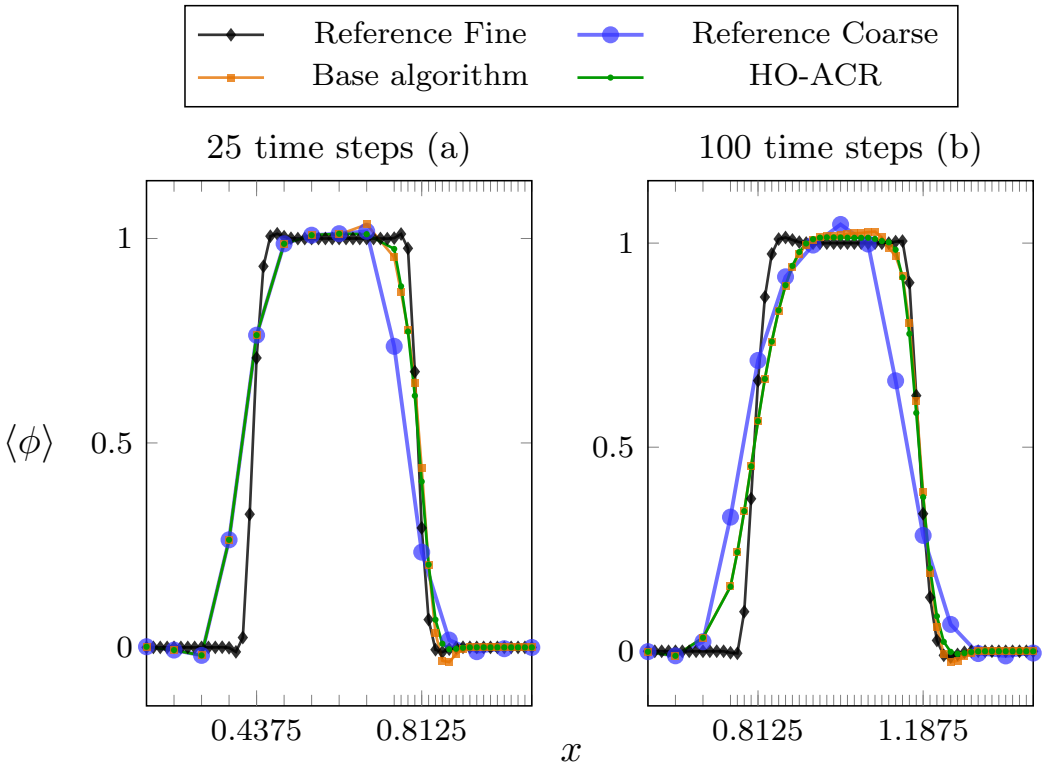


Figure 3.8: 1D advection with AMR at a fixed location in space, with refinement shown by the tick marks. The orange lines are the solution using the base algorithm of Chord, and green lines show the solution with the HO-ACR method. The blue lines show the values of the coarse solution with no AMR, and the black lines the solution purely on the fine grid. The plot window is adjusted to follow the profile in time, and the labeled points of the x-axis are the analytic location of the discontinuities.

Table 3.3: Interpolation overshoots for 1D advection with AMR at a fixed location in space, measured at 100 time steps. Absolute overshoot is defined by solution values exceeding the bounds $[0, 1]$. Overshoot relative to the fine solution is calculated using values exceeding the minimum and maximum present in the fine solution at the same time. Where the HO-ACR method has a smaller minimum value than the fine solution, no relative minimum value is recorded.

	Absolute Overshoot			Overshoot Relative to Fine	
	Base	HO-ACR	Fine	Base	HO-ACR
Max	$2.820 \cdot 10^{-2}$	$1.411 \cdot 10^{-2}$	$1.374 \cdot 10^{-2}$	$1.446 \cdot 10^{-2}$	$3.713 \cdot 10^{-4}$
Min	$-2.742 \cdot 10^{-2}$	$-1.113 \cdot 10^{-2}$	$-1.375 \cdot 10^{-2}$	$-1.368 \cdot 10^{-2}$	—
L_2	$7.006 \cdot 10^{-5}$	$2.011 \cdot 10^{-5}$	$6.872 \cdot 10^{-5}$	$6.332 \cdot 10^{-5}$	$9.153 \cdot 10^{-7}$
L_1	$3.338 \cdot 10^{-3}$	$1.725 \cdot 10^{-3}$	$1.187 \cdot 10^{-3}$	$1.073 \cdot 10^{-3}$	$7.237 \cdot 10^{-6}$

interpolation stencil for filling invalid ghost cells. After the time $t = 0.125$, Figure 3.9(a) shows the base algorithm solution has new extrema emerge in both the coarse and fine regions, and begins to move away from the reference fine solution. The HO-ACR method prevents these new extrema and is able to closely match the reference fine solution with less than half the number of cells. This becomes more pronounced as the solutions advance to the final time $t = 0.5$ (Figure 3.9(b)). At the end time Table 3.4 again shows the extrema of the base algorithm exceeding the analytic and fine solutions. Comparing extrema of the fine and HO-ACR solutions shows good agreement. Although we are hesitant to recommend this as a general practice, this test shows that using the HO-ACR method enables smaller regions of refinement (and reduced computational cost) with minimal loss of solution quality.

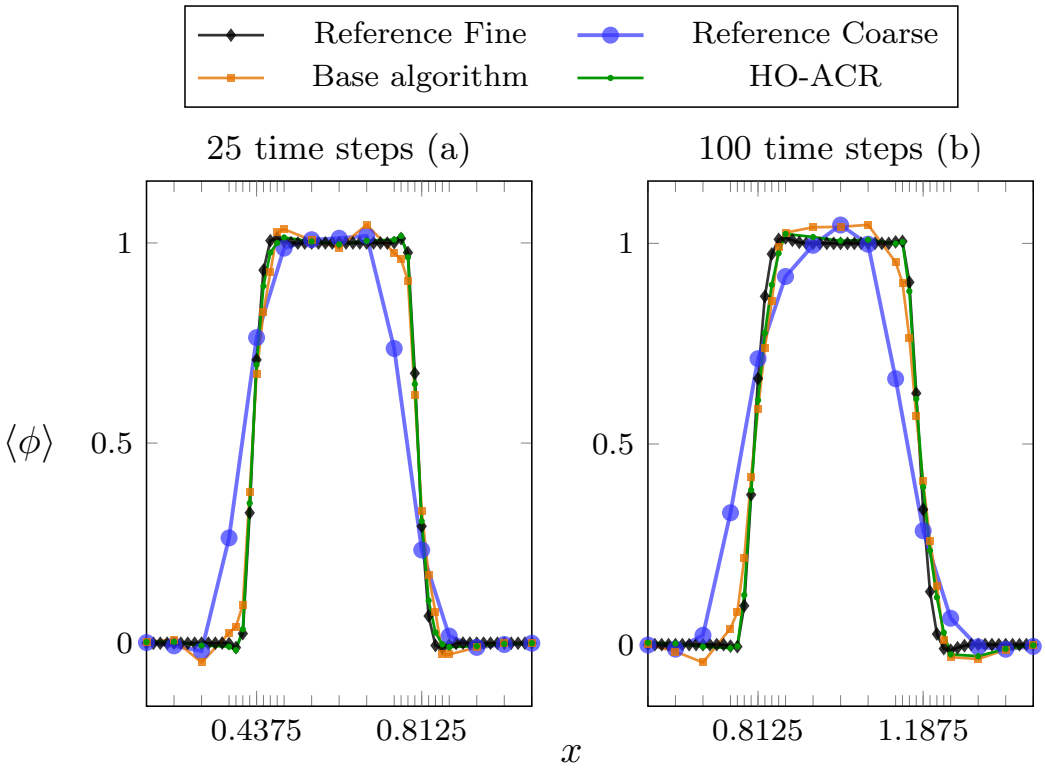


Figure 3.9: 1D advection with AMR using the minimal tagging of two coarse cells spanning the solution discontinuity, with refinement shown by the tick marks. The orange lines are the solution using the base algorithm of Chord, and green lines show the solution using the HO-ACR method. The blue lines show the values of the coarse solution with no AMR, and the black lines the solution purely on the fine grid. The plot window is adjusted to follow the profile in time, and the labeled points of the x-axis indicate the analytic solution discontinuities.

Table 3.4: Interpolation overshoots for the 1D advection, after 100 time steps, with minimal refinement of two coarse cells across the solution discontinuity. Absolute overshoot is defined by solution values exceeding the bounds $[0, 1]$. Overshoot relative to the fine solution is calculated using values exceeding the minimum and maximum present in the fine solution at the same time.

	Absolute Overshoot			Overshoot Relative to Fine	
	Base	HO-ACR	Fine	Base	HO-ACR
Max	$4.619 \cdot 10^{-2}$	$2.358 \cdot 10^{-2}$	$1.374 \cdot 10^{-2}$	$3.245 \cdot 10^{-2}$	$9.839 \cdot 10^{-3}$
Min	$-4.435 \cdot 10^{-2}$	$-2.874 \cdot 10^{-2}$	$-1.375 \cdot 10^{-2}$	$-3.060 \cdot 10^{-2}$	$-1.499 \cdot 10^{-2}$
L_2	$3.829 \cdot 10^{-4}$	$1.969 \cdot 10^{-4}$	$6.872 \cdot 10^{-5}$	$2.443 \cdot 10^{-4}$	$7.677 \cdot 10^{-5}$
L_1	$3.087 \cdot 10^{-3}$	$1.945 \cdot 10^{-3}$	$1.187 \cdot 10^{-3}$	$1.795 \cdot 10^{-3}$	$3.893 \cdot 10^{-4}$

3.3.2 Advection of a smooth profile

A Gaussian profile in density is specified by

$$\rho = \rho_0 + s(r)\,\Delta\rho\,\mathrm{e}^{-(100r^2)}\,, \tag{3.11}$$

where

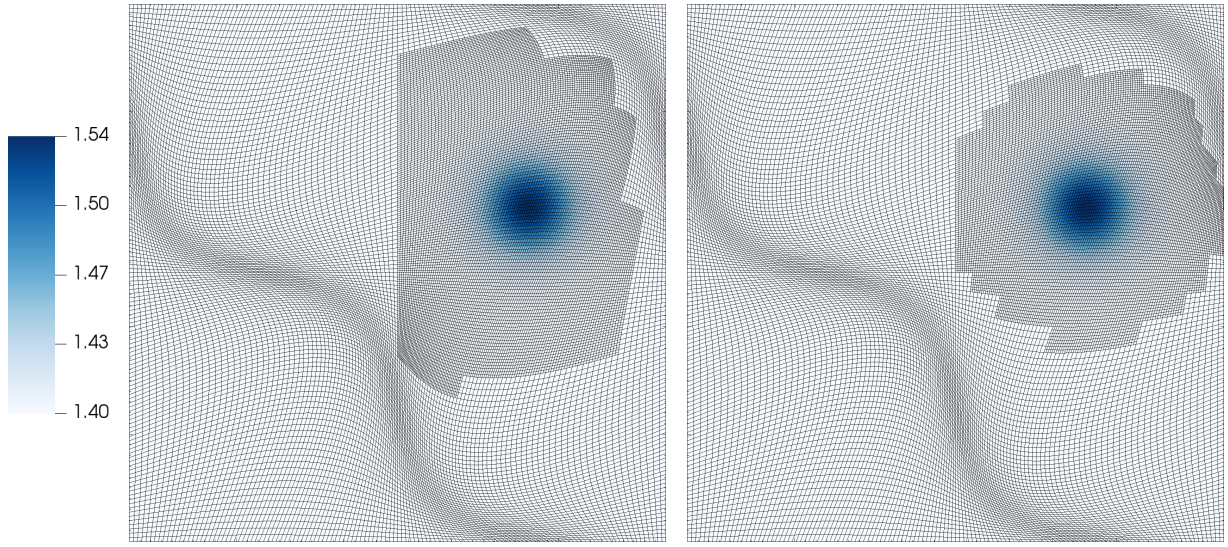
$$s(r) = \begin{cases} 0 & : \quad |4r| \geq 1 \\ \cos^6(2\pi r) & : \quad |4r| < 1 \end{cases}, \tag{3.12}$$

$\rho_0 = 1.4$, $\Delta\rho = 0.14$, and r specifies the distance of a point from initial profile center at $(0.25, 0.375)$ in the periodic domain, $[0, 1]^D$ where D is 2. Constant pressure and velocity are set as 1 and $(1.0, 0.5)$, respectively. An initial profile is specified on a coarse grid and advected in time. The grid mapping is defined as

$$x_d = \xi_d + \prod_{p=1}^{D} \sin(2\pi\xi_p) \qquad d = 1, 2\,. \tag{3.13}$$

To test invalid ghost cell interpolation, a fixed region of AMR (also called fixed AMR throughout the chapter) is added as shown in Figure 3.10(a). The initial profile does not initially fall inside the refined region, but is advected into it over time. Testing regridding is done by initially starting

with the coarse resolution only, then refining the solution profile based on the gradient of ρ with a level of AMR after a specified time of 0.25 time units, as shown in Figure 3.10(b).



(a) Fixed AMR **(b)** AMR

Figure 3.10: Warped meshes and solutions of density at a point in time for testing invalid ghost cell and regrid interpolation.

For a smooth profile, the HO-ACR scheme is designed to allow for new interpolated extrema to maintain the 4^{th}-order accuracy of the underlying finite-volume scheme. The profile is advected for a total of 0.5 time units, after which solution errors are calculated by comparing the numerical and exact solutions. Tables 3.5 and 3.6 show the measured solution errors in L_1, L_2, L_∞ for grids sizes 64^2, 128^2 256^2 and 512^2, and the convergence rates in L_1, L_2, L_∞ between each two consecutive grids, respectively. Clearly, the measures verify the high-order accuracy for both the fixed AMR and regridded AMR cases using a smooth profile.

Table 3.5: Convergence rates for advection of a Gaussian profile using a fixed region of AMR.

Coarse Cells	L_1	Rate	L_2	Rate	L_∞	Rate
64^2	$3.971 \cdot 10^{-5}$	—	$2.586 \cdot 10^{-4}$	—	$4.227 \cdot 10^{-3}$	—
128^2	$2.675 \cdot 10^{-6}$	3.89	$1.734 \cdot 10^{-5}$	3.90	$2.919 \cdot 10^{-4}$	3.86
256^2	$1.708 \cdot 10^{-7}$	3.97	$1.108 \cdot 10^{-6}$	3.97	$1.850 \cdot 10^{-5}$	3.98
512^2	$1.080 \cdot 10^{-8}$	3.98	$6.980 \cdot 10^{-8}$	3.99	$1.166 \cdot 10^{-6}$	3.99

Table 3.6: Convergence rates for advection of a Gaussian profile using AMR.

Coarse Cells	L_1	Rate	L_2	Rate	L_∞	Rate
64^2	$3.878 \cdot 10^{-5}$	—	$2.409 \cdot 10^{-4}$	—	$4.101 \cdot 10^{-3}$	—
128^2	$2.607 \cdot 10^{-6}$	3.90	$1.631 \cdot 10^{-5}$	3.88	$2.714 \cdot 10^{-4}$	3.92
256^2	$1.660 \cdot 10^{-7}$	3.97	$1.040 \cdot 10^{-6}$	3.97	$1.719 \cdot 10^{-5}$	3.98
512^2	$1.042 \cdot 10^{-8}$	3.99	$6.530 \cdot 10^{-8}$	3.99	$1.081 \cdot 10^{-6}$	3.99

Conservation is measured by computing a relative difference in the volume weighted sum of the solution. Ideally, these differences should be within machine precision of zero between the beginning and the end of a run. Using the fixed AMR case, Chord with the HO-ACR method is measured to have a relative difference in conservative values of $2.104 \cdot 10^{-14}$, whereas base Chord has value of $2.326 \cdot 10^{-14}$ on the 256^2 case. This small difference shows that the HO-ACR method is conservative. For the regridded AMR case, the relative conservation difference for the 256^2 mesh is measured as $3.416 \cdot 10^{-13}$ and $2.777 \cdot 10^{-13}$ using Chord with and without the HO-ACR method, respectively.

3.3.3 Advection of a sharp profile

In this test, using a similar setup to the Gaussian profile, a sharp profile is advected in a periodic domain allowing for verification that the HO-ACR method is bound-preserving. The domain is of size $[0, 1]^D$ where D is 2. The sharp profile is the "balls and jacks" inspired by Anderson et al. [48]. This profile is composed of three features prescribed in density $\rho_0 + \Delta\rho$ where $\rho_0 = 1.4$ and $\Delta\rho = 0.14$, as shown in Figure 3.11. Relative to a profile center of $(0.25, 0.375)$, a smaller circular shell is located at distance $(0.1, -0.1)$ from the center with an outer radius of 0.07 and an inner radius of 0.04. A larger circular shell is located at distance $(0.1, 0.1)$ from the profile center with an outer radius of 0.10 and an inner radius of 0.07. Finally, the jack base is placed at the distance $(-0.11, -0.11)$ from the profile center, with one arm specified by $[-0.04, -0.095] \times [-0.25, -0.125]$ the other by $[-0.095, -0.04] \times [-0.125, -0.25]$ and both are rotated counter-clockwise by $45\,^\circ$.

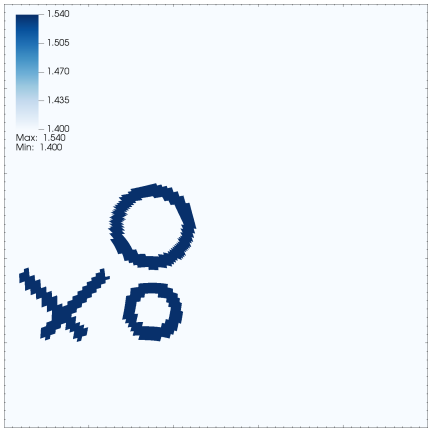
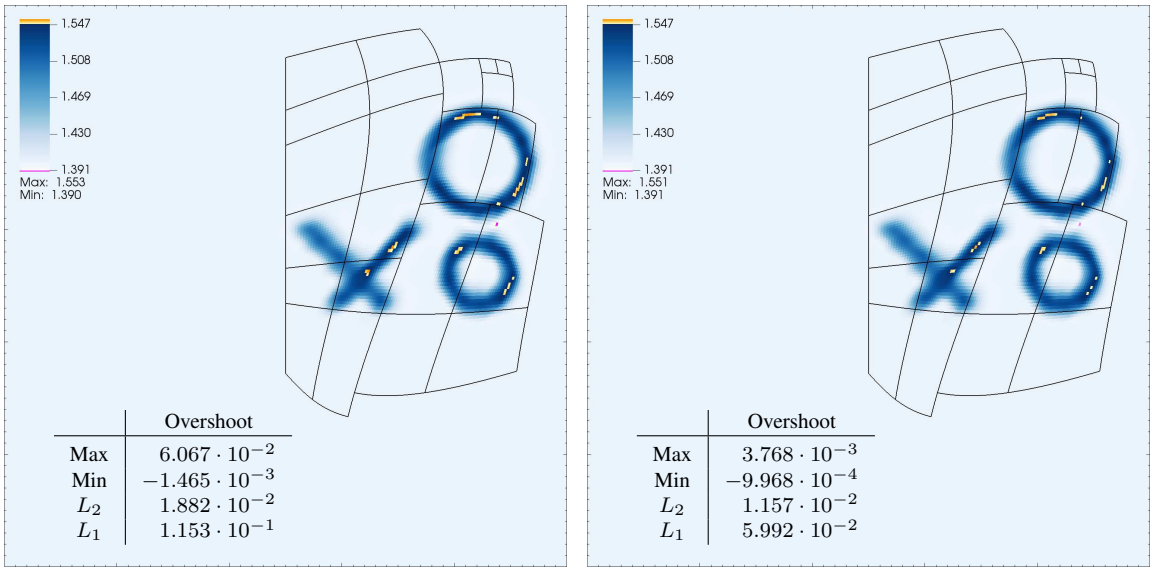


Figure 3.11: Initial density profile of the sharp "balls and jacks".

Beginning with the initial conditions as shown in Figure 3.11 and a fixed region of AMR, the profile is advected forward in time for 0.492843 time units. Figure 3.12 shows the fixed region of mesh refinement and compares the solution profiles obtained by Chord on the left and the inclusion of HO-ACR algorithm on the right. To emphasize, the HO-ACR method establishes the bounds from the solution on the coarse grid without AMR, serving as the basis to determine overshoots. Despite only bounding invalid ghost cells, a significant reduction in regions that experience overshoots beyond the coarse regions is seen when using the HO-ACR method. Although the improvement may not be immediately present based on the number of points producing new extrema, the magnitude of these overshoots is greatly reduced as shown by the tabulated data in Figure 3.12. Some new extrema are to be expected, as the high-order finite-volume algorithm includes other unbounded operations.

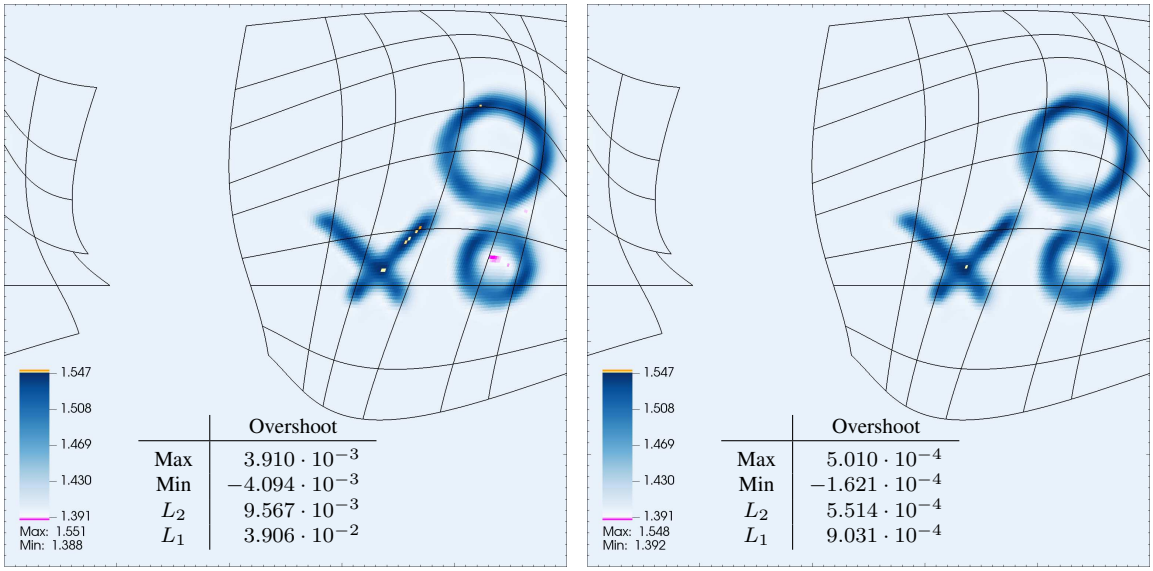
To test the HO-ACR method for the regridding process, the initial sharp profile in Figure 3.11 is advected forward in time for 0.492843 time units. A new region of refinement which tracks the profile is then added as shown in Figure 3.13, where the refinement spans the periodic boundary. The figure compares the solution profiles obtained by Chord without (a) and with (b) the HO-ACR algorithm. Clearly, the HO-ACR algorithm prevents many of the new extrema when regridding sharp solutions as in Figure 3.13. Notice from the HO-ACR solution profile that a small number of values appear to violate the bounds. However, this is due to the fact that some 4^{th}-order operations have happened between the final application of the HO-ACR and the output of the solution. These



(a) Base algorithm **(b)** With HO-ACR

Figure 3.12: AMR at a fixed location in space, testing invalid ghost cell interpolation.

operations will naturally produce new extrema beyond prior solution bounds. One example is the conversion from one set of cell-averaged quantities to another [52], such as done for computing primitives from conservative values or conservative values from the mapped values.



(a) Base algorithm **(b)** With HO-ACR

Figure 3.13: AMR added at this point in time, testing regrid interpolation.

Table 3.7 shows the solution growth of extrema, their L_1, and L_2-norms. For the fixed AMR test, the HO-ACR method is shown to have reduced the magnitude of the new maximum by a factor of 16, the minimum by a factor 1.5, and achieved modest reduction in the L_1 and L_2-norms. More impressively, the adaptive test reduces the magnitude of the new maximum by a factor of 8, and the minimum by a factor of 25. Decreases of over an order of magnitude are seen by the L_1 and L_2-norms. As in the prior section, conservation is measured and the HO-ACR method is found to have negligible impact on the total solution conservation. The relative conservation difference of the 256^2 fixed AMR case is measured as $3.318 \cdot 10^{-15}$ and $5.214 \cdot 10^{-15}$ using Chord with and without the HO-ACR method, respectively. Similarly, the regridded case measures relative conservation of $6.099 \cdot 10^{-13}$ and $1.054 \cdot 10^{-13}$ using Chord with and without the HO-ACR method.

Table 3.7: Interpolation overshoots for the advected "ball and jacks" case.

	Fixed		Adaptive	
	Base	HO-ACR	Base	HO-ACR
Max	$6.067 \cdot 10^{-2}$	$3.768 \cdot 10^{-3}$	$3.910 \cdot 10^{-3}$	$5.010 \cdot 10^{-4}$
Min	$-1.465 \cdot 10^{-3}$	$-9.968 \cdot 10^{-4}$	$-4.094 \cdot 10^{-3}$	$-1.621 \cdot 10^{-4}$
L_2	$1.882 \cdot 10^{-2}$	$1.157 \cdot 10^{-2}$	$9.567 \cdot 10^{-3}$	$5.514 \cdot 10^{-4}$
L_1	$1.153 \cdot 10^{-1}$	$5.992 \cdot 10^{-2}$	$3.906 \cdot 10^{-2}$	$9.031 \cdot 10^{-4}$

3.3.4 Solid Body Rotation

Another standard test problem for the HO-ACR method uses 2D solid body rotation to advect a passive scalar quantity with different localized initial profiles. The velocity field is defined by the profile $u = \omega(y, -x)$, where the rotational speed is $\omega = 20\pi$ in the clockwise direction. The initial profiles (Anderson et al. [53]) are specified by

$$\phi = \phi_0 + \Delta\phi(\phi_{\text{cone}}(r_{\text{cone}}) + \phi_{\text{bump}}(r_{\text{bump}}) + \phi_{\text{cyl}}(r_{\text{cyl}})) \tag{3.14}$$

where $\phi_0 = 0$, $\Delta\phi = 1$, and $r_{\text{cone}}, r_{\text{bump}}, r_{\text{cyl}}$ specify the distances from the initial profile center as $(\cos(\pi/6), \sin(\pi/6))$, $(\cos(3\pi/2), \sin(3\pi/2))$, and $(\cos(5\pi/6), \sin(5\pi/6))$ respectively. The cone,

bump, and notched cylinder profiles are given as

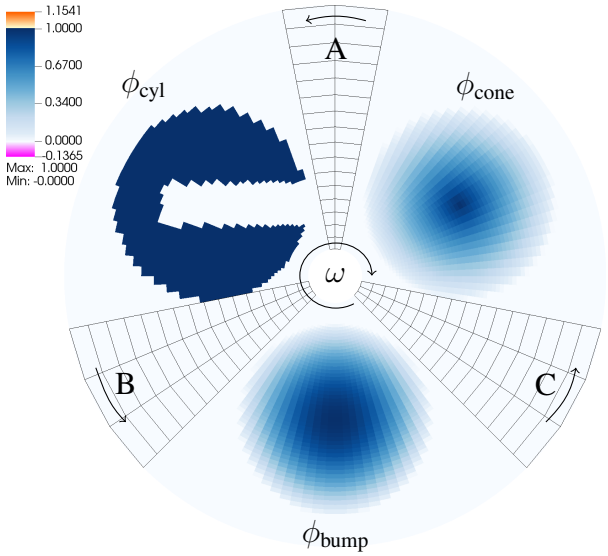
$$\phi_{\text{cone}}(r) = \begin{cases} 0 & \text{if } r > 1 \\ 1 - r & \text{otherwise} \end{cases} \tag{3.15}$$

$$\phi_{\text{bump}}(r) = \begin{cases} 0 & \text{if } r > 1 \\ \cos(\pi r) & \text{otherwise} \end{cases} \tag{3.16}$$

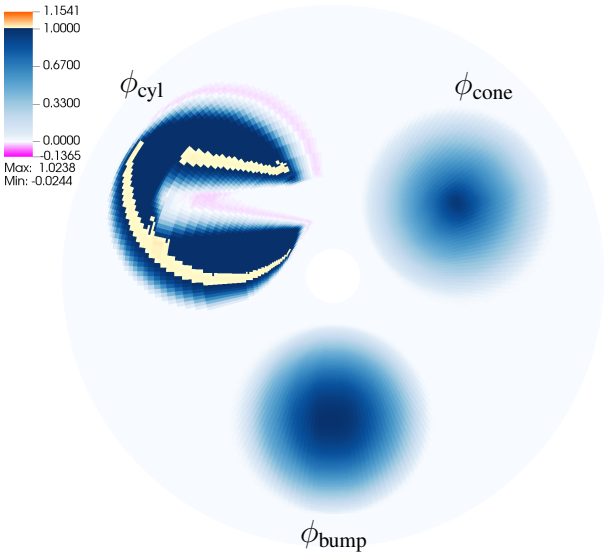
$$\phi_{\text{cyl}}(r) = \begin{cases} 0 & \text{if } r > 1 \\ 0 & \text{if } (1.3r_j > y > 0.7r_j) \text{ and } (x > 1.4r_i) \\ 1 & \text{otherwise} \end{cases} \tag{3.17}$$

and shown as evaluated on the initial mesh in Figure 3.14(a). The structured grid is defined by an annulus centered at the origin with outer radius $\sqrt{8}$ and inner radius $0.1\sqrt{8}$. The base mesh uses 64 cells in the radial direction and 128 in the angular. The inner and outer boundary conditions are specified by Dirichlet conditions using the analytic solution values. The solution is advanced in time for one rotation, or 0.1 units in time, after which the profiles should exactly match their initial conditions.

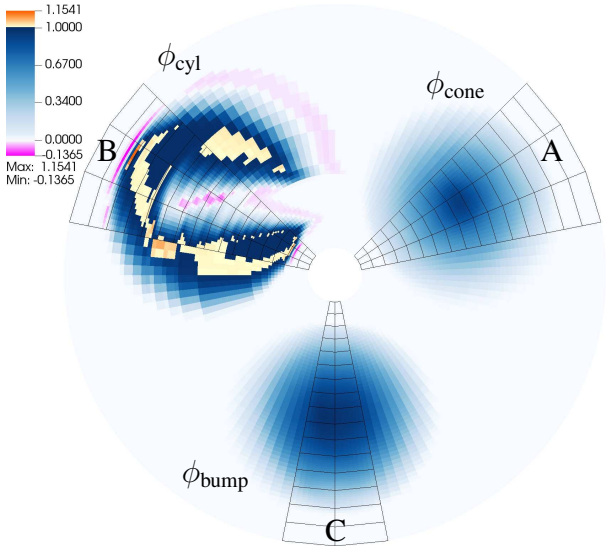
This test is motivated by tagging regions other than where the solution discontinuities are, and forcing the discontinuities to pass through the refined regions. For this imperfect application of AMR, the primary objective for the scalar solution is that it is stable and produces no new extrema as it is regridded and advects across AMR boundaries. This case is examined using three patches of counter rotating AMR at a refinement factor of 2. These patches are initially centered about $\pi/2$, $7\pi/6$, and $11\pi/6$ with widths of $\pi/8$, respectively marked A, B, and C in Figure 3.14(a). The AMR patches are rotated in the opposite direction of the scalar advection by an angular velocity of $-5\omega/6$, and regridded every 10 time steps. Initial profiles are defined only in coarse regions with this choice of AMR, whereas the final profiles primarily fall inside refined regions, indicated with black boxes in Figure 3.14. Results from using the base algorithm of Chord and with the



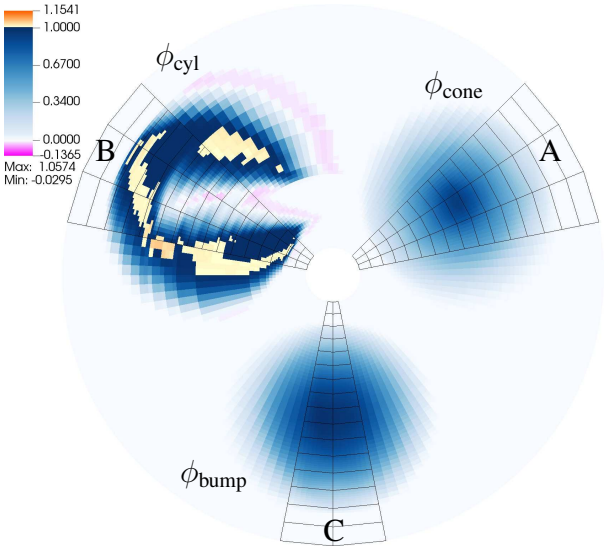
(a) Initial condition.



(b) Fine solution after one rotation.

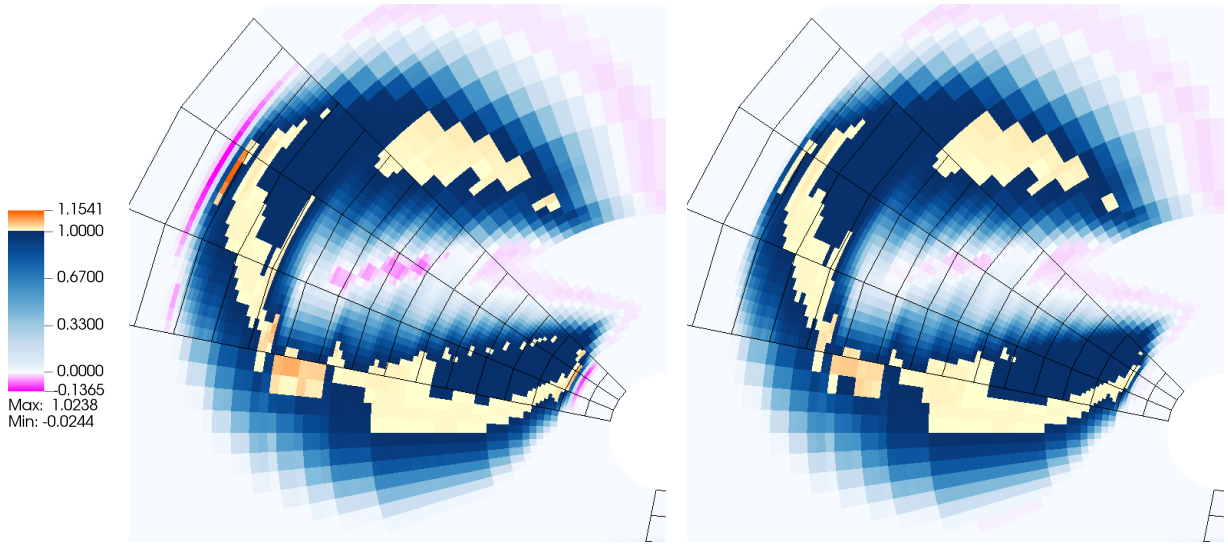


(c) Solution with counter rotating AMR after one rotation, using the base algorithm.



(d) Solution with counter rotating AMR after one rotation, with HO-ACR.

Figure 3.14: Solution of a passive scalar after advecting for one clockwise rotation. Negative mass fractions are shown in magenta, and positive overshoots beyond the analytic solution bounds of $[0, 1]$ are shown in yellow-orange. Black lines are a coarsened representation of the AMR region. Subfigures (b)-(d) show the results for a fine-only mesh, the base AMR algorithm with moving refinement domains, and the HO-ACR method with the same meshes, respectively. Each solution profile is labeled as ϕ_{cone}, ϕ_{bump}, and ϕ_{cyl} matching the definitions given in Equation 3.17. The refinement regions labeled as A, B, and C, are regridded to make $5/6^{\text{ths}}$ of a rotation counter-clockwise.



(a) Solution with counter rotating AMR after one rotation, using the base algorithm of Chord.

(b) Solution with counter rotating AMR after one rotation, with HO-ACR.

Figure 3.15: Close-up of the notched cylinder results after advecting for one rotation, as shown in Figure 3.14. Negative mass fractions are shown in magenta, and positive overshoots beyond the analytic solution bounds of $[0, 1]$ are shown in yellow-orange. Black lines are a coarsened representation of the AMR region.

addition of the HO-ACR method are examined and shown in Figure 3.14(c) and Figure 3.14(d), respectively. A reference using a fine grid for the entire domain is shown in Figure 3.14(b). In each of these cases values outside the analytic solution bounds of $[0, 1]$ are highlighted, with overshoots in the yellow-orange color, and undershoots in magenta.

The reference fine solution, in Figure 3.14(b), shows some small new extrema near the discontinuous features of the notched cylinder due to the 4^{th}-order flux calculations in Chord. Even with imperfect AMR, in Figure 3.14(c) and Figure 3.14(d), the cone and bump profiles do not show any overshoot. This is expected since the scheme's numerical dissipation smooths continuous profiles over time, and AMR interpolation is designed for smooth solutions. With this in mind, the continuous cone and bump regions of the solutions show no distinguishable differences with or without the HO-ACR method, which indicate it is not changing the algorithm substantially in smooth regions. However, the more extreme case of the notched cylinder using the counter rotating AMR with Chord in Figure 3.14(c), shows significantly larger overshoots and undershoots. These errors are the most pronounced at the edges of the notched cylinder, especially near the inner and

outer boundaries, parallel to the velocity. These larger extrema are quantified in Table 3.8, labeled as the "Base" columns, where the peak errors are seen to be an order of magnitude larger than what is observed in the fine solution. In contrast, the HO-ACR method shown in Figure 3.14(d) greatly reduces the extrema in the solution and closely resembles the fine solution. For better visual comparison between the two cases, Figure 3.15 shows solutions in a window containing only the notched cylinder. The extrema are measured in Table 3.8, (columns labeled as "HO-ACR") where we see that the peak value errors are an order of magnitude smaller than the base case, and comparable to the fine case. The same trend is apparent in the L_1 and L_2 measure of the errors, with the HO-ACR overshoot an order of magnitude smaller than the base case and closer to the fine solution.

Table 3.8: Measured interpolation overshoots for the solid body rotation case. The data for the absolute overshoot are calculated from solution values exceeding the analytic bounds $[0, 1]$. The overshoots relative to the fine solution are calculated at the same time.

	Absolute Overshoot			Overshoot Relative to Fine	
	Base	HO-ACR	Fine	Base	HO-ACR
Max	$1.541 \cdot 10^{-1}$	$5.741 \cdot 10^{-2}$	$2.381 \cdot 10^{-2}$	$1.303 \cdot 10^{-1}$	$3.362 \cdot 10^{-2}$
Min	$-1.365 \cdot 10^{-1}$	$-2.954 \cdot 10^{-2}$	$-2.435 \cdot 10^{-2}$	$1.121 \cdot 10^{-1}$	$-5.197 \cdot 10^{-3}$
L_2	$5.243 \cdot 10^{-5}$	$2.577 \cdot 10^{-5}$	$1.485 \cdot 10^{-5}$	$1.607 \cdot 10^{-5}$	$2.216 \cdot 10^{-6}$
L_1	$1.258 \cdot 10^{-3}$	$8.091 \cdot 10^{-4}$	$6.144 \cdot 10^{-4}$	$2.806 \cdot 10^{-4}$	$1.794 \cdot 10^{-5}$

3.4 Results and Discussion

The verified HO-ACR algorithm is applied to solve two types of cases. In the first case, a H_2 bubble is advected through a strong shock in 2D. This presents numerical challenges, including properly tracking of the thin H_2-O_2 flame fronts and the strong discontinuities of shock waves. The second case is the C_3H_8-air combustion in a bluff-body combustor in which the chemistry is extremely stiff. In both these cases, the HO-ACR method is applied to invalid ghost cells and newly refined cells with no fundamental changes to the base algorithm. The investigation has demonstrated that the HO-ACR method has played a critical role in ensuring the stability and

accuracy in the process of obtaining solutions. Immediately what follows is the description of each problem. Then, results are presented and discussed.

3.4.1 H_2-O_2 Shock Induced Combustion

The configuration is shown in Figure 3.16 and follows the setup used in a previous study [27]. A Mach 2 steady planar shock is located $0.75\,\mathrm{cm}$ to the right of the origin and parallel to the y-axis. A hydrogen bubble is located just upstream of the shock. The upper and lower boundaries of the y-direction are periodic, while x-min boundary condition is Dirichlet and the x-max is extrapolated.

The initial velocities $U_u = 1.24 \cdot 10^5\,\mathrm{cm/s}$ and $U_s = 4.35 \cdot 10^4\,\mathrm{cm/s}$ are the velocities upstream and downstream of the shock. The hydrogen mass fraction is initialized to

$$c_{\mathrm{H_2}} = \frac{1}{2}\left[1 + \tanh\left(\frac{r_c - r}{C_2}\right)\right], \quad r = \sqrt{((x - x_0)^2 + (y - y_0)^2)}, \tag{3.18}$$

with r_c the radius of the bubble and the coordinate (x_0, y_0) the center of the bubble. The coefficient C_2 determines the sharpness of the interface between the H_2 bubble and the surrounding air. The case parameters are $C_2 = 3 \cdot 10^{-3}\,\mathrm{cm}^{-1}$, $r_c = 0.28\,\mathrm{cm}$, $(x_0, y_0) = (0.4, 0.75)\mathrm{cm}$, and $\lambda = 1.5\,\mathrm{cm}$. The mass fractions for the surrounding air are simplified to $c_{\mathrm{N_2}} = 0.767$ and $c_{\mathrm{O_2}} = 0.233$. The H_2-O_2 combustion is a 9-species, 19-reaction mechanism described by Billet et al. [54].

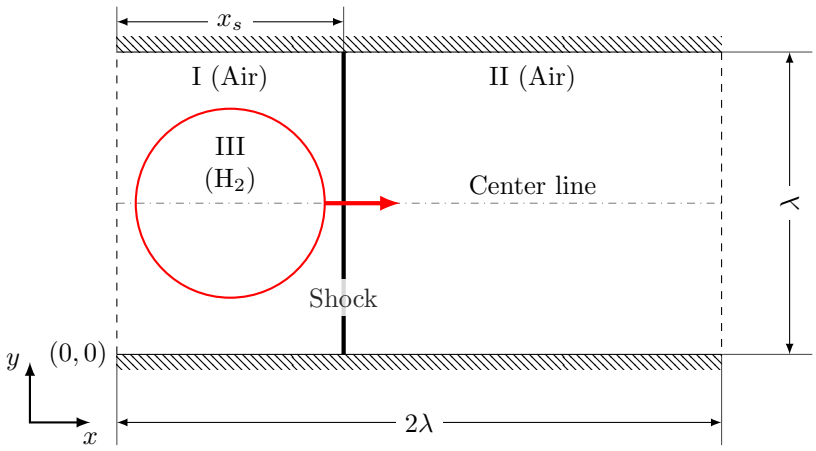


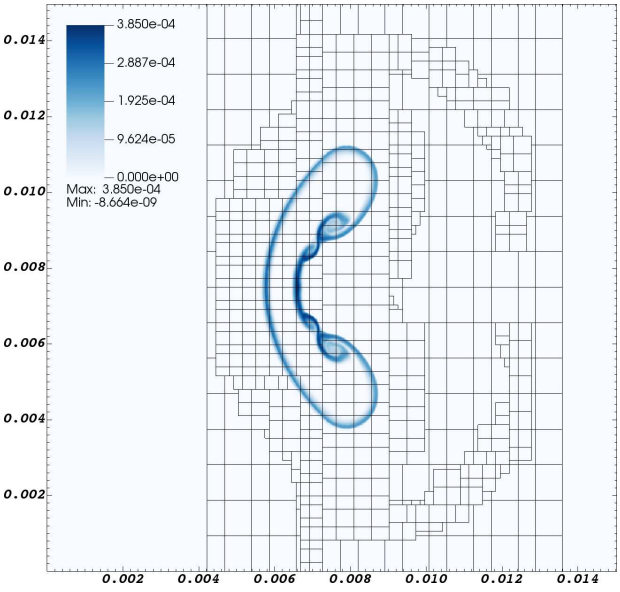
Figure 3.16: Shock bubble case configuration.

A traditional approach for this case would use AMR to completely refine around the discontinuities. Instead, an imperfect AMR tagging scenario that is representative of what might occur in an embedded-DNS simulation is considered. Assume only a small window of refinement can be afforded in the solution process, and due to this limitation significant features will cross AMR interfaces. Additionally, refinement is only added once the flow is developed. Ideally, the HO-ACR algorithm will handle this imperfect tagging.

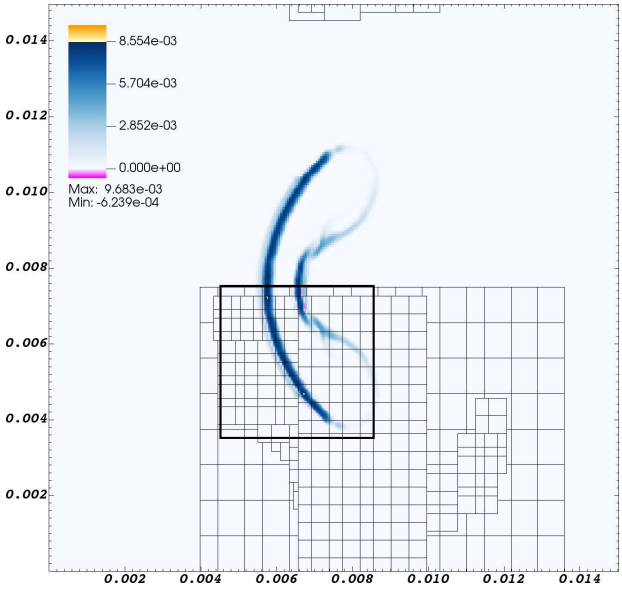
In total, three simulations are performed. Two cases are run for comparison: one with base Chord (without the HO-ACR method), and the other with the HO-ACR method. The base mesh has 512×256 cells. Both cases begin with only the base mesh until $t = 5.3\,\mu\text{s}$. Then two AMR levels with a refinement ratio of 2 are added based on the gradients of density and pressure for the lower half of the domain, with a small buffer wrapping around the periodic boundary. For comparison, the third case is run by using AMR on the entire domain from the initial solution, which is considered as the "ideal" case. The goal here is to demonstrate that the HO-ACR method enables us to obtain a quality solution even if imposing AMR in arbitrary space-time and crossing strong discontinuities. This strategy may be performed for embedded-DNS or embedded-LES where only a portion of the space-time mesh is appropriately refined due to computational expense.

Figure 3.17 shows the solution of H_2O at $t = 5.3\,\mu\text{s}$. In the figure, there are 4 subfigures: (a) the "ideal" case where AMR has been used for the entire run; (b) showing an initially coarse solution from Chord with refinement added; (c) and (d) comparing the closeups of solutions between Chord without and with the HO-ACR algorithm for the domain regridded on the bottom half, as indicated by the black rectangles in the prior subfigures. After the initial refinement as seen in the subfigure (c), negative species mass fractions, as indicated by magenta, are large, and positive overshoots beyond the coarse solution are shown in yellow-orange. While mass fractions of species must be always positive to be physically meaningful, Chord does have a small tolerance for negative mass fractions to cope with the numerical instability in the nonlinear solvers employed during the solution process [55]. In the subfigure (d), the magnitude of the negative mass fractions is reduced

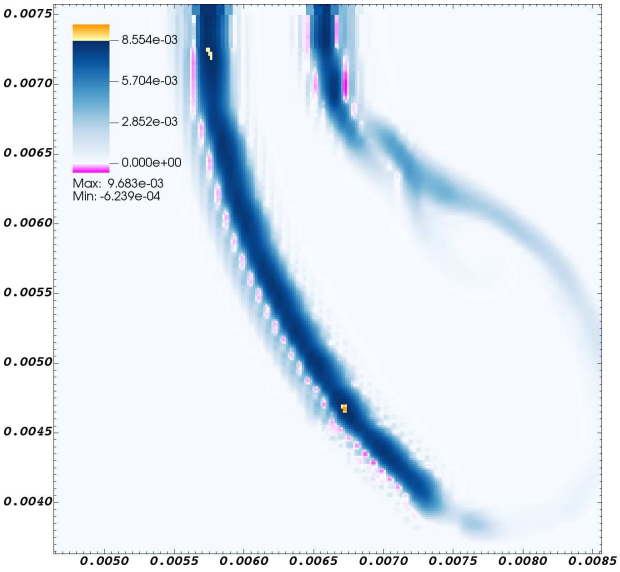
by an order of magnitude, which is sufficient to have kept the overall algorithm stable and allow the solution to develop properly.



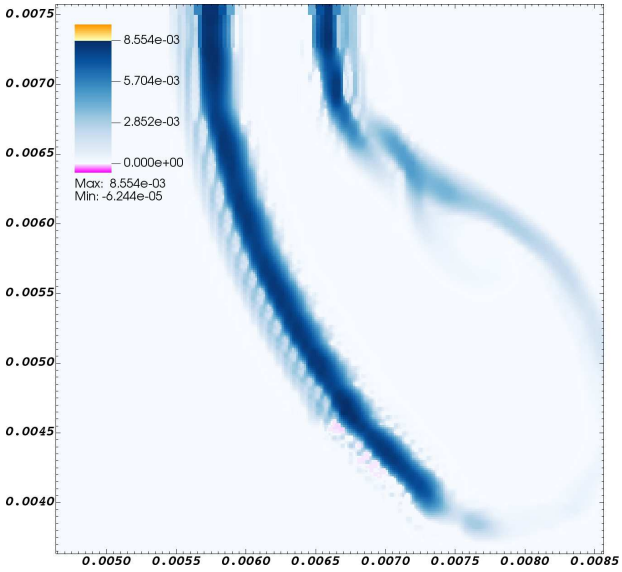
(a) Full AMR the entire run time.



(b) Solution regridded on half the domain at current time, using Chord.



(c) Close-up view of the boxed region in (b) for the solution using Chord.



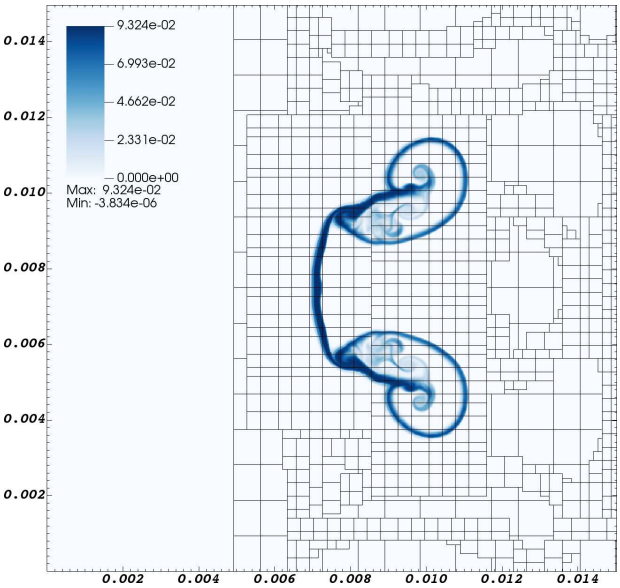
(d) Close-up view of the boxed region in (b) for the solution using Chord with HO-ACR.

Figure 3.17: Solution of H_2O at $t = 5.3\,\mu s$. In subfigures (b)-(d) negative species mass fractions are shown in magenta, and positive overshoots beyond the coarse solution are shown in yellow-orange.

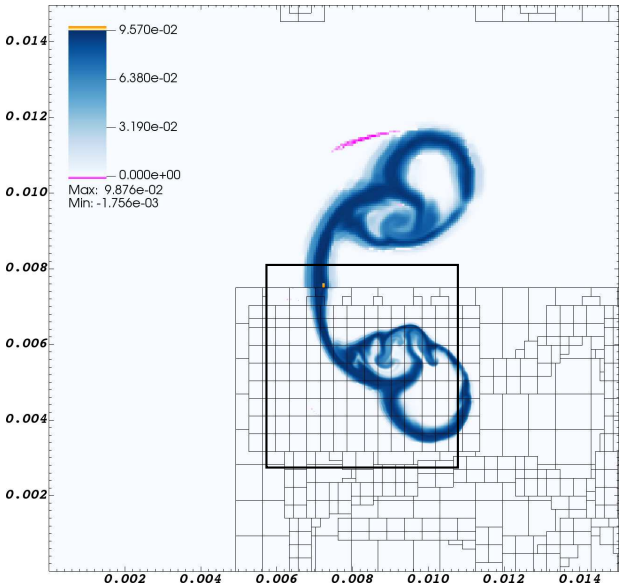
Figure 3.18 shows the H_2O after continuing to evolve the solution to $t = 10\,\mu s$. The layout of this figure is the same as Figure 3.18. Although not matching the "ideal" solution, the partially added refinement is seen to begin capturing fine scale features not visible in the coarse solution while using significantly less computational resources. This provides an ability to assess the validity of embedded-DNS or LES, although we refrain from making judgment here. Instead, we focus on the quality of the interpolations. The magnitude of the solution difference with and without the HO-ACR method grows smaller in comparison to that of the earlier time. There is a significant overshoot along the AMR interface that is effectively prevented by the HO-ACR algorithm. Despite what may appear to be minor differences, this is of significant impact for chemically reacting flows with stiff reactions where minor errors can push the solution to unphysical states that are unsolvable. Running further in time, this overshoot would eventually destabilize the solution if the HO-ACR method is not applied.

3.4.2 C_3H_8-Air Combustion in a Bluff-Body Combustor

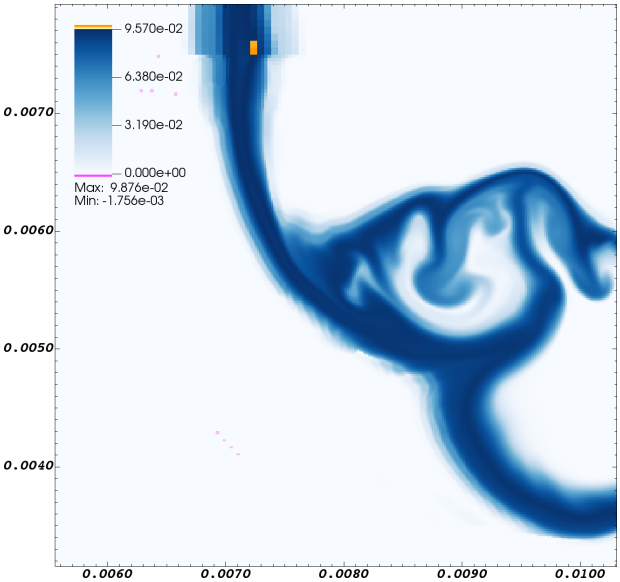
The premixed combustion of C_3H_8-air is simulated in a bluff-body combustor [56, 57]. As shown in Figure 3.19, the triangular bluff-body lies in a rectangular channel. At the inlet, a gaseous mixture of 4.01% C_3H_8, 22.36% O_2, and 73.64% N_2 by mass fraction flows into the domain at a velocity of $15.7\,\mathrm{m/s}$. The mixture has a temperature of $310\,\mathrm{K}$ and a pressure of $101{,}325\,\mathrm{Pa}$. The flow has a bulk Mach number of 0.053 and bulk Reynolds number of $50{,}000$. The reaction mechanism for C_3H_8-air is a stiff system including 25-species and 66-reactions [31]. To ensure stability, only the ACR method is explored for this application and the high-order extension is disabled. We revert to the low order (strictly bound-preserving) ACR method for this case, because the reacting species are highly sensitive to any overshoots allowed by the HO-ACR method. The initial conditions in the domain are set to the same values as the inlet, with the exception of a small region surrounding the bluff-body. Near the bluff-body, the gas mixture is initialized to 12% CO_2, 6.54% H_2O, 5.14% O_2, and 73.62% N_2, with a temperature of $1300\,\mathrm{K}$. This hot spot provides the ignition of the combustion.



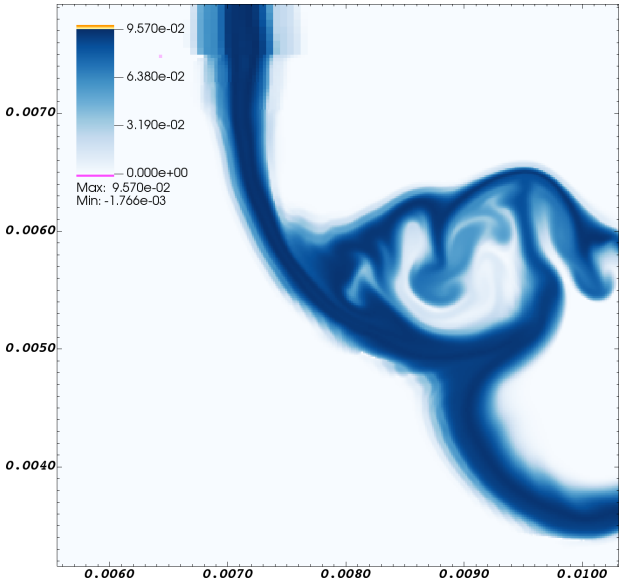
(a) With full AMR the entire run time.



(b) Current solution after AMR added on half the domain at $t = 5.3\,\mu s$, using Chord.



(c) Close-up view of the boxed region in (b) for the solution using Chord.



(d) Close-up view of the boxed region in (b) for the solution using Chord with HO-ACR.

Figure 3.18: Solution H_2O at $t = 10\,\mu s$. In subfigures (b)-(d) negative species mass fractions are shown in magenta, and positive overshoots beyond the coarse solution of the HO-ACR case (d) are shown in yellow-orange.

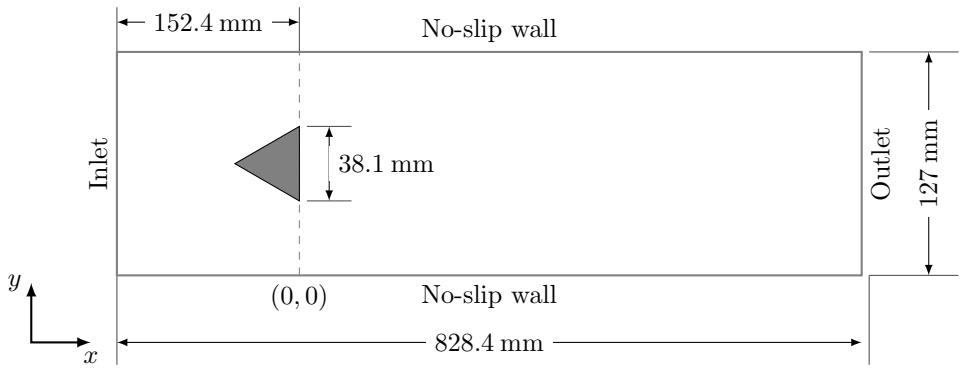


Figure 3.19: The geometry of the bluff-body combustor.

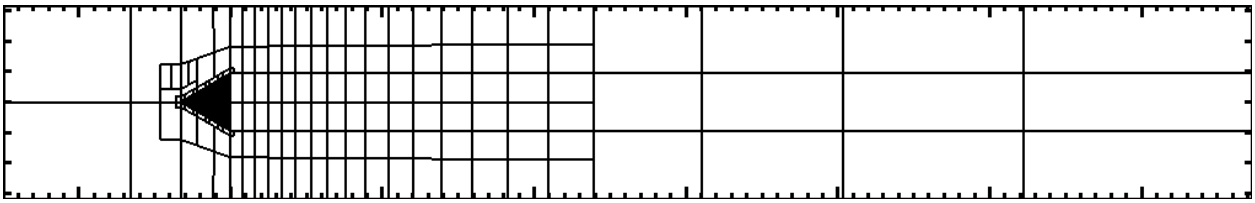
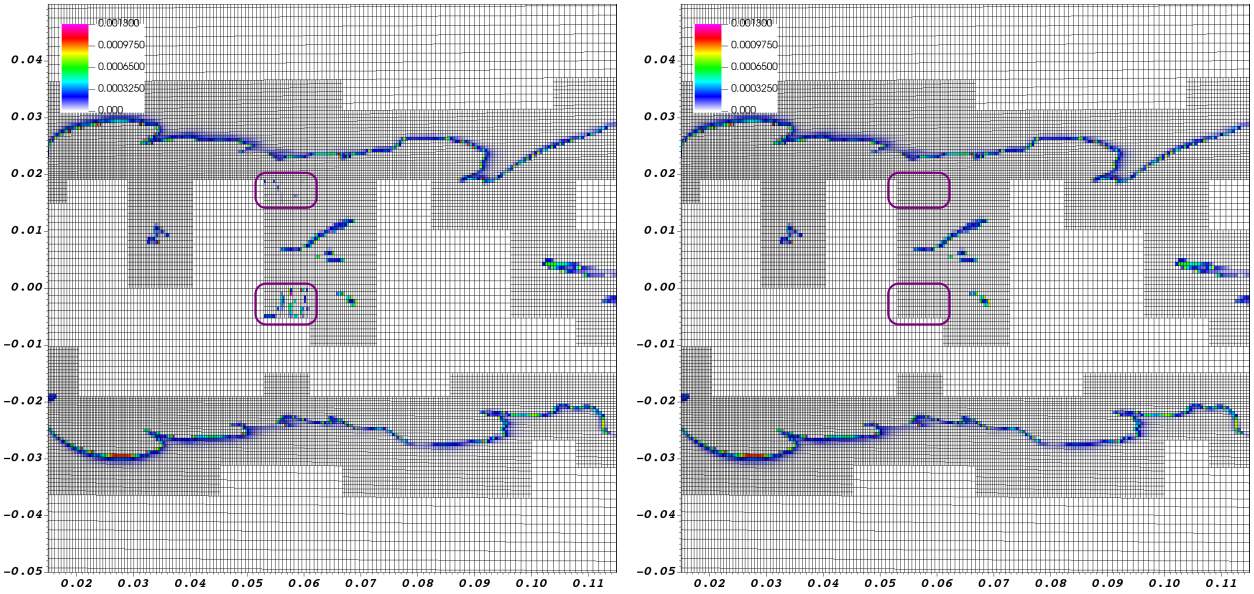


Figure 3.20: Initial grid boxes in physical space for the bluff-body combustor.

A base mesh of $11\,008$ cells is used, with two levels of refinement at a ratio of 2. The 2^{nd} level is fixed behind the bluff-body and spans the experimental observation window. The 3^{rd} level dynamically tracks the flame and boundary layers. The bluff-body case uses the conforming mapped multiblock technique to fit a structured mesh to the geometry. The initial grid boxes are shown in Figure 3.20. Species OH and CH_2O are good flame markers. Therefore, during the run time, cells defining the 3^{rd} level are tagged based on $c_{\text{OH}} \times c_{\text{CH}_2\text{O}} > 2.0 \cdot 10^{-9}$.

Figure 3.21 shows the CH_2O contour in the region immediately downstream of the trailing edge of the bluff-body. On the left, the solution obtained from base Chord after regridding displays small artifacts arising from interpolation overshoots as highlighted by the violet bubbles. On the right, the presence of these noises in the solution is diminished greatly by instead applying the ACR method. Similarly, Figure 3.22 compares the OH mass distributions between the base and the ACR algorithms after regridding for the same physical location and solution time. While the differences in OH are barely visible in Figure 3.22, after 80 time steps, these grow into quite large perturbations as shown in Figure 3.23. The effectiveness of using the ACR to suppress the interpolation overshoots is clearly seen in Figure 3.23, as highlighted by gray bubbles. Through the

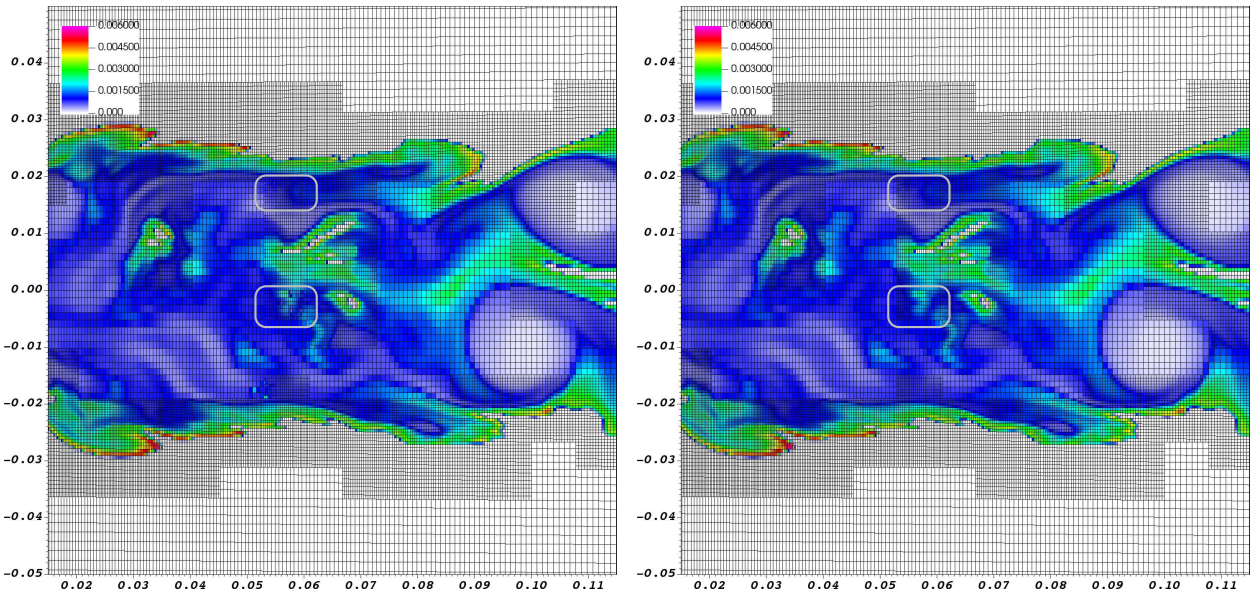
numerical experiments, the ACR algorithm is found to be necessary to ensure the overall simulation stability.



(a) Chord **(b)** Chord with ACR

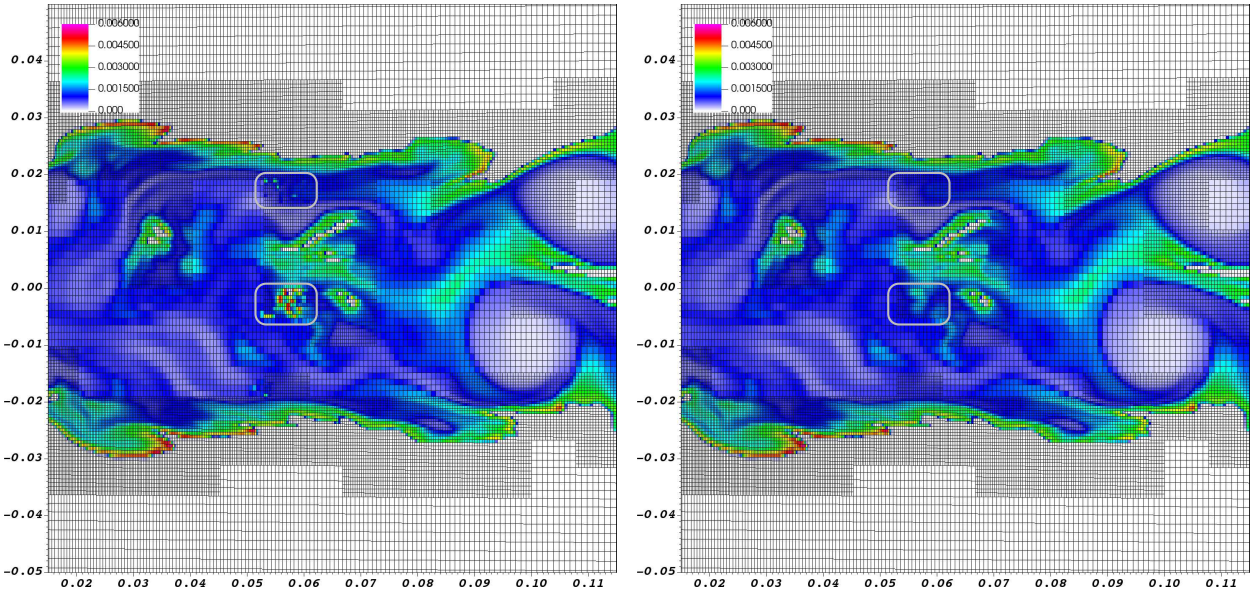
Figure 3.21: Solution of CH_2O after regridding. Small artifacts from interpolation overshoot are present in the base algorithm, but suppressed using the ACR method in regions highlighted by the violet bubbles.

With the ACR method, a full simulation of the three-dimensional version of the bluff-body case was performed. The z-direction is defined with width of $76.2\,\mathrm{mm}$, and enforced with periodic boundaries. The base mesh is approximately $330\,000$ cells. One additional level with a refinement factor of 2 is added. Figure 3.24 shows the instantaneous isosurfaces of the vorticity, temperature, mass fractions of H_2O and OH, respectively, for the 3D C_3H_8-air flame at a solution time of $t = 246\,\mathrm{ms}$. Without using AMR, it is much more expensive to capture the flow and flame dynamics near the bluff-body as shown in Figure 3.24. Furthermore, without using the ACR method, this 3D combustion simulation has numerical difficulty where the flame crosses AMR interfaces. One could refine the flame everywhere in the domain to avoid numerical issues from AMR, but this would require significantly more computing resources for little benefit – high fidelity is only required in the experimental observation window.



(a) Chord **(b)** Chord with ACR

Figure 3.22: Solution of OH after regridding. Small artifacts from interpolation overshoot are present in the base algorithm, but suppressed using the ACR method in regions highlighted by the light gray bubbles.



(a) Chord **(b)** Chord with ACR

Figure 3.23: Solution of OH run 80 time steps after regridding. The originally small artifacts have developed into significant erroneous features in the flow. After this point the chemistry of the base algorithm is no longer numerically stable, while solution using the ACR method encounters no such issues.

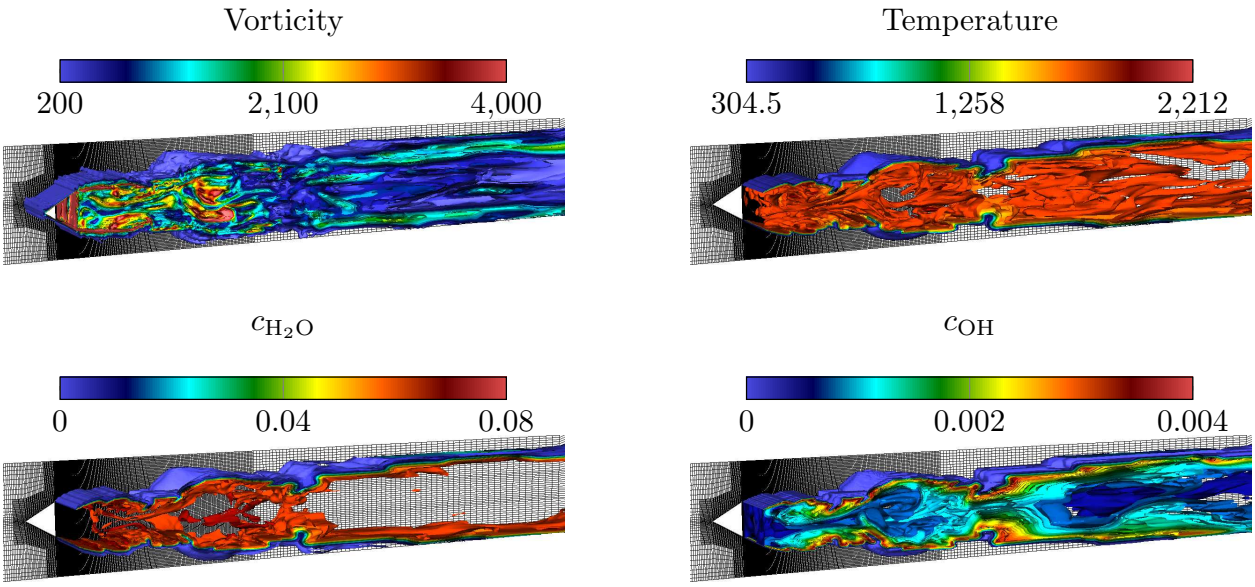


Figure 3.24: Vorticity, temperature, c_{H_2O}, and c_{OH} of the 3D Bluff-Body C_3H_8-air case at solution time $t = 246\,\text{ms}$.

Chapter 4

High-Order Mapped Multi-Block for Arbitrary Geometry

The focus of this chapter is to develop a method representing any arbitrary geometry by multiple mapped grid blocks while maintaining fourth-order FVMs with AMR. First, the starting point and background for previously developed methods are described. Then, the strategy for interpolating a grid mapping from discrete sources using B-splines to support AMR at high-order is described. The approach for the conforming mapped multi-block (MMB) methods are then extended to general geometries. The approaches are verified and validated for both mapped single-block and multi-block methods with AMR when grids are defined by discrete sources. Finally, the application of the generalized mapped multi-block method is demonstrated for practical problems.

4.1 Motivations

Prior to this work, mapped grids existed in Chord and Chombo but with restriction to smooth analytically-defined geometries. The focus here is to develop a method enabling the MMB method to work with any arbitrary geometry specified by a discrete grid, while maintaining fourth-order accuracy with AMR. We specifically aim to operate on discrete structured grids because they are the industry standard for mesh generation in CFD, and highly developed meshing tools [58] and expertise can be leveraged. Powerful analytic grid generation methods do exist [59] and have been shown to create grids for highly complex geometries that directly support AMR [60]. However, there are few standards for how to manage these analytic meshes. As a consequence, using one of these analytic grid generation methods requires either directly interfacing with existing software, or expert support. This is a significant long term investment, and requires learning a domain specific tool for creating grids.

The approach taken in this work starts by addressing the challenge of how to enable AMR at high-order accuracy on a mapped single-block grid defined by a discrete source. In particular, the standard approach to describe a grid is by storing a structured set of node locations, which are often generated from a geometry defined by NURBS curves. Connections between these nodes are known implicitly from the data layout, but generally geometry information between points is not provided. For low-ordered schemes, piecewise linear interpolation between these node locations is generally sufficient and no further geometry information is required. However, when using a high-ordered scheme with AMR, high-order interpolations of geometric information is required at potentially an location in order to compute the grid metrics described in Section 2.2.1. With the goal to produce this geometry information from a discrete source, what may seem a simple solution is to generate a grid and required metric quantities at the finest level of refinement available, and then coarsen the geometry information as needed for a given AMR level. This solution, although simple, is counter to the purpose of AMR since it requires prior knowledge of the maximum refinement, and stores excessive information. An alternative method to allow AMR from discrete grids is to interpolate the mapping function at points needed with sufficient accuracy. Prior approaches for block-based AMR on discrete grids have been made by Steinthorsson et at. [61] at second order accuracy, where a hybrid of splines for interpolation along coarse grid lines and Hermite interpolation for refinement off the coarse grid lines are used. Although effective, this method does not easily transfer to high-order and has continuity challenges between coarse and fine levels. The concept of using B-splines, or basis-splines, for interpolation of grid-based data is not a new idea, and has been studied by many, such as de Boor [62] and Piegl [63]. Further study of B-splines for a number of applications to CFD simulations, such as zonal grids and Galerkin methods, has been done by Karvchenko et at. [64] where B-splines are concluded to be highly effective for various forms of interpolation. Use of B-splines for interpolating grid mappings also has the benefit of being, or at least behaving similarly, how geometry is managed inside many mesh generation algorithms, and is likely to recreate the original geometry. This dissertation takes the approach

of multi-dimensional sixth-order B-spline to interpolate a grid mapping that allows for AMR in a fourth-order FVM.

Given the ability to operate on mapped grids of discrete origins, this work then extends to general MMB grids. The conforming multi-block scheme developed by McCorquodale et al. [10] manages block intersections with a smooth continuation of each block's mapping function which requires extrapolation. Although this method has been demonstrated to work well for chosen analytic mapping functions that avoid boundary conditions, its application to arbitrary complex geometries is unclear. To use the MMB method with discrete grids, each block needs to create an individual mapping function using the B-spline approach as in the single-block case. These B-spline mapping functions can then be used for extrapolation to neighboring blocks, although with some restrictions that are examined in this chapter. Communication between blocks comes with logistical challenges where inverse mappings are not known. This work proposes to overcome the lack of inverse mapping efficiently using targeted root finding strategies, in addition to development of an approach for communication between blocks in the presence of boundaries and sharp corners.

4.2 Technical Approach for Mapped Single-Block Grids

First, we describe the technical approach for mapped single-block grids from discrete sources, before extending to the multi-block scenario in the following section. The algorithm for mapped single-block grids in Section 2.2.1 assumes an analytic formulation of the mapping function, $\boldsymbol{x}(\boldsymbol{\xi})$, and its derivatives. Furthermore, the mapping function is assumed to be smooth, one-to-one, and invertible inside its defined domain. When using AMR, the mapping function is potentially required at any location in space. For use of AMR on a discrete grid, an interpolation method is needed to provide the mapping function at any arbitrary location inside the domain. When constructing a mapping function for a specified numerical scheme, the mapping function needs to appear smooth relative to the reconstruction order used by the numerical scheme. A sufficient interpolation of discrete data points should create a smooth function of chosen accuracy over the entire domain and exactly reproduce the known points. When interpolating from a discrete mesh,

the node locations are the only known geometry. It is important for an interpolation to construct these known point exactly, while elsewhere, because no geometry information is made available, we settle for any reasonable interpolation. Ideally this interpolation function will also have readily known derivatives, so the grid metrics can be calculated directly. Practically, this allows for the mapping function to be constructed using piecewise polynomials, so long as the construction is globally at least one order of accuracy higher than the underlying FVM. This dissertation targets fourth-order FVMs, and as a result, requires mapping functions which are at least fifth-order accurate.

An interpolating polynomial of order p, where the order specifies the number of terms in the polynomial, is in general order of accuracy p [62]. In piecewise polynomial interpolation, rather than polynomial order, the limiting factor for accuracy is connectivity between each segment. C^m connectivity, meaning that first m derivatives are continuous, is generally of accuracy $m+1$ for piecewise polynomial interpolation. The order of accuracy of a piecewise polynomial can be derived from the connectivity, as a polynomial of order p will have $p-1$ non-zero derivatives, and thus be up to C^{p-1}. For use in piecewise polynomial interpolation of grid metrics C^0 connectivity is required for a valid mapping, and C^1 for valid grid metrics. However, this is only second order accurate. For application to fourth-order algorithm with AMR, grid metrics must be at minimum fifth-order. Thus, the resulting piecewise polynomial interpolation requires quintic, or 6^{th} order, piecewise polynomials with C^4 connectivity. These piecewise polynomials with connectivity constraints are achieved with use of a B-spline [62]. For this application, B-spline interpolation can be treated as a globally 6-th order accurate multi-dimensional piecewise polynomial $P(\boldsymbol{\xi}) = \boldsymbol{x}$. This polynomial has four continuous derivatives, which are readily calculated as is required for the grid metric terms. An inverse of the interpolating polynomial, $P^{-1}(\boldsymbol{x}) = \boldsymbol{\xi}$, does exist but is not explicitly defined or easily solved.

4.2.1 B-Spline Interpolation

A method for creating polynomial fits with global accuracy properties is B-spline interpolation [62]. A B-spline is effectively a set of piecewise polynomials with the added constraint — continuity between curves is specified. This constraint allows for higher-order splines to be created in a memory efficient manner. However, solving the polynomial coefficients of the set of piecewise polynomials must be done simultaneously because they are coupled.

A B-spline [63] polynomial of finite span is constructed as

$$P(t) = \sum_{j=0}^{a-1} c_j B_{j,p}(t)\,, \tag{4.1}$$

where c_j is a control point in a vector of size a, and $B_{j,p}$ is the basis function of specified polynomial order, p. For interpolation over n points, a vector of known sequential points, $\boldsymbol{\tau}$, exists. Each element τ_i and corresponding $f(\tau_i)$ define an interpolation site, and are defined from $i = 0, 1, 2, \cdots, n-1$. The B-spline is created to fit these points by setting

$$P(\tau_i) = f(\tau_i) = \sum_{j=0}^{a-1} c_j B_{j,p}(\tau_i).$$

By solving for the control points that force the B-spline to smoothly pass though specified known points, a set of piecewise C^{p-2} connected curves are created. This is most simply solved if, for the spline $P(t)$, each piecewise segment $P_i(t)$ spans two interpolation sites, $[\tau_i, \tau_{i+1})$, and is accomplished by setting the knot vector $\boldsymbol{t}$ to include $\boldsymbol{\tau}$. This particular knot vector choice gives the ability to specify extra information, and explicitly define levels of connectivity at each interpolation site.

The basis function $B_{j,p}(t)$ is key in ensuring connectivity. The first index j is the knot number, and the index p is the basis function order. A knot vector $\boldsymbol{t} = [t_0, \cdots, t_i, \cdots t_{m-1}]$ must exist for each spline, where the number of knots $m \geq n$, where n is the number of interpolation points. The knot vector defines the break points at which polynomial segments are joined. In the definition of

a B-spline use herein, the basis functions are defined as

$$B_{j,1}(t) = \begin{cases} 1 & t_j \leq t < t_{j+1} \\ 0 & \text{otherwise} \end{cases}, \tag{4.2}$$

and for higher order basis functions

$$B_{j,p}(t) = \frac{t - t_j}{t_{j+p-1} - t_j} B_{j,p-1}(t) + \frac{t_{j+p} - t}{t_{j+p} - t_{j+1}} B_{j+1,p-1}(t). \tag{4.3}$$

In curve fitting it is useful to define derivatives, such as when solving grid metrics in this application. Derivatives of the basis functions are specified as

$$B'_{j,p}(t) = (p-1)\left(\frac{B_{j,p-1}(t)}{t_{j+p-1} - t_j} - \frac{B_{j+1,p-1}(t)}{t_{j+p} - t_{j+1}}\right).$$

Some manipulation of this definition allows for differentiation of a B-spline as [63]

$$P'(t) = (p-1)\sum_{j=1}^{a-1} \frac{c_j - c_{j-1}}{t_{j+p-1} - t_j} B_{j,p-1}(t). \tag{4.4}$$

This form can be extended to any number of derivatives k as

$$P^{(k)}(t) = \sum_{j=k}^{a-1} c_j^{(k+1)} B_{j,p-k}(t), \tag{4.5}$$

with

$$c_j^{(k+1)} = \begin{cases} c_j & \text{for } k = 0 \\ (p-k)\frac{c_j^{(k)} - c_{j-1}^{(k)}}{t_{j+p-k} - t_j} & \text{for } k > 0 \end{cases}.$$

The choice of knot vector is particularly important in creation of B-splines. All knot vectors must be defined in sequential increasing order, but allow for repeated knots. There are several implications of knot choice, however only the two most relevant will be covered here. First, for end point interpolation, the beginning and end of the knot vector must contain p repeated knots.

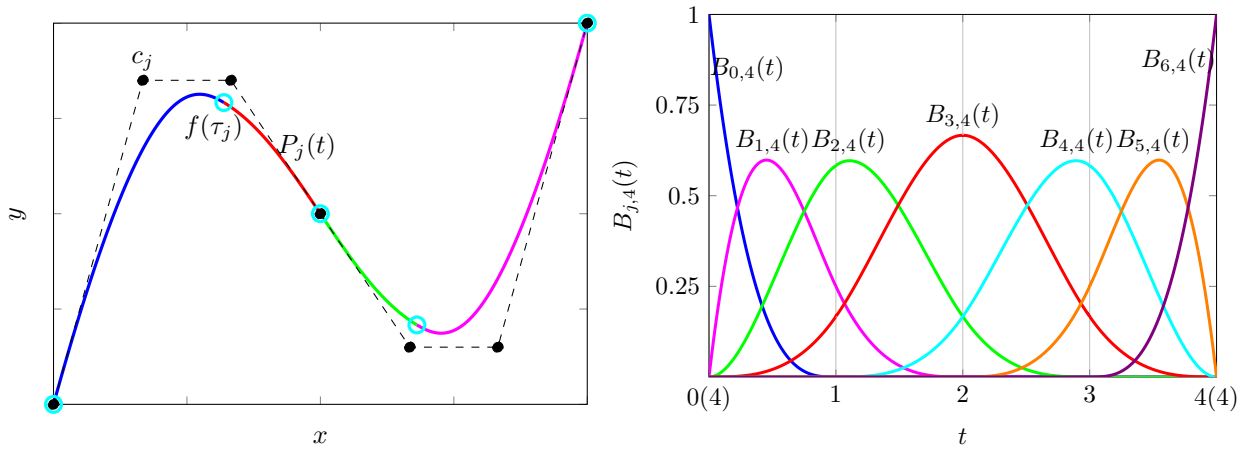


Figure 4.1: Illustration of a cubic B-spline(left), and the associated basis functions(right) for a specified knot vector $\boldsymbol{t} = [0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4]$. In the spline, each polynomial segment is shown by a different color and is connected to neighboring segments at the interpolation sites. The control points are visualized by black points, and seen to govern the curve. The basis functions to construct this curve are each shown, with break points occurring at knot indices.

The second important property deals with repeated internal knots. A B-spline usually has C^{p-2} connectivity, however this is only the case for uniform internal knots. A knot repeated r times has connectivity decreased to C^{p-2-r} at the knot. This is potentially useful in cases when some level of discontinuity is desired at a specific point.

For an interpolation spline, the definition of basis functions and knot vector provides a relation between the number of knots m, control points a, and order of spline p such that

$$m = a + p. \tag{4.6}$$

This also allows for a relation of a and m to the number of interpolation points n and total number of interior repeated knots r_t such that

$$m = n + 2(p - 1) + r_t, \tag{4.7}$$

and

$$a = n + p + r_t - 2. \tag{4.8}$$

By breaking the piecewise polynomial, Equation 4.1, into sections P_i over the interval $t \in [t_i, t_{i+1})$, the definition of basis functions allows for a reduced equation for a spline segment as

$$P_i(t) = \sum_{j=i-p}^{i} c_j B_{j,p}(t). \tag{4.9}$$

When solving for the control points, the previous Equation 4.9 becomes

$$P_i(\tau_i) = f(\tau_i) = \sum_{j=i-p}^{i} c_j B_{j,p}(\tau_i),$$

for each τ_i in $\boldsymbol{\tau}$. This results in a system of equations that can be conveniently represented in the form BC $=$ Y where B becomes a banded matrix of bandwidth $p-1$, C is the vector of control points, and Y is the vector of interpolation points.

Example of control point solution method

An example of the popular cubic B-spline demonstrates the solution method. Assuming n interpolation points and uniform internal knots, a total of $a = n + 2$ control points are needed, and the knot vector will be of size $m = n + 6$. Using $\boldsymbol{\tau}$ provides n constraints, so 2 additional constraints are needed. A simple way to generate 2 constraints is specifying end derivatives. The end control points can be specified by setting

$$f'(\tau_0) = (p-1) \sum_{j=i-p+1}^{i} \frac{c_j - c_{j-1}}{t_{j+p-1} - t_j} B_{j,p-1}(\tau_0).$$

At location τ_0, the corresponding knot is at t_3, and the basis function $B_{0,p-1}(\tau_0) = 1$ with all others being zero. This allows for

$$f'(\tau_0) = (p-1)\frac{c_1 - c_0}{t_p - t_1}.$$

Using the same process for τ_{n-1} and known end point interpolation gives control points

$$
\begin{aligned}
c_0 &= f(\tau_0), & c_{n+1} &= f(\tau_{n-1}),\\
c_1 &= c_0 + \frac{t_4 - t_1}{3} f'(\tau_0), & c_n &= c_{n+1} - \frac{t_{n+5} - t_{n+2}}{3} f'(\tau_{n-1}).
\end{aligned}
$$

The remaining interior points are solved in the form of BC $=$ R $-$ D, which becomes

$$
\begin{bmatrix}
B_2(\tau_1) & B_3(\tau_1) & 0 & \cdots & 0 & 0 & 0\\
B_2(\tau_2) & B_3(\tau_2) & B_4(\tau_2) & \cdots & 0 & 0 & 0\\
0 & \ddots & & \ddots & & \ddots & 0\\
0 & 0 & 0 & \cdots & B_{n-3}(\tau_{n-3}) & B_{n-2}(\tau_{n-3}) & B_{n-1}(\tau_{n-3})\\
0 & 0 & 0 & \cdots & 0 & B_{n-2}(\tau_{n-2}) & B_{n-1}(\tau_{n-2})
\end{bmatrix}
\begin{bmatrix} c_2\\ c_3\\ \vdots\\ c_{n-2}\\ c_{n-1} \end{bmatrix} =
$$

$$
\begin{bmatrix} f(\tau_1)\\ f(\tau_2)\\ \vdots\\ f(\tau_{n-3})\\ f(\tau_{n-2}) \end{bmatrix} -
\begin{bmatrix} c_1 B_1(\tau_1)\\ 0\\ \vdots\\ 0\\ c_n B_n(\tau_{n-2}) \end{bmatrix},
$$

where all basis function are of order $p = 4$, which is dropped for convenience. The end derivatives $f'(\tau_0)$ and $f'(\tau_{n-1})$ are typically unknown, but one-sided finite difference approximations are sufficient so long as they meet the spline order of accuracy.

The methodology for constructing cubic B-splines is easily extended to any order. For any number of interpolation points, n, Equation 4.7 for a smooth curve requires $m = n + 2(p-1)$ knots and Equation 4.8 yields $a = n + p - 2$ control points. Solving the a unknowns is done by using the n interpolation point constraints, and requires an additional $p - 2$ extra constraints. Extra constraints are commonly specified boundary conditions, such as the prior example of cubic splines with fixed end derivatives. For symmetry of boundary conditions, splines of even order are preferred. Extending from cubic to quintic spline interpolation is done by addition of two boundary

conditions. The quintic spline with fixed first and second end derivatives adjusts coefficients to

$$
\begin{aligned}
c_0 &= f(\tau_0), & c_{n+3} &= f(\tau_{n-1}),\\
c_1 &= c_0 + \frac{t_6 - t_1}{5} f'(\tau_0), & c_{n+2} &= c_{n+3} - \frac{t_{n+9} - t_{n+4}}{5} f'(\tau_{n-1}),
\end{aligned}
$$

and adds

$$
c_2 = (t_7 - t_2)\left[\frac{(t_6 - t_2)}{20} f''(\tau_0) - \frac{(t_7 + t_6 - t_2 - t_1)c_1}{(t_6 - t_1)(t_7 - t_2)} + \frac{c_0}{t_6 - t_1}\right],
$$

$$
\begin{aligned}
c_{n+1} = (t_{n+8} - t_{n+3})\Bigg[&\frac{(t_{n+9} - t_{n+4})}{20} f''(\tau_{n-1}) - \\
&\frac{(t_{n+9} + t_{n+8} - t_{n+4} - t_{n+3})c_{n+2}}{(t_{n+8} - t_{n+3})(t_{n+9} - t_{n+4})} + \frac{c_{n+3}}{t_{n+8} - t_{n+3}}\Bigg].
\end{aligned}
$$

In the prior solution for interior control points BC = R − D, the B matrix is now expanded using basis function of order $p = 6$ becoming penta-diagonal rather than tri-diagonal. Since $a = n + 4$, all elements grow to account for the control points. Additionally, the vector D now accounts for extra boundary conditions and becomes

$$
\mathrm{D} = \begin{bmatrix}
c_1 B_1(\tau_1) + c_2 B_2(\tau_1) \\
c_2 B_2(\tau_2) \\
0 \\
\vdots \\
0 \\
c_{n+2} B_{n+2}(\tau_{n-3}) \\
c_{n+2} B_{n+2}(\tau_{n-2}) + c_{n+3} B_{n+3}(\tau_{n-2})
\end{bmatrix}.
$$

Uniform Knot Vectors

B-splines for uniform interior knots, also known as cardinal splines, allow for some vast simplification [62]. Cardinal splines have knot vectors with constant spacing Δt between each element,

except for the end repeated knots, which allows for simpler form of the end conditions such that for quintic splines,

$$
\begin{aligned}
c_0 &= f(\tau_0), & c_{n+3} &= f(\tau_{n-1}),\\
c_1 &= c_0 + \frac{\Delta t}{5} f'(\tau_0), & c_{n+2} &= c_{n+2} - \frac{\Delta t}{5} f'(\tau_{n-1}),\\
c_2 &= (2\Delta t)\left[\frac{\Delta t}{20} f''(\tau_0) - \frac{3c_1}{2\Delta t} + \frac{c_0}{\Delta t}\right], & c_{n+1} &= (2\Delta t)\left[\frac{\Delta t}{20} f''(\tau_{n-1}) - \frac{3c_{n+2}}{2\Delta t} + \frac{c_{n+3}}{\Delta t}\right].
\end{aligned}
$$

This scaling only must be done when calculating the end points and evaluating x values for given t values.

A benefit of using a uniform knot vector is that basis functions are simplified. If each knot is uniformly spaced, then there will only be a single unique set of basis functions, and all others are translates of that set. The uniform basis functions $M_{j,d}(t)$ is defined as

$$
M_{j,1}(t) = B_{j,1}(t) = \begin{cases} 1 & t_j \le t < t_{j+1} \\ 0 & \text{otherwise} \end{cases},
$$

and the recurrence relation defined previously is reduced to

$$
M_{j,p}(t) = \frac{t-j}{\Delta t(p-1)} M_{j,p-1}(t) + \left(1 - \frac{t-j}{\Delta t(p-1)}\right) M_{j+1,p-1}(t). \tag{4.10}
$$

As an example using this relation, the uniform basis functions for cubic order with unit spacing are given as

$$
M_{j,4}(t) = \frac{1}{6} \begin{cases} (t-t_j)^3 & t_j \le t < t_{j+1} \\ -3(t-t_j)^3 + 12(t-t_j)^2 - 12(t-t_j) + 4 & t_{j+1} \le t < t_{j+2} \\ 3(t-t_j)^3 - 24(t-t_j)^2 + 60(t-t_j) - 44) & t_{j+2} \le t < t_{j+3} \\ (3 + t_j - t)^3 & t_{j+3} \le t < t_{j+4} \\ 0 & \text{otherwise} \end{cases}.
$$

This clearly is simpler to calculate than the traditional recurrence relation basis function, and since only one unique set exists it grants some computational savings. For this reason, this work is restricted to use of cardinal splines with knots located at the interpolation points.

4.2.2 Tensor Product B-Splines

Construction of multi-dimensional splines, such as needed for any grid, is done with a tensor product of one-dimensional (1D) splines [63]. Extending upon Equation 4.1 for creation of 1D splines, 2D splines are created as a product of two directional 1D splines by

$$P(t,s) = \sum_{i=0}^{n_t} \sum_{j=0}^{n_s} c_{i,j} B_{i,p}(t) B_{j,p}(s), \tag{4.11}$$

where **s**, similar to $\boldsymbol{t}$, is a second knot vector. This is broken into solving two 1D problems as

$$P(t,s) = \sum_{i=0}^{n_t} \mathbf{L}_i(s) B_{i,p}(t),$$
$$\mathbf{L}_i(s) = \sum_{j=0}^{n_s} c_{i,j} B_{j,p}(s),$$

where $\mathbf{L}(s)$ is a vector of independent piecewise polynomials, and the subscript indicates the piecewise polynomial set. Using the control curves, a set of control points is chosen at some s_j and used to create B-splines of $\boldsymbol{t}$ that create the surface. These are evaluated at a chosen value of s_j to give p points. Using these points, another spline is fit in the $\boldsymbol{t}$ direction and evaluated at a chosen t_i point, yielding a point on the surface. This process is direction independent so $\boldsymbol{t}$ and **s** may be evaluated in any order. In this representation, **c** is a two-dimensional set of control points. To evaluate a particular point $P(t,s)$, the control points **c** are sliced along the j direction and the set of control curves $\mathbf{L}(\mathbf{s})$ created. Each curve $\mathbf{L}_i(\mathbf{s})$ follows **s** along the row where i is constant. Now that all curves in the j direction exist, curves in the i direction are created. The control curves are evaluated at any point s resulting in a set of secondary control points $\mathbf{L}(s)$ in the i direction, from which is created a B-spline following $\boldsymbol{t}$. A single point t can then be evaluated on the surface. Stretching

the evaluation of s and t over the whole domain creates the final surface $P(\boldsymbol{t}, \mathbf{s})$, with this process illustrated in Figure 4.2. This process is direction independent so $\boldsymbol{t}$ and $\mathbf{s}$ may be evaluated in any order.

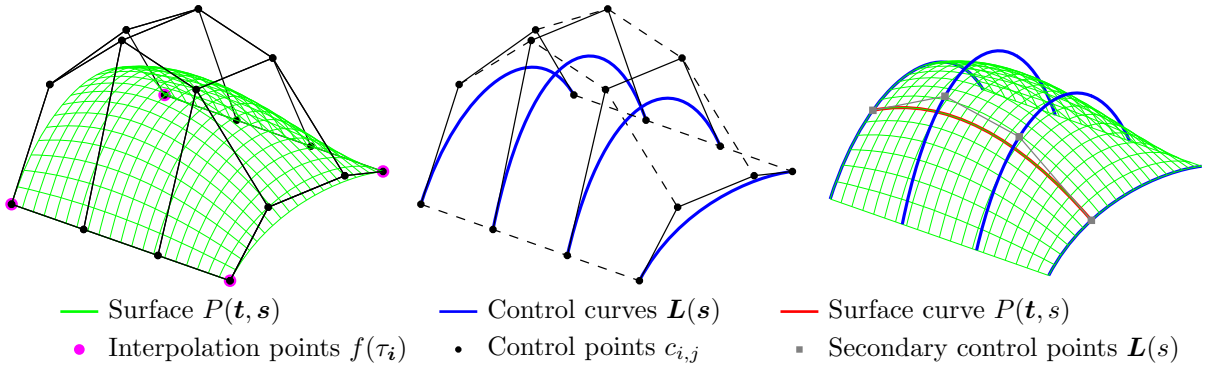


Figure 4.2: Illustration of tensor spline creation.

Further expansion to 3D follows a similar procedure, were a number of 1D B-splines are solved and a tensor product is performed to create a higher dimensional structure. This becomes

$$P(t,s,r) = \sum_{i=0}^{n_t} \sum_{j=0}^{n_s} \sum_{k=0}^{n_r} c_{i,j,k} B_{i,p}(t) B_{j,p}(s) B_{k,p}(r), \tag{4.12}$$

where $\mathbf{r}$ is another knot vector. This can once again be split into a set of 1D problems as

$$\begin{aligned} P(t,s,r) &= \sum_{i=0}^{n_t} \mathbf{L}_i(s,r) B_{i,p}(t), \\ \mathbf{L}_i(s,r) &= \sum_{j=0}^{n_s} \mathbf{K}_{i,j}(r) B_{j,p}(s), \\ \mathbf{K}_{i,j}(r) &= \sum_{k=0}^{n_r} c_{i,j,k} B_{k,p}(r). \end{aligned}$$

In order to use a tensor product B-spline to interpolate $f(\tau_{\boldsymbol{i}})$, the control net of points must be solved for. This is done by reducing a multi-dimensional problem to a set of 1D problems. By fixing one direction, a set of 1D B-splines is fit to interpolation points, and the control points for each of these 1D splines is found. This set of control points is only temporary, as the fixed

direction is now shifted and the previous control points are now used as interpolation sites. Cycling through each direction, a new set of 1D splines is fit through the previously existing control points, recursing through all dimensions with the final result a tensor of control points. This method is direction-independent and can be extended to any number of higher dimensions as needed.

B-splines, and specifically cardinal splines, are well suited for interpolating grid mappings. Using the tensor product of B-splines allows for a concise procedure for multi-dimensional interpolation. For multi-dimensional problems, this procedure is reduced to a set of single-dimensional problems, and thus no significant complexity is added. Due to good scaling in both the number of dimensions and the order of accuracy, B-splines for high-order grid mappings perform well.

4.3 Technical Approach for Mapped Multi-Block Grids

Now, we extend the approach to mapped multi-block grids. The major challenge of the MMB method, as described in Section 2.2.2, is determining how to compute the fluxes on multi-block faces. For high-order methods, an additional challenge is introduced due to the large stencils used. For example, fluxes at interior cell-faces near a block boundary may have a stencil that extends beyond the block-domain. McCorquodale et al. [10] reduces the problem of solving stencils over multi-block boundaries to one of performing high-order interpolations. Employing MMB ghost cells, artificial cells which are extended outside the block-domains, allows the centered scheme to be applied everywhere. For the mapped multi-block method, MMB ghost cells from each block are extended over block interfaces. Their values are then interpolated from the existing valid regions. We further define MMB ghost cells as either interior or exterior. Interior ghost cells are those whose cell center lies inside the global physical domain, while exterior ghost cells have a cell center that falls outside the global physical domain. To emphasize, all cells except the ghost cells are referred to as valid cells, i.e. those cells being part of the global domain. This concept is illustrated in Figure 4.3, where each block is shown in different colors and given black boundaries. The solid lines of each block show the valid cells, and dashed lines are for MMB ghost cells. The

enhanced blue ghost cell in the top left is an interior ghost cell, while the enhanced red ghost cell in the bottom right is an exterior ghost cell.

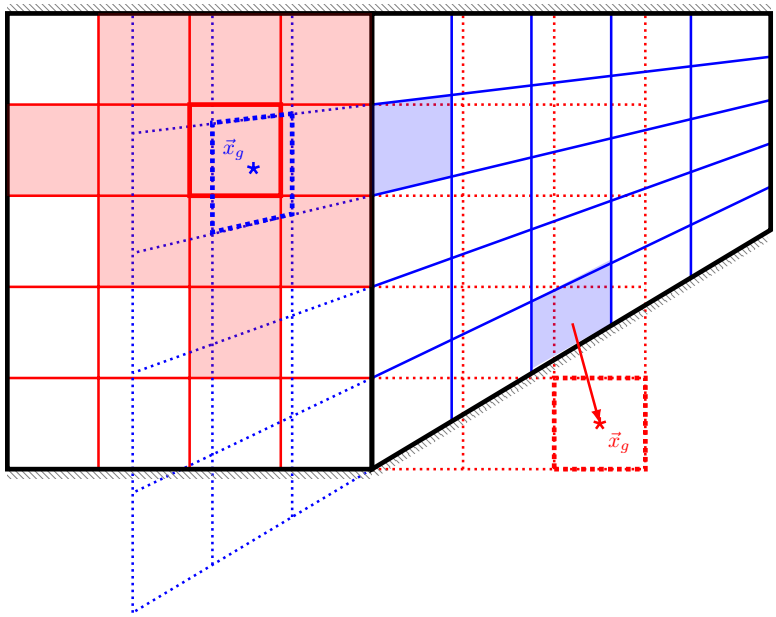


Figure 4.3: A mapped multi-block grid with two blocks is shown in physical space, one in red and the other in blue. For each colored grid, solid lines show the valid cells and dashed lines the ghost cells. The emphasized blue ghost cell is an interior ghost cell, which is filled by interpolation from the surrounding highlighted neighborhood of cells. The emphasized red ghost cell in the bottom right is an exterior ghost cell. This exterior ghost cell value is extrapolated as a constant from the indicated nearest highlighted valid cell.

Interior ghost cells are interpolated using a high-order polynomial, as has been previously developed [10]. For cases with exterior ghost cells, such as the red ghost cell seen in Figure 4.3, interpolation is not possible, so an extrapolation method must be devised instead. Logistical concerns about how to identify where ghost cells should interpolate values from arise when dealing with arbitrary grids and require care to address efficiently. Finally, considerations must be made for restrictions of the method, being that the mapping function must be of sufficient quality to interpolate ghost cells. These challenges must be addressed for the algorithm to operate on general geometries with an acceptable computational cost and are discussed in the subsequent sections.

4.3.1 Ghost Cell Interpolation

The approach for filling interior ghost cells is accomplished by interpolating a fourth-order multi-dimensional polynomial from valid solution data using the least squares method, as detailed in Section 2.3.3. For each ghost cell $\boldsymbol{g}$, the interpolation is centered about the physical location of the ghost cell center $\boldsymbol{x_g}$, as labeled in the enhanced blue cell in Figure 4.3. A region of valid neighboring cells, $\mathcal{I}(\boldsymbol{g})$, is established that is sufficiently sized to produce a fourth-order polynomial, as is illustrated by the highlighted cells surrounding the blue $\boldsymbol{x_g}$ in Figure 4.3. In Chord, where fourth-order reconstruction is used in either two or three dimensions, the resulting neighborhoods must contain at minimum 10 or 20 cells, respectively. This choice of interpolation region is detailed by McCorquodale et al. [10], and in general spans multiple blocks while having the smallest footprint reasonable. The interpolating polynomial $\mathbf{U}_{d,\boldsymbol{g}}(\boldsymbol{x})$ for each solution component d of cell $\boldsymbol{g}$ is constructed using the form in Equation 2.13

$$\mathbf{U}_{d,\boldsymbol{g}}(\boldsymbol{x}) = \sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} \left(\boldsymbol{x} - \boldsymbol{x_g}\right)^{\mathbf{q}} + O(h^Q)\,, \tag{4.13}$$

where to achieve fourth-order accuracy in Chord, $Q = 4$ is used. The coefficients $c_{\mathbf{q}}$ for each cell are solved by the least squares method to fit the neighborhood $(\langle \mathbf{U}_d \rangle_{\boldsymbol{i}} : \boldsymbol{i} \in \mathcal{I}(\boldsymbol{g}))$. Once the coefficients are solved, the ghost cell is then interpolated as

$$\langle \mathbf{U}_d \rangle_{\boldsymbol{g}} = \frac{1}{\mathcal{V}_{\boldsymbol{g}}} \sum_{\|\mathbf{q}\|_1 < Q} c_{\mathbf{q}} m_{\boldsymbol{g}}^{\mathbf{q}}(\boldsymbol{x_g})\,. \tag{4.14}$$

Notably, there are no feasible ways to enforce conservation constraints in this approach for ghost cell construction. However, this does not impact the solution conservation because ghost cells are only intermediate data used for flux reconstruction and each face flux is still single valued by restriction to conforming multi-block grids.

Exterior Ghost Cell Extrapolation

One of the challenges for general mapped multi-block grids is that ghost cells fall outside the global physical domain, as shown by the red emphasized cell in Figure 4.3. For such cells an interpolation can not be well-defined. Instead, an extrapolation scheme is developed to fill these ghost cells with reasonable values. These values are then used to reconstruct fluxes needed.

As the first step, the multi-block algorithm must detect the exterior ghost cells. Determining whether a ghost cell is exterior is not straightforward and can be computationally expensive. This is because the multi-block boundaries are generally complex and the computational geometry algorithm using high-order B-splines is expensive to evaluate. As part of the multi-block setup, the physical location of the center of each ghost cell, $\boldsymbol{x_g}$, must be solved through its inverse mapping $\boldsymbol{x}(\boldsymbol{\xi_g})$ to find the corresponding valid cell. Since the mapping and its inverse must be one-to-one, exterior ghost cells can then be detected if an inverse fails to exist inside the global domain.

Once detected, external ghost cell extrapolation is performed. Ideally, the extrapolation should be smoothly extended from the interior. The simplest solution is to perform a constant extrapolation along the grid lines to the exterior ghost cells. This is a low-order method, but the extrapolation can be reasonable and stable. However, the extrapolation can be improved by taking values from the physically nearest cell, that is the cell $\boldsymbol{i}$ such that $\arg\min_{\boldsymbol{i}} \|\boldsymbol{x_i} - \boldsymbol{x_g}\|$, which intuitively is more reasonable while still being stable. Inspecting Figure 4.3, one can see this from the enhanced red ghost cell (i.e., an exterior ghost cell) in the bottom right. The exterior ghost cell is extrapolated as a constant from the indicated nearest valid cell. This approach has been observed to yield reasonable results in the present study.

4.3.2 Inverse Mapping

One computationally challenging piece of this algorithm is finding the nearest valid cell to a given location in physical space $\boldsymbol{x_g}$. This is required for creating the interpolation neighborhood $\mathcal{I}(\boldsymbol{g})$ required for filling each ghost cell. One such example stencil is shown in Figure 4.3 by the shaded cells. With a high-order scheme that uses many ghost cells, the number of total ghost cells

can frequently rival or even exceed the number of valid cells in the global domain. Although this stencil creation and search for the nearest valid cell must happen for every ghost cell, it needs only happen once per mesh. However, if implemented poorly, this one-time stencil creation algorithm can still be prohibitively expensive.

This challenge is addressed by formulating a root finding problem, where the point $\boldsymbol{\xi}_{\text{inv}}$ interior to the domain is desired to satisfy $f(\boldsymbol{\xi}_{\text{inv}}) = 0$. For a given point $\boldsymbol{x_g}$, the distance function to satisfy is

$$f(\boldsymbol{\xi}) = \|\boldsymbol{x}(\boldsymbol{\xi}) - \boldsymbol{x_g}\| .$$

Since solving the function $\boldsymbol{x}^{-1}(\boldsymbol{\xi})$ directly is infeasible, a root solving method is required. Newton's method may be used to solve the root finding problem efficiently, since the mapping function is represented by B-splines which have readily computed derivatives. However, Newton's method requires a sufficient initial estimate in order to converge to the root. Quantifying a sufficient initial estimate is challenging in general. Empirically, we find that Newton's method will converge to the correct root when the initial estimate $\boldsymbol{\xi}_{\text{init}}$ is of distance $\|\boldsymbol{\xi}_{\text{init}} - \boldsymbol{\xi}_{\text{inv}}\| < h$ on the coarsest grid.

To reach a region where iterative root solvers such as Newton's method can converge to the correct solution, a global root solver must first be used to arrive at a coarse approximation $\boldsymbol{\xi}_{\text{init}}$. However, global root solvers are generally expensive to evaluate. In this case, using a global root solver requires additional care because the global domain may be quite large, and a root solve must be performed for every ghost cell. We develop an optimized approach for the global root solver specifically for this scenario.

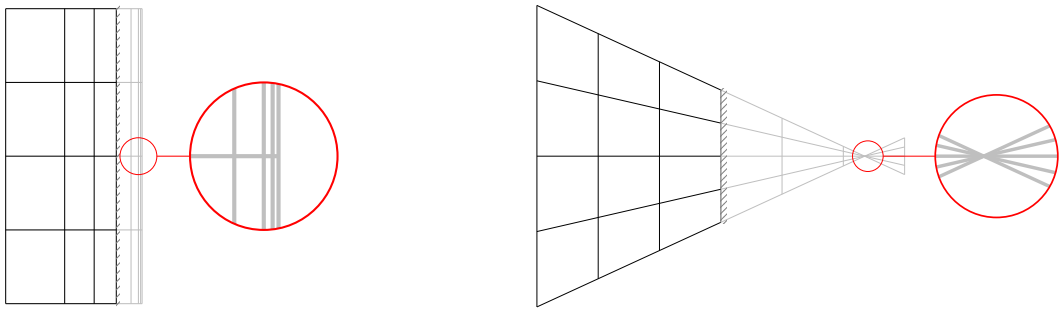
For the global root solve, each block has a cache of coarse nodes built into a KD-tree using the nanoflann library [65]. A KD-tree is in essence a multi-dimensional extension of a binary tree specially designed for efficiently partitioning regions in space. Of particular interest, a KD-tree has construction time of $O(DN\log(N))$ and average nearest neighbor lookup time of $O(\log(N))$, where N is the number of coarse nodes in a given block. A coarse approximation for the global root of any given point is found by searching for the nearest neighbor of each block following a breadth first search of block connectivity. Given the nearest neighbor node, the point $\boldsymbol{x_g}$ can

quickly be determined to lie inside the block based upon its proximity to boundary and exterior nodes. The case where the nearest node is further than two nodes away from a boundary can be immediately determined as inside or outside a block. When the nearest node is on or adjacent to a boundary, an inverse existing inside the block is questionable and requires solving the exact root using Newtons method. Points exterior to each block are stored as potential candidates for exterior ghost cells if no interior ghost cell is found. A point interior to a block clearly is near the global inverse, which is then used as the starting location for a Newton solve to converge to the true global inverse. In practice, the use of a KD-tree has insignificant cost relative to many other parts of the grid initialization algorithm. Additionally, there are other geometry operations of the FVM which are able to leverage the KD-tree to reduce computational expense.

4.3.3 Mapping Restrictions Imposed by Ghost Cells

Ghost cells are created by extending each block-domain with a required number of cells that enables the centered scheme (e.g., centered stencil) to evaluate the block-boundary fluxes. An important consideration in this process is how the mapping function behaves when extended beyond block-domains. The mapping in ghost cells is assumed to still hold its prescribed shape and properties — one-to-one, smooth, and invertible. However, this assumption can be problematic for a number of mappings. For example, as illustrated in Figure 4.4, extending the original block-domain to ghost cells results in a region with degenerated grids. This is further exacerbated for grids with a high-stretching ratio. The figure shows a continuation of the mapping may result in points where grid lines cross and the mapping is no longer one-to-one. When this situation occurs, the discrete grid needs to be recreated with the guidance provided by the mapped multi-block algorithm, including the adjustment to the amount of stretching near block boundaries or the refinement to the grids near boundaries to improve the ghost cell quality. As cell size shrinks, the distance of ghost cells beyond the block-domains also reduces.

For discrete grids, the mapping function is represented using B-splines defined within each block-domain. This introduces another challenge. Extrapolation, as required of the mapping func-



(a) Contracting grid.

(b) Grid crossing.

Figure 4.4: Visualization of examples where ghost cells degenerate.

tion to create ghost cells, is known to be error-prone for polynomials. Fortunately, the extrapolation region is always known to be small, and so long as the mapping remains one-to-one the error is inconsequential. However, in cases where the mapping degenerates due to extrapolation errors from the B-splines, it is again left to the researcher to alter the grid.

4.3.4 Multi-block Domains with Adaptive Mesh Refinement

Enabling the AMR capability for the multi-block domains adds significant logistical challenges. For AMR regions, ghost cells are used to communicate data between levels and allow each level to be updated independently. In the context of multi-block grids, ghost cells can be divided into in two categories, invalid and MMB ghost. Invalid ghost are cells interpolated from a coarser level and MMB ghost cells are those which are interpolated on a single level over a MMB boundary. On a grid with a single level, the MMB ghost cells are interpolated using the method presented in Section 4.3. Invalid ghost cells are interpolated onto level $\ell + 1$ from level ℓ using a conservative high-order multi-dimensional interpolation. The addition of AMR adds complication to filling ghost cells on finer levels. On fine levels, grids typically do not span the entire domain with AMR. The MMB ghost cell interpolation scheme creates stencils assuming the entire domain is available and caches those stencils for efficiency. This is problematic when filling MMB ghost cells, since it either requires regenerating stencils every time the grid adapts, which is expensive, or dealing with incomplete stencils. The approach taken in this work is to deal with incomplete stencils, by using MMB stencils where possible and stencils that interpolate from invalid ghost

cells elsewhere. Figure 4.5 shows an example of this, with regions able fill ghost cell by MMB interpolation in green and regions requiring invalid ghost cell interpolation in gray.

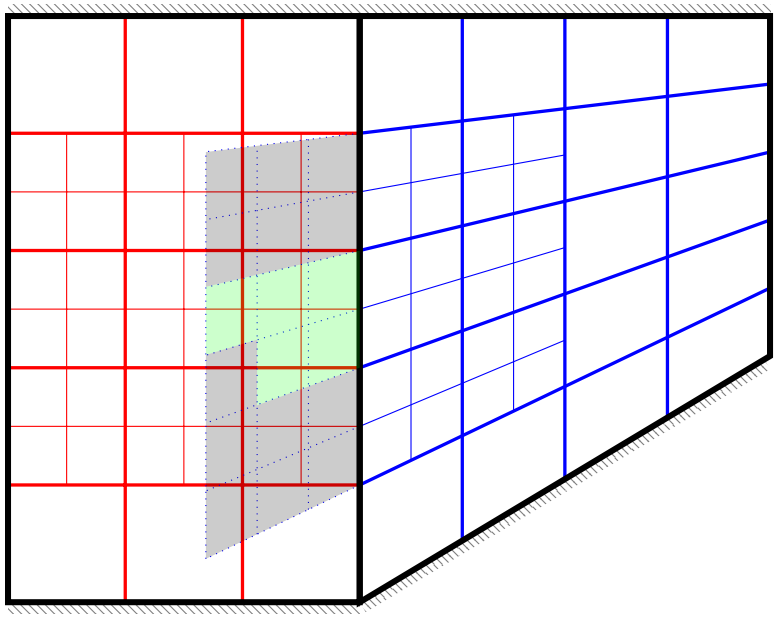


Figure 4.5: A mapped multi-block grid with two blocks is shown in physical space, one in red and the other in blue. A region of AMR spans the block boundary, with a refinement factor of two. Ghost cells of the blue grid are shown with blue dashed lines. The green highlighted ghost cells are those with complete stencils on the fine level, which are interpolated over the MMB boundary from other fine cells on the level. The gray highlighted ghost cells are those with incomplete stencils, which are interpolated up using the coarser levels ghost cells.

To simplify the process for filling both invalid and MMB ghost cells, the interpolation of all ghost cells is coupled and described by the following procedure:

1. Assume the coarse level ℓ has all ghost cells filled. In a correct AMR hierarchy, all ghost cells of the fine level $\ell+1$ are contained within the coarse level ghost cells and have sufficient coarse cells for all required stencil operations.

2. Interpolate from the coarse to fine level for all ghost cells as though they are invalid. This happens on each block individually with no knowledge of MMB interfaces. This fills all ghost cells, of which only the invalid are correct while the MMB ghost cells are inaccurate but of reasonable value.

3. Do a multi-block interpolation for all ghost cells on the fine level that have a complete stencil on the current level. On all but the coarsest level, it is possible for MMB ghost cells to have incomplete MMB stencils on their given level due to the variability in mesh refinement. When stencils can not be fully satisfied on an AMR level, the previously computed invalid ghost cell interpolation is used. Although not ideal, this is still a sufficient interpolation over MMB boundaries.

4.4 Verification and Validation

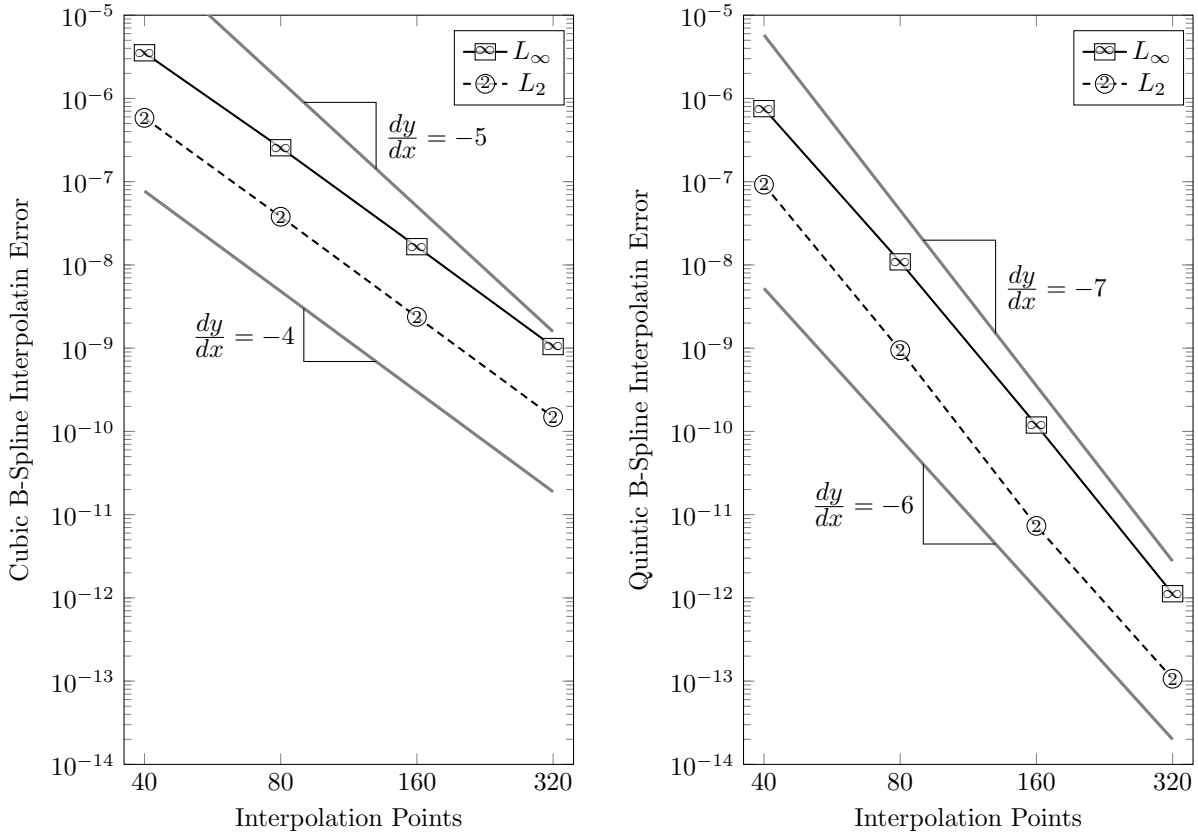
We have implemented the MMB techniques described in this chapter into Chord. To verify the MMB algorithm on discrete grids, we consider three cases. First, the B-spline interpolation is verified to show the desired global accuracy. Second, a periodic advection cube with AMR is used to verify the B-spline mapping functions. Third, the periodic advection cube with AMR is again used for verification of a MMB grid coming from a discrete source. With the periodic advection cube, we further verify that the MMB algorithm maintains conservation and achieves the desired order of accuracy over multi-block boundaries.

4.4.1 B-Spline Order of Accuracy

One of the main challenges for the development of a fourth-order AMR algorithm on arbitrary mapped grids is maintaining the order of accuracy. In the development of B-spline theory, they are claimed to have global order of accuracy p, and validation of this takes place.

Implementation of 1D cubic and quintic splines can be validated against any smooth function, ideally of reasonable complexity. Convergence tests are performed using a sample function, which happens to be a viscosity-temperature relation, with uniform interpolation points and results shown in Figure 4.6. This work specially makes use of cardinal splines, that is the knot locations coincide with interpolation points.

Results of this convergence test show the expected order of accuracy for both the cubic and quintic splines is observed. The cubic spline maintains a convergence rate of 4.0. The quintic spline



(a) Cubic spline.

(b) Quintic spline.

Figure 4.6: Convergence rate of interpolation B-splines for the function $f(x) = \exp(0.746 \log x + \frac{43.555}{x} - \frac{3259.934}{x^2} + 0.13556)$ with the number of interpolation points uniformly spaced over $x \in [300, 700]$.

exhibits a convergence rate between the 6.0 and 7.0. This verifies 1D B-spline implementation, and provides confidence for multi-dimensional B-splines since they simply apply recursion to one-dimensional problems, and thus are suitable for high-order grid interpolation.

4.4.2 Advection on a Mapped Single-Block Grid with AMR

The Gaussian advection case is chosen as a test for validation of geometry implementation, as seen by Guzik et al. [8]. Using the Euler equations on a mapped grid with periodic boundary conditions, a Gaussian density bump is initialized with a specified flow velocity. The density profile is initialized as

$$\rho = \rho_0 + s(r)\,\Delta\rho\,\mathrm{e}^{-(100r^2)}, \tag{4.15}$$

where

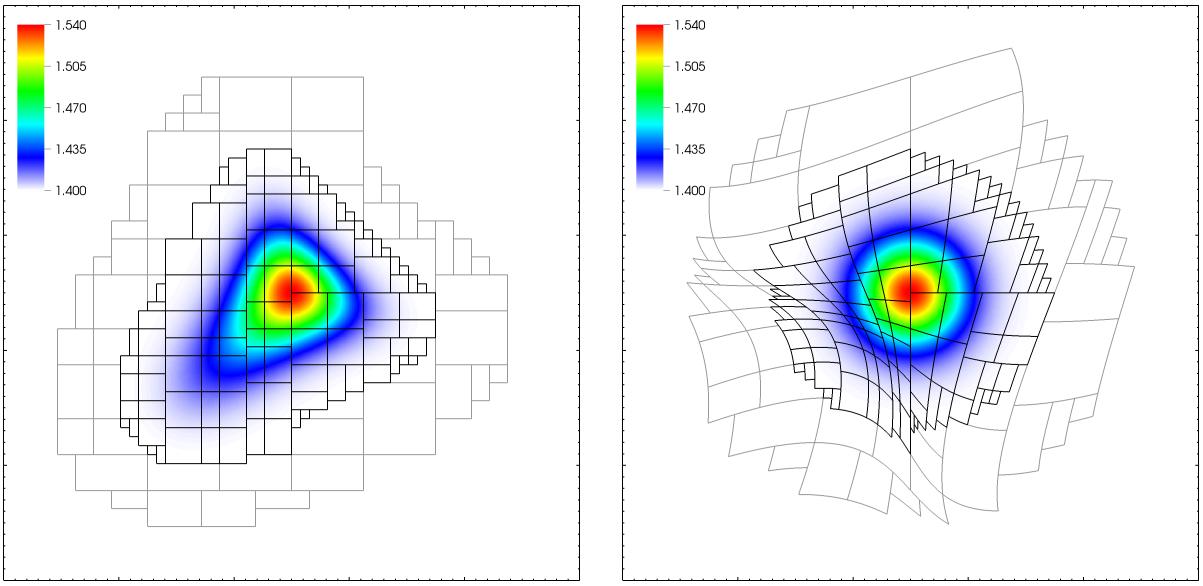
$$\begin{cases} 0 & : \quad |2r| \geq 1 \\ \cos^6(\pi r) & : \quad |2r| < 1 \end{cases}, \tag{4.16}$$

$\rho_0 = 1.4$, and $\Delta\rho = 0.14$. The value r specifies the distance from the center of the periodic domain, $[0,1]^D$. Pressure is initialized to a constant of 1 and the velocity is set to $(1.0, 0.5)$. With this choice of velocity, the Gaussian profile will return to the initial location after 2 time periods, and solution error is calculated using the exact solution. The mapping is defined by

$$x_d = \xi_d + c_d \prod_{p=1}^{D} \sin(2\pi\xi_p) \qquad d = 1, 2\,. \tag{4.17}$$

Initialization of this case is shown in Figure 4.7 in both computational and physical space. Boxes, a collection of cells, are shown to describe the mesh with two levels of AMR applied. The refinement is set analytically such that the first level is an approximate circle of radius 0.35 from the center of the Gaussian bump, and the second level is of radius 0.225 from the center. This is chosen such that, when performing a convergence test, the refinement regions cover nearly the same area regardless of the base grid.

A convergence study is performed to analyze the effectiveness of using (cardinal) splines to recover geometry representation. Using the mapping described, two sets of convergence test are run, with the first using the analytically defined geometry, and the second using a discretized form. Quintic spline interpolation is applied to the discretized geometry and used to provide grid information as needed. Using the analytic and interpolated geometry, a convergence test is run both with and without AMR applied. The cases with AMR have two layers of refinement by 2, where, when compared, the most refined grid matches the base grid of the single level cases. Ie., a 64×64 grid with two levels of AMR has a refined region matching the single level 256×256 grid. All single level convergence rates are measured at 4.0 as expected. The addition of AMR does not significantly reduce convergence rates, as they are measured between 3.78 and 4.0 independent



(a) Computational space.

(b) Physical space.

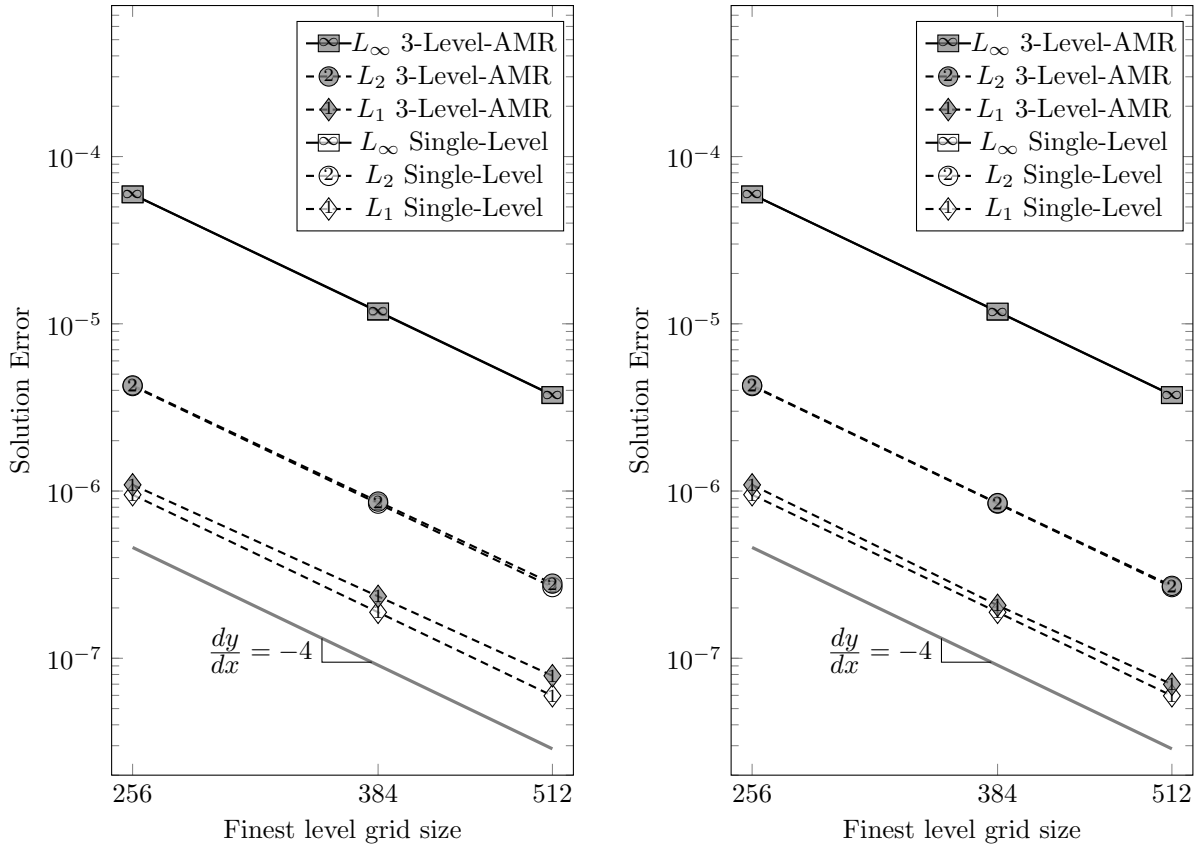
Figure 4.7: Advection case initial Gaussian density profile. Boxes show the grid with gray the first level of refinement and black the second.

of mapping. It is expected that the discontinuous grid created by AMR can induce a reduction in accuracy by up to one order.

Additionally, the Gaussian advection case with periodic boundaries allows for testing conservation. A finite volume scheme with no source or sink terms does not change mass, which is tested by integration of the conservative variables in computational space $J\mathbf{U}$. This is computed for initialized and completed solutions of the 512×512 AMR case, using both the analytic and interpolated mappings. These results and the difference are tabulated in Table 4.1 and Table 4.2, and it is seen that conservation is preserved to machine precision. In this test, the difference in conservative quantities over the run time of the advection case is on the order of machine zero, and thus the algorithm works as expected.

Table 4.1: Initial and final conservative values for the advection case, using the analytic mapping

$J\mathbf{U}$	Initial	Final	Difference
$J\rho$	$2.299\,286\,119\,588\,755 \cdot 10^4$	$2.299\,286\,119\,588\,766 \cdot 10^4$	$0.000\,000\,000\,000\,011 \cdot 10^4$
$J\rho u$	$2.299\,286\,119\,588\,755 \cdot 10^4$	$2.299\,286\,119\,588\,766 \cdot 10^4$	$0.000\,000\,000\,000\,011 \cdot 10^4$
$J\rho v$	$1.149\,643\,059\,794\,377 \cdot 10^4$	$1.149\,643\,059\,794\,385 \cdot 10^4$	$0.000\,000\,000\,000\,008 \cdot 10^4$
$J\rho E$	$5.533\,053\,824\,742\,974 \cdot 10^4$	$5.533\,053\,824\,742\,998 \cdot 10^4$	$0.000\,000\,000\,000\,024 \cdot 10^4$



(a) Analytic mapping.

(b) Interpolated mapping.

Figure 4.8: Convergence rates of the advection case. The Single-Level L_2 and L_{inf} plots are overlapped by the corresponding 3-Level-AMR plots, and thus not visible.

Table 4.2: Initial and final conservative values for the advection case, using the quintic spline interpolated mapping

$J\mathbf{U}$	Initial	Final	Difference
$J\rho$	$2.299\,286\,119\,578\,092 \cdot 10^4$	$2.299\,286\,119\,578\,102 \cdot 10^4$	$0.000\,000\,000\,000\,010 \cdot 10^4$
$J\rho u$	$2.299\,286\,119\,578\,092 \cdot 10^4$	$2.299\,286\,119\,578\,102 \cdot 10^4$	$0.000\,000\,000\,000\,010 \cdot 10^4$
$J\rho v$	$1.149\,643\,059\,789\,046 \cdot 10^4$	$1.149\,643\,059\,789\,051 \cdot 10^4$	$0.000\,000\,000\,000\,005 \cdot 10^4$
$J\rho E$	$5.533\,053\,824\,736\,307 \cdot 10^4$	$5.533\,053\,824\,736\,325 \cdot 10^4$	$0.000\,000\,000\,000\,018 \cdot 10^4$

4.4.3 Advection on a Mapped Multi-Block Grid

In a periodic advection cube, a Gaussian density profile is created and advected across the domain. Solutions from the present mapped multi-block method are compared to those previously obtained on a single-block. Using the Euler equations on a periodic domain, a constant uniform

flow velocity is specified as in Section 4.4.2. The density profile is again initialized using Equation 4.15 and Equation 4.15. The solution error is calculated relative to the exact solution.

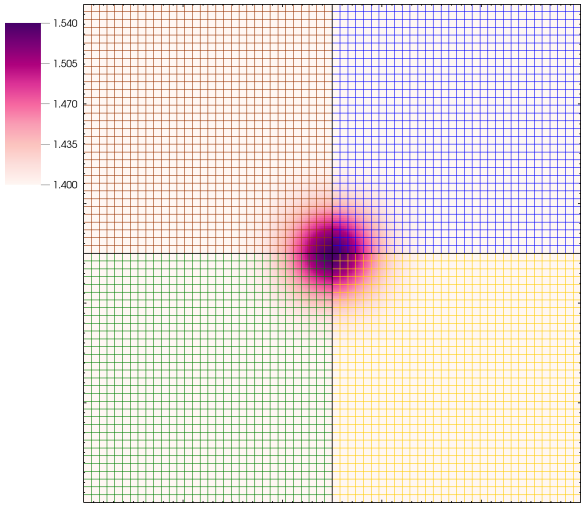


Figure 4.9: The density profile of the periodic advection case is shown on the Cartesian multi-block grid. The grid of each block is shown in a different color.

For verification, an analytically mapped single-block and multi-block grid are generated to be identical in physical space. The multi-block grid is shown in Figure 4.9, and although not shown, the grid in the single-block is identical except without the block distinctions. For both of these methods, a convergence study is performed. Both cases exhibit the expected fourth-order convergence, as shown in Figure 4.10. The solution errors between the single-block and multi-block method are seen to be similar, with the multi-block errors only marginally higher. For this simple Cartesian geometry, results from discrete grids interpolated using the B-spline approach are identical to their analytic counterparts within machine precision.

Additionally, it is possible to test conservation of the periodic advection case. Without source or sink terms in the governing equations, the sum of conservative quantities in the control volume should remain constant. This is tested by comparing integration of the solution variables $\mathbf{U}$ in physical space at the initial and final solution times. Results from a global grid of size 256×256 are tabulated for the single-block method in Table 4.3 and multi-block method in Table 4.4. The conservation is maintained in both cases; solutions are conserved to near machine precision.

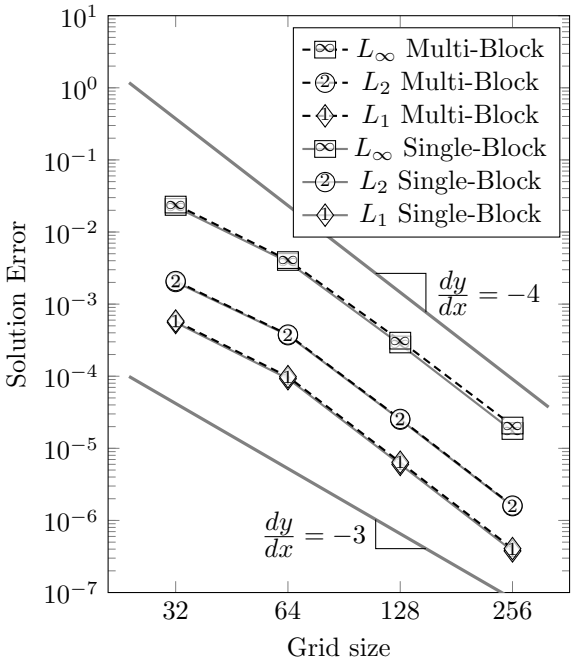


Figure 4.10: Convergence rates for the periodic advection cube, using both a mapped single-block grid and a multi-block one. Solution errors from both methods lie very near one another, and both show fourth-order convergence.

Importantly for this study, the multi-block scheme is shown to maintain conservation equally well as the single-block scheme.

Table 4.3: Initial and final conservative values for the advection case, using the single-block grid

U	Initial	Final	Difference
ρ	$9.197\,144\,478\,240\,506 \cdot 10^4$	$9.197\,144\,478\,240\,158 \cdot 10^4$	$0.000\,000\,000\,000\,348 \cdot 10^4$
ρu	$9.197\,144\,478\,240\,506 \cdot 10^4$	$9.197\,144\,478\,240\,144 \cdot 10^4$	$0.000\,000\,000\,000\,362 \cdot 10^4$
ρv	$4.598\,572\,239\,120\,253 \cdot 10^4$	$4.598\,572\,239\,120\,067 \cdot 10^4$	$0.000\,000\,000\,000\,186 \cdot 10^4$
ρE	$2.213\,221\,529\,890\,319 \cdot 10^5$	$2.213\,221\,529\,890\,331 \cdot 10^5$	$0.000\,000\,000\,000\,012 \cdot 10^4$

Table 4.4: Initial and final conservative values for the advection case, using the multi-block grid

U	Initial	Final	Difference
ρ	$9.197\,144\,478\,240\,506 \cdot 10^4$	$9.197\,144\,478\,240\,123 \cdot 10^4$	$0.000\,000\,000\,000\,383 \cdot 10^4$
ρu	$9.197\,144\,478\,240\,506 \cdot 10^4$	$9.197\,144\,478\,240\,107 \cdot 10^4$	$0.000\,000\,000\,000\,405 \cdot 10^4$
ρv	$4.598\,572\,239\,120\,253 \cdot 10^4$	$4.598\,572\,239\,120\,053 \cdot 10^4$	$0.000\,000\,000\,000\,200 \cdot 10^4$
ρE	$2.213\,221\,529\,890\,319 \cdot 10^5$	$2.213\,221\,529\,890\,325 \cdot 10^5$	$0.000\,000\,000\,000\,006 \cdot 10^5$

4.5 Results and Discussion

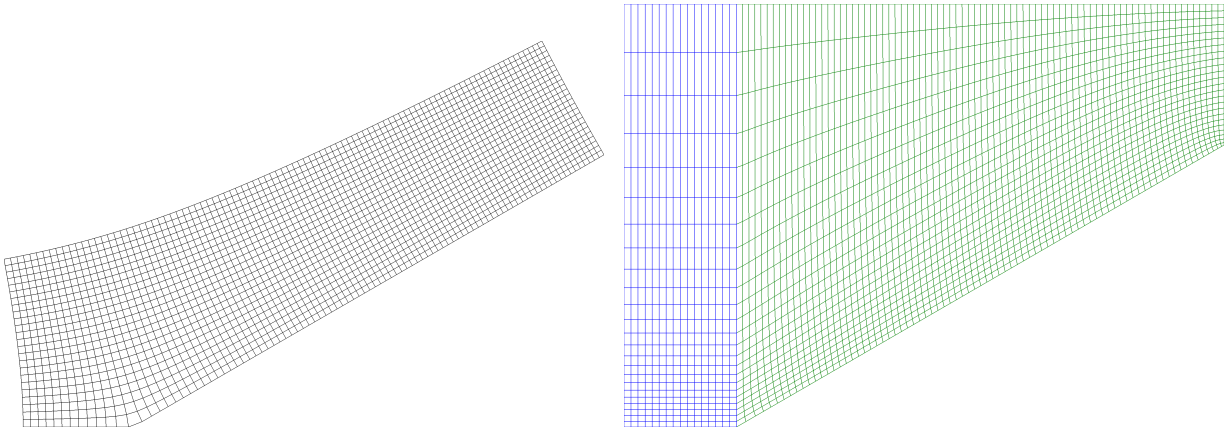
The verified and validated MMB algorithm is now applied to solve problems of engineering interest. First, the Mach reflection problem is examined, and we compare the shock capturing capability in the MMB algorithm by comparing to literature. Second, the premixed combustion of C_3H_8-air in a bluff-body combustor is demonstrated. This represents common complex geometric features and fluid dynamics occurring in practical combustion devices.

4.5.1 Mach Reflection

In previous work, the Woodward-Colella mach reflection case has been studied in detail using Chord [6]. There are a number of interesting physical phenomenon in this flow, such as shocks and induced flow instabilities. In this study, the case is of interest because it is one of the most geometrically complex cases using a mapped single-block grid. The single-block solution uses a Schwarz-Christoffel mapping to generate the ramp. Although the geometry of the bottom boundary with a sharp feature is possible, it comes at loss of geometry specification in the remaining domain. Due to the geometry, applying boundary conditions requires special treatment, and the grid can no longer be aligned with the shock. However, the multi-block methodology can create a grid with full geometric flexibility, allowing for simpler boundaries and grid alignment to flow features. A comparison study is made between the previously validated mapped single-block grid solution, and the new multi-block one.

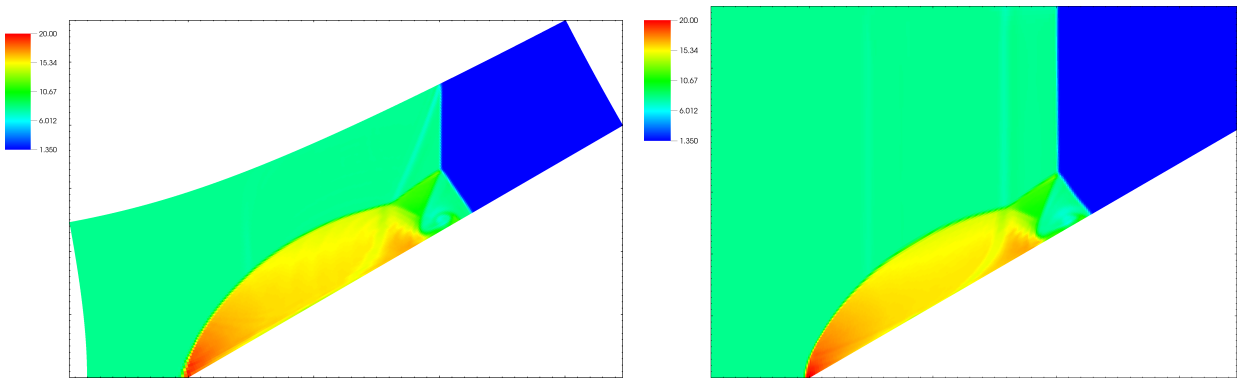
A ramp of angle of 30° is created. An incident shock of Mach 10 reflects off the ramp. The front of the shock has specified flow conditions $p = 1$, $\rho = \gamma$, and $\mathbf{u} = 0$. Conditions behind the shock are set from the shock relations. The bottom boundary is treated as a slip wall, while all others have specified Dirichlet conditions. The top boundary is specially adjusted based on the analytic position of the shock.

This case is of particular interest when using the multi-block scheme, since it introduces a scenario that exhibits exterior ghost cells. Both solutions clearly capture the same physics, with minor differences, as shown in Figure 4.12. This is particularly encouraging near the corner of the



(a) Single-block analytic Schwarz-Christoffel grid. **(b)** Multi-block discretely defined grid.

Figure 4.11: Coarsened representations of the grid used to capture the mach reflection geometry.



(a) Single-block solution. **(b)** multi-block solution.

Figure 4.12: A comparison of the density contours between the mapped single-block and multi-block solutions.

ramp, where the exterior ghost cell extrapolation method is active. The multi-block case, with the grid represented in Figure 4.11, appears better able to capture the off body vertical shock along the top boundary where the grid is aligned with the physics. However, the grid is no longer entirely aligned with the shock orthogonal to the wall and a minor distortion is observed. From the results of this comparison, we are able to validate the mapped multi-block methodology on arbitrary discrete grids.

4.5.2 C_3H_8-air Bluff-Body Combustor

The bluff-body combustor geometry features a triangular body inside a channel followed by a plenum region, as shown in figure Figure 4.13. The geometry is extended into three-dimension, with the z-direction defined with a depth of $76.2\,\mathrm{mm}$ and enforced with periodic boundaries. The boundary conditions in the channel are defined by adiabatic no-slip (zero velocity) walls. Walls immediately upstream of the channel and inside the plenum are specified with an adiabatic slip wall condition. The inlet is specified by a gaseous mixture of 4.01% C_3H_8, 22.36% O_2, and 73.62% N_2 by mass fraction entering at a velocity of $15.7\,\mathrm{m\,s^{-1}}$. The mixture has a temperature of $310\,\mathrm{K}$. The outlet is specified by species mass fractions of 12% CO_2, 6.54% H_2O, 5.14% O_2, and 73.62% N_2. The outlet has a temperature of $1300\,\mathrm{K}$, and pressure of $101\,325\,\mathrm{Pa}$.

The initial conditions for the plenum are set to the same values as the inlet, except for a small region surrounding the bluff-body. In this region, the gas mixture is initialized to the state defined for the outlet condition. This hot spot serves as the ignition of the combustion. The flow has a bulk Mach number of 0.053 and bulk Reynolds number of $50\,000$. The reaction mechanism for C_3H_8-air is a stiff system including 25 species and 66 reactions [31].

The bluff-body geometry is represented by a conforming mapped multi-block grid, with the blocks shown in green in Figure 4.13 and the black lines showing a coarsened representation of the grid. In the z-direction the grid is uniformly extended and not shown. This multi-block grid consists of 25 connected blocks. The solution has three levels of grids, each refined by a factor of 2. The base mesh contains $393\,216$ cells, generated by mesh generation software demonstrating that Chord works with pre-created grids. Using the grid interpolation feature of Chord, AMR is feasible. The second level in the AMR grid, which is fixed in time, contains $1\,302\,528$ cells. The third level is tagged dynamical based on the criteria $c_{\mathrm{OH}} \times c_{\mathrm{CH_2O}} > 2.0 \cdot 10^{-9}$ which is found as an effective flame indicator.

Figure 4.14 shows an instantaneous isosurface of the OH species for the 3D C_3H_8-air flame. Without using AMR, it is computationally infeasible to use a uniform fine mesh to capture the flow and flame dynamics near the bluff-body as shown in Figure 4.14. At the time shown, the grid

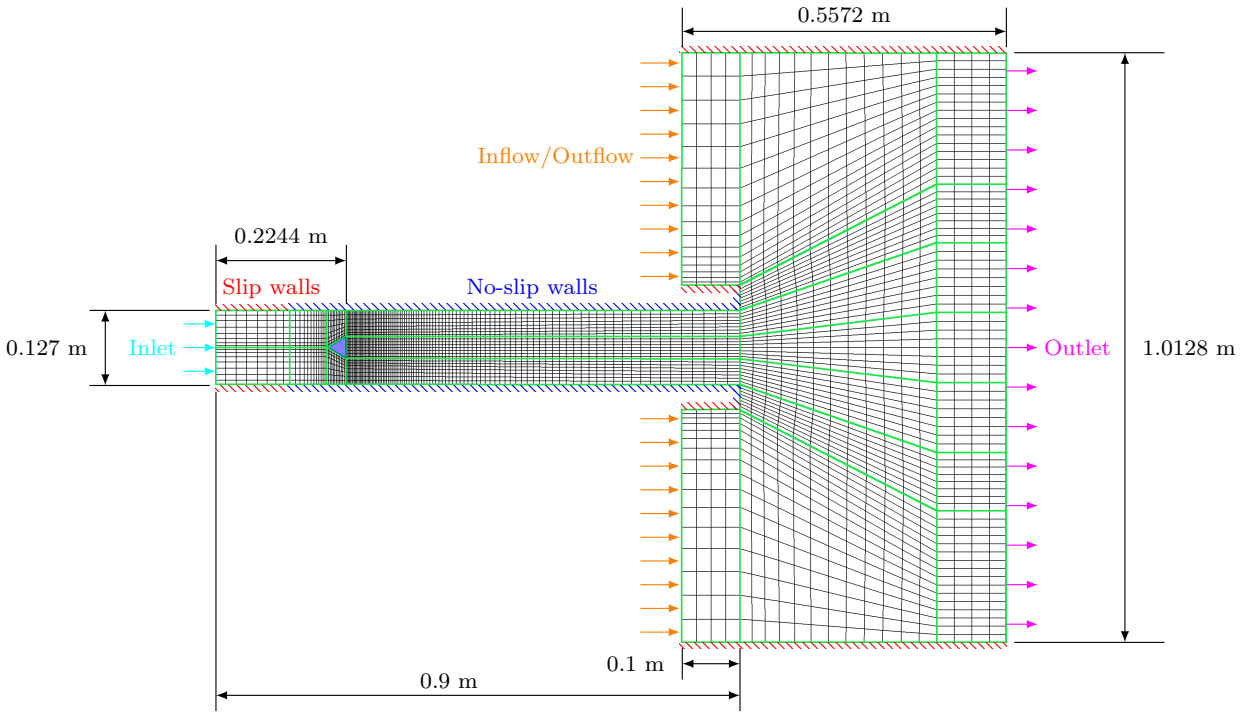


Figure 4.13: Coarsened grid for the bluff-body case. Boundary conditions are labeled and color coded. Block boundaries are highlighted in green.

contains a total of 5 477 376 cells using AMR. For comparison, the same resolution using a single level would require 25 165 824 cells, which is nearly 4.6 times as many as required with AMR.

In this mesh, several regions experience the mapping limitations as described in Section 4.3.3. Careful considerations are made to avoid these limitations, and they often determine the allowed amount of grid stretching. For example, the expansion rate of the mesh in the plenum region is restricted to maintain positive ghost cell sizes. This mapped multi-block grid also contains sharp corners around the bluff-body and at the entrance of the plenum. Each of these regions must deal with exterior ghost cells, and there are notably no erroneous solution features appearing in these regions. Additionally, the solution for this case has many complex flow features (e.g., shearing, mixing, and recirculating) and AMR interfaces that regularly cross block boundaries. Despite this, no distortion is found due to multi-block interfaces. Successful applications from using this MMB algorithm have been shown in other studies [30, 55].

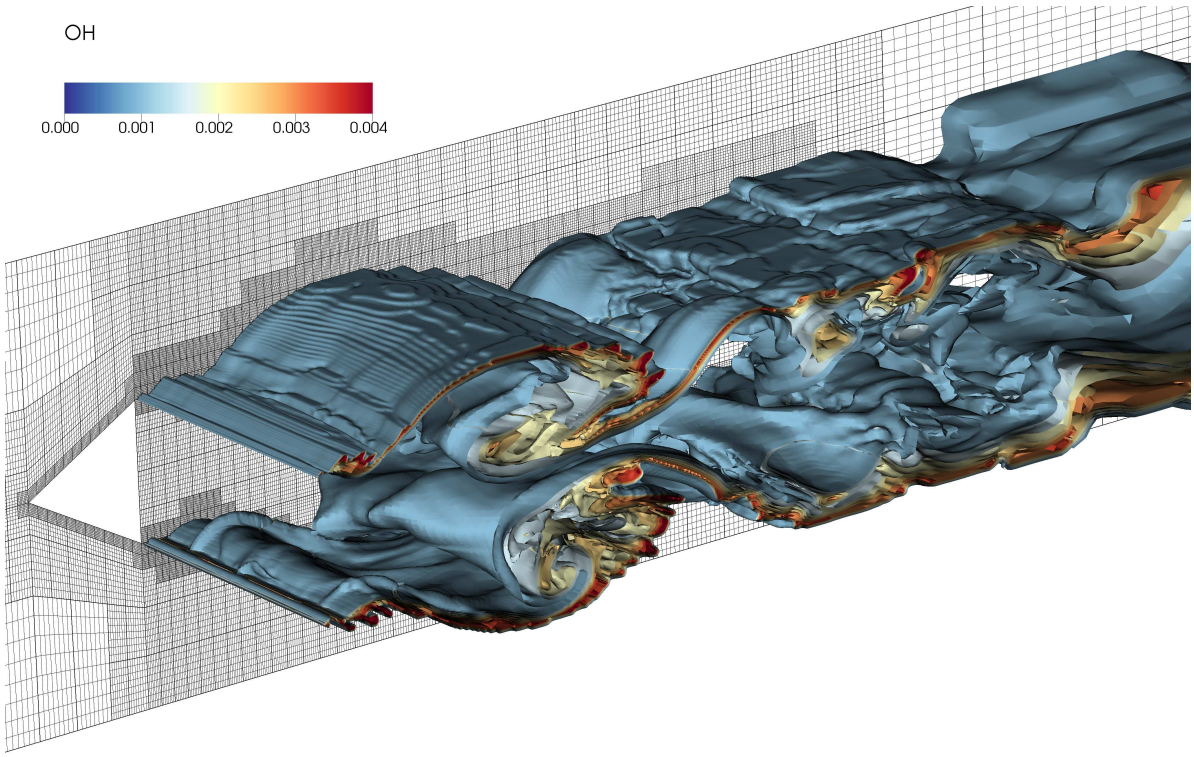


Figure 4.14: c_{OH} isosurfaces of the 3D Bluff-Body C_3H_8-air case, with a two dimensional slice of the grid and AMR.

Chapter 5

High-Order Embedded-Boundary Method

In this chapter, we devise a new high-order embedded-boundary scheme for the time dependent Stokes equations. First, the motivations for developing this method are discussed, along with the current state of embedded-boundary methods. Next, the technical approach for the high-order EB method is described. The method is then verified and validated, and finally results of the Stokes equations featuring complex geometries are demonstrated.

5.1 Motivations

The embedded-boundary method has been used with great success in a number of fluid dynamics applications, and has been shown capable of representing highly complicated geometries with automated meshing [44, 66–68]. Previous time dependent applications have only been achieved up to second-order accurate solutions, with first-order or inconsistent results near embedded boundaries [35]. Elaborate reconstruction schemes are required near the boundaries of embedded-boundary methods, since regular or grid-aligned stencils are not accurate or stable in the presence of cut cells. Volumes of cut cells may also be arbitrarily small with respect to the Cartesian cells they originate from, and maintaining stability of these small cells requires special care. Dealing with these irregular reconstructions and small cells has been achieved for low-order schemes, but not widely explored for high-order schemes. The related immersed boundary method has been successfully developed for high-order solutions of complex flows using finite difference methods [69,70], although this method is not considered in this dissertation. The immersed boundary method is in general diffusive at the boundaries and not conservative.

A high-order finite volume embedded-boundary method for smooth and kinked (C^0) domains was demonstrated for Poisson's equation by Devendran et al [12]. The goal of this work is to extend the fourth-order embedded-boundary method to solve the time dependent Stokes equations. Spatial discretizations are based on a weighted least-squares technique that mitigates the "small

cut cell" problem, without mesh modifications, cell merging, or redistribution. Advancing the viscous term in time is done using a fourth-order implicit time marching method, while the source terms are done explicitly. These two schemes are combined using an additive Runge-Kutta (ARK) scheme coupled with a projection method to solve the unsteady Stokes equations.

5.2 Technical Approach

The challenge of embedded-boundary methods is to approximate the flux terms $\langle \mathbf{F} \rangle_f$ when regular grid stencils cannot be used due to nearby cut cells. To solve this requires knowledge of the flux as a function, so that face averages may be computed appropriately. A general reconstruction is done by creating local polynomials and evaluating their value and derivatives on the face as needed to evaluate fluxes. For a structured grid, reconstructed polynomials lead to grid-aligned regular stencils. When using an embedded-boundary method, the cells near the boundary require a consistent approach to generate stencils that depend on the local geometry. On irregular regions, our approach is to use a weighted least-squares polynomial approximation from the cell average values in a local region of neighboring cells. The scheme presented can theoretically produce any order of spatial discretization that is desired, but for the context of this dissertation, fourth-order is the focus. The present study continues and extends the work started by Devendran et al. [12] and Schwartz et al. [11], and the procedure is briefly summarized in this section.

5.2.1 The Problem of Small Cut Cells

To reconstruct the stencils required for the spatial discretization in the presence of EB geometries, the WLS method Equation 2.31 is used. First, we select a neighborhood of cells within a given radius (including any cut-cells and their contained subset of the boundary). Rather than exhaustively searching for an interpolation neighborhood that can determine all the required coefficients with finite volume stencils, we use a larger number of neighbors and a weighting scheme that assigns relative importance to each cell's entry in the WLS system. Smaller relative weights mean that cells have less importance and a smaller coefficient in the resulting stencils. As in Devendran

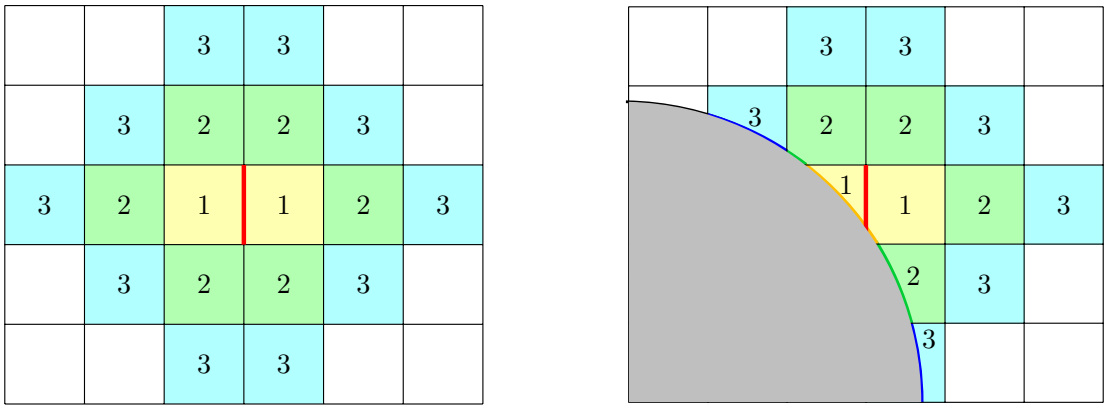
et al. [12], an effective weighting for a fourth-order interpolation is:

$$\mathrm{W}_{\boldsymbol{i},\boldsymbol{f}} = \max(D_{\boldsymbol{i},\boldsymbol{f}}, 1)^{-5}\,, \tag{5.1}$$

where $D_{\boldsymbol{i},\boldsymbol{f}}$ is the Euclidean distance between cell $\boldsymbol{i}$ center and face $\boldsymbol{f}$ center. This weighting was shown to improve solution stability and spectral properties, especially when interpolation neighborhoods are large and there are many possible consistent stencils.

5.2.2 Interpolation Neighborhoods

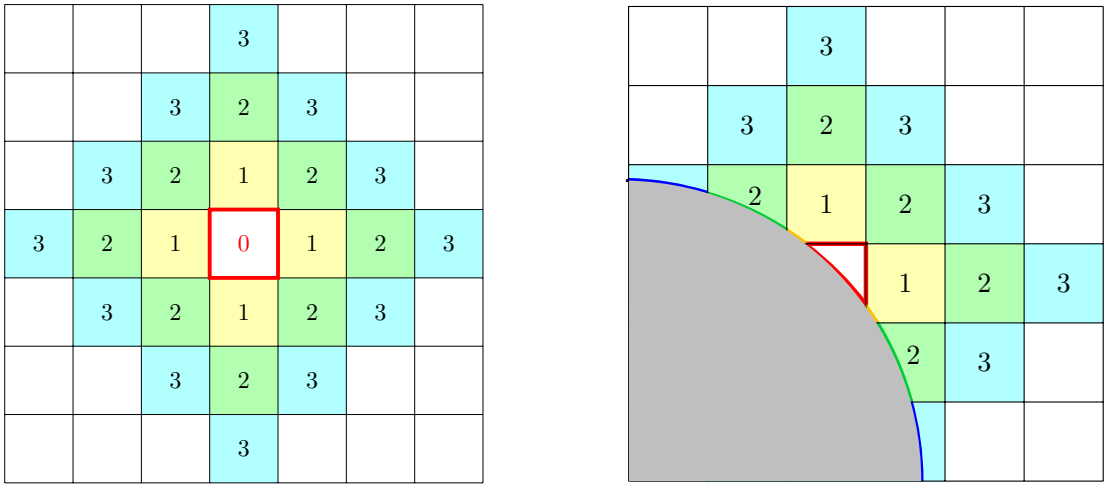
An important aspect of the WLS reconstruction is neighborhood selection. Conceptually, we require a large enough interpolation neighborhood to attain the desired accuracy and stability properties, while not making it so far-reaching so as to create large (expensive), ill-conditioned interpolation matrices. In this dissertation, the focus is on fourth-order accurate stencils, which in the general 2D case, requires a minimum of 10 neighbors (with at least Q in each direction) for all of the polynomial coefficients for $|\mathbf{q}| < Q = 4$. Higher-order derivative operations will require more neighbors to maintain the same accuracy, and values near an embedded boundary will require the boundary condition and further cells because of the lower accuracy of one-sided differences. For example, for a fourth-order WLS viscous operator (Laplacian), flux stencils of radius 3 cells from the reconstructed face can be used for a third-order accurate gradient. This is illustrated for a radius of 3 in Figure 5.1(a) for regular regions, and Figure 5.1(b) in the presence of an EB region with the inclusion of boundary conditions. However, for stencils that do not specify a boundary condition, such as the divergence on outflow boundaries, this radius of cells may need to be expanded to make the system sufficiently over-determined. Similarly, for stencils that evaluate cell-averaged values, such as the gradient for the projection operator, cell-average reconstructions are used. An example of a stencil of radius 3 centered about a cell is shown in Figure 5.2(a) for regular cells and Figure 5.2(b) for a cut-cell, following a similar pattern to reconstruction centered on faces. Cell-average reconstructions may include the cell where the reconstruction takes place, which is indicated as a radius of zero.



(a) An example regular flux stencil with 18 cells and no boundary conditions.

(b) An example irregular flux stencil, using 13 cells and 6 boundary conditions.

Figure 5.1: Stencil neighborhoods for flux construction in 2D on the red highlighted faces, with cells numbered according to the Manhattan distance from the indicated red face. Boundary sections included in the stencil are colored in (b).



(a) An example regular cell stencil with 25 cells and no boundary conditions.

(b) An example irregular cell stencil, using 16 cells and 7 boundary conditions.

Figure 5.2: Stencil neighborhoods for cell value construction around the red highlighted cells, with cells numbered according to the Manhattan distance from the red cell. Boundaries included in the stencil are colored in (b).

5.2.3 Higher-order EB Viscous Flux Stencil

We can use the WLS approach to calculate face-average fluxes, on partial or curved faces, for Equation 2.4 from cell-average quantities (as in [12]). This is accomplished by determining the stencil $\mathrm{S}_{\boldsymbol{f}}$ applied to neighbor values V, derived from the coefficient vector C:

$$\begin{aligned}\mathcal{A}_{\boldsymbol{f}}\langle \mathrm{F}_d\rangle_{\boldsymbol{f}} &\equiv \mathrm{S}_{\boldsymbol{f}}^{\intercal}\,\mathrm{V}\\ &= \mathrm{B}^{\intercal}\,\mathrm{C}\\ &= \left(\mathrm{B}^{\intercal}\,(\mathrm{W}\,\mathrm{M})^{\dagger}\,\mathrm{W}\right)\mathrm{V}\,,\end{aligned}$$

where B is a vector that approximates any face-average flux (*or other quantity*) from the known coefficients. For a linear flux, such as the viscous term $\mathrm{F}_d = \nu\nabla\boldsymbol{u}_d$, we determine B (and thus $\mathrm{S}_{\boldsymbol{f}}$) using a Taylor expansion from the face center $\bar{\boldsymbol{x}}_{\boldsymbol{f}}$:

$$\begin{aligned}\mathcal{A}_{\boldsymbol{f}}\langle \mathrm{F}_d\rangle_{\boldsymbol{f}} &= \int_{\mathcal{A}_{\boldsymbol{f}}} \nabla\boldsymbol{u}_d\cdot\hat{\mathbf{n}}_{\boldsymbol{f}}\,\mathrm{d}\boldsymbol{x} + O(h^{\mathrm{Q}})\\ &= \int_{\mathcal{A}_{\boldsymbol{f}}} \nabla\left(\sum_{|\mathbf{q}|<Q} c_d^{\mathbf{q}}(\boldsymbol{x}-\bar{\boldsymbol{x}}_{\boldsymbol{f}})^{\mathbf{q}}\right)\cdot\hat{\mathbf{n}}_{\boldsymbol{f}}\,\mathrm{d}\boldsymbol{x} + O(h^{\mathrm{Q}})\\ &\equiv \sum_{|\mathbf{q}|<Q} c_d^{\mathbf{q}} b_{\boldsymbol{f}}^{\mathbf{q}} = \mathrm{B}^{\intercal}\mathrm{C}\,.\end{aligned}$$

Each $b_{\boldsymbol{f}}^{\mathbf{q}}$ can be expressed as a sum of *normal-weighted* face moments $m_{\boldsymbol{f},d}^{\mathbf{q}}$:

$$b_{\boldsymbol{f}}^{\mathbf{q}} = \sum_{d=1}^{D}\int_{\mathcal{A}_{\boldsymbol{f}}}\frac{\partial}{\partial x_d}(\boldsymbol{x}-\bar{\boldsymbol{x}}_{\boldsymbol{f}})^{\mathbf{q}}\,\hat{\mathbf{n}}_{\boldsymbol{f},d}\,\mathrm{d}\boldsymbol{x} = \sum_{d=1}^{D}\mathrm{q}_d m_{\boldsymbol{f},d}^{\mathbf{q}-\boldsymbol{e}^d}\,, \tag{5.2}$$

where $\boldsymbol{q}-\boldsymbol{e}^d$ is required to be positive (derivatives of constants are zero).

For grid-aligned cell faces, normals $\boldsymbol{e}^d$ are constant, so the normal-weighted moments are simply the face moments. However, for curved embedded boundaries all the components of the normal play a role in the interpolation, especially in the case of inhomogeneous boundary conditions. Ultimately, the viscous flux stencil $s_{\boldsymbol{f}}$ depends only on the local neighbor volume and boundary

moments, so it may be initialized once per geometry and stored as a sparse matrix operator over the (much smaller) subset of irregular cells. This process requires having moments, m, cell volumes, $\mathcal{V}$, and face areas, $\mathcal{A}$ available to sufficient order of accuracy. These quantities are computed from geometries defined by implicit functions using the divergence theorem based algorithm detailed by Schwartz et al. [11].

5.2.4 Approximate Projection

The higher-order projection operator starts with cell-average velocities, $\langle \boldsymbol{u} \rangle_{\boldsymbol{i}}$, and modifies them to be *approximately* divergence-free (see Equation 2.55). To accomplish this, we require discretizations and boundary conditions for each operator in Equation 2.53. First, the Laplacian operator $\mathbf{L}$ is essentially the same as the viscous flux in section 5.2.3, as a divergence of face-averaged quantities, but with different boundary conditions based on Equation 2.56. For the cell-average gradient operator, $\mathbf{G}$, the WLS stencil is similar to Equation 5.2. However, in this case we use cell moments instead of face moments to calculate a cell-average gradient:

$$G^{\mathbf{q}}_{d,\boldsymbol{i}} = \int_{\mathcal{V}_{\boldsymbol{i}}} \frac{\partial}{\partial x_d} (\boldsymbol{x} - \bar{\boldsymbol{x}}_{\boldsymbol{i}})^{\mathbf{q}} \, \mathrm{d}\boldsymbol{x} = \mathrm{q}_d m_{\boldsymbol{i}}^{\mathbf{q}-\mathbf{e}^d} \,. \tag{5.3}$$

The gradient operator uses the same boundary conditions as the Laplacian operator.

Finally, we define a cell-average divergence operator $\mathbf{D}$ using the divergence theorem,

$$\int_{\mathcal{V}_{\boldsymbol{i}}} \nabla \cdot \boldsymbol{u} \, \mathrm{d}\boldsymbol{x} = \int_{\partial \mathcal{V}_{\boldsymbol{i}}} \boldsymbol{u} \cdot \hat{\mathbf{n}} \, \mathrm{d}\boldsymbol{x} \,,$$

so the cell-average of the divergence of the velocity field can be determined from face-average quantities as

$$\langle \nabla \cdot \boldsymbol{u} \rangle_{\boldsymbol{i}} = \sum_{\boldsymbol{f} \in \partial \mathcal{V}_{\boldsymbol{i}}} \frac{\mathcal{A}_{\boldsymbol{f}}}{\mathcal{V}_{\boldsymbol{i}}} \langle \boldsymbol{u} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}} \,.$$

The divergence flux $\langle \boldsymbol{u} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}}$ can be constructed similarly to the Laplacian, with an operator that computes an average velocity flux U_f from each velocity component coefficient

$$U_{\boldsymbol{f}}^{\mathbf{q}} = \sum_{d=1}^{D} \int_{\mathcal{A}_{\boldsymbol{f}}} (\boldsymbol{x} - \bar{\boldsymbol{x}}_{\boldsymbol{i}})^{\mathbf{q}} \hat{\mathbf{n}}_{\boldsymbol{f},d} \, \mathrm{d}\boldsymbol{x} = \sum_{d=1}^{D} m_{\boldsymbol{f},d}^{\mathbf{q}} \,. \tag{5.4}$$

Again, the terms on regular faces have single component normal vectors $\hat{\mathbf{n}}_{\boldsymbol{f}} = \boldsymbol{e}^d$, and as a result $\langle \boldsymbol{u} \cdot \hat{\mathbf{n}} \rangle_{\boldsymbol{f}} = \langle \boldsymbol{u}_d \rangle_{\boldsymbol{f}}$. The boundary condition for the divergence is $\boldsymbol{u} \cdot \hat{\mathbf{n}} = 0$ at any solid wall, including EB boundaries. The velocity is specified at inflow boundaries, while outflow has no boundary conditions applied [71].

5.2.5 Physical Boundary Conditions

For Dirichlet boundary conditions, each face with a prescribed function uses a boundary average value $\langle \boldsymbol{u} \rangle_{\boldsymbol{f}} = \langle \boldsymbol{u}_{\mathrm{bc}}(\boldsymbol{x}) \rangle_{\boldsymbol{f}}$. A polynomial fitting the function about a face can be reconstructed using:

$$\begin{aligned} \mathcal{A}_{\boldsymbol{f}} \langle \boldsymbol{u}_{\mathrm{bc},d} \rangle_{\boldsymbol{f}} &= \int_{\mathcal{A}_{\boldsymbol{f}}} \boldsymbol{u}_{\mathrm{bc},d} \, \mathrm{d}\boldsymbol{x} \\ &= \int_{\mathcal{A}_{\boldsymbol{f}}} \sum_{|\mathbf{q}|<Q} c_{\boldsymbol{f}}^{\mathbf{q}} (\boldsymbol{x} - \bar{\boldsymbol{x}}_{\boldsymbol{f}})^{\mathbf{q}} \, \mathrm{d}\boldsymbol{x} \\ &= \sum_{|\mathbf{q}|<Q} c_{\boldsymbol{f}}^{\mathbf{q}} m_{\boldsymbol{f}}^{\mathbf{q}} \,. \end{aligned}$$

Including this additional equation and value into the stencil system will match the boundary condition in a least-squares sense, with a similar method for Neumann boundaries. These extra boundary condition equations are used when any neighboring cell included in the reconstruction contains a portion of the boundary, and so can accommodate different parts of the boundary (such as corners, etc.). This does require that the boundary conditions and derivatives are known at least to order Q.

5.2.6 Time Marching Method

When solving the Stokes equations, time constraints for the source term and diffusion portions of the fluxes can be significantly different. The source terms are hyperbolic in nature and generally non-linear and less stiff, making them well suited for explicit time marching methods. In contrast, the diffusion fluxes are parabolic, more stiff, and linear, making them well suited for implicit time marching methods using fast linear solvers. In the context of embedded-boundary methods, where cut cells can become arbitrarily small, the time step stability constraints between advection and diffusion terms can differ by orders of magnitude. To maintain reasonable time step sizes, a hybrid implicit-explicit (ImEx) Runge-Kutta method is chosen for the embedded-boundary algorithm [72]. This allows for the advection terms to be updated using an explicit method, and the diffusion terms with an implicit method. As a result, the time evolution is generally not limited by the small time steps required for the diffusion physics. At each stage of the implicit RK time marching, a large sparse matrix system must be solved. To do this efficiently, we use a geometric multigrid solver included in Chombo [73] with PETSC [74, 75] as a bottom solver.

To achieve the desired fourth-order accuracy of the presented algorithm, the ARK4(3)6L[2]SA time marching method [21] is used. This method has successfully been demonstrated for stiff, higher-order finite volume methods that use AMR for advection-diffusion problems [76].

5.3 Verification and Validation

We verify the projection and viscous operators separately on simple embedded boundary geometries. The projection operator was verified for correctness and stability using the Taylor Green vortex. The viscous operator with boundary conditions was verified using manufactured solutions in space and time. The order of accuracy for solving the Stokes equations was verified on a Couette flow between concentric circles.

5.3.1 Projection of the Taylor-Green Vortex

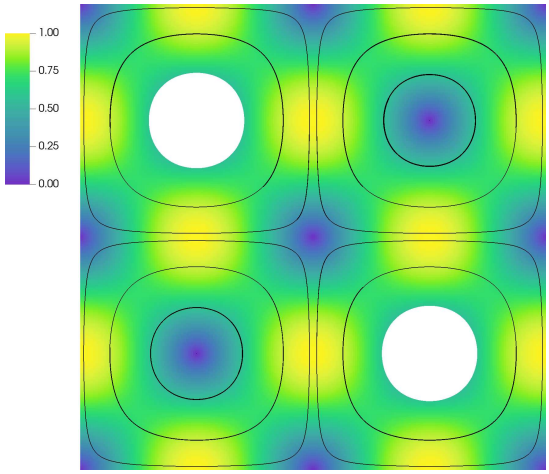
The approximate projection operator $\mathbf{P}(\boldsymbol{u})$ is demonstrated to be both fourth-order accurate and stable by solving the classic Taylor-Green vortex [32] on a unit domain. The stream function is defined by

$$\psi = \sin(n\pi x)\sin(n\pi y)\,. \tag{5.5}$$

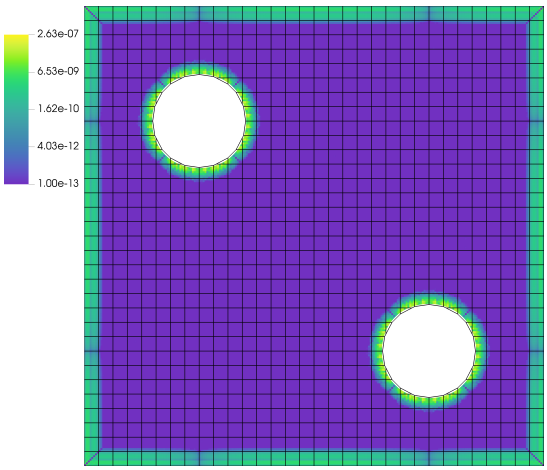
and the corresponding velocity field in Cartesian coordinates is,

$$u = \sin(n\pi x)\cos(n\pi y)\,, \quad v = \cos(n\pi x)\sin(n\pi y)\,, \tag{5.6}$$

which is analytically divergence free. For this case, we choose the period $n = 2$, and as a result there is no flow through the domain boundary. In addition, EB boundaries are cut along the contours of the stream function at $\psi = -0.8$. Initial conditions use the exact velocity profiles, integrated to fourth-order cell-averages, while projection boundary conditions are specified (no-flow, but not viscous, walls). The geometry and velocity field are shown in Figure 5.3(a).



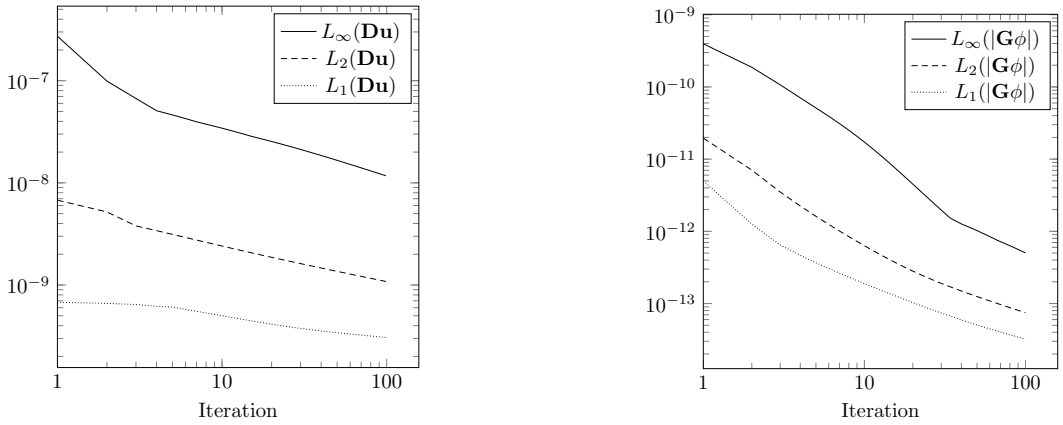
(a) The Taylor-Green velocity magnitude $|\boldsymbol{u}|$ and streamlines.



(b) Log plot of the divergence magnitude $\mathbf{D}\boldsymbol{u}$ and a coarsened grid representation.

Figure 5.3: The Taylor-Green vortex at $h = 1/256$.

Stability of the approximate projection operator is demonstrated by showing that repeated applications of the projection on a velocity field reduce the discrete divergence monotonically towards zero. The velocity divergence $\mathbf{D}\boldsymbol{u}$ and pressure gradient $\mathbf{G}\phi$ are computed on a grid with cell size $h = 1/256$ for 100 projection applications and plotted in Figure 5.4. These quantities strictly decrease, showing stability of the fourth-order approximate projection.



(a) Error norms of the velocity divergence.

(b) Error norms of the pressure gradient.

Figure 5.4: Solution error norms from repeated projections of the Taylor-Green vortex with cell size $h = 1/256$.

Convergence tests are performed to ensure the targeted fourth-order accuracy is achieved. The projection is applied once for grids of decreasing refinement, and the divergence field $\mathbf{D}\boldsymbol{u}$ is used to evaluate the solution error. The finest levels are chosen with cell sizes $h = 1/256$ and $h = 1/192$, and subsequent coarser levels each double h. The L_1, L_2, and L_∞ errors are calculated and plotted in Figure 5.5, where convergence rates reach and exceed expected fourth-order accuracy. Divergence errors are the largest in cut-cells at the embedded boundaries, as seen in Figure 5.3(b), and dominate the L_∞ errors. Nevertheless, the solution in cut-cells still converges at the expected rate.

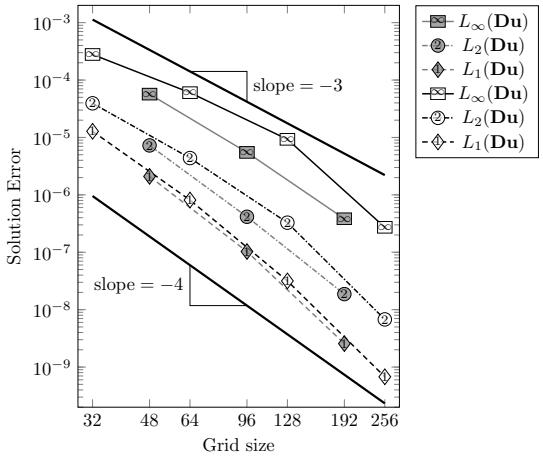


Figure 5.5: Convergence rates of the divergence for the Taylor-Green vortex in 2D. The gray series of points are coarsened from a fine level of $h = 1/192$, and the white series coarsened from $h = 1/256$.

5.3.2 Manufactured Solution for Diffusion Inside a Circle

To demonstrate the accuracy of the viscous operator in the Stokes equations, we use a manufactured solution and solve without the projection operator. This completely decouples the velocity equations. The algorithm targets fourth-order in space and time using the high-order stencils described in section 5.2.6 and the ImEx scheme in section 5.2.6. A circular domain is created of radius 0.3 centered about the point $(0.5, 0.5)$. The manufactured solution is the same for each component, defined by

$$\boldsymbol{u}_d(\boldsymbol{x}, t) = \sin(2\pi t)\sin(R^2 - (\boldsymbol{x} - \boldsymbol{x}_0)^2) \tag{5.7}$$

where $R = 0.3$ matches the domain radius, and $\boldsymbol{x}_0 = (0.5, 0.5)$ gives the domain center. The embedded boundaries are specified by Dirichlet conditions with values determined by the manufactured solution. Following the method of manufactured solutions, a source term is added to balance the equation where

$$\begin{aligned}\mathrm{s}_d(\boldsymbol{x}, t) =& 2\pi\cos(2\pi t)\sin(R^2 - (\boldsymbol{x} - \boldsymbol{x}_0)^2) \\ &- \sin(2\pi t)\sum_d^D \left(-4\mathrm{x}_d \sin(R^2 - (\boldsymbol{x} - \boldsymbol{x}_0)^2) + 2\cos(R^2 - (\boldsymbol{x} - \boldsymbol{x}_0)^2)\right) .\end{aligned}$$

The source term is evaluated explicitly in time, while the Laplacian term is evaluated implicitly. The solution is initialized using fourth-order cell-averages at time 0.125, and a viscosity of $\nu = 1$. On the finest level, with cell size $h = 1/128$, a time step of $\Delta t = 0.1$ is taken to advance the solution forward for 128 steps. Subsequent coarser levels double the cell size and time step, while halving the number of time steps to reach the same end time. Using the chosen exact solution in Equation 5.7, the L_1, L_2, and L_∞ errors and convergence rates are calculated and compiled in Table 5.1. Third-order truncation error is anticipated at the embedded boundaries, and will dominate the L_∞ norm. However, the L_1 and L_2 norms still attain fourth-order accuracy because the embedded boundary is only codimension one [77]. Fourth-order accuracy is demonstrated in these error norms, verifying the algorithm.

Table 5.1: Viscous operator convergence errors and rates for diffusion inside a circle.

N	L_1	Rate(L_1)	L_2	Rate(L_2)	L_∞	Rate(L_∞)
16	$3.676 \cdot 10^{-7}$	—	$5.323 \cdot 10^{-7}$	—	$1.271 \cdot 10^{-6}$	
32	$1.421 \cdot 10^{-8}$	4.693	$2.111 \cdot 10^{-8}$	4.656	$6.810 \cdot 10^{-8}$	4.223
64	$6.449 \cdot 10^{-10}$	4.463	$9.529 \cdot 10^{-10}$	4.470	$3.186 \cdot 10^{-9}$	4.418
128	$3.688 \cdot 10^{-11}$	4.128	$5.467 \cdot 10^{-11}$	4.123	$2.195 \cdot 10^{-10}$	3.860

5.3.3 Spherical Couette Flow

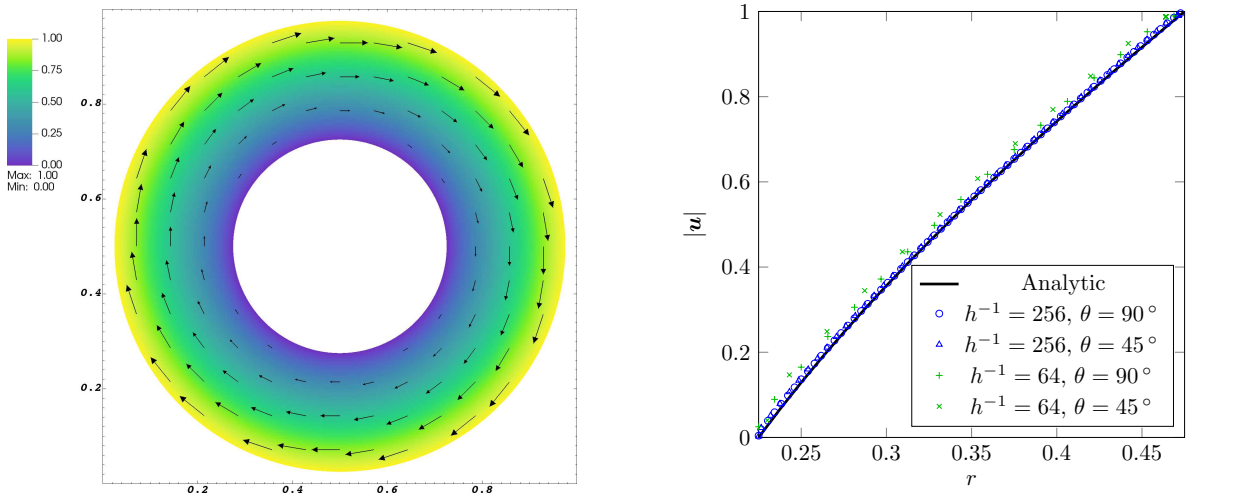
A pair of concentric spheres are created with radii $r_{\text{inner}} = 0.25$ and $r_{\text{outer}} = 0.475$. The inner sphere is held stationary, while the outer sphere is rotated about the z-axis with constant angular velocity $\omega_{\text{outer}} = \frac{1}{0.475}$ so that the outer sphere has a peak tangential velocity of 1. Velocity of the rotating sphere is specified purely tangential to the surface by

$$u_{\text{tan}} = \sin(\varphi) R\omega$$

where R is the sphere radius, ω the angular velocity, and φ the polar angle from the z-axis. Converting this to Cartesian coordinates prescribes a velocity field

$$u = \omega y\,, \qquad v = -\omega x\,, \qquad w = 0\,,$$

where x and y are the Cartesian coordinates on the sphere centered at the origin. The solution is initialized to zero velocity uniformly with a viscosity of $\nu = 1$. Using the developed method in this work, the Stokes equations are solved to steady state. A two-dimensional case normal to the z-axis is shown in Figure 5.6(a), and the three-dimensional version in Figure 5.9.



(a) Velocity magnitude and vectors for the $h = 1/256$ solution.

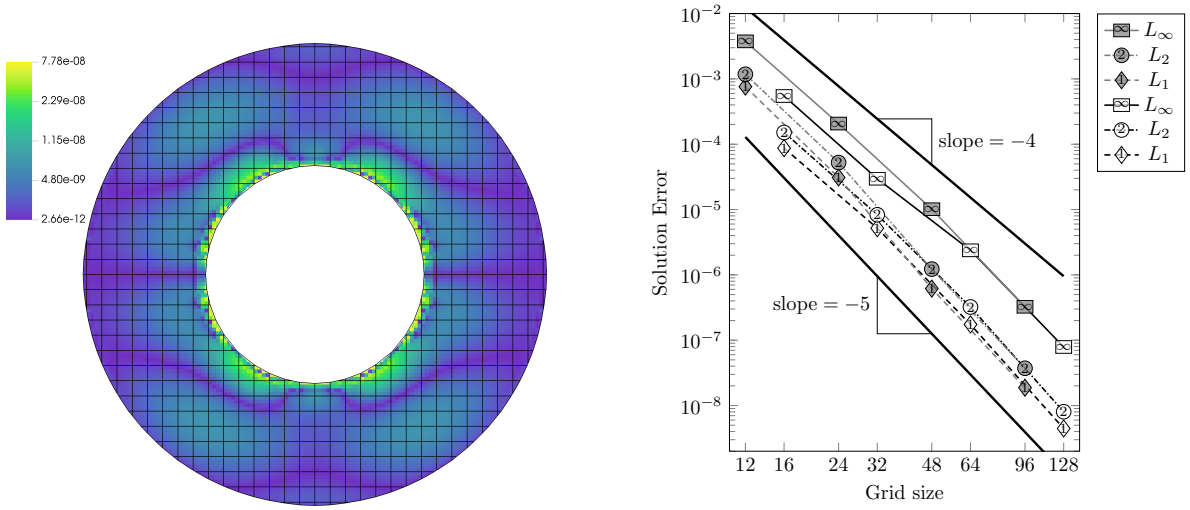
(b) Velocity magnitude profiles compared to the analytic solution.

Figure 5.6: The two-dimensional circular Couette flow at steady state.

On the finest level, of cell size $h = 1/256$, a time step of $\Delta t = 1 \cdot 10^{-3}$ is taken to advance the solution forward until the divergence norm converges in the two-dimensional case. To validate the solutions, radial profiles of the solution in two dimensions are compared to the analytic solution [78]

$$u_\theta(r) = \omega_{\text{outer}} r_{\text{outer}} \frac{r/r_{\text{inner}} - r_{\text{inner}}/r}{r_{\text{outer}}/r_{\text{inner}} - r_{\text{inner}}/r_{\text{outer}}}\,,$$

in Figure 5.6(b), where good agreement is observed. Convergence rates are measured and shown in Figure 5.7(b) using the Richardson extrapolation method for a series of grids coarsened by a factor of 2 from refinements of both $h = 1/256$ and $h = 1/192$. Solution norms for the u and v velocity components only have differences on the order of machine precision, since the flow is symmetric, and so only one set of errors are shown. Results show that fourth-order accuracy is achieved or exceeded for all solution norms. The L_∞ norm in particular is dominated by cut-cell values, as shown in Figure 5.7(a). On the finest grid there are $39\,364$ valid cells, of which 716 are cut-cells with volume fractions as small as $\kappa = 1.317 \cdot 10^{-5}$, demonstrating the robustness of the method. The distribution of cut-cell sizes is shown in Figure 5.8.



(a) Solution errors of the u velocity on the $h = 1/128$ grid, with a coarsened grid representation.

(b) Convergence rates of the u velocity. The gray series of points are coarsened from a fine level of $h = 1/192$, and the white series from $h = 1/256$.

Figure 5.7: The two-dimensional circular Couette flow solution errors.

5.4 Results and Discussion

Using the verified and validated fourth-order EB algorithm, Stokes flow over a circle and sphere in a channel are tested, and convergence order is verified. The Stokes equations are also used to solve for flow through a complex bio-inspired material of engineering interest.

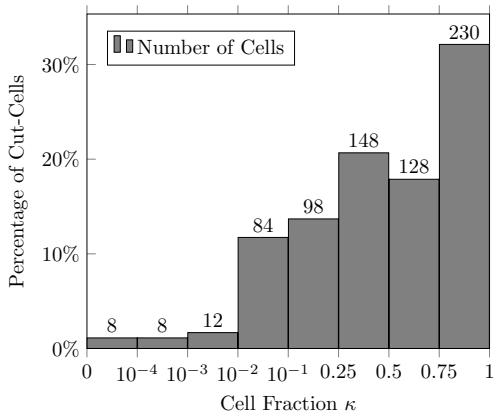


Figure 5.8: Distribution of cut-cell sizes for the two-dimensional Couette flow at cell size $h = 1/256$.

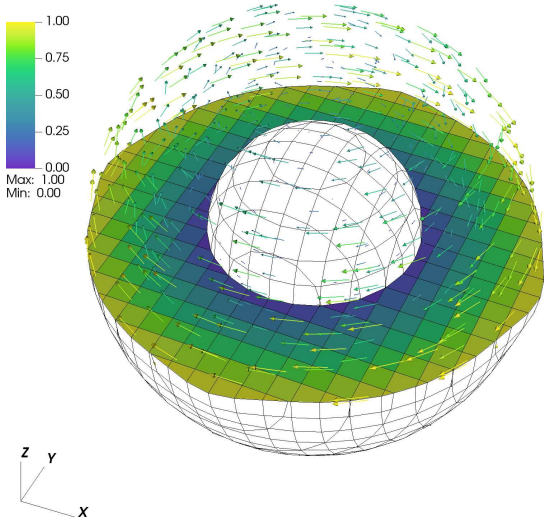


Figure 5.9: The three-dimensional spherical Couette flow at steady state. The velocity vector field is shown, and a slice along the x-y plane shows the grid and velocity magnitude.

5.4.1 Steady Stokes Flow Over a Sphere in a Channel

A square channel is generated with a channel length of 2 in the x-direction, and an inlet length of 1 in the y and z directions. A sphere of radius $r = 0.15$ is centered in the channel at $x = 1$. A developed inflow of peak value 1 is specified at the inflow on the left most boundary, while an outflow is specified for the right most boundary. All other boundary conditions are specified as walls. The Stokes equations are solved to steady state in both two-dimensions, in Figure 5.10, and three-dimensions, in Figure 5.13. Although no analytic solution for this flow can be used for comparison, qualitatively the flow is observed to be highly symmetric and respect the imposed boundary conditions. Notably, the inflow boundary and outflow nearly match, and have streamlines normal to the boundaries as expected. At wall boundaries, solution velocities approach zero, although particularly along the sphere there is some discrepancy due to the projection and viscous operators smearing the third-order boundary solution with the fourth-order interior.

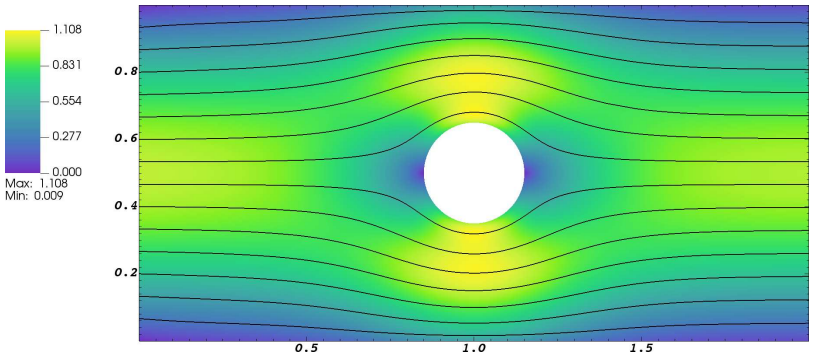


Figure 5.10: Stokes flow for a circle in a channel, where the left boundary is an inlet, the right an outlet, and all other boundaries walls. Streamlines are shown in black, and the contours plot the velocity magnitude.

Convergence rates for the u and v velocity from Richardson extrapolation are plotted in Figure 5.11. L_1 and L_2 norms indicate fourth-order accuracy, but the L_∞ norm lags slightly. The plot of solution error (Figure 5.12) indicates lower accuracy in cut-cells and parts of the domain boundary.

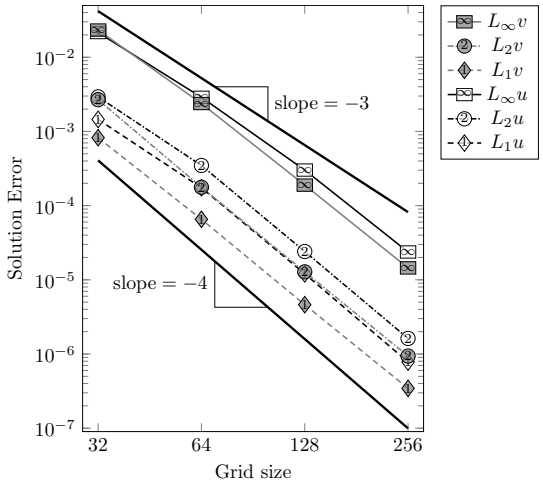


Figure 5.11: Convergence rates of the 2D circle in a channel.

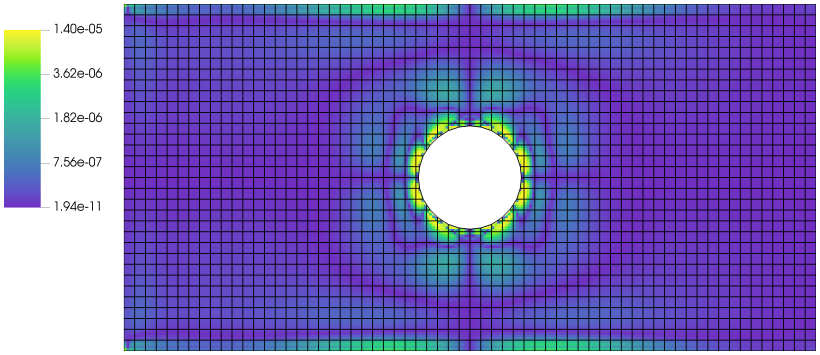


Figure 5.12: Solution errors for the u velocity of the 2D Stokes flow over a circle in a channel on the $h = 1/265$ grid, with a coarsened grid representation.

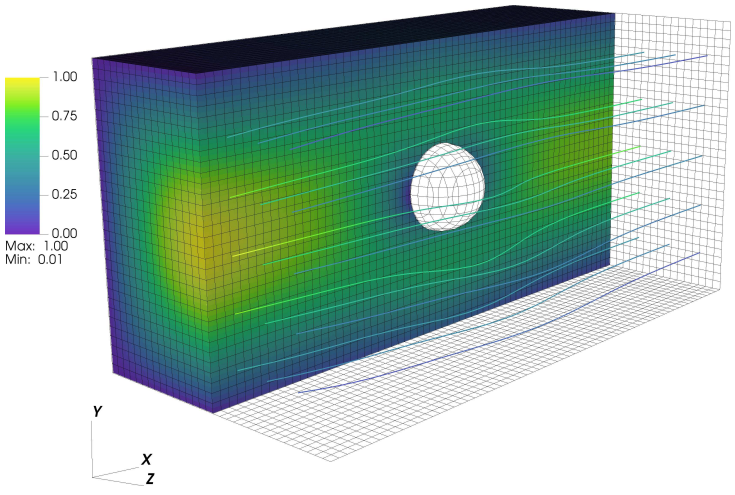


Figure 5.13: Stokes flow over a sphere in a channel, where the left boundary is an inlet, the right an outlet, and all other boundaries walls. The streamlines and contour plot show the velocity magnitude.

5.4.2 Stokes Flow for Bone Scaffolding

Cellular structures have been a topic of interest for a range of bio-inspired materials because their geometries can be quickly adapted to target desired mechanical properties. In the work by Asbai-Ghoudan el at. [79] a structure for bone growth scaffold is analyzed, and one of the key challenges identified when modeling such structures is mesh generation. We construct models of similar geometry using the EB method to significantly simplify the meshing process. The geometry of interest is defined by the gyroid function

$$f(\boldsymbol{x}) = \cos(n\pi x)\sin(n\pi y) + \cos(n\pi y)\sin(n\pi z) + \cos(n\pi z)\sin(n\pi x)\,, \tag{5.8}$$

evaluated in two dimensions along the $z = 0.1$ plane where $f(\boldsymbol{x}) = 0$ with a period of $n = 2$. The surface is approximately thickened to width of $1/6$. Flow is solved through a pipe of radius $r = 0.95$ and length 8, with a cut of the gyroid centered in the pipe at $x = 4$. A cut is made through the centerline with radius $r = 0.175$. The cut boundaries are smoothed (as in Devendran et al. [12]) to prevent singular solutions around sharp exterior corners. Stokes flow through this structure is shown in Figure 5.14 for a grid refinement of $h = 1/512$ in the flow direction.

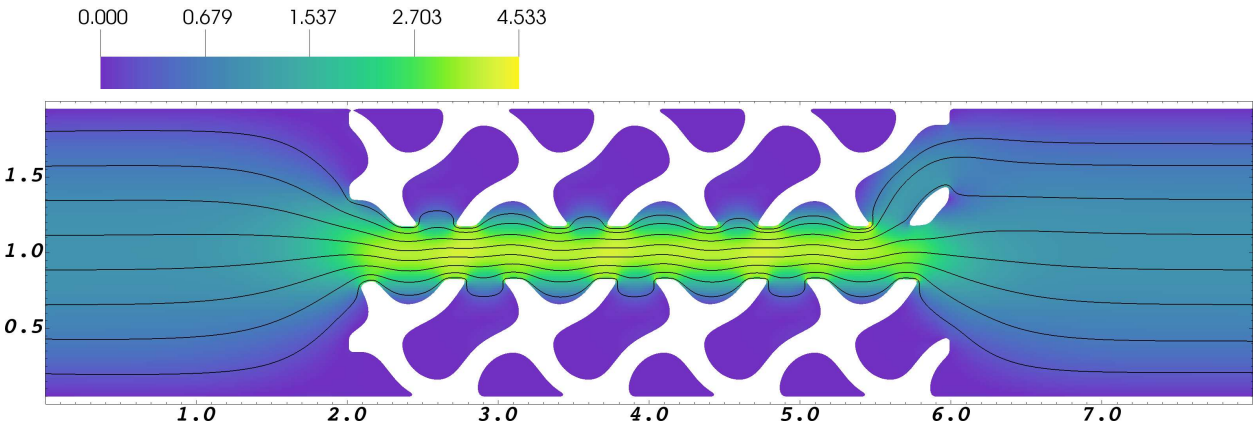
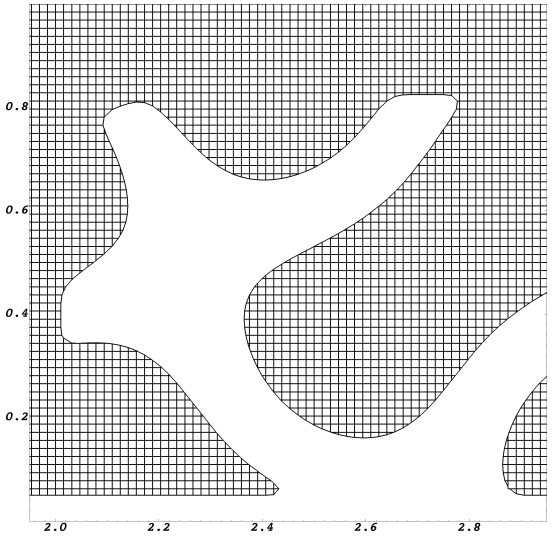
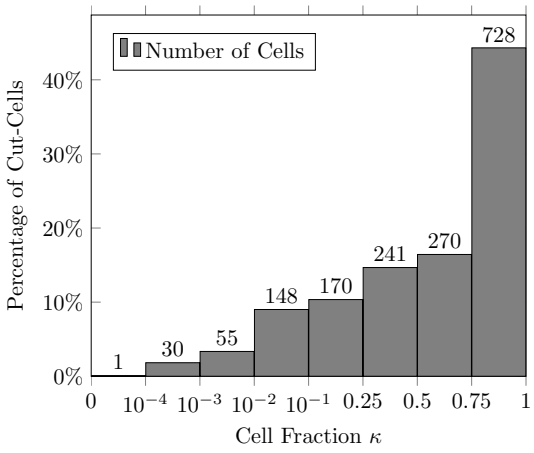


Figure 5.14: Velocity magnitude and streamlines for a two-dimensional representation of the bone scaffold geometry.

The flow is found to be symmetric, and well-behaved along the cut-cell boundaries. The EB grid for this case contains $26\,585$ valid cells, of which 1643 are cut-cells with the smallest volume fraction of $\kappa = 5.933 \cdot 10^{-5}$. A close up view of this grid is shown in Figure 5.15(a), and the distribution of cut-cell sizes is shown in Figure 5.15(b). In particular, the mesh generation requires trivial involvement, and because the geometry is specified analytically, adjustments such as gyroid size, position, and thickness are easily made. This shows promise for more detailed analysis using the high-order EB method, where a large design space of geometries could be examined quickly without special considerations for mesh generation or solution stability.



(a) A close up of the EB grid.



(b) Distribution of the cut-cell sizes.

Figure 5.15: Bone scaffold grid.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

This research has developed three methods to enable complex geometries for high-order FVMs on structured grids with AMR. These methods include the HO-ACR method, the MMB method for arbitrary discrete grids with AMR, and the high-order EB method. The HO-ACR and MMB methods have been implemented in Chord, a fourth-order finite-volume CFD software, and enabled modeling complex fluid dynamics involving turbulence and combustion in the presence of complex geometries. AMR is a foundational feature for capturing multi-scale flows with computational efficiency in Chord. Combusting flows are highly sensitive to changes in refinement, and interpolation overshoots near discontinuities can destabilize the overall simulation. The HO-ACR method has been show as essential for stable chemistry in the presence of AMR by preventing these overshoots. The HO-ACR method has been verified to ensure high-order accuracy for smooth flow problems, preserve bounds for non-smooth ones, and maintain conservation on mapped grids. The MMB method for arbitrary discrete geometries has allowed Chord to model geometries such as the bluff-body combustor. The method is capable of using discrete grids created from traditional meshing tools while still allowing for high-order AMR by constructing mapping functions using B-splines. This differs from some high-order applications which utilize coupled analytic functions with grids. Further, the conforming MMB method is used to provide gridding flexibility. A number of strategies are proposed, implemented, and tested to allow the high-order conforming MMB to handle arbitrary discrete geometries such as sharp corners, although with some limitations. Finally, as an alternative approach to the MMB method, the high-order EB method has been implemented as a standalone code using the Chombo framework. The high-order EB method is verified and validated, and demonstrated to solve Stokes flows for highly complex geometries. Additionally, the method is stable in the presence of small cells, without any form of cell merging or redistribution.

6.2 Original Contributions

This research has made novel contributions, enabling the high-order FVM algorithm to solve flow on complex geometries using structured AMR grids. Original contributions include:

- Development of the HO-ACR method for conservative bounded interpolation of averaged quantities. This method is required for stability of solutions with AMR and stiff physics, such as turbulent chemistry. This method makes approaches for embedded-LES or embedded-DNS possible for high-order finite-volume schemes, and allows greater flexibility in when and where AMR may be applied.

- Demonstration of high-order AMR on discretely defined structured grids by use of a B-spline mapping function. This allows for Chord to utilize grids from industry standard mesh generation tools with no loss of accuracy.

- Development of the high-order conforming multi-block method for arbitrary geometries. This allows complex non-analytic geometries on structured grids, while maintaining permitting AMR and maintaining solution conservation. This work has enabled modeling complex cases such as the bluff-body combustor in Chord.

- Creation of a time dependent fourth-order embedded-boundary method for the Stokes equations. Prior work with embedded-boundary methods has been limited to second-order in time, and fourth-order methods have only previously been shown for the Poisson equation.

6.3 Future Work

There are a number of directions for future work, such as improvements to existing algorithms and new areas of exploration. These include:

- Extending the HO-ACR method to the computational geometries represented by general mapped multi-block domains. This would require having some form of bound-preserving property in the interpolation operation across blocks, where the grid metrics may also be

discontinuous. Interpolation between blocks requires a somewhat different approach for bounds preservation since no reasonable conservation constraint can be maintained.

- The mapped multi-block algorithm as implemented in Chord is restricted by the number of ghost cells used. Many ghost cells make designing grids difficult, and the data exchanges between blocks are costly. To alleviate the ghost cell restriction, the number of ghost cells could be reduced at the expense of more frequent data exchanges. This requires moderate modification to the current algorithm of Chord, but no fundamental changes to the MMB approach.

- Ghost cell restrictions of the MMB method may be removed altogether by instead building a reconstruction function at block interfaces. The reconstruction could then be incorporated into stencils near the block boundaries, and eliminate ghost cells between blocks entirely. This would remove the challenges and complexity of filling ghost cells, but require rethinking the MMB approach and significantly change stencils near boundaries.

- The high-order EB methodology has shown great promise, but has yet to be shown for more complex physics. An immediate goal is to expand the method developed in this work to the full Navier-Stokes equations and utilize AMR. To do so, the most immediate challenge is modeling the non-linear advection term.

- The combination of EB with MMB technologies could ideally provide the advantages of both. This would allow for meshes which are mapped to control grid quality for coarse features, while capable of representing small scale sharp features that only unstructured or EB meshes can capture. This requires additional development of the EB methodology to consider mapped coordinates.

- For both the MMB and EB methods, the choice of stencil greatly impacts the resulting interpolation stencils. Choosing large symmetric stencils as done in this dissertation is effective, but can be expensive to evaluate. Ideally, these stencils could be reduced automatically in

a way that has no adverse effects on the stability and greatly improves computation time. With this goal in mind, stencil selection may be viewed from the angle of a basis pursuit problem. For MMB applications, this approach may be able to detect when grids conform and correctly reduce the stencil interpolation. The EB method could take advantage of this to prune stencils in ways appropriate for the particular operator being evaluated.

Bibliography

[1] J. Slotnick, A. Khodadoust, J. Alonso, D. Darmofal, W. Gropp, E. Lurie, and D. Mavriplis. CFD vision 2030 study: A path to revolutionary computational aerosciences. Technical Report 218178, NASA/CR, 2014.

[2] T. H. Pulliam and D. W. Zingg. *Fundamental Algorithms in Computational Fluid Dynamics*. Scientific Computation. Spinger International Publishing, New Delhi, India, 1 edition, 2014.

[3] Z. J. Wang, K. Fidkowski, R. Abgrall, F. Bassi, D. Caraeni, A. Cary, H. Deconinck, R. Hartmann, K. Hillewaert, H. T. Huynh, N. Kroll, G. May, P.-O. Persson, B. van Leer, and M. Visbal. High-order CFD methods: current status and perspective. *Int. J. Numer. Methods Fluids*, 72(8):811–845, 2013.

[4] P. McCorquodale and P. Colella. A high-order finite-volume method for conservation laws on locally refined grids. *Comm. App. Math. Comput. Sci.*, 6(1):1–25, 2011.

[5] H. Lomax, T. H. Pulliam, and D. W. Zingg. *Fundamentals of Computational Fluid Dynamics*. Scientific Computation. Springer-Verlag Berlin Heidelberg, 1 edition, 2001.

[6] S. M. Guzik, P. McCorquodale, and P. Colella. A freestream-preserving high-order finite-volume method for mapped grids with adaptive-mesh refinement. In *50th AIAA Aerospace Sciences Meeting*, number AIAA 2012-0574. AIAA, 2012. https://doi.org/10.2514/6.2012-574.

[7] P. Colella, M. R. Dorr, J. A. F. Hittinger, and D. F. Martin. High-order finite-volume methods in mapped coordinates. *J. Comput. Phys.*, 230:2952–2976, 2011.

[8] S. M. Guzik, X. Gao, L. D. Owen, P. McCorquodale, and P. Colella. A freestream-preserving fourth-order finite-volume method in mapped coordinates with adaptive-mesh refinement. *Comput. Fluids*, 123:202–217, 2015.

[9] W. D. Henshaw and D. W. Schwendeman. Moving overlapping grids with adaptive mesh refinement for high-speed reactive and non-reactive flow. *J. Comput. Phys.*, 216:744–779, 2006.

[10] P. McCorquodale, M. R. Dorr, J. A. F. Hittinger, and P. Colella. High-order finite-volume methods for hyperbolic conservation laws on mapped multiblock grids. *J. Comput. Phys.*, (288):181–195, May 2015.

[11] P. Schwartz, J. Percelay, T. Ligocki, H. Johansen, D. Graves, D. Devendran, P Colella, and E. Ateljevich. High-accuracy embedded boundary grid generation using the divergence theorem. *Communications in Applied Mathematics and Computational Science*, 10(1):83 – 96, 2015.

[12] D. Devendran, D. T. Graves, H. Johansen, and T. Ligocki. A fourth-order Cartesian grid embedded boundary method for Poisson's equation. *Comm. App. Math. and Comp. Sci.*, 12(1):51–79, 2017.

[13] S. M. Guzik, X. Gao, and C. Olschanowsky. A high-performance finite-volume algorithm for solving partial differential equations governing compressible viscous flows on structured grids. *Comput. Math Appl.*, 72:2098–2118, 2016.

[14] T.J. Barth and P.O. Frederickson. High-order solution of the Euler equations on unstructured grids using quadratic reconstruction. Paper AIAA-90-0013, AIAA, 1990.

[15] A. Harten, B. Engquist, S. Osher, and S. Chakravarthy. Uniformly high order essentially non-oscillatory schemes iii. *J. Comput. Phys.*, 71:231–303, 1987.

[16] G. Jiang and C.-W. Shu. Efficient implementation of weighted ENO schemes. *J. Comput, Phys.*, 126:202–228, 1996.

[17] C.-W. Shu. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. *Advanced numerical approximation of nonlinear hyperbolic*

equations (eds B. Cockburn, C. Johnson, C.-W. Shu, E. Tadmor, A. Quarteroni), 1697:325–432, 1998.

[18] B. Van Leer. Towards the ultimate conservative difference scheme. IV. A new approach to numerical convection. *J. Comput. Phys.*, 23(3):276–299, 1977.

[19] P. Colella and M. Sekora. A limiter for PPM that preserves accuracy at smooth extrema. *J. Comput. Phys.*, 227(15):7069–7076, 2008.

[20] *Solving Linear Equations*, chapter 12, pages 211–245. John Wiley and Sons, Ltd, 2008.

[21] C. Kennedy and M. Carpenter. Additive Runge-Kutta schemes for convection-diffusion-reaction equations. *Applied Numerical Mathematics*, 44:139–181, 2003.

[22] X. Gao and C. P. T. Groth. A parallel adaptive mesh refinement algorithm for predicting turbulent non-premixed combusting flows. *Int. J. Comut. Fluid Dyn.*, 20(5):349–357, 2006.

[23] X. Gao. *A parallel solution-adaptive method for turbulent non-premixed combusting flows*. PhD thesis, University of Toronto, 2008.

[24] X. Gao and C. P. T. Groth. A parallel solution-adaptive method for three-dimensional turbulent non-premixed combusting flows. *J. Comput. Phys.*, 229:3250–3275, 2010.

[25] X. Gao, S. Northrup, and C.P.T. Groth. Parallel solution-adaptive method for two-dimensional non-premixed combusting flows. *Progress in Computational Fluid Dynamics, An International Journal*, 11(2):76–95, 2011.

[26] X. Gao, L. D. Owen, and S. M. Guzik. A high-order finite-volume method for combustion. AIAA 2016-1808, 54th AIAA Aerospace Sciences Meeting, 2016.

[27] L. Owen, S. Guzik, and X. Gao. A fourth-order finite-volume algorithm for compressible flow with chemical reactions on mapped grids. (AIAA 2017-4498), 2017.

[28] L. D. Owen, S. M. Guzik, and X. Gao. A high-order adaptive algorithm for multispecies gaseous flows on mapped domains. *Comput. Fluids*, 170:249–260, 2018.

[29] L. D. Owen, X. Gao, and S. M. Guzik. Techniques for improving monotonicity in a fourth-order finite-volume algorithm solving shocks and detonations. *J. Comput. Phys.*, 415, 2020.

[30] Y. Wang. *Bayesian Data Assimilation for CFD Modeling of Turbulent Combustion*. Ph.D. dissertation, Colorado State University, 2022.

[31] N. Zettervall, K. Nordin-Bates, E. J. K. Nilsson, and C. Fureby. Large eddy simulation of a premixed bluff body stabilized flame using global and skeletal reaction mechanisms. *Combust. Flame*, 179:1–22, 2017.

[32] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22(104):745–745, 1968.

[33] J. B. Bell, P. Colella, and H. M. Glaz. A second-order projection method for the incompressible Navier-Stokes equations. *J. Comput. Phys.*, 85(2):257–283, December 1989.

[34] D. F. Martin, P. Colella, and D. Graves. A cell-centered adaptive projection method for the incompressible Navier–Stokes equations in three dimensions. *J. Comput. Phys.*, 227(3):1863–1886, 2008.

[35] D. Trebotich and D. T. Graves. An adaptive finite volume method for the incompressible Navier-Stokes equations in complex geometries. *Comm. App. Math. and Comp. Sci.*, 10(1):43–82, 2015.

[36] X. Gao, S. M. J. Guzik, and P. Colella. A fourth-order boundary treatment for viscous fluxes on Cartesian grid finite-volume methods. AIAA 2014-1277, 52nd AIAA Aerospace Sciences Meeting, 2014.

[37] X. Gao and S. M. J. Guzik. A fourth-order scheme for the compressible Navier-Stokes equations. AIAA 2015-0298, 53rd AIAA Aerospace Sciences Meeting, 2015.

[38] X. Gao, L. D. Owen, and S. M. J. Guzik. A parallel adaptive numerical method with generalized curvilinear coordinate transformation for compressible Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 82:664–688, 2016.

[39] M. Adams, P. Colella, D. T. Graves, J. N. Johnson, H. S. Johansen, N. D. Keen, T. J. Ligocki, D. F. Martin, P. W. McCorquodale, D. Modiano, P. O. Schwartz, T. D. Sternberg, and B. Van Straalen. Chombo software package for AMR applications—design document. Technical Report LBNL-6616E, Lawrence Berkeley National Laboratory, 2015.

[40] P. Colella, D. T. Graves, T. J. Ligocki, G. Miller, D. Modiano, P. Schwartz, B. V. Straalen, J. Pillod, D. Trebotich, and M. Barad. EBChombo software package for Cartesian grid, embedded boundary application. Technical Report LBNL-6615E, Lawrence Berkeley National Laboratory, 2014.

[41] D. J. Hill, C. Pantano, and D. I. Pullin. Large-eddy simulation and multi-scale modelling of a Richtmyer-Meshkov instability with reshock. *J. Fluid Mech.*, 557:29–61, 2006.

[42] W. Gao, W. Zhang, W. Cheng, and R. Samtaney. Wall-modelled large-eddy simulation of turbulent flow past airfoils. *J. Fluid Mech.*, 873:174–410, 2019.

[43] S. Walters, X. Gao, H. Johansen, and S. Guzik. Assessing stretched-vortex subgrid-scale models in finite volume methods for unbounded turbulent flows. Accepted for the Journal of Flow,Turbulence and Combustion, 2019.

[44] D. T. Graves, P. Colella, D. Modiano, J. Johnson, B. Sjogreen, , and X. Gao. A Cartesian grid embedded boundary method for the compressible Navier Stokes equations. *Comm. App. Math. Comp. Sci.*, 8(1):99–122, 2013.

[45] T. Ligocki, P. Schwartz, J. Percelay, and P. Colella. Embedded boundary grid generation using the divergence theorem, implicit functions, and constructive solid geometry. *Journal of Physics: Conference Series*, 125, 2008.

[46] J. Holgate, A. Skillen, T. Craft, and A. Revell. A review of embedded large eddy simulation for internal flows. *Arch. Comput. Method. E.*, pages 1–18, 2018.

[47] C. Michalak and C. Ollivier-Gooch. Accuracy preserving limiter for the high-order accurate solution of the Euler equations. *J. Comput. Phys.*, 228(23):8693–8711, dec 2009.

[48] R. Anderson, V. Dobrev, Tz. Kolev, D. Kuzmin, M. Quezada de Luna, R. Rieben, and V. Tomov. High-order local maximum principle preserving (MPP) discontinuous galerkin finite element method for the transport equation. *J. Comput. Phys.*, 334:102 – 124, 2017.

[49] L. Ivan, H. De Sterck, A. Susanto, and C.P.T. Groth. High-order central ENO finite-volume scheme for hyperbolic conservation laws on three-dimensional cubed-sphere grids. *J. Comput. Phys.*, 282:157 – 182, 2015.

[50] L. Freret, L. Ivan, H. Sterck, and C. P. Groth. High-order finite-volume method with block-based AMR for magnetohydrodynamics flows. *J. Sci. Comput.*, 79(1):176–208, apr 2019.

[51] J. Hilditch and P. Colella. A projection method for low mach number fast chemistry reacting flow. Paper AIAA 97-0263, AIAA, January 1997.

[52] S. M. J. Guzik, X. Gao, T. Weisgraber, B. Alder, and P. Colella. An adaptive mesh refinement strategy with conservative space-time coupling for the lattice-Boltzmann method. In *51st AIAA Aerospace Sciences Meeting*, number AIAA 2013-0866. AIAA, 2013. https://doi.org/10.2514/6.2013-866.

[53] R. W. Anderson, V. A. Dobrev, Tz. V. Kolev, and R. N. Rieben. Monotonicity in high-order curvilinear finite element arbitrary Lagrangian–Eulerian remap. *Int. J. Numer. Meth. Fl.*, 77(5):249–273, 2015.

[54] G. Billet, V. Giovangigli, and G. de Gassowski. Impact of volume viscosity on a shock–hydrogen-bubble interaction. *Combust. Theory Model.*, 12(2):221–248, 2008.

[55] J. Christopher, X. Gao, and S.Guzik. High-order implicit-explicit additive Runge-Kutta schemes for numerical combustion with adaptive mesh refinement. Int. J. Numer. Meth. Fl., 2021. Accepted for publication.

[56] B. Paxton, C. Fugger, B. Rankin, and A. Caswell. Development and characterization of an experimental arrangement for studying bluff-body-stabilized turbulent premixed propane-air flames. Number AIAA 2019-0118. AIAA Scitech 2019 Forum, January 2019. https://doi.org/10.2514/6.2019-0118.

[57] A. Comer, M. Ihme, C. Li, S. Menon, J. Oefeline, B. Rankin, V. Sankaran, and V. Sankaran. Proceedings of the third model validation for propulsion (MVP 3) workshop, January 2019. AIAA SciTech Forum.

[58] https://www.pointwise.com/index.html.

[59] W. D. Henshaw. *Automatic grid generation*. Acta Numerica, 1996. 5:121-148.

[60] W. D. Henshaw. Overture: An object-oriented system for solving PDEs in moving geometries on overlapping grids. In L. Sakell and D. D. Knight, editors, *First AFOSR Conference on Dynamic Motion CFD*, volume 10, pages 281–290, 1996.

[61] E. Steinthorsson, D. Modiano, and P. Colella. Computations of unsteady viscous compressible flows using adaptive mesh refinement in curvilinear body-fitted grid systems. Technical report, 25th AIAA Fluid Dynamics Conference, 1994.

[62] C. De Boor. *A practical guide to splines; rev. ed.* Applied mathematical sciences. Springer, Berlin, 2001.

[63] L. Piegl and W. Tiller. *The NURBS Book (2nd Ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[64] A. G. Kravchenko, P. Moin, and K. Shariff. B-spline method and zonal grids for simulations of complex turbulent flows. *J. Comput. Phys.*, 151(2):757 – 789, 1999.

[65] J. L. Blanco and P. K. Rai. nanoflann: a C++ header-only fork of FLANN, a library for nearest neighbor (NN) with kd-trees. https://github.com/jlblancoc/nanoflann, 2014.

[66] M. Aftosmis, M. Berger, and G. Adomavicius. *A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries*. 2000.

[67] K. J. Richards, P. K. Senecal, and E. Pomraning. Converge 3.0*. Convergent Science, Madison, WI, 2021.

[68] M. J. Berger and M. J. Aftosmis. Progress towards a Cartesian cut-cell method for viscous compressible flow. In *50th AIAA Aerospace Sciences Meeting*, number AIAA 2012-1301. AIAA, 2012.

[69] M. E. Khalili, M. Larsson, and B. Müller. High-order ghost-point immersed boundary method for viscous compressible flows based on summation-by-parts operators. *Int. J. Numer. Meth. Fl.*, 89(7):256–282, 2019.

[70] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[71] P. Colella and D. P. Trebotich. Numerical simulation of incompressible viscous flow in deforming domains. *Proceedings of the National Academy of Sciences*, 96(10):5378–5381, 1999.

[72] U. M. Ascher, S. J. Ruuth, and R. J. Spiteri. Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. *Applied Numerical Mathematics*, 25(2):151 – 167, 1997. Special Issue on Time Integration.

[73] D. F. Martin and K. L. Cartwright. Solving Poisson's equation using adaptive mesh refinement. Technical Report UCB/ERI M96/66, University of California, Berkeley, 1996.

[74] S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, D. A. May, L. Curfman McInnes, R. T.

Mills, T. Munson, K. Rupp, P. Sanan, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.9, Argonne National Laboratory, 2018.

[75] S. Balay, W. D. Gropp, L. Curfman McInnes, and B. F. Smith. Efficient management of parallelism in object oriented numerical software libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[76] Q. Zhang, H. Johansen, and P. Colella. A fourth-order accurate finite-volume method with structured adaptive mesh refinement for solving the advection-diffusion equation. *SIAM J. Sci. Comput.*, 34:B179–B201, 2012.

[77] H. Johansen and P. Colella. A Cartesian grid embedded boundary method for Poisson's equation on irregular domains. *J. Comput. Phys.*, 147:60–85, 1998.

[78] K. Masatsuka. *I do like CFD, VOL.1, Second Edition*. Number v. 1. K. Masatsuka, 2013.

[79] R. Asbai-Ghoudan, S. Ruiz de Galarreta, and N. Rodriguez-Florez. Analytical model for the prediction of permeability of triply periodic minimal surfaces. *Journal of the Mechanical Behavior of Biomedical Materials*, 124:104804, 2021.