NUMERICAL STUDY AND IMPROVEMENT OF THE METHODS IN UINTAH FRAMEWORK: THE MATERIAL POINT METHOD AND THE IMPLICIT CONTINUOUS-FLUID EULERIAN METHOD

by

Lethuy Thi Tran

A dissertation submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computing

School of Computing

The University of Utah

December 2012

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Lethuy Thi Tran**

has been approved by the following supervisory committee members:

Martin Berzins	, Chair	**10/08/2012** Date Approved
Robert M. Kirby	, Member	**10/04/2012** Date Approved
Christopher R. Johnson	, Member	**10/04/2012** Date Approved
Todd Harman	, Member	**10/05/2012** Date Approved
Andrej V. Cherkaev	, Member	**10/04/2012** Date Approved

and by **Alan Davis** , Chair of

the Department of **School of Computing**

and by Charles A. Wight, Dean of The Graduate School.

ABSTRACT

The Material Point Method (MPM) and the Implicit Continuous-fluid Eulerian method (ICE) have been used to simulate and solve many challenging problems in engineering applications, especially those involving large deformations in materials and multimaterial interactions. These methods were implemented within the Uintah Computational Framework (UCF) to simulate explosions, fires, and other fluids and fluid-structure interaction. For the purpose of knowing if the simulations represent the solutions of the actual mathematical models, it is important to fully understand the accuracy of these methods. At the time this research was initiated, there were hardly any error analysis being done on these two methods, though the range of their applications was impressive. This dissertation undertakes an analysis of the errors in computational properties of MPM and ICE in the context of model problems from compressible gas dynamics which are governed by the one-dimensional Euler system. The analysis for MPM includes the analysis of errors introduced when the information is projected from particles onto the grid and when the particles cross the grid cells. The analysis for ICE includes the analysis of spatial and temporal errors in the method, which can then be used to improve the method's accuracy in both space and time. The implementation of ICE in UCF, which is referred to as Production ICE, does not perform as well as many current methods for compressible flow problems governed by the one-dimensional Euler equations – which we know because the obtained numerical solutions exhibit unphysical oscillations and discrepancies in the shock speeds. By examining different choices in the implementation of ICE in this dissertation, we propose a method to eliminate the discrepancies and suppress the nonphysical oscillations in the numerical solutions of Production ICE – this improved Production ICE method (IMPICE) is extended to solve the multidimensional Euler equations. The discussion of the IMPICE method for multidimensional compressible flow problems includes the method's detailed implementation and embedded boundary implementation. Finally, we propose a discrete adjoint-based approach to estimate the spatial and temporal errors in the numerical solutions obtained from IMPICE.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the help from many people on the path to where I am today. I may not individually thank many of you in the following note, but I will always remember your support.

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Martin Berzins, whose supervision, advice, and guidance helped me to have a thorough understanding of the subject. In so doing, he still left room for my own ideas and let me grow to be an independent researcher. He has given me a number of valuable insights which are useful for not only my graduate school but also my future career. In various ways, he provided me necessary encouragement and support in overcoming obstacles. I am truly inspired by his intuition and passion for science which have made a huge impact on the decisions of my future career. Above all and the most needed, he always showed sympathy for all the life events that happened to me outside school that needed my immediate attention. I consider myself very fortunate to be his student.

My deepest gratitude is also due to the members of the supervisory committee: Prof. Mike Kirby, Prof. Christopher Johnson, Prof. Todd Harman, and Prof. Andrej Cherkaev. I especially thank Prof. Harman for many helpful discussions.

I would like to thank my parents, Le Hung Tran and Huong Tran, for their unending love, trust, and support. My parents hardly had any opportunities to go to school, but they always reminded me how important education is. I would not have made it to this far if it were not for them. I would also like to thank my parents-in-law, Vo Luong and Muoi Nguyen, for giving me help when needed. Without their support and understanding, it would be hard for me to pursue my graduate studies. I would also like to thank my uncle and aunt Alex and Jeanette Tam. For what they have done for me, I am forever indebted. I also thank my sister and brother for their caring and loving support.

Most importantly, I would like to thank my husband, Phong, and my children, Kha and San, for being by my side for this long journey to share the joys and happinesses and to endure the hardships. They have sacrified much for my success. To my beloved husband and children: “You all are far more important to me than any success and have made my life much more complete.”

Lastly, I would like to thank all of my friends and relatives – the list is too numerous to mention all by name – who have supported me throughout the years. I can not imagine how stressful my life would be without them.

This work was supported by the U.S. Department of Energy through the Center for the Simulation of Accidental Fires and Explosions (C-SAFE) under grant W-7405-ENG-48.

CHAPTER 1

INTRODUCTION

The last few decades have seen a significant increase in the use of numerical methods in many research areas. Numerical methods are no longer merely simulation tools; they are now used to study and understand phenomena represented as mathematical models. A broad range of physical processes in engineering applications are simulated and studied using numerical methods; for example, micromechanics of heterogeneous materials [9]; deformation processes in energetic materials [14]; large deformation fluid-structure interaction [52, 44]; aerodynamics of vocal fold movement [31]; explosions of energetic devices [45]; nano-scale magnetization dynamics [59]; accidental fires and explosions [56] and densification of real open-celled foam microstructures [19], to name just a few. These examples clearly demonstrate the need to use numerical methods to solve increasingly complex problems. At the Center for the Simulation of Accidental Fires and Explosions (C-SAFE) at the University of Utah – created through the Advanced Simulation and Computing Program of the Department of Energy – numerical methods were used to provide state-of-the-art, science-based tools for the numerical simulation of accidental fires and explosions [56]; methods included in the Uintah Computational Framework (UCF) [40, 56, 95, 96]. These methods are the product of more than a decade of cutting-edge research.

The UCF was developed by a number of highly skilled researchers to provide a software system for simulating complex physical phenomena [88], such as reacting flows, material properties, and multimaterial interactions. There are four main simulation slgorithms in the UCF: the ARCHES simulation code, the Implicit Continuous-fluid Eulerian method (ICE), the Material Point Method (MPM), and an integrated combination of MPM and ICE (MPMICE). Each simulation component facilitates the solution of partial differential equations on structured adaptive mesh refinement grids using hundreds to thousands of processors. Each component is specifically designed for solving certain types of problems. Specifically, the ARCHES component, a finite-volume incompressible flow C.F.D solver, was initially designed for predicting the heat flux from large buoyant pool fires, and then later was extended for solving many industrial relevant problems. ICE is for compressible flows; MPM is for solids; MPMICE is for fluid-structure. These methods have been used to simulate a wide range of applications. Examples of simulated applications using MPM include biomechanics of microvessels, effects of wounding on heart tissue, and the properties of foam under large deformation [19]; densification of foam [6]; compression of wood [94]; large-scale complex fluid-structure interactions arising from the modeling of safety studies involving explosions [56, 97]; sea ice dynamics [115]; and energetic device explosions [45]. Examples of simulated applications using ICE include fluidized dust beds, the flow of a liquid with entrained bubbles, atmospheric condensation with the fall of precipitation, the expansion and compression of a bubble formed by highly explosive gases under water, dynamics resulting from intense atmospheric explosions from the early time highly compressible flow [51, 68]; and explosions, fires, and other fluid and fluid-structure interaction phenomena [45]. Finally, examples of simulated applications using MPMICE include modeling of explosives and their interaction with solid structures, modeling of blast and soil [117], and modeling of the material dynamics and aerodynamics of phonation [31].

Given that MPM, ICE, and MPMICE are used on such challenging problems, it is important to fully understand the accuracy of these methods and to have these methods accurately simulate the solutions of the actual mathematical models. At the time this research was initiated in 2005, there were hardly any error analyses being done on any of these methods. Thus our aim in this dissertation is to provide a numerical study of some variation of these methods. In addition, we propose an improvement to the version of ICE that is currently implemented in the Uintah Computational Framework, including an estimate of the errors in the improved version. Several other researchers at C-SAFE were working simultaneously on MPM and ICE, and this dissertation both builds upon and complements their work.

1.1 Contributions

By providing a numerical study of and improvement to the MPM and ICE algorithms, this dissertation makes the following contributions:

- *A study of errors in numerical solutions that are obtained from a variation of MPM which we specifically proposed for gas dynamics.* We undertake an analysis of MPM in modeling compressible flow problems governed by the one-dimensional Euler equations. Though this analysis is done for a specific variation of MPM, it is described in a way that can be applied to other versions of MPM. In this analysis, we focus on two sources of error: errors introduced when information from particles is projected onto the grid, as well as errors introduced when particles cross grid cells. In addition to the analysis obtained from studying the algorithm of the method, we include results obtained from observing the method's performance on the Sod's shocktube problem. These aforementioned contributions are presented in Chapter 5 and were reported in the published article: "Solving Time-Dependent PDEs using the Material Point Method, A Case Study from Gas Dynamics," L.-T. Tran, J. Kim, and M. Berzins, International Journal for Numerical Methods in Fluids, Volume 62, Issue 7, pages 709–732, Copyright ©2009 John Wiley & Sons, Ltd. [123].

- *An improvement to the Production Implicit Continuous-fluid Eulerian Method (Production ICE) in the Uintah Computational Framework for the case of one-dimensional compressible flows, including an error study of this improved version.* The implementation of Production ICE depends upon choosing among several different implementation choices of the cell-centered ICE method proposed by Kashiwa *et al.* [68]. We will explore such different implementation choices of the cell-centered ICE method of Kashiwa *et al.* [68] in order to propose an improved version of Production ICE. This Improved Production ICE method (IMPICE) aims to eliminate the unphysical oscillations in the numerical solutions to the one-dimensional compressible flow problems. Of comparable importance to having a nonoscillating numerical solution is determining the accuracy of the IMPICE method in time and space – which we study both theoretically and numerically. We can increase the orders of accuracy in time and space by applying a high order time discretization and a nonlinear spatial discretization respectively. These aforementioned contributions are presented in Chapter 6 and were also reported in the published article: "Improved Production Implicit Continuous-fluid Eulerian Method for Compressible Flow Problems in Uintah," L.-T. Tran, and M. Berzins, International Journal for Numerical Methods in Fluids, Volume 69, Issue 5, pages 926–965, Copyright ©2012 John Wiley & Sons, Ltd. [122].
- *An extension of IMPICE to solve multidimensional compressible flow problems, including the implementation of boundary conditions.* The IMPICE method shows great improvement in its accuracy and ability to capture discontinuities when solving one-dimensional compressible flow problems. We propose the extension of IMPICE to solve the compressible problems governed by the system of Euler equations in multidimensional space. In this proposed IMPICE method's detail implementation is the implementation of boundary conditions, including the embedded boundary treatment which enables the solution of problems with potentially complex geometries. These aforementioned contributions are presented in Chapter 7 and will be submitted for publication in the future.
- *An error estimate of the IMPICE Method for one-dimensional compressible flow problems.* We formulate a discrete adjoint-based approach for estimating spatial and temporal errors in the numerical solutions of the time-dependent partial differential equations

(PDEs). We appropriate and modify the adjoint global error estimate discussed in Cao and Petzold [21] to formulate our adjoint-based approach, which is then tested against the numerical solutions obtained via the Backward Differentiation method (BDF) of the ordinary differential equations (ODEs) and PDEs defined in this dissertation. Finally, we apply our adjoint-based error estimate to the estimation of the temporal and spatial errors in the IMPICE's numerical solutions for one-dimensional compressible flow problems. These aforementioned contributions are presented in Chapter 4 and Chapter 8 and will be submitted for publication in the future.

1.2 Content

This dissertation is organized as follows. Chapter 2 reviews the background and relevant work supporting the MPM and ICE methods used in the Uintah Computational Framework. Chapter 3 provides an overview of the compressible flow problems which includes the necessary mathematical formula and physical quantities for the discussion of MPM and ICE. In Chapter 4, we include the discrete adjoint-based approach for estimating spatial and temporal errors for the method-of-lines PDEs. In Chapter 5, we introduce a variation of the MPM developed for gas dynamics and provide an in-depth study of the method's accuracy properties on a well-known test problem in one dimensional space. In Chapter 6, we examine the different implementation choices in the cell-centered ICE method and propose the IMPICE method, which is an improvement to Production ICE for one-dimensional compressible flow problems. We also include in Chapter 6 an error analysis of IMPICE. Chapter 7 discusses a generalization of this method to work with the multidimensional compressible flow problems including the implementation of boundary conditions is presented. In Chapter 8, we estimate the errors in IMPICE's numerical solutions for one-dimensional compressible flow problems using the discrete adjoint-based approach. Chapter 9 summarizes the work presented in this dissertation and conclusions are discussed.

CHAPTER 2

BACKGROUND AND RELEVANT WORK

Computational Fluid Dynamics (CFD) is an area of computational science that uses numerical methods and algorithms to solve a broad range of physical processes in engineering applications involving the motion of liquids and gases. Several examples of physical processes and engineering applications which are solved by CFD include simulating flows around automobile surfaces and airplane wings; simulating fluid-structure interaction; simulating manufacturing processes; calculating forces and moments on aircraft; determining the mass flow rate of petroleum through pipelines; predicting weather patterns; understanding nebulae in interstellar space; and reportedly modeling fission weapon detonation [1]. Much current research in CFD is devoted to improving upon early (and still fundamental) methods developed by the Fluid Dynamics Group (T-3) at Los Alamos National Laboratory (LANL). For example, the Material Point Method (MPM) and the Implicit Continuous Eulerian (ICE) method used in modern simulation were adopted from the early CFD methods of T-3 at LANL. The early precursors of MPM and ICE include the Particle-In-Cell (PIC) method of Harlow and Evans [46, 35], the Marker-And-Cell (MAC) method of Harlow and Welch [47, 48], and the Implicit Continuous-fluid Eulerian (ICE) method of Harlow and Amsden [49, 50, 51]. While these early precursors are used in the field of computational fluid dynamics, the MPM method, which resulted from reformulating and modifying PIC, is for use in computational solid dynamics.

MPM is one of the fairly new computational methods – among these new computational methods are meshfree and particle methods as surveyed by Li and Liu [77] – that solve problems involving large deformations in materials. For a comprehensive history of MPM, see the doctoral dissertation of Steffen [109]. Here we list only a selection relevant developments of MPM to provide the necessary background for our current MPM work. The first variation of MPM, is also referred to as the original MPM, introduced by Sulsky *et al.*, [113, 114], may, perhaps, be described as a quasi-meshless method. This MPM's variation has evolved from the Particle-In-Cell (PIC) and Fluid-Implicit-Particle (FLIP) methods [15] originally developed by Brackbill *et al.*; see [17] and the references within. These two methods and their important theoretical results are discussed by Grigoryev *et al.* [43]. One of the fundamental aspects of PIC methods is a discretization of a material into particles, and the interpolation of information from particles to grids and vice-versa. Evolving from PIC, MPM is a mixed Lagrangian-Eulerian method with moving particles on a background grid. The particles are used to represent the Lagrangian state of a material, and the equations of motion are solved on the background grid. In MPM, the Lagrangian particles (or points) are used to discretize the volume of the fluid or solid. These material points carry with them properties such as mass, velocity, stress, strain, and so on. The background grid in MPM is used as a scratchpad for calculations; hence, MPM has a quasi-meshless characterization. An important feature of MPM is its capability to model solid materials undergoing large deformation. Bardenhagen *et al.* [5] later proposed their variation of MPM which is referred to as the Generalized Interpolation Material Point method (GIMP) to provide a general formulation covering MPM methods. These previously mentioned methods - PIC, MPM, and GIMP - use similar approaches to Smoothed Particle Hydrodynamics (SPH) to solve the governing equations [109]. A comparison between MPM and SPH has been undertaken by Ma *et al.* [83].

MPM has not yet been subjected to as much analysis as many of the methods surveyed by Li and Liu [77]. Prior to our research, the significant contribution to the analysis of MPM was the analysis of time integration errors of Bardenhagen [4]. Still, projection errors of MPM remained to be addressed. Though Vshivkov [135] provided a detailed analysis of the projection errors for PIC, the difference between shape functions in PIC and MPM makes the straightforward application of these results to MPM difficult [109]. In 2008, while we were working on our analysis of MPM, Steffen *et al.* [110, 111] published an analysis on quadrature errors and some of the spatial integration errors of the original MPM method. Their analysis studied different contributing factors in the errors introduced by the quadrature employed in the method's algorithm, and examined the spatial integration errors in internal force due to these quadrature errors. Although the original MPM was designed for solid mechanics problems, we were performing our analysis on a variation of the method proposed for gas dynamics in the context of a shock propagation problem. The shock propagation problem for compressible gas dynamics has the advantage of being sufficiently simple to allow analysis of the method. This problem has also been studied by Brackbill [20], Sulsky [113], York *et al.* [140], and very recently in the context of SPH methods by Brown *et al.* [18]. Furthermore, the shock propagation problem's analytical solution makes it possible to evaluate the various sources of error in our variation of MPM. In the present analysis of our MPM's variation, we focused on two sources of error: errors introduced when information from particles is projected onto the grid, as well as errors introduced when particles cross grid cells. With the aforementioned advantages of the settings for our study of MPM, we aimed to provide an analysis of all error sources in our variation of the method and would later apply them to other variations. The result of this analysis was published in 2009 by Tran *et al.* [123]. Then again in 2010, Steffen *et al.* [112] published another analysis of MPM in which they considered our analysis of temporal errors when integrating past a jump in continuity of the velocity field for their analysis on the impact of spatial quadrature errors on time stepping.

With the availability of several analyses for MPM as mentioned above, we turned our focus on the analysis of ICE when looking at the potential combination of MPM and ICE in simulation; see [31, 117] for this potential combination of these two methods in simulation. At the time, the analysis of ICE also had not received much attention.

While MPM is often used in computational solid dynamics, ICE is often used to simulate fluid dynamics. The ICE method was developed by Harlow and Amsden in 1968 [49] with the aim of calculating the compressible flows in all velocity ranges. With the use of semi-implicit time discretization, in which the acoustic waves are treated implicitly while the advection terms are treated explicitly, the method can remove the Courant stability limitation based on the speed of sound in the fluid. According to Harlow and Amsden [49], this is a numerically stable and efficient method for calculating transient, viscous fluid flows in several space dimensions. In 1971, Harlow and Amsden [50] simplified the method and also greatly extended its scope of applicability. There are several improved versions of the ICE method using a pressure-correction solution procedure as seen in [22, 63, 62, 102, 128], and one typical pressure-correction method is referred as PISO (Pressure Implicit with Splitting of Operations). ICE was first developed to simulate single-phase fluid dynamics problems. It was later extended by Harlow and Amsden in 1975 [51] and Kashiwa *et al.* in 1994 [68] to work with multiphase flow simulations. The ICE method by Harlow and Amsden used a staggered grid with normal velocity components at cell faces and all other variables at cell centers. The cell-centered ICE method by Kashiwa *et al.* [68] uses a different approach from the previous ICE method proposed by Harlow and Amsden. As mentioned in Kashiwa and Lee [67], Kashiwa *et al.* [68] uses a nonstaggered grid in an ongoing effort to deal with the difficulties of the ICE method with staggered grid. The main difficulties in the use of the staggered mesh include the addition of the artificial terms corresponding to a bulk viscosity to the equations in order to obtain reasonably smooth variation in density near shock waves and the development of spurious fluid as a result of a purely nonphysical circumstance. In the nonstaggered approach in Kashiwa *et al.* [68], all variables including velocity are located at the cell-center. In this approach, the velocity at cell faces is not computed directly, but is defined using the flow field or other dependent variables. The definition of the face-centered velocity is a crucial matter for the robustness of the method as mentioned in Kashiwa and Lee [67].

With its ability to handle complex flow problems, the ICE method for multiphase flows is utilized by UCF to simulate explosions, fires, and other fluid and fluid-structure interaction phenomena [45]. The ICE method in UCF is designed to solve "full physics" simulations of fluid-structure interactions involving large deformations and phase change [82]. As mentioned in Luitjens *et al.* [82], "full physics" refers to problems involving strong coupling between the fluid and solid phases with a full Navier Stokes representation of fluid phase materials and the transient, nonlinear response of solid phase materials, which may include chemical or phase transformation between the solid and fluid phases.

The implemented ICE method in the Uintah Computational Framework is referred to as Production ICE in Tran and Berzins [122]. The implementation of Production ICE is based on the cell-centered ICE method by Kashiwa *et al.* [68] with several exceptions that are discussed in detail in Chapter 5. The implementation of the cell-centered ICE method by Kashiwa *et al.* [68] uses a regular Cartesian grid to divide the computational domain into cells and to evaluate the changes in mass, momentum, and energy in each cell with two stages: the Lagrangian Stage and the Eulerian Stage. For the Lagrangian Stage, the advection changes in mass, momentum, and energy along a path moving with fluid velocity are evaluated by neglecting the convective terms. For the Eulerian Stage, the changes in mass, momentum, and energy in each cell due to advection are calculated. The fully cell-centered ICE method of Kashiwa *et al.* [68] defines the face-centered velocity, the rate of volume flux at cell boundary, and leaves a degree of freedom in the choice of conservation variables. The numerical scheme used in Production ICE [44, 45, 52, 79, 80] solves the conservation of mass, linear momentum and internal energy. However, the Lagrangian part of Production ICE is defined in a nonconservative form which appears to be an exception to the standard ICE method. While this may not be a problem for some cases, it appears to be a problem when applying this Production ICE code to single-fluid cases that are governed by the Euler equations in which the obtained numerical solutions exhibit some discrepancies in the shock speeds and they additionally show unphysical oscillations. Several researchers have investigated the effect of nonconservative schemes approximating hyperbolic conservation laws; for example, see [60, 75, 121]. In these investigations, they found that the numerical solutions obtained from nonconservative schemes might converge to wrong solutions.

Because of the importance of having a numerical solution that does not have spurious oscillations, we propose improvements to Production ICE in order to eliminate unphysical oscillations in its numerical solutions to compressible flow problems. We first focus on the improvements to Production ICE in one-dimensional space and then later extend these improvements to this method in multidimensional space. The cause of oscillations in numerical solutions of Production ICE is the lack of a special treatment for data discontinuities. When dealing with discontinuities in data, the idea of using a nonoscillatory piecewise linear reconstruction is typically used. This data discontinuity treatment originates from the technique used in second-order Godunov-type methods for solving numerically hyperbolic conservation laws; for some examples, see see [120] and the references within. In this approach, the value at the discontinuous point is the solution of the Generalized Riemann Problem whose piecewise constant data are obtained from a piecewise linear reconstruction. We used this treatment of data discontinuity in our improvement of Production ICE.

To enable this improved Production ICE method (IMPICE) the capability to solve significant problems, it is necessary to extend IMPICE to solve multidimensional flows in complex geometries. The use of Cartesian grids in IMPICE has the advantage of seamless grid generation for simple regular geometries, but the disadvantage of being unable to deal with complex ones. To solve complex geometries, we used in IMPICE the method of cut cells to handle the case when the computational boundary is not aligned with the cell edges; several implementations of cut cells are found in [38, 55, 124, 139]. In these implementations, there are several techniques that have previously been proposed, but the trimming of cell surfaces has mostly been used for compressible inviscid flows; see [124] and the references within. We will discuss the implementation of this technique as well as how to overcome the "small cell problem" in the IMPICE's method of cut cells; for explanations of "small cell problem," see [55] and its references. To address the "small cell problem," cell merging techniques were used by many authors; for example, see [24, 26, 36, 37, 98, 101]. We derived a new variant of the cell merging technique for merging small cells in IMPICE.

Since IMPICE is used to solve a wide range of important applications, its error analysis is necessary. In this dissertation, we determine both theoretically and numerically the accuracy of the temporal and spatial errors in the IMPICE method. The orders of accuracy in time and space can be increased by applying a high order time discretization and a nonlinear spatial discretization, respectively. We will use this error analysis of ICE to estimate the errors in this method.

As mentioned above, in addition to the use of MPM and ICE as independent simulation tools, an advanced new simulation tool is being developed that uses the combination of these two methods to simulate multimaterial and fluid-structure interactions. The combined tool, MPMICE, uses MPM to model the materials and uses ICE to model aerodynamics. As discussed in [117], the modeling techniques used by MPMICE differ from traditional methods and hold promise for increased accuracy. Basic analysis of MPM and ICE presented in this dissertation may help to understand the fluid-structure interaction modeling MPMICE in the future.

CHAPTER 3

COMPRESSIBLE FLOW PROBLEMS

Compressible flows appear in many processes in nature and technology, so the study of these flows is very important. Compressible flows model the fluids in which the fluid density varies significantly in response to a change in pressure. Such flows are obtained in gases, and they are referred to as compressible gas flows. Compressible gas flows are the subjects in the study of gas dynamics. Two of the most distinctive phenomena which occur in compressible flow problems are shock waves and choked flows [137]. Shock waves occur when there is a very sharp discontinuity in the fluid properties such as velocity, pressure, and temperature and choked flows are phenomena in which the flow rate and the velocity remain the same after the downstream change in pressure has reached a certain point. However, many of the numerical methods developed for compressible flow problems consider only the ability to accurately capture shock waves, but not choked flow; shock capturing schemes can be found in many papers such as [84, 75, 54, 61, 106, 107, 138]. The ability to capture the sharp changes in the fluid properties is essential in numerical methods for compressible flow problems in order to deal with shock waves.

The Navier-Stokes equations for compressible flow problems are time-dependent and consist of a set of nonlinear partial differential equations that describe the flow of fluids. These equations are obtained from the principles of conservation of mass, conservation of momentum, and conservation of energy with the assumption that the fluid is a continuum. In allowing shock waves to be treated as discontinuities, the system of Euler equations are usually utilized. As mentioned in Toro [120], the system of Euler equations is derived from the Navier-Stokes equations by neglecting the effects of body forces, viscous stresses and heat flux.

In this chapter, we provide a summary of the Euler equations for compressible flow problems. Here we study the system of time-dependent Euler equations in one-dimensional and multidimensional space with several important derived equations and boundary conditions. Though there are many different forms of the Euler equations along with basic physical quantities and thermodynamics relations, we include here only the forms which are useful to the discussion of the numerical methods included in this dissertation, namely the Material Point Method

(MPM) and the Improved Production Implicit Continuous-fluid Eulerian method (IMPICE). Many forms and equations in the discussion of the Euler equations in this chapter are obtained from Toro [120].

3.1 The Multidimensional Euler Equations

The system of time-dependent Euler equations of nonlinear hyperbolic conservation laws that governs the compressible flow problems in d-dimensional space can be written in the following compact form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{3.1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \nabla p = 0, \tag{3.2}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho E \mathbf{u}) + \nabla \cdot (p \mathbf{u}) = 0, \tag{3.3}$$

where $\rho(\mathbf{x}, t)$ is the density function, $\mathbf{u}(\mathbf{x}, t) = (u_1, u_2, ..., u_d)^T(\mathbf{x}, t)$ is the vector of the velocity functions, $p(\mathbf{x}, t)$ is the pressure function, and $E(\mathbf{x}, t)$ is the function of specific total energy on the problem domain $\mathbf{x} = (x_1, x_2, ..., x_d)^T \in \mathbb{R}^d$ and $t \in \mathbb{R}^+$. In Equation (3.2), $\otimes$ denotes the tensor product. The tensor product of $\mathbf{u}$ and $\mathbf{u}$ is defined as:

$$\mathbf{u} \otimes \mathbf{u} = \begin{bmatrix} {u_1}^2 & u_1 u_2 & \dots & u_1 u_d \\ u_2 u_1 & {u_2}^2 & \dots & u_2 u_d \\ \vdots & & \ddots & \\ u_d u_1 & u_d u_2 & \dots & {u_d}^2 \end{bmatrix}. \tag{3.4}$$

An equation of state is required to close the system given by Equations (3.1)–(3.3); the commonly used equation of state derived from the ideal gas law is as follows:

$$p = (\gamma - 1)\rho \left(E - \frac{1}{2} \mathbf{u} \cdot \mathbf{u} \right), \tag{3.5}$$

where γ is the specific heat ratio with the value of 1.4 for ideal gas.

The speed of sound, which is the transmission speed of a small disturbance through a medium, is a variable of interest. This variable is often used in controlling the time integration step in numerical methods for the system of Euler equations. The speed of sound, $c(\mathbf{x}, t)$, in an ideal gas is approximately given by:

$$c = \sqrt{\frac{\gamma p}{\rho}}. \tag{3.6}$$

There are two different sets of variables often used to describe the flow of compressible fluids governed by the system of Euler equations. The first set of variables called *the set of conserved variables* includes the mass density ρ, the momentum $\rho\mathbf{u}$, and the total energy per unit mass ρE. The time derivatives of the conserved variables are directly obtained from conservation laws and shown in Equations (3.1)-(3.3). Hereafter, the vector of conserved variables is denoted using the column vector $\mathbf{U}$. The second set of variables called *the set of primitive variables* or *physical variables* includes the mass density ρ, the velocity $\mathbf{u}$, and the pressure p. The time derivatives of variables in vector $\mathbf{W} = [\rho, \mathbf{u}, E, p]^T$ are shown as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{3.7}$$

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{1}{\rho} \nabla p = 0, \tag{3.8}$$

$$\frac{\partial E}{\partial t} + \mathbf{u} \cdot \nabla E + \frac{1}{\rho} \nabla \cdot (p\mathbf{u}) = 0, \tag{3.9}$$

$$\frac{\partial p}{\partial t} + \mathbf{u} \cdot \nabla p + c^2 \rho \nabla \cdot \mathbf{u} = 0. \tag{3.10}$$

These equations are useful for the discussion of the numerical methods in Chapters 5, 6, 7, and 8. It is also useful for the discussion of the IMPICE method in these chapters if we know about material derivatives. The material derivative of a quantity is a derivative taken along a moving path with the moving velocity $\mathbf{u}$. The material derivative of a scalar field $\phi(\mathbf{x}, t)$ and a vector field $\mathbf{b}(\mathbf{x}, t)$ are defined respectively as:

$$\frac{D\phi}{Dt} = \frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi, \quad \frac{D\mathbf{b}}{Dt} = \frac{\partial \mathbf{b}}{\partial t} + (\mathbf{u} \cdot \nabla)\, \mathbf{b}. \tag{3.11}$$

Using these definitions, the material derivatives of the velocity vector $\mathbf{u}$ in Equation (3.8) and pressure p in Equation (3.10) are respectively given by:

$$\frac{D\mathbf{u}}{Dt} = -\frac{1}{\rho} \nabla p, \tag{3.12}$$

$$\frac{Dp}{Dt} = -c^2 \rho \nabla \cdot \mathbf{u}. \tag{3.13}$$

These equations are also called the Lagrangian forms of Equations (3.8) and (3.10). We also use the definitions in Equation (3.11) to rewrite (3.1), (3.8), and (3.9) as follows:

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{u}, \tag{3.14}$$
$$\rho\frac{D\mathbf{u}}{Dt} = -\nabla p, \tag{3.15}$$
$$\rho\frac{DE}{Dt} = -\nabla\cdot(p\mathbf{u}). \tag{3.16}$$

In order to obtain the changes in mass, momentum, and energy along the path moving with the fluid velocity, we need to derive the equations of material derivatives for a fluid volume corresponding to the system in Equations (3.1)–(3.3). In Vallis [125], the material derivative of a finite fluid volume V for fluid density is given by:

$$\frac{D}{Dt}\int_V \rho dV = \int_V \left(\frac{D\rho}{Dt} + \rho\nabla\cdot\mathbf{u}\right) dV. \tag{3.17}$$

Also in Vallis [125], the material derivative of a finite fluid volume V for the multiplication of some fluid property ϕ and the fluid density ρ is given by:

$$\frac{D}{Dt}\int_V \rho\phi dV = \int_V \rho\frac{D\phi}{Dt} dV. \tag{3.18}$$

As mentioned in Vallis [125], the above formula also holds if ϕ is a vector. Applying Equation (3.18) to fluid velocity $\mathbf{u}$ and fluid specific total energy E, we have the following equations:

$$\frac{D}{Dt}\int_V \rho\mathbf{u} dV = \int_V \rho\frac{D\mathbf{u}}{Dt} dV, \tag{3.19}$$
$$\frac{D}{Dt}\int_V \rho E dV = \int_V \rho\frac{DE}{Dt} dV. \tag{3.20}$$

In these equations, the volume V changes due to the movement of the bounding surface. Let S be the bounding surface of the volume V, then the change in volume is described by the following equation:

$$\frac{D}{Dt}\int_V dV = \int_S \mathbf{u}\cdot d\mathbf{S}. \tag{3.21}$$

From Equations (3.14)–(3.20), the following equations are obtained:

$$\frac{D}{Dt}\int_V \rho dV = 0, \tag{3.22}$$
$$\frac{D}{Dt}\int_V \rho\mathbf{u} dV = -\int_V \nabla p dV, \tag{3.23}$$
$$\frac{D}{Dt}\int_V \rho E dV = -\int_V \nabla\cdot(p\mathbf{u}) dV. \tag{3.24}$$

Equations (3.22)–(3.24) are used to evaluate the changes in mass, momentum, and energy along a path moving with fluid velocity $\mathbf{u}$ neglect the convective terms. In the other hand, the changes in mass, momentum, and energy due to the convective terms are governed by:

$$\frac{d}{dt}\int_V \rho dV = -\int_V \nabla \cdot (\rho \mathbf{u}) dV, \tag{3.25}$$

$$\frac{d}{dt}\int_V \rho \mathbf{u} dV = -\int_V \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) dV, \tag{3.26}$$

$$\frac{d}{dt}\int_V \rho E dV = -\int_V \nabla \cdot (\rho E \mathbf{u}) dV. \tag{3.27}$$

The concepts of material derivatives introduced above are used to explain the implementation of numerical methods for the system of Euler equations with a separate Lagrangian Phase and an Eulerian Phase; for example, the implementation of the cell-centered ICE method of Kashiwa *et al.* [68], the Production ICE method, and the improved Production ICE method which will be discussed in detail in Chapter 6 and Chapter 7.

3.2 The One-Dimensional Euler Equations

In the case of one-dimensional space ($d = 1$), the system of Euler equations in conservation form given by Equations (3.1)–(3.3) is now simplified as follows:

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0, \tag{3.28}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \left(\rho u^2 + p\right)}{\partial x} = 0, \tag{3.29}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \left(\rho u E + p u\right)}{\partial x} = 0, \tag{3.30}$$

where $u(x,t)$ is the velocity in the one-dimensional space and $x \in \mathbb{R}$. The equation of state (3.5) becomes:

$$p = (\gamma - 1)\rho \left(E - \frac{1}{2}u^2\right). \tag{3.31}$$

One nonconservative form of the one-dimensional system of Euler equation is given by

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0, \tag{3.32}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \left(\rho u^2\right)}{\partial x} + \frac{\partial p}{\partial x} = 0, \tag{3.33}$$

$$\frac{\partial \rho e}{\partial t} + \frac{\partial \left(\rho u e\right)}{\partial x} + p\frac{\partial u}{\partial x} = 0, \tag{3.34}$$

where $e(x,t)$ is the specific internal energy and its relationship with the total energy per unit mass, $E(x,t)$, is given by:

$$E = e + \frac{1}{2}u^2. \tag{3.35}$$

In an ideal gas, the internal energy is a function of temperature. The equation of state in (3.31) is rewritten in terms of the density and the internal energy as follows:

$$p = (\gamma - 1)\rho e. \tag{3.36}$$

The material derivatives of velocity in Equation (3.12) and pressure in Equation (3.13) for the one-dimensional space are written as follows:

$$\frac{Du}{Dt} = -\frac{1}{\rho}\frac{\partial p}{\partial x}, \tag{3.37}$$
$$\frac{Dp}{Dt} = -c^2\rho\frac{\partial u}{\partial x}. \tag{3.38}$$

The one-dimensional forms of material derivatives in Equations (3.22)–(3.24) are now given by:

$$\frac{D}{Dt}\int_V \rho dV = 0, \tag{3.39}$$
$$\frac{D}{Dt}\int_V \rho u dV = -\int_V \frac{\partial p}{\partial x}dV, \tag{3.40}$$
$$\frac{D}{Dt}\int_V \rho E dV = -\int_V \frac{\partial (pu)}{\partial x}dV, \tag{3.41}$$

where the volume change is governed by the following equation:

$$\frac{D}{Dt}\int_V dV = \int_S u\ dS. \tag{3.42}$$

The changes in mass, momentum, and energy due to the convective terms described by Equations (3.25)–(3.27) are rewritten for the one-dimensional space as follows:

$$\frac{d}{dt}\int_V \rho dV = -\int_V \frac{\partial (\rho u)}{\partial x}dV, \tag{3.43}$$
$$\frac{d}{dt}\int_V \rho u dV = -\int_V \frac{\partial (\rho u u)}{\partial x}dV, \tag{3.44}$$
$$\frac{d}{dt}\int_V \rho E dV = -\int_V \frac{\partial (\rho E u)}{\partial x}dV. \tag{3.45}$$

Another form of Equations (3.32)–(3.34) is used in the discussion of the Material Point Method in Chapter 5 as given by Sulsky *et al.* [113] as follows:

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} + \rho\frac{\partial u}{\partial x} = 0, \tag{3.46}$$

$$\frac{\partial e}{\partial t} + u\frac{\partial e}{\partial x} + \frac{p}{\rho}\frac{\partial u}{\partial x} = 0. \tag{3.47}$$

The equations of material derivatives corresponding to Equations (3.46) and (3.47) are given by:

$$\frac{D\rho}{Dt} = -\rho\frac{\partial u}{\partial x}, \tag{3.48}$$

$$\frac{De}{Dt} = -\frac{p}{\rho}\frac{\partial u}{\partial x}. \tag{3.49}$$

3.3 Boundary Conditions

We have, so far, presented the fluid flows in $\mathbb{R}^d$. If the fluid flows are bounded in a spatial region $\Omega \subset \mathbb{R}^d$, then it is necessary to impose conditions on the boundary $\partial\Omega$. Different boundary conditions associated with partial differential equations include the Neumann boundary condition, the Dirichlet boundary condition, and the mixed boundary condition which is a combination of the Dirichlet and Neumann boundary conditions [134]. In a Dirichlet boundary condition, the value of a variable at the boundary is prescribed; in a Neumann boundary condition, the derivative of a variable at the boundary is prescribed; in a mixed boundary condition, a linear combination of the Dirichlet and Neumann boundary conditions at the boundary is prescribed. At a given boundary, different types of boundary conditions can be used for different variables.

The commonly used boundary conditions for the system of Euler equations are discussed in Sod [108]. Great care is necessary in the implementation of the numerical boundary conditions for numerical simulations of the compressible flow problems. At a computational boundary, a boundary condition is used to direct the flow between the boundary inlet and outlet. This flow is specified by a wide range of boundary condition types.

In the following sections, we will describe several examples of numerical boundary conditions and the Euler characteristic boundary conditions.

3.3.1 Examples of Numerical Boundary Conditions

The following is the list of the most often implemented numerical boundary conditions for the Euler equations. These numerical boundary conditions are discussed in [99, 120].

Periodic Boundary. As mentioned in Poinsot and Veynante [99], the computation domain is folded on itself for the case of periodic boundary. In the treatment of periodic boundary conditions, the value of a variable at the inlet boundary is set to the value of that variable at the outlet boundary.

Transmissive Boundary. As mentioned in Toro [120], transmissive boundaries arise from the need to define finite (or sufficiently small) computational domains. At a transmissive boundary, the waves are allowed to pass without any change being made to the waves. In the treatment of the transmissive boundary condition, the value of a variable at the boundary is defined using the value of this variable inside the computational domain.

Reflective Boundary. At a reflective boundary, the waves reflect and move in the inverse direction. In the treatment of the reflective boundary condition, the value of all variables at the boundary except for velocity is defined using the value of these variables inside the computational domain; the normal component of the velocity at boundary is the negation of the normal component of the velocity inside the computational domain.

Solid Boundary. At a solid boundary, the fluid flow will have zero velocity relative to the boundary. This result is derived from the no-slip condition for viscous fluids in fluid dynamics. Generally this boundary condition implies that the fluid in contact with a solid wall will have the same velocity as the velocity of the solid wall.

Nonreflecting Inflow Boundary. At the inflow boundary, the fluid enters the computational domain. In the treatment of the nonreflecting inflow boundary, the inlet values of the flow velocity vector and temperature are imposed. The pressure gradient will again be given by momentum considerations under the assumption that the flow is fully developed at the entrance.

Outflow Boundary. At the outflow boundary, the fluid leaves the computational domain. In order to maintain the smoothness of the flow through the boundary, the normal derivative of velocity at the boundary is set to zero.

3.3.2 Euler Characteristic Boundary Condition (ECBC)

There are two classes of boundary conditions used to specify dependent variables at the boundaries: physical boundary conditions – using known physical behaviors, and numerical boundary conditions – using numerical descriptions. Physical boundary conditions are independent of the method used to solve the relevant equations; on the contrary, numerical boundary conditions are dependent of the method used. For the case of system of Euler equations, when the number of physical boundary conditions is lower than the number of primitive variables, then the variables which are not specified using physical attributes must be obtained using numerical boundary conditions. The numerical boundary conditions of these variables may be obtained from extrapolation or the set of characteristic relations. However, it seems reasonable to obtain the numerical boundary conditions using characteristic relations and so to avoid extrapolations. The method of Euler Characteristic Boundary Conditions (ECBC) uses characteristic relations based on the analysis of the different waves crossing the boundary to specify boundary conditions for the system of Euler equations. In ECBC methods, some of the variables on the boundaries may be obtained from extrapolations while some others may be obtained using characteristic relations.

The following derivation of the ECBCs in x_i-direction of the multidimensional space is shown in [99]. Using a wave analysis of the Euler equations, the decomposition of the normal terms in x_i-direction into vector $\mathbf{h} = [h_1, h_2, ..., h_{d+2}]^T$ as given by:

$$h_1 = \frac{\partial \rho u_i}{\partial x_i}, \tag{3.50}$$

$$h_2 = \rho c^2 \frac{\partial u_i}{\partial x_i} + u_i \frac{\partial p}{\partial x_i}, \tag{3.51}$$

$$h_{2+k} = \begin{cases} u_i \dfrac{\partial u_i}{\partial x_i} + \dfrac{1}{\rho}\dfrac{\partial p}{\partial x_i} & \text{if} \quad (k = i) \\ u_i \dfrac{\partial u_k}{\partial x_i} & \text{otherwise,} \end{cases} \tag{3.52}$$

where $k = 1, ..., d$ and d is the dimension of the multidimensional space. An explanation of how these terms are decomposed as shown can be found in [118]. The system of Euler equations in (3.1)–(3.3) is now rewritten as follows:

$$\frac{\partial \rho}{\partial t} + h_1 + \sum_{\substack{j=1 \\ j \neq i}}^{d} \frac{\partial (\rho u_j)}{\partial x_j} = 0, \tag{3.53}$$

$$\frac{\partial (\rho E)}{\partial t} + \frac{1}{2}\sum_{j=1}^{d} u_k^2 h_1 + \frac{h_2}{\gamma - 1} + \rho \sum_{j=1}^{d} u_j h_{2+j} + \sum_{\substack{j=1 \\ j \neq i}}^{d} \frac{\partial (\rho E u_j)}{\partial x_j} + \sum_{\substack{j=1 \\ j \neq i}}^{d} \frac{\partial (p u_j)}{\partial x_j} = 0, \tag{3.54}$$

$$\frac{\partial (\rho u_k)}{\partial t} + u_k h_1 + \rho h_{2+k} + \sum_{\substack{j=1 \\ j \neq i}}^{d} \frac{\partial (\rho u_k u_j)}{\partial x_j} = \begin{cases} 0 & \text{if } (k = i) \\ -\dfrac{1}{\rho}\dfrac{\partial p}{\partial x_k} & \text{otherwise,} \end{cases} \tag{3.55}$$

for $k = 1, ..., d$. Define the vector $\mathbf{L} = [L_1, L_2, ..., L_{d+2}]^T$ of the amplitudes of characteristic waves associated with the characteristic velocities in x_i-direction. These characteristic wave amplitudes are specified as follows:

$$L_1 = (u_i - c)\left(\frac{\partial p}{\partial x_i} - \rho c \frac{\partial u_i}{\partial x_i}\right), \tag{3.56}$$

$$L_2 = u_i\left(c^2\frac{\partial \rho}{\partial x_i} - \frac{\partial p}{\partial x_i}\right), \tag{3.57}$$

$$L_{2+k} = \begin{cases} u_i\dfrac{\partial u_k}{\partial x_i} & \text{if } (k \neq i) \\ (u_i + c)\left(\dfrac{\partial p}{\partial x_i} + \rho c \dfrac{\partial u_i}{\partial x_i}\right) & \text{otherwise,} \end{cases} \tag{3.58}$$

for $k = 1, ..., d$. The ECBC defines $\{L_j : j = 1, ..., d+2\}$, which are the amplitudes of the waves crossing the boundary, by imposing the physical conditions.

CHAPTER 4

ADJOINT-BASED ERROR ESTIMATION FOR NUMERICAL SOLUTIONS OF PARTIAL DIFFERENTIAL EQUATIONS

The importance of obtaining a reliable error estimate for numerical solutions to time-dependent ordinary differential equations (ODEs) and partial differential equations (PDEs) is well understood, see [21, 34, 58, 81, 91, 92, 93]. As mentioned in Cao and Petzold [21], many methods of global error estimation have been proposed, studied carefully, and implemented in several ODE solvers. These error estimators either use residual errors for the error indicators or error recovery techniques. Residual errors are the errors resulting from failing to satisfy exactly the differential equations of numerical solutions. The estimate of residual error is also sometimes used to gain confidence in a numerical solution. The global error estimates that use error recovery techniques often solve the problem a second time with a reduced step size or tolerance and assume the second integration is more accurate; the error in the first integration is then recovered by the difference between the two numerical solutions. These estimates may sometimes be inaccurate since the second integration may not yield a more accurate solution [21, 104]. As a consequence, there have been many error estimates that use residual errors and multipliers obtained from the solution of the adjoint problem [21].

There are two different approaches for ODE global error estimates: the classical approach (based on the forward integration of an error equation) and the adjoint-based approach (based on residual errors and the backward integration of the adjoint problem). These approaches are compared by Lang and Verwer [73] for their reliability and efficiency. One disadvantage of adjoint-based methods is the need to store the forward solution that is required during the backward time integration. Lang and Verwer [73] suggested that the adjoint-based approach may not be competitive against the classical approach due to its huge storage demand for large problems, even though both approaches work well in terms of reliability. On the other hand, Cao and Petzold [21] suggested that the adjoint-based approach was an attractive choice and proposed a novel approach to reduce the number of backward time integrations using the small sample statistical method. Furthermore, adjoint systems are linear, so they can be solved in parallel. Even though solving the adjoint system requires extra work and storage, the adjoint solutions are useful for adaptively control the global error as they are the appropriate weighting cofficients of local errors contributed to the global error.

Adjoint-based error estimates have also been used and become increasingly important in the error analysis of applications in Computational Fluid Dynamics (CFD). The use of adjoint methods in CFD error analysis has been discussed in many papers; for example, see [8, 41, 7, 133, 2, 64, 65, 66]. The main factor that contributes to the growing interest in adjoint methods is their application towards sensitivity analysis for large-scale systems governed by PDEs. Sensitivity analysis is applied in a wide range of applications in science and engineering that involve optimal design problems and error control problems. In these applications, the impact of input parameters on the errors in functional outputs is determined using sensitivity analysis. As mentioned in Venditti and Darmofal [133] and the references within, invoking the adjoint problem has the primary advantage of directly relating the error in a chosen functional output to the local residual errors. We show in this chapter that if the initial condition in the adjoint problem is properly set then the global error of the numerical solutions to systems of PDEs can be formulated using the local errors (the sum of ODE local error and the PDE truncation error) and the solutions of the adjoint problems. The proper initial condition for the adjoint problem was discussed in Cao and Petzold [21].

The adjoint problem when solving a system of PDEs can be formulated using either the continuous approach or the discrete approach. In the continuous approach, the adjoint problem is obtained from discretizing the analytic adjoint PDE. In the discrete approach, the adjoint problem is obtained from the system of ODEs approximating the PDEs. As mentioned in Li and Petzold [76], the system resulting from the continuous approach is much simpler than the system obtaining from the discrete approach. However, the discrete approach has the advantage of not requiring the explicit derivation and the discretization of the adjoint equations and corresponding boundary conditions as mentioned in Venditti and Darmofal [133].

In this chapter, we discuss our discrete adjoint-based approach for estimating spatial and temporal errors for the method-of-lines PDEs. We also test this approach on numerical solutions to several ODE and PDE problems using the Backward Differentiation Formula (BDF) method implemented in the DASSL DAL solver described in Ascher and Petzold [3]. In order to use our adjoint-based approach for numerical solutions obtained via the BDF method, we derive a technique for estimating the local error by sampling the integration residual error at two points per time interval.

4.1 Errors in Numerical Solutions to Partial Differential Equations

Consider the following class of time-dependent PDEs:

$$\frac{\partial \mathbf{Y}}{\partial t} = \mathbf{G}(t, \mathbf{Y}, \nabla \mathbf{Y}), \tag{4.1}$$

where $\mathbf{x} \in \Omega \subset \mathbb{R}^d$ and $t \in (0, T_e]$. Boundary conditions are imposed on $\partial\Omega$, and the initial condition has the form:

$$\mathbf{Y}(\mathbf{x}, 0) = \mathbf{Y}_0(\mathbf{x}), \quad \forall \mathbf{x} \in \Omega. \tag{4.2}$$

The system of Euler equations discussed in Chapter 3 is an instance of the above class of PDEs.

Let Ω_H be some space discretization of Ω. In Ω_H, the solution to the PDE system in (4.1) is numerically computed at the discrete points $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$. Let

$$\mathbf{Y}_H(t) = \left[\mathbf{Y}_H(\mathbf{x}_0, t), \mathbf{Y}_H(\mathbf{x}_1, t), \mathbf{Y}_H(\mathbf{x}_2, t), ..., \mathbf{Y}_H(\mathbf{x}_N, t)\right]^T, \tag{4.3}$$

where $\mathbf{Y}_H(t)$ is the solution to the following ODE system:

$$\begin{cases} \dot{\mathbf{Y}}_H(t) & = \mathbf{G}_H\left(t, \mathbf{Y}_H(t)\right) \\ \mathbf{Y}_H(0) & = \mathbf{Y}_{0H}, \end{cases} \tag{4.4}$$

and vector $\mathbf{G}_H(t, \mathbf{Y}_H(t))$ approximates the column vector of values of $\mathbf{G}$ at discrete points $\mathbf{G}\left(t, \mathbf{Y}(\mathbf{x}_0), \nabla\mathbf{Y}(\mathbf{x}_0)\right)$, $\mathbf{G}\left(t, \mathbf{Y}(\mathbf{x}_1), \nabla\mathbf{Y}(\mathbf{x}_1)\right)$, ..., $\mathbf{G}\left(t, \mathbf{Y}(\mathbf{x}_N), \nabla\mathbf{Y}(\mathbf{x}_N)\right)$. The initial condition in Equation (4.4) is given by:

$$\mathbf{Y}_{0H} = \left[\mathbf{Y}_0(\mathbf{x}_0), \mathbf{Y}_0(\mathbf{x}_1), \mathbf{Y}_0(\mathbf{x}_2), ..., \mathbf{Y}_0(\mathbf{x}_N)\right]. \tag{4.5}$$

Let $\tilde{\mathbf{Y}}_H(t)$ be a perturbed solution of $\mathbf{Y}_H(t)$. The temporal error (also known as the time integration error), $\mathbf{et}_H(t)$, is defined as:

$$\mathbf{et}_H(t) = \mathbf{Y}_H(t) - \tilde{\mathbf{Y}}_H(t). \tag{4.6}$$

Let $\mathbf{Y}(\mathbf{x}, t)$ be the exact solution of the system in (4.1). The restriction of this exact solution to the discretized mesh is denoted as:

$$\mathbf{Y}[H](t) = \left[\mathbf{Y}(\mathbf{x}_0, t), \mathbf{Y}(\mathbf{x}_1, t), \mathbf{Y}(\mathbf{x}_2, t), ..., \mathbf{Y}(\mathbf{x}_N, t)\right]. \tag{4.7}$$

The error introduced by the space discretization, also known as the spatial error, is denoted as $\mathbf{es}_H(t) = \mathbf{Y}[H](t) - \mathbf{Y}_H(t)$. The overall error in numerical solutions of PDEs, $\mathbf{ge}_H(t)$, is then written as:

$$\begin{aligned} \mathbf{ge}_H(t) &= \mathbf{Y}[H](t) - \tilde{\mathbf{Y}}_H(t) = (\mathbf{Y}[H](t) - \mathbf{Y}_H(t)) + \Big(\mathbf{Y}_H(t) - \tilde{\mathbf{Y}}_H(t)\Big) \\ &= \mathbf{es}_H(t) + \mathbf{et}_H(t), \end{aligned} \tag{4.8}$$

thus showing that there are two parts of errors in the numerical solutions to PDEs: the spatial error and the temporal error. The spatial error comes from the spatial discretization of the PDEs and the temporal error comes from the time integration of the discretized ODEs.

4.2 Adjoint-based Error Estimation for ODEs

The adjoint-based global error estimate of Cao and Petzold given in [21] is described here in a slightly modified form. We consider the class of ODEs given by:

$$\begin{cases} \dot{\mathbf{Y}}(t) &= \mathbf{G}(\mathbf{Y}, t) \qquad 0 \le t \le T_e \\ \mathbf{Y}(0) &= \mathbf{Y}_0, \end{cases} \tag{4.9}$$

where $\mathbf{Y} \in \mathbb{R}^d$. The numerical solution $\tilde{\mathbf{Y}} \in \mathbb{R}^d$ satisfies the following perturbed system:

$$\begin{cases} \dot{\mathbf{Y}}(t) &= \mathbf{G}(\mathbf{Y}, t) + \mathbf{r}(t) \qquad 0 \le t \le T_e \\ \mathbf{Y}(0) &= \mathbf{Y}_0 + \mathbf{r}_0, \end{cases} \tag{4.10}$$

where $\mathbf{r}(t)$ denotes the pertubation of the numerical solution at time t and the initial pertubation $\mathbf{r}(0) = \mathbf{r}_0$. The pertubation function $\mathbf{r}(t)$ is also referred to as the residual error.

Let $\mathbf{et}(t)$ be defined by $\mathbf{et}(t) = \tilde{\mathbf{Y}}(t) - \mathbf{Y}(t)$, which is the error in the numerical solution $\tilde{\mathbf{Y}}$ of $\mathbf{Y}$ at time t. Then $\mathbf{et}(t)$ approximately satisfies the following ODE system:

$$\begin{cases} \dot{\mathbf{et}}(t) &= \mathbf{J}(\tilde{\mathbf{Y}}, t)\mathbf{et}(t) + \mathbf{r}_1(\mathbf{Y}, \tilde{\mathbf{Y}}, t) + \mathbf{r}(t) \\ \mathbf{et}(0) &= \mathbf{r}_0, \end{cases} \tag{4.11}$$

where $\mathbf{J}(\tilde{\mathbf{Y}}, t)$ is the Jacobian of $\mathbf{G}$ at $\tilde{\mathbf{Y}}$. The residual $\mathbf{r}_1(\mathbf{Y}, \tilde{\mathbf{Y}}, t)$ is an approximation to the quadratic and subsequent Taylor series terms given by $\mathbf{r}_1(\mathbf{Y}, \tilde{\mathbf{Y}}, t) = \mathbf{G}(\tilde{\mathbf{Y}}, t) - \mathbf{G}(\mathbf{Y}, t) - \mathbf{J}(\tilde{\mathbf{Y}}, t)(\tilde{\mathbf{Y}} - \mathbf{Y})$ with $\|\mathbf{r}_1(\mathbf{Y}, \tilde{\mathbf{Y}}, t)\|_\infty$ is assumed to be small when $\tilde{\mathbf{Y}}(t)$ is close to $\mathbf{Y}(t)$. The adjoint-based global error estimate of Cao and Petzold [21] was derived with the assumption that the term $\|\mathbf{r}_1(\mathbf{Y}, \tilde{\mathbf{Y}}, t)\|_\infty$ was small enough to be neglected. Therefore, the approach of global error estimate described in this section cannot be trusted if the system of ODEs in (4.9) is not solved to a sufficient accuracy. With the assumption that $\|\mathbf{r}_1(\mathbf{Y},\tilde{\mathbf{Y}},t)\|_\infty$ may be neglected, we have:

$$\begin{cases} \dot{\mathbf{et}}(t) & \approx \mathbf{J}(\tilde{\mathbf{Y}},t)\mathbf{et}(t)+\mathbf{r}(t) \\ \mathbf{et}(0) & = \mathbf{r}_0. \end{cases} \tag{4.12}$$

Let $\boldsymbol{\lambda}(t)$ be some vector in $\mathbb{R}^d$ that solves the following system:

$$\begin{cases} \dot{\boldsymbol{\lambda}}(t) & = -\mathbf{J}^T(\tilde{\mathbf{Y}},t)\boldsymbol{\lambda}(t) \qquad 0 \le t \le T_e \\ \boldsymbol{\lambda}(T_e) & = \mathbf{l}, \end{cases} \tag{4.13}$$

for some vector $\mathbf{l}$ in $\mathbb{R}^d$. Multiplying both sides of first equation in (4.12) by $\boldsymbol{\lambda}^T(t)$ gives:

$$\begin{aligned} \boldsymbol{\lambda}^T(t)\dot{\mathbf{et}}(t) &\approx \boldsymbol{\lambda}^T(t)\mathbf{J}(\tilde{\mathbf{Y}},t)\ \mathbf{et}(t)+\boldsymbol{\lambda}^T(t)\mathbf{r}(t) \\ &= (\mathbf{J}^T(\tilde{\mathbf{Y}},t)\boldsymbol{\lambda}(t))^T\ \mathbf{et}(t)+\boldsymbol{\lambda}^T(t)\mathbf{r}(t) \\ &= \left(-\dot{\boldsymbol{\lambda}}(t)\right)^T\ \mathbf{et}(t)+\boldsymbol{\lambda}^T(t)\mathbf{r}(t). \end{aligned} \tag{4.14}$$

Rearranging this yields:

$$\boldsymbol{\lambda}^T(t)\dot{\mathbf{et}}(t)+\left(\dot{\boldsymbol{\lambda}}(t)\right)^T\ \mathbf{et}(t) \approx \boldsymbol{\lambda}^T(t)\mathbf{r}(t), \tag{4.15}$$

and, in turn, gives:

$$\frac{d}{dt}(\boldsymbol{\lambda}^T(t)\ \mathbf{et}(t)) \approx \boldsymbol{\lambda}^T(t)\mathbf{r}(t). \tag{4.16}$$

Integrating both sides of the above equation gives:

$$\begin{aligned} \int_0^{T_e}\frac{d}{dt}(\boldsymbol{\lambda}^T(t)\ \mathbf{et}(t))dt &\approx \int_0^{T_e}\boldsymbol{\lambda}^T(t)\mathbf{r}(t)dt, \\ \boldsymbol{\lambda}^T(T_e)\ \mathbf{et}(T_e)-\boldsymbol{\lambda}^T(0)\ \mathbf{et}(0) &\approx \int_0^{T_e}\boldsymbol{\lambda}^T(t)\mathbf{r}(t)dt, \\ \mathbf{l}^T\ \mathbf{et}(T_e)-\boldsymbol{\lambda}^T(0)\mathbf{r}_0 &\approx \int_0^{T_e}\boldsymbol{\lambda}^T(t)\mathbf{r}(t)dt. \end{aligned}$$

Or:

$$\mathbf{l}^T\ \mathbf{et}(T_e) \approx \int_0^{T_e}\boldsymbol{\lambda}^T(t)\mathbf{r}(t)dt+\boldsymbol{\lambda}^T(0)\mathbf{r}_0. \tag{4.17}$$

It is perhaps worth remarking that if we replace $\mathbf{r}(t)$ by $\mathbf{r}(t) + \mathbf{r}_1(\mathbf{Y}, \tilde{\mathbf{Y}}, t)$, then this equation is exact. To estimate the ith-component of error vector $\mathbf{et}(T_e)$, we solve system in (4.13) with initial condition $\mathbf{l} = \mathbf{e}_i = [0, 0, ..., 0, 1, 0, ...0]^T$ with a value of 1 at the i^{th}-component and 0 elsewhere. So in order to estimate the global error vector $\mathbf{et}(T_e)$, we have to solve the system in (4.13) d times (d: number of ODE equations) with d different values of vector $\mathbf{l}$: $\mathbf{e}_1, \mathbf{e}_2, ...$, or $\mathbf{e}_d$, where $\mathbf{e}_1, \mathbf{e}_2, ...$, and $\mathbf{e}_d$ are the standard basis for $\mathbb{R}^d$. Since the value of $\boldsymbol{\lambda}(t)$ can only be obtained numerically, the adjoint-based global error estimate cannot be trusted either if the adjoint system in (4.13) is not solved to a sufficient accuracy.

The global error estimate using Equation (4.17) requires the estimate of the residual error $\mathbf{r}(t)$ in addition to the solution of the adjoint system in (4.13). The residual error defined by Equation (4.10) is as follows:

$$\mathbf{r}(t) = \dot{\tilde{\mathbf{Y}}}(t) - \mathbf{G}(\tilde{\mathbf{Y}}, t), \tag{4.18}$$

where $\tilde{\mathbf{Y}}(t)$ is some approximation of function $\mathbf{Y}(t)$ obtained from interpolating the temporal discrete numerical solutions.

Assume that some time integration procedure is used and the discrete temporal numerical solutions, $\{\tilde{\mathbf{Y}}^n \,:\, n = 1, ..., m\}$, are obtained at $t_1 = 0$, t_2, t_3, ..., $t_m = T_e$. We then approximate the left side of Equation (4.17) as follows:

$$\mathbf{l}^T \mathbf{et}(T_e) \approx \sum_{j=1}^{m} \int_{t_j}^{t_{j+1}} \boldsymbol{\lambda}^T(t)\mathbf{r}(t)dt + \boldsymbol{\lambda}^T(0)\mathbf{r}_0 \tag{4.19}$$

At $t = t_n$, define the following local problem:

$$\begin{cases} \dot{\mathbf{Z}}_{n+1}(t) = \mathbf{G}(\mathbf{Z}_{n+1}(t), t), & t \in [t_n, t_{n+1}], \\ \mathbf{Z}_{n+1}(t_n) = \tilde{\mathbf{Y}}^n. \end{cases} \tag{4.20}$$

The error for this local problem is given by:

$$\mathbf{le}\left(t; t_n, \tilde{\mathbf{Y}}^n\right) = \mathbf{Z}_{n+1}(t) - \tilde{\mathbf{Y}}(t). \tag{4.21}$$

The local error per time step, $\mathbf{le}\left(t_{n+1}; t_n, \tilde{\mathbf{Y}}^n\right)$, is then defined by:

$$\mathbf{le}\left(t_{n+1}; t_n, \tilde{\mathbf{Y}}^n\right) = \mathbf{Z}_{n+1}(t_{n+1}) - \tilde{\mathbf{Y}}^{n+1}. \tag{4.22}$$

The residual error, $\mathbf{r}(t)$, in Equation (4.18) is then estimated by:

$$\begin{aligned}\mathbf{r}(t) &= \dot{\tilde{\mathbf{Y}}}(t) - \mathbf{G}(\tilde{\mathbf{Y}}(t), t) = \dot{\tilde{\mathbf{Y}}}(t) - \dot{\mathbf{Z}}_{n+1}(t) + \mathbf{G}(\mathbf{Z}_{n+1}(t), t) - \mathbf{G}(\tilde{\mathbf{Y}}(t), t) \\ &\approx \dot{\tilde{\mathbf{Y}}}(t) - \dot{\mathbf{Z}}_{n+1}(t) - \mathbf{J}(\tilde{\mathbf{Y}}(t), t)(\tilde{\mathbf{Y}}(t) - \mathbf{Z}_{n+1}(t))).\end{aligned} \tag{4.23}$$

Using a similar derivation as given by Equations (4.15)–(4.16), we have:

$$\boldsymbol{\lambda}^T(t)\mathbf{r}(t) \approx \frac{d}{dt}\left(\boldsymbol{\lambda}^T(t)(\tilde{\mathbf{Y}}(t) - \mathbf{Z}_{n+1}(t)\right). \tag{4.24}$$

Therefore:

$$\begin{aligned}\int_{t_n}^{t_{n+1}} \boldsymbol{\lambda}^T(t)\mathbf{r}(t)dt &\approx \int_{t_n}^{t_{n+1}} \frac{d}{dt}\left(\boldsymbol{\lambda}^T(t)(\tilde{\mathbf{Y}}(t) - \mathbf{Z}_{n+1}(t)\right) dt \\ &= \boldsymbol{\lambda}^T(t_{n+1})\left(\tilde{\mathbf{Y}}(t_{n+1}) - \mathbf{Z}_{n+1}(t_{n+1})\right) - \boldsymbol{\lambda}^T(t_n)\left(\tilde{\mathbf{Y}}^n - \mathbf{Z}_{n+1}(t_n)\right) \\ &= -\boldsymbol{\lambda}^T(t_{n+1})\mathbf{le}\left(t_{n+1}; t_n, \tilde{\mathbf{Y}}^n\right).\end{aligned}$$

Equation (4.19) is then given by:

$$\mathbf{l}^T\mathbf{et}(T_e) \approx \sum_{j=1}^{m} -\boldsymbol{\lambda}^T(t_{j+1})\mathbf{le}\left(t_{j+1}; t_j, \tilde{\mathbf{Y}}^j\right) + \boldsymbol{\lambda}^T(0)\mathbf{r}_0. \tag{4.25}$$

The global error (also time integration error $\mathbf{et}(T_e)$ of numerical solutions to ODEs) in the adjoint-based approach is then calculated by the summation of products of the discrete adjoint solution and the local error.

4.3 Adjoint-based Error Estimation for Numerical Solutions of PDEs

From Equations (4.6) and (4.12), the temporal error $\mathbf{et}_H(t)$ in the numerical solution to PDEs (4.1) approximately satisfies the following system:

$$\begin{cases}\dot{\mathbf{et}}_H(t) &= \mathbf{J}_H(t, \tilde{\mathbf{Y}}_H)\mathbf{et}_H + \mathbf{r}_H(t) \\ \mathbf{et}_H(0) &= \mathbf{r}_{0_H},\end{cases} \tag{4.26}$$

where $\mathbf{J}_H(t, \tilde{\mathbf{Y}}_H)$ is Jacobian of $\mathbf{G}_H(t, \mathbf{Y}_H(t))$ with respect to $\mathbf{Y}_H(t)$ and $\mathbf{r}_H(t) = \dot{\tilde{\mathbf{Y}}}(t) - \mathbf{G}_H(t, \tilde{\mathbf{Y}}_H(t))$.

According to Berzins [13], the approximate equation for spatial error $\mathbf{es}_H(t)$ is given by:

$$\begin{cases} \dot{\mathbf{es}}_H(t) & = \mathbf{J}_H(t, \tilde{\mathbf{Y}}_H)\mathbf{es}_H(t) + \mathbf{TE}_H(t) \\ \mathbf{es}_H(0) & = 0, \end{cases} \tag{4.27}$$

where $\mathbf{TE}_H(t) = \dot{\mathbf{Y}}[H](t) - \mathbf{G}_H(t, \mathbf{Y}[H](t))$.

From Equations (4.8), (4.26) and (4.27), we have:

$$\begin{cases} \dot{\mathbf{ge}}_H(t) & = \mathbf{J}_H(t, \tilde{\mathbf{Y}}_H)\mathbf{ge}_H(t) + \mathbf{r}_H(t) + \mathbf{TE}_H(t) \\ \mathbf{ge}_H(0) & = \mathbf{r}_{0_H}. \end{cases} \tag{4.28}$$

Consider the following adjoint system:

$$\begin{cases} \dot{\boldsymbol{\lambda}}(t) & = -\mathbf{J}_H^T(t, \tilde{\mathbf{Y}}_H)\boldsymbol{\lambda}(t), \quad 0 \le t \le T_e \\ \boldsymbol{\lambda}(T_e) & = \mathbf{l}, \end{cases} \tag{4.29}$$

for some vector $\mathbf{l}$ in $\mathbb{R}^d$ where d is the number of ODEs in system in (4.4) . Given the similar form of Equations (4.12) and (4.27), the adjoint-based spatial error estimate is then given as follows:

$$\mathbf{l}^T\mathbf{es}_H(T_e) = \int_0^{T_e} \boldsymbol{\lambda}^T(s)\mathbf{TE}_H(s)ds + \boldsymbol{\lambda}^T(0)\mathbf{r}_{0_H}. \tag{4.30}$$

With the assumption that $\boldsymbol{\lambda}(t_j + \tau) \approx \boldsymbol{\lambda}(t_j)$ for $0 \le \tau \le (t_{j+1} - t_j)$, Equation (4.30) now becomes:

$$\mathbf{l}^T\mathbf{es}_H(T_e) = \sum_{j=1}^{m} \boldsymbol{\lambda}^T(t_j) \int_{t_j}^{t_{j+1}} \mathbf{TE}_H(t)dt + \boldsymbol{\lambda}^T(0)\mathbf{r}_{0_H}. \tag{4.31}$$

The combination of spatial and temporal error for numerical solutions of time-dependent PDEs is then approximated using:

$$\begin{aligned} \mathbf{l}^T\mathbf{ge}_H(T_e) &= \sum_{j=1}^{m} \boldsymbol{\lambda}^T(t_j) \int_{t_j}^{t_{j+1}} \left(\mathbf{r}_H(t) + \mathbf{TE}_H(t)\right) dt + \boldsymbol{\lambda}^T(0)\left(\mathbf{r}_0 + \mathbf{r}_{0_H}\right), \quad (4.32) \\ &\approx \sum_{j=1}^{m} \boldsymbol{\lambda}^T(t_j) \left(\mathbf{le}_H\left(t_{j+1}; t_j, \tilde{\mathbf{Y}}_H^j\right) + (t_{j+1} - t_j)\mathbf{TE}_H(t_j)\right) + \boldsymbol{\lambda}^T(0)\left(\mathbf{r}_0 + \mathbf{r}_{0_H}\right), \end{aligned}$$

where $\mathbf{le}_H\left(t_{j+1}; t_j, \tilde{\mathbf{Y}}_H^j\right)$ is the local integration error.

Exactly as in Section 4.2, once the vector $\mathbf{l}$ is chosen, the value in each component of error vector $\mathbf{ge}_H(T_e)$ is estimated using the solution to the adjoint system in (4.29), the ODE local error, and the PDE truncation error.

The subscript H in above notations is called the *mesh characteristic length*. Hereafter, the mesh characteristic length H is used within these notations only if different meshes simultaneously exist and there is a need to specify which mesh is being referred to; otherwise, this subscript is omitted.

4.4 Error Norms, Error Indices, and Error Notations

It is important to seek quantitative information on the error of the obtained numerical solutions to decide if the numerical solutions can be trusted. In order to make an assessment of error in the numerical solutions obtained with different numerical methods discussed later on in this dissertation, we consider several definitions of error norms, definition of error indices, and several error notations for the variables in the system of Euler equations. In the following discussion, the error vector $\mathbf{e}$ can be replaced by the vector of global error $\mathbf{ge}$, the vector of spatial error $\mathbf{es}$, or the vector of temporal error $\mathbf{et}$.

4.4.1 Error Norms

Let $\mathbf{e}(t)$ be some vector of errors defined at discrete points $x_j \in \mathbb{R}$ $(j = 1, ..., N)$ at time t,

$$\mathbf{e}(t) = \left[e_1(t), e_2(t), ..., e_N(t)\right]^T , \tag{4.33}$$

we consider the following approximations to standard error norms:
L_1-norm:

$$\|\mathbf{e}(t)\|_{L_1} \approx \sum_{j=1}^{N-1} (x_{j+1} - x_j) \frac{(e_{j+1}(t_n) + e_j(t_n))}{2}. \tag{4.34}$$

L_2-norm:

$$\|\mathbf{e}(t)\|_{L_2} \approx \sqrt{\sum_{j=1}^{N-1} (x_{j+1} - x_j) \frac{\left((e_{j+1}(t))^2 + (e_j(t))^2\right)}{2}}. \tag{4.35}$$

L_∞-norm:

$$\|\mathbf{e}(t)\|_{L_\infty} = \max_{j=1}^{N} |e_j(t)|. \tag{4.36}$$

Depending on the property of the problem we are interested in, one norm may be more favored than others.

4.4.2 Error Indices

For the case that the exact error is available, it is reasonable to know how the estimate value of error is compared to the exact value. In order to know how reliable is the error estimate method, we compute the error index which is the ratio of the estimate error and the exact error. Let $\mathbf{e}_{true}(t)$ be the exact error and $\mathbf{e}(t)$ be the estimate value of $\mathbf{e}_{true}(t)$. The error index $eindex(\mathbf{e}(t))$ is then defined as follows:

$$eindex(\mathbf{e}(t)) = \frac{\|\mathbf{e}(t)\|}{\|\mathbf{e}_{true}(t)\|}. \tag{4.37}$$

The error norms in the above equation can be L_1-norm, L_2-norm, or L_∞-norm.

4.4.3 Error Notations for System of Euler Equations

In order to distinguish the errors for different interested quantities in the numerical solutions of the system of Euler equations, we use the superscript that represents the quantity of interest along with the error notations. More specifically, the notation $\mathbf{e}^q(t)$ is used to denote the error in numerical solutions of quantity q at t where $q = \rho, \mathbf{u}, p, E, \rho\mathbf{u}$, or ρE. Also we consider the following error notations:

$$\mathbf{e}^{\mathbf{W}}(t) = \left[\mathbf{e}^{\rho}(t), \mathbf{e}^{\mathbf{u}}(t), \mathbf{e}^{E}(t), \mathbf{e}^{p}(t)\right]^T, \tag{4.38}$$

where $\mathbf{W} = [\rho, \mathbf{u}, E, p]$, and:

$$\mathbf{e}^{\mathbf{U}}(t) = \left[\mathbf{e}^{\rho}(t), \mathbf{e}^{\rho\mathbf{u}}(t), \mathbf{e}^{\rho E}(t)\right]^T, \tag{4.39}$$

where $\mathbf{U}$ is the vector of conserved variables in the system of Euler equations.

4.5 Examples of the Adjoint-based Approach to the Global Error Estimate

4.5.1 Backward Differentiation Formula Method for Method-of-lines PDEs

The Backward Differentiation Formula (BDF) methods as described in Ascher and Petzold [3] are widely used for obtaining solutions to stiff differential equations and differential algebraic equations. The fixed leading coefficient BDF method is implemented in the DASSL DAE Solver.

The DASSL DAE Solver uses divided formulae to represent the numerial solution to DAEs. The divided difference $\mathbf{Y}[t_n, t_{n-1}, ..., t_{n-k}]$ on the nodal values $\mathbf{Y}(t_n)$, $\mathbf{Y}(t_{n-1})$, ..., $\mathbf{Y}(t_{n-k})$ is defined by:

$$\mathbf{Y}[t_n, t_{n-1}, ..., t_{n-k}] = \frac{\mathbf{Y}[t_n, t_{n-1}, ..., t_{n-k+1}] - \mathbf{Y}[t_{n-1}, t_{n-2} ..., t_{n-k}]}{t_n - t_{n-k}}, \tag{4.40}$$

where $\mathbf{Y}[t_n] = \mathbf{Y}(t_n)$ and $\mathbf{Y}[t_n, t_{n-1}] = \dfrac{\mathbf{Y}[t_n] - \mathbf{Y}[t_{n-1}]}{t_n - t_{n-1}}$.

Suppose that numerical solutions $\tilde{\mathbf{Y}}^n, \tilde{\mathbf{Y}}^{n-1}, \tilde{\mathbf{Y}}^{n-2}, ..., \tilde{\mathbf{Y}}^{n-k}$ are given at time levels $t_n, t_{n-1}, t_{n-2}, ..., t_{n-k}$, then the standard Newton divided difference form of the interpolating polynomial used by the DASSL DAE Solver to predict the numerical solution at any point in the interval $[t_{n-k}, t_{n+1}]$ is given by:

$$\begin{aligned}\mathbf{Y}^{n+1(p)}(t) = b_{0,n}(t)\ \tilde{\mathbf{Y}}[t_n] + b_{1,n}(t)\ \tilde{\mathbf{Y}}[t_n, t_{n-1}] + b_{2,n}(t)\ \tilde{\mathbf{Y}}[t_n, t_{n-1}, t_{n-2}] + ...\\ + b_{k,n}(t)\ \tilde{\mathbf{Y}}[t_n, t_{n-1}, t_{n-2}, ..., t_{n-k}],\end{aligned} \tag{4.41}$$

where:

$$b_{0,n}(t) = 1, b_{1,n}(t) = (t - t_n), \quad b_{2,n}(t) = (t - t_n)(t - t_{n-1}), ... \quad . \tag{4.42}$$

The predicted derivative may be similarly written as:

$$\begin{aligned}\frac{d\mathbf{Y}^{n+1(p)}(t)}{dt} = \frac{db_{1,n}(t)}{dt}\ \tilde{\mathbf{Y}}[t_n, t_{n-1}] + \frac{db_{2,n}(t)}{dt}\ \tilde{\mathbf{Y}}[t_n, t_{n-1}, t_{n-2}] + ...\\ + \frac{db_{k,n}(t)}{dt}\ \tilde{\mathbf{Y}}[t_n, t_{n-1}, t_{n-2}, ..., t_{n-k}].\end{aligned} \tag{4.43}$$

BDF codes such as the DASSL DAE Solver also make use of these polynomials to predict the numerical solution at the next time step. The system of equations solved for the new solution at time t_{n+1} is given by:

$$\frac{d\mathbf{Y}^{n+1(p)}}{dt}(t_{n+1}) - \frac{\alpha_s}{(t_{n+1} - t_n)} \left(\tilde{\mathbf{Y}}^{n+1} - \mathbf{Y}^{n+1(p)}(t_{n+1})\right) = \mathbf{G}(t_{n+1}, \tilde{\mathbf{Y}}^{n+1}), \tag{4.44}$$

where $\alpha_s = -\displaystyle\sum_{i=1}^{k} \frac{1}{i}$ for a method of order k. Substituting Equations (4.41)–(4.43) into Equation (4.44) and multiplying Equation (4.44) by $\dfrac{(t_{n+1} - t_n)}{\alpha_s}$ enables Equation (4.44) to be written in a more recognizable BDF form as:

$$\left(\tilde{\mathbf{Y}}^{n+1} - \tilde{\mathbf{Y}}_n\right) - \sum_{j=1}^{k}\left[b_{j,n} + \frac{(t_{n+1} - t_n)}{\alpha_s}\frac{db_{j,n}}{dt}\right]\tilde{\mathbf{Y}}[t_{n-j}, ..., t_n] = \frac{(t_{n+1} - t_n)}{(-\alpha_s)} f(t_{n+1}, \tilde{\mathbf{Y}}_{n+1}). \tag{4.45}$$

The above equation is solved to obtain the numerical solution $\tilde{\mathbf{Y}}^{n+1}$ at t_{n+1}. The numerical solution at any point t that lies between t_n and t_{n+1} is obtained using the interpolating polynomial defined as in (4.41) but with a different set of nodal values $\tilde{\mathbf{Y}}^{n+1}, \tilde{\mathbf{Y}}^{n}, ..., \tilde{\mathbf{Y}}^{n+1-k}$. This polynomial may be rewritten in Lagrange form as:

$$\tilde{\mathbf{Y}}(t) = \sum_{i=0}^{k} \prod_{j=0, j\neq i}^{k} \frac{(t - t_{n+1-j})}{(t_{n+1-i} - t_{n+1-j})} \tilde{\mathbf{Y}}^{n+1-i}. \tag{4.46}$$

We use a shorthand notation $\amalg_i^{k_1..k_2}(t)$ for $\displaystyle\prod_{j=k_1, j\neq i}^{k_2} \frac{(t - t_{n+1-j})}{(t_{n+1-i} - t_{n+1-j})}$ for convenience of exposition. Using this shorthand notation in the Lagrange form of $\tilde{\mathbf{Y}}(t)$ gives:

$$\tilde{\mathbf{Y}}(t) = \sum_{i=0}^{k} \amalg_i^{0..k}(t) \tilde{\mathbf{Y}}^{n+1-i}. \tag{4.47}$$

Equation (4.47) is used to evaluate the numerical solution at any point $t \in [t_n, t_{n+1}]$ using the discrete numerical solutions at t_{n+1}, t_n, ..., t_{n+1-k}.

4.5.2 Residual Error and Global Error Estimation using Approach of Cao and Petzold [21]

Cao and Petzold [21] give way to estimate the global error of the numerical solutions obtained with BDF method implemented in the DASSL DAE solver using the adjoint-based approach. The global error estimate in Cao and Petzold [21] will be reiterated as follows. The local solution on the interval $[t_n, t_{n+1}]$ used in [21] satisfies:

$$\begin{cases} \dot{\mathbf{V}}_{n+1}(t) = \mathbf{G}(\mathbf{V}_{n+1}(t), t) \quad & t \in [t_{n+1-k}, t_{n+1}], \\ \mathbf{V}_{n+1}(t_{n+1-k}) = \tilde{\mathbf{Y}}_{n+1-k}. \end{cases} \tag{4.48}$$

This local problem is different from the local problem defined in Equation (4.20). Let $\mathbf{S}_{n+1}(t)$ be the polynomial that interpolates $k+1$ points $(t_{n+1}, \mathbf{S}_{n+1}^0)$,$(t_n, \mathbf{S}_{n+1}^1)$, ..., $(t_{n-k+1}, \mathbf{S}_{n+1}^k)$ where $\mathbf{S}_{n+1}^i$ is the notation for $\mathbf{S}_{n+1}(t_{n+1-i})$ and $\mathbf{S}_{n+1}^i = \mathbf{V}_{n+1}(t_{n+1-i})$ for $i = 0, ..., k$. The polynomial

$\mathbf{S}_{n+1}(t)$ is then written in Lagrange form using the shorthand notation as in Equation (4.47) as follows:

$$\mathbf{S}_{n+1}(t) = \sum_{i=0}^{k} \amalg_i^{0..k}(t)\mathbf{S}_{n+1}^i. \tag{4.49}$$

Then for any t in $[t_n, t_{n+1}]$, we have:

$$\mathbf{V}_{n+1}(t) = \mathbf{S}_{n+1}(t) + \mathbf{IE}(t) \tag{4.50}$$

where $\mathbf{IE}(t)$ is interpolation error at t, and is defined as:

$$\mathbf{IE}(t) = (t - t_{n+1})(t - t_n)...(t - t_{n-k+1})\frac{\mathbf{V}_{n+1}^{(k+1)}(\tau)}{(k+1)!}, \tag{4.51}$$

for some value τ that lies in the interval $[t_{n-k+1}, t_{n+1}]$. For any t in $[t_n, t_{n+1}]$, the time derivative $\dot{\amalg}_i^{a..b}$ of $\amalg_i^{a..b}$ is given by:

$$\dot{\amalg}_i^{a..b} = \sum_{j=a, j\neq i}^{b} \frac{1}{(t_{n+1-i} - t_{n+1-j})} \prod_{l=a, l\neq i,j}^{b} \frac{(t - t_{n+1-l})}{(t_{n+1-i} - t_{n+1-l})}. \tag{4.52}$$

Differentiating Equation (4.50) gives:

$$\dot{\mathbf{V}}_{n+1}(t) = \sum_{i=0}^{k} \dot{\amalg}_i^{0..k}(t)\mathbf{S}_{n+1}^i + \dot{\mathbf{IE}}(t). \tag{4.53}$$

We now rewrite the residual error, $\mathbf{r}(t)$, for the perturbed system in (4.10) on the interval $[t_n, t_{n+1}]$ using the definition of local solution in Equation (4.48) as:

$$\mathbf{r}(t) = \dot{\tilde{\mathbf{Y}}}(t) - \mathbf{G}(\tilde{\mathbf{Y}}, t) = \dot{\tilde{\mathbf{Y}}}(t) - \dot{\mathbf{V}}_{n+1}(t) + \mathbf{G}(\mathbf{V}_{n+1}(t), t) - \mathbf{G}(\tilde{\mathbf{Y}}(t), t). \tag{4.54}$$

Cao and Petzold [21] assume that the function $\mathbf{G}(\mathbf{Y}, t)$ is sufficiently smooth and satisfies the Lipschitz condition, $\|\mathbf{G}(\mathbf{V}_{n+1}(t), t) - \mathbf{G}(\tilde{\mathbf{Y}}(t), t)\| \leq L\|\mathbf{V}_{n+1}(t) - \tilde{\mathbf{Y}}(t)\|$ for some constant L. It is pointed out in [21] that if $|\Delta t\ L| \leq 1$ then $\mathbf{V}_{n+1}(t) - \tilde{\mathbf{Y}}(t) = O(\Delta t^{k+1})$ while $\dot{\mathbf{V}}_{n+1}(t) - \dot{\tilde{\mathbf{Y}}}(t) = O(\Delta t^k)$. Consequently, the term $\mathbf{G}(\mathbf{V}_{n+1}(t), t) - \mathbf{G}(\tilde{\mathbf{Y}}(t), t)$ may be disregarded as not making a significant contribution to the residual error. The idea of disregarding $\mathbf{G}(\mathbf{V}_{n+1}(t), t)-$

$\mathbf{G}(\tilde{\mathbf{Y}}(t), t)$ from the residual error is also used in Enright [32]. The residual error in Equation (4.54) is thus given by:

$$\begin{aligned} \mathbf{r}(t) &\approx \dot{\tilde{\mathbf{Y}}}(t) - \dot{\mathbf{V}}_{n+1}(t), \\ &\approx \sum_{i=0}^{k} \dot{\amalg}_i^{0..k}(t)(\tilde{\mathbf{Y}}_{n+1-i} - \mathbf{S}_{n+1}^i) - \mathbf{I}\dot{\mathbf{E}}(t). \end{aligned} \tag{4.55}$$

This equation may be rewritten as:

$$\mathbf{r}(t) \approx \sum_{i=0}^{k} \dot{\amalg}_i^{0..k}(t)\mathbf{d}_{n+1}^i - I\dot{E}(t), \tag{4.56}$$

where $\mathbf{d}_{n+1}^i = \tilde{\mathbf{Y}}_{n+1-i} - \mathbf{S}_{n+1}^i$ for $i = 0, ..., k$. Cao and Petzold [21] state that $\mathbf{d}_{n+1}^i = \mathbf{O}((t_{n+1} - t_n)^{k+1})$ and so:

$$\mathbf{r}(t) \approx \sum_{i=0}^{k} (t_{n+1} - t_n)^{k+1} \dot{\amalg}_i^{0..k} \mathbf{C}_0 - \mathbf{I}\dot{\mathbf{E}}(t), \tag{4.57}$$

where $\mathbf{C}_0 = C_{k+1}\mathbf{Y}^{(k+1)}$ with C_{k+1} estimated using the DASSL DAE Solver and

$$\mathbf{I}\dot{\mathbf{E}}(t) \approx \frac{\mathbf{Y}^{(k+1)}(\tau)}{(k+1)!} \sum_{i=0}^{k} \prod_{j=0, j\neq i}^{k} (t - t_{n+1-i}) \approx \sum_{i=0}^{k} \frac{1}{i+1} \mathbf{Y}^{(k+1)}(t)(t_{n+1} - t_n)^k. \tag{4.58}$$

The calculation of the above term requires the estimation of $\mathbf{Y}^{(k+1)}(t)$ which is available within the DASSL DAE Solver. In the DASSL DAE Solver, the estimation of $\mathbf{Y}^{(k+1)}(t)$ is obtained from the divided difference representation of the numerical solution. The residual error may then be written as:

$$\mathbf{r}(t) \approx \mathbf{C}(t_{n+1} - t_n)(t_{n+1} - t_n)^k, \tag{4.59}$$

where $\mathbf{C}(t_{n+1} - t_n)$ is calculated using Equations (24) and (26) of Cao and Petzold [21] as follows:

$$\mathbf{C}(t_{n+1} - t_n) \approx C_{k+1}\mathbf{Y}^{(k+1)} \sum_{i=0}^{k} \dot{\amalg}_i^{0..k}(t_{n+1} - t_n) + \sum_{i=0}^{k} \frac{1}{(i+1)} \mathbf{Y}^{(k+1)}. \tag{4.60}$$

Cao and Petzold [21] use residual error $\mathbf{r}(t)$ defined in Equation (4.59) to rewrite Equation (4.19) as follows:

$$\mathbf{l}^T\mathbf{et}(T_e) \approx \sum_{j=1}^{m} \boldsymbol{\lambda}^T(t_j)\mathbf{C}(t_{j+1}-t_j)(t_{j+1}-t_j)^{k+1} + \boldsymbol{\lambda}^T(0)\mathbf{r}_0. \tag{4.61}$$

This equation is used by Cao and Petzold [21] to estimate the global error. Therefore, in order to estimate the global error using Cao and Petzold's approach in [21] with the DASSL DAE Solver, we need to solve the ODE system in (4.9), solve the adjoint system in (4.13), and evaluate the residual error using the available data given by DASSL DAE Solver when solved for the ODE system in (4.9). For the global error estimate using our approach as using Equation (4.25), we need also to evaluate the local error $\mathbf{le}\left(t_{j+1}; t_j, \tilde{\mathbf{Y}}^j\right)$ for each time integration step. We will discuss how to evaluate the local error using residual error sampling in the following section.

4.5.3 Estimation of the Local Error and the Truncation Error

4.5.3.1 Estimation of the Local Error using Residual Error Sampling

Consider the local problem given by Equation (4.20) and the local error given by Equation (4.22). Define, $\mathbf{P}(t)$, a polynomial of degree k on $[t_{n+1-k}, t_{n+1}]$, that satisfies:

$$\begin{cases} \mathbf{P}(t_{n+1-j}) & = \tilde{\mathbf{Y}}^{n+1-j}, \quad j = 1, ..., k, \\ \mathbf{P}(t_{n+1}) & = \mathbf{Z}_{n+1}(t_{n+1}), \end{cases} \tag{4.62}$$

where $\mathbf{Z}_{n+1}(t_{n+1})$ is the solution to the system in (4.20). Then $\mathbf{P}(t)$ is an interpolation polynomial of degree k that interpolates $k+1$ known points on the interval $[t_{n+1-k}, t_{n+1}]$ and approximates $\mathbf{Z}_{n+1}(t)$ on the interval $[t_n, t_{n+1}]$. The interpolation polynomial $\mathbf{P}(t)$ may also be written in Lagrange form:

$$\mathbf{P}(t) = \sum_{i=1}^{k} \amalg_i^{0..k}(t)\tilde{\mathbf{Y}}^{n+1-i} + \amalg_0^{0..k}(t)\mathbf{Z}_{n+1}(t_{n+1}). \tag{4.63}$$

Since $\mathbf{P}(t)$ approximates the local solution on the interval $[t_n, t_{n+1}]$, we have:

$$\mathbf{Z}_{n+1}(t) \approx \mathbf{P}(t) + \mathbf{Q}(t), \tag{4.64}$$

where the term $\mathbf{Q}$ may be written as a divided difference term as follows:

$$\mathbf{Q}(t) \approx \pi(t)[\mathbf{Z}_{n+1}(t_{n+1}), \tilde{\mathbf{Y}}^{n}, \tilde{\mathbf{Y}}^{n-1}, ..., \tilde{\mathbf{Y}}^{n-k}], \tag{4.65}$$

and $\pi(t) = (t - t_{n+1})(t - t_n)...(t - t_{n+1-k})$. As mentioned in Section 4.5.2, Cao and Petzold [21] assume the term $\mathbf{G}(\mathbf{Z}_{n+1}(t), t) - \mathbf{G}(\tilde{\mathbf{Y}}(t), t)$ may be disregarded as not making a significant contribution to the overall residual error. The residual error, $\mathbf{r}(t)$, in Equation (4.23) may be then estimated as follows:

$$\mathbf{r}(t) \approx \dot{\tilde{\mathbf{Y}}}(t) - \dot{\mathbf{Z}}_{n+1}(t). \tag{4.66}$$

From Equations (4.47), (4.63), (4.64), and (4.65), we have:

$$\tilde{\mathbf{Y}}(t) - \mathbf{Z}_{n+1}(t) \approx -\amalg_0^{0..k}\, \mathbf{le}(t_{n+1}; t_n, \tilde{\mathbf{Y}}^{n}) - \pi(t)[\mathbf{Z}_{n+1}(t_{n+1}), \tilde{\mathbf{Y}}^{n}, \tilde{\mathbf{Y}}^{n-1}, ..., \tilde{\mathbf{Y}}^{n-k}]. \tag{4.67}$$

Therefore:

$$\mathbf{r}(t) \approx -\dot{\amalg}_0^{0..k}\mathbf{le}(t_{n+1}; t_n, \tilde{\mathbf{Y}}^{n}) - \dot{\pi}(t)[\mathbf{Z}_{n+1}(t_{n+1}), \tilde{\mathbf{Y}}^{n}, \tilde{\mathbf{Y}}^{n-1}, ..., \tilde{\mathbf{Y}}^{n-k}]. \tag{4.68}$$

Two quantities that determine the form of the estimated residual over a step are $\mathbf{le}(t_{n+1}; t_n, \tilde{\mathbf{Y}}^{n})$ and $[\mathbf{Z}_{n+1}(t_{n+1}), \tilde{\mathbf{Y}}^{n}, \tilde{\mathbf{Y}}^{n-1}, ..., \tilde{\mathbf{Y}}^{n-k}]$. So with two samples of the residual error at $t1$, and $t2$ in the interval $[t_n, t_{n+1}]$, we form the system:

$$\begin{bmatrix} \dot{\amalg}_0^{0..k}(t_1) & \dot{\pi}(t_1) \\ \dot{\amalg}_0^{0..k}(t_2) & \dot{\pi}(t_2) \end{bmatrix} \begin{bmatrix} -\mathbf{le}(t_{n+1}; t_n, \tilde{\mathbf{Y}}^{n}) \\ -[\mathbf{Z}_{n+1}(t_{n+1}), \tilde{\mathbf{Y}}^{n}, \tilde{\mathbf{Y}}^{n-1}, ..., \tilde{\mathbf{Y}}^{n-k}] \end{bmatrix} = \begin{bmatrix} \mathbf{r}(t_1) \\ \mathbf{r}(t_2) \end{bmatrix}. \tag{4.69}$$

Solve the above system with two samples of the residual error for the time interval $[t_n, t_{n+1}]$, we obtain the local error $\mathbf{le}(t_{n+1}; t_n, \tilde{\mathbf{Y}}^{n})$ per time step.

4.5.3.2 Truncation Error Estimation by Richardson Extrapolation

Berzins [13] proposed a method to estimate the spatial truncation error by Richardson extrapolation. We will reiterate the method here and use it in the global error estimation for the numerical solutions of PDEs proposed in this chapter.

Consider two different discretized meshes Ω_h and Ω_H of the PDE system in (4.1) where h and H represent the mesh characteristic length and $h = \frac{H}{2}$. The mesh Ω_h is the actual mesh used to compute the numerical solution to the PDEs and also be the "fine" mesh in the Richardson extrapolation; the mesh Ω_H is the "coarse" mesh. The truncation error for the mesh Ω_h as defined in Section 4.3 is:

$$\mathbf{TE}_h(t) = \dot{\mathbf{Y}}[h](t) - \mathbf{G}_h(\mathbf{Y}[h](t), t). \tag{4.70}$$

Let $\mathbf{Y}_H^h(t)$ and $\dot{\mathbf{Y}}_H^h(t)$ be the restriction of the numerical solution $\mathbf{Y}_h(t)$ and the numerical derivative $\dot{\mathbf{Y}}_h(t)$ from the "fine" mesh to the "coarse" mesh. The truncation error for the "coarse" mesh is then obtained by evaluating the following equation as proposed in Berzins [13]:

$$\mathbf{TE}_H(t) = \frac{4}{3}\left[\dot{\mathbf{Y}}_H(t) - \mathbf{G}_H(t, U_H^h(t))\right] + \frac{4}{3}[\dot{\mathbf{et}}_H(t) - \frac{\partial \mathbf{G}_H}{\partial \mathbf{Y}_H(t)}\mathbf{et}_H(t)]. \tag{4.71}$$

The trunction error at the grid nodes that belong to both "fine" mesh and "coarse" mesh is then given by:

$$[\mathbf{TE}_h(t)]_{2i-1} = \frac{1}{4}[\mathbf{TE}_H(t)]_i. \tag{4.72}$$

The truncation error at grid nodes that are in "fine" mesh, but not in "coarse" mesh is then obtained by extrapolating using the following equation:

$$[\mathbf{TE}_h(t)]_{2i} = \frac{1}{8}([\mathbf{TE}_H(t]_i + [\mathbf{TE}_H(t)]_{i+1}). \tag{4.73}$$

This method of Richardson extrapolation for truncation error estimation as discussed above is based on the assumption that the spatial discretization error is second order in space in terms of the mesh characteristic length.

4.5.4 Numerical Results

4.5.4.1 Adjoint-based Global Error Estimate for ODEs

We consider six examples below for testing of the adjoint-based global error estimate. In these examples: Examples 1–4 are from [21] and Examples 5–6 are from [81].

Example 1.

$$\begin{cases} \dot{\mathbf{Y}} & = a\mathbf{Y}, \qquad 0 < t \le T_e, \\ \mathbf{Y}(0) & = \mathbf{Y}_0. \end{cases} \tag{4.74}$$

This problem is solved with three different cases:

$$\begin{aligned} a &= 1, & \mathbf{Y}_0 &= \left[10^{-4}\right], & T_e &= 10.0, \\ a &= -1, & \mathbf{Y}_0 &= [1.0], & T_e &= 1.0, \\ a &= -20, & \mathbf{Y}_0 &= [1.0], & T_e &= 1.0. \end{aligned}$$

Example 2.

$$\begin{cases} \dot{\mathbf{Y}} &= -(0.25 + \sin \pi t)\mathbf{Y} \cdot \mathbf{Y}, \qquad 0 < t \leq T_e, \\ \mathbf{Y}(0) &= [1.0]. \end{cases} \tag{4.75}$$

The analytical solution at t is $\mathbf{Y}(t) = [\pi/(\pi + 1 + 0.25\pi t - \cos \pi t)]$ and we consider the time integration at $T_e = 1.0$.

Example 3.

$$\begin{cases} \dot{\mathbf{Y}} &= \begin{bmatrix} \dfrac{1}{2(1+t)} & -2t \\ 2t & \frac{1}{2(1+t)} \end{bmatrix} \mathbf{Y} \qquad 0 < t \leq T_e, \\ \mathbf{Y}(0) &= [1.0 \quad 0.0]^T. \end{cases} \tag{4.76}$$

The analytical solution at t is $\mathbf{Y}(t) = \left[(1+t)^{\frac{1}{2}} \cos(t^2),\ (1+t)^{\frac{1}{2}} \sin(t^2)\right]^T$ and we consider the time integration at $T_e = 10.0$.

Example 4.

$$\begin{cases} \dot{\mathbf{Y}} &= \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix} \mathbf{Y} \qquad 0 < t \leq T_e \\ \mathbf{Y}(0) &= \left[2 \times 10^{-4} \quad 0.0\right]^T. \end{cases} \tag{4.77}$$

The analytical solution at t is $\mathbf{Y}(t) = \left[10^{-4}(e^t + e^{-t}), 10^{-4}(e^{-t} - e^t)\right]^T$ and we consider the time integration at $T_e = 10.0$.

Example 5.

$$\begin{cases} \dot{\mathbf{Y}} &= \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \mathbf{Y} \qquad 0 < t \leq T_e \\ \mathbf{Y}(0) &= [0.0 \quad 1.0]^T. \end{cases} \tag{4.78}$$

The analytical solution at t is $\mathbf{Y}(t) = [\sin(t), \cos(t)]^T$ and we consider the time integration at $T_e = 50.0$.

Example 6. Let $\mathbf{Y} = [y_1, y_2, y_3, y_4, y_5]^T$. Consider the ODE system:

$$\begin{cases} \dot{\mathbf{Y}} &= \begin{bmatrix} y_1 \\ y_2 + y_1 y_1 \\ y_3 + y_1 y_2 \\ y_4 + y_1 y_3 + y_2 y_2 \\ y_5 + y_1 y_4 + y_2 y_3 \end{bmatrix} \qquad 0 < t \le T_e \\ \mathbf{Y}(0) &= [1.0, \quad 1.0, \quad 0.5, \quad 0.5, \quad 0.25]^T . \end{cases} \tag{4.79}$$

The analytical solution at t is $\mathbf{Y} = \left[e^t, e^{2t}, \ \frac{1}{2}e^{3t}, \ \frac{1}{2}e^{4t}, \ \frac{1}{2}e^{5t}\right]^T$ and we consider the time integration at $T_e = 50.0$.

The ODE systems of Examples 1–6 are solved using DASSL DAE Solver with local error tolerances of 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, and 10^{-10}. Applying the adjoint global error estimate given by Equation (4.25) where the local error is determined by sampling the residual error at two points per time interval as mentioned in Section 4.5.3.1, we obtain the error indices (the ratio of estimated global error and exact global error L_2-norms) in Table 4.1. As shown in Table 4.1, our approach of adjoint global error estimate is reliable when the obtained error indices are very close to one (the ideal value of error indices) in most cases. For the result of Example 1 with $\lambda = -20$, the obtained error indices for large TOL (such as 10^{-4} and 10^{-5}) are not close to one as the term $\mathbf{G}(\mathbf{Z}_{n+1}(t), t) - \mathbf{G}(\tilde{\mathbf{Y}}(t), t)$ makes a significant contribution to the residual error but being disregarded from the estimation of the residual error as mentioned in Section 4.5.2.

In order to compare the obtained results using our approach and Cao and Petzold's approach in [21], we include in Table 4.2 the error indices of Cao and Petzold's global error estimate for

Table 4.1. Error indices $eindex(\mathbf{et}(T_e))$ of the estimated adjoint-based global errors for numerical solutions to Examples 1–6 using DASSL DAE Solver and the residual error sampling technique in Section 4.5.3.1 with different values of local error tolerance (TOL).

	TOL						
Example	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
$1(\lambda = 1)$	0.95	0.95	0.95	0.96	0.96	0.96	0.95
$1(\lambda = -1)$	0.96	0.97	0.99	0.96	0.96	0.96	0.98
$1(\lambda = -20)$	2.71	2.63	1.80	1.07	1.05	0.96	1.03
2	0.97	0.98	0.98	1.07	1.00	0.99	0.99
3	0.96	0.95	0.97	0.98	0.98	0.97	0.97
4	0.95	0.95	0.95	0.95	0.97	0.96	0.98
5	0.96	0.97	0.97	0.97	0.97	0.96	0.98
6	0.96	0.97	0.97	0.97	0.97	0.98	0.98

Table 4.2. Error indices $eindex(\mathbf{et}(T_e))$ of the estimated adjoint-based global errors for numerical solutions to Examples 1–6 using DASSL DAE Solver and Cao and Petzold's approach described in Section 4.5.2 with different values of local error tolerance (TOL).

	TOL						
Example	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
$1(\lambda = 1)$	7.23	7.09	9.12	8.95	8.54	16.72	9.18
$1(\lambda = -1)$	3.22	2.96	129.2	6.74	2.45	8.67	5.73
$1(\lambda = -20)$	1.59	0.46	0.45	2.08	7.63	10.12	10.36
2	9.27	106.9	19.36	72.67	13.98	16.38	0.31
3	13.02	13.66	13.00	11.59	10.92	10.77	11.35
4	7.25	7.08	6.45	8.68	12.10	15.70	12.45
5	8.89	15.04	7.98	1.45	7.63	8.64	4.16
6	10.54	14.31	8.09	12.94	4.62	8.35	13.86

numerical solutions to Examples 1–6 using DASSL DAE Solver. When comparing the data shown in Table 4.1 and Table 4.2, here are some comments that we have. Though we need an extra of two residual error samplings (this is equivalent to two evaluations of the system in (4.9)) per time step; the result shown in Table 4.1 is much more accurate than the result shown in Table 4.2. As shown in Table 4.2, the global error estimate gives a wide range of error indices, and often the global error estimate for Examples 1–6 is overestimated. Although the error is sometimes overestimated using the approach of Cao and Petzold [21], it helps to control well the global error in [21].

4.5.4.2 Adjoint-based Global Error Estimate for PDEs

We consider here a PDE problem in one-dimensional space with a nonlinear source term and a reaction diffusion equation:

$$\frac{\partial y}{\partial t} = \frac{\partial^2 y}{\partial x^2} + y^2(1-y) \quad (x,t) \in (0,10) \times (0, T_e] \tag{4.80}$$

with Dirichlet boundary conditions and initial conditions consistent with the analytic solution of:

$$y(x,t) = \frac{1}{1 + e^{p(x-pt)}} \tag{4.81}$$

where $p = 0.5\sqrt{2}$. We consider the numerical solution at $T_e = 1.0$.

When solving the above problem, we discretize this PDE with different numbers of mesh points (NPTS) and approximate the spatial derivatives with a second order central finite difference. The discretized ODEs are then solved using DASSL DAE solver with different values of TOL. Applying the adjoint-based global error estimate given by Equation (4.32) where the trunction error is obtained via Richardson extrapolation discussed in Section 4.5.3.2 and the local error is determined by sampling the residual error at two points per time interval as mentioned in Section 4.5.3.1, the obtained error indices are shown in Table 4.3. As shown in Table 4.3, our approach of adjoint-based global error estimate gives a good approximation to the overall error in the numerical solutions of PDEs when the obtained error indices are very close to the ideal value of one.

4.6 Summary

Spatial and temporal errors are the error sources associated with the discretization of time-dependent PDEs when using the method of lines. We have presented in this chapter the adjoint-based approach for estimating both temporal and spatial errors in the numerical solutions of time-dependent PDEs. Our adjoint-based temporal error estimate is based upon the adjoint ODE error estimate proposed by Cao and Petzold [21], but improved with the addition of the residual error sampling technique presented in this chapter. Making use of the similarity between the systems of spatial and temporal error evolution, we derived the discrete adjoint-based approach for spatial error estimate in which the PDE trunction error obtained from Richardson extrapolation is being used. Numerical results presented in this chapter have shown that our adjoint-based error estimate gives a reliable and close estimate of the true global error. We will use this approach towards the global error estimation of the numerical solutions obtained from the Improved Production Implicit-Continuous Eulerian (IMPICE) method in Chapter 8.

Table 4.3. Error indices $eindex(\mathbf{ge}(T_e))$ of the estimated adjoint-based global errors for numerical solutions to the PDE problem discussed in Section 4.5.4.2. The numerical solutions to this problem are obtained for different number of mesh points (NPTS) of spatial discretization. The discretized ODEs are solved with DASSL DAE solver using different values of local error tolerance (TOL).

	TOL						
$NPTS$	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}	10^{-10}
11	0.90	0.89	0.88	0.88	0.87	0.87	0.87
21	0.98	0.97	0.95	0.97	0.97	0.97	0.97
41	1.01	1.01	0.96	0.99	0.99	0.99	0.99
81	0.99	1.00	0.89	1.00	1.00	1.01	1.01
161	0.98	0.99	0.92	1.01	1.01	1.01	1.01
321	0.97	0.98	0.98	1.02	1.01	1.01	1.01

CHAPTER 5

SOLVING TIME-DEPENDENT PDES USING THE MATERIAL POINT METHOD

The Material Point Method is a particle method [109]. It is used in computational solid dynamics to simulate large material deformations as the spatial mesh in MPM remains fixed throughout the calculation. As mentioned in Steffen [109], the method is a mixed Lagrangian and Eulerian method with particles representing the discrete Lagrangian state of a material. Though the method is widely used in many applications in the field of solid dynamics, there is not much analysis of the method. In this chapter, we will study in depth the accuracy of a variation of MPM proposed for gas dynamics on a well-known test problem in one-dimensional space. The test problem is Sod's shocktube problem presented in Sod [108] where the motion of the compressible and inviscid fluid is governed by the one-dimensional system of Euler equations. Though this analysis is done for a specific variation of MPM, it is described in a way that can be applied to other versions of MPM. We will perform analysis on two sources of error: errors introduced when information from particles is projected onto the grid, as well as errors introduced when particles cross grid cells.

The content of this chapter is organized as follows. Section 5.1 includes the spatial discretization of MPM. In Section 5.2, we describe an abbreviated form of the computational method of MPM discussed in Steffen *et al.* [111]. The application to gas dynamics of MPM is carried out in Section 5.3. In Section 5.4, we will experiment with numerical solutions of Sod's shocktube problem corresponding to different cases of particles' distribution and different initial numbers of particles per cell used in this variation of MPM. We will analyze the method's time integration error and space discretization error in Section 5.5 and Section 5.6 respectively. The discussion on the combined error for numerical solutions of Sod's shocktube problem is presented in Section 5.7. Finally, Section 5.8 is a summary of this chapter.

5.1 MPM Spatial Discretization

MPM is a particle method based on the Finite Element Method in which the computational domain $\Omega = [a_1, b_1] \subset \mathbb{R}$ is discretized into a mesh on which a set of particles are placed; see Figure 5.1. Let N_p be the number of particles in the computational domain Ω, p be the subscript index of the particle where $p = 1, .., N_p$, and Ω_p be the particle domain of particle p. Each particle p (referred to as the material point) is associated with the particle volume, V_p, the particle position, x_p, the particle mass, m_p, and the particle momentum, P_p. In MPM, the motion of these particles is solved on a background grid. The background grid is a set of points (referred to as nodes or grid nodes) that divides the computational domain into cells. Let N be the number of nodes in the computational domain Ω, and j be the subscript index of the nodes where $j = 1, .., N$. Unlike particles, the position of nodes is fixed at x_j. The spatial domain of a cell j where $j = 1, .., N - 1$ is denoted as Ω_j where $\Omega_j = [x_j, x_{j+1}]$. We consider a uniform background grid with the mesh spacing of h and the same initial number of particles in each cell. The movement of particles between cells is based on the nodal velocity and the nodal acceleration whose calculation will be discussed in detail in Section 5.2. The following discussion will explain how to approximate a function value using particles' values, how to represent a continuous function using discrete nodal data, and how to map from particles to grid nodes.

5.1.1 Particle Basis Functions

Let $f(x)$ be a function defined on the computational domain $x \in \Omega$ and f_p be the value of function f at particle p. The approximation to the function $f(x)$ in terms of particle values is written as:

$$f(x) \approx \sum_{p=1}^{N_p} f_p \chi_p(x) \qquad \forall x \in [a_1, b_1], \tag{5.1}$$

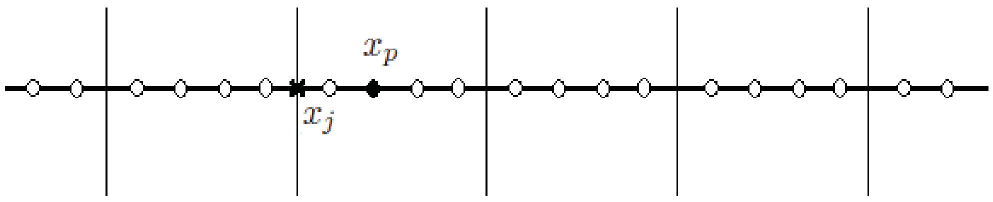


Figure 5.1. MPM spatial discretization in one-dimensional space.

where $\chi_p(x)$ is the basis function associated with particle p. In the original form of the MPM, Delta functions are used for the particle basis functions and the basis function for particle p is defined by:

$$\chi_p(x) = \delta(x - x_p)V_p, \quad p = 1, ..., N_p, \tag{5.2}$$

where:

$$\delta(x) = \begin{cases} 1 & \text{if} \quad x = 0, \\ 0 & \text{otherwise}, \end{cases} \tag{5.3}$$

and the paticle volume, V_p, will be defined later in this section. Bardenhagen and Kober [5] use the piecewise constant form for the particle basis function which is given by:

$$\chi_p(x) = \begin{cases} 1 & \text{if} \quad x \in \Omega_p, \\ 0 & \text{otherwise} \end{cases} \tag{5.4}$$

where Ω_p is the interval $[x_p - h_p/2, x_p + h_p/2]$ and h_p is the particle width. This has the advantage that the functions form a partition of unity on the interval $[a_1, b_1]$:

$$\sum_{p=1}^{N_p} \chi_p(x) = 1 \quad \forall x \in [a_1, b_1]. \tag{5.5}$$

For the case when the particle basis function is defined in Equation (5.4), the particle volume is then defined by:

$$V_p = \int_{\Omega} \chi_p(x)dx. \tag{5.6}$$

5.1.2 Grid Basis Functions

Let $g(x)$ be a function defined on the computational domain $x \in \Omega$ and g_j be the value of function g at node j. The continuous representation function $g(x)$ using discrete data g_j $(j = 0, ..., N)$ is given by:

$$g(x) = \sum_{j=1}^{N} g_j S_j(x), \tag{5.7}$$

where $S_j(x)$ is a grid basis function at node j and these basis functions form a partition of unity on the interval $[a_1, b_1]$. The most commonly used grid basis functions are the piecewise-linear basis functions; see Figure 5.2. The piecewise-linear grid basis function for node j is given by:

$$S_j(x) = \begin{cases} 1 - \dfrac{|x - x_j|}{h} & \text{if} \quad |x - x_j| < h, \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

where h is mesh spacing.

5.1.3 Mapping from Particles to Grid

Let define $\bar{S}_{jp}$ and $\bar{G}_{jp}$ as follows:

$$\bar{S}_{jp} = \frac{1}{V_p} \int_\Omega S_j(x) \chi_p(x) dx, \tag{5.9}$$

and

$$\bar{G}_{jp} = \frac{1}{V_p} \int_\Omega \frac{dS_j}{dx}(x) \chi_p(x) dx. \tag{5.10}$$

In the case of the standard MPM when Delta functions are used for the particle basis functions and the linear basis functions are used for the grid basis functions, then, according to [5],

$$\bar{S}_{jp} = S_j(x_p), \tag{5.11}$$

and

$$\bar{G}_{jp} = \frac{dS_j}{dx}(x_p). \tag{5.12}$$

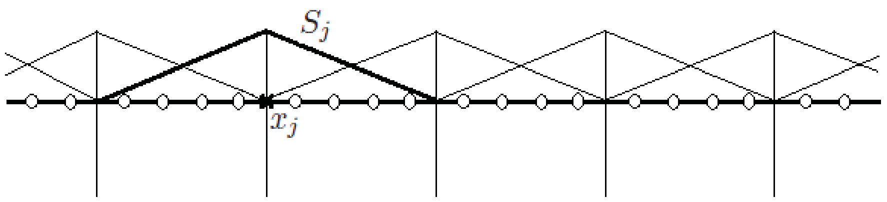


Figure 5.2. Piecewise-linear basis functions.

Consider the particle basis functions $\chi_p(x)$ that form a partition of unity on the domain Ω as shown in Equation (5.5) and function $\bar{S}_{jp}$ which is defined in Equation (5.9). We define two different mappings as given by:

$$f(x_j) = \sum_{p=1}^{N_p} f(x_p)\ \bar{S}_{jp}, \tag{5.13}$$

and

$$G(x_j) = -\sum_{p=1}^{N_p} \frac{dG}{dx}(x_p)\ \bar{G}_{jp} V_p. \tag{5.14}$$

These mappings are used to map particle values onto a value at node j.

5.2 MPM Computational Method

Given an initial distribution of particles on the domain at time $t = t_0$, each particle p is assigned a point mass, m_p^0, which is defined in terms of density as given by:

$$m_p^0 = \int_{\Omega_p} \rho(x,0)\chi_p(x)dx. \tag{5.15}$$

The initial particle density average, ρ_p^0, may also be defined by:

$$\rho_p^0 = m_p^0 / V_p^0, \tag{5.16}$$

where V_p^0 is the initial particle volume. The particle p is also initially assigned a momentum, P_p^0, which is defined in terms of density and velocity by:

$$P_p^0 = \int_{\Omega_p} \rho(x,0)u(x,0)\chi_p(x)dx. \tag{5.17}$$

The Cauchy stresses are:

$$\sigma_p^0 = \int_{\Omega_p} \sigma(x,0)\frac{\chi_p(x)}{V_p^0}dx, \tag{5.18}$$

where $\sigma(x,0)$ is continuum bodies initial Cauchy stress. In the most general case, the stress tensor is given by $\sigma = -pI + T$, where p is the pressure, T denotes the viscous stress tensor and I is an identity tensor whose size is same as the modeling dimension. In a perfect fluid model such as the gas dynamics problem considered here, the stress at a particle is equal to the pressure:

$$\sigma_p = -p_p. \tag{5.19}$$

5.2.1 Mesh and Particle Movement per Time Step

This subsection describes an abbreviated form of the original MPM; a detail description of this method can be found in Steffen *et al.* [111]. This abbreviated description includes the steps to advance the numerical solution from time level t_n to t_{n+1}. Since the motion of the particles is solved on a background grid, the particle data (the particle mass and the particle momentum) are projected onto the nodes at the start of the time step. The nodal mass, m_j^n, is approximated using the mass of the particles via the lumped mass matrix form of MPM in [113]. As given by Equation (5.13), the nodal mass is as follows:

$$m_j^n = \sum_{p=1}^{N_p} \bar{S}_{jp} m_p^n, \qquad j = 1, ..., N. \tag{5.20}$$

Similarly, the nodal momemtum, P_j^n, is given by:

$$P_j^n = \sum_{p=1}^{N_p} \bar{S}_{jp} m_p^n u_p^n, \qquad j = 1, ..., N. \tag{5.21}$$

where u_p^n is the particle velocity at t_n. The movement of the particles is determined by the velocity and the acceleration at the nodes on the background grid. The nodal velocity, u_j^n, is calculated from the nodal mass, m_j^n, and the nodal momentum, P_j^n, which is given by:

$$u_j^n = \frac{P_j^n}{m_j^n}. \tag{5.22}$$

Assuming that the nodal internal force, $Fint_j^n$, is defined, then the nodal acceleration, a_j^n, is given by:

$$a_j^n = \frac{Fint_j^n}{m_j^n} \ . \tag{5.23}$$

Since $\sigma = \dfrac{dFint}{dx}$, the following equation is derived from Equations (5.14) and (5.19):

$$Fint_j^n = \sum_{p=1}^{N_p} p_p^n \bar{G}_{jp} V_p^n, \tag{5.24}$$

where p_p^n is the particle pressure at t_n. The relationship between the acceleration, the velocity, and the displacement of the material is given by kinematics:

$$\dot{u}(x,t) = a(x,t), \tag{5.25}$$

$$\dot{x}(x,t) = u(x,t). \tag{5.26}$$

The nodal velocity at the end of Lagrangian step is calculated using the Euler method for the time derivative of velocity in Equation (5.25) as follows:

$$u_j^{n+1} = u_j^n + a_j^n \Delta t, \tag{5.27}$$

where $\Delta t = t_{n+1} - t_n$. The particle velocity and location are time-advanced using the Euler method for the ODEs (5.25) and (5.26) where the function on the right side of these ODEs is evaluated using the projection of nodal values. The updated particle velocity and location are then given by:

$$u_p^{n+1} = u_p^n + \sum_{j=1}^{N} \bar{S}_{jp} a_j^n \Delta t, \tag{5.28}$$

$$x_p^{n+1} = x_p^n + \sum_{j=1}^{N} \bar{S}_{jp} u_j^{n+1} \Delta t. \tag{5.29}$$

Remark If u_p^{n+1} was used to replace the sum in the right side of Equation (5.29), the time integration method could be viewed as a first-order Runge-Kutta-Nystrom method, Chawla and Subramanian [23].

5.3 Application to Gas Dynamics

At the start of the time step t_n, the approximate particle volume for particle p can be calculated using the number of particles in the cell that contains the particle, N_j^n, by:

$$V_p^n = \frac{h}{N_j^n}, \tag{5.30}$$

where h is mesh spacing. While this is a reasonable approximation for compressible flows, and was first used by [70], it represents a departure from the standard MPM approach for solid mechanics, in which the volumes associated with particles are tracked; see Steffen *et al.* [110, 111, 112] for an analysis of this case. The particle's mass is calculated from the density and the volume of the particle as:

$$m_p^n = \rho_p^n V_p^n. \tag{5.31}$$

The nodal mass is calculated from the projection of the particle properties as shown in (5.20) and nodal momentum is given by Equation (5.21). The nodal velocity is calculated from the mass and the momentum of the node as given by Equation(5.22). The nodal force may be written as the jump on the averaged particle pressures:

$$Fint_j^n = p_j^{n(-)} - p_j^{n(+)}, \tag{5.32}$$

where

$$p_j^{n(-)} = \sum_{p:x_p^n \in \Omega_{j-1}} p_p^n \frac{1}{N_{j-1}^n}, \tag{5.33}$$

$$p_j^{n(+)} = \sum_{p:x_p^n \in \Omega_j} p_p^n \frac{1}{N_j^n}. \tag{5.34}$$

The internal force at a node is thus equal to the averaged pressure drop around that node. The nodal acceleration is calculated from the nodal force and the nodal mass as follows:

$$a_j^n = \frac{p_j^{n(-)} - p_j^{n(+)}}{m_j^n}. \tag{5.35}$$

The particle velocity and location are then updated using Equations (5.27)–(5.29). This method of force calculation has been developed here as being more appropriate for compressible gas dynamics as it assumes that the particles within a cell have the same volume.

5.3.1 Particle Energy, Density and Pressure Update

Once the nodal velocity has been determined using (5.28), it is possible to update the velocity gradient and hence calculate the particle density, ρ_p^{n+1}, and the particle energy, e_p^{n+1}, for the next time step following Equations (3.48) and (3.49) by:

$$e_p^{n+1} = e_p^n - \Delta t \frac{p_p^n}{\rho_p^n} \frac{\partial u_p^{n+1}}{\partial x}, \tag{5.36}$$

and

$$\rho_p^{n+1} = \rho_p^n - \Delta t \rho_p^n \frac{\partial u_p^{n+1}}{\partial x}, \tag{5.37}$$

where the particle velocity gradient, $\frac{\partial u_p^{n+1}}{\partial x}$, is calculated using nodal velocities and the gradients of the nodal basis functions as given by:

$$\frac{\partial u_p^{n+1}}{\partial x} = \sum_{j=1}^{N} \bar{G}_{jp} u_j^{n+1}, \tag{5.38}$$

where $\bar{G}_{jp}$ is defined by Equation (5.10). Note that all particles in the same cell have the same velocity gradient as calculated using Equation (5.38). The updated particle pressure is then calculated using equation of state (3.36) and an added viscosity term as follows:

$$p_p^{n+1} = (\gamma - 1)\rho_p^{n+1} e_p^{n+1} + \nu_p^{n+1}, \tag{5.39}$$

where the term ν_p^{n+1} is a standard artificial viscosity term which is defined by:

$$\nu_p^{n+1} = \begin{cases} C^2 dx^2 \rho_p^{n+1} \left(\frac{\partial u_p^{n+1}}{\partial x} \right)^2 & \text{if} \quad \frac{\partial u_p^{n+1}}{\partial x} \leq 0 \\ 0 & \text{otherwise,} \end{cases}$$

where $C = 2.5$. This form of artificial viscosity was used by Monaghan and Gingold [89, 90] to reduce oscillations in the numerical solutions of the SPH methods. This formula exploits the property of shock front that the gradient of velocity is less than zero. Using Equation (5.39) to obtain the value of the pressure/density ratio and substitute into Equation (5.36), we then have:

$$e_p^{n+1} = e_p^n \left(1 - (\gamma - 1) \frac{\partial u_p^{n+1}}{\partial x} \Delta t \right) - \frac{\nu_p^{n+1}}{\rho_p^n} \frac{\partial u_p^{n+1}}{\partial x} \Delta t, \tag{5.40}$$

$$p_p^{n+1} = \left[\left(p_p^n - a_p^{n-1} \right) \left(1 - \frac{\partial u_p^{n+1}}{\partial x} \Delta t \right) + a_p^{n+1} \right] \left[1 - (\gamma - 1) \frac{\partial u_p^{n+1}}{\partial x} \Delta t \right]. \tag{5.41}$$

Using Equation (5.40), Equation (5.39) is rewritten as follows:

$$p_p^{n+1} = \left[\left(p_p^n - \nu_p^{n-1}\right)\left(1 - \frac{\partial u_p^{n+1}}{\partial x}\Delta t\right) + \nu_p^{n+1}\right]\left[1 - (\gamma - 1)\frac{\partial u_p^{n+1}}{\partial x}\Delta t\right]. \tag{5.42}$$

5.3.2 Positivity, Overshoots and Stability

Since the values of density, energy, and pressure are positive, their numerical approximations should also be positive. From Equations (5.36)-(5.37), it may be seen that this occurs for the discrete density and energy equations under a Courant-like condition:

$$0 \leq \frac{\partial u_p^{n+1}}{\partial x}\Delta t \leq 1. \tag{5.43}$$

Although this ensures the values of density and energy remain positive; local extrema may be caused by the use of the velocity gradient from “old” cell when cell crossing occurs. Suppose that there are two adjacent particles in different cells whose densities satisfy the following equation:

$$\rho_p^n < \rho_{p+1}^n, \tag{5.44}$$

and whose velocity gradients satisfy the following equation:

$$\left(1 - \Delta t\frac{\partial u_p^{n+1}}{\partial x}\right) >> \left(1 - \Delta t\frac{\partial u_{p+1}^{n+1})}{\partial x}\right), \tag{5.45}$$

then it is possible that one particle will over take the other in magnitude:

$$\rho_p^{n+1} > \rho_{p+1}^{n+1}; \tag{5.46}$$

this may result in a new extremal value. A similar argument may be developed for the creation of new extrema in energy. When extrema occur in the velocity, it is necessary to apply the artificial diffusion to the calculation of the nodal velocity. An extremum occurs in the velocity at node j if the following condition is met:

$$\left(u_{j-1}^n - u_j^n\right)\left(u_{j+1}^n - u_j^n\right) > 0. \tag{5.47}$$

The new value of velocity is then calculated by the addition of an artificial viscosity-like term that approximates $\frac{h^2}{3}\frac{\partial^2 u}{\partial x^2}$ which gives:

$$u_j^n = u_j^n + \frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{3}. \tag{5.48}$$

The same approach is applied if extrema are detected in density.

5.3.3 Particle Redistribution

When the particles move among the cells, the number of particles in a cell is changed. The number of particles in a cell is used in the calculation of particle volume in Equation (5.30), and therefore the calculation of particle mass in Equation (5.31). If there were too few particles per cell and some of these particles move from one cell to another, it is possible for a cell not to have any particles. This may cause stability problems. To prevent this situation, care must be taken in the initial assignment of particles; see Section 5.4.2 and Section 5.4.3. The main idea is to ensure that there is always sufficient number of particles per cell. This may be obtained by redistributing particles or by ensuring that particles are placed where they will move into cells with fewer particles. It may also be necessary to create new particles in the empty cells with the particles' properties obtained by interpolating the particles' properties in the adjacent cells. We have not experienced the idea of creating new particles in the empty cells in our variation of MPM for gas dynamics.

5.4 Gas Dynamics Computational Experiments

5.4.1 Problem Description

The model problem used here is that of Sod [108] who used a simple gas dynamics problem to investigate finite difference schemes for shock propagation type problems. This problem has an analytical solution and may be used in the comparison against the numerical solution obtained from the variation of MPM for gas dynamics. This problem has often been used as a test problem for PIC and MPM methods; see [140]. Sod's shocktube problem consists of a shock tube, where a diaphragm is located in the middle of the tube. Two sides of the diaphragm have different pressures and densities, which make the fluids flow when the diaphragm is broken. At time $t = 0$, the diaphragm is removed and the motion of the compressible and inviscid fluid is governed by the one-dimensional system of Euler equations in (3.28)–(3.31). The initial condition at $t = 0$ of this problem is defined as:

$$(\rho, u, p)(x, 0) = \begin{cases} (1.0, \quad 0.0, \ 1.0) & \text{if} \quad x < 0.5 \\ (0.125, \ 0.0, \ 0.1) & \text{otherwise,} \end{cases} \tag{5.49}$$

on spatial domain $\Omega = [0.0,\ 1.0]$ and the diaphragm is placed at $x_0 = 0.5$. The final time for this problem is $T_e = 0.2$.

5.4.2 Initial Uniform Particle Distribution

As particles can move from one cell to another, the number of particles in a cell varies, and so does their volume according to Equation (5.30). Since we assume each material point is part of a perfect compressible gas, changing the particle's volume is a reasonable modeling assumption. In solving the Sod's shocktube problem using our variation of MPM, we initially assign the same number of particles for each cell. The result in Figure 5.3 is obtained when the initial number of particles in each cell is 8, the cell size (h) is 0.005, and the time step (Δt) is 0.00025. In this figure, each dot represents a material point and the solid line is the analytical solution. As seen in Figure 5.3, the obtained numerical result shows large errors behind the shock front. In order to reduce the error in this numerical solution, the smoothing process described in Section 5.3.2 was applied. The solution of the Sod's shocktube problem after the smoothing process was applied is shown in Figure 5.4. The error norms in the obtained numerical solution with smoothing process is about 67 to 90% of that when the smoothing process is not applied.

To investigate the relationship between the global error and the initial number of particles assigned to each cell, we examine the errors in the numerical solutions of the Sod's shocktube problem obtained from our variation of MPM with various choices of initial number of particles. In these numerical solutions, we either vary the size of mesh spacing h and keep the time step Δt fixed or vary the time step Δt and keep the mesh spacing h fixed. Figure 5.5(a) shows the change of errors in density when the size of mesh spacing h changes and the time step Δt is

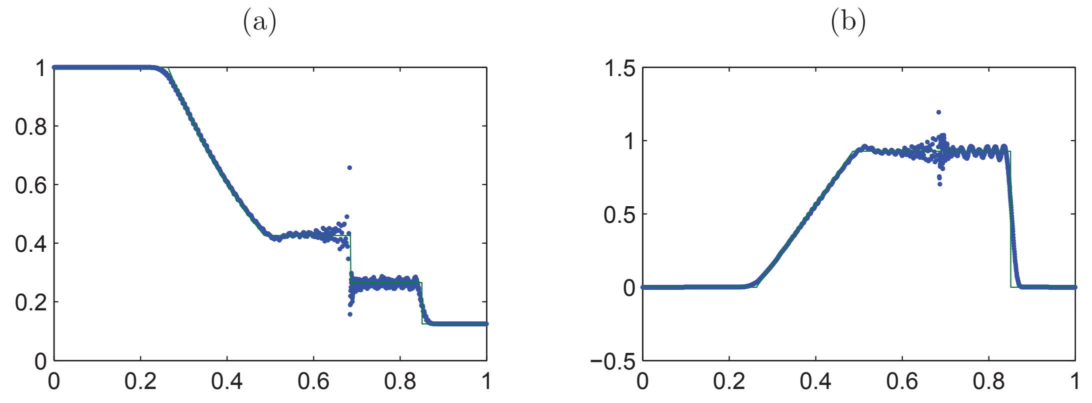


Figure 5.3. Numerical solutions for Sod's problem in Section 5.4.1 using our variation of MPM for gas dynamics at $T_e = 0.2$ with 200 cells;
(a)density: $\|\mathbf{ge}^{\rho}(T)\|_{L_1} = 6.4 \times 10^{-3}$, $\|\mathbf{ge}^{\rho}(T)\|_{L_2} = 1.52 \times 10^{-2}$
(b)velocity: $\|\mathbf{ge}^{u}(T)\|_{L_1} = 1.85 \times 10^{-2}$, $\|\mathbf{ge}^{u}(T)\|_{L_2} = 5.80 \times 10^{-2}$

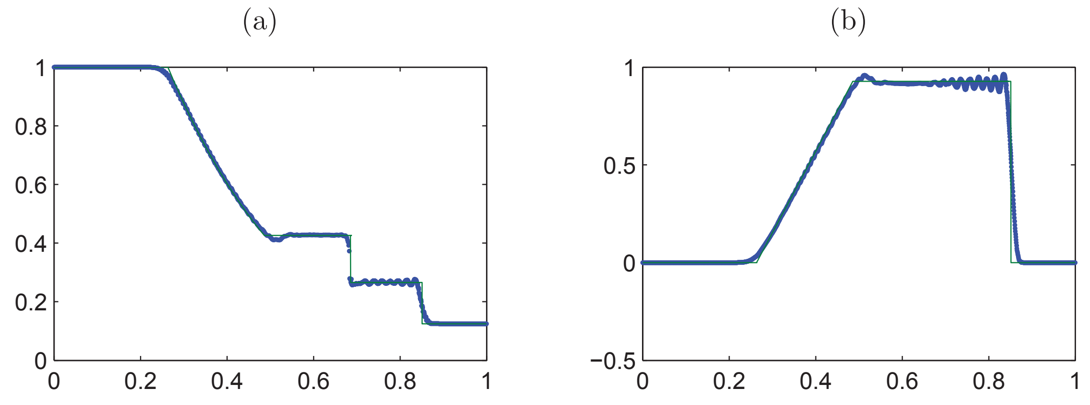
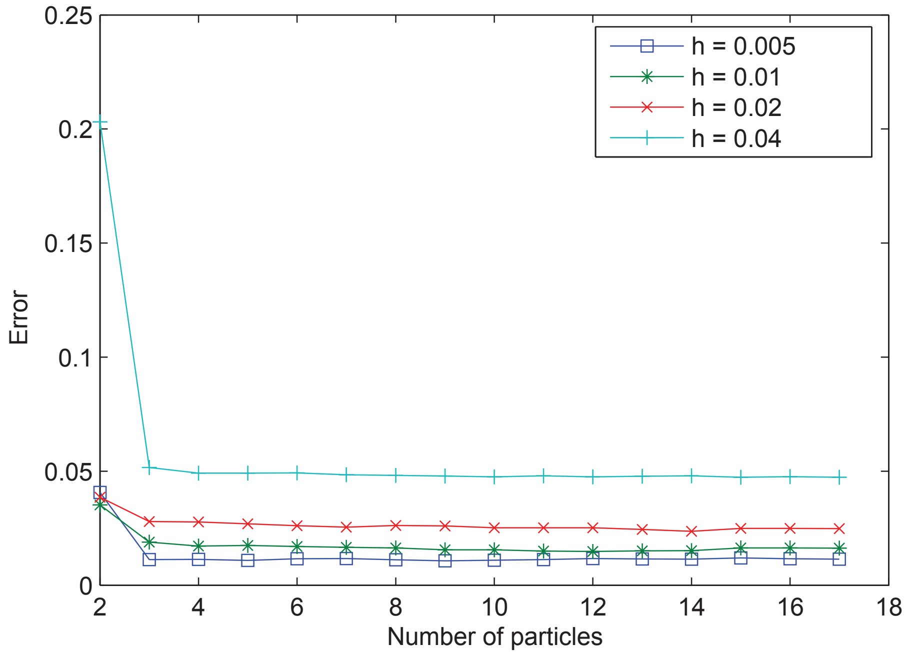


Figure 5.4. Numerical solutions for Sod's problem in Section 5.4.1 using our variation of MPM for gas dynamics at $T_e = 0.2$ with the smoothing process applied by adding viscosity-like terms described in Section 5.3.2;
(a)density: $\|\mathbf{ge}^{\rho}(T)\|_{L_1} = 4.3 \times 10^{-3}$, $\|\mathbf{ge}^{\rho}(T)\|_{L_2} = 1.05 \times 10^{-2}$
(b)velocity: $\|\mathbf{ge}^{u}(T)\|_{L_1} = 1.47 \times 10^{-2}$, $\|\mathbf{ge}^{u}(T)\|_{L_2} = 5.07 \times 10^{-2}$

fixed. As seen in Figure 5.5(a), smaller mesh spacing generates more accurate results. It is also seen in Figure 5.5(a) that the computation is inaccurate or unstable when the number of particles is too small (less than 3). For the Sod's shocktube problem, the numerical result is at best if the number of particles in a cell is between 4 to 8 and the mesh spacing is sufficiently small. It is interesting to see that the smaller mesh spacing does not reduce the need for a certain number of particles in a cell in order to obtain a stable and accurate result. In Section 5.4.3, we will show the result for the case when the initial particle distribution is not uniform. The initital particle distribution in Section 5.4.3 is based on the difference of density in various spatial regions. Figure 5.5(b) shows the change of errors in density when the time step Δt changes and the size of mesh spacing h is fixed. It also means that the results obtained in Figure 5.5(b) are from fixed mesh spacing h and varied CFL $\left(\frac{\Delta t}{h}\right)$. Figure 5.5(b) shows that the error does not change much for the numerical results obtained with the same mesh spacing but different time steps which satisfy the condition CFL < 0.1. Figure 5.5(b) also shows that there is a slight increase in error as the time step decreases; perhaps it is due to the buildup of the global error over the larger number of steps, but the global error is still dominated by the spatial error. To investigate the choice of CFL number to maintain the stability of the discussed MPM method for gas dynamics, we keep the mesh spacing and number of particles per cell fixed and vary the time step. Table 5.1 shows the allowed maximum time step to keep the method stable for three different values of mesh spacing: $h = 0.005$, $h = 0.01$, and $h = 0.015$. As seen in Table 5.1, the method is unstable if the time step is bigger than 0.00057 when the mesh spacing is 0.005. The meaning of "unstable" is that the particle velocity is so large that the particle leaves the spatial domain. Table 5.1 shows that the method generates stable results

(a) Errors for various choices of h.



(b) Errors for various choices of CFL.

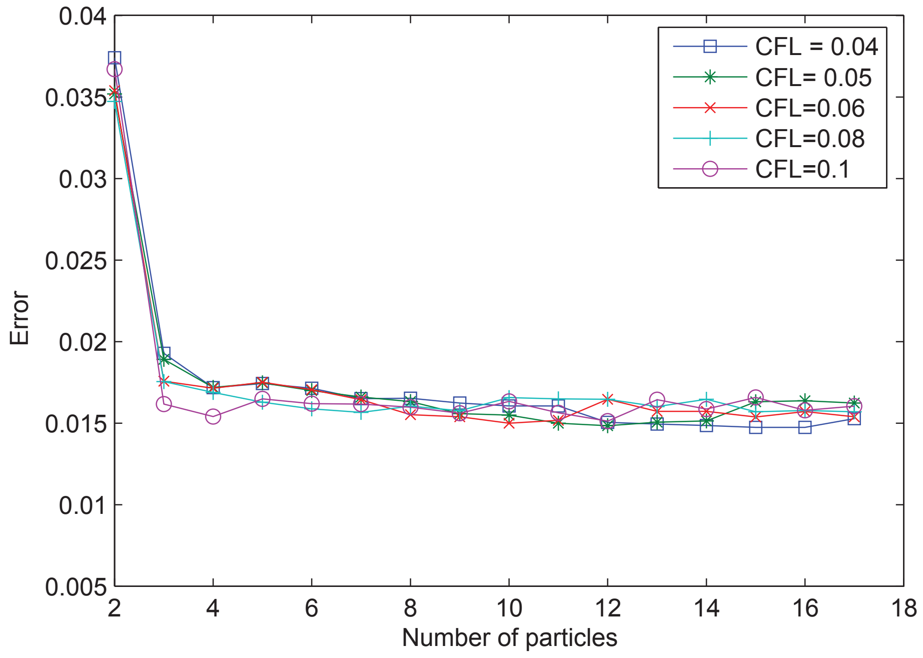


Figure 5.5. Examination of the relationship between $\|\mathbf{ge}^{\rho}(T)\|_{L_2}$ and the number of particles for our variation of MPM for gas dynamics showing errors versus the number of particles for various choices of mesh spacing (h) and CFL number.

Table 5.1. Values of Stable Time Step.

Mesh Spacing (h)	0.005		0.01		0.015	
	N.S.*	S*	N.S.	S	N.S.	S
Max stable time step (Δt)	0.00057	0.0006	0.00114	0.00124	0.00171	0.00185
Max stable CFL($\Delta t/h$)	0.114	0.124	0.114	0.124	0.114	0.123

*(N.S.: Nonsmoothing Process,   S: Smoothing Process Applied)

only if the CFL number is smaller than about 0.11 $\sim$ 0.12. If the smoothing process is applied, the maximum CFL number to maintain the stability of the method is slightly larger.

5.4.3 Alternative Particle Distribution

Although the smoothing process reduced much of instability of the particles, there are still remaining spurious oscillations in the solution. Brackbill [16] showed that the ringing instability in the PIC method was reduced with smaller number of particles per cell; see Section 5.6 below. However, the result in Section 5.4.2 shows that the use of 2 $\sim$ 3 particles increases the error and the use of one particle may generate unstable results. We would like to experiment with an alternative initial particle distribution by noting that, based on the given initial condition, the gas to the right of the diaphragm has a lower density. Hence we consider the initial distribution of particles based on the density of the gas on the computational domain in which the number of particles per cell is in proportion to the density of gas in that cell. Since the density of gas on the left side of the computational domain is 1.0 and on the right side is 0.125, eight particles are assigned to each cell on the left and one particle is assigned to each cell on the right. This nonuniform particle distribution gives a stable result as shown in Figure 5.6, although the number of particles on the right side is only one per cell. This is due to the particles on the left side of the computational domain move rightwards during the time integration process. Because there are enough particles on the left side of the computational domain and these particles move to the right, the solution process remains stable as we are constantly introducing particles into the cells on the right. Figure 5.6 shows the numerical solution of the Sod's shocktube problem obtained from our variation of MPM with fewer particles on the right hand side of the diaphram and application of the smoothing process discussed in Section 5.3.2. Comparing the result in Figure 5.6 to the result in Figure 5.4, the result in Figure 5.6 has fewer oscillations, but has a similar error norm to the previous cases.

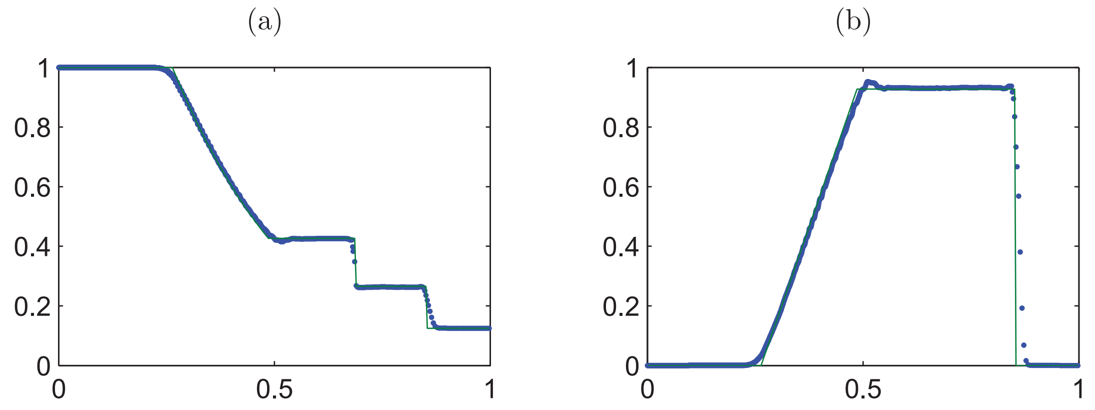


Figure 5.6. Numerical solutions for the Sod's shocktube problem in Section 5.4.1 using our variation of MPM for gas dynamics at T_e=0.2 with nonuniform initial particle distribution discussed in Section 5.4.3 and application of the smoothing process;
(a)density: $\|\mathbf{ge}^{\rho}(T)\|_{L_1} = 5.4 \times 10^{-3}$, $\|\mathbf{ge}^{\rho}(T)\|_{L_2} = 1.38 \times 10^{-2}$
(b)velocity: $\|\mathbf{ge}^{u}(T)\|_{L_1} = 1.49 \times 10^{-2}$, $\|\mathbf{ge}^{u}(T)\|_{L_2} = 5.65 \times 10^{-2}$

5.5 Time Integration Error and Grid Crossing by Particles

5.5.1 Time Integration Discontinuities Arising from Grid Crossing

The comparative lack of smoothness of the spatial basis grid functions used in the MPM translates into a lack of smoothness in time when particles cross grid points and then have properties that are redefined in terms of the basis functions of the cell into which the particles move.

Since the updated particle velocity is calculated using Equation (5.28), it means that the higher time derivatives of the particle velocity are discontinuous when a particle crosses a grid point. This may be illustrated by considering Equation (5.27) which is a forward Euler discretization of:

$$\dot{u}_p^n = \sum_j^N \bar{S}_{jp} a_j^n. \tag{5.50}$$

If the point x_p^n is in the cell domain Ω_{j-1} then Equation (5.50) may be written as:

$$\dot{u}_p^{n(-)} = \alpha_j^n a_{j-1}^n + (1-\alpha_j^n) a_j^n, \qquad \alpha_j^n = \frac{x_p^n - x_j}{x_{j-1} - x_j}, \tag{5.51}$$

whereas if the point x_p^n is in the cell domain Ω_j then Equation (5.50) may be written as:

$$\dot{u}_p^{n(+)} = \alpha_{j+1}^n a_j^n + (1-\alpha_{j+1}^n) a_{j+1}^n, \qquad \alpha_{j+1}^n = \frac{x_p^n - x_{j+1}}{x_j - x_{j+1}}. \tag{5.52}$$

The second derivative of u_p^n when the point x_p^n is in the cell domain Ω_{j-1} is given by:

$$\ddot{u}_p^{n(-)} = \alpha_j^n \dot{a}_{j-1}^n + (1-\alpha_j^n)\dot{a}_j^n + \dot{x}_p^n \frac{a_{j-1}^n - a_j^n}{x_{j-1} - x_j}, \tag{5.53}$$

or if the point x_p^n is in the cell domain Ω_j then:

$$\ddot{u}_p^{n(+)} = \alpha_{j+1}^n \dot{a}_j^n + (1-\alpha_{j+1}^n)\dot{a}_{j+1}^n + \dot{x}_p^n \frac{a_j^n - a_{j+1}^n}{x_j - x_{j+1}}. \tag{5.54}$$

The jump in the second derivative of particle velocity as the particle crosses the point x_j is given by:

$$[\ddot{u}_p^{n(+)} - \ddot{u}_p^{n(-)}] = \dot{x}_p^n \left[\frac{a_j^n - a_{j+1}^n}{x_j - x_{j+1}} - \frac{a_{j-1}^n - a_j^n}{x_{j-1} - x_j} \right]. \tag{5.55}$$

The local error at particle p associated with one step of the forward Euler method applied to Equation (5.28) is given by:

$$le_p^u(t_{n+1}; t_n, u_p^n) = \frac{\Delta t^2}{2} \ddot{u}_p^n. \tag{5.56}$$

This formula does not apply if $\ddot{u}_p$ is discontinuous with "left" and "right" values denoted by $\ddot{u}_p^{n(-)}$ and $\ddot{u}_p^{n(+)}$ respectively. One standard ODE method for crossing a discontinuity is to march up to it with one step of size Δt_1 and one step from it of size Δt_2. The local error for an Euler time step in region one may be estimated by:

$$le_p^u(t_n + \Delta t_1; t_n, u_p^n) \approx \frac{\Delta t_1^2}{2} \ddot{u}_p^{n(-)}, \tag{5.57}$$

and the local error for an Euler time step in region two is estimated by:

$$le_p^u(t_{n+1}; t_n + \Delta t_1, u_p^{t_n + \Delta t_1}) \approx \frac{\Delta t_2^2}{2} \ddot{u}_p^{n(+)}, \tag{5.58}$$

by assuming that the second derivatives may be regarded as constant on a step. It may be shown by using techniques such as those used by Shampine [105] that the error introduced over one time step that crosses the discontinuity is then the sum of the local errors of the two substeps and the difference between the solutions obtained using one big step and two substeps, i.e.,

$$le_p^u(t_{n+1}) = le_p^u(t_n + \Delta t_1; t_n, u_p^n) + le_p^u(t_{n+1}; t_n + \Delta t_1, u_p^{t_n+\Delta t_1}) + (\bar{u}_p^{n+1} - u_p^{n+1}), \tag{5.59}$$

where u_p^{n+1} is the solution computed using one Euler step of size Δt and where $\bar{u}_p^{n+1}$ is the solution computed using two Euler steps of size Δt_1 and Δt_2. The next two sub-sections will show that the gap between the two Euler solutions $(\bar{u}_p^{n+1} - u_p^{n+1})$ is one power of Δt less than the local errors for both velocity and position errors.

5.5.2 Time Integration Errors in Velocity

Having determined the nature of the discontinuity, it now remains to determine the error introduced by stepping over it. It is worth noting that with a standard PDE method, discontinuities in time derivatives do not occur in the same way as when material point method particles cross cells. In the case when a particle x_p^n lies in the cell $[x_{j-1}, x_j]$ and passes over a grid node and moves into the cell $[x_j, x_{j+1}]$ then the particle velocity is updated using the following equation:

$$u_p^{n+1} = u_p^n + \left[a_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}}(a_j^n - a_{j-1}^n)\right]\Delta t. \tag{5.60}$$

Alternatively, the forward Euler method may be applied to march up to the grid node at x_j in one step and then take another step to bring the particle to the spatial position x_p^n. For the first substep of length Δt_1 the particle velocity is given by:

$$\bar{u}_p^n = u_p^n + \left[a_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}}(a_j^n - a_{j-1}^n)\right]\Delta t_1. \tag{5.61}$$

For the second substep of length Δt_2, the particle velocity is as follows:

$$\bar{u}_p^{n+1} = \bar{u}_p^n + \left[a_j^n + \Delta t_1 \dot{a}_j^n\right]\Delta t_2, \tag{5.62}$$

where $\Delta t = \Delta t_1 + \Delta t_2$. Hence the difference in the velocities calculated using the two approaches is given by:

$$\bar{u}_p^{n+1} - u_p^{n+1} = (a_j^n - a_{j-1}^n)\left[\frac{x_j - x_p^n}{x_j - x_{j-1}}\right]\Delta t_2 + \Delta t_1 \Delta t_2 \dot{a}_j^n, \tag{5.63}$$

and so may be written as:

$$\bar{u}_p^{n+1} - u_p^{n+1} \approx C\Delta t_2(a_j^n - a_{j-1}^n) + h.o.t, \tag{5.64}$$

where $C = \left[\dfrac{x_j - x_p^n}{x_j - x_{j-1}}\right]$ and where $0 \leq C \leq 1$. For the Euler equations considered here the values of $(a_j^n - a_{j-1}^n)$ may be as large as 10^3. This dictates the use of a time step of the order of that used in Section 5.4. [1]

5.5.3 Time Integration Errors in Spatial Position

We are now determine the error introduced in spatial position when the particle x_p lies in the cell domain Ω_{j-1} and crosses the grid node at x_j and moves into the cell Ω_j using the same approach of time integration errors in velocity. In the variation of MPM discussed above, the particle position is updated as follows:

$$x_p^{n+1} = x_p^n + \left[u_{j-1}^{n+1} + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}}(u_j^{n+1} - u_{j-1}^{n+1})\right]\Delta t\ , \tag{5.65}$$

which may be written as:

$$x_p^{n+1} = x_p^n + \left[u_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}}(u_j^n - u_{j-1}^n)\right]\Delta t + \left[a_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}}(a_j^n - a_{j-1}^n)\right]\Delta t^2. \tag{5.66}$$

As stated above, consider using the forward Euler method to march up to the edge of the cell in one step and then in another step to step to the same point in time. For the first step, the particle position is calculated as follows:

$$\bar{x}_p^n = x_p^n + \left[u_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}}(u_j^n - u_{j-1}^n)\right]\Delta t_1 + \left[a_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}}(a_j^n - a_{j-1}^n)\right]\Delta t_1^2. \tag{5.67}$$

For the second step, the particle position is given by:

$$\bar{x}_p^{n+1} = \bar{x}_p^n + u_j^{n+1}\Delta t_2\ , \tag{5.68}$$

and so:

$$\bar{x}_p^{n+1} = \bar{x}_p^n + u_j^n\Delta t_2 + a_j^n\Delta t\ \Delta t_2\ , \tag{5.69}$$

[1]The reader should note that throughout C will be used as a generic constant whose value may be different each time it is used.

where $\Delta t = \Delta t_1 + \Delta t_2$. Hence the difference between the positions calculated by the two approaches is:

$$\bar{x}_p^{n+1} - x_p^{n+1} = (u_j^{n+1} - u_{j-1}^{n+1}) \left[\frac{x_j - x_p^n}{x_j - x_{j-1}} \right] \Delta t_2 - \left[a_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{i-1}} (a_j^n - a_{j-1}^n) \right] \Delta t_1 \, \Delta t_2 \quad (5.70)$$

Figure 5.7 illustrates the different values of spatial position that may result when the discontinuity is and is not considered.

Dividing both sides of Equation (5.70) by $(x_j - x_{j-1})$ gives:

$$\frac{\bar{x}_p^{n+1} - x_p^{n+1}}{x_j - x_{j-1}} \approx \Delta t_2 \frac{(u_j^{n+1} - u_{j-1}^{n+1})}{(x_j - x_{j-1})} C - \frac{\Delta t_1 \Delta t_2}{x_j - x_{j-1}} \left[a_{j-1}^n + \frac{x_p^n - x_{j-1}}{x_j - x_{j-1}} (a_j^n - a_{j-1}^n) \right] , \quad (5.71)$$

where $C = \left[\frac{x_j - x_p^n}{x_j - x_{j-1}} \right]$ and where $0 \leq C \leq 1$.

5.6 Spatial Error Estimation

There are several sources of error that contribute to the overall spatial error at grid nodes in the variant of MPM for gas dynamics. These error sources includes the error from mapping the particle values onto grid nodes, the error from crossing particles cells boundaries, and the error from projecting the material movement at grid nodes onto particles. In this section, we consider the estimation of error from mapping the values of particles onto the value of a grid node. In particular, we evaluate the mass mapping error introduced by Equation (5.20), the momentum mapping error introduced by Equation (5.21), and the force mapping error introduced by Equation (5.24). In order to distinguish between the mapping spatial error and the overall spatial error, we use **ep** for mapping spatial error instead of using **es** for the overall spatial error. The vector of mapping spatial errors of grid nodes is denoted as:

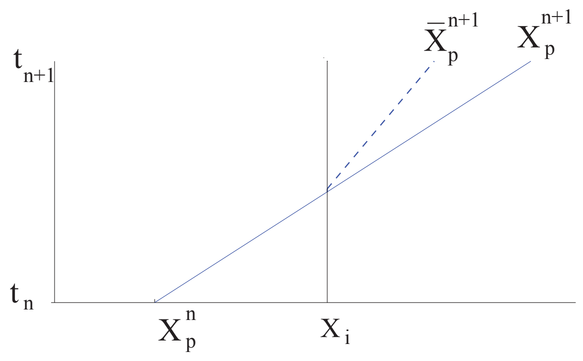


Figure 5.7. Mesh Crossing Diagram.

$$\mathbf{ep}^q(t) = \left[ep_1^q(t), ep_2^q(t), ep_3^q(t), ..., ep_N^q(t)\right]^T , \tag{5.72}$$

where q is the quantity that error is being measured and N is the number of grid nodes. In the following analysis, we consider $q = m$ (for mass), P (for momentum), F (for force), u (for velocity), a (for acceleration), and ∇u (for velocity gradient). Before estimating errors in these quantities, it is helpful to establish some notation relating to an important result of quadrature error bound result by Hickernell [57].

5.6.1 Hickernell's Quadrature Error Bound

Theorem 2.3 of Hickernell [57] on quadrature error bound proves that for any function $f(x) \in X^2 \equiv \left\{f : \dfrac{df}{dx} \in L^2([0,1])\right\}$ and some random or deterministic sample Q of N_p points in $[0,1]$, then the following inequality holds:

$$\left| \int_0^1 f(y)dy - \frac{1}{N_p}\sum_{p=1}^{N_p} f(z_p) \right| \le D_2(Q) \left\| \frac{df}{dx} \right\|_2 , \tag{5.73}$$

where:

$$D_2(Q) = \sqrt{\frac{1}{12N_p^2} + \frac{1}{N_p}\sum_{p=1}^{N_p}\left(z_p - \frac{2p-1}{2N_p}\right)^2}, \tag{5.74}$$

and:

$$\left\| \frac{df}{dx} \right\|_2 = \left[\int_0^1 \left(\frac{df}{dx}\right)^2 dx\right]^{1/2} , \tag{5.75}$$

and $\{z_p\}$ is an ordered set of the points of sample Q. Although Hickernell proves the result for more general norms, the above result is sufficient for this analysis.

It is important to map Hickernell's result to a cell domain used in the discussed variant of MPM. Let $\{z_p \;:\; p = 1, ..., N_{j+1}\}$ be an ordered set of N_{j+1} points in $[0,1]$ of sample Q. When mapping this set of points into the cell domain Ω_j, we obtain the set of points as given by $\{hz_p + x_j \;:\; p = 1, ..., N_{j+1}\}$. The set of N_{j+1} equidistant points in the cell domain Ω_j is $\left\{x_j + \dfrac{(2p-1)h}{2N_{j+1}} \;:\; p = 1, ..., N_{j+1}\right\}$. A similar inequality to the inequality in Equation (5.73) when we integrate the function f over the cell length h is given as follows:

$$\left| \frac{1}{h}\int_{x_j}^{x_{j+1}} f(y)dy - \frac{1}{N_{j+1}}\sum_{p=1}^{N_{j+1}} f(hz_p + x_j) \right| \le D_2(Q)\, h^{\frac{1}{2}} \left\| \frac{df}{dx} \right\|_{2,\Omega_j} , \tag{5.76}$$

where:

$$\left\| \frac{df}{dx} \right\|_{2,\Omega_j} = \left[\int_{\Omega_j} \left(\frac{df}{dx} \right)^2 dx \right]^{1/2} , \tag{5.77}$$

and:

$$D_2(Q) \;= \sqrt{\frac{1}{12N_{j+1}^2} \;+ \frac{1}{N_{j+1}h^2} \sum_{p=1}^{N_{j+1}} \left((hz_p + x_j) - \left(x_j + \frac{(2p-1)h}{2N_p^{j+1}} \right) \right)^2} \; . \tag{5.78}$$

It should also be noted that from the mean value theorem for integration, we then have:

$$h^{\frac{1}{2}} \left[\int_{\Omega_j} \left(\frac{df}{dx} \right)^2 dx \right]^{\frac{1}{2}} = h \left| \frac{df}{dx}(\xi) \right| , \tag{5.79}$$

for some $\xi \in \Omega_j$. Hence:

$$\left| \int_{x_j}^{x_{j+1}} f(y)dy - \frac{h}{N_{j+1}} \sum_{p=1}^{N_{j+1}} f(hz_p + x_j) \right| \leq D_2(Q)\; h^2 \left| \frac{df}{dx}(\xi) \right| \; . \tag{5.80}$$

The values of $D_2(Q)$ clearly depend on the point distribution of sample Q and thus in turn on the problem being solved. Considering the worst case of particles' negligible distances apart at the end of an interval, it is straightforward to show that:

$$\frac{1}{2\sqrt{3}N_{j+1}} \leq D_2(Q) \leq \frac{1}{\sqrt{3}} \; . \tag{5.81}$$

This result has a similar form to the results of Vshivkov [135] (as quoted by Brackbill, [17]) except that the key difference here lies in the choice of quadrature rule. Vshivkov calculates the error, δ_k, in the charge density at node k as computed with the PIC. His result states that:

$$\delta_k \leq \left(\frac{3\rho_{\mathrm{av}}^2}{2\rho_{\mathrm{min}}} + h \frac{\rho_{\mathrm{av}}^2 \rho_{\mathrm{max}}}{6\rho_{\mathrm{min}}^3} \left| \frac{\partial \rho}{\partial x} \right|_{\mathrm{max}} \right) \frac{1}{N^2} + \frac{h^2}{12} \left| \frac{\partial^2 \rho}{\partial x^2} \right|_{\mathrm{max}} , \tag{5.82}$$

where N is the average number of particles in a cell.

5.6.2 Ringing Instability

It is also important to remark that, as with any quadrature rule, there exist values of x_j such that $f(x_j) = 0$. For example if:

$$f(x) = \prod_{j=1}^{N_p^i} (x - x_j) \ , \tag{5.83}$$

then the integral approximation is zero and the error is the value of the integral. Furthermore there are functions that are nonzero at the particle points such as:

$$f(x_i) = (-1)^i \ , \tag{5.84}$$

which in the case of even numbers of mesh points will give a zero contribution to the integral. The problem is made worse by the fact that the quadrature rule is essentially using a piecewise constant approximation to function in forming the integral in the most general case. This loss of information due to quadrature is known as the "Ringing Instability" and is a well-known feature of particle methods that is attributed to the under-representation of particle data on the grid. Brackbill [16] and MacNeice [85] explain this instability in terms of Fourier analysis.

5.6.3 Mass Projection Error

The mass projection error at node j at time t_n is denoted as $ep_j^m(t_n)$ and is defined by:

$$ep_j^m(t_n) = \int_\Omega \rho(x, t_n) S_j(x) dx - m_j^n, \tag{5.85}$$

where m_j^n is nodal mass defined in Equation (5.20). From Equations (5.20), (5.11), and (5.31), the nodal mass is given by:

$$m_j^n = \sum_{p=1}^{N_p} S_j(x_p) \rho_p^n V_p^n. \tag{5.86}$$

With the piecewise-linear grid basis function for node j defined in Equation (5.8), the contribution to the nodal mass at node j is from the particles in cell Ω_{j-1} and Ω_j. With the definition of particle volume in (5.30), Equation (5.85) is then rewritten as:

$$ep_j^m(t_n) = \int_{\Omega_{j-1}} \rho(x,t_n)S_j(x)dx - \sum_{p:x_p^n \in \Omega_{j-1}} S_j(x_p)\rho_p^n \frac{h}{N_{j-1}^n}$$
$$+ \int_{\Omega_j} \rho(x,t_n)S_j(x)dx - \sum_{p:x_p^n \in \Omega_j} S_j(x_p)\rho_p^n \frac{h}{N_j^n}. \tag{5.87}$$

The mass projection error in (5.87) is thus composed of two terms each of which is similar to the right side of Equation (5.80). Using the result in Equation (5.80), we obtain the following bound of mass projection error:

$$\left|ep_j^m(t_n)\right| \leq D_2(Q_{j-1})h^2 \left|\frac{d(\rho(x,t_n)S_j(x))}{dx}(\xi_{j-1})\right| + D_2(Q_j)h^2 \left|\frac{d(\rho(x,t_n)S_j(x))}{dx}(\xi_j)\right| \ , \tag{5.88}$$

for some $\xi_{j-1} \in \Omega_{j-1}$ and $\xi_j \in \Omega_j$. In Equation (5.88), sample Q_j is a set of N_j^n points in cell Ω_j which is the ordered set of particle positions, $\{x_p^n : p = 1,..,N_p \text{ and } x_p^n \in \Omega_j\}$, in this cell. As the basis grid function, $S_j(x)$, is defined using (5.8), the first-order spatial derivative of $S_j(x)$ depends on $\frac{1}{h}$. This results in the mass projection error $ep_j^m(t_n)$ being first-order in h. In order to approximate the mass projection error in Equation (5.87), we use the trapezoidal quadrature rule to approximate the integrals in this equation.

The result in Figure 5.8 shows how the mass projection error in L_1-norm, $\|\mathbf{ep}^m(t_n)\|_{L_1}$, grows for different mesh sizes and is first-order of mesh size as expected. The errors grow in time in a way that is consistent with first time integration using the forward Euler method.

5.6.4 Momentum Projection Error

The momentum projection error at node j at t_n associated with Equation (5.35) is denoted as $ep_j^P(t_n)$. Using a similar derivation to the derivation of the mass projection error in Section 5.6.3, the bound of momentum projection error, $ep_j^P(t_n)$, is given by:

$$\left|ep_j^P(t_n)\right| \leq D_2(Q_{j-1})h^2 \left|\frac{d(\rho(x,t_n)u(x,t_n)S_j(x))}{dx}(\xi_{j-1})\right|$$
$$+D_2(Q_j)h^2 \left|\frac{d(\rho(x,t_n)u(x,t_n)S_j(x))}{dx}(\xi_j)\right|, \tag{5.89}$$

for some $\xi_{j-1} \in \Omega_{j-1}$ and $\xi_j \in \Omega_j$. In Equation (5.89), sample Q_j is a set of N_j^n points in cell Ω_j which is the ordered set of particle positions, $\{x_p^n : p = 1,..,N_p \text{ and } x_p^n \in \Omega_j\}$, in this cell. As the first-order spatial derivative of $S_j(x)$ depends on $\frac{1}{h}$, it follows that the momentum projection error is also first-order in h.

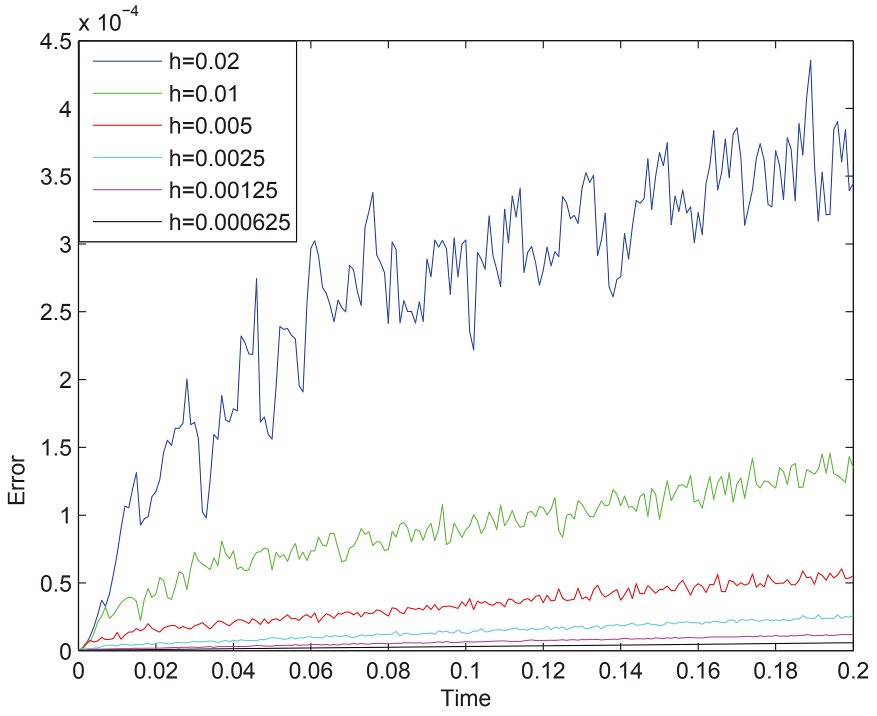


Figure 5.8. Mass projection error in L_1-norm, $\|\mathbf{ep}^m(t_n)\|_{L_1}$, for different mesh spacings, h.

5.6.5 Velocity Projection Error

Consider the exact nodal projected velocity, $u_j(t_n)$, which is the ratio of the exact nodal projected momentum and the exact nodal projected mass as given by:

$$u_j(t_n) = \frac{\int_\Omega \rho(x,t_n)u(x,t_n)S_j(x)dx}{\int_\Omega \rho(x,t_n)S_j(x)dx}. \tag{5.90}$$

The error in the velocity projection, $ep_j^{\bar{u}}(t_n)$, is defined by:

$$ep_j^{\bar{u}}(t_n) = u_j(t_n) - u_j^n, \tag{5.91}$$

where u_j^n is given by Equation (5.28). Let $u(x_j,t_n)$ be the exact nodal velocity at node j at t_n. Define the error from projection in the exact value as:

$$ep_j^{\bar{\bar{u}}}(t_n) = u(x_j,t_n) - u_j(t_n). \tag{5.92}$$

Then overall error in velocity projection may be split into two parts:

$$ep_j^u(t_n) = u(x_j,t_n) - u_j^n = ep_j^{\bar{\bar{u}}}(t_n) + ep_j^{\bar{u}}(t_n). \tag{5.93}$$

Let:

$$\frac{\delta^2 U}{\delta x^2}(x,t) = \rho(x,t)u(x,t) \ , \tag{5.94}$$

and:

$$\frac{\delta^2 V}{\delta x^2}(x,t) = \rho(x,t). \tag{5.95}$$

Then the exact velocity is defined by:

$$u(x,t) = \frac{h\frac{\delta^2 U}{\delta x^2}(x,t)}{h\frac{\delta^2 V}{\delta x^2}(x,t)}. \tag{5.96}$$

Using integration by parts, the projection of the velocity is then given by:

$$\begin{aligned} u_j(t_n) &= \frac{\int_\Omega S_j(x)\rho(x,t_n)u(x,t_n)dx}{\int_\Omega S_j(x)\rho(x,t_n)dx} \\ &= \frac{\frac{1}{h}(U(x_j-h,t_n)-2U(x_j,t_n)+U(x_j+h,t_n))}{\frac{1}{h}(V(x_j-h,t_n)-2V(x_j,t_n)+V(x_j+h,t_n))}. \end{aligned}$$

Define two projection errors $ep_j^U(t_n)$ and $ep_j^V(t_n)$ by:

$$ep_j^U(t_n) = h\frac{\delta^2 U}{\delta x^2}(x_j,t_n) - \int_\Omega S_j(x)\rho(x,t_n)u(x,t_n)dx, \tag{5.97}$$

and:

$$ep_j^V(t_n) = h\frac{\delta^2 V}{\delta x^2}(x_j,t_n) - \int_\Omega S_j(x)\rho(x,t_n)dx. \tag{5.98}$$

Using standard finite difference analysis, we have $ep_j^U = O(h^3) + h.o.t$, and and $ep_j^V = O(h^3) + h.o.t.$ From Equations (5.92), (5.90), and (5.96), we have:

$$\begin{aligned} ep_j^{\bar{\bar{u}}}(t_n) &= \frac{h\frac{\delta^2 U}{\delta x^2}(x,t_n)}{h\frac{\delta^2 V}{\delta x^2}(x,t_n)} - \frac{\int_\Omega S_j(x)\rho(x,t_n)u(x,t_n)dx}{\int_\Omega S_j(x)\rho(x,t_n)dx} \\ &= \frac{1}{\int_\Omega S_j(x)\rho(x,t_n)dx}(ep_j^U - u(x_j,t_n)ep_j^V). \end{aligned}$$

As ep_j^U and ep_j^V are third-order in h and $\int_\Omega S_j(x)\rho(x,t_n)dx$ is first-order in h, it follows that $ep_j^{\bar{\bar{u}}}(t_n)$ is second-order in h.

From Equations (5.91), (5.90), and (5.28), we have:

$$\begin{aligned} ep_j^{\bar{u}}(t_n) &= \frac{\int_\Omega S_j(x)\rho(x,t_n)u(x,t_n)dx}{\int_\Omega S_j(x)\rho(x,t_n)dx} - \frac{P_j^n}{m_j^n} \\ &= \frac{1}{m_j^n}\left(ep_j^P(t_n) - u(x_j,t_n)ep_j^m(t_n)\right), \end{aligned}$$

where:

$$\begin{aligned} ep_j^P(t_n) = \int_{\Omega_{j-1}} S_j(x)\rho(x,t_n)u(x,t_n)dx - \frac{h}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}} S_j(x_p^n)\rho_p^n u_p^n \\ + \int_{\Omega_j} S_j(x)\rho(x,t_n)u(x,t_n)dx - \frac{h}{N_j^n}\sum_{p:x_p^n\in\Omega_j} S_j(x_p^n)\rho_p^n u_p^n. \end{aligned} \tag{5.99}$$

Using a Taylors series expansion of velocity about x_j gives:

$$ep_j^P(t_n) = u(x_j,t_n)ep_j^m(t_n) + u_x(x_j,t_n)ep_j^{up1}(t_n) + \frac{u_{xx}(x_j,t_n)}{2}ep_j^{up2}(t_n) + ...+ \tag{5.100}$$

where:

$$\begin{aligned} ep_j^{upk}(t_n) = \int_{\Omega_{j-1}} S_j(x)\rho(x,t_n)(x-x_j)^k dx - \frac{h}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}} S_j(x_p^n)\rho_p^n(x_p^n-x_j)^k \\ + \int_{\Omega_j} S_j(x)\rho(x,t_n)(x-x_j)^k dx - \frac{h}{N_j^n}\sum_{p:x_p^n\in\Omega_j} S_j(x_p^n)\rho_p^n(x_p^n-x_j)^k. \end{aligned} \tag{5.101}$$

Therefore:

$$ep_j^u(t_n) = ep_j^{\bar{\bar{u}}}(t_n) + \frac{1}{m_j^n}\left(u_x(x_j,t_n)ep_j^{up1}(t_n) + \frac{u_{xx}(x_j,t_n)}{2}ep_j^{up2}(t_n) + ...\right). \tag{5.102}$$

Using Hickernell's result from Equation (5.76) to Equation (5.101) gives:

$$\begin{aligned} \left|ep_j^{upk}(t_n)\right| \le D_2(Q_{j-1})h^2\left|\frac{d(S_j(x)\rho(x,t_n)(x-x_j)^k)}{dx}(\xi_{j-1})\right| \\ +D_2(Q_j)h^2\left|\frac{d(S_j(x)\rho(x,t_n)(x-x_j)^k)}{dx}(\xi_j)\right|, \end{aligned} \tag{5.103}$$

for some $\xi_{j-1}\in\Omega_{j-1}$ and some $\xi_j\in\Omega_j$. For the lowest order term $k=1$ this is second-order.

5.6.6 Acceleration Projection Error

We define the projection error in acceleration, $ep_j^a(t_n)$, as:

$$ep_j^a(t_n) = a(x_j, t_n) - a_j^n, \tag{5.104}$$

where $a(x_j, t_n)$ is the exact acceleration at node x_j at time t_n. Same as for the velocity projection error, the acceleration projection error may be split into two parts:

$$ep_j^a(t_n) = (a(x_j, t_n) - a_j(t_n)) + \left(a_j(t_n) - a_j^n\right) = ep_j^{\bar{\bar{a}}}(t_n) + ep_j^{\bar{a}}(t_n)\ , \tag{5.105}$$

where a_j^n is calculated nodal acceleration from (5.29) and $a_j(t_n)$ is exact nodal acceleration obtained by projecting the exact pressure and density onto the mesh points as given by:

$$a_j(t_n) = \frac{\frac{1}{h}(\int_{\Omega_{j-1}} p(x,t_n)dx - \int_{\Omega_j} p(x,t_n)dx)}{\int_\Omega \rho(x,t_n)S_j(x)dx}. \tag{5.106}$$

The error $ep_j^{\bar{\bar{a}}}(t_n)$ may be shown to be second-order in h using the same approach as in Equation (5.90)–(5.101). In the other hand, we have:

$$ep_j^{\bar{a}}(t_n) = \frac{\frac{1}{h}(\int_{\Omega_{j-1}} p(x,t_n)dx - \int_{\Omega_j} p(x,t_n)dx)}{\int_\Omega \rho(x,t_n)S_j(x)dx} - \frac{1}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}} p_p^n - \frac{1}{N_j^n}\sum_{p:x_p^n\in\Omega_j} p_p^n\ . \tag{5.107}$$

Then:

$$ep_j^{\bar{a}}(t_n) = \frac{1}{m_j^n}(ep_j^F(t_n) - a(x_j,t_n)ep_j^m(t_n), \tag{5.108}$$

where:

$$\begin{aligned} ep_j^F(t_n) = &\left(\frac{1}{h}\int_{\Omega_{j-1}} p(x,t_n)dx - \frac{1}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}} p_p^n\right) \\ &+ \left(-\frac{1}{h}\int_{\Omega_j} p(x,t_n)dx + \frac{1}{N_j^n}\sum_{p:x_p^n\in\Omega_j} p_p^n\right). \end{aligned} \tag{5.109}$$

Expanding the values of pressure about x_j gives:

$$\frac{1}{h}\int_{\Omega_{j-1}} p(x,t_n)dx - \frac{1}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}} p_p^n = p_x(x_j^n)\left(\frac{x_j+x_{j-1}}{2} - \frac{1}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}} x_p^n\right)$$
$$+\frac{p_{xx}(x_j^n)}{2}\left(\int_{\Omega_{j-1}}\frac{(x-x_j)^2}{h}dx - \frac{h}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}}\frac{(x_p^n-x_j)^2}{h}\right),$$

and similarly for the interval Ω_j. The lowest order term in the error is then:

$$ep_j^F = p_x(x_j)\left(h - \frac{1}{N_{j-1}^n}\sum_{p:x_p^n\in\Omega_{j-1}} x_p^n + \frac{1}{N_j^n}\sum_{p:x_p^n\in\Omega_j} x_p^n\right) + h.o.t. \tag{5.110}$$

In order to investigate the order of this term, it is necessary to consider the evolution of the points that contribute to the calculation of acceleration at the point x_j at time t_n. Let means of particle positions and velocities be defined by:

$$\bar{x}_{j+1}^n(t) = \frac{1}{N_j^n}\sum_{p:x_p^n\in\Omega_j} x_p(t), \tag{5.111}$$

$$\bar{u}_{j+1}^n = \frac{1}{N_{j+1}^n}\sum_{p:x_p^n\in\Omega_j} u_p(t). \tag{5.112}$$

Furthermore define:

$$\frac{d\bar{u}_j^n}{dx}(t) = \frac{\bar{u}_{j+1}^n(t)-\bar{u}_j^n(t)}{\bar{x}_{j+1}^n(t)-\bar{x}_j^n(t)}. \tag{5.113}$$

From Equations (5.111),(5.112) and (5.113) it follows that:

$$\bar{x}_{j+1}^n(t_{n+1}) - \bar{x}_j^n(t_{n+1}) = \left[1+\Delta t\frac{d\bar{u}_j^n}{dx}(t_n)\right](\bar{x}_{j+1}^n(t_n)-\bar{x}_j^n(t_n)), \tag{5.114}$$

and hence that the gap between the means may be related back to the initial mesh distribution as given by:

$$\bar{x}_{j+1}^n(t_{n+1}) - \bar{x}_j^n(t_{n+1}) = \prod_k\left[1+\Delta t\frac{d\bar{u}_j^n}{dx}(t_k)\right](\bar{x}_{j+1}^n(t_0)-\bar{x}_j^n(t_0)). \tag{5.115}$$

Suppose that initially all the points are evenly distributed at time t_0 with spacing h_p, then:

$$(\bar{x}_{j+1}^n(t_0)-\bar{x}_j^n(t_0)) = h_p(N_{j+1}^n+N_j^n)/2, \tag{5.116}$$

where the interval spacing h is connected to the initial particle spacing h_p through:

$$h = h_p(N^0 + 1) \tag{5.117}$$

where N^0 is the total number of points in every interval at t_0. Hence:

$$\bar{x}^n_{j+1}(t_{n+1}) - \bar{x}^n_j(t_{n+1}) = h \prod_k \left[1 + \Delta t \frac{d\bar{u}^n_j}{dx}(t_k)\right] \left[\frac{N^n_{j+1} + N^n_j}{2(N^0 + 1)}\right]. \tag{5.118}$$

Using the CFL condition as defined by $\frac{\Delta t}{dx}$ then gives:

$$\bar{x}^n_{j+1}(t_{n+1}) - \bar{x}^n_j(t_{n+1}) = h\left[1 + h\ CFL\ K\right] \frac{N^n_{j+1} + N^n_j}{2(N^0 + 1)} + h.o.t\ , \tag{5.119}$$

where:

$$K = \sum_k \left[\frac{d\bar{u}^n_j}{dx}(t_k)\right]. \tag{5.120}$$

This result shows that the acceleration order may be first-order if local velocity gradients are "small" if particles are rezoned as to be closer to evenly spaced as in Section 5.3.3.

5.6.7 Velocity Gradient Error

The accuracy of the equations used to update energy and density in Section 5.3.1 depends on the accuracy of the velocity gradient, and the velocity gradient at any particle $x^n_p \in \Omega_j$ is defined as:

$$\frac{\delta u}{\delta x}(x^n_p) = \frac{u^n_{j+1} - u^n_j}{x_{j+1} - x_j} - \left(\frac{x_{j+1} + x_j}{2} - x^n_p\right) \frac{\delta^2 u}{\delta x^2}(x^n_p) + h.o.t\ . \tag{5.121}$$

The velocity gradient error at particles is rewritten as:

$$ep^{\nabla u}_j = \frac{ep^u_{j+1} - ep^u_j}{h} + \frac{\Delta t}{h}\left[ep^a_{j+1} - ep^a_j\right] - \left(\frac{x_{i+1} + x_i}{2} - x^n_p\right) \frac{\delta^2 u}{\delta x^2}(x^n_p)\ . \tag{5.122}$$

Thus the velocity gradient error depends on the first-order interpolation error.

5.7 Combining the Error Estimate Results

The density errors, $\mathbf{ge}^\rho(T)$, at $T_e = 0.2$ in L_1-Norm, L_2-Norm, and L_∞-Norm for the Sod's shocktube problem discussed in Section 5.4.1 with different sizes of mesh spacing are shown in Table 5.2. These results are obtained for the case $CFL = 0.1$ and the initial number of particles per cell is 8. The numbers in Table 5.2 indicate that the density error is order of h in L_1-Norm, order of $h^{\frac{1}{2}}$ in L_2-Norm, and order of h^0 in L_∞-Norm. In order to understand the orders of these norms, a detailed inspection of the order of accuracy in each part of the spatial domain was made. In the regions around the contact discontinuity and the shock, the maximum pointwise error does not decrease as the size of mesh spacing decreases. The error in L_∞-Norm is therefore order of h^0. The density error in L_1-Norm, $\|\mathbf{ge}^\rho(T)\|_{L_1}$, is proportional to $h\|\mathbf{ge}^\rho(T)\|_{L_\infty}$ while the approximate L_2-Norm is $\sqrt{h}\|\mathbf{ge}^\rho(T)\|_{L_\infty}$, thus giving rise to the observed orders of convergence. Figure 5.9 shows the evolution in time of the L_1-Norm of the density error, $\|\mathbf{ge}^\rho(t)\|_{L_1}$, for different mesh sizes. This error is first-order in mesh spacing h.

Table 5.2. The density errors at $T = 0.2$ in L_1-Norm, $\|\mathbf{ge}^\rho(T)\|_{L_1}$, L_2-Norm, $\|\mathbf{ge}^\rho(T)\|_{L_2}$, and L_∞-Norm, $\|\mathbf{ge}^\rho(T)\|_{L_\infty}$for the Sod's shocktube problem discussed in Section 5.4.1.

h	$\|\mathbf{ge}^\rho(T)\|_{L_1}$	$\|\mathbf{ge}^\rho(T)\|_{L_2}$	$\|\mathbf{ge}^\rho(T)\|_{L_\infty}$
0.02	0.00161	0.02484	0.1051
0.01	0.00831	0.01587	0.0812
0.005	0.00434	0.01046	0.1139
0.0025	0.00231	0.00759	0.1063
0.00125	0.00136	0.00626	0.1002

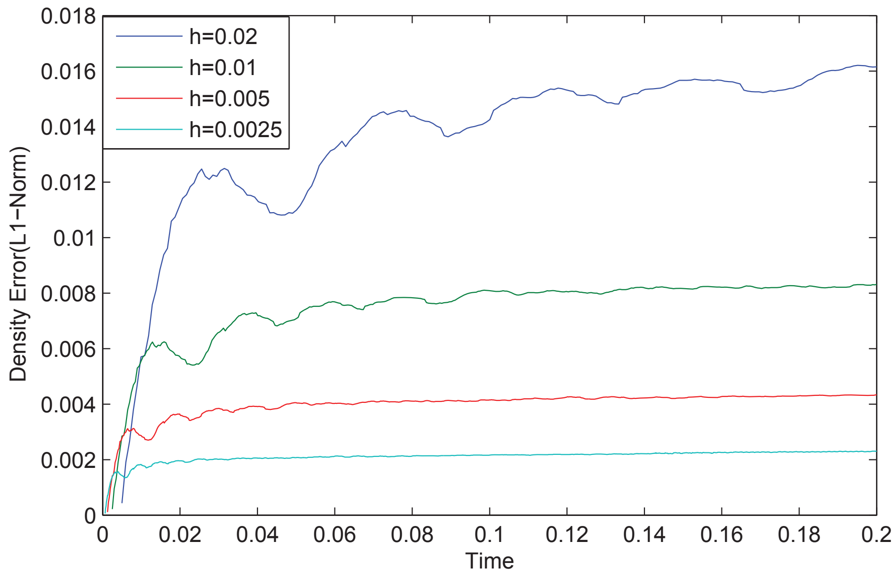


Figure 5.9. L_1-Norm of density global error, $\|\mathbf{ge}^\rho(T_e)\|_{L_1}$, in time for different mesh sizes.

5.8 Summary

In this chapter, we have presented a numerical analysis of our variation of MPM proposed for gas dynamics. In this study, we consider the global errors resulting from different particle distributions, the errors in time integration, and the mapping errors in spatial discretization. The analyses of these errors are obtained numerically and experimentally on the well-known Sod's shocktube test problem in one-dimensional space. Analysis shows that the accuracy of the method depends on a sufficiently well-behaved point distribution. For time integration errors, we consider time integration error in velocity and spatial position when particles cross cells. These integration errors are first-order in time. For mapping errors in spatial discretization, we consider the estimation of errors from mapping the values of particles onto the values of grid nodes; these mapping errors include the errors in mapping mass, momentum, velocity, acceleration, and velocity gradient. The importance of this analysis is that it provides a way to make a more formal assessment of many of the errors in MPM type methods.

CHAPTER 6

THE IMPROVED PRODUCTION IMPLICIT CONTINUOUS-FLUID EULERIAN METHOD FOR COMPRESSIBLE FLOW PROBLEMS

The Implicit Continuous-fluid Eulerian method (ICE) for multiphase flows is utilized by the Uintah Computational Framework (UCF) to simulate explosions, fires and other fluid and fluid-structure interaction phenomena [44]. The ICE code in UCF is referred to as Production ICE. The implementation of Production ICE is based on the fully cell-centered implementation of the ICE method (cell-centered ICE) by Kashiwa *et al.* [68, 69] with a few exceptions that are discussed in detail in Section 6.2. Cell-centered ICE of Kashiwa *et al.* [68] employs a conservative advection operator and a Lagrangian part which leaves a degree of freedom in the choice of conservation variables. The conservation laws used include at least those of mass, linear momentum, and internal energy (or alternatively the total energy). The Lagrangian method in most standard ICE implementations is fully conservative and it usually conserves the internal energy rather than the total energy. The numerical scheme used in Production ICE [44, 45, 52, 79, 80] solves the conservation of mass, linear momentum and internal energy. However, the Lagrangian method of Production ICE is a nonconservative form. While this may not be a problem for some cases, it appears to be a problem when applying this Production ICE code to single-fluid cases that are governed by the Euler equations in which the obtained numerical solutions exhibit some discrepancies in the shock speeds and they additionally show unphysical oscillations. With the need to improve the implementation of the Production ICE method, we start the improvement of Production ICE in the one-dimensional case. In order to improve Production ICE for the numerical solutions of compressible flow problems, we will explore the various choices in the implementation of cell-centered ICE and discuss how various choices affect the obtained numerical solutions. By exploring the various choices in the implementation of cell-centered ICE and by proposing an improvement of Production ICE for the one-dimensional case, we will provide some insights into the improvement of Production ICE for the multidimensional case.

The one-dimensional Production ICE method solves the time-dependent Euler equations of gas dynamics described by Equations (3.32)-(3.34) and the equation of state (3.36). It has been mentioned in [60, 75, 121] that nonconservative schemes approximating hyperbolic conservation laws do not converge to the correct solution in general. So the existence of discrepancies in the numerical solutions for nonlinear hyperbolic systems using Production ICE is quite understandable. Therefore, in order to improve the Production ICE method, we first must change the method to solve the system of one-dimensional Euler equations in conservative form where the total energy instead of the internal energy is conserved. The Improved Production ICE method, that will be referred to hereafter as IMPICE, is a cell-centered ICE method which solves the system of one-dimensional Euler equations in conservation form described by governing Equations (3.28)-(3.30) and the equation of state (3.31). As a result of changing the Production ICE method to solve the system of Euler equations in conservative form, the computational results show the disappearance of the discrepancies in the obtained numerical solutions. However, cell-centered ICE suffers from unphysical oscillations when there are moving contact discontinuities. Typically, methods in the literature use a variety of techniques such as constrained data reconstruction so as to avoid spurious oscillations; for example, [10, 11, 25, 129, 130, 131, 132]. To suppress oscillations, we will use a similar approach in which the data at the cell interface are the approximate Riemann solution to the local Riemann problem that is constructed by using slope-limited interpolation of the left and right cell-centered data. The approximate Riemann solver which was proposed by Harten *et al.* [53] and used by Davis [28] to satisfy consistency with the integral forms of the conservation law and entropy condition will be used to solve the local Riemann problem. The slope limiter used is selected from the extensive literature on the functions for slope limiters in the last few decades; see, for example, [39, 54, 116, 126, 129, 130, 131, 136]. The IMPICE method is cell-centered ICE with the oscillations being suppressed using the aforementioned technique. In effect, although the original ICE method is a Von Neumann type method in which the fluxes are fully dependent on the time increment, we have now introduced a ICE method with Riemann solver approach in which the fluxes depend directly on the approximate solution to the Riemann problem.

As for many numerical methods for the solution of PDEs, the IMPICE method approximates the Euler equations by a finite volume method using a spatial discretization of the problem and a time integration technique. Of comparable importance to having a nonoscillating numerical solution is determining the accuracy of the IMPICE method in time and space. cell-centered ICE with first-order advection is first-order in space and time. However, first-order methods are known to be not accurate enough to be used for large problems on relatively coarse grids. We can increase the orders of accuracy in time and space by applying a high order time discretization and a nonlinear spatial discretization, respectively. The goal is to obtain an IMPICE method with second-order accuracies in both time and space.

The content of this chapter is organized as follows. In Section 6.1, we recap the cell-centered ICE method by Kashiwa *et al.* [68] which includes the spatial discretization and essential steps in the time integration. In Section 6.2, we present the detail implementation of the Production ICE method and describe how the Production ICE method is different from the cell-centered ICE method by Kashiwa *et al.* [68]. In Section 6.3, we discuss the proposed method by Kwatra *et al.* [72] that can be applied to calculate the time integration step of the semi-implicit ICE method. This method removes the restriction of sound speed in calculating the time step, but still maintains stability. In Section 6.4, we propose a modification to the Production ICE method to remove the unphysical oscillations in the numerical solutions. The numerical solutions of the IMPICE method are presented and compared to the numerical solutions of the Production ICE method in Section 6.5. The spatial and temporal accuracies of the IMPICE method are shown in Section 6.6. We discuss in Section 6.7 and Section 6.8 how to obtain second-order accuracy in time and space, respectively. The conclusions are drawn in Section 6.9.

6.1 Cell-centered ICE by Kashiwa *et al.* [68]

6.1.1 General Cell-centered ICE

Cell-centered ICE for the one-dimensional space, which is described in detail in [68], is a finite-volume solver in which the computational domain $\Omega = [a_1, b_1] \in \mathbb{R}$ is discretized into N uniform cells of width $\Delta x = (b_1 - a_1)/N$. The cells are centered at $x_j = a_1 + (j - \frac{1}{2})\Delta x$ where $j = 1, ..., N$. The boundaries of these cells are located at $x_{j+\frac{1}{2}} = a_1 + j\Delta x$ where $j = 0, ..., N$ and are called *face-centers* or *cell interfaces*. With this discretization, the domain boundaries are aligned with the first and last cell edges.

A time integration method is used to estimate the averages of cell variables at some time $t = T_e$ from the averages of cell variables at $t = 0$. For each time integration step, assuming that the cell averages at time t_n are known, the goal is to compute the averages of cell variables at the next time step t_{n+1}. The time integration step of ICE invokes operator splitting in which the solution consists of a Lagrangian phase and an Eulerian phase. The Lagrangian phase advances cell values without advection and maps new values to cell variables and the Eulerian phase advects the cell variables. The essential point that makes the ICE method an all-speed scheme is to use an implicit scheme for the Lagrangian phase and an explicit scheme for the Eulerian phase. The time integration of cell-centered ICE comprises the following phases:

6.1.1.1 The Primary Phase

With the spatial discretization as discussed above, the spatial derivatives in the governed equations are approximated using finite differences of quantities at face-centers. Since an implicit scheme is used for the Lagrangian phase, the variables involved in the Lagrangian phase are those evaluated at face-centers at $t_n + \frac{\Delta t}{2}$ and are determined in the Primary phase. It is also necessary in the Primary phase to estimate the fluxing velocity which is going to be used in the Eulerian phase. The fluxing velocity, $u^*_{j+\frac{1}{2}}$, is the flux of volume across the cell interface. In order to make clear which variables are defined at face center, the superscript * is used here for these variables as required.

6.1.1.2 The Lagrangian Phase

Let V^n_j be the volume of cell j and $\mathbf{U}^n_j$ be the vector of averaging cell conserved variables at t_n. In particular, V^n_j is equal to Δx for the above discretization. Assume that the cell volume is changed during the Lagrangian phase to V^L_j and $V^L_j = V^n_j + \Delta t(u^*_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}})$ where $u^*_{j-\frac{1}{2}}$ and $u^*_{j+\frac{1}{2}}$ are fluxing velocities at cell interfaces. There is also a change in the vector of averaging cell variables to $\mathbf{U}^L_j$ after the Lagrangian phase has been completed. A numerical scheme obtained from neglecting the convective effects is used to evaluate the change in the material state and in turn evaluate $\mathbf{U}^L_j$.

6.1.1.3 The Eulerian Phase

For this phase, we have to evaluate the change in the solution due to advection. Let V^{n+1}_j be the cell volume at t_{n+1} and assume that the mesh is stationary, then $V^{n+1}_j = \Delta x$. The change in the solution due to advection is as follows:

$$V^{n+1}_j \mathbf{U}^{n+1}_j = V^L_j \mathbf{U}^L_j - \Delta t(u^*_{j+\frac{1}{2}} \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}} \langle \mathbf{U} \rangle^n_{j-\frac{1}{2}}), \tag{6.1}$$

where $\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} = \dfrac{1}{\Delta t} \displaystyle\int_{t_n}^{t_{n+1}} \mathbf{U}(x_{j+\frac{1}{2}}, t) dt$ is the vector of advected quantities and is numerically determined. As suggested by [68], this numerical value may be determined using $\mathbf{U}^n_j$ or $\mathbf{U}^L_j$; however, how to numerically determine these values is not described.

6.1.1.4 State Variables Update Phase

In this phase, we update the primitive cell variables $\mathbf{W}^{n+1}_j$ using the values of conserved variables $\mathbf{U}^{n+1}_j$ and their satisfaction of the equation of state. The value of $\mathbf{W}^{n+1}_j$ is then used in the following time integration step.

6.1.2 The Implementation of Cell-centered ICE by Kashiwa *et al.* [68]

Above is the general description of the ICE method. Kashiwa *et al.* [68] have made some improvements to the ICE method by changing the implementation of the Primary and Lagrangian phase. Two important quantities used in these phases are the face-centered velocity and pressure as denoted by $u^*_{j+\frac{1}{2}}$ and $p^*_{j+\frac{1}{2}}$. The face-centered velocity, $u^*_{j+\frac{1}{2}}$, is also the advected speed for the Eulerian phase, so [68] refers this as the fluxing velociy. The face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, is calculated based on the time-advanced equation for velocity as given by Equation (3.37). The fluxing velocity in cell-centered ICE of Kashiwa *et al.* [68] is obtained using the semi-implicit Euler scheme in the Lagrangian frame of Equation (3.37) as given by:

$$u^*_{j+\frac{1}{2}} = \langle\langle u^n_{j+\frac{1}{2}}\rangle\rangle^\rho - \frac{\Delta t}{2\Delta x}\frac{p^{n+\frac{1}{2}}_{j+1} - p^{n+\frac{1}{2}}_{j}}{\langle\langle \rho^n_{j+\frac{1}{2}}\rangle\rangle}, \tag{6.2}$$

where $p^{n+\frac{1}{2}}_j$ is the cell-centered pressure at $t_n + \frac{\Delta t}{2}$, $\langle\langle u^n_{j+\frac{1}{2}}\rangle\rangle^\rho$ is the mass-weighted average face-centered velocity, and $\langle\langle \rho^n_{j+\frac{1}{2}}\rangle\rangle$ is the average face-centered density at time t_n. The mass-weighted average velocity, $\langle\langle u^n_{j+\frac{1}{2}}\rangle\rangle^\rho$, of left and right states at face-center is given by:

$$\langle\langle u^n_{j+\frac{1}{2}}\rangle\rangle^\rho = \frac{\rho^n_j u^n_j + \rho^n_{j+1} u^n_{j+1}}{\rho^n_j + \rho^n_{j+1}}, \tag{6.3}$$

and the average face-centered density, $\langle\langle \rho^n_{j+\frac{1}{2}}\rangle\rangle$, of left and right cell-centered densities is given by:

$$\langle\langle \rho^n_{j+\frac{1}{2}}\rangle\rangle = \frac{\rho^n_j + \rho^n_{j+1}}{2}. \tag{6.4}$$

As the cell-centered pressure at $t_n + \frac{\Delta t}{2}$, $p^{n+\frac{1}{2}}_j$, is not readily available, it needs to be obtained from correcting the cell-centered pressure at t_n, p^n_j. Let $\delta p^n_j = p^{n+\frac{1}{2}}_j - p^n_j$ be the difference between the cell-centered pressures at these two time levels and which is referred to in Kashiwa *et al.* [68] as the "pressure corrector"; Equation (6.2) is then rewritten by using these values as:

$$u^*_{j+\frac{1}{2}} = \tilde{u}^*_{j+\frac{1}{2}} - \frac{\Delta t}{2\Delta x}\frac{\delta p^n_{j+1} - \delta p^n_j}{\langle\langle \rho^n_{j+\frac{1}{2}}\rangle\rangle}, \tag{6.5}$$

where:

$$\tilde{u}^*_{j+\frac{1}{2}} = \langle\langle u^n_{j+\frac{1}{2}}\rangle\rangle^\rho - \frac{\Delta t}{2\Delta x}\frac{p^n_{j+1} - p^n_j}{\langle\langle \rho^n_{j+\frac{1}{2}}\rangle\rangle}. \tag{6.6}$$

In order to determine the face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, the "pressure corrector" values, δp^n_{j+1} and δp^n_j, need to be determined using Equation (3.38). In [68], two different ways to determine the "pressure corrector" values are discussed. The explicit "pressure corrector" uses the explicit discrete form and the implicit "pressure corrector" uses the semi-implicit discrete form of (3.38). The explicit form of "pressure corrector" is given by:

$$\delta p^n_j = -\frac{\Delta t}{2} u^n_j \left(\frac{p^n_{j+1} - p^n_{j-1}}{2\Delta x}\right) - \frac{\Delta t}{2\Delta x}(c^2\rho)^n_j(\langle\langle u^n_{j+\frac{1}{2}}\rangle\rangle^\rho - \langle\langle u^n_{j-\frac{1}{2}}\rangle\rangle^\rho). \tag{6.7}$$

The implicit "pressure corrector" is obtained using:

$$\delta p^n_j = -\frac{\Delta t}{2} u^n_j \left(\frac{p^n_{j+1} - p^n_{j-1}}{2\Delta x}\right) - \frac{\Delta t}{2\Delta x}(c^2\rho)^n_j(u^*_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}}). \tag{6.8}$$

The implicit "pressure corrector" is complicated since Equations (6.5) and (6.8) show that the face-centered fluxing velocities and the "pressure corrector" values are interrelated, so there is a need to calculate the fluxing velocities using these two equations and this is not explicitly discussed in [68]. After having determined the "pressure corrector" values at the cell centers, the face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, is determined using Equation (6.5) where $\tilde{u}^*_{j+\frac{1}{2}}$ is defined in (6.6).

There are several suggested choices for calculating the face-centered pressure, $p^*_{j+\frac{1}{2}}$, in [68] and [69]. Two of these choices, which are derived from the pressure equation in Kashiwa *et al* in 1994 [68], will be discussed in Appendix A.4. In this chapter, we employ the choice that is mentioned in Kashiwa in 2001 [69]. This choice aims to satisfy continuity of acceleration by equating acceleration increments for the left/right half spaces. The equation as specified in [69] is:

$$p^*_{j+\frac{1}{2}} = \left(\frac{\frac{1}{\rho^n_{j+1}} p^{n+\frac{1}{2}}_{j+1} + \frac{1}{\rho^n_j} p^{n+\frac{1}{2}}_j}{\frac{1}{\rho^n_j} + \frac{1}{\rho^n_{j+1}}}\right). \tag{6.9}$$

In the above equation, the face-centered pressure is thus calculated using specific volumes-weighted of the left and right time-advanced cell-centered pressures.

The Primary phase is executed after the face-centered velocity and pressure, $u^*_{j+\frac{1}{2}}$ and $p^*_{j+\frac{1}{2}}$, are calculated. The choice of the vector of conserved variables, $\mathbf{U}$, and the numerical procedure to determine the vector of the face-centered advected quantities, $\langle \mathbf{U} \rangle_{j+\frac{1}{2}}$, are neccessary for the implementation of the Lagrangian phase and the Eulerian phase.

6.2 Production ICE in the Uintah Computational Framework

The term Production ICE is used to denote the ICE method as implemented in the Uintah Computational Framework (UCF) by [44, 45, 52, 79, 80] to simulate fluid flows that are governed by the Euler and Navier Stokes equations. Production ICE solves the Euler system in nonconservative form described by Equations (3.32)–(3.34) with the vector of variables $\mathbf{U} = [\rho, \rho u, \rho e]^T$. The detail implementation of the phases in Production ICE follows the description given in Kashiwa *et al.* [68] with some exceptions that will be pointed out explicitly in the following discussion.

6.2.1 The Primary Phase

The first exception is that the face-centered quantities in Production ICE are not time-centered. The face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, and pressure, $p^*_{j+\frac{1}{2}}$, for the time step $[t_n, t_{n+1}]$ are approximated at the face-center at time t_{n+1}. For this reason, we use the notations $u^{**}_{j+\frac{1}{2}}$ and $p^{**}_{j+\frac{1}{2}}$ for the face-centered fluxing velocity and pressure in Production ICE. Using a different approach from Equation (6.2), the face-centered fluxing velocity in Production ICE, $u^{**}_{j+\frac{1}{2}}$, is approximated as:

$$u^{**}_{j+\frac{1}{2}} = \langle\langle u^n_{j+\frac{1}{2}} \rangle\rangle^{\rho} - \frac{\Delta t}{\Delta x} \frac{p^n_{j+1} - p^n_j}{\langle\langle \rho^n_{j+\frac{1}{2}} \rangle\rangle}, \tag{6.10}$$

where the mass-weighted average velocity, $\langle\langle u^n_{j+\frac{1}{2}} \rangle\rangle^{\rho}$, is defined in (6.3) and the average face-centered density, $\langle\langle \rho^n_{j+\frac{1}{2}} \rangle\rangle$, is defined as follows:

$$\langle\langle \rho^n_{j+\frac{1}{2}} \rangle\rangle = \frac{2.0 \left(\rho^n_j + \rho^n_{j+1} \right)}{\rho^n_j \rho^n_{j+1}}. \tag{6.11}$$

So another exception in calculating the fluxing velocity in Production ICE is that the scheme in (6.10) is not semi-implicit when the pressures used are defined at t_n.

The face-centered pressure, $p^{**}_{j+1/2}$, in Production ICE is calculated using the following equation:

$$p^{**}_{j+\frac{1}{2}} = \left(\frac{\frac{1}{\rho^n_{j+1}} p^{n+1}_{j+1} + \frac{1}{\rho^n_j} p^{n+1}_j}{\frac{1}{\rho^n_j} + \frac{1}{\rho^n_{j+1}}} \right). \tag{6.12}$$

This is similar to Equation (6.9), but with the cell-centered pressures at time t_{n+1}, p^{n+1}_j, where p^{n+1}_j is evaluated using an explicit scheme applied to the Lagrangian form of the equation of pressure evolution in (3.38), which is given by:

$$p^{n+1}_j = p^n_j - \frac{\Delta t}{\Delta x}(c^2\rho)^n_j(\langle\langle u^n_{j+\frac{1}{2}}\rangle\rangle^\rho - \langle\langle u^n_{j-\frac{1}{2}}\rangle\rangle^\rho). \tag{6.13}$$

6.2.2 The Lagrangian Phase

Production ICE chooses the vector of conserved variables to include mass, linear momentum and internal energy. The use of the nonconservative form of the system of Euler equations in (3.32)–(3.34) means that the Lagrangian part of Production ICE is given by:

$$V^L_j \mathbf{U}^L_j = V^n_j \mathbf{U}^n_j - \Delta t \begin{bmatrix} 0 \\ p^{**}_{j+1/2} - p^{**}_{j-1/2} \\ p^{n+1}_j(u^{**}_{j+1/2} - u^{**}_{j-1/2}) \end{bmatrix}. \tag{6.14}$$

6.2.3 The Eulerian Phase

The change in solution values due to advection over the step $[t_n, t_{n+1}]$ is given by:

$$V^{n+1}_j \mathbf{U}^{n+1}_j = V^L_j \mathbf{U}^L_j - \Delta t(u^{**}_{j+\frac{1}{2}}\langle\mathbf{U}\rangle^n_{j+\frac{1}{2}} - u^{**}_{j-\frac{1}{2}}\langle\mathbf{U}\rangle^n_{j-\frac{1}{2}}). \tag{6.15}$$

However, the numerical values of face-centered advected quantities in the following definition:

$$\langle\mathbf{U}\rangle^n_{j+\frac{1}{2}} = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{U}(x_{j+\frac{1}{2}}, t)dt \tag{6.16}$$

has not been quantified so far in this chapter and we will now show how to approximate it. Normally $\mathbf{U}(x_{j+\frac{1}{2}}, t)$ is not constant for the step $[t_n, t_{n+1}]$, but first-order accuracy is obtained by assuming that this is constant and is an upwinded cell-centered value. However, there are the cell-centered values at two different time levels that are available for the Eulerian phase. These are the value at t_n and the value after the Lagrangian step. By chosing the upwinded cell-centered values at time t_n for the face-centered advected quantities, we have:

$$\langle\mathbf{U}\rangle^n_{j+\frac{1}{2}} = \begin{cases} \mathbf{U}^n_{j+1} & \text{if} \quad (u^*_{j+\frac{1}{2}} < 0) \\ \mathbf{U}^n_j & \text{otherwise.} \end{cases} \tag{6.17}$$

Alternatively, if the upwinded cell-centered values at Lagrangian time level are considered for face-centered advected quantities, we have:

$$\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} = \begin{cases} \mathbf{U}^L_{j+1} & \text{if} \quad (u^*_{j+\frac{1}{2}} < 0) \\ \mathbf{U}^L_j & \text{otherwise.} \end{cases} \tag{6.18}$$

The Production ICE code uses Equation (6.18) to define the face-centered advected quantities, [52, 79, 80].

6.2.4 State Variables Update Phase

The averages of cell variables ρ, u, e, and p are then updated using the averages of cell conserved variables ρ, ρu, ρe, and the equation of state (3.31).

6.3 CFL Condition

The choice of the time step Δt in time integration affects the stability of the ICE method. As mentioned in [120], one requirement for the method to be stable is the fastest wave at a given time is allowed to travel, at most, one cell length Δx in the chosen time step Δt. For the system of Euler equations, the time step Δt is chosen to satisfy the condition:

$$\Delta t = \frac{C_{cfl} \Delta x}{S^n_{max}}, \tag{6.19}$$

where C_{cfl} is a Courant or CFL coefficient satisfying $0 < C_{cfl} < 1$ and S^n_{max} is the largest wave speed present through the domain at time t_n. A practical choice of S^n_{max} as mentioned in Toro [120] is:

$$S^n_{max} = \max_j \left(|u^n_j| + c^n_j \right). \tag{6.20}$$

However, Kwatra *et al.* [72] proposed a novel method for alleviating the stringent CFL condition imposed by the sound speed in simulating highly nonlinear compressible flow with shocks, contacts and rarefactions. It is mentioned in [72] that the maximum speed in Equation (6.20) is too restrictive for flows where the sound speed, c, may be much larger than $|u|$, so the stringent CFL time step restriction imposed by the acoustic waves can be avoided and only the material velocity CFL restriction is used in calculating the maximum speed. The proposed method of [72] is well suited to the semi-implicit solver like the ICE method where only the advection step is the explicit part. The proposed maximum speed calculation in [72] is:

$$S^n_{max} = \max_j |u^n_j|. \tag{6.21}$$

The time step used is determined using (6.19) where S^n_{max} is calculated using (6.21).

6.4 IMPICE Method

We now propose the IMPICE method, an improved implementation of Production ICE, which aims to eliminate the discrepancies and suppress the nonphysical oscillations in the numerical solutions to the one-dimensional, time-dependent Euler equations of gas dynamics. Foremost, the IMPICE method makes an improvement to Production ICE by solving the Euler equations in conservative form in (3.28)–(3.30) and the equation of state in (3.31). We denote cell-centered ICE that solves the conservative form of Euler equations as the conservative cell-centered ICE. As shown in Appendix A.3, conservative cell-centered ICE can eliminate the discrepancies in the obtained numerical solutions. However, the obtained numerical solutions have unphysical oscillations that need to be reduced or eliminated. The oscillations in the numerical solutions of conservative cell-centered ICE cannot be diminished by decreasing the time step, so in this section, we will describe the algorithm used to suppress these oscillations numerically by using a simple approximate Riemann solver.

6.4.1 Numerical Discussion

To help explain the IMPICE method, we start with a discussion of schemes that approximate conservation laws as follows and consider a one-dimensional system in a conservation law form:

$$\frac{\partial \mathbf{U}(x,t)}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U}(x,t))}{\partial x} = 0, \quad x \in [a_1, b_1] \quad \text{and} \quad t \geq 0, \tag{6.22}$$

where $\mathbf{U}(x,t)$ is the vector of conserved variables and $\mathbf{F}(\mathbf{U}(x,t))$ is the vector of fluxes. In order to approximate the solution of (6.22) with the initital condition:

$$\mathbf{U}(x,0) = \mathbf{U}_0(x), \tag{6.23}$$

we discretize space into N uniform cells as in Section 6.1. The cell average of the cell $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ at time t_n is denoted by $\mathbf{U}^n_j$, where:

$$\mathbf{U}^n_j = \frac{1}{\Delta x}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \mathbf{U}(x,t_n)dx. \tag{6.24}$$

A standard approach is used in integrating system (6.22) in space and time in the control volume $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [t_n, t_{n+1}]$ to give:

$$\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} [\mathbf{U}(x,t_{n+1}) - \mathbf{U}(x,t_n)]\,dx = -\int_{t_n}^{t_{n+1}} \Big[\mathbf{F}(\mathbf{U}(x_{j+\frac{1}{2}},t)) - \mathbf{F}(\mathbf{U}(x_{j-\frac{1}{2}},t))\Big]\,dt.$$

This can then be written in the standard conservation form:

$$\Delta x \mathbf{U}_j^{n+1} = \Delta x \mathbf{U}_j^n - \Delta t \left(\mathbf{F}_{j+\frac{1}{2}}(t_n) - \mathbf{F}_{j-\frac{1}{2}(t_n)}\right) \tag{6.25}$$

where:

$$\mathbf{F}_{j+\frac{1}{2}}(t_n) = \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \mathbf{F}(\mathbf{U}(x_{j+\frac{1}{2}},t))dt. \tag{6.26}$$

Equation (6.25) is used by finite volume methods to solve the system (6.22) approximately. In order to use this relation, a spatial integration of the initital condition is required and the approximations of the fluxes at the cell interfaces are needed.

The numerical flux derivation follows cell-centered ICE of Kashiwa *et al.* [68] will be derived shortly. The system of Euler equations (3.28)–(3.30) of gas dynamics is written in the form (6.22) where $\mathbf{U} = [\rho, \rho u, \rho E]^T$ and $\mathbf{F}(\mathbf{U}) = \left[\rho u, \rho u^2 + p, \rho u E + up\right]^T$. The face-centered flux (6.26) in this case is written as:

$$\mathbf{F}_{j+\frac{1}{2}}(t_n) = \begin{bmatrix} \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho u)(x_{j+\frac{1}{2}},t)dt \\ \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho u^2 + p)(x_{j+\frac{1}{2}},t)dt \\ \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho u E + pu)(x_{j+\frac{1}{2}},t)dt \end{bmatrix}. \tag{6.27}$$

A Taylor series approximation of $u(x_{j+\frac{1}{2}},t)$ is given by:

$$u(x_{j+\frac{1}{2}},t) = u(x_{j+\frac{1}{2}},t_{n+\frac{1}{2}}) + (t - t_{n+\frac{1}{2}})u_t(x_{j+\frac{1}{2}},t_{n+\frac{1}{2}}) + O(\Delta t^2). \tag{6.28}$$

Using the notation $u_{j+\frac{1}{2}}^{n+\frac{1}{2}} = u(x_{j+\frac{1}{2}},t_{n+\frac{1}{2}})$ and $(u_t)_{j+\frac{1}{2}}^{n+\frac{1}{2}} = u_t(x_{j+\frac{1}{2}},t_{n+\frac{1}{2}})$, we have:

$$\begin{aligned} \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \rho u(x_{j+\frac{1}{2}},t)dt &= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \rho(x_{j+\frac{1}{2}},t)\left(u_{j+\frac{1}{2}}^{n+\frac{1}{2}} + (t - t_{n+\frac{1}{2}})(u_t)_{j+\frac{1}{2}}^{n+\frac{1}{2}} + O(\Delta t^2)\right)dt \\ &= u_{j+\frac{1}{2}}^{n+\frac{1}{2}}\left(\frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \rho(x_{j+\frac{1}{2}},t)dt\right) + O(\Delta t^2). \end{aligned}$$

With a similar approach, we also have:

$$\frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho u^2 + p)(x_{j+\frac{1}{2}},t)dt = u_{j+\frac{1}{2}}^{n+\frac{1}{2}}\left(\frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho u)(x_{j+\frac{1}{2}},t)dt\right) + p_{j+\frac{1}{2}}^{n+\frac{1}{2}} + O(\Delta t^2)$$

and:

$$\frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho u E + up)(x_{j+\frac{1}{2}},t)dt = u_{j+\frac{1}{2}}^{n+\frac{1}{2}}\left(\frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho E)(x_{j+\frac{1}{2}},t)dt\right) + (up)_{j+\frac{1}{2}}^{n+\frac{1}{2}} + O(\Delta t^2).$$

Then face-centered flux $\mathbf{F}_{j+\frac{1}{2}}(t_n)$ vector in (6.27) is rewritten as:

$$\mathbf{F}_{j+\frac{1}{2}}(t_n) = u_{j+\frac{1}{2}}^{n+\frac{1}{2}}\begin{bmatrix}\frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}\rho(x_{j+\frac{1}{2}},t)dt \\ \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho u)(x_{j+\frac{1}{2}},t)dt \\ \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}(\rho E)(x_{j+\frac{1}{2}},t)dt\end{bmatrix} + \begin{bmatrix}0 \\ p_{j+\frac{1}{2}}^{n+\frac{1}{2}} \\ (up)_{j+\frac{1}{2}}^{n+\frac{1}{2}}\end{bmatrix} + \mathbf{O}(\Delta t^2). \tag{6.29}$$

The reader should note that Equation (6.25) with the terms $\mathbf{F}_{j-\frac{1}{2}}$ and $\mathbf{F}_{j+\frac{1}{2}}$ are defined by Equation (6.29) will be used in Section 6.6 to assess the numerical accuracy of the IMPICE method.

6.4.2 IMPICE Implementation

The scheme used for approximating the fluxing velocity, $u_{j+\frac{1}{2}}^*$, in the IMPICE method is similar to Equation (6.2), that is:

$$u_{j+\frac{1}{2}}^* = u_{j+\frac{1}{2}}^n - \frac{\Delta t}{2\Delta x}\frac{\left(p_{j+1}^{n+\frac{1}{2}} - p_j^{n+\frac{1}{2}}\right)}{\rho_{j+\frac{1}{2}}^n}. \tag{6.30}$$

This equation was obtained by replacing $\langle\langle u_{j+\frac{1}{2}}^n\rangle\rangle^\rho$ in (6.2) with $u_{j+\frac{1}{2}}^n$, and $\langle\langle \rho_{j+\frac{1}{2}}^n\rangle\rangle$ with $\rho_{j+\frac{1}{2}}^n$. While these quantities denote the velocity and density at face-center at t_n, their numerical values are determined differently. While $\langle\langle u_{j+\frac{1}{2}}^n\rangle\rangle^\rho$ and $\langle\langle \rho_{j+\frac{1}{2}}^n\rangle\rangle$ are determined using the weighted averages in (6.3) and (6.4), the values $u_{j+\frac{1}{2}}^n$ and $\rho_{j+\frac{1}{2}}^n$ are determined based on the simple approximate Riemann solver that will be discussed in Section 6.4.3 below.

Since the pressures used in (6.30) are time-advanced values, we need to perform a "pressure corrector" to obtain these. The explicit "pressure corrector" in (6.7) is the one used in our implementation of the IMPICE method. However, it is worth looking at the implicit "pressure corrector" and seeing the difference between the solutions of these two methods. By substituting (6.5) into (6.8), the equation for cell-centered "pressure corrector" now becomes:

$$\begin{aligned}\delta p_j^n &= -\frac{\Delta t}{2}u_j^n\left(\frac{p_{j+1}^n - p_{j-1}^n}{2\Delta x}\right) - \frac{\Delta t}{2\Delta x}(c^2\rho)_j^n(\tilde{u}_{j+\frac{1}{2}}^* - \tilde{u}_{j-\frac{1}{2}}^*) \\ &+ \left[\frac{\Delta t}{2\Delta x}\right]^2(c^2\rho)_j^n\left[\frac{\delta p_{j+1}^n - \delta p_j^n}{\rho_{j+\frac{1}{2}}^n} - \frac{\delta p_j^n - \delta p_{j-1}^n}{\rho_{j-\frac{1}{2}}^n}\right].\end{aligned}$$

where $\tilde{u}^*_{j+\frac{1}{2}}$ is defined by Equation (6.6).

Let $\sigma = \dfrac{\Delta t}{2\Delta x}$ and rearrange the terms of above equation to get:

$$\left[1 + \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j+\frac{1}{2}}} + \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j-\frac{1}{2}}}\right] \delta p^n_j - \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j+\frac{1}{2}}} \delta p^n_{j+1} - \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j-\frac{1}{2}}} \delta p^n_{j-1}$$
$$= -\sigma u^n_j \left(\frac{p^n_{j+1} - p^n_{j-1}}{2}\right) - \sigma (c^2\rho)^n_j (\tilde{u}^*_{j+\frac{1}{2}} - \tilde{u}^*_{j-\frac{1}{2}}).$$

Therefore, the values δp^n_j are the solutions of the tri-diagonal linear system:

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{6.31}$$

where:

$$\mathbf{A} = \begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & . & \\ & & . & . & c_{N-1} \\ 0 & & & a_N & b_N \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ . \\ d_N \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} \delta p_1 \\ \delta p_2 \\ \delta p_3 \\ . \\ \delta p_N \end{bmatrix} \tag{6.32}$$

and a_j, b_j, c_j, d_j are defined as follows:

$$\begin{aligned} a_j &= \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j-\frac{1}{2}}} && \text{for} \quad j = 2..(N-1), \\ b_j &= 1 + \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j+\frac{1}{2}}} + \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j-\frac{1}{2}}} && \text{for} \quad j = 2..(N-1), \\ c_j &= \sigma^2 \frac{(c^2\rho)^n_j}{\rho^n_{j+\frac{1}{2}}} && \text{for} \quad j = 2..(N-1), \\ d_j &= -\sigma u^n_j \left(\frac{p^n_{j+1} - p^n_{j-1}}{2}\right) - \sigma (c^2\rho)^n_j (\tilde{u}^*_{j+\frac{1}{2}} - \tilde{u}^*_{j-\frac{1}{2}}) && \text{for} \quad j = 2..(N-1), \end{aligned}$$

and b_1, b_N, d_1 and d_N are obtained from the boundary condition. So in order to use the implicit "pressure corrector", we have to solve the tri-diagonal linear system in (6.31). While this obviously takes more time to calculate than the explicit "pressure corrector" in (6.7), the results computed using the two methods show that there is not much difference between the numerical solutions obtained from using the implicit and explicit "pressure corrector" in the IMPICE method for the Euler equations examples used here.

The IMPICE method calculates the face-centered pressure, $p^*_{j+\frac{1}{2}}$, the same way as the cell-centered ICE method does in Section 6.1.2. It uses the calculation described in Equation (6.9).

The IMPICE method chooses to conserve the total energy instead of internal energy, so the vector of conserved variables is $\mathbf{U} = [\rho, \rho u, \rho E]^T$. The Lagrangian and Eulerian phases of the IMPICE method are then given by:

$$V_j^L \mathbf{U}_j^L = V_j^n \mathbf{U}_j^n - \Delta t \begin{bmatrix} 0 \\ p^*_{j+1/2} - p^*_{j-1/2} \\ p^*_{j+1/2} u^*_{j+1/2} - p^*_{j-1/2} u^*_{j-1/2} \end{bmatrix}, \tag{6.33}$$

and:

$$V_j^{n+1} \mathbf{U}_j^{n+1} = V_j^L \mathbf{U}_j^L - \Delta t \left(u^*_{j+\frac{1}{2}} \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}} \langle \mathbf{U} \rangle^n_{j-\frac{1}{2}} \right), \tag{6.34}$$

in which $V_j^L = V_j^n + \Delta t \left(u^*_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}} \right)$ and the terms $\langle \mathbf{U} \rangle^n_{j-\frac{1}{2}}$ and $\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}}$ are given by Equation (6.18).

6.4.3 Application of Slope Limiters in the IMPICE Method

In common with many methods for conservative laws, slope limiters may be applied to the calculation of face-centered fluxing velocity, $u^*_{j+1/2}$. For the face-centered fluxing velocity, slope limiters are used in the estimation of face-centered quantities at t_n; in particular, they are used in the calculation of $\rho^n_{j+\frac{1}{2}}$ and $u^n_{j+\frac{1}{2}}$. This approach originates from the idea of approximating the cell-centered state by the reconstructed states obtained from the left and right cell-averaged states of the previous time step. The slope limited, reconstructed states are used as inputs to a Riemann solver to determine the state at the cell interface. This will be discussed in detail below.

Let $\mathbf{W}_j^n$, $\mathbf{W}_j^n = \left[\rho_j^n, u_j^n, E_j^n, p_j^n \right]^T$, be the vector of average cell-centered values of primitive variables of cell j at time t_n, then the value of $\mathbf{W}$ on the spatial domain at t_n is represented by the piecewise constant data $\{\mathbf{W}_j^n\}$. The simplest and widely used way to modify the piecewise constant data $\{\mathbf{W}_j^n\}$ is to replace the constant state $\mathbf{W}_j^n$ by a piecewise linear functions $\mathbf{W}_j^n(x)$. The construction of the piecewise linear functions can be found in many papers; the construction in Toro [120] will be used as described below.

As for the first-order Godunov method, one assumes that $\mathbf{W}_j^n$ represents an integral average in cell $I_j = [x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}]$ as given by:

$$\mathbf{W}_j^n = \frac{1}{\Delta x} \int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \mathbf{W}_j^n(x) dx. \tag{6.35}$$

A piecewise linear, local reconstruction of $\mathbf{W}_j^n$ is:

$$\mathbf{W}_j^n(x) = \mathbf{W}_j^n + (x - x_j)\Delta\mathbf{W}_j^n, \quad x \in I_j, \tag{6.36}$$

where $\Delta\mathbf{W}_j^n$ is a suitably chosen slope of $\mathbf{W}_j^n(x)$ in cell I_j. The integral of $\mathbf{W}_j^n(x)$ in cell I_j is identical to that of $\mathbf{W}_j^n$ and thus the reconstruction process is conservative. The slope $\Delta\mathbf{W}_j^n$ can be approximated by a simple finite difference formula given by:

$$\Delta\mathbf{W}_j^n = \frac{\mathbf{W}_{j+1}^n - \mathbf{W}_j^n}{\Delta x}. \tag{6.37}$$

However, to achieve a higher order scheme and to maintain bounded solutions, the slope at the current node is usually limited based on adjacent slopes. The obtained slope is a "limited slope" $\bar{\Delta}\mathbf{W}_j^n$ which is used as:

$$\mathbf{W}_j^n(x) = \mathbf{W}_j^n + (x - x_j)\bar{\Delta}\mathbf{W}_j^n, \quad x \in I_j, \tag{6.38}$$

to approximate $\mathbf{W}$ on I_j. The ratio $\mathbf{r}_j^n$ represents the ratio of successive gradients on the solution mesh at x_j:

$$\mathbf{r}_j^n = \frac{\mathbf{W}_j^n - \mathbf{W}_{j-1}^n}{\mathbf{W}_{j+1}^n - \mathbf{W}_j^n}, \tag{6.39}$$

and the limited slope $\Delta\mathbf{W}_j^n$ may be written in the form:

$$\bar{\Delta}\mathbf{W}_j^n = \boldsymbol{\phi}(\mathbf{r}_j^n)\Delta\mathbf{W}_j^n, \tag{6.40}$$

where $\boldsymbol{\phi}(\mathbf{r}_j^n)$ is some flux limiter function. Note that the division and multiplication in (6.39) and (6.40) are the component-wise operations. For the results in this chapter, we choose the Monotonized Central(MC) limiter function for calculating the limited slope in (6.40). The MC limiter function by Van Leer [130] is:

$$\boldsymbol{\phi}(\mathbf{r}) = max[0, min(2\mathbf{r}, 0.5 + 0.5\mathbf{r}, 2)]. \tag{6.41}$$

At each interface $x_{j+\frac{1}{2}}$, we now may consider the so-called Generalized Riemann Problem(GRP) as follows:

$$\frac{\partial\mathbf{U}}{\partial t} + \frac{\partial\mathbf{F}(\mathbf{U})}{\partial x} = 0, \tag{6.42}$$

$$\mathbf{W}(x,t_n) = \begin{cases} \mathbf{W}_j^n(x) & \text{if } \ (x < x_{j+\frac{1}{2}}) \\ \mathbf{W}_{j+1}^n(x) & \text{if } \ (x > x_{j+\frac{1}{2}}) \end{cases} \tag{6.43}$$

where $\mathbf{W}_j^n(x)$ is the limited local reconstruction in (6.38). Naturally, for nonlinear systems the exact solution of the GRP is exceedingly complicated, but for the purpose of evaluating face-centered states, an approximate solution may be suffice. In this approach, we are not trying to evaluate the solution of the GRP in (6.42) analytically but rely on the boundary extrapolated values at the interfaces. The values of $\mathbf{W}_{j+\frac{1}{2}}^n$ at cell boundaries using local reconstructions $\mathbf{W}_j^n(x)$ and $\mathbf{W}_{j+1}^n(x)$ are denoted as $W_{j+\frac{1}{2}}^{n(L)}$ and $W_{j+\frac{1}{2}}^{n(R)}$ where:

$$\mathbf{W}_{j+\frac{1}{2}}^{n(L)} = \mathbf{W}_j^n(x_{j+\frac{1}{2}}) = \mathbf{W}_j^n + 0.5\phi(\mathbf{r}_j^n)\left(\mathbf{W}_{j+1}^n - \mathbf{W}_j^n\right); \tag{6.44}$$

$$\mathbf{W}_{j+\frac{1}{2}}^{n(R)} = \mathbf{W}_{j+1}^n(x_{j+\frac{1}{2}}) = \mathbf{W}_{j+1}^n - 0.5\phi(\mathbf{r}_{j+1}^n)\left(\mathbf{W}_{j+2}^n - \mathbf{W}_{j+1}^n\right). \tag{6.45}$$

The values $\mathbf{W}_{j+\frac{1}{2}}^{n(L)}$ and $\mathbf{W}_{j+\frac{1}{2}}^{n(R)}$ are left and right extrapolated values at the boundary $x_{j+\frac{1}{2}}$ at time t_n. In this way, one may instead consider the conventional Riemann Problem with piecewise constant data in a new coordinate (ξ,τ) where $\xi = x - x_{j+\frac{1}{2}}$ and $\tau = t - t_n$ as:

$$\frac{\partial \mathbf{U}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \xi} = 0, \tag{6.46}$$

$$\mathbf{W}(\xi,0) = \begin{cases} \mathbf{W}_{j+\frac{1}{2}}^{n(L)} & \text{if } \ (\xi < 0) \\ \mathbf{W}_{j+\frac{1}{2}}^{n(R)} & \text{if } \ (\xi > 0). \end{cases} \tag{6.47}$$

The face-centered state at t_n, $\mathbf{W}(0,0)$, is the value at the origin immediately after the interaction of the piecewise constant data $\mathbf{W}_j^{n(L)}$ and $\mathbf{W}_{j+1}^{n(R)}$ where:

$$\mathbf{W}(0,0) = \lim_{\tau \to 0^+} \mathbf{W}(0,\tau). \tag{6.48}$$

By determining $\mathbf{W}(0,0)$ in (ξ,τ) coordinate, we have the values of face-centered states given by $\mathbf{W}_{j+\frac{1}{2}}^n = [\rho_{j+\frac{1}{2}}^n, u_{j+\frac{1}{2}}^n, p_{j+\frac{1}{2}}^n]^T$ at t_n. There are several ways to approximate the solution to the piecewise constant data Riemann problem (6.46) and therefore to approximate $\mathbf{W}(0,0)$. In this paper, we use the simple approximate Riemann solver which was proposed by Harten *et al.* [53] and discussed in Davis [28] to approximate $\mathbf{W}(0,0)$. In order to use the approximate Riemann solver described in these papers, we rewrite Equation (6.46) as:

$$\frac{\partial \mathbf{U}}{\partial \tau} + \frac{\partial \mathbf{F}}{\partial \xi} = 0, \tag{6.49}$$

$$\mathbf{U}(\xi, 0) = \begin{cases} \mathbf{U}_L & \text{if } (\xi < 0) \\ \mathbf{U}_R & \text{if } (\xi > 0), \end{cases} \tag{6.50}$$

where $\mathbf{U}_L$ and $\mathbf{U}_R$ are obtained from $\mathbf{W}^{n(L)}_{j+\frac{1}{2}}$ and $\mathbf{W}^{n(R)}_{j+\frac{1}{2}}$ respectively. The approximate Riemann solution of (6.49) is given by:

$$\mathbf{U}(x/t; \mathbf{U}_L, \mathbf{U}_R) = \begin{cases} \mathbf{U}_L & \text{for } (x/t < a_L) \\ \mathbf{U}_{LR} & \text{for } (a_L < x/t < a_R) \\ \mathbf{U}_R & \text{for } (a_R < x/t)\,, \end{cases} \tag{6.51}$$

where a_L and a_R are lower and upper bounds, respectively, for the smallest and largest signal velocity and:

$$\mathbf{U}_{LR} = \frac{a_R \mathbf{U}_R - a_L \mathbf{U}_L}{a_R - a_L} - \frac{\mathbf{F}(\mathbf{U}_R) - \mathbf{F}(\mathbf{U}_L)}{a_R - a_L}. \tag{6.52}$$

The bounds a_R and a_L for the Euler equations are defined in Davis [28] as:

$$a_L = u_L - c_L, \quad a_R = u_R + c_R, \tag{6.53}$$

where u_L, c_L are the velocity and wave speed respectively obtained from $\mathbf{U}_L$, and u_R, c_R are the velocity and wave speed obtained from $\mathbf{U}_R$. The solution $\mathbf{W}(0,0)$ in (6.48) is derived from the approximate solution $\mathbf{U}(0; \mathbf{U}_L, \mathbf{U}_R)$ in (6.51) which includes the approximations of $u^n_{j+\frac{1}{2}}$ and $\rho^n_{j+\frac{1}{2}}$; these in turn are used in Equation (6.30) instead of using the mass-weighted quantities in Equations (6.3) and (6.4).

In summary, the face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, is estimated via the following steps. First, using the local recontruction in (6.38), the left and right extrapolated values at this cell-center are obtained using (6.44) and (6.45). These extrapolated values then form the piecewise constant data to the Riemann problem (6.46). Second, this Riemann problem is solved approximately using the approach of Harten *et al.* [53] and Davis [28]. The approximate Riemann solution includes the approximate face-centered density, $\rho^n_{j+\frac{1}{2}}$, and face-centered velocity, $u^n_{j+\frac{1}{2}}$. Third, the "pressure correctors", δp^n_j, are calculated using Equation (6.8). Finally, Equations (6.5) and (6.6) are used to calculate $u^*_{j+\frac{1}{2}}$.

6.5 Numerical Results and Comparisons

The following well-known test problems are often used to test the accuracy and robustness of many numerical methods in fluids. These tests for the one-dimensional, time-dependent

Euler equations for ideal gases can be found in Toro [120]. In [120], these examples are used to access the performance of the numerical schemes being presented in the book. These tests are also employed here to illustrate the performance of the Production ICE method and the IMPICE method. In these chosen problems, two constant states, $\mathbf{W}_L = [\rho_L, u_L, p_L]^T$ and $\mathbf{W}_R = [\rho_R, u_R, p_R]^T$, are separated by a discontinuity at a position $x = x_0$. The states $\mathbf{W}_L$ and $\mathbf{W}_R$ are given in Table 6.1 and the problem domain is $(x,t) \in [0,1] \times (0, T_e]$. We use the transmissive boundary for these problems.

P1 is the well-known Sod's problem and **P2** is a modified version of **P1**. These tests are considered very mild, but as mentioned in Toro [120] they are useful for assessing numerical methods. **P3** is considered a very hard problem for numerical methods. As mentioned in Toro [120], the solution to **P4** represents the collision of two strong shocks and consists of a left-facing shock, a right travelling contact discontinuity and a right-travelling shock wave. Problem **P5** is Lax's test problem [74].

Beside the problems with known exact solutions in Table 6.1, we also include here the numerical solutions to the Shu and Osher [107] test problem. This test problem contains detailed features and structures and is considered by Greenough and Rider [42] to be a good one-dimensional surrogate for the interaction of a shock wave with a turbulent field. The initial condition at $t = 0$ of the problem is defined as:

$$(\rho, u, p)(x, 0) = \begin{cases} (3.85714, 2.62936, 10.33333) & \text{if } \ (x < -4.0) \\ (1.0 + 0.2sin(5x), 0.0, 1.0) & \text{otherwise,} \end{cases} \tag{6.54}$$

on spatial domain $[-5.0, 5.0]$. The end time for this problem is $T_e = 1.8$. As the analytical solution of this test problem is not readily available, we use the "exact" solution obtained from our implementation of the unmodified WENO-JS scheme discussed in Martin *et al.* [86] to show how accurate of the numerical methods presented in this chapter. The "exact" solution of Shu and Osher test problem in this chapter is obtained with the unmodified WENO-JS scheme with $r = 3$ and $p = 2$ on 6400 grid points.

Table 6.1. Data for one-dimensional test problems with known exact solutions, for the time-dependent, one-dimensional Euler equations

Problem	ρ_L	u_L	p_L	ρ_R	u_R	p_R	x_0	T_e
P1	1.0	0.0	1.0	0.125	0.0	0.1	0.3	0.2
P2	1.0	0.75	1.0	0.125	0.0	0.1	0.3	0.2
P3	1.0	0.0	1000	1.0	0.0	0.1	0.5	0.011
P4	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950	0.4	0.034
P5	0.445	0.698	3.528	0.5	0.0	0.571	0.5	0.16

The numerical results for the above test problems of the IMPICE method are compared against those of the Production ICE method and shown in Figures 6.1–6.6. The time step in Production ICE is chosen using Equations (6.19) and (6.20) while the time step in the IMPICE method is chosen using Equations (6.19) and (6.21). It shows that the maximum speed calculation of Kwatra *et al.* [72] indeed alleviates the stringent CFL condition imposed by the sound speed in simulating these test problems. As seen in Figures 6.1–6.6, a significant improvement in numerical solutions of the IMPICE method is shown. The profiles of the numerical solutions of Production ICE in these figures are not close to the exact solutions. This is due to the use of the nonconservative form in Production ICE. To see how the use of the nonconservative scheme affects the numerical solution profiles, we include in Appendix

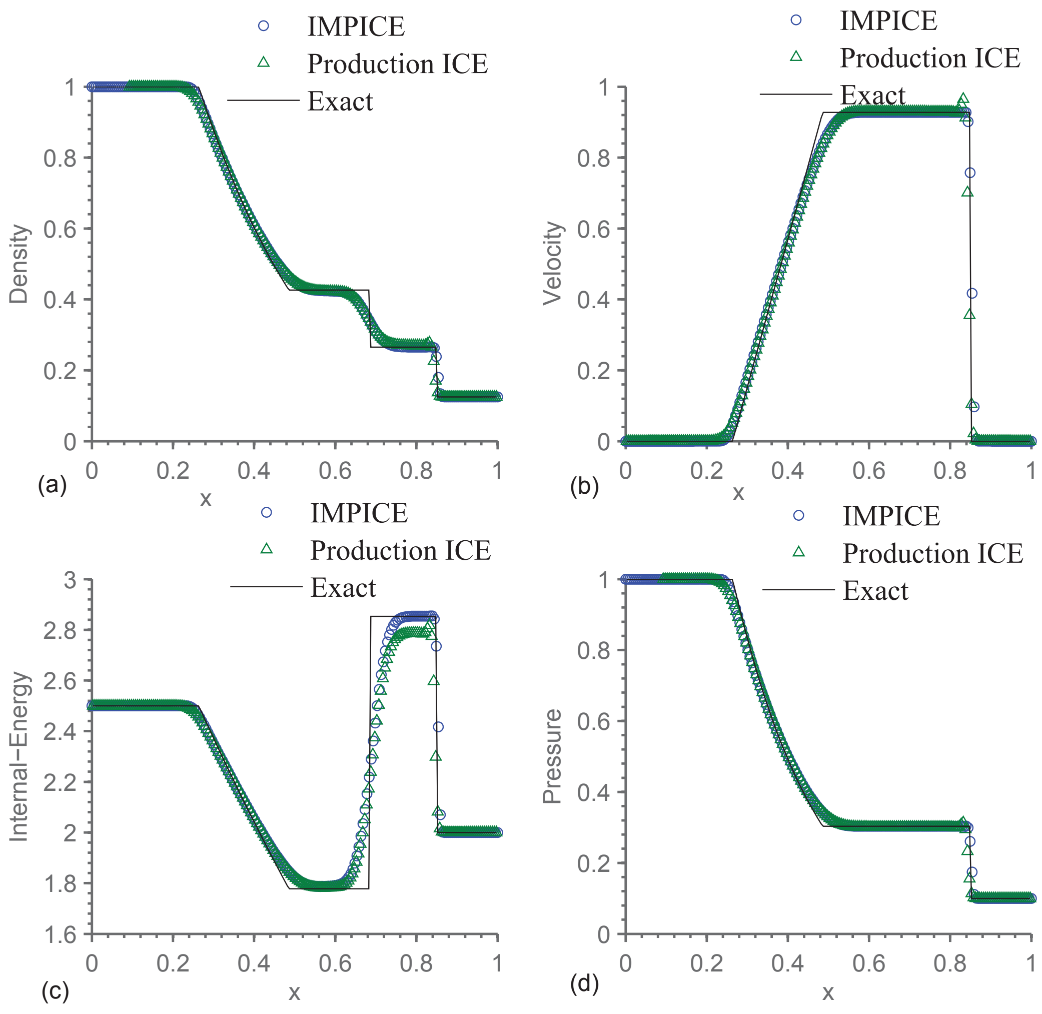


Figure 6.1. Production ICE and IMPICE numerical solutions for test **P1** with N=200 (cells), $C_{cfl} = 0.2$, and first-order advection: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

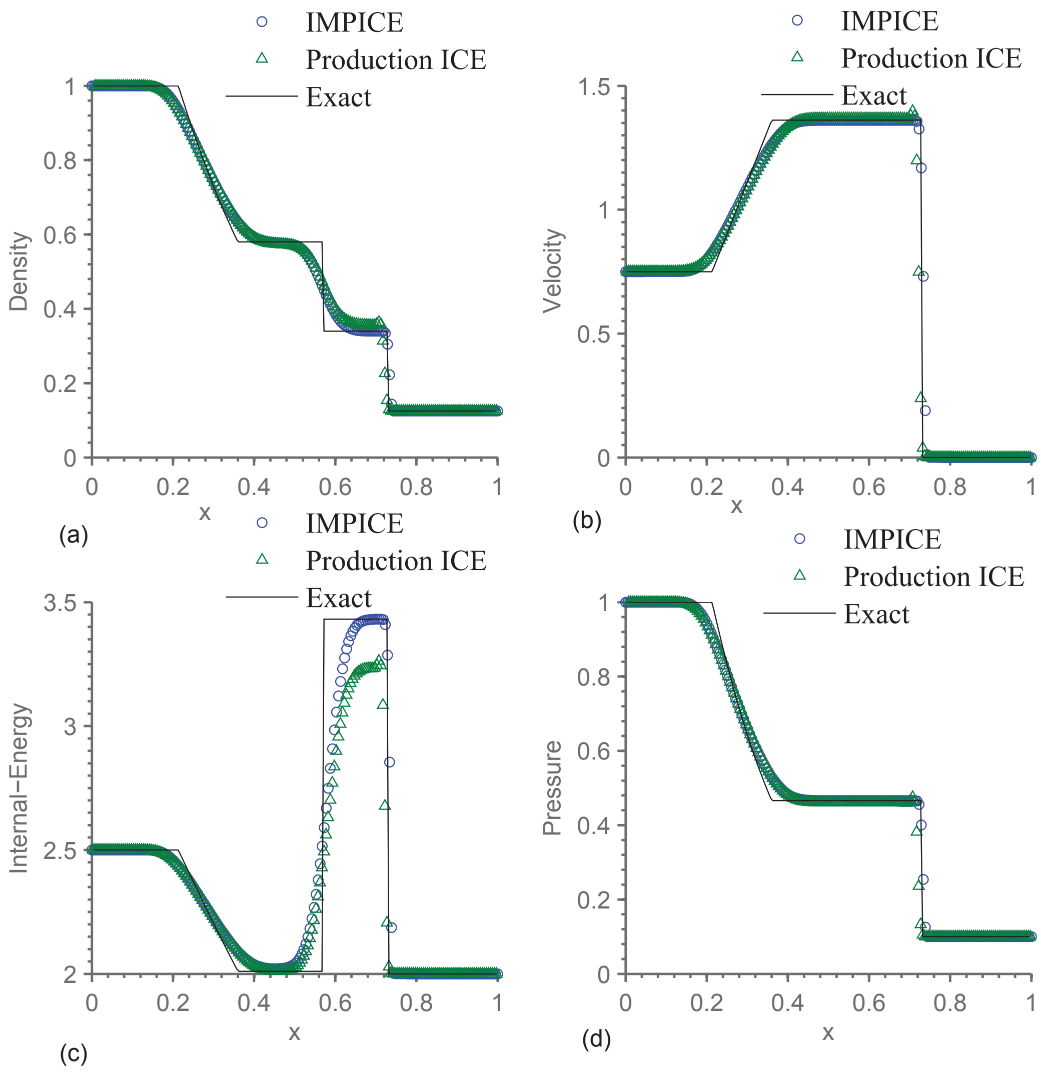


Figure 6.2. Production ICE and IMPICE numerical solutions for test **P2** with N=200 (cells), C_{cfl} = 0.2, and first-order advection: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

A.3 the comparision between the numerical results of the IMPICE method and conservative cell-centered ICE. Conservative cell-centered ICE denotes the method that is implemented using cell-centered ICE of Kashiwa *et al.* [68] described in Section 6.1.2 which conserves mass, linear momentum and total energy. From the results in Figures 6.1–6.6 and Appendix A.3, it may be seen that the use of conservation form improves the solution profiles. It can also be seen that, there are no existing oscillations at the shock-front in the numerical solutions of the IMPICE method. This results from the application of slope limiters in the data reconstruction of the Riemann problem as shown in Appendix A.3. In short, the results in 6.1–6.6 and Appendix A.3 show that the use of conservation form improves the solution profiles and the reconstruction of

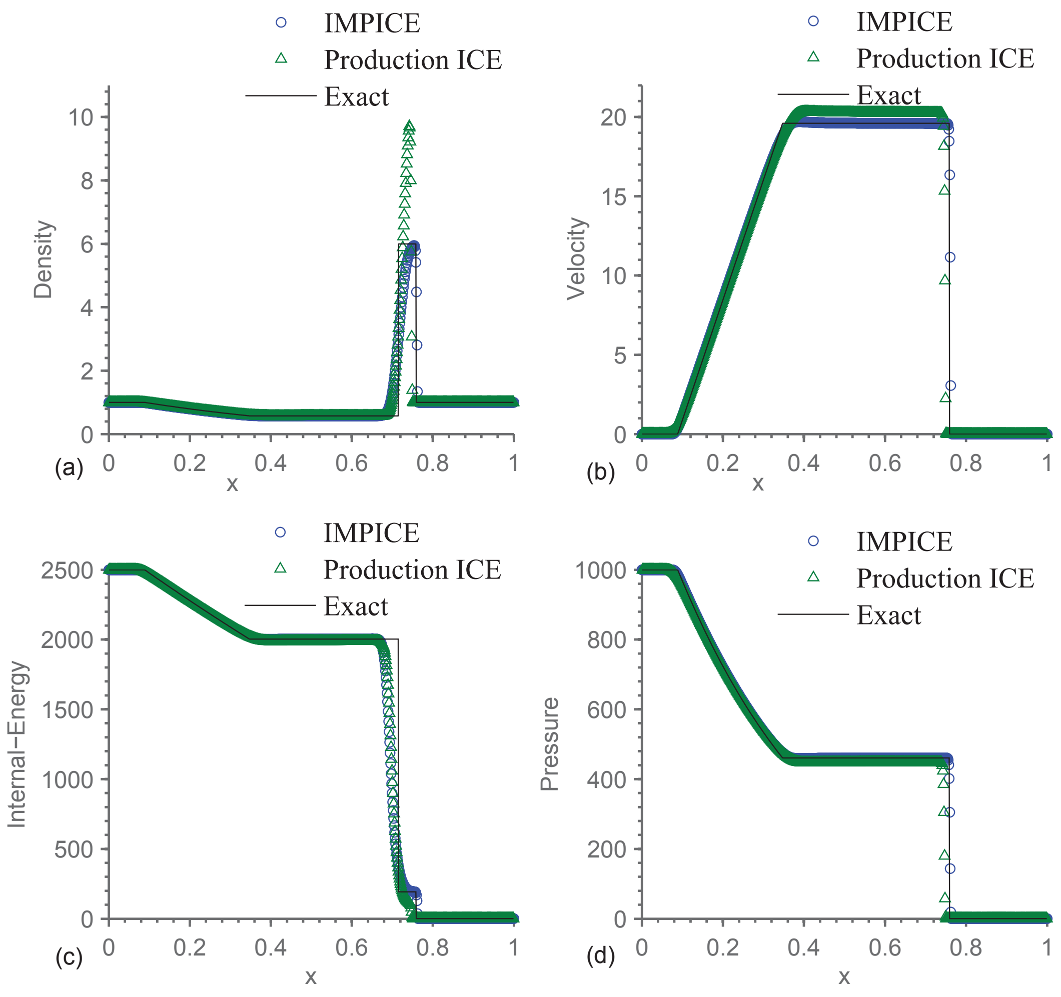


Figure 6.3. Production ICE and IMPICE numerical solutions for test **P3** with N=800 (cells), $C_{cfl} = 0.2$, and first-order advection: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

the Riemann problem with the slope limiters helps to eliminate the nonphysical oscillations.

6.6 Accuracy in Space and Time

6.6.1 Temporal Error

Let $\mathbf{le}^{\mathbf{U}}(t_{n+1})$ be the time integration error in $\mathbf{U}$ introduced at the cell centers over the step $[t_n, t_{n+1}]$ of the IMPICE method. The time integration error per step at cell j is then given by:

$$\mathbf{le}_j^{\mathbf{U}}(t_{n+1}) = \mathbf{U}_j\left[t_{n+1}; t_n, \mathbf{U}_j^n\right] - \mathbf{U}_j^{n+1}, \tag{6.55}$$

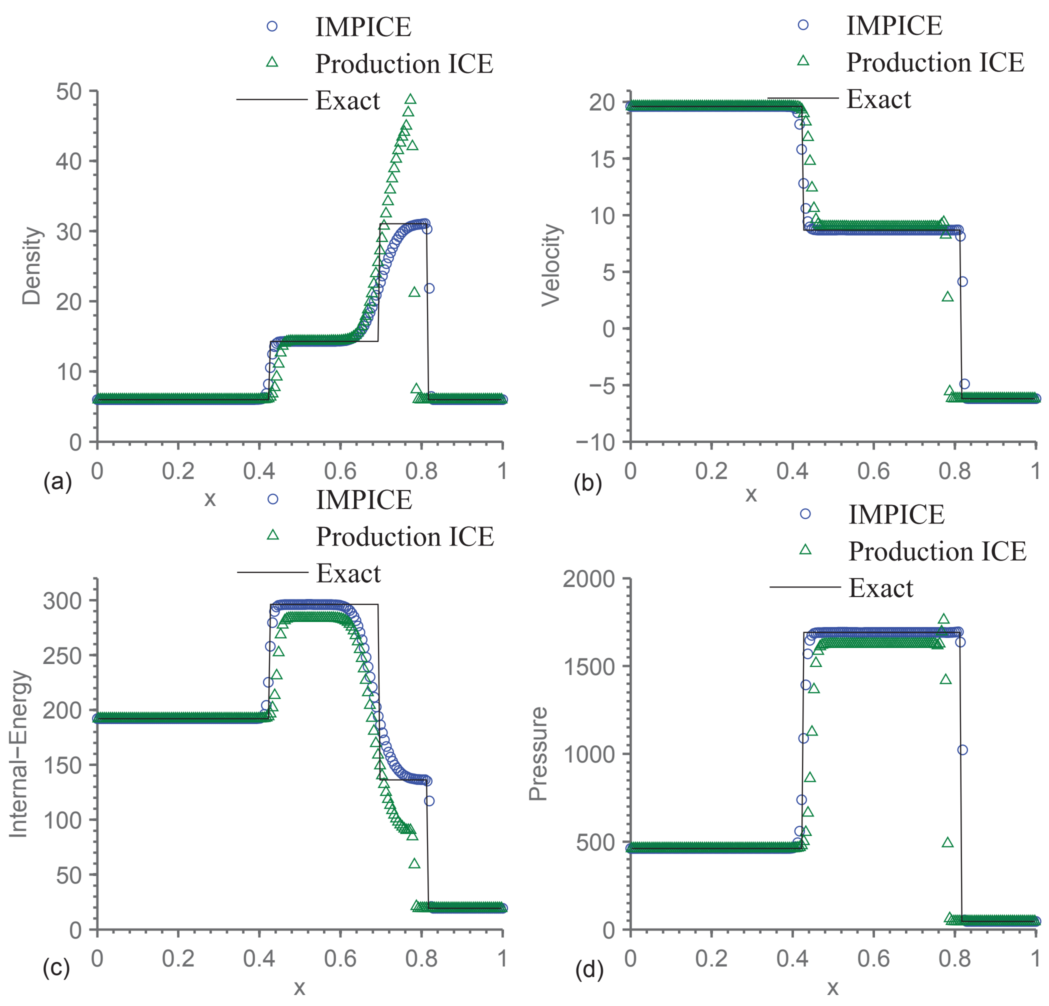


Figure 6.4. Production ICE and IMPICE numerical solutions for test **P4** with N=200 (cells), $C_{cfl} = 0.2$, and first-order advection: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

where $\mathbf{U_j}\left[t_{n+1}; t_n, \mathbf{U}_j^n\right]$ is the exact cell value at t_{n+1} if the exact cell value at t_n is $\mathbf{U}_j^n$. From Equation (6.25), the exact cell value $\mathbf{U}_j\left[t_{n+1}; t_n, \mathbf{U}_j^n\right]$ is given by:

$$\mathbf{U}_j\left[t_{n+1}; t_n, \mathbf{U}_j^n\right] = \mathbf{U}_j^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{j+\frac{1}{2}}(t_n) - \mathbf{F}_{j-\frac{1}{2}(t_n)}\right), \tag{6.56}$$

where $F_{j+\frac{1}{2}}(t_n)$ is defined in (6.29). On the other hand, the IMPICE solution at t_{n+1} is:

$$\mathbf{U}_j^{n+1} = \mathbf{U}_j^n - \frac{\Delta t}{\Delta x}\left(\mathbf{F}_{j+\frac{1}{2}}^{IMPICE}(t_n) - \mathbf{F}_{j-\frac{1}{2}}^{IMPICE}(t_n)\right), \tag{6.57}$$

where:

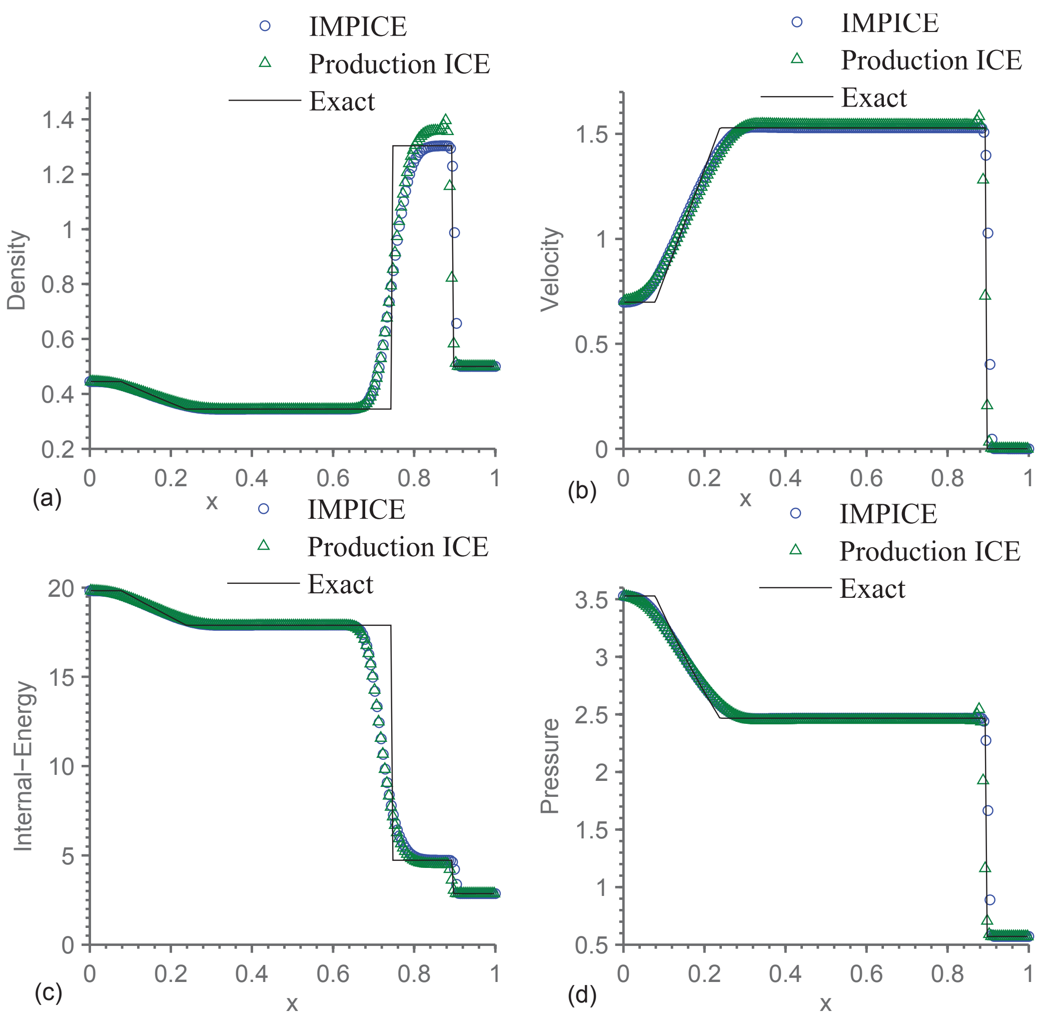


Figure 6.5. Production ICE and IMPICE numerical solutions for test **P5** with N=200 (cells), C_{cfl} = 0.2, and first-order advection: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

$$\mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n) = u^*_{j+\frac{1}{2}} \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} + \begin{bmatrix} 0 \\ p^*_{j+\frac{1}{2}} \\ u^*_{j+\frac{1}{2}} p^*_{j+\frac{1}{2}} \end{bmatrix}. \tag{6.58}$$

Therefore:

$$\mathbf{le}^{\mathbf{U}}_j(t_{n+1}) = \frac{\Delta t}{\Delta x}\left[\left(\mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n) - \mathbf{F}_{j+\frac{1}{2}}(t_n)\right) - \left(\mathbf{F}^{IMPICE}_{j-\frac{1}{2}}(t_n) - \mathbf{F}_{j-\frac{1}{2}}(t_n)\right)\right]. \tag{6.59}$$

From Equations (6.29) and (6.58), we have:

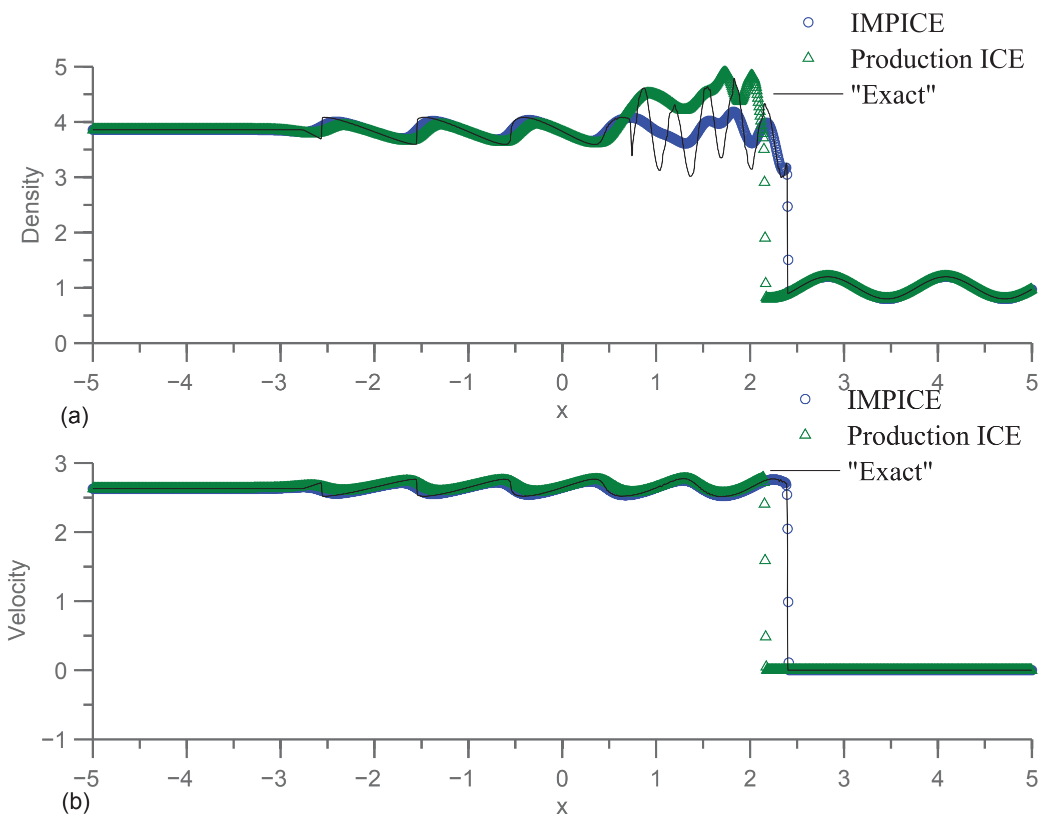


Figure 6.6. Production ICE and IMPICE numerical solutions for Shu and Osher test problem with N=1600 (cells), $C_{cfl} = 0.2$, and first-order advection: (a) density and (b) velocity.

$$\mathbf{F}_{j+\frac{1}{2}}(t_n) - \mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n) = u^{n+\frac{1}{2}}_{j+\frac{1}{2}} \left(\frac{1}{\Delta t} \int_{t_{n-1}}^{t_n} \mathbf{U}(x_{j+\frac{1}{2}}, t)dt - \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} \right)$$
$$+ \left(u^{n+\frac{1}{2}}_{j+\frac{1}{2}} - u^*_{j+\frac{1}{2}} \right) \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} + \begin{bmatrix} 0 \\ p^{n+\frac{1}{2}}_{j+\frac{1}{2}} - p^*_{j+\frac{1}{2}} \\ (up)^{n+\frac{1}{2}}_{j+\frac{1}{2}} - u^*_{j+\frac{1}{2}} p^*_{j+\frac{1}{2}} \end{bmatrix} + \mathbf{O}(\Delta t^2). \tag{6.60}$$

As $\left(u^{n+\frac{1}{2}}_{j+\frac{1}{2}} - u^*_{j+\frac{1}{2}} \right) = O(\Delta t^2)$ and $\left(p^{n+\frac{1}{2}}_{j+\frac{1}{2}} - p^*_{j+\frac{1}{2}} \right) = O(\Delta t^2)$, we then have:

$$\mathbf{F}_{j+\frac{1}{2}}(t_n) - \mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n) = u^{n+\frac{1}{2}}_{j+\frac{1}{2}} \left(\frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \mathbf{U}(x_{j+\frac{1}{2}}, t)dt - \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} \right) + \mathbf{O}(\Delta t^2). \tag{6.61}$$

By considering the expansion of $\mathbf{U}(x_{j+\frac{1}{2}}, t)$ about $\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}}$:

$$\mathbf{U}(x_{j+\frac{1}{2}}, t) = \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} + (t - t_n) \left(\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} \right)_t + \frac{(t-t_n)^2}{2} \left(\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} \right)_{tt} + ..., \tag{6.62}$$

Equation (6.61) now becomes:

$$\mathbf{F}_{j+\frac{1}{2}}(t_n) - \mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n) = \frac{\Delta t}{2} u^{n+\frac{1}{2}}_{j+\frac{1}{2}} \left(\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} \right)_t + \mathbf{O}(\Delta t^2). \tag{6.63}$$

Therefore:

$$\left(\mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n) - \mathbf{F}_{j+\frac{1}{2}}(t_n)\right) - \left(\mathbf{F}^{IMPICE}_{j-\frac{1}{2}}(t_n) - \mathbf{F}_{j-\frac{1}{2}}(t_n)\right) \tag{6.64}$$
$$= \frac{\Delta t}{2} \left[u^{n+\frac{1}{2}}_{j+\frac{1}{2}} \left(\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}} \right)_t - u^{n+\frac{1}{2}}_{j-\frac{1}{2}} \left(\langle \mathbf{U} \rangle^n_{j-\frac{1}{2}} \right)_t \right] + \mathbf{O}(\Delta t^2). \tag{6.65}$$

From Equations (6.59) and (6.65), $\mathbf{le}^{\mathbf{U}}_j(t_n)$ is second-order in Δt for a fixed Δx. Therefore, $\mathbf{et}^{\mathbf{U}}_j(T_e)$ is first-order in Δt.

As discussed in Chapter 4, the overall temporal error at cell j at t_n is given by:

$$\mathbf{et}^{\mathbf{U}}_j(t_n) = \mathbf{U}_j \left[t_n; t_0, \mathbf{U}^0_j \right] - \mathbf{U}^n_j. \tag{6.66}$$

In order to calculate the overall temporal error at T_e in Equation (6.66), we need to determine the IMPICE time-integrated exact solution $\mathbf{U}_j \left[T_e; t_0, \mathbf{U}^0_j \right]$. As we do not have the exact solution $\mathbf{U}_j \left[T_e; t_0, \mathbf{U}^0_j \right]$, we assume that the calculated solution U^n_j converges to the time-integrated exact solution $\mathbf{U}_j \left[T_e; t_0, U^0_j \right]$ when reducing C_{cfl}. Therefore, we use a highly resolved solution as the time-integrated exact solution $\mathbf{U}_j \left[T_e; t_0, U^0_j \right]$ with $C_{cfl} = 0.0001$. This solution meets the criterion mentioned in Greenough and Rider [42] that the grid converged solutions should be at least 8 times finer than the finest grid examined for error. The temporal error norms and their orders of accuracy of the conserved and primitive variables for the above test cases at T_e are shown in Table 6.2. The results in Table 6.2 show that the orders of accuracy of the conserved variables for these test cases are very close to one; this is consistent with the above analysis. The orders of accuracy of the primitive variables are also very close to one. With the result of the method with first-order-in-time shown in Appendix A.2, we may confirm that we do indeed get first-order convergence in the case of smooth solutions.

6.6.2 Spatial Error

With the linear spatial discretization as discussed above, the spatial error of the vector of conserved variables $\mathbf{U}$ is first-order in Δx. As discussed in Chapter 4, the overall spatial error at cell j at t_n is given by:

$$\mathbf{es}^{\mathbf{U}}_j(t_n) = \mathbf{U}(x_j, t_n) - \mathbf{U}_j \left[t_n; t_0, \mathbf{U}^0_j \right]. \tag{6.67}$$

Table 6.2. Temporal Error: L_1-norms and the order of accuracy n of the conserved and primitive variables for the test cases in Table 6.1 using N=200 (cells). The time-integrated exact solutions $\mathbf{U}_j\left[T_e; t_0, U_j^0\right]$ for the discretized problems of these test cases are obtained by using $C_{cfl} = 0.0001$. The notation aE-b used here stands for $a \times 10^{-b}$.

		$\mathbf{et}^{\rho}(T_e)$		$\mathbf{et}^{\rho u}(T_e)$		$\mathbf{et}^{\rho E}(T_e)$		$\mathbf{et}^{u}(T_e)$		$\mathbf{et}^{p}(T_e)$	
	C_{cfl}	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n
	0.4	6.92E-04	—	5.71E-04	—	1.16E-03	—	1.05E-03	—	4.42E-04	—
P1	0.2	3.37E-04	1.04	2.74E-04	1.06	5.54E-04	1.07	5.14E-04	1.03	2.11E-04	1.07
	0.1	1.67E-04	1.01	1.36E-04	1.01	2.76E-04	1.00	2.55E-04	1.00	1.05E-04	1.00
	0.05	8.36E-05	1.00	6.82E-05	1.00	1.39E-04	1.00	1.27E-04	1.00	5.29E-05	1.00
	0.4	1.53E-03	—	1.42E-03	—	3.15E-03	—	1.98E-03	—	1.11E-03	—
P2	0.2	7.55E-04	1.02	6.96E-04	1.03	1.56E-03	1.01	9.90E-04	1.00	5.55E-04	1.00
	0.1	3.74E-04	1.01	3.45E-04	1.01	7.78E-04	1.01	4.93E-04	1.00	2.76E-04	1.00
	0.05	1.86E-04	1.01	1.71E-04	1.01	3.88E-04	1.00	2.46E-04	1.00	1.38E-04	1.00
	0.4	1.08E-02	—	2.18E-01	—	3.44E-00	—	3.12E-02	—	6.61E-01	—
P3	0.2	5.20E-03	1.06	1.04E-01	1.06	1.63E-00	1.08	1.53E-02	1.03	3.11E-01	1.09
	0.1	2.57E-03	1.02	5.16E-02	1.02	8.06E-01	1.02	7.64E-03	1.00	1.53E-01	1.02
	0.05	1.27E-03	1.02	2.55E-02	1.01	4.00E-01	1.01	3.81E-03	1.01	7.59E-02	1.01
	0.4	3.94E-02	—	3.24E-01	—	5.26E-00	—	1.85E-02	—	2.05E-00	—
P4	0.2	1.94E-02	1.02	1.59E-01	1.03	2.59E-00	1.02	8.87E-03	1.06	9.76E-00	1.07
	0.1	9.45E-03	1.04	7.90E-02	1.01	1.26E-00	1.04	4.41E-03	1.01	4.85E-01	1.01
	0.05	5.01E-03	0.91	3.94E-02	1.00	6.10E-01	1.05	2.23E-03	0.98	2.47E-01	0.98
	0.4	2.11E-03	—	3.53E-03	—	6.32E-03	—	1.39E-03	—	1.64E-03	—
P5	0.2	1.04E-03	1.02	1.75E-03	1.02	3.21E-03	0.98	7.25E-04	0.94	8.55E-03	0.94
	0.1	5.18E-04	1.01	8.67E-04	1.01	1.61E-03	1.00	3.66E-04	0.98	4.30E-04	0.99
	0.05	2.58E-04	1.01	4.32E-04	1.01	8.03E-04	1.00	1.83E-04	1.00	2.15E-04	1.00

In order to access the spatial errors of a test case, we need the exact solution $\mathbf{U}_j\left[T_e; t_0, U_j^0\right]$; see Equation (6.67). The result in Section 6.1.1 gives the rate at which the time integration computed solution approaches the true time integration solution, so a more accurate approximation to the exact solution $\mathbf{U}_j\left[T_e; t_0, U_j^0\right]$ might be obtained by comparing the numerical solution to a finer-mesh numerical solution. Therefore, we estimate the exact solution $\mathbf{U}_j\left[T_e; t_0, U_j^0\right]$ in Table 6.3 using the computed solutions of the IMPICE method with $C_{cfl} = 0.025$ and one with $C_{cfl} = 0.0125$.

The spatial error norms and their orders of accuracy for the above test cases at T_e are shown in Table 6.3.

Theoretically the spatial error order of accuracy is first-order. However it is shown in Table 6.3 that the order of accuracy is mostly below one due to the discontinuities in the solutions of these test cases. Greenough and Rider [42] mention earlier work showing less than first-order accuracy for the first-order version of Godunov's method and suggest that this is due to the low resolution computed solutions being very different from the highly resolved solution. For reference purposes, we include in Appendices A.1 and A.2 of this dissertation the spatial errors and the orders of accuracy for the inviscid and viscous Burgers' problems. For the inviscid Burgers' problem, the order of accuracy is below one. For the viscous Burgers' problem, the order is around one.

As numerical solutions obtained with first-order methods are diffusive and not accurate enough to be used for some large problems on relatively coarse grids – for example, the numerical solution to Shu and Osher test problem shown in Figure 6.6 – we raise the order of accuracy of the IMPICE method to second-order in both space and time in the following sections.

6.7 Higher-order Accuracy in Time

In order to raise the order of accuracy globally in time, we use the method of extrapolation to raise the order of accuracy locally. By raising the local order of accuracy of the temporal error to third-order, we raise the order of accuracy of the temporal error to second-order globally. The second-order-in-time IMPICE method is achieved using Richardson extrapolation. The steps in the IMPICE method with second-order temporal error to obtain the solution for next time step $\mathbf{U}_j^{n+1}$ from current time step solution $\mathbf{U}_j^n$ are:

- Perform one step of the first-order IMPICE method with stepsize Δt to obtain the solution $\mathbf{U1}_j^{n+1}$ at t_{n+1}.
- Perform two consecutive steps of the first-order IMPICE method with stepsize $\frac{\Delta t}{2}$ to obtain the solution $\mathbf{U2}_j^{n+1}$ at t_{n+1}.
- Set the solution at t_{n+1} of second-order-in-time IMPICE method to $\left(2\mathbf{U2}_j^{n+1} - \mathbf{U1}_j^{n+1}\right)$.

Table 6.3. Spatial Error: L_1-norms and the order of accuracy m of the conserved and primitive variables for the test cases in Table 6.1. The exact solutions $\mathbf{U}_j\left[T_e; t_0, \mathbf{U}_j^0\right]$ are the converged numerical solutions discussed in Section 6.6.1.

		$\mathbf{es}^{\rho}(T_e)$		$\mathbf{es}^{\rho u}(T_e)$		$\mathbf{es}^{\rho E}(T_e)$		$\mathbf{es}^{u}(T_e)$		$\mathbf{es}^{p}(T_e)$	
	N	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m
	100	1.38E-02	—	1.15E-02	—	2.68E-02	—	2.15E-02	—	1.06E-02	—
	200	9.16E-03	0.59	7.44E-03	0.63	1.56E-02	0.79	1.20E-02	0.84	6.10E-03	0.79
P1	400	5.83E-03	0.65	4.65E-03	0.68	9.10E-03	0.77	6.67E-03	0.85	3.48E-03	0.81
	800	3.68E-03	0.66	2.92E-03	0.67	5.29E-03	0.78	3.63E-03	0.88	1.95E-03	0.84
	1600	2.40E-03	0.62	1.89E-03	0.62	3.06E-03	0.79	1.95E-03	0.90	1.09E-03	0.84
	100	2.17E-02	—	1.89E-02	—	4.47E-02	—	2.83E-02	—	1.66E-02	—
	200	1.45E-02	0.58	1.26E-02	0.59	2.87E-02	0.64	1.63E-02	0.80	1.02E-02	0.71
P2	400	9.87E-03	0.55	8.93E-03	0.50	1.80E-02	0.67	9.25E-03	0.82	6.11E-03	0.74
	800	6.45E-03	0.61	5.98E-03	0.58	1.10E-02	0.71	5.15E-03	0.85	3.58E-03	0.77
	1600	4.24E-03	0.60	4.13E-03	0.53	7.04E-03	0.64	3.13E-03	0.72	2.17E-03	0.72
	100	1.80E-01	—	3.49E+00	—	8.06E+01	—	5.56E-01	—	1.16E+01	—
	200	1.47E-01	0.29	2.96E+00	0.24	4.84E+01	0.73	3.38E-01	0.72	6.93E+00	0.74
P3	400	1.09E-01	0.43	2.20E+00	0.43	3.48E+01	0.48	1.95E-01	0.79	4.10E+00	0.76
	800	7.60E-02	0.52	1.50E+00	0.55	2.26E+01	0.62	1.06E-01	0.87	2.33E+00	0.81
	1600	5.44E-02	0.48	1.07E+00	0.49	1.29E+01	0.81	5.63E-02	0.92	1.25E+00	0.90
	100	7.70E-01	—	6.80E+00	—	6.93E+01	—	2.11E-01	—	1.73E+01	—
	200	5.76E-01	0.42	5.21E+00	0.39	4.23E+01	0.71	9.28E-02	1.19	8.83E+00	0.97
P4	400	3.93E-01	0.55	3.46E+00	0.59	2.47E+01	0.77	5.32E-02	0.80	4.42E+00	1.00
	800	2.69E-01	0.55	2.39E+00	0.53	1.58E+01	0.65	2.47E-02	1.10	2.19E+00	1.01
	1600	1.91E-01	0.50	1.67E+00	0.52	9.99E+00	0.66	1.41E-02	0.81	1.29E+00	0.76
	100	3.90E-02	—	6.47E-02	—	1.53E-01	—	3.04E-02	—	3.35E-02	—
	200	3.00E-02	0.38	4.93E-02	0.39	8.37E-02	0.87	1.82E-02	0.74	2.10E-02	0.68
P5	400	2.03E-02	0.56	3.26E-02	0.60	5.19E-02	0.69	1.00E-02	0.86	1.16E-02	0.86
	800	1.41E-02	0.53	2.26E-02	0.53	3.52E-02	0.56	5.90E-03	0.77	6.95E-03	0.73
	1600	9.80E-03	0.53	1.54E-02	0.55	2.17E-02	0.70	3.25E-03	0.86	3.82E-03	0.86

The temporal error norms and the orders of accuracy of the conserved and primitive variables for the above test cases using the second-order-in-time IMPICE method are shown in Table 6.4. We use a highly resolved solution as the exact solution $\mathbf{U_j}\left[T; t_0, U_j^0\right]$ by setting $C_{cfl} = 0.0001$ when calculating temporal errors. It is shown in Table 6.4 that the time integration accuracy for both conserved and primitive variables is very close to second-order.

In doing so, we note that this extrapolated method corresponds to the Runge-Kutta method whose positivity properties are described by Mehdizadeh Khalsaraei [87].

6.8 Higher-order Advection

The solutions with first-order accuracy of advection where advected quantities obtained from (6.18) are highly smeared at contact discontinuities. We have improved the spatial error accuracy of the IMPICE method by using a higher-order advection method. A higher-order Van Leer advection method is discussed in VanderHeyden and Kashiwa [127], in which the compatible fluxes are also derived for this type of advection method. There are several minor differences between the derivation of the face-centered advected quantity for IMPICE in this section and [127]. A higher-order advection scheme for IMPICE is derived based on a higher-order approximation of the advected quantities in (6.16). This is done by assuming that $\mathbf{U}(x_{j+\frac{1}{2}}, t)$ in Equation (6.16) is not a constant for the time step $[t_n, t_{n+1}]$. The advection equations of conserved variables in the Eulerian phase in Section 6.4.2 are given by:

$$\mathbf{U}_t + (u\mathbf{U})_x = 0. \tag{6.68}$$

In order to determine $\mathbf{U}(x_{j+\frac{1}{2}}, t)$, we will use Equation (6.68) and the constructed values, $\mathbf{W}_j^n(x,t)$, of primitive variables in the control volume $[x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}] \times [t_n, t_{n+1}]$. Within this control volume, the constructed values, $\mathbf{W}_j^n(x,t)$, are obtained by using Taylor series:

$$\mathbf{W}_j^n(x,t) = \mathbf{W}_j^n + (x - x_j)\left(\frac{\partial \mathbf{W}}{\partial x}\right)_j^n + (t - t_n)\left(\frac{\partial \mathbf{W}}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2). \tag{6.69}$$

In VanderHeyden and Kashiwa [127], the first-order term in time in the above Taylor series expansion is omitted. The extrapolated values at cell boundaries obtained by using the constructed values, $\mathbf{W}_j^n(x,t)$, are:

$$\begin{aligned}
\mathbf{W}_j^n(x_{j-\frac{1}{2}}, t) &= \mathbf{W}_j^n - \frac{\Delta x}{2}\left(\frac{\partial \mathbf{W}}{\partial x}\right)_j^n + (t - t_n)\left(\frac{\partial \mathbf{W}}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2), && (6.70)\\
\mathbf{W}_j^n(x_{j+\frac{1}{2}}, t) &= \mathbf{W}_j^n + \frac{\Delta x}{2}\left(\frac{\partial \mathbf{W}}{\partial x}\right)_j^n + (t - t_n)\left(\frac{\partial \mathbf{W}}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2). && (6.71)
\end{aligned}$$

Table 6.4. Temporal Error using the second-order-in-time IMPICE: L_1-norms and the order of accuracy n of the conserved and primitive variables for the test cases in Table 6.1 using N=200 (cells). The exact solutions $\mathbf{U}_j\left[T_e; t_0, U_j^0\right]$ for the discretized problems of these test cases are obtained by using $C_{cfl} = 0.0001$.

		$\mathbf{et}^{\rho}(T_e)$		$\mathbf{et}^{\rho u}(T_e)$		$\mathbf{et}^{\rho E}(T_e)$		$\mathbf{et}^{u}(T_e)$		$\mathbf{et}^{p}(T_e)$	
	C_{cfl}	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n
	0.4	6.64E-005	—	5.20E-005	—	1.63E-004	—	1.19E-004	—	6.75E-005	—
P1	0.2	1.47E-005	2.17	1.16E-005	2.16	3.65E-005	2.15	2.71E-005	2.14	1.50E-005	2.17
	0.1	2.90E-006	2.35	2.39E-006	2.29	7.44E-006	2.30	5.71E-006	2.25	3.03E-006	2.31
	0.05	7.11E-007	2.03	5.98E-007	2.00	1.84E-006	2.01	1.46E-006	1.97	7.58E-007	2.00
	0.4	1.56E-005	—	2.33E-005	—	4.99E-005	—	4.10E-005	—	1.45E-005	—
P2	0.2	4.89E-006	1.67	6.24E-006	1.90	1.50E-005	1.74	1.20E-005	1.77	4.72E-006	1.62
	0.1	1.06E-006	2.21	1.49E-006	2.07	3.38E-006	2.15	2.90E-006	2.05	1.02E-006	2.21
	0.05	2.92E-007	1.86	3.84E-007	1.96	9.04E-007	1.90	7.31E-007	1.99	2.88E-007	1.83
	0.4	2.26E-004	—	4.91E-003	—	3.11E-001	—	4.96E-003	—	1.27E-001	—
P3	0.2	6.05E-005	1.90	1.27E-003	1.95	7.49E-002	2.05	1.18E-003	2.08	3.01E-002	2.07
	0.1	2.07E-005	1.55	3.82E-004	1.74	2.10E-002	1.84	3.39E-004	1.79	8.30E-003	1.86
	0.05	7.32E-006	1.50	1.19E-004	1.68	6.80E-003	1.63	1.10E-004	1.62	2.88E-003	1.53
	0.4	5.99E-003	—	3.38E-002	—	9.27E-001	—	1.88E-003	—	3.93E-001	—
P4	0.2	1.65E-003	1.86	1.08E-002	1.65	2.35E-001	1.98	6.42E-004	1.55	1.07E-001	1.88
	0.1	6.23E-004	1.41	4.25E-003	1.35	0.67E-001	1.81	2.56E-004	1.32	4.06E-002	1.40
	0.05	2.37E-004	1.40	1.55E-003	1.46	2.10E-002	1.67	0.93E-004	1.46	1.40E-002	1.54
	0.4	3.11E-005	—	5.99E-005	—	2.94E-004	—	8.45E-005	—	1.14E-004	—
P5	0.2	7.00E-006	2.15	1.44E-005	2.06	6.69E-005	2.13	2.02E-005	2.06	2.62E-005	2.13
	0.1	1.91E-006	1.87	3.72E-006	1.95	1.84E-005	1.86	5.44E-006	1.89	7.19E-006	1.87
	0.05	4.54E-007	2.08	8.77E-007	2.08	3.96E-006	2.22	1.11E-006	2.29	1.52E-006	2.25

Therefore, there are two existing extrapolated values at the cell boundary at $x_{j+\frac{1}{2}}$ for the time interval $[t_n, t_{n+1}]$. These values are denoted as $\mathbf{W}_j^n(x_{j+\frac{1}{2}}, t)$ and $\mathbf{W}_{j+1}^n(x_{j+\frac{1}{2}}, t)$, and one may be chosen for the face-centered value based on the face-centered fluxing velocity at this cell boundary. The value of the vector of primitive variables at face-center is determined using:

$$\mathbf{W}(x_{j+\frac{1}{2}}, t) = \begin{cases} \mathbf{W}_{j+1}^n(x_{j+\frac{1}{2}}, t) & \text{if } \left(u^*_{j+\frac{1}{2}} < 0\right) \\ \mathbf{W}_j^n(x_{j+\frac{1}{2}}, t) & \text{otherwise.} \end{cases} \tag{6.72}$$

Now, as the extrapolated primitive variables at the cell boundary at $x_{j+\frac{1}{2}}$ are readily available, we will show how to obtain the vector of advected quantities in (6.16). We derive the advected quantities for the case $u^*_{j+\frac{1}{2}} > 0$. The advected quantities for the case $u^*_{j+\frac{1}{2}} < 0$ are derived similarly. The vector of advected quantities $\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}}$ includes $\langle \rho \rangle^n_{j+\frac{1}{2}}$, $\langle \rho u \rangle^n_{j+\frac{1}{2}}$ and $\langle \rho E \rangle^n_{j+\frac{1}{2}}$. Equation (6.68) is rewritten as follows:

$$\frac{\partial \rho}{\partial t} = -u\frac{\partial \rho}{\partial x} - \rho\frac{\partial u}{\partial x}, \tag{6.73}$$

$$\frac{\partial u}{\partial t} = -u\frac{\partial u}{\partial x}, \tag{6.74}$$

$$\frac{\partial E}{\partial t} = -u\frac{\partial E}{\partial x}. \tag{6.75}$$

Equations (6.71) and (6.73)–(6.75) are used to derive the mass advected quantity in Equation (6.16); for the case of $u^*_{j+\frac{1}{2}} > 0$, we have:

$$\begin{aligned} \langle \rho \rangle^n_{j+\frac{1}{2}} &= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \rho(x_{j+\frac{1}{2}}, t)dt \\ &= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \left(\rho_j^n + \frac{\Delta x}{2}\left(\frac{\partial \rho}{\partial x}\right)_j^n + (t - t_n)\left(\frac{\partial \rho}{\partial t}\right)_j^n\right) dt + O(\Delta x^2, \Delta t^2) \\ &= \rho_j^n + \frac{\Delta x}{2}\left(\frac{\partial \rho}{\partial x}\right)_j^n + \frac{\Delta t}{2}\left(\frac{\partial \rho}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2) \\ &= \rho_j^n + \frac{\Delta x}{2}\left(\frac{\partial \rho}{\partial x}\right)_j^n - \frac{\Delta t}{2}\left(u_j^n\left(\frac{\partial \rho}{\partial x}\right)_j^n + \rho_j^n\left(\frac{\partial u}{\partial x}\right)_j^n\right) + O(\Delta x^2, \Delta t^2). \end{aligned}$$

Therefore:

$$\langle \rho \rangle^n_{j+\frac{1}{2}} = \rho_j^n + \left(\frac{\Delta x}{2} - u_j^n\frac{\Delta t}{2}\right)\left(\frac{\partial \rho}{\partial x}\right)_j^n - \frac{\Delta t}{2}\rho_j^n\left(\frac{\partial u}{\partial x}\right)_j^n + O(\Delta x^2, \Delta t^2). \tag{6.76}$$

$$\begin{aligned}
\rho u(x_{j+\frac{1}{2}}, t) &= \left(\rho_j^n + \frac{\Delta x}{2}\left(\frac{\partial \rho}{\partial x}\right)_j^n + (t-t_n)\left(\frac{\partial \rho}{\partial t}\right)_j^n\right) \\
&\quad \times \left(u_j^n + \frac{\Delta x}{2}\left(\frac{\partial u}{\partial x}\right)_j^n + (t-t_n)\left(\frac{\partial u}{\partial t}\right)_j^n\right) + O(\Delta x^2, \Delta t^2) \\
&= \left(\rho_j^n + \frac{\Delta x}{2}\left(\frac{\partial \rho}{\partial x}\right)_j^n + (t-t_n)\left(\frac{\partial \rho}{\partial t}\right)_j^n\right) u_j^n \\
&\quad + \rho_j^n \left(\frac{\Delta x}{2}\left(\frac{\partial u}{\partial x}\right)_j^n + (t-t_n)\left(\frac{\partial u}{\partial t}\right)_j^n\right) \\
&\quad + \left(\frac{\Delta x}{2}\left(\frac{\partial \rho}{\partial x}\right)_j^n + (t-t_n)\left(\frac{\partial \rho}{\partial t}\right)_j^n\right)\left(\frac{\Delta x}{2}\left(\frac{\partial u}{\partial x}\right)_j^n + (t-t_n)\left(\frac{\partial u}{\partial t}\right)_j^n\right) \\
&\quad + O(\Delta x^2, \Delta t^2).
\end{aligned}$$

The fluxed momentum at the face-center is then given by:

$$\begin{aligned}
\langle \rho u \rangle_{j+\frac{1}{2}}^n &= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \rho u(x_{j+\frac{1}{2}}, t)dt \\
&= \langle \rho \rangle_{j+\frac{1}{2}}^n u_j^n + \rho_j^n \left(\frac{\Delta x}{2} - u_j^n \frac{\Delta t}{2}\right)\left(\frac{\partial u}{\partial x}\right)_j^n + \frac{\Delta x}{2}\left(\frac{\partial \rho}{\partial x}\right)_j^n \left(\frac{\Delta x}{2} - u_j^n \frac{\Delta t}{2}\right)\left(\frac{\partial u}{\partial x}\right)_j^n \\
&\quad + \frac{\Delta x}{2}\frac{\Delta t}{2}\left(\frac{\partial u}{\partial x}\right)_j^n \left(\frac{\partial u}{\partial t}\right)_j^n + \frac{\Delta t^3}{3}\left(\frac{\partial u}{\partial t}\right)_j^n \left(\frac{\partial u}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2).
\end{aligned}$$

This gives us the approximation:

$$\langle \rho u \rangle_{j+\frac{1}{2}}^n = \langle \rho \rangle_{j+\frac{1}{2}}^n u_j^n + \rho_{j+\frac{1}{2}}^n \left(\frac{\Delta x}{2} - u_j^n \frac{\Delta t}{2}\right)\left(\frac{\partial u}{\partial x}\right)_j^n + O(\Delta x \Delta t) + O(\Delta x^2, \Delta t^2). \tag{6.77}$$

With a similar derivation, we also have:

$$\langle \rho E \rangle_{j+\frac{1}{2}}^n = \langle \rho \rangle_{j+\frac{1}{2}}^n E_j^n + \rho_{j+\frac{1}{2}}^n \left(\frac{\Delta x}{2} - u_j^n \frac{\Delta t}{2}\right)\left(\frac{\partial E}{\partial x}\right)_j^n + O(\Delta x \Delta t) + O(\Delta x^2, \Delta t^2). \tag{6.78}$$

We thus obtain second-order accuracy in space if C_{cfl} remains contant. Equations (6.76), (6.77), and (6.78) are used to calculate the face-centered fluxed quantities for the time step $[t_n, t_{n+1}]$ when the face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, is greater than 0. A set of similar equations can be easily derived for the case when the fluxing velocity is less than 0. However, when using these equations to estimate the face-centered advected quantities, we need to have numerical estimations for $\left(\frac{\partial \rho}{\partial x}\right)_j^n$, $\left(\frac{\partial u}{\partial x}\right)_j^n$, and $\left(\frac{\partial E}{\partial x}\right)_j^n$. These spatial numerical derivatives are limited to eliminate artificial extrema and preserve monotonicity [127]. In this chapter, we choose one limiter from one-parameter family of minmod limiters [54, 131],

$$\left(\frac{\partial \mathbf{W}}{\partial x}\right)_j^n = \text{minmod}(\theta\frac{\mathbf{W}_j^n - \mathbf{W}_{j-1}^n}{\Delta x}, \frac{\mathbf{W}_{j+1}^n - \mathbf{W}_{j-1}^n}{2\Delta x}, \theta\frac{\mathbf{W}_{j+1}^n - \mathbf{W}_j^n}{\Delta x}), \tag{6.79}$$

to estimate the spatial derivatives of primitive variables by setting $\theta = 1$. Note that, the minmod limiter in (6.79) is applied component-wise where the multivariable minmod limiter for a scalar quantity is defined as:

$$\text{minmod}(z_1, z_2, z_3, ...) = \begin{cases} min(z_1, z_2, z_3, ...) & \text{if } (z_j > 0 \quad \forall j) \\ max(z_1, z_2, z_3, ...) & \text{if } (z_j < 0 \quad \forall j) \\ 0 & \text{otherwise.} \end{cases} \tag{6.80}$$

The numerical solution of Shu and Osher test problem in Figure 6.7 is obtained using the second-order-in-space IMPICE method. Comparing to the numerical solution of this problem in Figure 6.6, the solution using the second-order-in-space IMPICE method is less diffusive and more accurate.

The spatial error norms and the orders of accuracy for the test cases in Table 6.1 using the second-order-in-space IMPICE method are shown in Table 6.5. When calculating the spatial errors, we use the converged numerical solutions of these test problems as described in Section 6.6.2 for the exact solutions.

The result in Table 6.5 shows that the second-order-in-space IMPICE method does reduce the spatial errors and increase the orders of accuracy in both conserved and primitive variables. However, the orders of spatial accuracy are not close to second-order as expected, but degenerate into first-order and below. The observation concurs with those of Greenough and Rider [42] in that when discontinuities are present high-order methods may not always deliver the expected advantages and may reduce their order of accuracy to first-order. In addition, Berzins [12] shows how unless there is sufficient resolution in terms of meshpoints in a front then the positivity preservation will tend to favor the use of lower order methods. We also would like to estimate the spatial error in the numerical solutions of the Shu and Osher test problem. Since the analytic solution to the Shu and Osher test problem is not readily available, a highly resolved numerical solution is used to estimate the integral term in Equation (4.27) when calculating spatial errors. The highly resolved numerical solution is generated from running the second-order-in-space IMPICE method with $N = 25,600$ (cells) and $C_{cfl} = 0.2$. The exact cell average in Equation (4.27), $\frac{1}{\Delta x}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \mathbf{U}(x, t_n)dx$, is the numerical integration obtained from the highly resolved solution while the exact solution of time integration, $\mathbf{U_j}\left[t_n; t_0, U_j^0\right]$, is the

Table 6.5. Spatial Error using the second-order-in-space IMPICE: L_1-norms and the order of accuracy m of the conserved and primitive variables for the test cases in Table 6.1. The exact solutions $\mathbf{U_j}\left[T_e; t_0, U_j^0\right]$ are the converged numerical solutions as described in Section 6.6.2.

		$\mathbf{es}^{\rho}(T_e)$		$\mathbf{es}^{\rho u}(T_e)$		$\mathbf{es}^{\rho E}(T_e)$		$\mathbf{es}^{u}(T_e)$		$\mathbf{es}^{p}(T_e)$	
	N	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m
P1	100	7.10E-003	—	5.97E-003	—	1.50E-002	—	1.25E-002	—	5.72E-003	—
	200	4.10E-003	0.79	3.51E-003	0.77	7.37E-003	1.03	6.02E-003	1.05	2.81E-003	1.03
	400	2.23E-003	0.88	1.92E-003	0.87	3.70E-003	0.99	2.85E-003	1.08	1.36E-003	1.05
	800	1.21E-003	0.87	1.05E-003	0.87	1.87E-003	0.99	1.31E-003	1.12	6.50E-004	1.07
	1600	7.21E-004	0.75	6.13E-004	0.78	9.33E-004	1.00	5.83E-004	1.17	3.14E-004	1.05
P2	100	9.92E-003	—	9.39E-003	—	2.07E-002	—	1.50E-002	—	7.39E-003	—
	200	5.49E-003	0.85	5.34E-003	0.81	1.14E-002	0.86	7.28E-003	1.04	3.72E-003	0.99
	400	3.39E-003	0.69	3.48E-003	0.62	6.18E-003	0.88	3.51E-003	1.05	1.85E-003	1.00
	800	1.94E-003	0.80	2.01E-003	0.79	3.25E-003	0.93	1.63E-003	1.11	9.11E-004	1.03
	1600	1.15E-003	0.75	1.26E-003	0.68	1.94E-003	0.75	9.91E-004	0.71	5.15E-004	0.82
P3	100	8.56E-002	—	1.71E+000	—	5.34E+001	—	3.38E-001	—	8.03E+000	—
	200	6.35E-002	0.43	1.32E+000	0.37	2.70E+001	0.98	2.36E-001	0.52	4.47E+000	0.85
	400	4.01E-002	0.66	8.39E-001	0.65	1.83E+001	0.56	1.36E-001	0.79	2.64E+000	0.76
	800	2.57E-002	0.64	5.05E-001	0.73	1.08E+001	0.76	6.97E-002	0.97	1.39E+000	0.92
	1600	1.76E-002	0.54	3.47E-001	0.54	4.66E+000	1.21	3.50E-002	0.99	7.01E-001	0.99
P4	100	3.57E-001	—	3.61E+000	—	4.97E+001	—	1.17E-001	—	1.33E+001	—
	200	2.80E-001	0.35	2.89E+000	0.32	3.62E+001	0.45	7.85E-002	0.57	9.76E+000	0.45
	400	1.74E-001	0.69	1.83E+000	0.66	2.25E+001	0.69	4.12E-002	0.93	5.91E+000	0.72
	800	1.11E-001	0.64	1.22E+000	0.58	1.67E+001	0.43	2.87E-002	0.52	4.27E+000	0.47
	1600	7.12E-002	0.64	7.97E-001	0.61	1.01E+001	0.72	1.81E-002	0.66	2.62E+000	0.70
P5	100	1.63E-002	—	2.78E-002	—	9.54E-002	—	1.90E-002	—	1.99E-002	—
	200	1.25E-002	0.38	2.15E-002	0.37	4.14E-002	1.20	1.02E-002	0.89	1.08E-002	0.88
	400	7.55E-003	0.73	1.23E-002	0.80	2.30E-002	0.85	4.82E-003	1.09	5.10E-003	1.09
	800	4.63E-003	0.70	7.66E-003	0.69	1.48E-002	0.63	2.55E-003	0.92	2.72E-003	0.91
	1600	2.86E-003	0.70	4.56E-003	0.75	8.10E-003	0.87	1.20E-003	1.08	1.28E-003	1.08

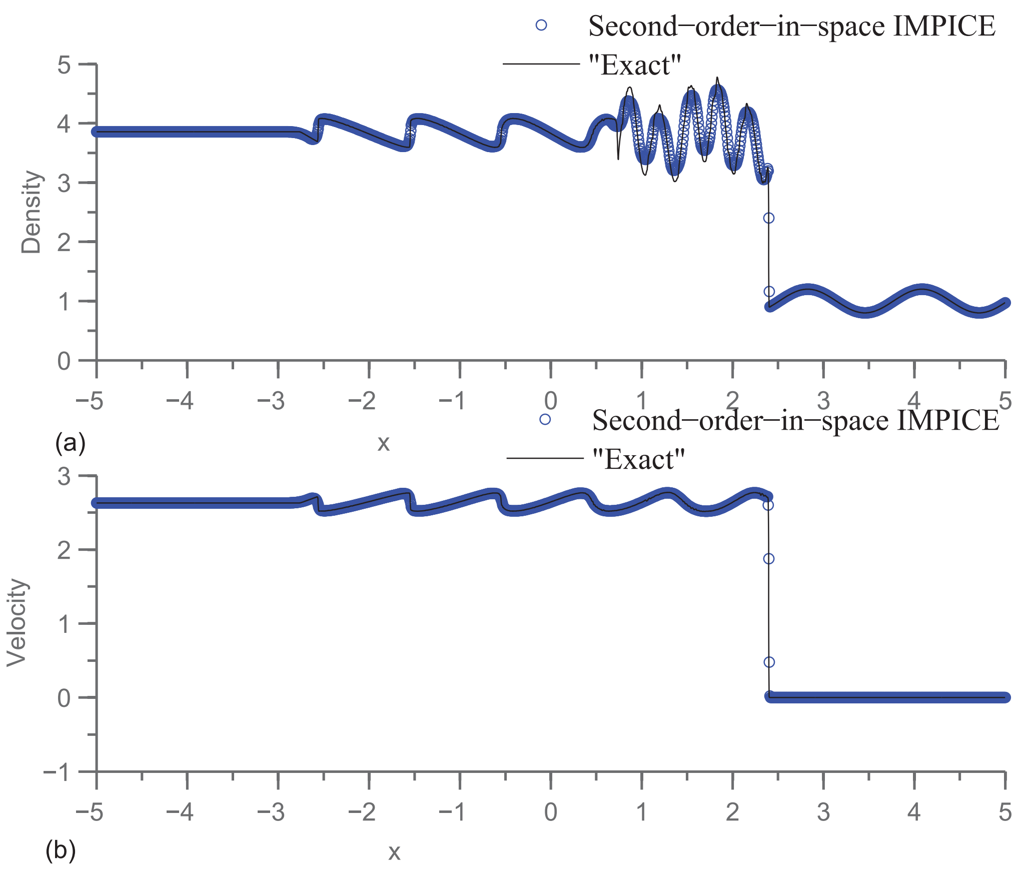


Figure 6.7. The second-order-in-space IMPICE numerical solution for Shu and Osher test problem with N=1600 (cells) and $C_{cfl} = 0.2$: (a) density and (b) velocity.

converged numerical solutions as discussed earlier in Section 6.2.2. The spatial error norms and the orders of accuracy of the Shu and Osher test problem are shown in Table 6.6.

As shown in Table 6.6, there is also a degeneration in the orders of accuracy for the result of the Shu and Osher test problem when the mesh size N is below 1600 and an improvement in its orders when the mesh size N is above 1600. This result is consistent with that of Greenough and Rider [42]. The numerical results of the second-order-in-space IMPICE for inviscid and viscous Burgers' problem are included in Appendices A.1 and A.2. The spatial error norms and the orders of accuracy for the numerical solutions of the inviscid Burgers' problem obtained from using the second-order-in-space IMPICE method included in Appendix A.1 show that the orders of convergence are not very close to second-order. However, the orders of convergence for the numerical solutions of the viscous Burgers' problem using the second-order-in-space IMPICE method included in Appendix A.2 are close to second-order for the cases of $\epsilon = 0.05$ and $\epsilon = 0.01$ as the solutions for these cases are smooth as shown in Appendix A.2. But the orders degenerate into first-order for the case of $\epsilon = 0.0001$. This is due to the development of the smooth steep front that appears close to a discontinuity in the solution of the viscous

Table 6.6. Spatial Error using the second-order-in-space IMPICE: L_1-norms and the order of accuracy m of the conserved and primitive variables for Shu and Osher test problem. The exact solutions $\mathbf{U_j}\left[T_e; t_0, U_j^0\right]$ are the converged numerical solutions.

	$\mathbf{es}^{\rho}(T_e)$		$\mathbf{es}^{\rho u}(T_e)$		$\mathbf{es}^{\rho E}(T_e)$		$\mathbf{es}^{u}(T_e)$		$\mathbf{es}^{p}(T_e)$	
N	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m
200	9.41E-01	—	1.95E+00	—	4.85E+00	—	2.45E-01	—	1.44E+00	—
400	7.30E-01	0.37	1.65E+00	0.25	3.28E+00	0.57	1.19E-01	1.05	7.03E-01	1.04
800	4.60E-01	0.67	1.08E+00	0.61	1.99E+00	0.72	6.04E-02	0.98	3.42E-01	1.04
1600	2.03E-01	1.18	4.79E-01	1.18	8.95E-01	1.15	2.85E-02	1.08	1.58E-01	1.11
3200	8.12E-02	1.32	1.87E-01	1.36	3.50E-01	1.36	1.22E-02	1.23	6.71E-02	1.24

Burgers' problem when the viscosity ϵ becomes small.

6.9 Summary

We have presented IMPICE, an improved Production ICE method, that uses a conservative scheme, slope limiters and a simple approximate Riemann solver to improve and eliminate existing oscillations to numerical solutions of Production ICE which is currently implemented in the UCF to simulate fluid flows. We have also examined how each of these different implementations individually impacts the overall numerical solutions of IMPICE. The IMPICE method with a linear spatial and temporal discretization is expected to be first-order accuracy in time and space. However, for the cases with existing discontinuities in their solutions, the order of accuracy in space is less than one as shown in Section 6.6. As it is important to have the method of higher-order of accuracy in both time and space, we have presented the nonlinear spatial and temporal discretization of the IMPICE method. These are the method of temporal extrapolation and the higher-order advection. While the method of temporal extrapolation successfully raises the order of accuracy to second-order-in-time, a less-than-expected order of accuracy in space is obtained from using the higher-order advection for the problems with discontinuities.

It has shown that the IMPICE method is capable of capturing shocks and contact surfaces. The higher-order IMPICE method is even able to capture the detailed features and structures of the flow with shock-turbulence interaction in Shu-Osher problem.

CHAPTER 7

THE IMPROVED PRODUCTION IMPLICIT CONTINUOUS-FLUID EULERIAN METHOD FOR COMPRESSIBLE FLOW PROBLEMS IN MULTIDIMENSIONAL SPACE AND ITS EMBEDDED BOUNDARY TREATMENT

In this chapter, we extend the one-dimensional IMPICE method to solve multidimensional nonlinear systems of conservation laws and particularly consider the problem of multidimensional compressible Euler equations of gas dynamics which plays an important role in mechanics and physics. The multidimensional IMPICE method is a finite-volume solver on a regular Cartesian meshes. We present numerical results obtained using IMPICE with first and second order of spatial accuracy to the compressible flow problems governed by the system of Euler equations in multidimensional space.

The use of Cartesian grids in IMPICE has the advantage of ease of grid generation for simple regular geometries, but has the disadvantage of being unable to deal with complex geometries. In order to allow IMPICE solve complex geometries, we implement the method of cut cells to handle the case when the computational boundary is not aligned with the cell edges. Small cut cells on the embedded boundary cause time step restriction as the time step is proportional to the size of a grid cell. We discuss in this chapter a new variation of the cell merging technique used in IMPICE for merging cells to prevent time step restriction. This new variation of the cell merging technique makes use of the magnitudes of surrounding face-centered fluxing velocities and is described in Section 7.5.2.5.

The content of the chapter is organized as follows. Section 7.1 describes the spatial and temporal discretization of the method. In Section 7.2, we discuss the dimensional-split Riemann problem and its HLL approximate solver. In Section 7.3, a detailed description of the multidimensional IMPICE method is given. In Section 7.4, we discuss how to increase the order of accuracy in space to second order. The implementation of boundary conditions is described in Section 7.5. For boundary conditions, we discuss the implementation of the Euler Characteristic Boundary Condition and the embedded boundary technique. In Section 7.6, we present the numerical results to a suite of test problems for the Euler equations which includes the widely used double Mach reflection problem for testing the implementation of embedded boundary. In Section 7.7, we assess the accuracy of the embedded boundary implementation by investigating the convergent rate of the numerical solutions to the advection problem on a bounded domain. Conclusions are drawn in Section 7.8.

7.1 Spatial Discretization, CFL Condition and Adaptive Time Step

7.1.1 Spatial Discretization and Notations

We consider a Cartesian mesh in the computational domain $\Omega = [a_1, b_1] \times [a_2, b_2] \times ... \times [a_d, b_d] \subset \mathbb{R}^d$ in which a uniform spatial mesh divides the computational domain into $N_1 \times N_2 \times ... \times N_d$ equal cells where N_i is the number of cells in x_i-dimension. The cell width in the i^{th}-dimension is then $\Delta x_i = \frac{(b_i - a_i)}{N_i}$.

In each cell j, the state variables are located at the centroid, $\mathbf{x}_j$, of the control volume, V_j, and represent the cell average values. The variables which represent the average values of the cell j at time t_n consist of the cell-centered density ρ_j^n, the cell-centered velocity $\mathbf{u}_j^n$, the cell-centered total energy per unit mass E_j^n, the cell-centered pressure p_j^n, and the speed of sound c_j^n. The time integration method in the ICE method calculates the cell average values at discrete time levels $t_1,\ t_2,\ t_3,\ ...,\ t_N$ from the initial cell average values at $t_0 = 0$. The cell average values at next time step are obtained from evaluating the changes in cell mass, momentum, and energy via cell boundaries.

The cell boundaries are usually referred to as faces and the variables that are associated with faces are referred to as face-centered variables. While the subscript j is used with cell-centered variables, the subscript $j + \frac{1}{2}$ is used in connection with face-centered variables.

7.1.2 CFL Condition and Adaptive Time Step

For the multidimensional system of Euler equations, the time step Δt is normally chosen to satisfy the condition:

$$\Delta t = C_{cfl} \times \min_{i=1}^{d} \left(\frac{\Delta x_i}{S_{max}^{n,i}} \right), \tag{7.1}$$

where C_{cfl} is a Courant or CFL coefficient satisfying $0 < C_{cfl} < 1$ and $S_{max}^{n,i}$ is the largest wave speed present in the domain at time t_n in x_i-dimension. A reliable estimate for the largest wave speed $S^{n,i}_{max}$ has been known to be a critical part of maintaining the method stability. A simple choice of $S^{n,i}_{max}$ is given by:

$$S^{n,i}_{max} = \max_{j \in I_c} \Big(|(u_i)^n_j| + c^n_j \Big). \tag{7.2}$$

In the IMPICE method for the one-dimensional sytem of Euler equations in Chapter 6, the method which was proposed by Kwatra *et al.* [72] for alleviating the stringent CFL condition imposed by the sound speed is used to determine the maximum speed. Though the method by Kwatra *et al.* [72] is said to be well suited to semi-implicit solvers where only the advection step is the implicit part, it appears not to be a good choice for the IMPICE method when it is used for determining the time steps for some additional test cases to the test cases in Chapter 6 where instabilities develop for even small CFL numbers. For this reason, Equation (7.2) is used to determine the maximum wave speeds in the IMPICE method for the multidimensional system of Euler equations.

7.2 The HLL Solver for x_k-split Riemann Problem

The system of Euler equations in (3.1)–(3.3) can be written in the differential form as follows:

$$\frac{\partial}{\partial t}\mathbf{U}(\mathbf{x},t) + \sum_{i=1}^{d} \frac{\partial}{\partial x_i}\mathbf{F}_i(\mathbf{U}(\mathbf{x},t)) = 0, \tag{7.3}$$

where $\mathbf{U}$ is the vector of conserved variables. Consider the special Initial Value Problem (IVP) for the above system in which the initial data consist of two constant states separated by the coordinate plane in the x_k-direction. The IVP is given by:

$$\left.\begin{array}{lcl} \mathbf{U}_t + \dfrac{\partial}{\partial x_k}\mathbf{F}_k(\mathbf{U}(\mathbf{x},t)) & = & 0, \\ \mathbf{U}(\mathbf{x},0) & = & \begin{cases} \mathbf{U}_L & \text{if } \ (x_k < 0), \\ \mathbf{U}_R & \text{if } \ (x_k > 0). \end{cases} \end{array}\right\} \tag{7.4}$$

The solution to the above IVP problem (also known as the x_k-split Riemann problem) is extremely complicated. To many numerical methods in which the solution to the Riemann problem is involved in some parts of the numerical procedure, the approximate solution of the Riemann problem is often used. As mentioned in Toro [120], the approximate Riemann solvers resulting from the combination of the HLL (Harten, Lax, and van Leer) approach in [53] and several different wave speed estimates – such as the estimates by Davis [28], Einfeldt [33], and

Roe [103] – form the bases of efficient and robust approximate Godunov-type methods. The HLL Riemann solver using the wave speed estimate by Davis [28] is simple, and it is also used in the one-dimensional IMPICE method for compressible flow problems. This approximate Riemann solver is also used in the multidimensional IMPICE method with the method description in Section 7.3. The HLL Riemann solver [53] for the x_k-split Riemann problem is given by:

$$\mathbf{U}(x_k/t;\mathbf{U}_L,\mathbf{U}_R) = \begin{cases} \mathbf{U}_L & \text{if} \quad (x_k/t \leq a_L)\,, \\ \dfrac{a_R\mathbf{U}_R - a_L\mathbf{U}_L}{a_R - a_L} - \dfrac{\mathbf{F}(\mathbf{U}_R) - \mathbf{F}(\mathbf{U}_L)}{a_R - a_L} & \text{if} \quad (a_L \leq x_k/t \leq a_R)\,, \\ \mathbf{U}_R & \text{if} \quad (a_R \leq x_k/t)\,, \end{cases} \tag{7.5}$$

where a_R and a_L are wave speeds. In Davis [28], the wave speeds are estimated as follows:

$$a_L = (u_k)_L - c_L, \quad a_R = (u_k)_R + c_R, \tag{7.6}$$

where c_L, $(u_k)_L$ respectively are the wave speed and velocity in x_k-direction obtained from $\mathbf{U}_L$ while the x_k-directional velocity $(u_k)_R$ and wave speed c_R are obtained from $\mathbf{U}_R$.

For the case when two constant states $\mathbf{U}_L$ and $\mathbf{U}_R$ are separated by a plane that is not one of the coordinate planes, we first need to rotate the coordinates so that the separating plane becomes a coordinate plane. We then solve the split Riemann problem in the new coordinate. Let two constant states $\mathbf{U}_L$ and $\mathbf{U}_R$ be separated by a plane whose normal vector is $\mathbf{n}$ which points from $\mathbf{U}_L$ to $\mathbf{U}_R$. Furthermore, let $\mathbf{R} = (r_{m,l})$ be the rotation matrix such that $\mathbf{n}_k = \mathbf{R}\mathbf{n}$ where $\mathbf{n}_k$ is the normal vector in the positive direction of the coordinate plane $x_k = 0$. With this rotation matrix, we obtain a new coordinate system $(\boldsymbol{\xi}, t)$ from the original coordinate system $(\mathbf{x}, t)$. The ξ_k-split Riemann problem in new coordinate is then given by:

$$\left.\begin{array}{lcl} \hat{\mathbf{U}}_t + \frac{\partial}{\partial \xi_k}\hat{\mathbf{F}}_k(\hat{\mathbf{U}}(\xi,t)) & = & 0, \\ \hat{\mathbf{U}}(\xi,0) & = & \begin{cases} \hat{\mathbf{U}}_L & \text{if} \quad (\xi_k < 0), \\ \hat{\mathbf{U}}_R & \text{if} \quad (\xi_k > 0), \end{cases} \end{array}\right\} \tag{7.7}$$

where $\hat{\mathbf{U}} = [\rho, \rho\mathbf{R}\mathbf{u}, \rho E]$ and:

$$\hat{\mathbf{F}}_k = \sum_{i=1}^{d} r_{k,i}\mathbf{F}_i. \tag{7.8}$$

From solving the ξ_k-split Riemann problem in (ξ, t) coordinates using the HLL Riemann solver, we obtain the approximate Riemann solution $\hat{\mathbf{U}}(\xi_k/t; \hat{\mathbf{U}}_L, \hat{\mathbf{U}}_R)$. The solution to the problem of two constant states that are separated by the plane whose normal vector is $\mathbf{n}$ is then given by:

$$\mathbf{U}(x_k/t; \mathbf{U}_L, \mathbf{U}_R) = \mathbf{R}^{-1}\hat{\mathbf{U}}(\xi_k/t; \hat{\mathbf{U}}_L, \hat{\mathbf{U}}_R). \tag{7.9}$$

The use of the above solver in the IMPICE method for multidimensional compressible flow problems is discussed in detail in Section 7.3.

7.3 Method Description

Assume that the cell-centered state variables are available at time t_n, the following steps are used to obtain the cell-centered state variables at $t_{n+1} = t_n + \Delta t$:

7.3.1 The Primary Phase

The exchanges of mass, momentum, and energy among cells at the cell surfaces are approximated based on the rates of volume fluxes at cell boundaries. So it is important to determine the face-centered fluxing velocity for all the faces. The face-centered fluxing velocity for the multidimensional IMPICE method is determined using a similar approach for the one-dimensional IMPICE method. In this approach, the Riemann problem is constructed using the face-centered extrapolated values of the left and right cell-centered data.

Consider evaluating the face-centered fluxing velocity of the face in x_k-direction that separates the cells located at $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$. This face's center is located at $\mathbf{x}_{j+\frac{1}{2}}$ and its face-centered fluxing velocity is denoted as $\mathbf{u}^*_{j+\frac{1}{2}}$. Let cell $j-1$ be the left cell of j and cell $j+2$ be the right cell of $j+1$ in x_k-direction, in order to apply the limiting process using higher order polynomial interpolation of Kim and Kim [71] the left and right extrapolated values at $\mathbf{x}_{j+\frac{1}{2}}$ are presented in different forms from (6.44) and (6.45) as follows:

$$\mathbf{W}^{n(L)}_{j+\frac{1}{2}} = \mathbf{W}^n_j \quad + 0.5\boldsymbol{\phi}\left(\mathbf{r}^{n(L)}_{j+\frac{1}{2}}\right)(\mathbf{W}^n_j - \mathbf{W}^n_{j-1}), \tag{7.10}$$

$$\mathbf{W}^{n(R)}_{j+\frac{1}{2}} = \mathbf{W}^n_{j+1} - 0.5\boldsymbol{\phi}\left(\mathbf{r}^{n(R)}_{j+\frac{1}{2}}\right)(\mathbf{W}^n_{j+2} - \mathbf{W}^n_{j+1}), \tag{7.11}$$

where:

$$\mathbf{r}^{n(L)}_{j+\frac{1}{2}} = \frac{\mathbf{W}^n_{j+1} - \mathbf{W}^n_j}{\mathbf{W}^n_j - \mathbf{W}^n_{j-1}}, \quad \mathbf{r}^{n(R)}_{j+\frac{1}{2}} = \frac{\mathbf{W}^n_{j+1} - \mathbf{W}^n_j}{\mathbf{W}^n_{j+2} - \mathbf{W}^n_{j+1}}, \tag{7.12}$$

and $\boldsymbol{\phi}$ is a limiting function which limits the local gradient of primitive variables to obtain monotonic condition. Note that all vector divisions and multiplications in this section are the component-wise operations. In order to control oscillations near shock discontinuity in multidimensional space, Kim and Kim [71] developed the multidimensional limiting process (MLP) which combines the multidimensional limiting function with a higher order polynomial interpolation. A family of interpolation schemes with different interpolation orders is used in MLP to control oscillations. We will use the MLP scheme with third order interpolation (MLP3) to limit the local gradient. In MLP3, the left and right extrapolated values in (7.10) and (7.11) are given in different forms as follows:

$$\mathbf{W}^{n(L)}_{j+\frac{1}{2}} = \mathbf{W}^n_j \quad + 0.5\boldsymbol{\phi}\left(\mathbf{r}^n_{j+\frac{1}{2}}(L), \boldsymbol{\alpha}^{n(L)}_{j+\frac{1}{2}}\right)(\mathbf{W}^n_j - \mathbf{W}^n_{j-1}), \tag{7.13}$$

$$\mathbf{W}^{n(R)}_{j+\frac{1}{2}} = \mathbf{W}^n_{j+1} - 0.5\boldsymbol{\phi}\left(\mathbf{r}^n_{j+\frac{1}{2}}(R), \boldsymbol{\alpha}^{n(R)}_{j+\frac{1}{2}}\right)(\mathbf{W}^n_{j+2} - \mathbf{W}^n_{j+1}), \tag{7.14}$$

where $\boldsymbol{\phi}(\mathbf{r}, \boldsymbol{\alpha}) = max(\mathbf{0}, min(\boldsymbol{\alpha}\mathbf{r}, \boldsymbol{\alpha}, \frac{1+2\mathbf{r}}{3}))$; the values of $\mathbf{r}^n_{j+\frac{1}{2}}(L)$ and $\mathbf{r}^n_{j+\frac{1}{2}}(R)$ are defined in (7.12); and the values of $\boldsymbol{\alpha}^{n(L)}_{j+\frac{1}{2}}$ and $\boldsymbol{\alpha}^{n(R)}_{j+\frac{1}{2}}$ are summarized as follows:

$$\boldsymbol{\alpha}^{n(L)}_{j+\frac{1}{2}} = \mathbf{g}\left[\frac{2max(1, \mathbf{r}^{n(L)}_{j+\frac{1}{2}})\left(1 + max(0, \frac{tan\theta^k_{j+1}}{\mathbf{r}^{n(R)}_{j+\frac{1}{2}}}\right)}{1 + tan\theta^k_j}\right], \tag{7.15}$$

$$\boldsymbol{\alpha}^{n(R)}_{j+\frac{1}{2}} = \mathbf{g}\left[\frac{2max(1, \mathbf{r}^{n(R)}_{j+\frac{1}{2}})\left(1 + max(0, \frac{tan\theta^k_{j}}{\mathbf{r}^{n(L)}_{j+\frac{1}{2}}}\right)}{1 + tan\theta^k_{j+1}}\right], \tag{7.16}$$

where $\mathbf{g}(\mathbf{x}) = max(\mathbf{1}, min(\mathbf{2}, \mathbf{x}))$ and the calculation of $tan\theta^k_j$ is dependent on the direction x_k which is defined as follows. Let $\Delta\mathbf{W}^k_j = (\mathbf{W}^n_{j+1} - \mathbf{W}^n_{j-1})$ where $j-1$ and $j+1$ are left and right cell of j in x_k-direction, $tan\theta^k_j$ is then given by:

$$tan\theta^0_j = \left|\frac{\Delta\mathbf{W}^0_j}{\Delta\mathbf{W}^1_j}\right|, \quad tan\theta^1_j = \left|\frac{\Delta\mathbf{W}^1_j}{\Delta\mathbf{W}^0_j}\right|. \tag{7.17}$$

After the face-centered extrapolated values of primitive variables, $\mathbf{W}^{n(L)}_{j+\frac{1}{2}}$ and $\mathbf{W}^{n(R)}_{j+\frac{1}{2}}$, have been obtained using (7.13) and (7.14), the left and right extrapolated conserved variables $\mathbf{U}^{n(L)}_{j+\frac{1}{2}}$ and $\mathbf{U}^{n(R)}_{j+\frac{1}{2}}$ are also obtained. Consider the x_k-split Generalized Riemann Problem (GRP) in which two constant states are separated by the face centered at $\mathbf{x}_{j+\frac{1}{2}}$ as given:

$$\left.\begin{aligned} \mathbf{U}_t + \frac{\partial}{\partial x_k}\mathbf{F}_k(\mathbf{U}(\mathbf{x},t)) &= 0, \\ \mathbf{U}(\mathbf{x},t_n) &= \begin{cases} \mathbf{U}^{n(L)}_{j+\frac{1}{2}} & \text{if } x_k < (\mathbf{x}_{j+\frac{1}{2}})_k, \\ \mathbf{U}^{n(R)}_{j+\frac{1}{2}} & \text{if } x_k > (\mathbf{x}_{j+\frac{1}{2}})_k, \end{cases} \end{aligned}\right\}. \tag{7.18}$$

As usual, the exact solution of the above GRP is potentially complicated, but an approximate solution may serve the purpose of evaluating the face-centered fluxing velocity. The approximate solution of the above GRP is obtained by solving the conventional Riemann Problem with piecewise constant data in the new coordinate (ξ, τ) where $\xi = \mathbf{x} - \mathbf{x}_{j+\frac{1}{2}}$ and $\tau = t - t_n$ given by:

$$\left.\begin{aligned} \mathbf{U}_t + \frac{\partial}{\partial \xi_k}\mathbf{F}(\mathbf{U}(\xi,\tau)) &= 0, \\ \mathbf{U}(\xi,0) &= \begin{cases} \mathbf{U}^{n(L)}_{j+\frac{1}{2}} & \text{if } (\xi_k < 0), \\ \mathbf{U}^{n(R)}_{j+\frac{1}{2}} & \text{if } (\xi_k > 0), \end{cases} \end{aligned}\right\}. \tag{7.19}$$

The face-centered conserved variables at t_n, $\mathbf{U}^n_{j+\frac{1}{2}}$, are defined as the value at the origin of new coordinate immediately after the interaction of the piece-wise contant data $\mathbf{U}^{n(L)}_{j+\frac{1}{2}}$ and $\mathbf{U}^{n(R)}_{j+\frac{1}{2}}$. The value at the origin of the new coordinate (ξ, τ) is defined as:

$$\mathbf{U}(\mathbf{0}, 0) = \lim_{\tau \to 0^+} \mathbf{U}(\mathbf{0}, \tau). \tag{7.20}$$

The HLL approximate Riemann solver discussed in Section 7.2 is used to solve the problem (7.19) and estimate the values of the conserved variables at the face-center at t_n, $\mathbf{U}^n_{j+\frac{1}{2}}$, and:

$$\mathbf{U}^n_{j+\frac{1}{2}} = \mathbf{U}\left(0; \mathbf{U}^{n(L)}_{j+\frac{1}{2}}, \mathbf{U}^{n(R)}_{j+\frac{1}{2}}\right). \tag{7.21}$$

Once $\mathbf{U}^n_{j+\frac{1}{2}}$ is approximated using (7.21), the vector of face-centered primitive variables at t_n, $\mathbf{W}^n_{j+\frac{1}{2}}$, is also known. The face-centered primitive variables used later on in this method include the face-centered density, $\rho^n_{j+\frac{1}{2}}$, the face-centered velocity, $\mathbf{u}^n_{j+\frac{1}{2}}$, and the face-centered pressure, $p^n_{j+\frac{1}{2}}$.

The face-centered approximate velocity at $t_{n+\frac{1}{2}}$, $\tilde{\mathbf{u}}^*_{j+\frac{1}{2}}$, is calculated using an explicit Euler step in the Lagrangian frame in (3.12) which is given by:

$$\tilde{\mathbf{u}}^*_{j+\frac{1}{2}} = \mathbf{u}^n_{j+\frac{1}{2}} - \frac{\Delta t}{2}\frac{\nabla p^n_{j+\frac{1}{2}}}{\rho^n_{j+\frac{1}{2}}}, \tag{7.22}$$

where $\nabla p^n_{j+\frac{1}{2}}$ is the numerical gradient of pressure at the face center which is approximated using central differences. Figure 7.1 illustrates the point stencil being used for calculating the face-centered pressure gradient. In this calculation, the i-component of gradient vector is calculated using the cell-centered pressure of left and right cells and otherwise using the face-centered pressure of left and right faces. Note that the face-centered pressure at t_n is obtained from numerically solving the GRP as discussed above. Since the pressure value at the points in the point stencil for calculating face-centered pressure gradient is at t_n, the calculated face-centered fluxing velocity in (7.22) is denoted as approximate fluxing velocity, $\tilde{\mathbf{u}}^*_{j+\frac{1}{2}}$. In Cell-centered ICE by Kashiwa *et al.* [68], the face-centered fluxing velocity $\mathbf{u}^*_{j+\frac{1}{2}}$ is calculated using a semi-implicit scheme of Equation (3.12) which is almost the same as the scheme in (7.22) except for the pressure gradient is approximated using the pressure values at $t_{n+\frac{1}{2}}$, and so their scheme is given as follows:

$$\mathbf{u}^*_{j+\frac{1}{2}} = \mathbf{u}^n_{j+\frac{1}{2}} - \frac{\Delta t}{2} \frac{\nabla p^{n+\frac{1}{2}}_{j+\frac{1}{2}}}{\rho^n_{j+\frac{1}{2}}}. \tag{7.23}$$

In order to estimate the pressure gradient in the above scheme, we need to determine the pressure values at $t_{n+\frac{1}{2}}$ at the points in the stencil used to calculate numerical pressure gradient. In other words, we need to determine the pressure at $t_{n+\frac{1}{2}}$ for all face centers and cell centers. The cell-centered pressure at $t_{n+\frac{1}{2}}$, $p^{n+\frac{1}{2}}_j$, is estimated using an explicit Euler step applied to Equation (3.13), that is:

$$p^{n+\frac{1}{2}}_j = p^n_j - \frac{\Delta t}{2} \left(c^2\rho\right)^n_j \nabla \cdot \mathbf{u}^n_j, \tag{7.24}$$

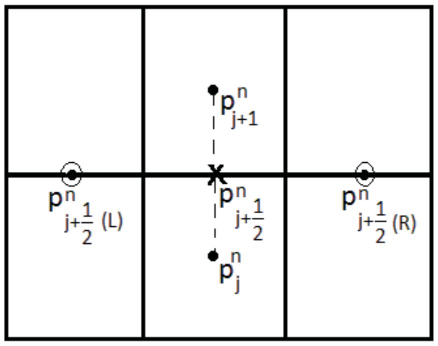


Figure 7.1. Point stencil for calculating face-centered pressure gradient $\nabla p^n_{j+\frac{1}{2}}$.

where $\nabla \cdot \mathbf{u}_j^n$ is numerical velocity divergence at cell center which is approximated using the limited cell-centered velocity gradient, $\nabla \mathbf{u}_j^n$. The face-centered pressure is then calculated by a density weighted average:

$$p_{j+\frac{1}{2}}^{n+\frac{1}{2}} = \frac{\rho_j^n p_{j+1}^{n+\frac{1}{2}} + \rho_{j+1}^n p_j^{n+\frac{1}{2}}}{\rho_j^n + \rho_{j+1}^n}. \tag{7.25}$$

The face-centered and cell-centered pressures calculated in (7.24) and (7.25) are used in the calculations of $\nabla p_{j+\frac{1}{2}}^{n+\frac{1}{2}}$ and then $\mathbf{u}_{j+\frac{1}{2}}^*$ using Equation (7.23). As mentioned above, the superscript $*$ is used to denote the face-centered variable at $t_{n+\frac{1}{2}}$, the face-centered pressure $p_{j+\frac{1}{2}}^{n+\frac{1}{2}}$ is also denoted as $p_{j+\frac{1}{2}}^*$.

7.3.2 The Lagrangian Phase

If we neglect the convective terms, the changes in cell-centered mass, momentum and energy are governed by Equations (3.21)–(3.24). The volume integrals on the right side of these equations are evaluated using the divergence theorem. Thus the changes in cell mass, momentum, and energy along a path moving with fluid velocity $\mathbf{u}$ are given by:

$$(\rho V)_j^L = (\rho V)_j^n , \tag{7.26}$$

$$(\rho \mathbf{u} V)_j^L = (\rho \mathbf{u} V)_j^n - \Delta t \sum_{k \in I_{j+\frac{1}{2}}} p_k^* S_k \mathbf{n}_k, \tag{7.27}$$

$$(\rho E V)_j^L = (\rho E V)_j^n - \Delta t \sum_{k \in I_{j+\frac{1}{2}}} p_k^* S_k \mathbf{n}_k \cdot \mathbf{u}_k^*, \tag{7.28}$$

where $I_{j+\frac{1}{2}}$ is the set of indices of faces which form the boundary of cell j, S_k is the area of face k, $\mathbf{n}_k$ is the face's outward surface normal, and V_j^L is the new cell volume which is determined using Equation (3.21), that is:

$$V_j^L = V_j^n + \Delta t \sum_{k \in I_{j+\frac{1}{2}}} S_k \mathbf{n}_k \cdot \mathbf{u}_k^* \; . \tag{7.29}$$

The new cell volume, V_j^L, is substituted into Equations (7.26) –(7.28) to determine the cell density, ρ_j^L; the cell velocity, $\mathbf{u}_j^L$; and the cell total energy per unit mass, E_j^L.

7.3.3 The Eulerian Phase

In this phase, we will take into account the convective terms which were neglected during the Lagrangian phase. The changes in solution values due to the advection of mass, momentum and energy over the step $[t_n, t_{n+1}]$ through the surrounding faces are evaluated. These changes are governed by Equations (3.25)–(3.27). In order to approximate the integrals on the right side of these equations, beside the rate of volume flux between the cells being determined in the Primary Phase, we need to determine the advected quantities per unit volume at the faces. If we assume that we are evaluating the advected quantities between the cells located at $\mathbf{x}_j$ and $\mathbf{x}_{j+1}$, the advected quantities include the face-centered advected density, $\langle \rho \rangle^n_{j+\frac{1}{2}}$, specific linear momentum, $\langle \rho \mathbf{u} \rangle^n_{j+\frac{1}{2}}$, and specific total energy, $\langle \rho E \rangle^n_{j+\frac{1}{2}}$, e.g.,

$$\langle q \rangle^n_{j+\frac{1}{2}} = \frac{1}{\Delta t} \int_{t_n}^{t_{n+1}} \frac{1}{S_{j+\frac{1}{2}}} \int_{S_{j+\frac{1}{2}}} q(\mathbf{x}, t) dS dt. \tag{7.30}$$

These quantities are determined using the following equation:

$$\langle q \rangle^n_{j+\frac{1}{2}} = \begin{cases} q^L_j & \text{if } \left(\mathbf{n}_{j+\frac{1}{2}} \cdot \mathbf{u}^*_{j+\frac{1}{2}} \right) > 0, \\ q^L_{j+1} & \text{otherwise,} \end{cases} \tag{7.31}$$

where $q = \rho$,$\rho\mathbf{u}$, or ρE, and $\mathbf{n}_{j+\frac{1}{2}}$ is the surface normal of cell j, and q^L_j is determined using Equations (7.26)–(7.29). The changes in mass, momentum and energy due to the advection are then:

$$\begin{aligned} (\rho V)^{n+1}_j &= (\rho V)^L_j - \Delta t \sum_{k \in I_{j+\frac{1}{2}}} S_k \left(\mathbf{n}_k \cdot \mathbf{u}^*_k \right) \langle \rho \rangle^n_k, & (7.32) \\ (\rho \mathbf{u} V)^{n+1}_j &= (\rho \mathbf{u} V)^L_j - \Delta t \sum_{k \in I_{j+\frac{1}{2}}} S_k \left(\mathbf{n}_k \cdot \mathbf{u}^*_k \right) \langle \rho \mathbf{u} \rangle^n_k, & (7.33) \\ (\rho E V)^{n+1}_j &= (\rho E V)^L_j - \Delta t \sum_{k \in I_{j+\frac{1}{2}}} S_k \left(\mathbf{n}_k \cdot \mathbf{u}^*_k \right) \langle \rho E \rangle^n_k. & (7.34) \end{aligned}$$

7.3.4 State Variables Update Phase

We update cell-centered pressure, p^{n+1}_j using the equation of state (3.5).

7.4 High Order Extensions

In order to achieve a high order extension in space, the higher-order advection method of Van Leer is used. The higher-order advection scheme is based on a higher-order approximation of the advected quantities in (7.30). The advection equations of conserved variables in the Eulerian phase in Section 7.3 are given by:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) &= 0, && (7.35)\\ \frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) &= 0, && (7.36)\\ \frac{\partial \rho E}{\partial t} + \nabla \cdot (\rho E \mathbf{u}) &= 0. && (7.37) \end{aligned}$$

In order to determine $q(\mathbf{x}, t)$ with $\mathbf{x} \in S_{j+\frac{1}{2}}$, we will use Equations (7.35)–(7.37) and the constructed values, $\mathbf{W}_j^n(\mathbf{x}, t)$, of primitive variables in the control volume $V_j \times [t_n, t_{n+1}]$. Within this control volume, the constructed values, $\mathbf{W}_j^n(\mathbf{x}, t)$, are obtained by using Taylor series:

$$\mathbf{W}_j^n(\mathbf{x}, t) = \mathbf{W}_j^n + \nabla \mathbf{W}_j^n(\mathbf{x} - \mathbf{x}_j) + (t - t_n)\left(\frac{\partial \mathbf{W}}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2). \qquad (7.38)$$

In order to control oscillations, $\nabla \mathbf{W}_j^n$ in the above equation is limited using limiter functions. There are many choices to limit the values of $\nabla \mathbf{W}_j^n$, but we choose to limit $\nabla \mathbf{W}_j^n$ with dimensional splitting. To limit the gradient of $\mathbf{W}_j^n$ in x_i-dimension, we apply the minmod limiter as follows:

$$\left(\frac{\partial \mathbf{W}}{\partial x_i}\right)_j^n = minmod(\theta \frac{\mathbf{W}_j^n - \mathbf{W}_{j-1}^n}{\Delta x}, \frac{\mathbf{W}_{j+1}^n - \mathbf{W}_{j-1}^n}{2\Delta x}, \theta \frac{\mathbf{W}_{j+1}^n - \mathbf{W}_j^n}{\Delta x}), \qquad (7.39)$$

where $j-1$ and $j+1$ are indices of the left and right cells in x_i-direction. The multivariable minmod limiter in (7.39) is defined by Equation (6.80).

The extrapolated values at cell boundaries obtained by using the constructed values, $\mathbf{W}_j^n(\mathbf{x}, t)$, where:

$$\begin{aligned} \mathbf{W}_j^n(\mathbf{x}, t) &= \mathbf{W}_j^n - \nabla \mathbf{W}_j^n\,(\mathbf{x}_j - \mathbf{x}) + (t - t_n)\left(\tfrac{\partial \mathbf{W}}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2),\ \mathbf{x} \in S_{j-\frac{1}{2}}, && (7.40)\\ \mathbf{W}_j^n(\mathbf{x}, t) &= \mathbf{W}_j^n + \nabla \mathbf{W}_j^n\,(\mathbf{x} - \mathbf{x}_j) + (t - t_n)\left(\tfrac{\partial \mathbf{W}}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2),\ \mathbf{x} \in S_{j+\frac{1}{2}}. && (7.41) \end{aligned}$$

Therefore, there are two existing extrapolated values at the cell boundary at $S_{j+\frac{1}{2}}$ for the time interval $[t_n, t_{n+1}]$. These values are denoted as $\mathbf{W}_j^n(\mathbf{x}, t)$ and $\mathbf{W}_{j+1}^n(\mathbf{x}, t)$ where $\mathbf{x} \in S_{j+\frac{1}{2}}$, and one may be chosen for the face-centered value based on the face-centered fluxing velocity at this cell boundary. The value of the vector of primitive variables at face-center $\mathbf{x} \in S_{j+\frac{1}{2}}$ is determined using:

$$\mathbf{W}(\mathbf{x}, t) = \begin{cases} \mathbf{W}_{j+1}^n(\mathbf{x}, t) & \text{if} \quad \left(\mathbf{n}_{j+\frac{1}{2}} \cdot \mathbf{u}^*_{j+\frac{1}{2}} < 0\right) \\ \mathbf{W}_j^n(\mathbf{x}, t) & \text{otherwise,} \end{cases} \qquad (7.42)$$

where $\mathbf{n}_{j+\frac{1}{2}}$ is the outward normal of surface $S_{j+\frac{1}{2}}$ in cell j. Now, as the extrapolated primitive variables at the cell boundary at $S_{j+\frac{1}{2}}$ are readily available, we will show how to obtain the vector of advected quantities in (7.30). We derive the advected quantities for the case $\left(\mathbf{n}_{j+\frac{1}{2}} \cdot \mathbf{u}^*_{j+\frac{1}{2}}\right) > 0$. The advected quantities for the case $\left(\mathbf{n}_{j+\frac{1}{2}} \cdot \mathbf{u}^*_{j+\frac{1}{2}}\right) < 0$ are derived similarly. The vector of advected quantities $\langle U\rangle_{j+\frac{1}{2}}$ includes $\langle \rho\rangle_{j+\frac{1}{2}}$, $\langle \rho u\rangle_{j+\frac{1}{2}}$ and $\langle \rho E\rangle_{j+\frac{1}{2}}$. Equations (7.35)–(7.37) are rewritten as follows:

$$\frac{\partial \rho}{\partial t} = -\nabla\rho\cdot\mathbf{u} - \rho\nabla\cdot\mathbf{u}, \tag{7.43}$$

$$\frac{\partial \mathbf{u}}{\partial t} = -\mathbf{u}\cdot\nabla\mathbf{u}, \tag{7.44}$$

$$\frac{\partial E}{\partial t} = -\mathbf{u}\cdot\nabla E. \tag{7.45}$$

Equations (7.41) and (7.43)–(7.45) are used to derive the mass, momentum, and energy advected quantities in Equation (7.30); for the case of $\left(\mathbf{n}_{j+\frac{1}{2}} \cdot \mathbf{u}^*_{j+\frac{1}{2}}\right) > 0$, we have:

$$\begin{aligned}
\langle\rho\rangle_{j+\frac{1}{2}} &= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \frac{1}{S_{j+\frac{1}{2}}}\int_{S_{j+\frac{1}{2}}} \rho(\mathbf{x},t)\,dSdt \\
&= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}} \frac{1}{S_{j+\frac{1}{2}}}\int_{S_{j+\frac{1}{2}}} \left(\rho_j^n + \nabla\rho_j^n(\mathbf{x}-\mathbf{x}_j) + (t-t_n)\left(\frac{\partial\rho}{\partial t}\right)_j^n\right) dSdt + O(\Delta x^2, \Delta t^2) \\
&= \rho_j^n + \nabla\rho_j^n(\mathbf{x}_{j+\frac{1}{2}}-\mathbf{x}_j) + \frac{\Delta t}{2}\left(\frac{\partial\rho}{\partial t}\right)_j^n + O(\Delta x^2, \Delta t^2) \\
&= \rho_j^n + \nabla\rho_j^n(\mathbf{x}_{j+\frac{1}{2}}-\mathbf{x}_j) - \frac{\Delta t}{2}\left(\nabla\rho_j^n\cdot\mathbf{u}_j^n + \rho_j^n\nabla\cdot\mathbf{u}_j^n\right) + O(\Delta x^2, \Delta t^2).
\end{aligned}$$

Let $\mathbf{r}_j^n = \mathbf{x}_{j+\frac{1}{2}} - \mathbf{x}_j - \frac{\Delta t}{2}\mathbf{u}_j^n$, we have:

$$\langle\rho\rangle_{j+\frac{1}{2}} = \rho_j^n + \nabla\rho_j^n\mathbf{r}_j^n - \frac{\Delta t}{2}\rho_j^n\nabla\cdot\mathbf{u}_j^n + O(\Delta x^2, \Delta t^2). \tag{7.46}$$

Let $\mathbf{r}_j(\mathbf{x}) = \mathbf{x} - \mathbf{x}_j$, we have:

$$
\begin{aligned}
\rho\mathbf{u}(\mathbf{x},t) &= \left(\rho_j^n + \nabla\rho_j^n\ \mathbf{r}_j(\mathbf{x}) + (t-t_n)\left(\frac{\partial\rho}{\partial t}\right)_j^n\right) \\
&\quad \times\left(\mathbf{u}_j^n + \nabla\mathbf{u}_j^n\ \mathbf{r}_j(\mathbf{x}) + (t-t_n)\left(\frac{\partial\mathbf{u}}{\partial t}\right)_j^n\right) + O(\Delta x^2,\Delta t^2) \\
&= \left(\rho_j^n + \nabla\rho_j^n\ \mathbf{r}_j(\mathbf{x}) + (t-t_n)\left(\frac{\partial\rho}{\partial t}\right)_j^n\right)\mathbf{u}_j^n + \rho_j^n\left(\nabla\mathbf{u}_j^n\ \mathbf{r}_j(\mathbf{x}) + (t-t_n)\left(\frac{\partial\mathbf{u}}{\partial t}\right)_j^n\right) \\
&\quad + \left(\nabla\rho_j^n\ \mathbf{r}_j(\mathbf{x}) + (t-t_n)\left(\frac{\partial\rho}{\partial t}\right)_j^n\right)\left(\nabla\mathbf{u}_j^n\ \mathbf{r}_j(\mathbf{x}) + (t-t_n)\left(\frac{\partial\mathbf{u}}{\partial t}\right)_j^n\right) \\
&\quad + O(\Delta x^2,\Delta t^2).
\end{aligned}
$$

The fluxed momentum at the face-center is then given by:

$$
\begin{aligned}
\langle\rho\mathbf{u}\rangle_{j+\frac{1}{2}} &= \frac{1}{\Delta t}\int_{t_n}^{t_{n+1}}\int_{S_{j+\frac{1}{2}}} \rho u(\mathbf{u},t)dSdt \\
&= \langle\rho\rangle_{j+\frac{1}{2}}\mathbf{u}_j^n + \rho_j^n\nabla\mathbf{u}_j^n\mathbf{r}_j^n + \left(\nabla\rho_j^n(\mathbf{x}_{j+\frac{1}{2}} - \mathbf{x}_j)\right)\left(\nabla\mathbf{u}_j^n\mathbf{r}_j^n\right) \\
&\quad + \frac{\Delta t}{2}\nabla\mathbf{u}_j^n\left(\mathbf{x}_{j+\frac{1}{2}} - \mathbf{x}_j\right)\left(\frac{\partial\mathbf{u}}{\partial t}\right)_j^n + \frac{\Delta t^3}{3}\left(\frac{\partial\mathbf{u}}{\partial t}\right)_j^n\left(\frac{\partial\mathbf{u}}{\partial t}\right)_j^n + O(\Delta x^2,\Delta t^2).
\end{aligned}
$$

This gives us the approximation:

$$
\langle\rho\mathbf{u}\rangle_{j+\frac{1}{2}} = \langle\rho\rangle_{j+\frac{1}{2}}\mathbf{u}_j^n + \rho_{j+\frac{1}{2}}^n\nabla\mathbf{u}_j^n\mathbf{r}_j^n + O(\Delta x\Delta t) + O(\Delta x^2,\Delta t^2). \tag{7.47}
$$

With a similar derivation, we also have:

$$
\langle\rho E\rangle_{j+\frac{1}{2}} = \langle\rho\rangle_{j+\frac{1}{2}}E_j^n + \rho_{j+\frac{1}{2}}^n\nabla E_j^n\mathbf{r}_j^n + O(\Delta x\Delta t) + O(\Delta x^2,\Delta t^2). \tag{7.48}
$$

We thus obtain second-order accuracy in space for problems with smooth true solution if C_{cfl} remains contant. Equations (7.46), (7.47), and (7.48) are used to calculate the face-centered fluxed quantities for the time step $[t_n, t_{n+1}]$ when the face-centered fluxing velocity in the direction of outward surface normal , $\mathbf{n}_{j+\frac{1}{2}}\mathbf{u}^*_{j+\frac{1}{2}}$, is greater than 0. A set of similar equations can be easily derived for the case when the fluxing velocity is less than 0.

7.5 Boundary Conditions

It is well-known that a good implementation of boundary conditions is important to ensure the stability of numerical methods whereas a not-well-chosen boundary condition might adversely affect the accuracy and stability of the numerical solutions. There are several approaches but the method of characteristic boundary conditions is often used; see [78] and references within. In this section, we discuss the method of characteristic boundary conditions and the method of cut cells to the multidimensional IMPICE method.

7.5.1 The Euler Characteristic Boundary Condition Implementation

The implementation of the Euler Characteristic Boundary Condition (ECBC) discussed in Section 3.3.2 involves the approximation of $\{L_j \,:\, j = 1, ..., (d+2)\}$ crossing the boundary. These wave amplitudes $\{L_j\}$ at the boundary in x_i-direction might be approximated using the characteristic analysis of the x_i-direction governing equations. The spatial derivatives in x_i-direction in Equations (3.56)-(3.58) are approximated based on the imposed physical conditions. The approximation of the spatial derivatives in each wave amplitude is based on the sign of the corresponding characteristic velocity; the corresponding characteristic velocity of wave amplitude L_j is the leading term in its description as shown in Equations (3.56)–(3.58). As mentioned in [78], the one-side difference method of the points inside the computational domain is used to estimate the outward wave amplitudes, but additional physical considerations must be made for the estimation of inward wave amplitudes. With the definitions of $\{L_j\}$ in (3.56)–(3.58), the following Local One Dimensional Inviscid (LODI) system of primitive variables is obtained from Equations (3.53)–(3.55) in which the transverse terms are neglected:

$$\frac{\partial \rho}{\partial t} = -\frac{1}{c^2}\left[L_2 + \frac{1}{2}(L_{2+i} + L_1)\right], \tag{7.49}$$

$$\frac{\partial p}{\partial t} = -\frac{1}{2}(L_{2+i} + L_1), \tag{7.50}$$

$$\frac{\partial u_k}{\partial t} = \begin{cases} -L_{2+k} & \text{if } (k \neq i) \\ -\dfrac{1}{2\rho c}\left(L_{2+i} - L_1\right) & \text{otherwise,} \end{cases} \tag{7.51}$$

for $k = 1, ..., d$. According to [78], the neglect of the transverse terms might cause numerical instabilities and numerical reflections if the derivative of the physical quantity in the transverse terms is large. In [78], it showed how to include these terms in the LODI system to avoid the problem of numerical reflections. The LODI relations in Equations (7.49)–(7.51) are used to estimate the temporal evolution of the primitive variables at the boundaries. The implementation of characteristic boundary condition for several situations of Navier-Stokes equations is discussed in [99] with the detail of how to calculate the amplitudes of characteristic waves $\{L_j\}$. Among these situations, we are interested in the cases of subsonic inflow/outflow and will discuss how to use the approach of characteristic boundary conditions for these cases in the multidimensional IMPICE method. The calculation of L_j for the cases of subsonic inflow/outflow in [99] in the x_i-direction is redescribed below.

For the case of subsonic outflow, only L_{2+i} is an inward wave at left boundary and only L_1 is an inward wave at right boundary in x_i-direction. As mentioned in [78], the amplitudes of the outward waves and estimated using interior points while the amplitudes of the inward waves are addressed using additional physical considerations. The inward wave amplitudes for the case of subsonic outflow are approximated as follows:

At the left boundary: $L_{2+i} = \alpha_p (p - p_\infty)$.

At the right boundary: $L_1 = \alpha_p (p - p_\infty)$,

where α_p is the relaxation coefficient of the pressure term.

For the case of subsonic inflow, all waves at left boundary except L_1 and all waves at right boundary except L_{2+i} are inward waves. The inward wave amplitudes are approximated as follows:

At the left boundary: $L_2 = \alpha_{p\rho}\left(\frac{p}{\rho} - \frac{p_l}{\rho_l}\right), \quad L_{2+k} = \alpha_{u_k}(u_k - (u_k)_l) \quad k = 1, ..., d.$

At the right boundary: $L_1 = \alpha_{u_i}(u_i - (u_i)_r), \ L_2 = \alpha_{p\rho}\left(\frac{p}{\rho} - \frac{p_r}{\rho_r}\right),$

$L_{2+k} = \alpha_{u_k}(u_k - (u_k)_r) \ \ k \neq i,$

where ρ_l, $\mathbf{u}_l$, and p_l are physical values imposed on left boundary; ρ_r, $\mathbf{u}_r$, and p_r are physical values imposed on right boundary; $\alpha_{p\rho}$ and α_{u_k} are the relaxation coefficients of the terms following these coefficients in the above calculations.

In the IMPICE method, the values $\{L_j\}$ are substituted into the LODI equations (7.49)–(7.51). The LODI system is used to advance the velocities and pressures at the computational boundary to obtain the face-centered fluxing velocities and pressures of the boundary faces aligned with the computational boundary. We will discuss how to handle boundary faces which are not aligned with the computational boundary next.

7.5.2 Embedded Boundary Method

A Cartesian grid approach is efficient for rectangular domain, but it is challenging to extend the IMPICE approach to the case of an embedded boundary [55]. Embedded boundary grids allow more automated grid generation procedures around complex objects, which is important especially for multidimensional problems. Embedded boundaries not aligned with cell edges cause cells that are cut. For these cut cells, there is a change in the cell boundary and the cell center. Therefore, the cell center, which is the center of cell mass, need to be recalculated. The change in cell boundary includes cut faces, like faces F_1 and F_2 in Figure 7.2, and boundary faces, like faces F_3 and F_4 in Figure 7.2. The implementation of the Lagrangian and Euler phases in Section 7.3 suggests that the face-centered velocity and pressure for the cut faces and the boundary faces need to be rederived. In this section, we will discuss how to calculate the

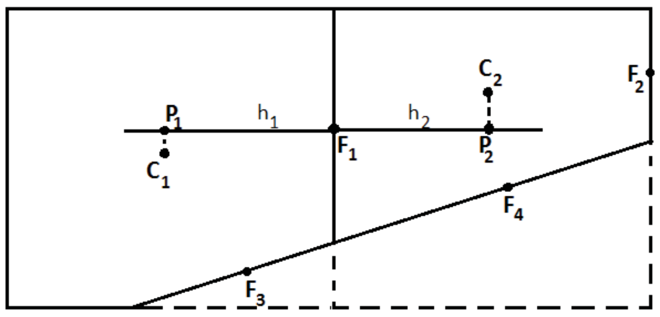


Figure 7.2. Boundary of cut cells.

cell-centered gradient of cut cells, the face-centered fluxing velocity and pressure of cut faces and boundary faces.

7.5.2.1 Limited Cell-Centered Gradients

As is well known, numerical results obtained from methods with the assumption that the cell variables are constant within each cell are very diffusive; in order to increase the order of accuracy for these methods, the distribution of cell variables is assumed to be varying within the cell and is determined using the gradients of cell variables. This requires a reconstruction of cell variables' gradients for each control volume. Typical methods for gradient reconstruction either use a least squares or a Gauss-Green formula approach. An in-depth study of computational complexity, discretization accuracy, and convergence rates of some of these methods is conducted in Diskin and Thomas [29]. In [29], the cell-centered node-averaging(CC-NA) schemes for gradient reconstruction show that a first-order accurate gradient reconstruction is sufficient for use with a second-order discretization scheme. The CC-NA gradient of the variable q in cell j as defined in Dukowicz and Kodis [30] is:

$$\langle \nabla q_j \rangle = \frac{1}{V_j^{NC}} \int_{V_j^{NC}} \nabla q dV, \tag{7.52}$$

where V_j^{NC} is the volume defined by the centroids of all the neighbors of the cell j as shown in Figure 7.3. The numerical value of the above integral is evaluated using the divergence theorem as given by:

$$\langle \nabla q_j \rangle = \frac{1}{V_j^{NC}} \oint_{S_j^{NC}} q\mathbf{n} dS, \tag{7.53}$$

where S_j^{NC} is the surface of volume V_j^{NC} and $\mathbf{n}$ is the outward surface normal of S_j^{NC}. Assume that q varies linearly along the surface, then the approximation in (7.53) is a first-order

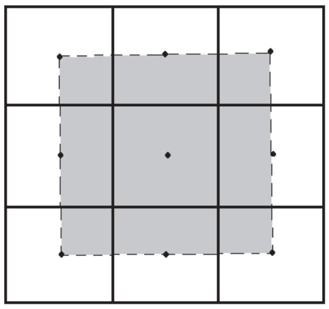


Figure 7.3. Cell-centered gradient of variables is approximated using values in the volume defined by the centroids of the neighboring cells.

approximation of the cell-centered average gradient of q in (7.52). In order to evaluate the integral over the face in (7.53), we sum up the products of the face directed area and the solution at the face. However, using the gradient estimate in (7.53) for construction of cell variables' distribution in the surrounding area of steep gradients might produce undershoots and/or overshoots when compared to neighboring data; limiting the gradient value has been used in the literature to prevent this. The multidimensional Van Leer limiting method uses the limiting coefficient $\alpha_j (0 \leq \alpha_j \leq 1)$ for each cell j such that the limited cell-centered gradient is defined as:

$$\nabla q_j = \alpha_j \langle \nabla q_j \rangle, \tag{7.54}$$

and coefficient α_j is determined as:

$$\alpha_j = \min\left(1, \alpha_{max}, \alpha_{min}\right), \tag{7.55}$$

where:

$$\alpha_{max} = \max\left(0, \frac{q_{max} - q_j}{q_{jmax} - q_j}\right), \tag{7.56}$$

$$\alpha_{min} = \max\left(0, \frac{q_{min} - q_j}{q_{jmin} - q_j}\right), \tag{7.57}$$

and q_{max}, q_{min} are the maximum and minimum values of q in the neighboring cells, and q_{jmax}, q_{jmin} are the maximum and minimum values of q in cell j. The maximum and minimum values of q in cell j are the maximum and minimum values of q at cell vertices. The value of q at a cell vertex, which is called the trial vertex value, is interpolated from the value at the cell center using the cell-centered averaging gradient. The trial vertex value, q_v, is then given as follows:

$$q_v = q_j + \langle \nabla q_j \rangle (\mathbf{x}_v - \mathbf{x}_j), \tag{7.58}$$

where $\mathbf{x}_v$ is the position of the vertex. The above gradient limiting procedure is used to calculate the limited cell-centered gradient of density and pressure. As mentioned in Vanderhayden and Kashiwa [127], the gradient limiting procedure for mass-specific transport quantities such as velocity, energy per unit mass, temperature, or species mass fraction needs to be implemented differently in order to eliminate artificial extrema and preserving the monotone character of the van Leer method. The formulation for the limited gradient of mass-specific transport quantities that maintains the monotone character of the van Leer method is called compatible by Vanderhayden and Kashiwa [127]. The compatible gradient limiting procedure proposed in [127] is used to calculate the cell-centered gradient of velocity. In the procedure for calculating the limited gradient of velocity of [127], the trial vertex value is:

$$\mathbf{u}_v = \mathbf{u}_j + \frac{\rho_j \langle \nabla \mathbf{u}_j \rangle (\mathbf{x}_v - \mathbf{x}_j)}{\rho_j + \nabla \rho_j (\mathbf{x}_v - \mathbf{x}_j)}. \tag{7.59}$$

After having determined the limited cell-centered gradient of density, pressure, and velocity using the above procedure, the limited cell-centered gradient of total energy is given by:

$$\nabla E_j = \frac{1}{(\gamma - 1)\rho_j} \nabla p_j - \frac{(E_j - \frac{1}{2}\mathbf{u}_j \cdot \mathbf{u}_j)}{\rho_j} \nabla \rho_j + \mathbf{u}_j^T \nabla \mathbf{u}_j. \tag{7.60}$$

The above equation is obtained from differentiating the equation of state in (3.5).

For the cells close to the embedded boundary, some neighboring cells are either cut or not included in the computational domain. In order to approximate the cell-centered gradient of cut cells, the divergence theorem is now applied to the volume defined by the centroids of the neighbors of these cut cells and the embedded boundary as shown in Figure 7.4.

7.5.2.2 Face-centered Pressure Gradient of Cut Faces

In order to estimate the pressure gradient at center F_1 of cut face, see Figure 7.2, in its normal direction x_i, we first find the projection points P_1 and P_2 of cell centers C_1 and C_2 onto the line that is perpendicular to the face and pass through the face center. The pressures at P_1

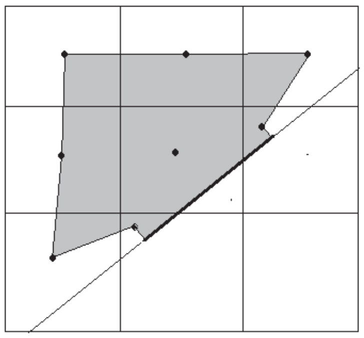


Figure 7.4. Cell-centered gradient of cut cell.

and P_2, $p(P_1)$ and $p(P_2)$, are obtained from interpolation the pressures at face centers C_1 and C_2 using the cell-centered limited gradient. The pressure gradient at F_1 is then estimated by:

$$\frac{\partial}{\partial x_i} p^n_{j+\frac{1}{2}} = \frac{p(P_2) - p(P_1)}{h_1 + h_2}. \tag{7.61}$$

The pressure gradient estimated with the above equation is used in the right side of Equation (7.22) when calculating the face-centered velocity.

7.5.2.3 Face-centered Fluxing Velocity and Pressure of Cut Faces

For the cut face located at $\mathbf{x}_{j+\frac{1}{2}}$, the left and right extrapolated values are defined as follows:

$$\mathbf{W}^{n(L)}_{j+\frac{1}{2}} = \mathbf{W}^n_j \quad + \nabla\mathbf{W}^n_j(\mathbf{x}_{j+\frac{1}{2}} - \mathbf{x}_j), \tag{7.62}$$

$$\mathbf{W}^{n(R)}_{j+\frac{1}{2}} = \mathbf{W}^n_{j+1} + \nabla\mathbf{W}^n_{j+1}(\mathbf{x}_{j+\frac{1}{2}} - \mathbf{x}_{j+1}), \tag{7.63}$$

where $\nabla\mathbf{W}^n_j$ is the limited cell-centered gradient whose calculation is discussed in detail in Section 7.5.2.1. The calculation of the face-centered fluxing velocity and pressure of the cut face follows equations (7.21)–(7.25) with the face-centered pressure gradient of the cut face calculated using (7.61).

7.5.2.4 Face-centered Fluxing Velocity and Pressure of Boundary Faces

The face-centered fluxing velocity and pressure of boundary faces are used to evaluate the changes in mass, momentum, and energy of cut cell that are described by Equations (7.26)–(7.28). In IMPICE, the face-centered fluxing velocity and pressure are derived from the numerical solution of the GRP problem constructed at the center of the face. A similar approach is also used in the calculation of the face-centered fluxing velocity and pressure of boundary faces. In this approach, in order to determine the fluxing velocity of the boundary face of the cell whose center is located at $\mathbf{x}_j$, see Figure 7.5, we need to construct a GRP problem at the center of the face, $\mathbf{x}_{j+\frac{1}{2}}$. Consider the Riemann problem where two constant states are separated at $\mathbf{x}_{j+\frac{1}{2}}$: the left constant state $\mathbf{U}^{n(L)}_{j+\frac{1}{2}}$ represents the value approaching $\mathbf{x}_{j+\frac{1}{2}}$ of the cut cell which is obtained using (7.62); the right constant state $\mathbf{U}^{n(R)}_{j+\frac{1}{2}}$, however, can not be determined using (7.63) since there is not a cell in the right side of the boundary. Alternatively, the right constant state can be determined using reflected values of the conserved quantities $\mathbf{U}^{n(L)}_{j+\frac{1}{2}}$. As mentioned in Helzel *et al.* [55], the method of reflecting the conserved quantities is a widely used procedure for obtaining boundary fluxes that simulate a reflecting boundary. With the method of reflecting of conserved quantities, first we have to rotate the coordinate so that the boundary plane becomes a coordinate plane in x_i-direction. With this rotation, the conserved quantities of left constant state becomes $\hat{\mathbf{U}}^{n(L)}_{j+\frac{1}{2}}$. The conserved quantities of right constant states $\hat{\mathbf{U}}^{n(R)}_{j+\frac{1}{2}}$ is obtained from $\hat{\mathbf{U}}^{n(L)}_{j+\frac{1}{2}}$ where:

$$\hat{\rho}^{n(R)}_{j+\frac{1}{2}} = \hat{\rho}^{n(L)}_{j+\frac{1}{2}}, \tag{7.64}$$

$$\hat{\mathbf{u}}^{n(R)}_{j+\frac{1}{2}} = \mathbf{A}\hat{\mathbf{u}}^{n(L)}_{j+\frac{1}{2}}, \tag{7.65}$$

$$\hat{p}^{n(R)}_{j+\frac{1}{2}} = \hat{p}^{n(L)}_{j+\frac{1}{2}}, \tag{7.66}$$

and matrix $\mathbf{A} = (a_{m,l})$ with entries:

$$a_{m,l} = \begin{cases} 1 & \text{if} \quad m = l \ \ \&\& \ \ m \neq i, \\ -1 & \text{if} \quad m = l \ \ \&\& \ \ m = i, \\ 0 & \text{otherwise.} \end{cases} \tag{7.67}$$

In order to determine the fluxing velocity at $\mathbf{x}_{j+\frac{1}{2}}$, we may consider the GRP problem either in the original grid, the unrotated grid, or the rotated grid where the embedded boundary is the coordinate plane in x_i-direction. If the considered GRP problem is in the unrotated grid, we need to obtain the boundary reflecting value $\hat{\mathbf{U}}^{n(R)}_{j+\frac{1}{2}}$ in the original grid which is denoted as $\mathbf{U}^{n(R)}_{j+\frac{1}{2}}$, and solve the GRP problem with two separated constant states $\left(\mathbf{U}^{n(L)}_{j+\frac{1}{2}} \text{ and } \mathbf{U}^{n(R)}_{j+\frac{1}{2}}\right)$ to evaluate the approximate fluxing velocity at the face-center $\mathbf{x}_{j+\frac{1}{2}}$ of the boundary face. If the considered GRP problem is in the rotated grid, the approximate fluxing velocity at the face-center $\mathbf{x}_{j+\frac{1}{2}}$ is evaluated using the approach of Section 7.2 to find the approximate solution of the Riemann problem where the two constant states are separated by a plane that is not a coordinate plane.

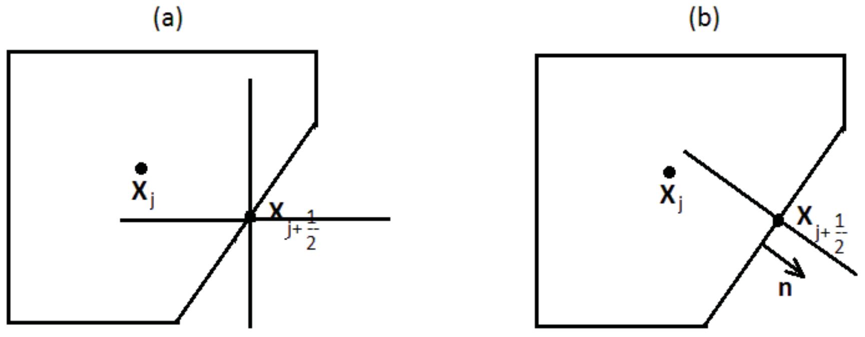


Figure 7.5. GRP at face centerer of boundary face. (a) unrotated grid and (b) rotated grid.

The approximate fluxing velocity is then time-advanced using the cell-centered limited gradient to obtain the face-centered fluxing velocity of the boundary face. The face-centered pressure of boundary face is obtained from interpolating the pressure of the cut cell using the cell-centered limited gradient.

7.5.2.5 Higher Order Advection

For the cut cells, the limited gradient used in Equation (7.38) is obtained using the limited cell-centered gradient calculated in Section 7.5.2.1 instead of using Equation (7.39).

7.5.2.6 Merge Very Small Cells

The grid cells near the embedded boundary may be orders of magnitude smaller than the regular Cartesian grid cells; it is then necessary to use a very small time step to maintain the stability of the method. In order to overcome this time step restriction, the cell merging technique is often used. Many cell merging techniques have been discussed in the literature, see Helzel *et al.* [55] and the references within. In these cell merging techniques, small irregular cut cells are merged together with a neighboring regular grid cell with several different variants mentioned in Helzel *et al.* [55]. In order to handle small cells in IMPICE, we use a new variation of the cell merging technique that makes use of the magnitudes of surrounding face-centered fluxing velocities. In this approach, we treat the small cells as independent cells and merge their values with neighbor cells at the end of integration step. At the end of the integration step, we determine if a cut cell is a small cell based on the ratio between the volume of the cut cell and the size of a regular grid cell. Let V_{reg} be the volume of regular cells and V_{cut} be the volume of the cut cell; at each cut cell the following volume ratio is calculated:

$$r_c = \frac{V_{cut}}{V_{reg}}. \tag{7.68}$$

A cut cell is then identified as a small cell using the following criteria:

$$r_c < a, \tag{7.69}$$

for some constant a. The small cell is then merged with a cell selected from its neighbors. The neighbor cell selected for merging is based on the face-centered fluxing velocity at the common face. The small cell values are merged with the cell that has the greatest value of the fluxing velocity at the common face. The values of the small cell and its selected merged cell at the end of the time integration step are the volume averages of their combined values. If we assume that we merge the small cell j with the merged neighbor cell, as denoted as M_j, then:

$$q_j^{n+1} = q_{M_j}^{n+1} = \frac{V_j^{n+1} q_j^{n+1} + V_{M_j}^{n+1} q_{M_j}^{n+1}}{V_j^{n+1} + V_{M_j}^{n+1}}, \tag{7.70}$$

where $q = \rho, \rho\mathbf{u}, \rho E$. We will use this approach to overcome the small cell problem in the IMPICE method. In IMPICE, when calculating the face-centered fluxing velocity, we use the time-advanced scheme shown in Equation (7.23). The face-centered fluxing velocity which is obtained from Equation (7.23), where time step Δt is "too large" for the size of the small cell (the time step should be proportional to the size of a grid cell to maintain stabilities), is "too large" that causes "too much" mass, momentum and energy of the small cell are fluxed to the neighbor cell. When this happens, these conserved quantities in some small cells become negative and the method becomes unstable. So the merging technique with the merging values determined using Equation (7.70) helps to redistribute mass, momentum, and energy to prevent the negative values of these quantities in small cells.

7.6 Numerical Results

We apply the multidimensional IMPICE method to a set of problems that are usually used for testing numerical methods for the system of Euler equations. For the performance of the embedded boundary method, the problem of shock reflection from a wedge is used.

7.6.1 Testing Problems

7.6.1.1 Modified Shock Tube Problem

In Chapter 6, we presented the numerical solutions of the one-dimensional IMPICE method to several classical one-dimensional shock tube problems. In order to check for the correctness of multidimensional setting of the method, we now present the numerical solution to the one-dimensional shock tube problem with a multidimensional extension. The computational domain of the chosen shock tube problem is $\mathbf{x} \in [0.0, 1.0] \times [0.0, 0.05]$ and $t \in (0, T_e)$ and two constant states are sparated by a discontinuity at $x_0 = 0.3$; the problem is governed by the system of Euler equations in two-dimensional space with the given initial condition:

$$(\rho, u_1, u_2, p)(\mathbf{x}, 0) = \begin{cases} (1.0, 0.75, 0.0, 1.0) & \text{if} \quad x_0 < 0.3, \\ (0.125, 0.0, 0.0, 0.1) & \text{if} \quad x_0 > 0.3. \end{cases} \tag{7.71}$$

This is a modified version of the standard shock tube problem, proposed in Toro [120].

7.6.1.2 Two-dimensional Explosion Problem

The two-dimensional explosion test problem in a rectangular domain $\mathbf{x} \in [-1.0, 1.0] \times [-1.0, 1.0]$ and $t \in (0, T_e)$. The problem is governed by the system of the two-dimensional Euler equations with the given initial condition:

$$(\rho, u_1, u_2, p)(\mathbf{x}, 0) = \begin{cases} (1.0, 0, 0, 1.0) & \text{if} \quad r < 0.4, \\ (0.125, 0, 0, 0.1) & \text{if} \quad r > 0.4, \end{cases} \tag{7.72}$$

where $r^2 = x_1^2 + x_2^2$. The detailed description of this problem can be found in Toro [120].

7.6.1.3 Two-dimensional Explosion Problem with a Large Jump in Pressure

This problem has the same computational domain as the above two-dimensional explosion problem, but it has a large jump in pressure which is over four orders of magnitude and thus will produce a very strong outward traveling shock wave; see Toro [120] for problem details. The initial condition is given by:

$$(\rho, u_1, u_2, p)(\mathbf{x}, 0) = \begin{cases} (10.0, 0, 0, 0.0, 1000) & \text{if} \quad r < 0.4, \\ (1.0, 0, 0, 0.0, 0.1) & \text{if} \quad r > 0.4, \end{cases} \tag{7.73}$$

where $r^2 = x_1^2 + x_2^2$.

7.6.1.4 Shock Reflection from a Wedge Problem

We study the approximation of a Mach 10 moving shock wave reflected from a 30-degree wedge which is originally proposed by Woodward and Colella in [138]. The computational domain of this problem is $\mathbf{x} \in [0, 3] \times [0, 2]$. The governing equations are the two-dimensional

Euler equations. The wedge is positioned at $(0.5, 0)$ with the initial condition in front and back of the shock is given by the Rankine-Hugnoniot conditions as follows:

$$(\rho, u_1, u_2, p)(\mathbf{x}, 0) = \begin{cases} (8.0, 8.25, 0.0, 116.5) & \text{if} \quad x_1 < 0.5, \\ (1.4, 0.0, 0.0, 1.0) & \text{if} \quad x_1 \geq 0.5, \end{cases} \tag{7.74}$$

The solution to this problem exhibits very strong discontinuities, wall-bounded flows and furthermore develops rich small-scale structures in time, which are difficult to resolve.

7.6.2 Numerical Results of the Multidimensional IMPICE Method with First-order Advection

The above test problems are now used to test the implementation of the multidimensional IMPICE method with first-order advection. We apply the transmissive boundary condition in all directions for all of the above problems except for the problem of shock reflection from a wedge. For the wedge problem, the exact solution is imposed for the segment $[0, 0.5]$ of the bottom boundary and the top boundary, a reflective boundary condition is placed on the wedge, the inflow boundary condition is applied on the left boundary, and the transmissive boundary condition is used on the right boundary. The numerical results of the multidimensional IMPICE method with first-order advection to these test problems are shown in Figures 7.6–7.9. The embedded boundary method is implemented for the wedge problem. In these figures, the numerical results of these problems (except for the shock reflection problem) are compared against either their exact or "exact" solution. The "exact" solutions for the explosion problems drawn in these figures are obtained from using the Random Choice Method (RCM) with the parameters decribed in Toro [120]. In [120], these RCM solutions are also regarded as the exact solutions since the RCM resolves discontinuities as true discontinuities and the errors are only from the position of the waves. In Figure 7.6, the computed density ρ and velocity u_1 in x_1-direction are plotted at $T_e = 0.2$; a cut along the x_1-axis of these quantities is plotted against the exact solution, which is the exact solution of the one-dimensional modified shock tube problem. In this figure, the result is oscillation-free and the solution profile of the modified shock tube problem in x_1-direction is in good agreement with the numerical solution obtained using the one-dimensional IMPICE method. The computed density ρ and pressure p in the numerical solutions to the two-dimensional explosion problems are plotted and compared against the "exact" values in the Figures 7.7–7.8; the numerical solutions in these figures approach the "exact" solutions and are mostly free of oscillations. The computed density ρ in the numerical solution to the wedge problem is depicted with forty-eight contour lines from 0.45 to 21.6 in Figure 7.9. There are very small cells resulting from the discretization of the computational domain of the wedge problem; therefore, the cells with volume that is smaller than 0.05 times of

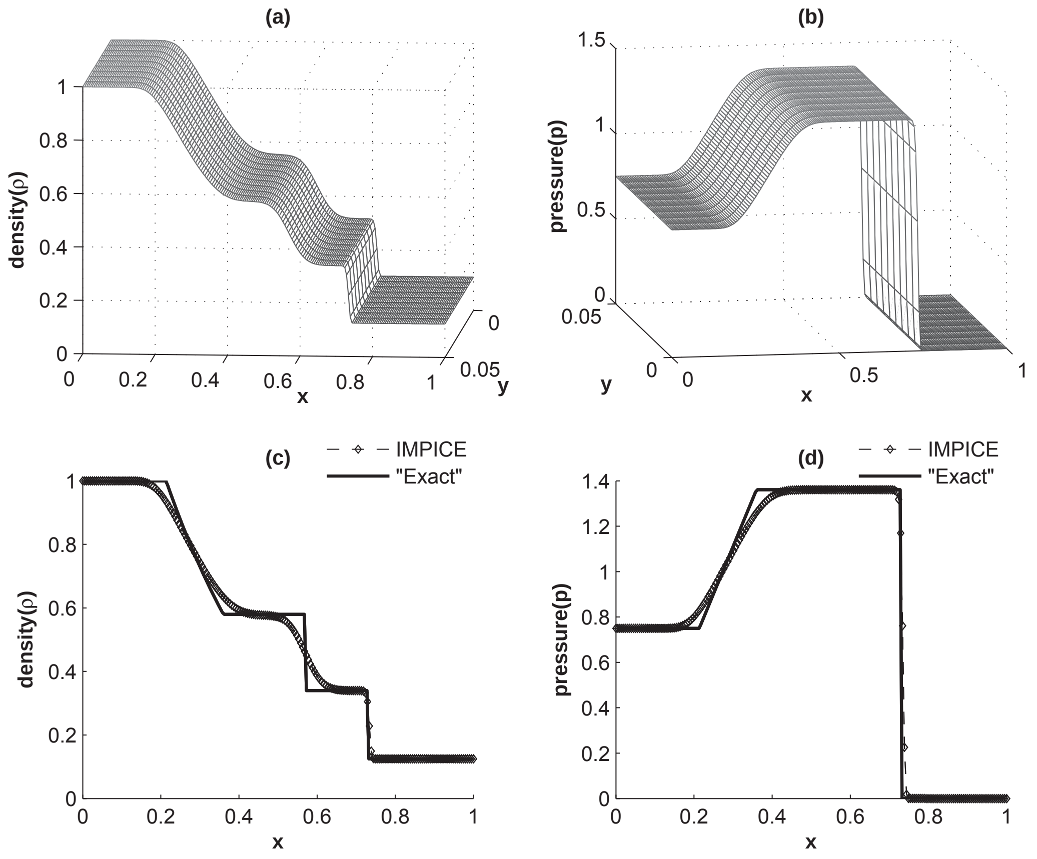


Figure 7.6. Modified shock tube problem. $T_e = 0.2$. IMPICE with first-order advection on $N_1 \times N_2 = 200 \times 10$ grid, $C_{cfl} = 0.3$. Two-dimensional distribution of (a)density and (b)velocity, and a cut along the x_1-axis of (c)density and (d)velocity.

the standard cell volume are merged with neighboring cells to overcome the time step restriction. The analysis in Chapter 6 suggested that the IMPICE method – the analysis is done for the case of one-dimensional space, but may be derived for the case of multidimensional space – is first-order in space and time, so the numerical solutions are highly smeared at contact discontinuities as they appear in these figures; for instance, the numerical solution to the shock reflection from a wedge problem can not capture the detail in the close-up region in Figure 7.9.

7.6.3 Numerical Results of the Multidimensional IMPICE Method with Second-order Advection

The second-order extension of the method in space in Section 7.4 increases the order of accuracy in space using the second-order advection. In the second-order advection approach, the advected quantities at cell intefaces are interpolated from the cell-centered data. The

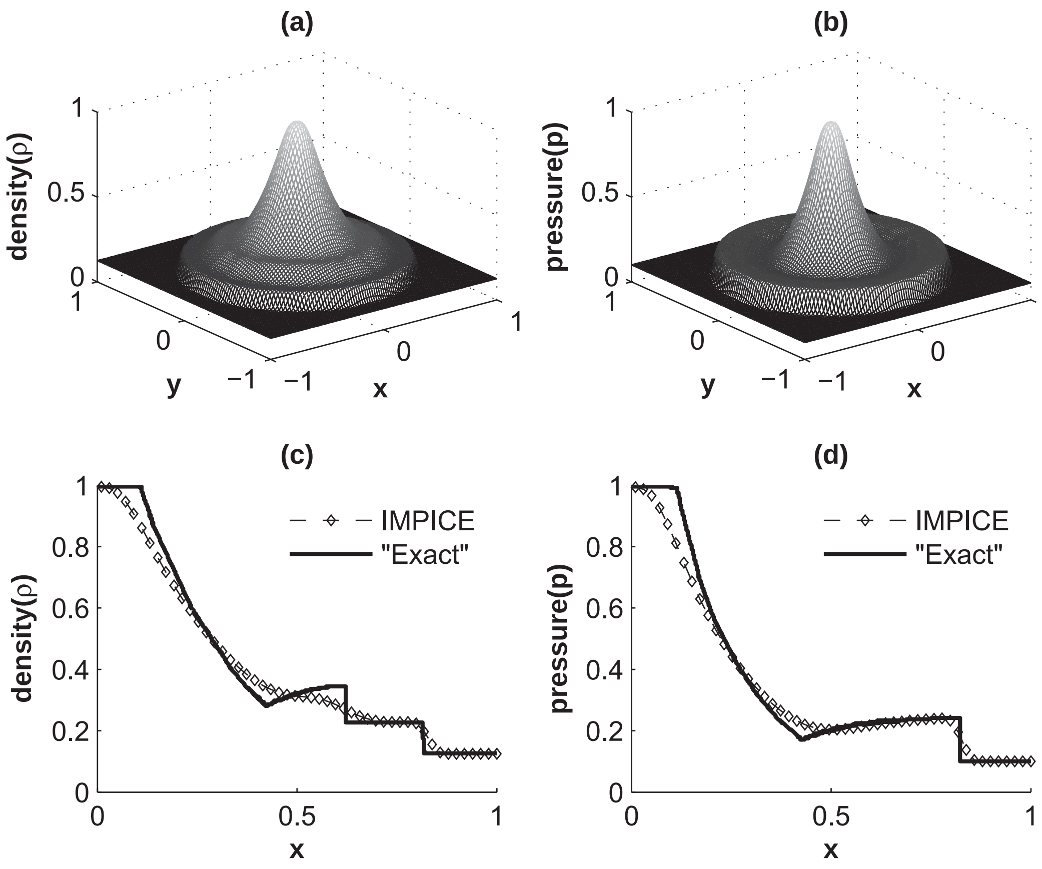


Figure 7.7. Two-dimensional explosion problem. $T_e = 0.25$. IMPICE with first-order advection on $N_1 \times N_2 = 100 \times 100$ grid, $C_{cfl} = 0.3$. Two-dimensional distribution of (a)density and (b)pressure, and a cut along the x_1-axis of (c)density and (d)pressure.

numerical results of this multidimensional IMPICE method with the discussed second-order advection are shown in Figures 7.10–7.14. As seen in these figures, we obtain more accurate numerical solutions when using the second-order advection. Especially for the wedge problem, the numerical solution resolves the detail in the close-up region of Figure 7.13, and the contour lines of the density ρ for the obtained result using the method described in this chapter is similar to the contour line plot of the results from previous publications for this problem; for example, see [55, 61, 100, 119].

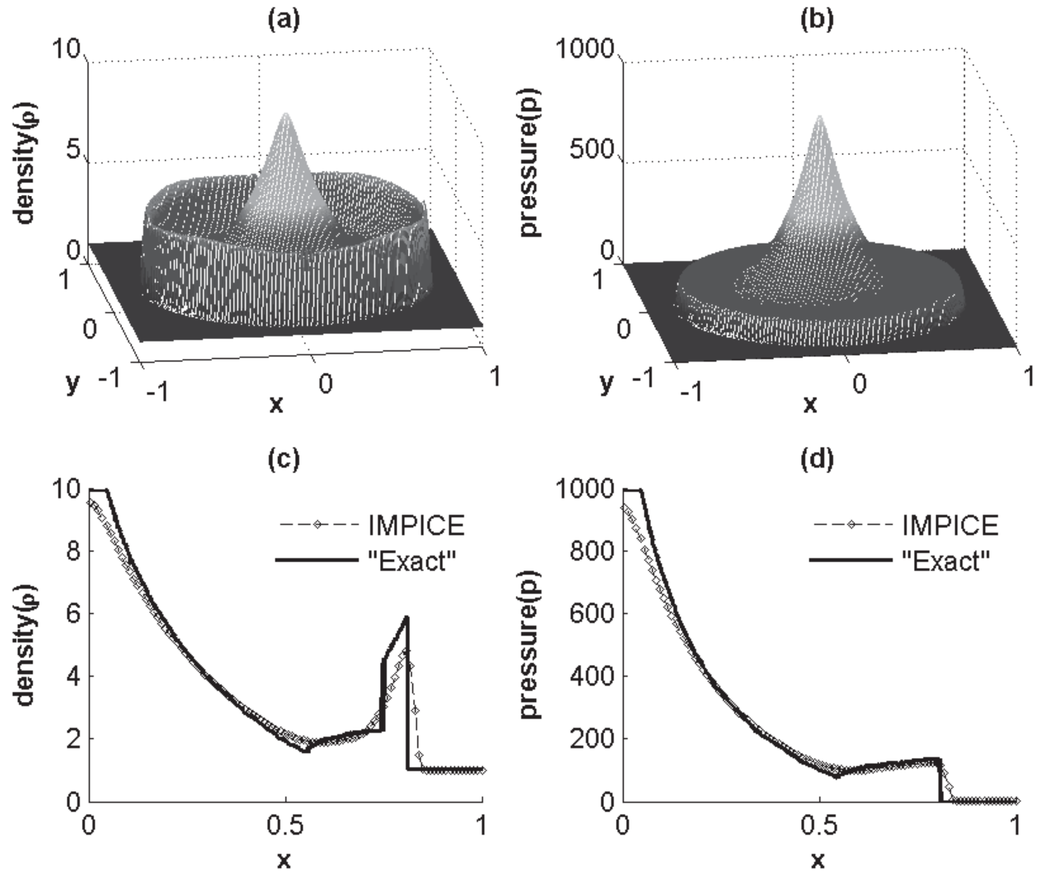


Figure 7.8. Two-dimensional explosion problem with large jump in pressure. $T_e = 0.03$. IMPICE with first-order advection on $N_1 \times N_2 = 300 \times 300$ grid, $C_{cfl} = 0.3$. Two-dimensional distribution of (a)density and (b)pressure, and a cut along the x_1-axis of (c)density and (d)pressure.

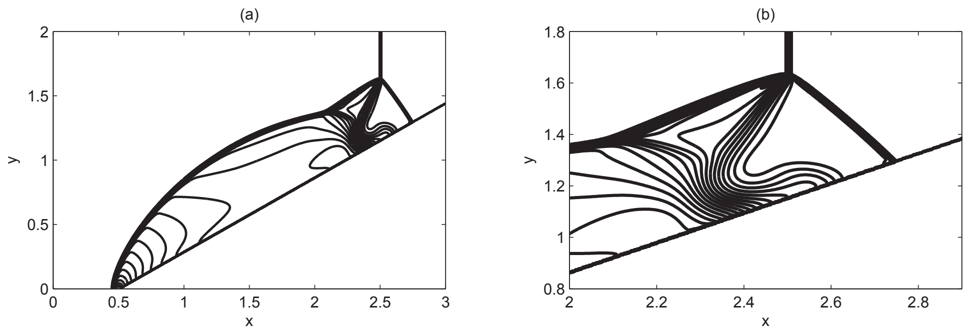


Figure 7.9. Shock reflection from a wedge problem. $T_e = 0.2$. IMPICE with first-order advection on $N_1 \times N_2 = 900 \times 600$ grid, $C_{cfl} = 0.3$. A cut cell is merged if the volume ratio, r_c, is less than 0.05. Forty-eight density contour lines from 0.45 to 21.6. (b) is zoomed area of (a).

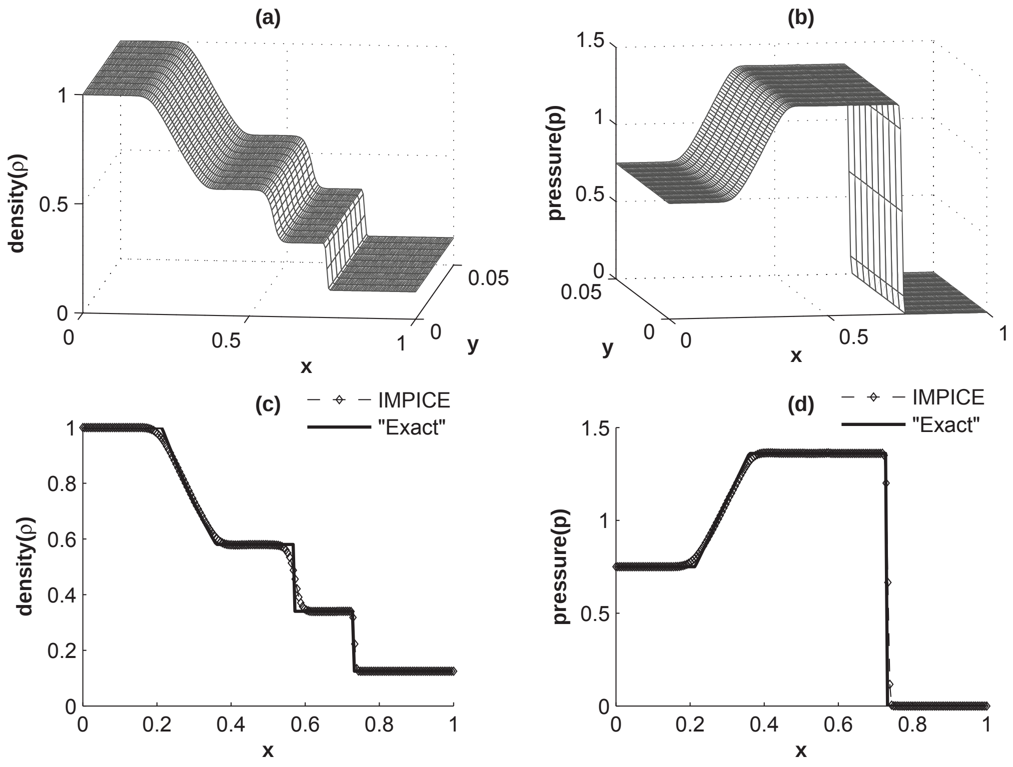


Figure 7.10. Modified shock tube problem. $T_e = 0.2$. IMPICE with second-order advection on $N_1 \times N_2 = 200 \times 10$ grid, $C_{cfl} = 0.3$. Two-dimensional distribution of (a)density and (b)velocity, and a cut along the x_1-axis of (c)density and (d)velocity.

7.7 Accuracy of the IMPICE Method for Solving the Advection Equation on an Embedded Boundary

Consider the following advection equation:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0, \tag{7.75}$$

where $\mathbf{u}(\mathbf{x}, t)$ is given advection velocity and $\rho(\mathbf{x}, T_e)$ is solved from initial condition $\rho(\mathbf{x}, 0)$.

The IMPICE method in Section 7.3 with the embedded boundary treatment in Section 7.5.2 is used to solve Equation (7.75) on the same computational domain for the wedge problem given in Section 7.6.1.5. We consider a constant advection velocity given by $\mathbf{u} = [cos(\theta), sin(\theta)]^T$ where θ is the angle between the wedge and the horizontal axis. In order to use the IMPICE method for this advection problem, we assign a constant to the initial pressure on the computational domain. Two different initial conditions of $\rho(\mathbf{x}, 0)$, one without a jump and one with a jump in density, of the advection equation are considered as follows:

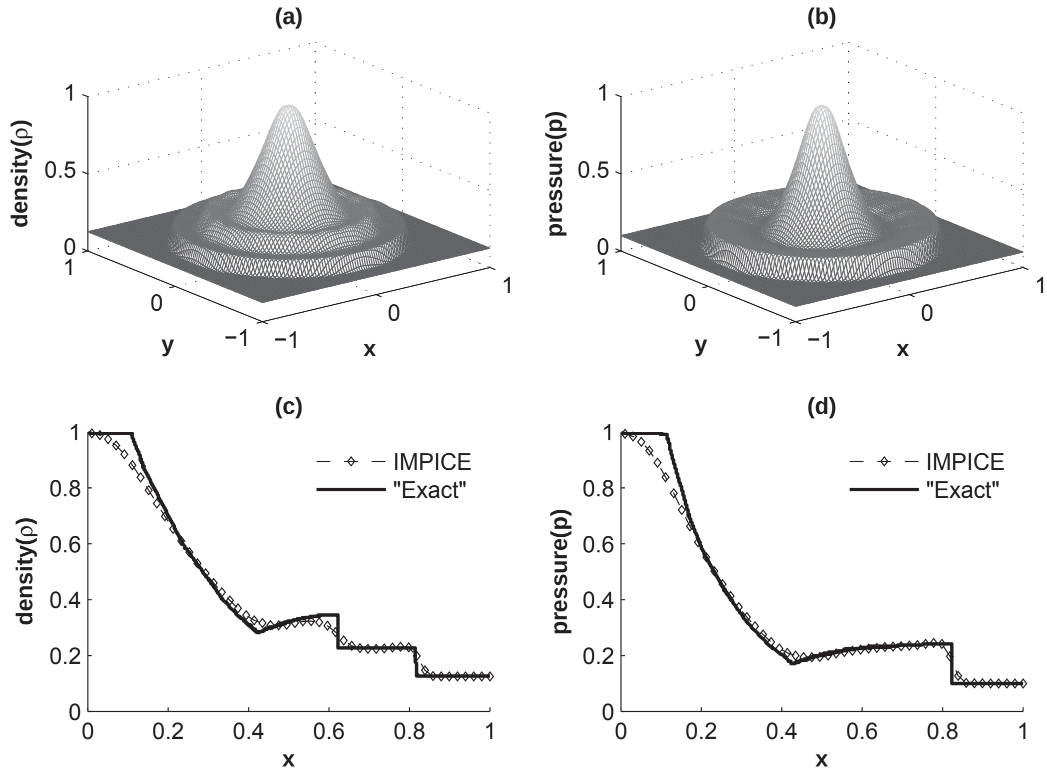


Figure 7.11. Two-dimensional explosion problem. $T_e = 0.25$. IMPICE with second-order advection on $N_1 \times N_2 = 100 \times 100$ grid, $C_{cfl} = 0.3$. Two-dimensional distribution of (a)density and (b)pressure, and a cut along the x_1-axis of (c)density and (d)pressure.

7.7.1 Advection 1

The initial value $\rho(\mathbf{x}, 0)$ is given by:

$$\rho(\mathbf{x},0) = \begin{cases} 2.0 & \text{if } \; x_1 \leq a, \\ 2.0 + sin\left(10(x_1 - a)\right) & \text{if } \; a < x_1 < a + \frac{\pi}{5}, \\ 2.0 & \text{if } \; a + \frac{\pi}{5} \leq x_1. \end{cases} \tag{7.76}$$

where $a = 0.6$.

7.7.2 Advection 2

The initial value $\rho(\mathbf{x}, 0)$ is given by:

$$\rho(\mathbf{x},0) = \begin{cases} 2.0 & \text{if } \; x_1 \leq 0.6, \\ 10.0 & \text{if } \; 0.6 < x_1 < 1.2, \\ 2.0 & \text{if } \; 1.2 \leq x_1. \end{cases} \tag{7.77}$$

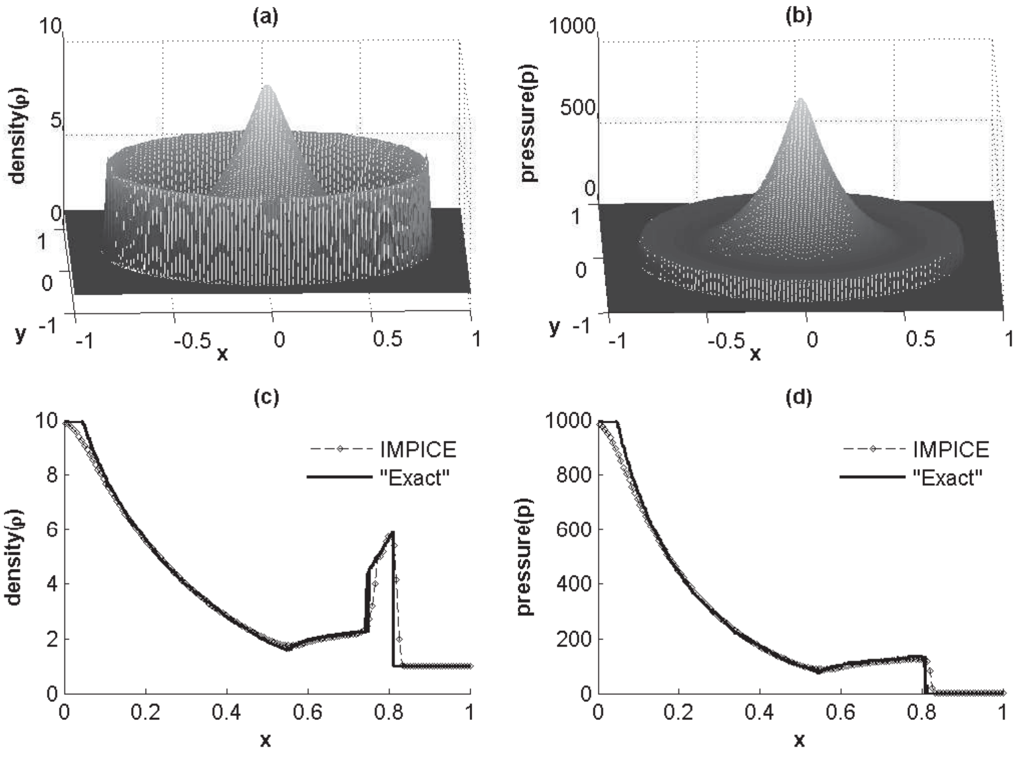


Figure 7.12. Two-dimensional explosion problem with large jump in pressure. $T_{end} = 0.03$. IMPICE with second-order advection on $N_1 \times N_2 = 300 \times 300$ grid, $C_{cfl} = 0.3$. Two-dimensional distribution of (a)density and (b)pressure, and a cut along the x_1-axis of (c)density and (d)pressure.

7.7.3 Numerical Results

The numerical solutions to the advection problems in Sections 7.7.1 and 7.7.2 at $T_e = 0.5$ using IMPICE with second-order advection are shown in Figure 7.15. The overall error norms and the orders of accuracy for these problems with several different mesh sizes are summarized in Table 7.1. For problem in Section 7.7.1 in which there is no jump in density, the orders of accuracy for first-order and second-order advection are as expected. For problem in Section 7.7.2 in which there is a jump in density, there is a degeneration in accuracies.

7.8 Conclusions

We have presented a generalization of the one-dimensional IMPICE method for solving multidimensional compressible flow problems. In order to prevent the oscillations in the IMPICE's numerical solutions to the multidimensional system of Euler equations, it is necessary to apply a multidimensional limiting process to limit the gradients. We tested the implementation of the multidimensional IMPICE method on a suite of test problems for the system of Euler

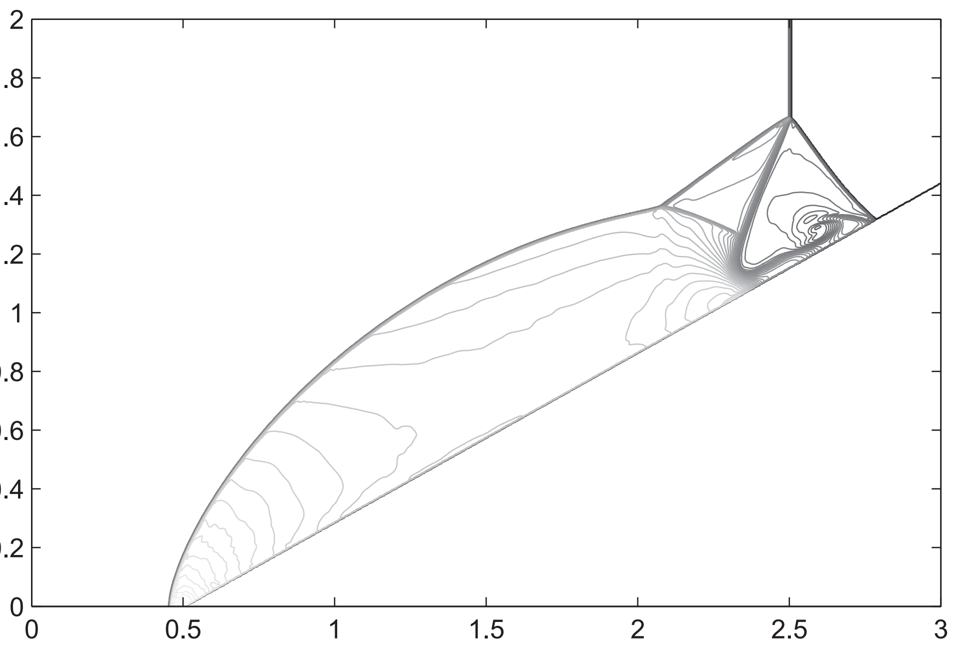


Figure 7.13. Shock reflection from a wedge problem. $T_e = 0.2$. IMPICE with second-order advection on $N_1 \times N_2 = 900 \times 600$ grid, $C_{cfl} = 0.3$. A cut cell is merged if the volume ratio, r_c, is less than 0.05. Sixty density contour lines from 0.4 to 23.5.

Table 7.1. L_1-norms and the order of accuracy m of the overall errors in the numerical solutions to the advection problem at $T_e = 0.5$.

		first-order		second-order	
	$N_1 \times N_2$	$\|\mathbf{ge}^\rho(T_e)\|_{L_1}$	m	$\|\mathbf{ge}^\rho(T_e)\|_{L_1}$	m
Advection 1	60×40	4.28E-01	—	9.06E-02	—
	120×80	2.74E-01	0.64	2.88E-02	1.65
	240×160	1.64E-01	0.74	8.91E-03	1.69
	480×320	9.55E-02	0.78	2.52E-03	1.82
Advection 2	60×40	2.69E-00	—	1.08E-00	—
	120×80	1.90E-00	0.50	6.32E-01	0.77
	240×160	1.33E-00	0.51	3.62E-01	0.80
	480×320	9.32E-01	0.51	2.11E-01	0.78

equations in multidimensional space where the obtained numerical solutions have shown the ability to capture shock waves. We implemented the method of cut cells to allow IMPICE to solve problems in complex geometries. The implementation of the method of cut cells was tested on the problem of shock reflection from a wedge. In this chapter, we have shown the idea of using linear spatial distribution of cell variables for the multidimensional case makes it possible to raise the order of accuracy in space.

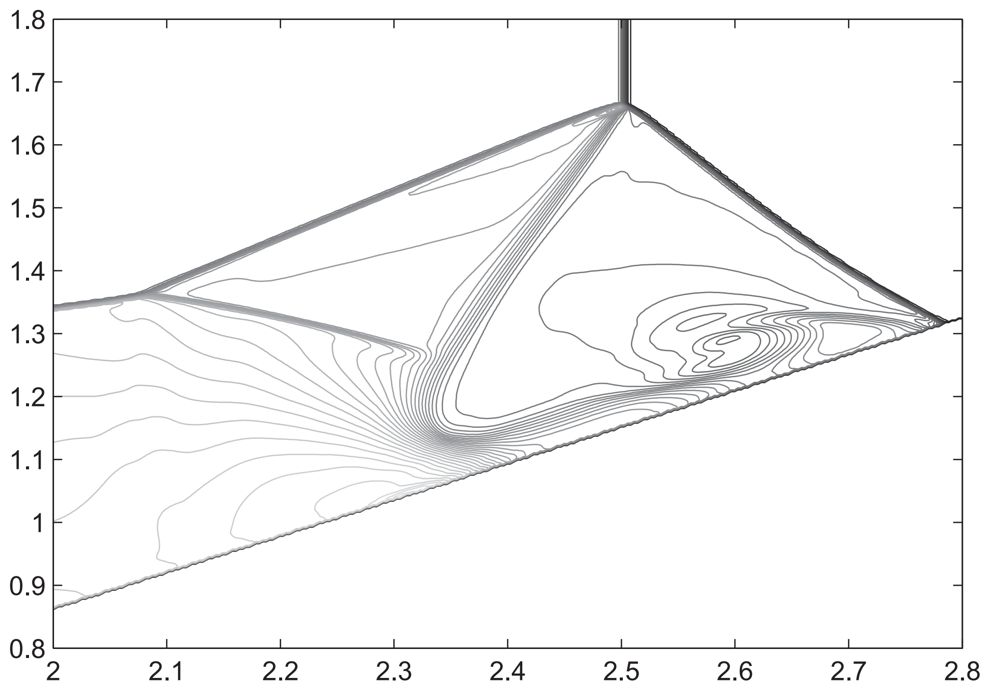


Figure 7.14. Shock reflection from a wedge problem. $T_e = 0.2$. IMPICE with second-order advection on $N_1 \times N_2 = 900 \times 600$ grid, $C_{cfl} = 0.3$. This is a zoomed part of Figure 7.13 with three hundred density contour lines to show the solution detail in the interested area.

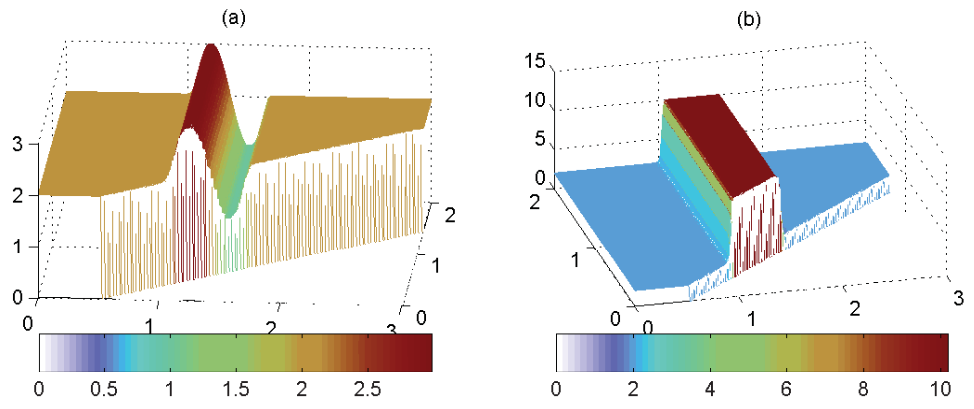


Figure 7.15. Numerical solutions to the advection problem at $T_e = 0.5$ using IMPICE with second-order advection on $N_1 \times N_2 = 240 \times 160$ grid, $C_{cfl} = 0.3$: (a) Advection 1 and (b) Advection 2.

CHAPTER 8

ADJOINT ERROR ESTIMATE FOR THE IMPROVED PRODUCTION IMPLICIT CONTINUOUS-FLUID EULERIAN METHOD

We have discussed in Chapter 4 the importance of being able to estimate the errors in a numerical solution and the increased use of adjoint-based error estimates in the error analysis of applications in Computational Fluid Dynamics (CFD). In this chapter, we propose a discrete adjoint approach for estimating the overall error in the numerical solutions of the one-dimensional IMPICE method.

8.1 Introduction

The IMPICE method discussed in Section 6.4 approximates the solution of the one-dimensional Euler equations in the conservation law form (6.22) at the set of discrete points $\{x_j \ : \ j = 1, ..., N\}$ which are the centers of the spatial mesh cells. The approximate solution at (x_j, t_n) is the average value of cell j, $\mathbf{U}_j^n = \left[\rho_j^n, (\rho u)_j^n, (\rho E)_j^n\right]^T$.

Let:

$$\mathbf{Y} = \left[\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, ..., \mathbf{U}_N\right]^T, \tag{8.1}$$

where:

$$\mathbf{U}_j = \left[\rho_j, (\rho u)_j, (\rho E)_j\right]^T. \tag{8.2}$$

Consider the following ODE system:

$$\begin{cases} \dot{\mathbf{Y}}(t) & = \mathbf{G}\left(t, \mathbf{Y}(t)\right) \\ \mathbf{Y}(0) & = \mathbf{Y}_0, \end{cases} \tag{8.3}$$

where:

$$\mathbf{G} = [\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3, ..., \mathbf{G}_N]^T \tag{8.4}$$

and $\mathbf{G}_j\left(t, \mathbf{Y}(t)\right)$ is defined at discrete points in time as follows:

$$\mathbf{G}_j(t_n, \tilde{\mathbf{Y}}^n) = \frac{1}{\Delta x}\left(\mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n) - \mathbf{F}^{IMPICE}_{j-\frac{1}{2}}(t_n)\right), \tag{8.5}$$

where $\mathbf{F}^{IMPICE}_{j+\frac{1}{2}}(t_n)$ is given by Equation (6.58) and $\tilde{\mathbf{Y}}^n = [\mathbf{U}^n_1, \mathbf{U}^n_2, ..., \mathbf{U}^n_N]^T$.

If we are about to numerically solve the system in (6.22) on the computational domain at the discrete points $\{x_j \ : \ j = 1, ..., N\}$ by solving the ODE system in Equation (8.3), then the overall errors at these discrete points, as derived in Chapter 4, can be estimated using the following equation:

$$\mathbf{l}^T\mathbf{ge}(T_e) \approx \sum_{j=1}^{m} \boldsymbol{\lambda}^T(t_j)\left(\mathbf{le}\left(t_{j+1}; t_j, \tilde{\mathbf{Y}}^j\right) + (t_{j+1} - t_j)\mathbf{TE}(t_j)\right) + \boldsymbol{\lambda}^T(0)\mathbf{r}_0, \tag{8.6}$$

where a set of vector $\mathbf{l}$ is chosen such that all components of vector $\mathbf{ge}(T_e)$ can be revealed; such set of vector $\mathbf{l}$ is discussed in Section 4.2. In Equation (8.6), $\mathbf{le}\left(t_{j+1}; t_j, \tilde{\mathbf{Y}}^j\right)$ is the local error from solving the ODE system (8.3) on $[t_j, t_{j+1}]$, $\mathbf{TE}(t_j)$ is the spatial truncation error, and $\boldsymbol{\lambda}$ is the solution to the following adjoint system:

$$\begin{cases} \dot{\boldsymbol{\lambda}}(t) & = -\mathbf{J}^T(t, \tilde{\mathbf{Y}})\boldsymbol{\lambda}(t), \quad 0 \le t \le T_e \\ \boldsymbol{\lambda}(T_e) & = \mathbf{l}, \end{cases} \tag{8.7}$$

where $\mathbf{J}(t, \tilde{\mathbf{Y}})$ is the Jacobian of $\mathbf{G}\left(t, \mathbf{Y}(t)\right)$ in Equation (8.3) with respect to $\mathbf{Y}$ evaluated at $\tilde{\mathbf{Y}}$.

In the overall error given by Equation (8.6), the contributions of the temporal error and spatial error are given as follows:

$$\mathbf{l}^T\mathbf{et}(T_e) \approx \sum_{j=1}^{m} \boldsymbol{\lambda}^T(t_j)\mathbf{le}\left(t_{j+1}; t_j, \tilde{\mathbf{Y}}^j\right) + \boldsymbol{\lambda}^T(0)\mathbf{r}_0, \tag{8.8}$$

$$\mathbf{l}^T\mathbf{es}(T_e) \approx \sum_{j=1}^{m} (t_{j+1} - t_j)\boldsymbol{\lambda}^T(t_j)\mathbf{TE}(t_j), \tag{8.9}$$

8.2 Adjoint Problem Formulation for the One-dimensional IMPICE Method

The formulation of the adjoint problem for IMPICE involves the determination of $\mathbf{J}(t,\mathbf{Y})$ which is the Jacobian of $\mathbf{G}(t,\mathbf{Y}(t))$ with respect to $\mathbf{Y}$. For the purpose of error estimation, the adjoint system in (8.7) is solved backward in time at $t_m = T_e$, t_{m-1}, ..., $t_1 = 0$ using the discrete values $\{\mathbf{J}(t_n, \tilde{\mathbf{Y}}^n) \; : \; n = m, ..., 0\}$. From the definitions of $\mathbf{G}$ given by (8.4) and $\mathbf{Y}$ given by (8.1), we have:

$$\mathbf{J}(t_n,\tilde{\mathbf{Y}}^n) = \frac{\partial \mathbf{G}}{\partial \mathbf{Y}}\left(t_n,\tilde{\mathbf{Y}}^n\right) = \begin{bmatrix} \frac{\partial \mathbf{G}_1}{\partial \mathbf{U}_1}(t_n,\tilde{\mathbf{Y}}^n) & \frac{\partial \mathbf{G}_1}{\partial \mathbf{U}_2}(t_n,\tilde{\mathbf{Y}}^n) & \dots & \frac{\partial \mathbf{G}_1}{\partial \mathbf{U}_N}(t_n,\tilde{\mathbf{Y}}^n) \\ \frac{\partial \mathbf{G}_2}{\partial \mathbf{U}_1}(t_n,\tilde{\mathbf{Y}}^n) & \frac{\partial \mathbf{G}_2}{\partial \mathbf{U}_2}(t_n,\tilde{\mathbf{Y}}^n) & \dots & \frac{\partial \mathbf{G}_2}{\partial \mathbf{U}_N}(t_n,\tilde{\mathbf{Y}}^n) \\ \vdots & & \ddots & \\ \frac{\partial \mathbf{G}_N}{\partial \mathbf{U}_1}(t_n,\tilde{\mathbf{Y}}^n) & \frac{\partial \mathbf{G}_N}{\partial \mathbf{U}_2}(t_n,\tilde{\mathbf{Y}}^n) & \dots & \frac{\partial \mathbf{G}_N}{\partial \mathbf{U}_N}(t_n,\tilde{\mathbf{Y}}^n) \end{bmatrix}. \quad (8.10)$$

In Equation (8.5), we do not define the function G_j, but only provide the evaluated values of this function at $\left(t_n, \tilde{\mathbf{Y}}^n\right)$. Based on the construction of the face-centered fluxes $\mathbf{F}^{IMPICE}_{j+\frac{1}{2}}$ for the IMPICE method as described by Equation (8.5), we will approximate the value of $\frac{\partial \mathbf{G}_j}{\partial \mathbf{U}_k}(t_n,\tilde{\mathbf{Y}}^n)$ using the following equation:

$$\frac{\partial \mathbf{G}_j}{\partial \mathbf{U}_k}(t_n,\tilde{\mathbf{Y}}^n) = \frac{1}{\Delta x}\left(\frac{\partial \mathbf{F}^{IMPICE}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}(t_n,\tilde{\mathbf{Y}}^n) - \frac{\partial \mathbf{F}^{IMPICE}_{j-\frac{1}{2}}}{\partial \mathbf{U}_k}(t_n,\tilde{\mathbf{Y}}^n)\right), \quad (8.11)$$

where the approximation of $\frac{\partial \mathbf{F}^{IMPICE}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}(t_n,\tilde{\mathbf{Y}}^n)$ is discussed next.

Based on Equation (6.58), we approximate $\frac{\partial \mathbf{F}^{IMPICE}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}(t_n,\tilde{\mathbf{Y}}^n)$ using the following equation:

$$\frac{\partial \mathbf{F}^{IMPICE}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}\left(t_n,\tilde{\mathbf{Y}}^n\right) \approx u^*_{j+\frac{1}{2}}\frac{\partial \langle \mathbf{U}\rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} + \langle \mathbf{U}\rangle^n_{j+\frac{1}{2}}\frac{\partial u^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} + \begin{bmatrix} 0 \\ \frac{\partial p^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} \\ \frac{\partial p^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}u^*_{j+\frac{1}{2}} + p^*_{j+\frac{1}{2}}\frac{\partial u^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} \end{bmatrix}, \quad (8.12)$$

where the partial derivative terms on the right side of Equation (8.12) will be described below. These partial derivatives terms approximate the change in the corresponding quantities with respect to the change in the cell-centered conserved variables. We will discuss the approximations to the partial derivatives $\frac{\partial \langle \mathbf{U}\rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$, $\frac{\partial u^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$, and $\frac{\partial p^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ in Equation (8.12) by following the steps in the IMPICE method next.

8.2.1 Partial Derivatives of Variables at Cell-centers

The vector of cell-centered variables $\mathbf{W}$ of cell j is given by:

$$\mathbf{W}_j = [\rho_j, u_j, E_j, p_j]^T . \tag{8.13}$$

With $\mathbf{U}_j$ defined by Equation (8.2), the partial derivatives of $\mathbf{W}_j$ with respect to $\mathbf{U}_k$ where $k \neq j$ are zero vectors. For the case of $k = j$, the partial derivatives of $\mathbf{W}_j$ is given by:

$$\frac{\partial \mathbf{W}_j}{\partial \mathbf{U}_j} = \left[\frac{\partial (\rho_j)}{\partial \mathbf{U}_j}, \frac{\partial (u_j)}{\partial \mathbf{U}_j}, \frac{\partial (E_j)}{\partial \mathbf{U}_j}, \frac{\partial (p_j)}{\partial \mathbf{U}_j} \right]^T , \tag{8.14}$$

where the elements in the above vector are shown in the following equations:

$$\frac{\partial \rho_j}{\partial \mathbf{U}_j} = [1, 0, 0] , \tag{8.15}$$

$$\frac{\partial u_j}{\partial \mathbf{U}_j} = \left[\frac{-u_j}{\rho_j}, \frac{1}{\rho_j}, 0 \right] , \tag{8.16}$$

$$\frac{\partial E_j}{\partial \mathbf{U}_j} = \left[\frac{-E_j}{\rho_j}, 0, \frac{1}{\rho_j} \right] , \tag{8.17}$$

$$\frac{\partial p_j}{\partial \mathbf{U}_j} = \left[\frac{1}{2} (\gamma - 1) (u_j)^2 , \ -(\gamma - 1) u_j , \ (\gamma - 1) \right] . \tag{8.18}$$

We introduce the partial derivative notation $\frac{\partial \mathbf{W}_j^n}{\partial \mathbf{U}_j}$ which is defined as follows:

$$\frac{\partial \mathbf{W}_j^n}{\partial \mathbf{U}_j} = \frac{\partial \mathbf{W}_j}{\partial \mathbf{U}_j}(t_n). \tag{8.19}$$

We approximate the partial derivatives $\frac{\partial \mathbf{W}_j}{\partial \mathbf{U}_j}$ at t_n by evaluating the right side of Equations (8.16)–(8.18) using the cell-centered numerical solution $\tilde{\mathbf{Y}}^n$.

8.2.2 Partial Derivatives of Limited Local Reconstructed Variables at Face-centers

In the time integration of the IMPICE method, we first have the two states constructed at each face-center. These constructed states are used in the calculation of the face-centered fluxing velocity $u^*_{j+\frac{1}{2}}$. In order to approximate the change in fluxing velocity with respect to the change cell-centered conserved variables, $\frac{\partial u^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$, we first approximate the change in the constructed states with respect to these variables, denoted as $\frac{\partial \mathbf{W}^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ and $\frac{\partial \mathbf{W}^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$. In this dissertation, there have been two different forms of the face-centered constructed states introduced by Equations (6.44)–(6.45) and (7.10)–(7.11) based on the use of the different limiting processes. We will approximate $\dfrac{\partial \mathbf{W}^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ and $\dfrac{\partial \mathbf{W}^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ based on the MLP limiting process, in which the reconstructed left and right states are given by Equations (7.10)–(7.11); a similar derivation can be applied to the reconstructed states are given by Equations (6.44)–(6.45).

From Equation (7.10), we have:

$$\begin{aligned}
\frac{\partial \mathbf{W}^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_{j-1}} &= 0.5\boldsymbol{\phi}_{\mathbf{r}}\left(\mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}\right)\frac{\partial \mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}}{\partial \mathbf{U}_{j-1}}\left(\mathbf{W}^n_j - \mathbf{W}^n_{j-1}\right) \\
&\quad -0.5\boldsymbol{\phi}\left(\mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}\right)\frac{\partial \mathbf{W}^n_{j-1}}{\partial \mathbf{U_{j-1}}}, \qquad (8.20)\\
\frac{\partial \mathbf{W}^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_{j}} &= 0.5\boldsymbol{\phi}_{\mathbf{r}}\left(\mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}\right)\frac{\partial \mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}}{\partial \mathbf{U}_{j}}\left(\mathbf{W}^n_j - \mathbf{W}^n_{j-1}\right) \\
&\quad +\left[\mathbf{1} + 0.5\boldsymbol{\phi}\left(\mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}\right)\right]\frac{\partial \mathbf{W}^n_{j}}{\partial \mathbf{U}_{j}}, \qquad (8.21)\\
\frac{\partial \mathbf{W}^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_{j+1}} &= 0.5\boldsymbol{\phi}_{\mathbf{r}}\left(\mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}\right)\frac{\partial \mathbf{r}^{n(L)}_{\mathbf{W}+\frac{1}{2}}}{\partial \mathbf{U}_{j+1}}\left(\mathbf{W}^n_j - \mathbf{W}^n_{j-1}\right), \qquad (8.22)
\end{aligned}$$

where:

$$\mathbf{r}^{n(L)}_{\mathbf{W}_{j+\frac{1}{2}}} = \frac{\mathbf{W}^n_{j+1} - \mathbf{W}^n_j}{\mathbf{W}^n_j - \mathbf{W}^n_{j-1}} \qquad (8.23)$$

and $\boldsymbol{\phi}(\mathbf{r})$ is defined in [71] as follows:

$$\boldsymbol{\phi}(\mathbf{r}) = max(\mathbf{0}, min(\mathbf{2}, 2\mathbf{r}, \frac{\mathbf{1}+2\mathbf{r}}{3})). \qquad (8.24)$$

In the same way as in previous chapters, the vector multiplications and divisions in the above equations, and also for the rest of this chapter, are component-wise operations. The max and min functions in Equation (8.24) and the derivative of function $\boldsymbol{\phi}$ with respect to $\mathbf{r}$ in Equations (8.20)–(8.22) are also performed on each component of the vector. The derivative of function $\boldsymbol{\phi}$ with respect to $\mathbf{r}$ evaluated at a scalar quantity r is given by:

$$\boldsymbol{\phi}_{\mathbf{r}}(r) = \frac{\partial \phi(r)}{\partial r}, \qquad (8.25)$$

where $\phi(r) = max(0, min(2, 2r, \frac{(1+2r)}{3}))$. As shown in Figure 8.1, the limiting function $\phi(r)$ in

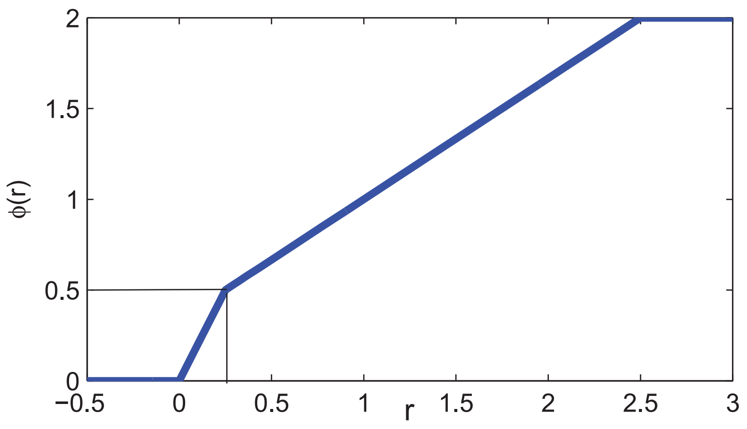


Figure 8.1. MLP3 limiting function $\phi(r)$.

Equation (8.25)is not differentiable everywhere. In particular, function $\phi(r)$ is not differentiable at $r = 0$, 0.25, and 2.5. However, $\phi(r)$ is both left and right differentiable at these points. The derivative of function $\phi(r)$ with respect to r at these values is chosen either the left derivative or the right derivative. Therefore, we have the following approximation:

$$\phi_{\mathbf{r}}(r) = \begin{cases} 0 & \text{if } (r \leq 0 \quad || \quad 2.5 \leq r), \\ 2 & \text{if } (0 < r < 0.25). \\ 2/3 & \text{if } (0.25 \leq r < 2.5). \end{cases} \tag{8.26}$$

We tested the IMPICE method with several differentiable limiters; the resulting method was nonmonotone when used with these tested differentiable limiters.

From Equation (8.23), we define the following partial derivatives:

$$\frac{\partial \mathbf{r}^{n(L)}_{\mathbf{W}_{j+\frac{1}{2}}}}{\partial \mathbf{U}_{j-1}} = \left(\frac{\mathbf{W}^n_{j+1} - \mathbf{W}^n_j}{(\mathbf{W}^n_j - \mathbf{W}^n_{j-1})^2} \right) \frac{\partial \mathbf{W}^n_{j-1}}{\partial \mathbf{U}_{j-1}}, \tag{8.27}$$

$$\frac{\partial \mathbf{r}^{n(L)}_{W_{j+\frac{1}{2}}}}{\partial \mathbf{U}_j} = \left(\frac{\mathbf{W}^n_{j+1} - \mathbf{W}^n_{j-1}}{(\mathbf{W}^n_j - \mathbf{W}^n_{j-1})^2} \right) \frac{\partial \mathbf{W}^n_j}{\partial \mathbf{U}_j}, \tag{8.28}$$

$$\frac{\partial \mathbf{r}^{n(L)}_{\mathbf{W}_{j+\frac{1}{2}}}}{\partial \mathbf{U}_{j+1}} = \left(\frac{1}{\mathbf{W}^n_j - \mathbf{W}^n_{j-1}} \right) \frac{\partial \mathbf{W}^n_{j+1}}{\partial \mathbf{U}_{j+1}}. \tag{8.29}$$

With the partial derivatives defined in Equations (8.26)–(8.29), we now can approximate the partial derivatives $\frac{\partial \mathbf{W}^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ in Equations (8.20) – (8.22) for $\mathbf{W}^{n(L)}_{j+\frac{1}{2}}$ defined by Equation (7.10). A similar derivation can be applied to the approximation of $\frac{\partial \mathbf{W}^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ for $\mathbf{W}^{n(R)}_{j+\frac{1}{2}}$ defined by

Equation (7.11). The partial derivatives of the constructed values of the speed of sound are defined using the partial derivatives of other constructed variables as follows:

$$\frac{\partial c^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\gamma^{\frac{1}{2}}}{2}\left[\frac{\rho^{n(L)}_{j+\frac{1}{2}}\frac{\partial p^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} - p^{n(L)}_{j+\frac{1}{2}}\frac{\partial \rho^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}}{\rho^{n(L)}_{j+\frac{1}{2}}\sqrt{\rho^{n(L)}_{j+\frac{1}{2}}p^{n(L)}_{j+\frac{1}{2}}}}\right], \quad \frac{\partial c^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\gamma^{\frac{1}{2}}}{2}\left[\frac{\rho^{n(R)}_{j+\frac{1}{2}}\frac{\partial p^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} - p^{n(R)}_{j+\frac{1}{2}}\frac{\partial \rho^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}}{\rho^{n(R)}_{j+\frac{1}{2}}\sqrt{\rho^{n(R)}_{j+\frac{1}{2}}p^{n(R)}_{j+\frac{1}{2}}}}\right]. \tag{8.30}$$

The above derivation is based on the equation of speed of sound in Equation (3.6). The constructed face-centered values are then used to construct the generalized Riemann problem (6.49) with the initial condition given by Equation (6.50). The partial derivatives of the HLL Riemann solution at face-centers are approximated next.

8.2.3 Partial Derivatives of the HLL Riemann Solution at Face-centers

The value of variables located at face-center is approximated using the HLL Riemann solver described by Equation (6.51). From Equation (6.51), we will approximate the change in the value of variables at face-center with respect to **Y**. In particular, the change in the face-centered velocity at t_n, $\frac{\partial u^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$, and the face-centered density at t_n, $\frac{\partial \rho^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$. The changes in the face-centered density and velocity corresponding to three different cases of the HLL Riemann solution are given as follows:

Case (i): $a^{n(L)}_{j+\frac{1}{2}} > 0$.

As shown in Equation (6.51), the face-centered velocity and density for this case are given by:

$$u^n_{j+\frac{1}{2}} = u^{n(L)}_{j+\frac{1}{2}}, \quad \rho^n_{j+\frac{1}{2}} = \rho^{n(L)}_{j+\frac{1}{2}}. \tag{8.31}$$

The partial derivatives of the face-centered velocity and density are then given by:

$$\frac{\partial u^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\partial u^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}, \quad \frac{\partial \rho^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\partial \rho^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}, \tag{8.32}$$

where $\frac{\partial u^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ and $\frac{\partial \rho^{n(L)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ are approximated using Equations (8.20) – (8.22).

Case (ii): $a^{n(R)}_{j+\frac{1}{2}} < 0$.

Similarly as in the previous case, the partial derivatives of the face-centered velocity and density for this case are approximated using the partial derivatives of the right constructed state as follows:

$$\frac{\partial u^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\partial u^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}, \quad \frac{\partial \rho^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\partial \rho^{n(R)}_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}. \tag{8.33}$$

Case (iii): $a^{n(L)}_{j+\frac{1}{2}} < 0 < a^{n(R)}_{j+\frac{1}{2}}$.

For this case, the face-centered velocity and density are defined using both the left and right constructed states. From Equation (6.51), we have:

$$\rho^n_{j+\frac{1}{2}} = \frac{(a_R\rho_R - a_L\rho_L) - (\rho_R u_R - \rho_L u_L)}{a_R - a_L}, \tag{8.34}$$

$$(\rho u)^n_{j+\frac{1}{2}} = \frac{(a_R\rho_R u_R - a_L\rho_L u_L) - \big(\rho_R(u_R)^2 + p_R - \rho_L(u_L)^2 - p_L\big)}{a_R - a_L}. \tag{8.35}$$

In the above equations, the variables are written in abbreviated forms where ρ_L and ρ_R represent $\rho^{n(L)}_{j+\frac{1}{2}}$ and $\rho^{n(R)}_{j+\frac{1}{2}}$ respectively, and also the same for all other face-centered variables at t_n. With a_L and a_R are defined by Equation (6.53), we can simplify Equation (8.34) as follows:

$$\rho^n_{j+\frac{1}{2}} = \frac{(c_R\rho_R + c_L\rho_L)}{(a_R - a_L)}. \tag{8.36}$$

The face-centered velocity at t_n is obtained from Equations (8.34) and (8.35) as follows:

$$u^n_{j+\frac{1}{2}} = \frac{(\rho u)^n_{j+\frac{1}{2}}}{\rho^n_{j+\frac{1}{2}}} = \frac{(c_R\rho_R u_R + c_L\rho_L u_L - p_R + p_L)}{(c_R\rho_R + c_L\rho_L)}\ . \tag{8.37}$$

As the partial derivatives with respect to $\mathbf{Y}$ of the terms on the right side of Equations (8.36) and (8.37) are previously defined in Equations (8.20)–(8.22), and Equation (8.30), we can determine the approximation to the partial derivatives of $\rho^n_{j+\frac{1}{2}}$ in Equation (8.36) and $u^n_{j+\frac{1}{2}}$ in Equation (8.37). These partial derivatives are denoted as $\frac{\partial \rho^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$ and $\frac{\partial u^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k}$.

8.2.4 Partial Derivatives of Fluxing Velocities at Face-centers

From Equation (6.30), we approximate the partial derivatives of the face-centered fluxing velocity using the following equation:

$$\frac{\partial u^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\partial u^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} - \sigma \left[\frac{1}{\rho^n_{j+\frac{1}{2}}} \frac{\partial \left(p^{n+\frac{1}{2}}_{j+1} - p^{n+\frac{1}{2}}_j \right)}{\partial \mathbf{U}_k} - \frac{\left(p^{n+\frac{1}{2}}_{j+1} - p^{n+\frac{1}{2}}_j \right)}{\left(\rho^n_{j+\frac{1}{2}} \right)^2} \frac{\partial \rho^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} \right], \tag{8.38}$$

where $\sigma = \dfrac{\Delta t}{2\Delta x}$. In Equation (8.38), $p^{n+\frac{1}{2}}_j$ is the cell-centered pressure at $t_{n+\frac{1}{2}}$. Following the calculation of the explicit "pressure corrector" discussed in Section 6.4.2, the cell-centered pressure $p^{n+\frac{1}{2}}_j$ is approximated using the following equation:

$$p^{n+\frac{1}{2}}_j = p^n_j - \sigma \left(c^2 \rho \right)^n_j \left(u^n_{j+\frac{1}{2}} - u^n_{j-\frac{1}{2}} \right). \tag{8.39}$$

From Equation (8.39) and the equation of state, we have:

$$p^{n+\frac{1}{2}}_j = p^n_j - \sigma \gamma p^n_j \left(u^n_{j+\frac{1}{2}} - u^n_{j-\frac{1}{2}} \right) = p^n_j \left[1 - \sigma \gamma \left(u^n_{j+\frac{1}{2}} - u^n_{j-\frac{1}{2}} \right) \right]. \tag{8.40}$$

The partial derivatives of $p^{n+\frac{1}{2}}_j$ with respect to $\mathbf{Y}$ are then approximated as follows:

$$\frac{\partial p^{n+\frac{1}{2}}_j}{\partial \mathbf{U}_k} = \frac{\partial p^n_j}{\partial \mathbf{U}_k} \left[1 - \sigma \gamma \left(u^n_{j+\frac{1}{2}} - u^n_{j-\frac{1}{2}} \right) \right] - \sigma \gamma p^n_j \left(\frac{\partial u^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} - \frac{\partial u^n_{j-\frac{1}{2}}}{\partial \mathbf{U}_k} \right). \tag{8.41}$$

Since the partial derivatives on the right side of the above equation are previously defined in Sections 8.2.1 and 8.2.3, the partial derivatives of the cell-centered pressure, $\dfrac{\partial p^{n+\frac{1}{2}}_j}{\partial \mathbf{U}_k}$, are also now defined.

8.2.5 Partial Derivatives of Pressures at Face-centers

From Equation (6.9), the partial derivatives with respect to $\mathbf{Y}$ of the face-centered pressure $\partial p^*_{j+\frac{1}{2}}$ may be approximated using the following equation:

$$\begin{aligned} \frac{\partial p^*_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{1}{\left(\rho^n_j + \rho^n_{j+1} \right)} & \left[\rho^n_j \frac{\partial p^{n+\frac{1}{2}}_{j+1}}{\partial \mathbf{U}_k} + \rho^n_{j+1} \frac{\partial p^{n+\frac{1}{2}}_j}{\partial \mathbf{U}_k} + p^{n+\frac{1}{2}}_{j+1} \frac{\partial \rho^n_j}{\partial \mathbf{U}_k} + p^{n+\frac{1}{2}}_j \frac{\partial \rho^n_{j+1}}{\partial \mathbf{U}_k} \right] \\ & - \frac{\left(\rho^n_j p^{n+\frac{1}{2}}_{j+1} + \rho^n_{j+1} p^{n+\frac{1}{2}}_j \right)}{\left(\rho^n_j + \rho^n_{j+1} \right)^2} \left[\frac{\partial \rho^n_j}{\partial \mathbf{U}_k} + \frac{\partial \rho^n_{j+1}}{\partial \mathbf{U}_k} \right], \end{aligned} \tag{8.42}$$

where $\dfrac{\partial p^{n+\frac{1}{2}}_j}{\partial \mathbf{U}_k}$, $\dfrac{\partial p^{n+\frac{1}{2}}_{j+1}}{\partial \mathbf{U}_k}$, $\dfrac{\partial \rho^n_j}{\partial \mathbf{U}_k}$, and $\dfrac{\partial \rho^n_{j+1}}{\partial \mathbf{U}_k}$ are approximated using Equations (8.16) and (8.41).

8.2.6 Partial Derivatives of Advected Quantities at Face-centers

For IMPICE with first-order advection as defined in Chapter 6, the advected quantities $\langle \mathbf{U} \rangle^n_{j+\frac{1}{2}}$ are defined by Equation (6.17). The partial derivatives of the advected quantities with respect to $\mathbf{Y}$ are then approximated using the following equation:

$$\frac{\partial \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \begin{cases} \mathbf{I} & \text{if} \quad \left(u^*_{j+\frac{1}{2}} < 0\right) \quad \text{and} \quad (k = j+1) \\ \mathbf{I} & \text{if} \quad \left(u^*_{j+\frac{1}{2}} \geq 0\right) \quad \text{and} \quad (k = j) \\ \mathbf{0} & \text{otherwise,} \end{cases} \tag{8.43}$$

where $\mathbf{0}$ is 3×3 zero matrix and $\mathbf{I}$ is 3×3 identity matrix.

For IMPICE with second-order advection discussed in Section 6.8, we need to approximate the following partial derivatives:

$$\frac{\partial \langle \mathbf{U} \rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \begin{bmatrix} \dfrac{\partial \langle \rho \rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} \\ \dfrac{\partial \langle \rho u \rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} \\ \dfrac{\partial \langle \rho E \rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} \end{bmatrix}, \tag{8.44}$$

where $\langle \rho \rangle^n_{j+\frac{1}{2}}$, $\langle \rho u \rangle^n_{j+\frac{1}{2}}$, and $\langle \rho \rangle^n_{j+\frac{1}{2}}$ are defined by Equations (6.76)–(6.78).

From Equation (6.76), we have:

$$\begin{aligned} \frac{\partial \langle \rho \rangle^n_{j+\frac{1}{2}}}{\partial \mathbf{U}_k} = \frac{\partial \rho^n_j}{\partial \mathbf{U}_k} \left[1 - \frac{\Delta t}{2} \left(\frac{\partial u}{\partial x} \right)^n_j \right] + \left(\frac{\Delta x}{2} - u^n_j \frac{\Delta t}{2} \right) \frac{\partial \left(\frac{\partial \rho}{\partial x} \right)^n_j}{\partial \mathbf{U}_k} \\ - \frac{\Delta t}{2} \left(\frac{\partial \rho}{\partial x} \right)^n_j \frac{\partial u^n_j}{\partial \mathbf{U}_k} - \frac{\Delta t}{2} \rho^n_j \frac{\partial \left(\frac{\partial u}{\partial x} \right)^n_j}{\partial \mathbf{U}_k}. \end{aligned} \tag{8.45}$$

Since $\left(\frac{\partial \rho}{\partial x} \right)^n_j$ and $\left(\frac{\partial u}{\partial x} \right)^n_j$ in Equation (8.45) are calculated using Equation (6.79), the partial derivatives with respect to $\mathbf{Y}$ of these terms are approximated using the following equation:

$$\frac{\partial \left(\frac{\partial \mathbf{W}}{\partial x} \right)^n_j}{\partial \mathbf{U}_k} = \text{minmod}(z_1, z_2, z_3) = \begin{cases} \dfrac{\partial z_j}{\partial \mathbf{U}_k} & \mathit{if} \quad \text{minmod}(z_1, z_2, z_3) = z_j \\ 0 & \mathit{otherwise} \end{cases}. \tag{8.46}$$

Equations (8.16), (8.17), and (8.46) are used to approximate the partial derivatives of $\dfrac{\partial\langle\rho\rangle^n_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}$ in Equation (8.45). From Equation (6.77), we have:

$$\frac{\partial\langle\rho u\rangle^n_{j+\frac{1}{2}}}{\partial\mathbf{U}_k} = u^n_j\frac{\partial\langle\rho\rangle^n_{j+\frac{1}{2}}}{\partial\mathbf{U}_k} + \langle\rho\rangle^n_{j+\frac{1}{2}}\frac{\partial u^n_j}{\partial\mathbf{U}_k} + \left(\frac{\Delta x}{2} - u^n_j\frac{\Delta t}{2}\right)\left(\frac{\partial u}{\partial x}\right)^n_j\frac{\partial\rho^n_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}$$
$$-\frac{\Delta t}{2}\rho^n_{j+\frac{1}{2}}\left(u^n_j\frac{\partial\left(\dfrac{\partial u}{\partial x}\right)^n_j}{\partial\mathbf{U}_k} + \left(\frac{\partial u}{\partial x}\right)^n_j\frac{\partial\rho^n_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}\right). \tag{8.47}$$

Note that all the partial derivative terms on the right side of Equation (8.47) have all been derived in this section.

Similarly, we can derive the partial derivatives of $\langle\rho E\rangle^n_{j+\frac{1}{2}}$ in Equation (6.78), as denoted as $\dfrac{\partial\langle\rho E\rangle^n_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}$. We are now able to estimate $\dfrac{\partial\mathbf{F}^{IMPICE}_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}\left(t_n, \tilde{\mathbf{Y}}^n\right)$ in Equation (8.12) using the proposed approximations to the partial derivatives $\dfrac{\partial\langle\mathbf{U}\rangle^n_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}$, $\dfrac{\partial u^*_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}$, and $\dfrac{\partial p^*_{j+\frac{1}{2}}}{\partial\mathbf{U}_k}$ in this section. Therefore, the approximation to partial derivatives $\dfrac{\partial\mathbf{G}_j}{\partial\mathbf{U}_k}(t_n, \tilde{\mathbf{Y}}^n)$ in Equation (8.11) is also defined. This approximation completes the definition of $\mathbf{J}(t_n, \tilde{\mathbf{Y}}^n)$ in Equation (8.10) and therefore the definition of the adjoint problem (8.7).

8.3 Local Error and Truncation Error Estimation

8.3.1 Local Error

The local ODE problem corresponding to the ODE system given by Equation (8.3) is as follows:

$$\begin{cases}\dot{\mathbf{Z}}_{n+1}(t) = \mathbf{G}(t, \tilde{\mathbf{Y}}^n), & t \in [t_n, t_{n+1}],\\ \mathbf{Z}_{n+1}(t_n) = \tilde{\mathbf{Y}}^n,\end{cases} \tag{8.48}$$

where $\tilde{\mathbf{Y}}^n = [\mathbf{U}^n_1, \mathbf{U}^n_2, \mathbf{U}^n_3, ..., \mathbf{U}^n_N,]^T$, and $\mathbf{G}(t, \tilde{\mathbf{Y}}^n)$ is defined by Equations (8.4) and (8.5). From Equations (6.57) and (8.5), the time integration solution of the local problem in (8.48) is given by:

$$\tilde{\mathbf{Y}}^{n+1} = \tilde{\mathbf{Y}}^n - \Delta t\mathbf{G}(t, \tilde{\mathbf{Y}}^n). \tag{8.49}$$

From Equations (8.48) and (8.49), the ODE local error is second-order in Δt. The local error can be then estimated using Richardson's extrapolation. The Richardson's extrapolation method for estimating the local error of order p on the single time step $[t_n, t_{n+1}]$ includes the following steps:

- Perform one step of the IMPICE method with stepsize $\Delta t = t_{n+1} - t_n$ to obtain the solution $\tilde{\mathbf{Y}}^{n+1}$ at t_{n+1}.
- Perform two consecutive steps of the IMPICE method with stepsize of $\frac{\Delta t}{2}$ to obtain the solution $\hat{\mathbf{Y}}^{n+1}$ at t_{n+1}.
- Estimate the local error for the local solution $\tilde{\mathbf{Y}}^{n+1}$ using the following equation:

$$\mathbf{le}\left(t_{n+1}; t_n, \tilde{\mathbf{Y}}^n\right) = \mathbf{Z}_{n+1}(t_{n+1}) - \tilde{\mathbf{Y}}^{n+1} = \frac{2^p}{2^p - 1}\left(\hat{\mathbf{Y}}^{n+1} - \tilde{\mathbf{Y}}^{n+1}\right). \tag{8.50}$$

The above estimate of the local time integration error is used in the adjoint-based error estimation of the one-dimensional IMPICE method discussed in Section 8.1.

8.3.2 Truncation Error

Richardson extrapolation has long been used to estimate the spatial truncation error in the method of lines for PDEs. The method was used by many different authors, e.g., Berzins [13]. In this approach, the spatial truncation error is estimated based on the obtained numerical solutions on coexisting different meshes. Assume that the system of Euler equations is also solved on the "coarse" mesh defined at points $\{x_j \ : \ j = 1, 3, 5...\}$ and p is the order of the spatial discretization. In order to distinguish the meshes, we use Ω_h to denote the "fine" mesh and Ω_H to denote the "coarse" mesh. In Section 4.5.3.2, we summarized the steps to estimate the spatial truncation error discussed in Berzins [13] for the method of lines with second-order spatial discretization ($p = 2$).

For the case the spatial discretization is order of p in general, let $\mathbf{Y}^h_H(t)$ and $\dot{\mathbf{Y}}^h_H(t)$ be the restriction of the numerical solution $\mathbf{Y}_h(t)$ and the numerical derivative $\dot{\mathbf{Y}}_h(t)$ from the "fine" mesh to the "coarse" mesh. The truncation error for the "coarse" mesh is then obtained by evaluating the following equation as proposed in Berzins [13]:

$$\mathbf{TE}_H(t) = \frac{2^p}{2^p - 1}\left[\dot{\mathbf{Y}}_H(t) - \mathbf{G}_H(t, U^h_H(t))\right] + \frac{2^p}{2^p - 1}\left[\dot{\mathbf{et}}_H(t) - \frac{\partial \mathbf{G}_H}{\partial \mathbf{Y}_H(t)}\mathbf{et}_H(t)\right] \tag{8.51}$$

The trunction error at the grid nodes that belong to both "fine" mesh and "coarse" mesh is then given by:

$$[\mathbf{TE}_h(t)]_{2i-1} = \frac{1}{2^p}[\mathbf{TE}_H(t)]_i. \tag{8.52}$$

The truncation error at grid nodes that are in “fine” mesh but not in “coarse” mesh is then obtained by extrapolating using the following equation:

$$[\mathbf{TE}_h(t)]_{2i} = \frac{1}{2^{p+1}}([\mathbf{TE}_H(t]_i + [\mathbf{TE}_H(t)]_{i+1}). \tag{8.53}$$

The calculation given by Equation (8.51) could yield a better estimation of the spatial truncation error if the time integration error $\mathbf{et}_H(t)$ is available. For the spatial error-dominated problems, we may simplify the estimation of the spatial truncation error as follows:

$$\mathbf{TE}_H(t) = \frac{2^p}{2^p - 1}\left[\dot{\mathbf{Y}}_H(t) - \mathbf{G}_H(t, U_H^h(t))\right]. \tag{8.54}$$

It is worth mentioning the observation of Debrabant and Lang [27] that, based on experiments, even in the case when the time integration error was not small, using a time error estimate does not yield a significantly better approximation of the spatial truncation error.

8.4 Numerical Results

8.4.1 Numerical Results of Adjoint-based Error Estimate for the One-dimensional IMPICE Method with First-order Advection

We applied the adjoint-based error approach discussed in Chapter 4 and this chapter to estimate the temporal error, the spatial error, and the overall error in the numerical solutions of the IMPICE method with first-order advection to the advection problems in Section 7.7 and the test problems in Table 6.1 in Section 6.5. The overall error, the temporal error, and the spatial are estimated using Equations (8.6), (8.8), and (8.9) respectively where the estimate of the local error and the spatial truncation error in these equations are discussed in Section 8.3. The spatial truncation error is assumed to be first-order. In Figures 8.2–8.8, the adjoint-based estimate of the overall error in density of a particular numerical solution to each test problem is compared against the true overall error. Figure 8.2 shows an accurate estimate of the overall error in the region with smooth true solution. Figure 8.3 shows a less accurate estimate of the overall error in the region with discontinuous true solution. It is known that the order of spatial discretization decreases at discontinuities, but we did not consider this in the estimation of the spatial truncation error. This is not an easy problem to address, but we will look into the problem in future work. The same problem with estimating the error in the region with discontinuous true solution for the numerical solutions of test problems **P1**–**P5** is shown in Figures 8.4–8.8. In Figure 8.4, we also included the error estimate for the specific momentum and specific total energy to show how the adjoint-based error estimate works for these quantities.

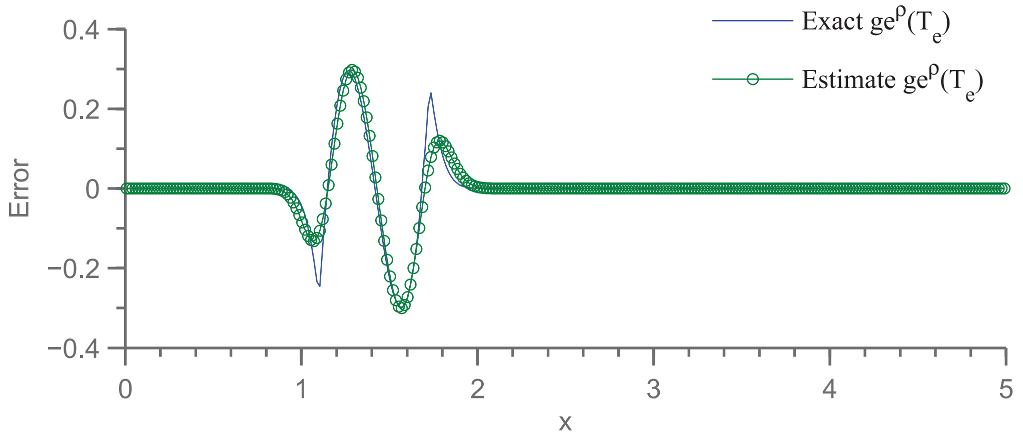


Figure 8.2. The adjoint-based estimate of the overall error in density of a numerical solution to test problem in Section 7.7.1 is compared against its true overall error. This numerical solution is obtained using the IMPICE method with first-order advection, $C_{cfl} = 0.2$, and $N = 300$ (cells).

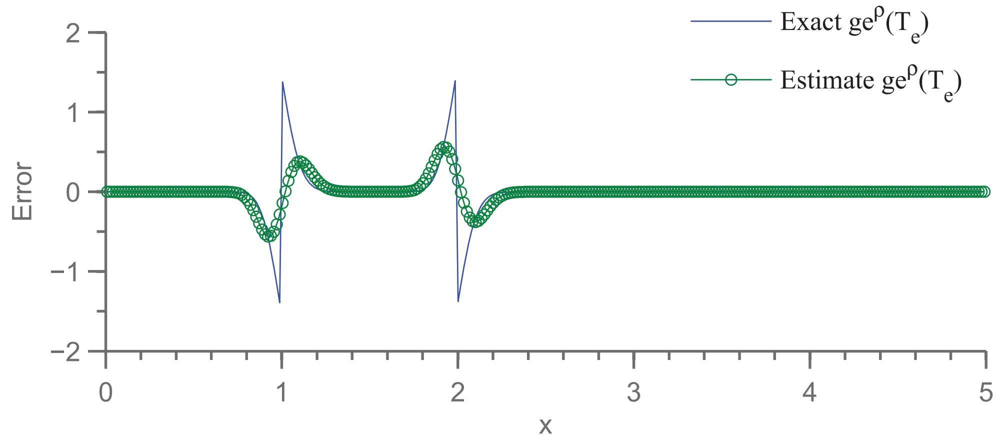


Figure 8.3. The adjoint-based estimate of the overall error in density of a numerical solution to test problem in Section 7.7.2 is compared against its true overall error. This numerical solution is obtained using the IMPICE method with first-order advection, $C_{cfl} = 0.2$, and $N = 300$ (cells).

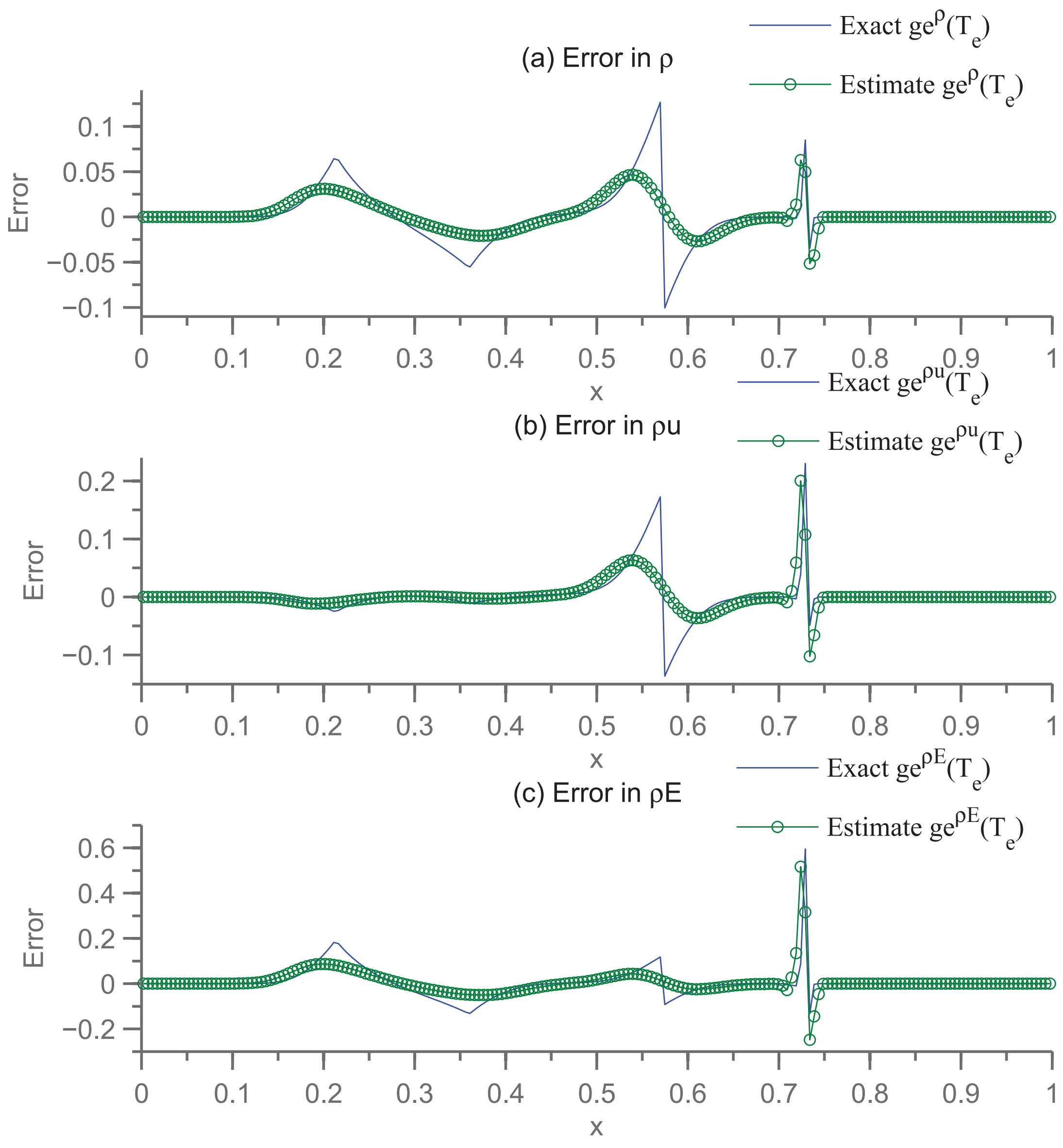


Figure 8.4. The adjoint-based estimate of the overall error in (a) density; (b) specific momentum; and (c) specific total energy of a numerical solution to test problem **P1** is compared against the true overall error. This numerical solution is obtained using the IMPICE method with first-order advection, $C_{cfl} = 0.2$, and $N = 200$ (cells).

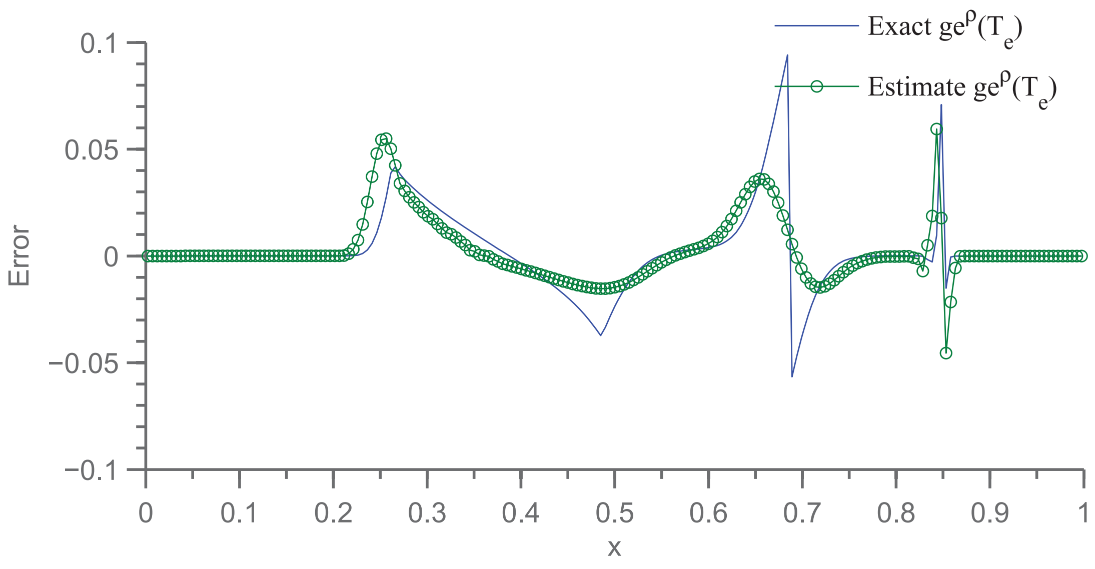


Figure 8.5. The adjoint-based estimate of the overall error in density of a numerical solution to test problem **P2** is compared against its true overall error. This numerical solution is obtained using the IMPICE method with first-order advection, $C_{cfl} = 0.2$, and $N = 200$ (cells).

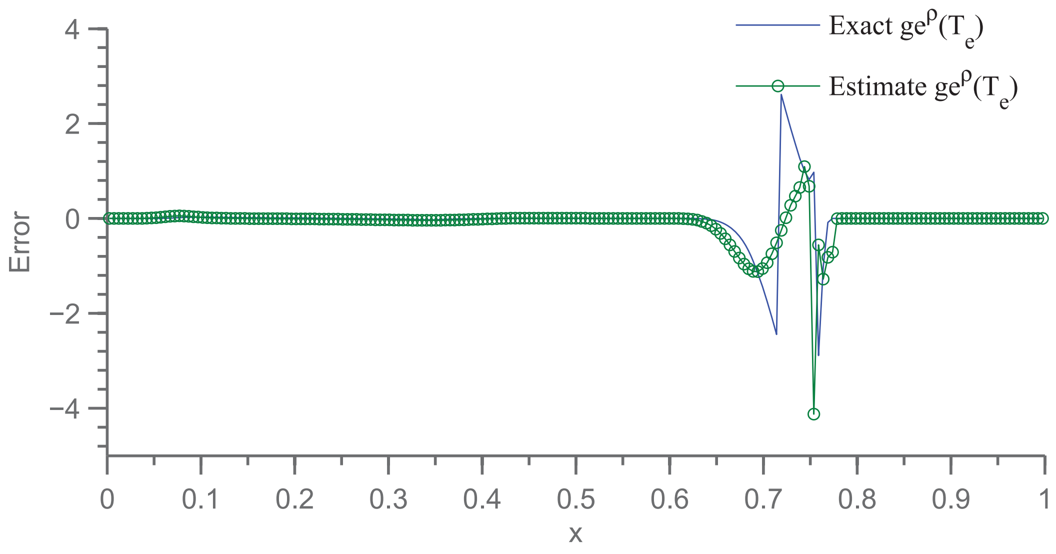


Figure 8.6. The adjoint-based estimate of the overall error in density of a numerical solution to test problem **P3** is compared against its true overall error. This numerical solution is obtained using the IMPICE method with first-order advection, $C_{cfl} = 0.2$, and $N = 200$ (cells).

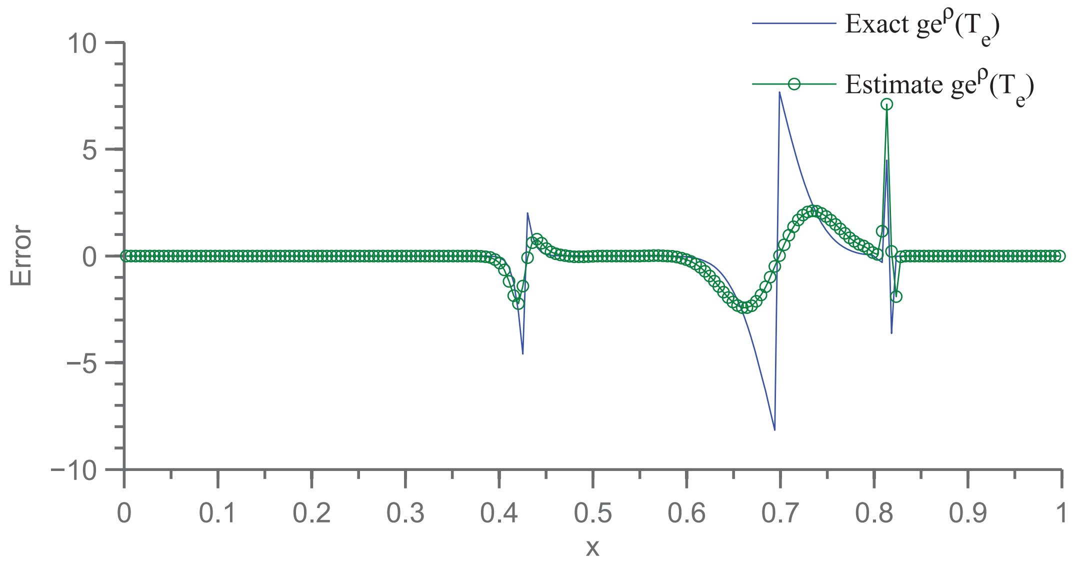


Figure 8.7. The adjoint-based estimate of the overall error in density of a numerical solution to test problem **P4** is compared against its true overall error. This numerical solution is obtained using the IMPICE method with first-order advection, $C_{cfl} = 0.2$, and $N = 200$ (cells).

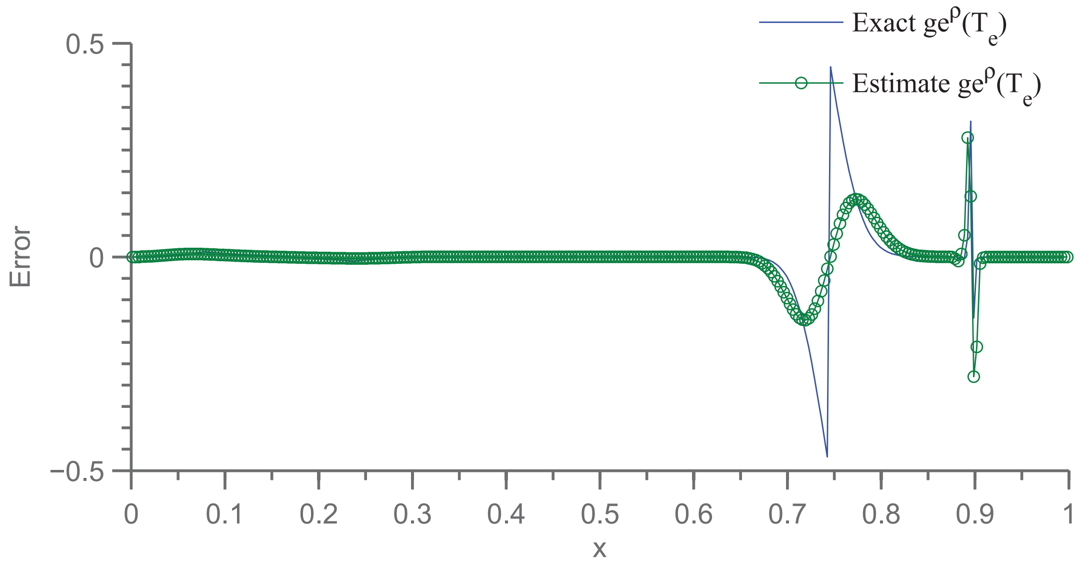


Figure 8.8. The adjoint-based estimate of the overall error in density of a numerical solution to test problem **P5** is compared against its true overall error. This numerical solution is obtained using the IMPICE method with first-order advection, $C_{cfl} = 0.2$, and $N = 300$ (cells).

We show in Table 8.1 the error indices when estimating the time integration error, the spatial error, and the overall error of the aforementioned tested problems. The results in Table 8.1 show that the error indices for the time integration error approach the value of one. The results in this table also show that an acceptable estimate of the spatial error is provided for all the tested problems. However, the error indices of the spatial error estimate for these tested problems are not so close to one due to the existence of discontinuities in the solution of these problems.

Table 8.1. Error indices $eindex(\mathbf{et}(Te))$, $eindex(\mathbf{et}(Te))$, and $eindex(\mathbf{ge}(Te))$ of the estimated adjoint-based global errors for numerical solutions to test problems discussed in Section 7.7 and Section 6.5. The numerical solutions to these problems are obtained from the one-dimensional IMPICE method with first-order advection and $C_{cfl} = 0.2$.

		$eindex(\mathbf{et}^{\mathbf{U}}(T_e))$			$eindex(\mathbf{es}^{\mathbf{U}}(T_e))$			$eindex(\mathbf{ge}^{\mathbf{U}}(T_e))$		
	N	ρ	ρu	ρE	ρ	ρu	ρE	ρ	ρu	ρE
Advection1	100	1.03	1.03	1.03	0.68	0.68	0.68	0.76	0.76	0.76
	200	1.03	1.03	1.03	0.87	0.87	0.87	0.85	0.85	0.85
	300	1.02	1.02	1.02	0.96	0.96	0.96	0.96	0.96	0.96
	400	1.02	1.02	1.02	1.01	1.01	1.01	1.01	1.01	1.01
Advection2	100	1.02	1.02	1.02	0.70	0.70	0.70	0.77	0.77	0.77
	200	1.02	1.02	1.02	0.68	0.68	0.68	0.75	0.75	0.75
	300	1.02	1.02	1.02	0.67	0.67	0.67	0.74	0.74	0.74
	400	1.02	1.02	1.02	0.67	0.67	0.67	0.74	0.74	0.74
P1	100	0.92	0.91	0.88	0.68	0.83	0.87	0.68	0.87	0.86
	200	0.95	0.94	0.91	0.70	0.77	0.82	0.70	0.77	0.82
	300	0.96	0.95	0.93	0.71	0.79	0.86	0.71	0.78	0.86
	400	0.96	0.96	0.96	0.70	0.75	0.81	0.70	0.74	0.81
P2	100	0.93	0.93	0.93	0.80	0.98	1.02	0.81	0.97	1.01
	200	0.94	0.95	0.93	0.79	0.93	0.99	0.79	0.92	0.98
	300	0.95	0.95	0.94	0.81	0.94	1.02	0.81	0.93	1.01
	400	0.95	0.96	0.94	0.79	0.90	0.98	0.79	0.90	0.98
P3	100	0.94	0.96	0.93	0.94	0.85	1.32	0.96	0.88	1.29
	200	0.96	0.97	0.94	0.77	0.73	1.08	0.78	0.75	1.10
	300	0.96	0.97	0.95	0.70	0.68	0.99	0.71	0.69	1.01
	400	0.97	0.98	0.95	0.69	0.66	0.95	0.69	0.67	0.96
P4	200	0.99	0.85	0.89	0.63	0.69	0.93	0.63	0.68	0.93
	400	0.91	0.89	0.86	0.65	0.73	1.02	0.68	0.76	1.02
	600	0.89	0.87	0.85	0.65	0.72	0.94	0.67	0.75	0.95
	800	0.90	0.90	0.84	0.66	0.75	0.98	0.68	0.77	0.98
P5	100	0.90	0.89	0.87	0.65	0.70	0.77	0.64	0.68	0.76
	200	0.92	0.94	0.90	0.68	0.76	0.97	0.68	0.76	0.97
	300	0.94	0.94	0.92	0.68	0.73	0.94	0.68	0.73	0.94
	400	0.94	0.94	0.92	0.65	0.68	0.86	0.65	0.67	0.86

8.4.2 Numerical Results of Adjoint-based Error Estimate for the One-dimensional IMPICE Method with Second-order Advection

We applied the adjoint-based error approach discussed in Chapter 4 and this chapter to estimate the temporal error, the spatial error, and the overall error in the numerical solutions of the IMPICE method with second-order advection to the advection problems in Section 7.7 and the test problems in Table 6.1 in Section 6.5. The overall error, the temporal error, and the spatial are estimated using Equations (8.6), (8.8), and (8.9) respectively where the estimate of the local error and the spatial truncation error in these equations are discussed in Section 8.3. The spatial truncation error is assumed to be second-order. Figure 8.9 shows an accurate estimate of the overall error in the region with smooth true solution. Figure 8.10 shows a less accurate estimate of the overall error in the region with discontinuous true solution.

We show the error indices of the estimates of the time integration error, the spatial error, and the overall error in Table 8.2. This table shows an accurate estimate of the time integration error when the error indices for the time integration error approach the value of one. The error indices of the spatial error for the second-order advection method are further from one compared to the error indices of the spatial error for the first-order advection method. As stated in Section 8.4.1, there is a drop in the order of the spatial discretization at discontinuities. For the first-order advection, the order of the spatial discretization drops from one to zero. For the

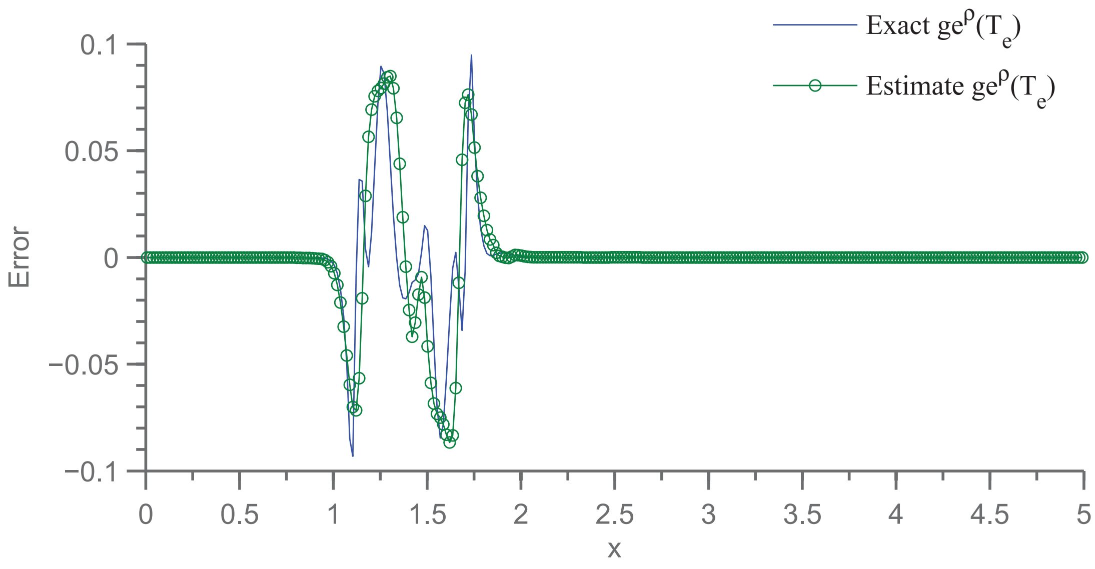


Figure 8.9. The adjoint-based estimate of the overall error in density of a numerical solution to test problem in Section 7.7.1 is compared against its true overall error. This numerical solution is obtained using the IMPICE method with second-order advection, $C_{cfl} = 0.2$, and $N = 300$ (cells).

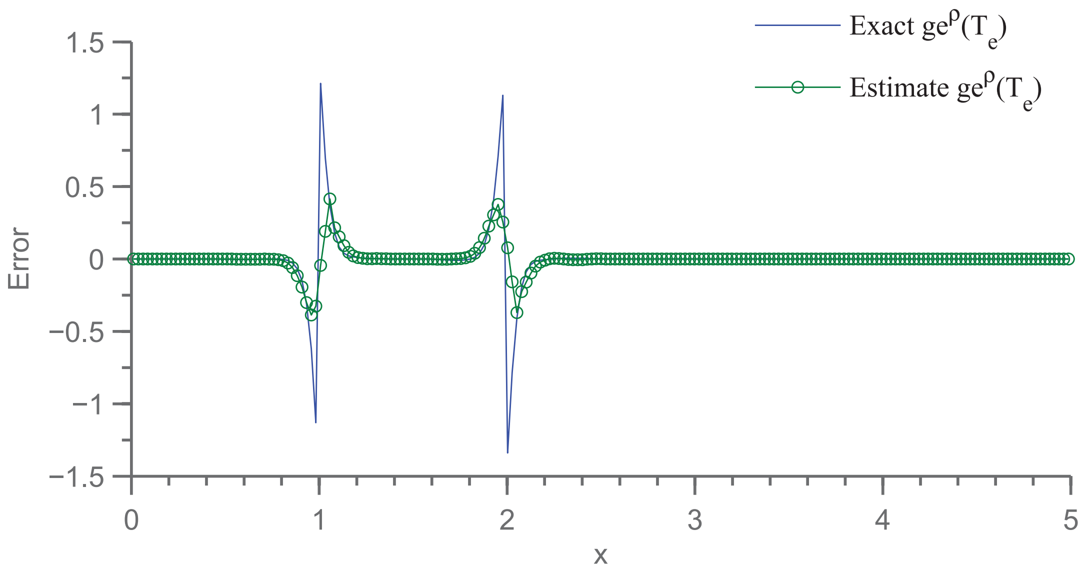


Figure 8.10. The adjoint-based estimate of the overall error in density of a numerical solution to test problem in Section 7.7.2 is compared against its true overall error. This numerical solution is obtained using the IMPICE method with second-order advection, $C_{cfl} = 0.2$, and $N = 300$ (cells).

second-order advection, the order of the spatial discretization may drop from two to zero. This results in an underestimation of the spatial truncation error in the region with discontinuous true solution in the IMPICE method with second-order advection. This explains why the error indices of the spatial error for the second-order advection method are further from one compared to the error indices of the spatial error for the first-order advection method.

We also applied the adjoint-based error estimate to the numerical solution of the Shu and Osher problem discussed in Section 6.5. We attempted to obtain the projected exact solution for the Shu and Osher problem using the numerical solution and the estimated overall error and compared to the "exact solution" as discussed in Section 6.5. We show this comparison in Figure 8.11. In order to see the error estimate more clearly we show in Figure 8.12 a close-up picture of Figure 8.11 for the region where the numerical solution has a significant error. As shown in Figure 8.12, the adjoint-based error estimate discussed in this chapter has provided a projected exact solution for the Shu and Osher problem which is very close to the "exact solution".

8.5 Summary

We have presented in this chapter a computable error estimate for the numerical solutions of the system of Euler equations solved with the one-dimensional IMPICE method using the discrete adjoint-based approach discussed in Chapter 4. In this discrete adjoint-based approach

Table 8.2. Error indices $eindex(\mathbf{et}(Te))$, $eindex(\mathbf{et}(Te))$, and $eindex(\mathbf{ge}(Te))$ of the estimated adjoint-based global errors for numerical solutions to test problems discussed in Section 7.7 and Section 6.5. The numerical solutions to these problems are obtained from the one-dimensional IMPICE method with second-order advection and $C_{cfl} = 0.2$.

		$eindex(\mathbf{et}^{\mathbf{U}}(T_e))$			$eindex(\mathbf{es}^{\mathbf{U}}(T_e))$			$eindex(\mathbf{ge}^{\mathbf{U}}(T_e))$		
	N	ρ	ρu	ρE	ρ	ρu	ρE	ρ	ρu	ρE
Advection1	100	1.01	1.01	1.00	0.71	0.70	0.68	0.70	0.68	0.66
	200	1.01	1.01	1.01	1.16	1.12	1.07	1.17	1.13	1.08
	300	1.01	1.01	1.03	1.37	1.35	1.31	1.39	1.37	1.32
	400	1.01	1.01	1.01	1.31	1.31	1.28	1.32	1.31	1.28
Advection2	100	1.03	1.03	1.03	0.53	0.53	0.52	0.57	0.56	0.56
	200	1.02	1.03	1.03	0.54	0.54	0.55	0.53	0.53	0.54
	300	1.02	1.03	1.04	0.55	0.55	0.56	0.54	0.54	0.55
	400	1.03	1.03	1.05	0.56	0.57	0.54	0.60	0.60	0.62
P1	100	0.93	0.92	0.94	0.44	0.52	0.56	0.48	0.52	0.52
	200	0.96	0.94	0.95	0.43	0.50	0.48	0.47	0.50	0.52
	300	0.96	0.95	0.96	0.45	0.54	0.55	0.50	0.55	0.58
	400	0.97	0.96	0.96	0.43	0.53	0.48	0.49	0.52	0.53
P2	100	0.89	0.92	0.92	0.48	0.53	0.57	0.51	0.53	0.57
	200	0.90	0.91	0.91	0.49	0.53	0.56	0.51	0.53	0.56
	300	0.91	0.92	0.92	0.53	0.58	0.62	0.55	0.57	0.61
	400	0.91	0.92	0.92	0.51	0.56	0.59	0.52	0.55	0.59
P3	100	0.97	0.98	0.95	0.99	1.24	1.18	1.02	1.27	1.12
	200	0.99	0.97	0.95	1.24	1.31	1.69	1.27	1.36	1.65
	300	0.95	0.94	0.94	0.88	0.89	1.29	0.94	0.95	1.29
	400	0.93	0.94	0.93	0.87	1.09	1.48	0.88	1.13	1.46
P4	200	1.12	1.07	1.04	0.47	0.57	0.67	0.49	0.50	0.75
	400	1.25	1.45	1.35	0.57	0.67	0.80	0.58	0.62	0.93
	600	1.32	1.47	1.52	0.61	0.63	0.65	0.64	0.58	0.76
	800	1.30	1.50	1.51	0.64	0.74	0.84	0.66	0.69	0.97
P5	100	0.87	0.89	0.90	0.45	0.47	0.51	0.43	0.45	0.50
	200	0.90	0.92	0.91	0.54	0.60	0.73	0.53	0.59	0.63
	300	0.90	0.91	0.91	0.57	0.58	0.59	0.56	0.57	0.61
	400	0.90	0.91	0.91	0.52	0.51	0.58	0.52	0.51	0.61

for estimating the error for a numerical solution of a PDE problem, it is necessary to obtain the adjoint problem of the spatial discretized problem, the local time integration error, and the spatial truncation error. We have also shown in this chapter how to derive the adjoint problem by following closely the steps in the IMPICE method as well as how to apply the method of Richardson extrapolation to estimate the local time integration error and the spatial truncation error.

We tested the proposed error estimate to a set of test problems with known exact solution to assess the performance of the estimate method. The results have shown that the estimate

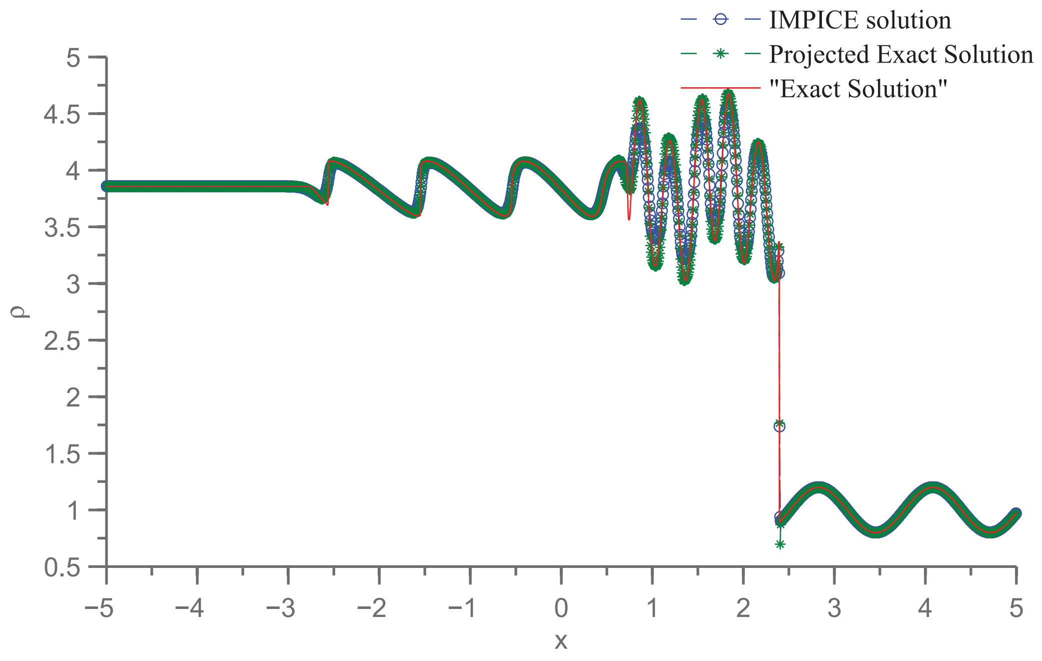


Figure 8.11. The numerical solution for Shu and Osher test problem obtained from the IMPICE method with second-order advection, $N = 1600$ (cells), and C_{cfl}=0.2; the "exact solution" of Shu and Osher test problem as discussed in Section 6.5; the projected exact solution obtained from adding the adjoint-based error estimate of the overall error to the numerical solution.

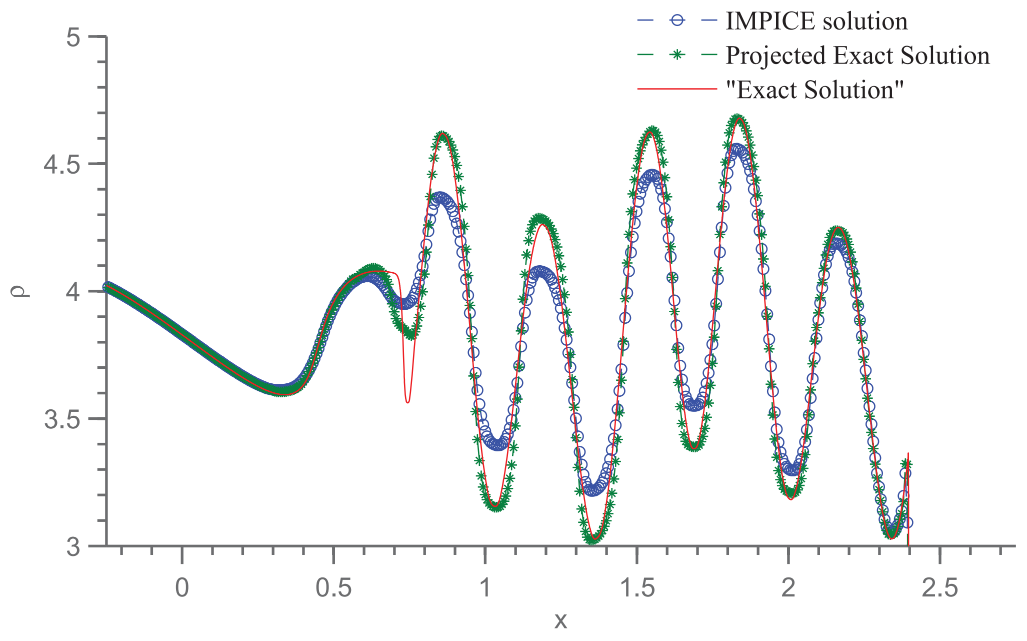


Figure 8.12. A close-up picture of Figure 8.11 for the region where the numerical solution has a significant error.

errors of the numerical solutions to the problems with smooth exact solutions are close to the true errors. For the case when the exact solution is discontinuous, the estimate of the error at the discontinuous region become less accurate due to the less accurate estimation of the spatial truncation error in this region. The less accurate estimation of the spatial truncation error at discontinuities results from the assumption that the order of the spatial discretization at discontinuities is the same as the order of the spatial discretization on smooth regions. In fact, the order of the spatial discretization at discontinuities is commonly less than its on smooth regions. In future, we may look into determining the order of spatial discretization at each grid point using numerical extrapolation to obtain a better estimate of the spatial truncation error and therefore a better error estimate for the numerical solutions.

As shown in this chapter, in order to estimate the global error on the spatial domain using the adjoint-based approach, we need to solve the adjoint system for every grid point on the domain. As mentioned in [21], this is the limitation with the adjoint-based global error estimation. However, the adjoint problem at each grid point can be solved independently; we can therefore take advantage of the availability of parallel computing to overcome this mentioned limitation.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

The Material Point Method (MPM) and the Implicit Coninuous-fluid Eulerian Method (ICE) make possible the simulation of a wide range of engineering applications. MPM, ICE, and the integrated combination of MPM and ICE (MPMICE) were implemented in the Uintah Computational Framework (UCF) at the Center for the Simulation of Accidental Fires and Explosions (C-SAFE) at the University of Utah to simulate accidental fires, explosions, and other multiphysics computational problems. This dissertation has provided an in-depth analysis and a possible improvement to some variation of MPM and ICE to better understand them. In particular, we have made several contributions to the study of these methods. First, we provide a study of various sources of time integration errors and spatial discretization errors in a variation of MPM proposed for gas dynamics. Though the MPM in UCF designed for solid dynamics, the analysis of this variation of MPM for gas dynamics in this dissertation is described in a way that can be applied to other versions of MPM. Second, we proposed an improvement to a version of ICE that is currently implemented in UCF. This improved version, is referred to as the Improved Production Implicit Continuous-fluid Eulerian Method (IMPICE), removes the discrepancies and eliminates the unphysical oscillations in the numerical solutions to the one-dimensional system of Euler equations for compressible flow problems. Third, we extended IMPICE to enable the solution of multidimensional compressible flow problems with potentially more complex geometries than just those consisting of hexahedral elements. Finally, we presented an adjoint-based approach to estimate errors in numerical solutions to partial differential equations and particularly we applied this approach to estimate errors in numerical solutions of IMPICE.

An error analysis for a variation of MPM proposed for gas dynamics is presented in Chapter 5. In order to maintain the stability of the method, we explored algorithms to ensure the positivity of the method's numerical approximations and prevent the creation of local extrema in various quantities. We investigated the relationship between the numerical errors and numbers of particles assigned to each cell. We analyzed errors in this numerical method including the time integration errors and the space discretization errors. These various error sources are introduced from projecting information from the particles onto the grid, mapping the movement at the grid points back onto the particles, and crossing the grid points of the particles.

The IMPICE method for compressible flow problems governed by the one-dimensional system of Euler equations is presented in Chapter 6. The use of a conservative scheme, slope limiters, and a simple approximate HLL Riemann solver all contributed to the elimination of oscillations in the numerical solutions of IMPICE. We have shown that the IMPICE method with a linear spatial and temporal discretization is expected to be first-order accurate in time and space. However, for the cases with discontinuities in their solutions, the order of accuracy in space is less than one. We have also shown in Chapter 6 how to use the method of temporal extrapolation and the higher-order advection in IMPICE to obtain a higher-order of accuracy in both time and space. While the method of temporal extrapolation successfully raises the order of accuracy to be second order in time, a less-than-expected order of accuracy in space is obtained from using the higher-order advection for the problems with discontinuities.

In Chapter 7, we extended the IMPICE method to solve the multidimensional system of Euler equations. In order to prevent the oscillations in the numerical solutions of the multidimensional system of Euler equations, it is necessary to apply a multidimensional limiting process to limit the gradients. We tested the implementation of the multidimensional IMPICE method on a suite of test problems for the system of Euler equations in multidimensional space where the obtained numerical solutions have shown the ability to capture shock waves. In order to allow IMPICE to solve problems in complex geometries, we implemented the method of cut cells which employes a new variation of the cell merging technique to overcome the "small cell problem" with the embedded boundary. The implementation of the method of cut cells is tested on the problem of shock reflection from a wedge. The contour lines of the IMPICE's numerical solution of the shock reflection problem are similar to the contour lines of numerical solutions of this problem obtained from many numerical methods in previous publications . In order to assess the accuracy of the embedded boundary implementation, we obtained the order of accuracy for the advection problem on the same bounded domain as the shock reflection problem.

In Chapter 4, we have presented the discrete adjoint-based approach for estimating spatial and temporal errors for the method-of-lines PDEs. We tested this approach on the numerical solutions of several ODE and PDE problems using the Backward Differentiation Formula (BDF) method implemented in DASSL DAL solver. These results showed that the adjoint-based approach accurately estimates both the temporal and spatial errors. Once more, this discrete adjoint-based error estimate was applied to estimate the errors in the numerical solutions obtained from the one-dimensional IMPICE method in Chapter 8. The challenging part in the application of the adjoint-based error estimate to the IMPICE method is the derivation of the adjoint problem. The derivation of the adjoint problem in this dissertation and the estimate of local error and truncation error using Richardson extrapolation have also yielded an accurate error estimation of the spatial error and temporal error for the numerical solutions obtained from the one-dimensional IMPICE method, especially for numerical solutions to the problems with smooth exact solutions. For the case when the exact solution is discontinuous, the estimate of the error at the discontinuous region become less accurate due to the less accurate estimation of the spatial truncation error in this region. In future, we will look into a more accurate estimation of the spatial truncation error at discontinuous region.

Overall, this dissertation has provided a comprehensive study of a variation of MPM for gas dynamics and an improved version of the Production ICE Method. The MPM method and the Production ICE method are two main numerical methods implemented in the UCF. The comprehensive analysis in this dissertation in combined with many current studies of MPM and ICE at C-SAFE would help to fully understand these two methods. The detailed analysis in this dissertation of different sources of errors in the numerical solutions of the proposed MPM version for gas dynamics allows it to be extended to other versions of MPM. The error estimate for the IMPICE method presented in this dissertation will help us to adaptively refine meshes to obtain more accurate solutions for the method or to keep the error in its numerical solutions under control. In the future, it will be necessary to extend the error estimate to work with the multidimensional IMPICE method. The critical part of this extension is to estimate the error on the embedded boundary. As the ICE method has also been proposed to simulate the multiphase flow problems, we would like to have extended the IMPICE method to solve these problems also, but again this is work for the future.

APPENDIX

A.1 IMPICE Method for Inviscid Burgers' Equation

Consider the one-dimensional Burgers' equation in the inviscid limit:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = 0, \tag{A.1}$$

with $f(u) = \frac{1}{2}u^2$ and initial data $u(x,0) = u_0(x)$, where $u_0(x)$ is a given function, $x \in \mathbb{R}$ and $0 < t \leq T_e$. The solution, $u(x/t; u^{n(L)}_{j+\frac{1}{2}}, u^{n(R)}_{j+\frac{1}{2}})$, to the Riemann problem of the Burgers' equation (A.1) with initial data:

$$u(x,t_n) = \begin{cases} u^{n(L)}_{j+\frac{1}{2}} & \text{if } \left(x < x_{j+\frac{1}{2}}\right) \\ u^{n(R)}_{j+\frac{1}{2}} & \text{if } \left(x > x_{j+\frac{1}{2}}\right), \end{cases} \tag{A.2}$$

at $x/t = 0$ is used in the IMPICE method. The approximate solution $u(0; u^{n(L)}_{j+\frac{1}{2}}, u^{n(R)}_{j+\frac{1}{2}})$ is given by:

$$u(0; u^{n(L)}_{j+\frac{1}{2}}, u^{n(R)}_{j+\frac{1}{2}}) = \begin{cases} u^{n(L)}_{j+\frac{1}{2}} & \text{if } \left(0 < u^{n(L)}_{j+\frac{1}{2}} < u^{n(R)}_{j+\frac{1}{2}}\right) \\ 0 & \text{if } \left(u^{n(L)}_{j+\frac{1}{2}} \leq 0 \leq u^{n(R)}_{j+\frac{1}{2}}\right) \\ u^{n(R)}_{j+\frac{1}{2}} & \text{if } \left(u^{n(L)}_{j+\frac{1}{2}} < u^{n(R)}_{j+\frac{1}{2}} < 0\right) \\ u^{n(L)}_{j+\frac{1}{2}} & \text{if } \left(u^{n(R)}_{j+\frac{1}{2}} < u^{n(L)}_{j+\frac{1}{2}} \text{ and } S > 0\right) \\ u^{n(R)}_{j+\frac{1}{2}} & \text{if } \left(u^{n(R)}_{j+\frac{1}{2}} < u^{n(L)}_{j+\frac{1}{2}} \text{ and } S < 0\right), \end{cases} \tag{A.3}$$

where $S = \left(u^{n(L)}_{j+\frac{1}{2}} + u^{n(R)}_{j+\frac{1}{2}}\right)/2$.

With the same spatial and temporal discretizations as in Section 6.1.1 and known cell averages at time t_n, the steps to obtain cell averages at time t_{n+1} are as follows.

A.1.1 IMPICE Method Description

A.1.1.1 The Primary Phase

At face center, a data reconstruction is done as follows:

$$u^{n(L)}_{j+\frac{1}{2}} = u^n_j + \frac{\Delta x}{2}\bar{\Delta} u^n_j, \qquad u^{n(R)}_{j+\frac{1}{2}} = u^n_{j+1} - \frac{\Delta x}{2}\bar{\Delta} u^n_{j+1}, \tag{A.4}$$

where $\bar{\Delta} u^n_j$ is the limited slope of u using van Leer limiter in Equation (6.41). The face-centered velocity, $u^n_{j+\frac{1}{2}}$, at t_n is determined using the approximate solution of Riemann problem where $u^n_{j+\frac{1}{2}} = u(0; u^{n(L)}_{j+\frac{1}{2}}, u^{n(R)}_{j+\frac{1}{2}})$. The equation of velocity evolution:

$$u_t + uu_x = 0, \tag{A.5}$$

is written in Lagrangian form as:

$$\frac{Du}{Dt} = 0. \tag{A.6}$$

The face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, is then given by:

$$u^*_{j+\frac{1}{2}} = u^n_{j+\frac{1}{2}}. \tag{A.7}$$

In order to apply the Lagrangian and Eulerian phases, we rewrite equation (A.1) as:

$$(\rho u)_t + \left(\rho u^2\right)_x = \frac{1}{2}\left(u^2\right)_x \tag{A.8}$$

where ρ is a constant and equal to one.

A.1.1.2 The Lagrangian Phase

The change in cell mass along a path moving with fluid velocity u is given by:

$$V^L_j (\rho u)^L_j = V^n_j (\rho u)^n_j + \Delta t \left(\frac{1}{2}\left(u^*_{j+\frac{1}{2}}\right)^2 - \frac{1}{2}\left(u^*_{j-\frac{1}{2}}\right)^2 \right), \tag{A.9}$$

where $V^n_j = \Delta x$ and $V^L_j = \Delta x + \Delta t(u^*_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}})$. As ρ is a constant and equal to one, the above equation may be rewritten as follows:

$$V^L_j u^L_j = V^n_j u^n_j + \Delta t \left(\frac{1}{2}\left(u^*_{j+\frac{1}{2}}\right)^2 - \frac{1}{2}\left(u^*_{j-\frac{1}{2}}\right)^2 \right) \tag{A.10}$$

A.1.1.3 The Eulerian Phase

The change in mass due to the advection is given by:

$$V_j^{n+1}(\rho u)_j^{n+1} = V_j^L(\rho u)_j^L - \Delta t\left(u^*_{j+\frac{1}{2}}\langle\rho u\rangle_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}}\langle\rho u\rangle_{j-\frac{1}{2}}\right) \tag{A.11}$$

where $V_j^{n+1} = \Delta x$. As ρ is a constant and equal to one, the above equation may be rewritten as follows:

$$V_j^{n+1}u_j^{n+1} = V_j^L u_j^L - \Delta t\left(u^*_{j+\frac{1}{2}}\langle u\rangle_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}}\langle u\rangle_{j-\frac{1}{2}}\right). \tag{A.12}$$

For first-order advection, $\langle u\rangle_{j+\frac{1}{2}}$ is approximated using (6.17) and for second-order advection, it is approximated using (6.77).

A.1.2 Numerical Results and Accuracy in Space and Time

The initial condition used is given by:

$$u_0(x) = \begin{cases} 1.0 & \text{if } \left(|x| < \frac{1}{3}\right) \\ 0.0 & \text{otherwise.} \end{cases} \tag{A.13}$$

The numerical solution of the inviscid Burgers' problem using the second-order-in-space and second-order-in-time IMPICE method is shown in Figure A.1. The spatial and temporal error norms and the orders of accuracy for this problem are summarized in Table A.1. For temporal errors, the orders of accuracy are as expected whereas the orders of accuracy are around one for the first-order method and very close to two for the second-order method. However, there is a degeneration in the spatial orders of accuracy as happened for the above test problems.

A.2 IMPICE Method for Viscous Burgers' Equation

The viscous form of Burgers' Equation:

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = \epsilon\frac{\partial^2 u}{\partial x^2}, \tag{A.14}$$

with $f(u) = \frac{1}{2}u^2$ and initial data $u(x,0) = u_0(x)$, where $u_0(x)$ is a given function and ϵ is a constant, $x \in \mathbb{R}$ and $0 < t < T_e$.

With the same spatial and temporal discretization as in Section 6.1.1 and known cell averages at time t_n, the steps to obtain cell averages at time t_{n+1} are as follows.

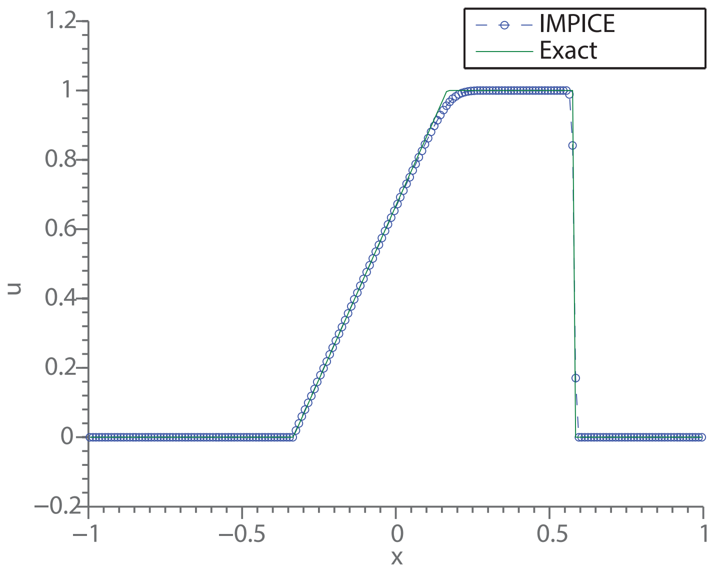


Figure A.1. The second-order (second-order-in-space and second-order-in-time) IMPICE numerical solutions for the inviscid Burgers' problem at $T_e = 0.5$ and on the spatial domain $[-1.0, 1.0]$ with N=200 (cells) and $C_{cfl} = 0.2$.

Table A.1. Spatial and Temporal Errors: L_1-norms and the order of accuracy for the inviscid Burgers' problem at $T_e = 0.5$ on the spatial domain $[-1.0, 1.0]$. The temporal errors are calculated for the grid using N=200 (cells) and the time-integrated exact solutions are the converged numerical solutions.

$\mathbf{et}^u(T_e)$					$\mathbf{es}^u(T_e)$				
	first-order		second-order			first-order		second-order	
C_{cfl}	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	N	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m
0.2	3.53E-03	—	6.89E-05	—	100	5.14E-02	—	1.86E-02	—
0.1	1.75E-03	1.02	1.67E-05	2.05	200	2.52E-02	1.03	6.08E-03	1.61
0.05	8.67E-04	1.01	4.22E-06	1.98	400	1.62E-02	0.64	3.27E-03	0.90
0.025	4.31E-04	1.01	1.05E-06	2.01	800	8.48E-03	0.93	1.52E-03	1.11
0.0125	2.15E-04	1.01	2.61E-07	2.01	1600	5.27E-03	0.69	8.16E-04	0.90

A.2.1 IMPICE Method Description

A.2.1.1 The Primary Phase

The equation of velocity evolution:

$$u_t + uu_x = \epsilon u_{xx}, \tag{A.15}$$

is written in Lagrangian form as:

$$\frac{Du}{Dt} = \epsilon u_{xx}. \tag{A.16}$$

The face-centered fluxing velocity, $u^*_{j+\frac{1}{2}}$, is approximated using an explicit scheme in the Lagrangian frame as:

$$u^*_{j+\frac{1}{2}} = u^n_{j+\frac{1}{2}} + \frac{\Delta t}{2}\epsilon\left[\frac{u^n_{j+\frac{3}{2}} - 2u^n_{j+\frac{1}{2}} + u^n_{j-\frac{1}{2}}}{\Delta x^2}\right], \tag{A.17}$$

where the calculation of $u^n_{j+\frac{1}{2}}$ has already been discussed in Appendix A.1. In order to apply the Lagrangian and Eulerian phases, we rewrite equation (A.14) as:

$$(\rho u)_t + \left(\rho u^2\right)_x = \left[\frac{1}{2}u^2 + \epsilon u_x\right]_x \tag{A.18}$$

where ρ is a constant and equal to one.

A.2.1.2 The Lagrangian Phase

The discrete form of the Lagrangian part of equation (A.18) is as follows:

$$V^L_j(\rho u)^L_j = V^n_j(\rho u)^L_j + \Delta t\left[\left(\frac{1}{2}\left(u^*_{j+\frac{1}{2}}\right)^2 + \epsilon\frac{u^*_{j+\frac{3}{2}} - u^*_{j-\frac{1}{2}}}{2\Delta x}\right) - \left(\frac{1}{2}\left(u^*_{j-\frac{1}{2}}\right)^2 + \epsilon\frac{u^*_{j+\frac{1}{2}} - u^*_{j-\frac{3}{2}}}{2\Delta x}\right)\right] \tag{A.19}$$

where $V^n_j = \Delta x$ and $V^n_j = \Delta x + \Delta t\left(u^*_{j+\frac{1}{2}} - u^*_{j-\frac{1}{2}}\right)$ and ρ is a constant and equal to one.

A.2.1.3 The Eulerian Phase

The Eulerian Phase for the viscous Burgers' Equation is the same as the Eulerian Phase for the inviscid Burgers' problem in Appendix A.1.

A.2.2 Numerical Results and Accuracy in Space and Time

The initial condition for the viscous Burgers' problem satisfies the below analytical solution:

$$u(x,t) = \begin{cases} \dfrac{(0.5 + 0.1e^{\frac{c}{\epsilon}} + e^{\frac{-a}{\epsilon}})}{(1 + e^{\frac{c}{\epsilon}} + e^{\frac{-a}{\epsilon}})} & \text{if } (a > 0) \text{ and } (a > b) \\ \dfrac{(0.1 + 0.5e^{\frac{-c}{\epsilon}} + e^{\frac{-b}{\epsilon}})}{(1 + e^{\frac{-c}{\epsilon}} + e^{\frac{-b}{\epsilon}})} & \text{if } (b > 0) \text{ and } (b > a) \\ \dfrac{(1 + 0.5e^{\frac{a}{\epsilon}} + 0.1e^{\frac{b}{\epsilon}})}{(1 + e^{\frac{a}{\epsilon}} + e^{\frac{b}{\epsilon}})} & \text{otherwise} \end{cases} \tag{A.20}$$

where:

$$a = \frac{x - 0.25 - 0.75t}{4}, \quad b = \frac{0.9x - 0.325 - 0.495t}{2}, \quad \text{and} \quad c = \frac{0.8x - 0.4 - 0.24t}{4}. \tag{A.21}$$

The numerical solutions of the viscous Burgers' problem at $T_e = 0.5$ on the spatial domain $[-2.0, 4.0]$ using the second-order-in-space and second-order-in-time IMPICE method with 200 (cells) and $C_{cfl} = 0.2$ for various values of ϵ are shown in Figures A.2 and A.3. In Figures A.2 and A.3, the plotted initial cell averages are obtained from numerical integrations of the initial condition in Equation (A.20) for these given values of ϵ. When ϵ is small, there exists a steep front in the solution of the viscous Burgers' problem.

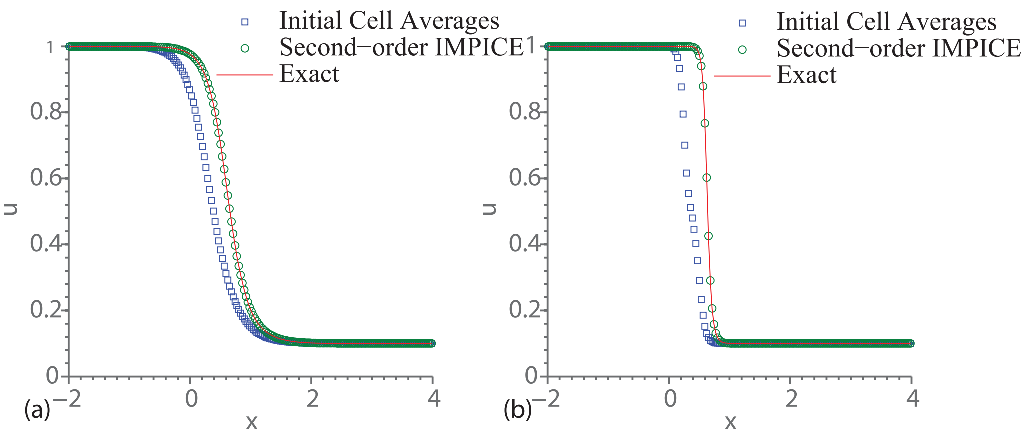


Figure A.2. The second-order (second-order-in-space and second-order-in-time) IMPICE numerical solutions for the viscous Burgers' problem at $T_e = 0.5$ from the plotted initial cell averages with N=200 (cells) and $C_{cfl} = 0.2$: (a)$\epsilon = 0.05$ and (b)$\epsilon = 0.01$.

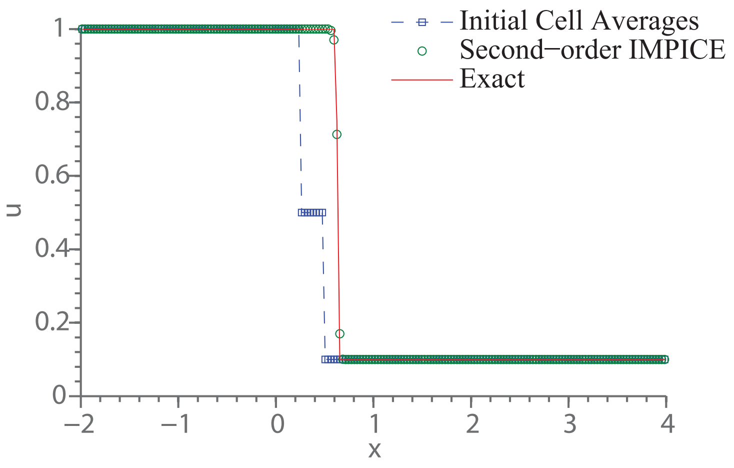


Figure A.3. The second-order (second-order-in-space and second-order-in-time) IMPICE numerical solutions for the viscous Burgers' problem at $T_e = 0.5$ from the plotted initial cell averages with N=200 (cells) and $C_{cfl} = 0.2$ and $\epsilon = 0.0001$.

The spatial and temporal error norms and orders of accuracy for the viscous Burgers' problem with these values of ϵ are summarized in Table A.2. The orders of accuracy for temporal errors are consistently around one for first-order method and around two for second-order method. The convergence rates of spatial errors for the viscous Burgers' problem improve for larger values of ϵ, and get close to one for the first-order method and two for the second-order method. However, there is a degeneration in accuracies for small ϵ. When $\epsilon = 0.0001$, the order is below one for the first-order method and approaching one for the second-order method. This is due to the development of the steep front that appears close to a discontinuity in the numerical solution of the viscous Burgers' problem when ϵ approaches zero.

A.3 IMPICE versus Conservative Cell-centered ICE

The results in Figures A.4–A.8 show the improvement obtained from the application of slope limiters in the data resconstruction of the Riemann problem. In Figures A.4–A.8, the numerical results of the IMPICE method are compared against the numerical results of the conservative cell-centered ICE method, which is implemented using Kashiwa *et al.* [68] and chooses to conserve mass, linear momentum and total energy as discussed in Section 6.4. Problems **P1**–**P5** are from Table 6.1. We use first-order advection for both of these methods. As seen in A.4–A.8, the IMPICE method helps to eliminate the nonphysical oscillations in the implementation of the conservative cell-centered ICE method.

A.4 Different Calculations of the Face-centered Pressure

When discussing how to calculate the face-centered pressure, $p^*_{j+\frac{1}{2}}$, in the implementation of the IMPICE method in Section 6.4, we mentioned that there were two other ways to calculate this quantity in Kashiwa *et al.* [68]. We will present in this section the proposed methods of [68] and see how these methods will change the results if implemented in the IMPICE method.

The following derivation is extracted from Kashiwa *et al.* [68]. The first step in calculating the face-centered pressure $p^*_{j+\frac{1}{2}}$ is to differentiate the momentum equation.

The one-dimensional form of equation (3.8) is given by:

$$u_t + uu_x = -\frac{p_x}{\rho}. \tag{A.22}$$

Taking the partial derivative of (A.22) in space, the obtained equation is:

$$(u_t + uu_x)_x = -\left(\frac{p_x}{\rho}\right)_x. \tag{A.23}$$

Table A.2. Spatial and Temporal Errors: L_1-norms and the order of accuracy for the viscous Burgers' problem at $T_e = 0.5$ on the spatial domain $[-2.0, 4.0]$. The temporal errors are calculated for the grid using N=200 (cells) and the time-integrated exact solutions are the converged numerical solutions.

		$\mathbf{et}^u(T_e)$					$\mathbf{es}^u(T_e)$			
		first-order		second-order			first-order		second-order	
ϵ	C_{cfl}	$\|.\|_{L_1}$	n	$\|.\|_{L_1}$	n	N	$\|.\|_{L_1}$	m	$\|.\|_{L_1}$	m
0.05	0.2	1.04E-03	—	5.13E-06	—	100	1.73E-02	—	5.01E-04	—
	0.1	5.17E-04	1.00	1.28E-06	2.00	200	9.00E-03	0.94	1.43E-04	1.81
	0.05	2.58E-04	1.00	3.21E-07	2.00	400	4.60E-03	0.97	4.11E-05	1.80
	0.025	1.29E-04	1.00	8.01E-08	2.00	800	2.32E-03	0.98	1.24E-05	1.73
	0.0125	6.41E-05	1.01	2.00E-08	2.00	1600	1.17E-03	0.99	3.53E-06	1.80
0.01	0.2	2.18E-03	—	2.48E-05	—	100	3.54E-02	—	4.43E-03	—
	0.1	1.08E-03	1.01	6.17E-06	2.01	200	2.04E-02	0.80	1.08E-03	2.04
	0.05	5.38E-04	1.01	1.54E-06	2.00	400	1.10E-02	0.89	3.15E-04	1.77
	0.025	2.68E-04	1.01	3.84E-07	2.00	800	5.78E-03	0.93	8.50E-05	1.89
	0.0125	1.33E-04	1.01	9.61E-08	2.00	1600	2.97E-03	0.96	2.23E-05	1.93
0.0001	0.2	2.91E-03	—	5.17E-05	—	100	5.78E-02	—	2.38E-02	—
	0.1	1.43E-03	1.02	1.27E-05	2.03	200	2.36E-02	1.29	1.20E-02	0.99
	0.05	7.11E-04	1.01	3.16E-06	2.00	400	1.46E-02	0.69	6.52E-03	0.88
	0.025	3.54E-04	1.01	7.84E-07	2.01	800	8.28E-03	0.82	3.26E-03	1.00
	0.0125	1.76E-04	1.01	1.96E-07	2.00	1600	4.94E-03	0.75	1.46E-03	1.16

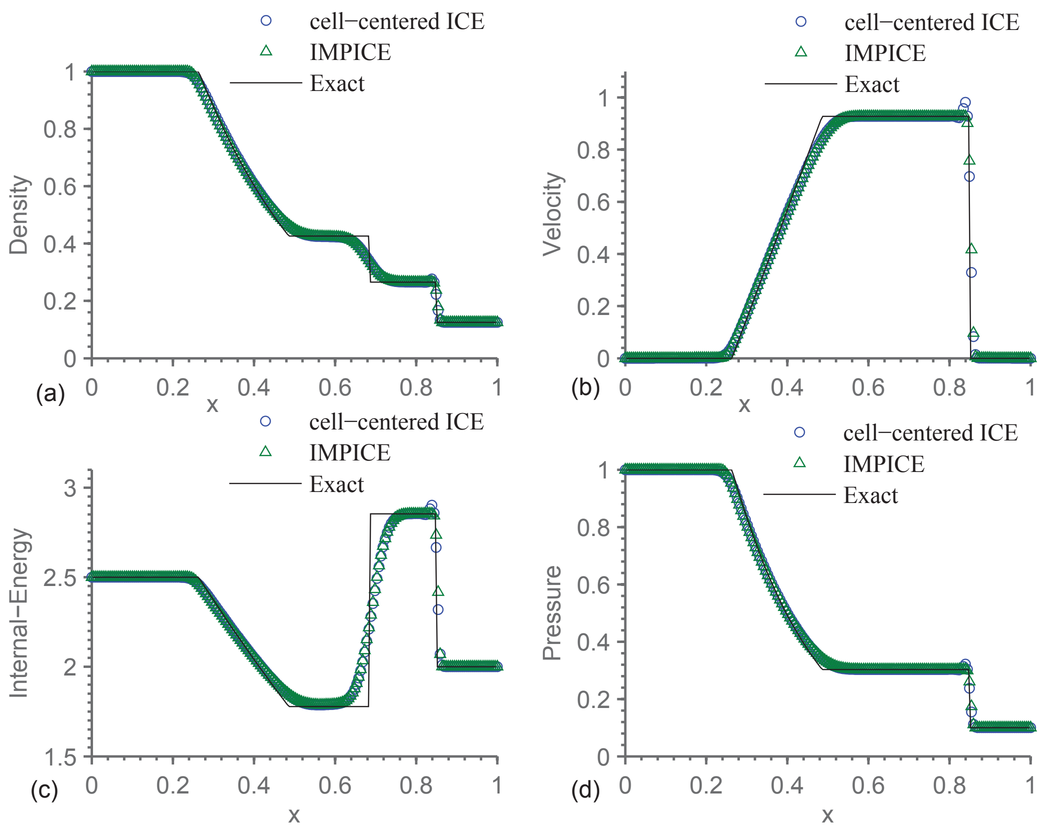


Figure A.4. Conservative cell-centered ICE and IMPICE numerical solutions for test **P1** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

The one-dimensional form of Equation (3.10) is given by:

$$p_t + up_x = -c^2 \rho u_x. \tag{A.24}$$

The time dependent quantity u_{tx} is eliminated using the partial time derivative of the pressure equation (A.24) which is given by:

$$(p_t + up_x)_t = -\left(c^2 \rho u_x\right)_t. \tag{A.25}$$

From these equations Kashiwa *et al.* [68] state without derivation that linearization produces:

$$u_{xt} = -\frac{u}{c^2 \rho}\left(\frac{Dp}{Dt}\right)_x, \tag{A.26}$$

and so derive the potential equation for face-centered pressure:

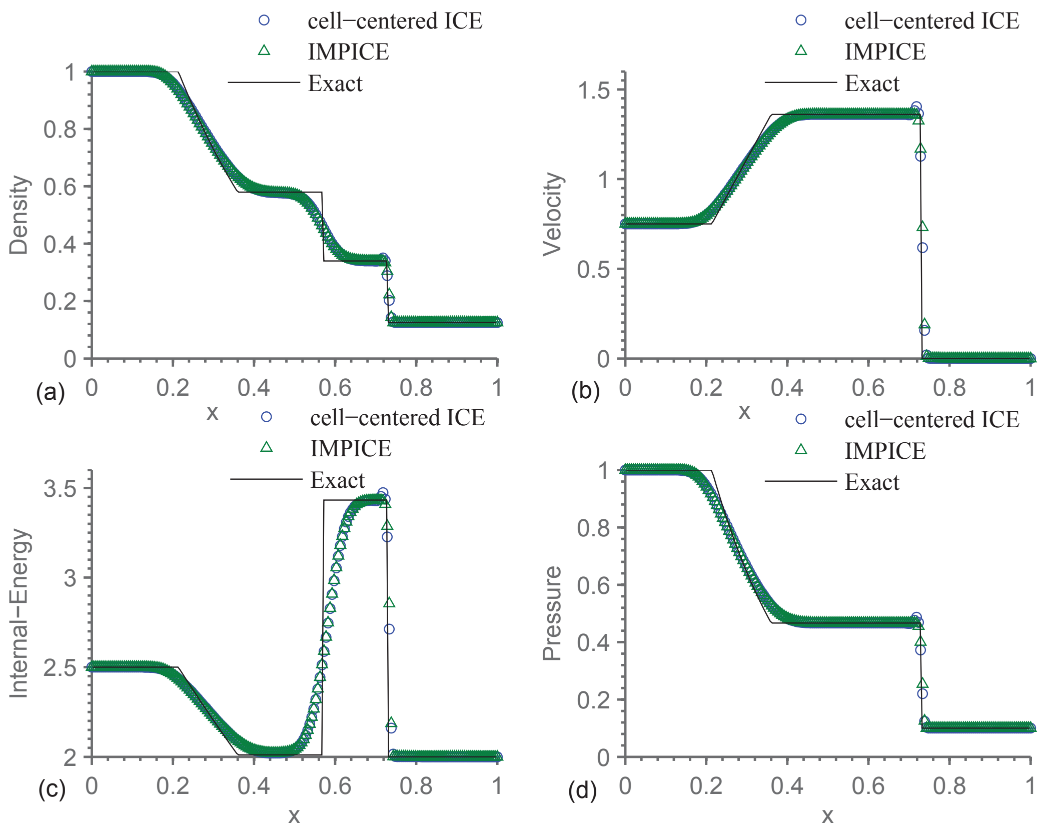


Figure A.5. Conservative cell-centered ICE and IMPICE numerical solutions for test **P2** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

$$\left(\frac{p_x}{\rho}\right)_x = \frac{u}{c^2\rho}\left(\frac{Dp}{Dt}\right)_x - (uu_x)_x \,. \tag{A.27}$$

One discrete form of this is:

$$\frac{1}{\Delta x^2}\left[\frac{p_{j+1}^{n+\frac{1}{2}} - p_{j+\frac{1}{2}}^{*}}{\rho_{j+1}^n} - \frac{p_{j+\frac{1}{2}}^{*} - p_j^{n+\frac{1}{2}}}{\rho_j^n}\right] = \left(\frac{u}{c^2\rho}\right)_{j+\frac{1}{2}}^n \frac{1}{\Delta t \Delta x}\left(\delta p_{j+1}^n - \delta p_j^n\right)$$
$$-\frac{1}{\Delta x^2}\left[u_{j+1}^n(u_{j+1}^n - u_{j+\frac{1}{2}}^*) - u_j^n(u_{j+\frac{1}{2}}^* - u_j^n)\right].$$

The face-centered pressure is then defined by:

$$\begin{aligned} p_{j+\frac{1}{2}}^* &= \left(\frac{\rho_j^n p_{j+1}^{n+\frac{1}{2}} + \rho_{j+1}^n p_j^{n+\frac{1}{2}}}{\rho_{j+1}^n + \rho_j^n}\right) \\ &+ \frac{\Delta x}{\Delta t}\left(\frac{u}{c^2\rho}\right)_{j+\frac{1}{2}}^n \left(\frac{\rho_{j+1}^n \rho_j^n}{\rho_{j+1}^n + \rho_j^n}\right)\left(\delta p_{j+1}^n - \delta p_j^n\right) \\ &- \left(\frac{\rho_{j+1}^n \rho_j^n}{\rho_{j+1}^n + \rho_j^n}\right)\left[u_{j+1}^n(u_{j+1}^n - u_{j+\frac{1}{2}}^*) - u_j^n(u_{j+\frac{1}{2}}^* - u_j^n)\right]. \end{aligned} \tag{A.28}$$

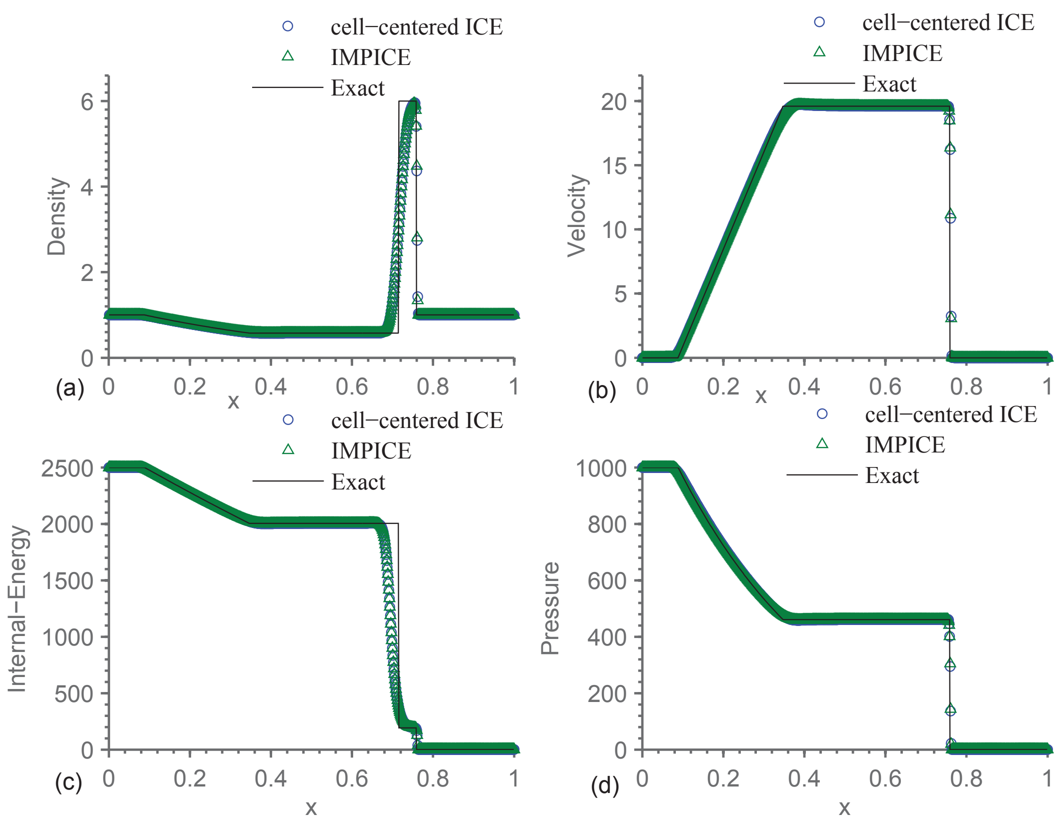


Figure A.6. Conservative cell-centered ICE and IMPICE numerical solutions for test **P3** with N=800 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

The above equation is used to estimate the face-centered pressure , $p^*_{j+\frac{1}{2}}$, that will be used in the Lagrangian phase. It is recognized in [68] that the second term in Equation (A.29) is important in high-speed problems and the third term looks somewhat like a bulk viscosity. These terms help to remove numerical noise, but introduces a diffusive effect in the method. A limited version of (A.29) is given by:

$$\begin{aligned} p^*_{j+\frac{1}{2}} &= \left(\frac{\rho^n_j p^{n+\frac{1}{2}}_{j+1} + \rho^n_{j+1} p^{n+\frac{1}{2}}_j}{\rho^n_{j+1} + \rho^n_j} \right) \\ &\quad + \psi \frac{\Delta x}{\Delta t} \left(\frac{u}{c^2 \rho} \right)^n_{j+\frac{1}{2}} \left(\frac{\rho^n_{j+1} \rho^n_j}{\rho^n_{j+1} + \rho^n_j} \right) \left(\delta p^n_{j+1} - \delta p^n_j \right) \\ &\quad - \psi \left(\frac{\rho^n_{j+1} \rho^n_j}{\rho^n_{j+1} + \rho^n_j} \right) \left[u^n_{j+1} (u^n_{j+1} - u^*_{j+\frac{1}{2}}) - u^n_j (u^*_{j+\frac{1}{2}} - u^n_j) \right], \end{aligned} \tag{A.29}$$

where ψ is a "limiter" that is designed such that $0 \le \psi \le 1$, with values tending towards zero if the velocity field is smooth to remove numerical noise in the velocity. The DIVU limiter

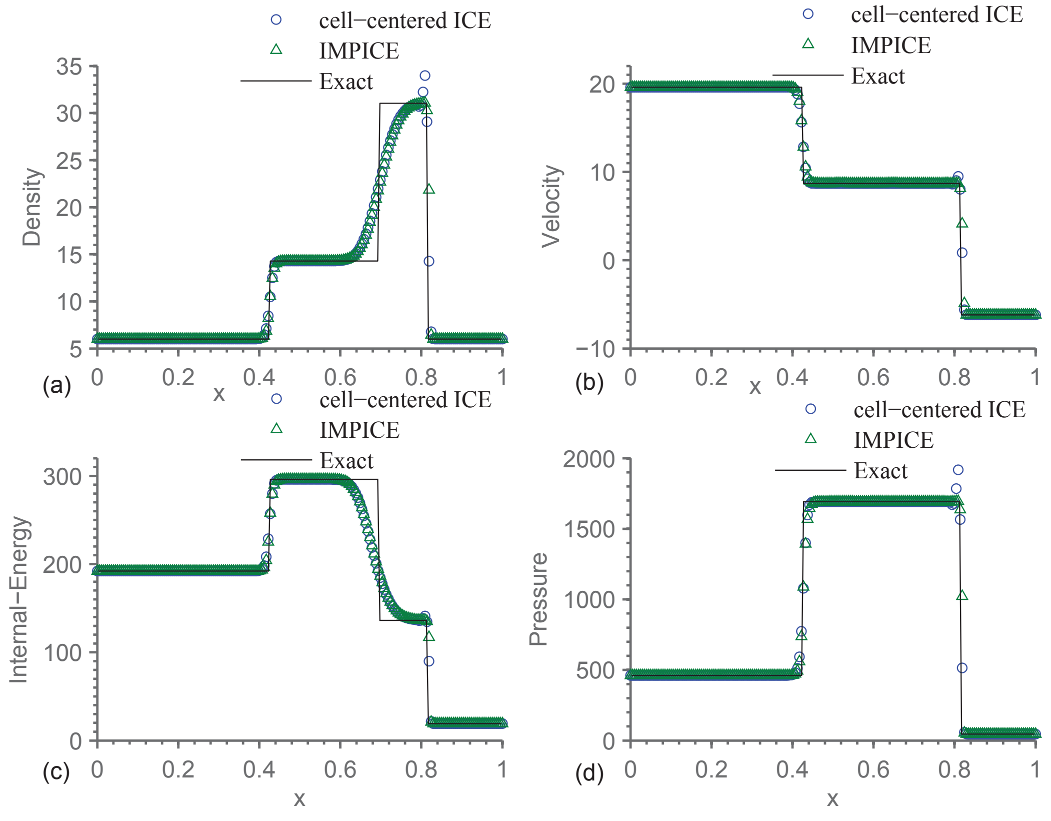


Figure A.7. Conservative cell-centered ICE and IMPICE numerical solutions for test **P4** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

is introduced by Kashiwa and Lee in [67] is used for the purpose of limiting the velocity field in calculating limited face-centered pressure $p^*_{j+\frac{1}{2}}$. The limiter is required at the cell interface and is a function of the face-centered velocity divergence $D^n_{j+\frac{1}{2}}$ and the face-centered velocity divergences on either side of the face, $D^{n(+)}_{j+\frac{1}{2}}$ and $D^{n(-)}_{j+\frac{1}{2}}$. We define these face-centered velocity divergences as

$$D^n_{j+\frac{1}{2}} = u^n_{j+1} - u^n_j; \quad D^{n(+)}_{j+\frac{1}{2}} = u^n_{j+2} - u^n_{j+1}; \quad D^{n(-)}_{j+\frac{1}{2}} = u^n_j - u^n_{j-1}. \tag{A.30}$$

Then the limiter is given by:

$$\psi = \begin{cases} 1 - \max\left[0, min\left(\frac{D^n_{j+\frac{1}{2}}}{D^{n(-)}_{j+\frac{1}{2}}}, \frac{D^n_{j+\frac{1}{2}}}{D^{n(+)}_{j+\frac{1}{2}}}, \frac{D^{n(-)}_{j+\frac{1}{2}}}{D^n_{j+\frac{1}{2}}}, \frac{D^{n(+)}_{j+\frac{1}{2}}}{D^n_{j+\frac{1}{2}}}\right)\right] & \text{if } \; D^n_{j+\frac{1}{2}} \leq 0. \\ 0 & \text{otherwise.} \end{cases} \tag{A.31}$$

In order to make sure the calculated face-centered pressure, $p^*_{j+\frac{1}{2}}$, is bounded by the surrounding cell-centered pressures at $t_{n+\frac{1}{2}}$, $p^{n+\frac{1}{2}}_j$ and $p^{n+\frac{1}{2}}_{j+1}$, its calculated value is clamped with respect to

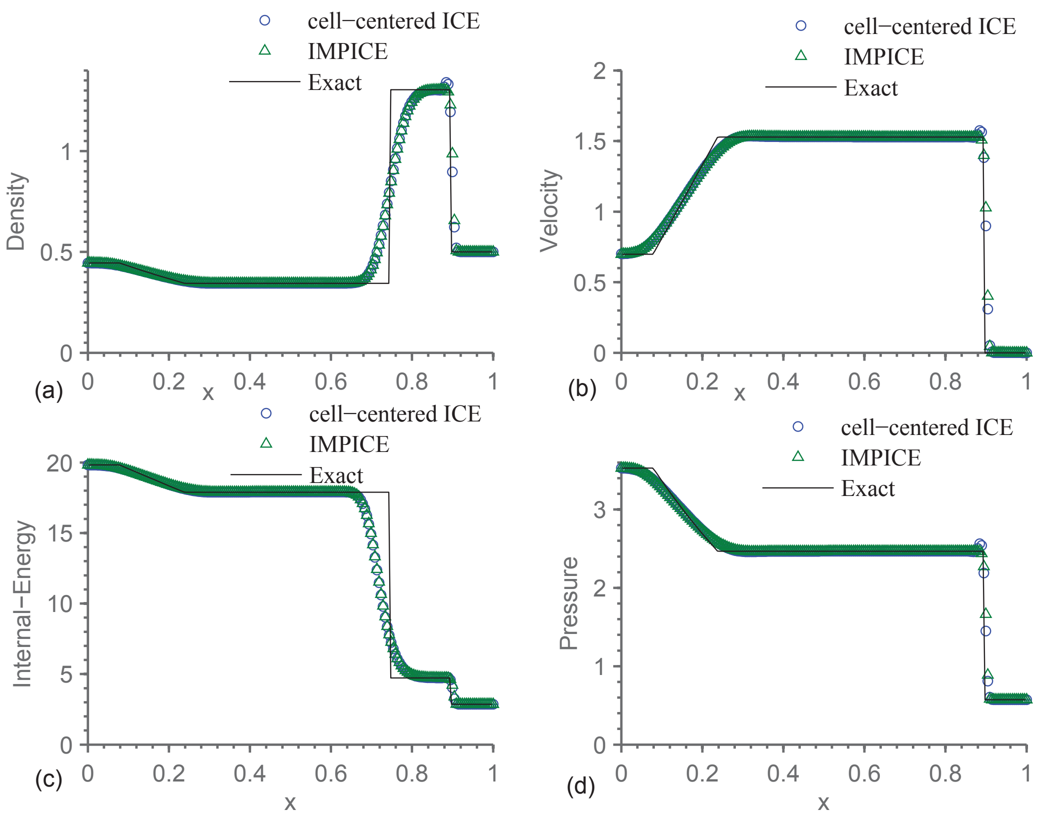


Figure A.8. Conservative cell-centered ICE and IMPICE numerical solutions for test **P5** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

$[p_{min}, p_{max}]$ where:

$$p_{min} = \min(p_j^{n+\frac{1}{2}}, p_{j+1}^{n+\frac{1}{2}}); \quad p_{max} = \max(p_j^{n+\frac{1}{2}}, p_{j+1}^{n+\frac{1}{2}}). \tag{A.32}$$

This means the face-centered pressure, $p^*_{j+\frac{1}{2}}$, is set to p_{min} if $\left(p^*_{j+\frac{1}{2}} < p_{min}\right)$ and is set to p_{max} if $\left(p^*_{j+\frac{1}{2}} > p_{max}\right)$.

We compare the numerical results obtained from the IMPICE method and the pressure-limited IMPICE method (PL-IMPICE) for the test cases in Table 6.1 in Figures A.9–A.12. The PL-IMPICE method uses the implementation of the IMPICE method in Section 6.4 except that the face-centered pressure, $p^*_{j+\frac{1}{2}}$, is calculated using the limited version in (A.29). As shown in Figures A.9–A.12, there is a slight difference in the numerical solutions of these methods at the discontinuous regions. However, there are no nonphysical oscillations presented in the numerical solutions of these two methods.

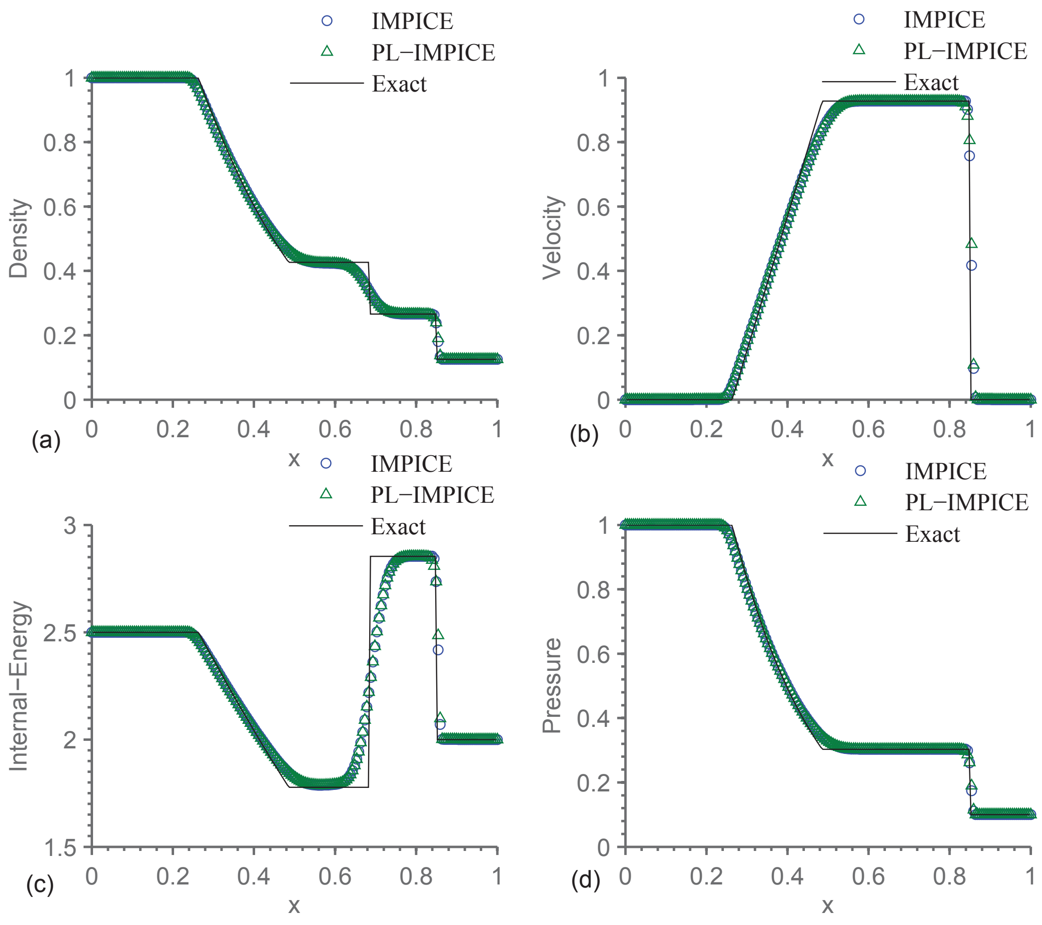


Figure A.9. PL-IMPICE and IMPICE numerical solutions for test **P1** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

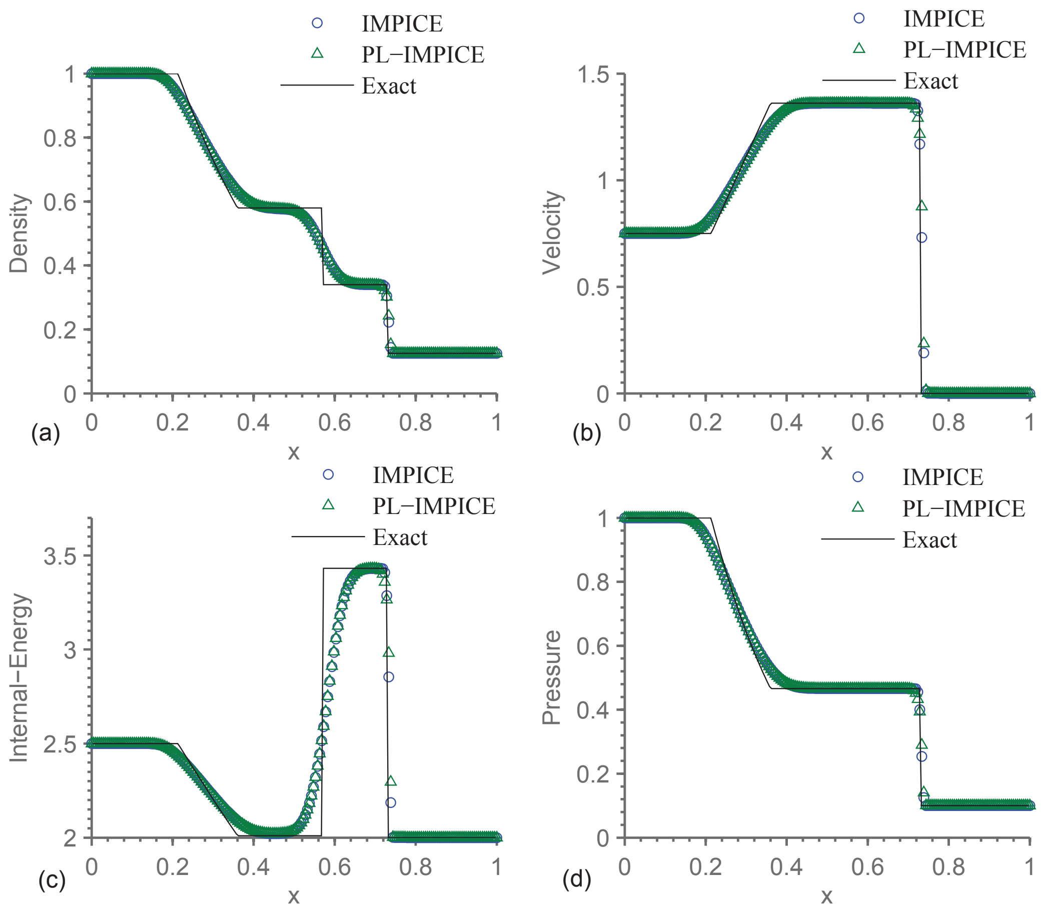


Figure A.10. PL-IMPICE and IMPICE numerical solutions for test **P2** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

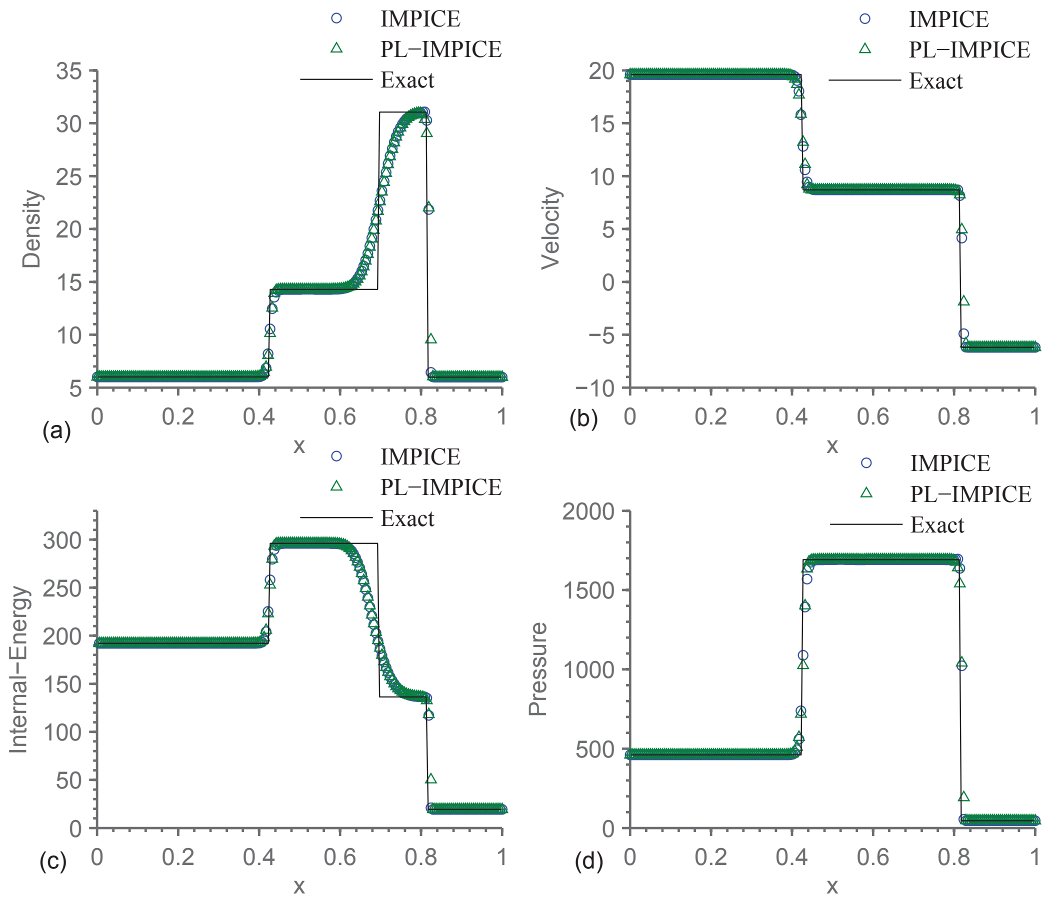


Figure A.11. PL-IMPICE and IMPICE numerical solutions for test **P4** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

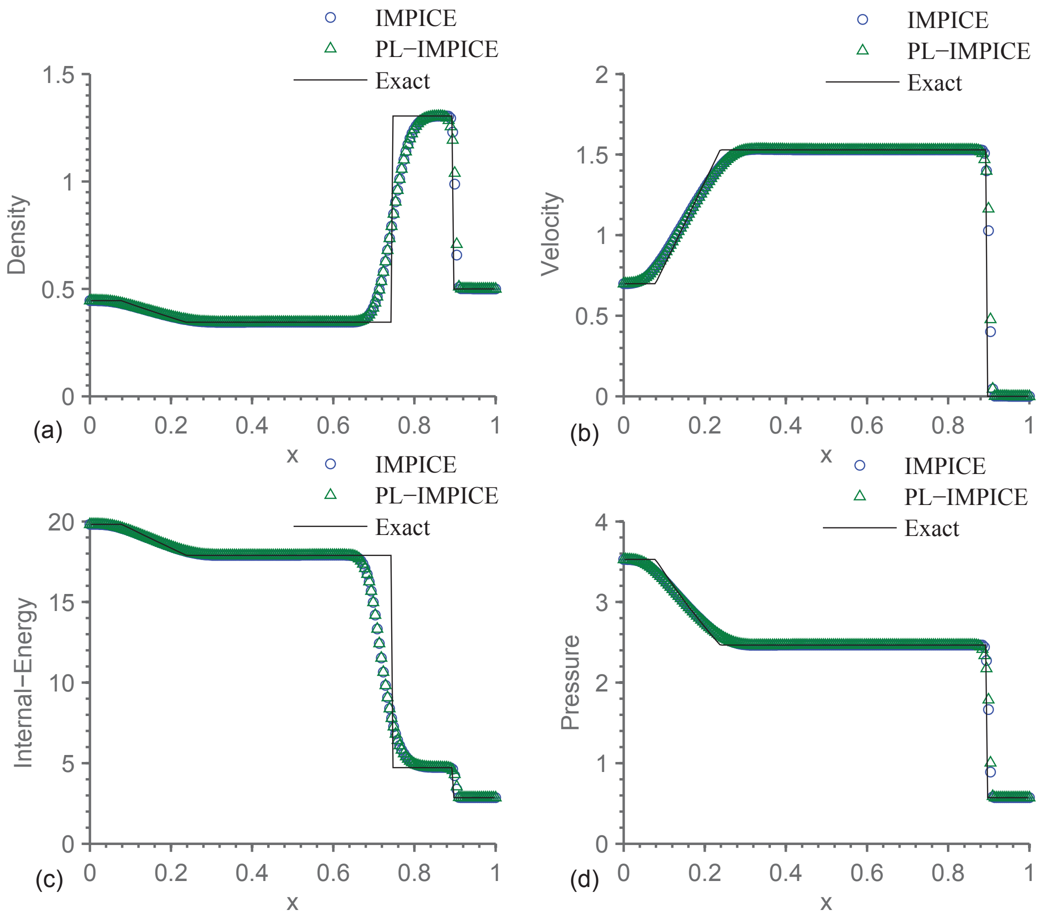


Figure A.12. PL-IMPICE and IMPICE numerical solutions for test **P5** with N=200 (cells) and $C_{cfl} = 0.2$: (a) density; (b) velocity; (d) internal-energy; and (c) pressure.

REFERENCES

[1] Acheson, D. J. Elementary fluid dynamics. *Oxford Applied Mathematics and Computing Science Series. Oxford: Clarendon Press* 1990.

[2] Anderson, W. K., and Venkatakrishnan, V. Aerodynamic design optimization on unstructured grids with a continous adjoint formulation. *AIAA Paper, 97-0643, 1997.*

[3] Ascher, U. M, and Petzold, L. R. Computer methods for ordinary differential equations and differential algebraic equations. *SIAM 1998.*

[4] Bardenhagen, S. G. Energy conservation error in the material point method for solid mechanics. *Journal of Computational Physics 180* (2002), 383–403.

[5] Bardenhagen, S. G., and Kober, E. M. The Generalized Interpolation Material Point Method. *Computer Modeling in Engineering and Sciences 5* (2004), 477–495.

[6] Bardenhagen, S. G., Brydon, A. D., and Guilkey, J. E. Insight into the physics of foam densification via numerical simulation. *Journal of the Mechanics and Physics of Solids 53*, 3 (2005), 597–617.

[7] Barth, T. Numerical methods and error estimation for conservation laws on structured and unstructured meshes. *Lecture notes, von Karman Institute for Fluid Dynamics, Series: 2003-04, Brussels, Belgium, March 2003.*

[8] Becker, R., and Rannacher., R. An optimal control approach to a posteriori error estimation in finite element methods. *Acta Numerica 2001 10* (2001), 1-102.

[9] Benson, D. J. Eulerian finite element methods for the micromechanics of hetheterogeneous materials: Dynamic prioritization of material interfaces. *Comput. Methods Appl. Mech. Eng. 151* (1998), 343–360.

[10] Ben-Artzi, M., and Falcovitz., J. A second order Godunov-Type scheme for compressible fluid dynamics. *J. Comput. Phys. 55* (1984), 1–32.

[11] Ben-Artzi, M. Application of the Generalised Riemann Problem Method to 1-D compressible flows with interfaces. *J. Comput. Phys. 65* (1986), 170–178.

[12] Berzins, M. Nonlinear data-bounded polynomial approximations and their applications in ENO methods. *Numerical Algorithms 55*, 2 (2010), 171–188.

[13] Berzins, M. Global error estimation in the method of lines for parabolic equations. *SIAM J. Sci. Statist. Comput. 9*, 4 (1988), 687–703.

[14] Bouma, R. H. B., van der Heijden, A. E. D. M., Sewell, T. D., and Thompson, D. L. (2011). Simulations of deformation processes in energetic materials, numerical simulations of physical and engineering processes, Jan Awrejcewicz (Ed.), ISBN: 978-953-307-620-1, InTech, Available from: http://www.intechopen.com/articles/show/title/simulations-of-deformation-processes-in-energetic-materials.

[15] Brackbill, J. U., and Ruppel, H. M. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flow in two dimensions. *Journal of Computational Physics 65* (1986), 314–343.

[16] Brackbill, J. U. The ringing instability in particle in cell calculations of low-speed flow. *Journal of Computational Physics 75* (1988), 469–492.

[17] Brackbill, J. U. Particle methods. *International Journal for Numerical Metheds in Fluids 47* (2005), 693–705.

[18] Brownlee, J., Levesley, J., Houston, P., and Rosswog., S. Enhancing SPH using moving least-squares and radial basis functions. In Proc. A4A5 (Algorithms for Approximation) , Chester UK, Jul. 18-22 2005, Springer, 2007.

[19] Brydon, A. D., Bardenhagen, S. G., Miller, E. A., and Seidler, G. T. Simulation of the densification of real open-celled foam microstructures. *Journal of the Mechanics and Physics of Solids 53* (2005), 2638-2660.

[20] Burgess, D., Sulsky, D., and Brackbill, J. U. Mass matrix formulation of the FLIP Particle in Cell method. *Journal of Computational Physics 103* (1992), 1-15.

[21] Cao, Y., and Petzold, L. A posteriori error estimation and global error control for ordinary differential equations by the adjoint method. *SIAM J. Sci. Comput. 26*, 2 (2004), 359–374.

[22] Casulli, V., and Greenspan, D. Pressure method for the numerical solution of transient, compressible fluid flows. *Int. J. Num. Meth. Fluids 4* (1984), 1001-1012.

[23] Chawla, M. M., and Subramanian, R. Regions of absolute stability of explicit Runge Kutta Nystrom methods for y00 = f(x; y; y0). *Journal of Computational and Applied Mathematics 11* (1984), 259–266.

[24] Colella, P., Graves, D. T., Keen, B. J., and Modiano, D. A Cartesian Grid Embedded Boundary Method for hyperbolic conservation laws. Tech. report LBNL-56420, Lawrence Berkeley National Laboratory, Berkeley, CA 2004.

[25] Colella, P. A Direct Eulerian MUSCL Scheme for gas dynamics. *SIAM J. Sci. Stat. Comput. 6* (1985), 104–117.

[26] Coirier, W., and Powell, K. An accuracy assessment of Cartesian-mesh approaches for the Euler equations. *J. Comput. Phys. 117* (1995), 121–131.

[27] Debrabant, K., and Lang, J. On global error estimation and control for parabolic equations. *Report No. 2512 (2007), Technische Universitt Darmstadt, Department of Mathematics.*

[28] Davis, S. F. Simplified Second-order Godunov-Type methods. *SIAM J. Sci. Stat. Comput. 9*, 3 (1988), 445-473.

[29] Diskin, B., and Thomas, J. L. Comparision of Node-Centered and Cell-Centered Unstructured Finite-Volume Discretizations: Inviscid Fluxed 48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition 4 - 7 January 2010, Orlando, Florida.

[30] Dukowicz, J. K., and Kodis, J. W. Accurate Conservative Remapping (Rezoning) for Arbitrary Lagrangian-Eulerian Computations. *SIAM J. Sci. Stat. Comput. 8*, 3 (1987), 305-321.

[31] Duncan, C., Harman, T., and Guilkey, J. Aerodynamics of Vocal Fold Movement: A Novel Fluid-Structure Interaction Model. *Proceedings 60th Annual Meeting of the Division of Fluid Dynamics Volume 52 Number 12, Salt Lake City, UT, 18-20 Nov 2007.*

[32] Enright, W. H., A New Error-Control for Initial Value Solvers. *Appl. Math. Comput. 31* (1989), 588–599.

[33] Einfeldt, B. On Godunov-Type Methods for Gas Dynamics. *SIAM J. Numer. Anal. 25*, 2 (1988), 294-318.

[34] Estep, D. J., Larson, M. G., and Williams, R. D. Estimating the Error of Numerical Solutions of Systems of Reaction-Diffusion Equations. *Mem. Amer. Math. Soc 696* (2000), 1–109.

[35] Evans, M. W., and Harlow, F. H. The Particle-in-Cell Method for Hydrodynamic Calculations. *Los Alamos Scientific Laboratory report LA-2139* (November 1957).

[36] Falcovitz, J., Alfandary, G., and Hanoch, G. A two-dimensional conservation laws scheme for compressible flows with moving boundaries. *J. Comput. Phys. 138* (1997), 83-102.

[37] Forrer, H., and Jeltsch, R. A high-order boundary treatment for Cartesian-grid methods. *J. Comput. Phys. 140* (1998), 259–277.

[38] Gao, T., Tseng, Y.-H., and Lu, X.-Y. An improved hybrid Cartesian/immersed boundary method for fluid-solid flows. *Int. J. Numer. Meth. Fluids 55* (2007), 1189-1211.

[39] Gaskell, P. H., and Lau, A. K. C. Curvature-Compensated Convective Transport: SMART, a New Boundedness-Preserving Transport Algorithm. *Int. J. Num. Meth. Fluids 8*, 6 (1988), 617–641.

[40] Germain, J. D. d. S., McCorquodale, J., Parker, S. G., and Johnson, C. R. Uintah: A massively parallel problem solving environment. In HPDC '00: Proceedings of the 9th IEEE International Symposium on High Performance Distributed Computing(Washington, DC, USA). *IEEE Computer Society* (2000), 33-42.

[41] Giles, M. B., and Pierce, N. A. Adjoint error correction for integral outputs. *Error Estimation and Adaptive Discretization Methods in Computational Fluid Dynamics, edited by T. Barth and H. Deconinck, Vol. 25 of Lecture Notes in Computational Science and Engineering, Springer-Verlag, 2002.*

[42] Greenough, J. A., and Rider, W. J. A quantitative comparison of numerical methods for the compressible Euler equations: fifth-order WENO and piecewise-linear Godunov. *Journal of Computational Physics 196* (2004), 259–281.

[43] Grigoryev, Y. N., Vshivkov, V. A., and Fedoruk, M. P. Numerical Particle in Cell Methods Theory and Applications. *VSP, Utrecht, Boston, 2002.*

[44] Guilkey, J. E., Harman, T., Xia, A., Kashiwa, B., and McMurtry, P.A. An Eulerian-Lagrangian Approach for Large Deformation Fluid Structure Interaction Problems, Part 1: Algorithm Development. *WIT Press* (2003), 143-156.

[45] Guilkey, J. E., Harman, T., and Banerjee, B. An Eulerian-Lagrangian Approach for Simulating Explosions of Energetic Devices. *Computers and Structures 85* (2007), 660–674.

[46] Harlow F. H. A Machine Calculation Method for Hydrodynamic Problems. *Los Alamos Scientific Laboratory report LAMS-1956* (November 1955).

[47] Harlow, F. H., and Welch, J. E. Numerical Calculation of Time-Dependent Viscous Incompressible Flow. *Phys. Fluids 8, 2182 (1965); Selected Papers in Physics," Vol. VI (The Physical Society of Japan, Tokyo, 1971).*

[48] Harlow, F. H., and Welch, J. E. Numerical Study of Large Amplitude Free Surface Motions. *Phys. Fluids 9* (1966), 842–851.

[49] Harlow, F. H., and Amsden, A. A. Numerical Calculation of Almost Incompressible Flow. *Journal of Computational Physics 3* (1968), 80–93.

[50] Harlow, F. H., and Amsden, A. A. A Numerical Fluid Dynamics Calculation Method for All Flow Speeds. *Journal of Computational Physics 8* (1971), 197-213.

[51] Harlow, F. H., and Amsden, A. A. Numerical Calculation of Multiple Fluid Flow. *Journal of Computational Physics 17* (1975), 19–52.

[52] Harman, T., Guilkey, J. E., Schmidt, J., Kashiwa, B. A., and McMurtry, P. An Eulerian-Lagrangian approach for large deformation fluid structure interaction problems, part 2: Multi-physics simulations. *Proceedings of the Second International Conference on Fluid Structure Interaction, Cadiz, Spain* 2003.

[53] Harten, A., Lax, P. D., and van Leer, B. On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws. *SIAM Rev. 25* (1983), 35–61.

[54] Harten, A., and Osher, S. Uniformly High-Order Accurate Nonoscillatory Schemes I. *SIAM J. Numer. Anal. 24* (1987), 279–309.

[55] Helzel, C., Berger, M. J., and Leveque, R. J. A High-Resolution Rotated Grid Method for Conservation Laws with Embedded Geometries. *SIAM J. Sci. Comput 26*, 3 (2005), 785-809.

[56] Henderson, T., McMurtry, P., Smith, P., Voth, G., Wright, C., and Pershing, D. Simulating accidental fires and explosions. *Computing in Science and Engineering 2* (1994), 64–76.

[57] Hickernel, F. J. A Generalized Discrepancy and Quadrature Bound. *Mathematics of Computation 67* (1998), 299–322.

[58] Higham, D. J. Global Error versus Tolerance for Explicit Runga-Kutta Methods. *IMA Journal of Numerical Analysis 11* (1991), 457–480.

[59] Horley, P., Vieira, V., Gonzalez-Hernandez, J., Dugaev, V., and Barnas, J. (2011). Numerical Simulations of Nano-Scale Magnetization Dynamics, Numerical Simulations of Physical and Engineering Processes, Jan Awrejcewicz (Ed.), ISBN: 978-953-307-620-1, InTech, Available from: http://www.intechopen.com/articles/show/title/numerical-simulations-of-nano-scale-magnetization-dynamics

[60] Hou, T. Y., and LeFloch, P. G. Why Nonconservative Schemes Converge to Wrong solutions: Error Analysis. *Mathematics of Computation 62* (1994), 497–530.

[61] Hu, C., and Shu, C.-W. Weighted essentially non-oscillatory schemes on triangular meshes. *Journal of Computational Physics 150* (1999), 97–127.

[62] Issa, R. I., Gosman, A. D, and Watkins, A. P. The Computation of Compressible and Incompressible Flow of Fluid with a Free Surface. *Phys. Fluids 8*, 12 (1965), 2182–2189.

[63] Issa, R. I. Solution of the Implicitly Discretised Fluid Flow Equations by Operator-Splitting. *Journal of Computational Physics 62* (1986), 40–65.

[64] Jameson, A., Pierce, N. A., and Martinelli, L. Optimum aerodynamic design using the Navier-Stokes equations. *AIAA Paper, 97-0101, 1997.*

[65] Jameson, A. Aerodynamic design via control theory. *J.Sci.Comput. 3* (1988), 233–260.

[66] Jameson, A., and Reuther, J. Control theory based airfoil design using the Euler equation. *AIAA Paper, 94-4272-CP, 1994.*

[67] Kashiwa B. A., and Lee W. H. Comparisons between the Cell-Centered and Staggered Mesh Lagrangian Hydrodynamics. In Advances in the Free Lagrange Method, Trease HE, Fritz MJ, Crowley WP (eds), Springer Verlag, Berlin 1991; 277-288

[68] Kashiwa, B. A., Padial, N. T., Rauenzahn, R. M., and Vanderheyden, W. B. A Cell-Centered ICE Method for Multiphase Flow Simulations. *Proceedings ASME Symposium on Numerical Methods for Multiphase Flows, Lake Tahoe, NV, 19-23 June 1994.*

[69] Kashiwa, B. A. A Multified Model and Method for Fluid-Structure Interaction Dynamics. Los Alamos National Laboratory, Los Alamos 2001; Technical Report LA-UR-01-1136.

[70] Kim, J. MPM Masters Project report unpublished. 2004

[71] Kim, K. H., and Kim, C. Accurate, efficient and monotonic numerical methods for multi-dimensional compressible flows. Part II: Multi-dimensional limiting process. *J. Comput. Phys. 208* (2005), 570–615.

[72] Kwatra, N., Su, J., Gretarsson, J., and Fedkiw, R. A method for avoiding the acoustic time step restriction in compressible flow. *Preprint submitted to JCP.*

[73] Lang, J., and Verwer, J. G. On global error estimation and control for initial value problems. *SIAM J. Sci. Comput. 29* (2007), 1460–1475.

[74] Lax, P. D. Weak Solutions of Nonlinear Hyperbolic Equations and their Numerical Computation. *Commun. Pure Appl. Math. 7* (1954), 159–193.

[75] Lax, P. D., and Wendroff, B. Systems of Conservation Laws. *Communications in Pure and Applied Mathematics 13* (1960), 217–237.

[76] Li, S., and Petzold, L. Adjoint Sensitivity Analysis for time-dependent Partial Differential Equations with Adaptive Mesh Refinement. *J. Comput. Phys. 198* (2004), 310–325.

[77] Li, S., and Liu, W. K. Meshfree particle methods and their applications. *Applied Mechanics Review 54* (2002), 1–34.

[78] Lin, C., and Dengbin, T. Navier-Stokes Characteristic Boundary Conditions for Simulations of Some Typical Flows. *Applied Mathematical Sciences 18*, 4 (2010), 879-893.

[79] Livne, O. E. ICE Algorithm and the Davis Advection Scheme. SCI Institute, University of Utah 2006; Technical Report No. UUSCI-2006-006.

[80] Livne, O. E. ICE Algorithm for the Shocktube Problem. SCI Institute, University of Utah 2006; Technical Report No. UUSCI-2006-007.

[81] Logg, A. Multi-Adaptive Error Control for ODEs. *Technical Report 98/20(1998), Oxford University, England.*

[82] Luitjens, J., Guilkey, J., Harman, T., Worthen, B., and Parker, S. G. Adaptive Computations in the Uintah Framework. In Advanced Computational Infastructures for Parallel/Distributed Adapative Applications, Ch. 1, Wiley Press, 2010.

[83] Ma, S., Zhang, X., and Qiui, X. M. Comparison study of MPM and SPH in modeling hypervelocity impact problems. *International Journal of Impact Engineering 36* (2009), 272–282.

[84] MacCormack, R. W. The Effect of viscosity in hypervelocity impact cratering. *AIAA Paper* (1969), 69–354.

[85] MAcNeice, P. Particle mesh techniques. NASA Contractor Report 4666, Hughes STX, Goddard Space Center, Greenbelt MD 20771. 1995

[86] Mart$\acute{i}$n, M. P., Taylor, E. M., Wu, M., and Weirs, V. G. A Bandwidth-optimized WENO Scheme for the Effective Direct Numerical Simulation of Compressible Turbulence. *Journal of Computational Physics 220* (2006), 270–289.

[87] Mehdizadeh Khalsaraei, M. An Improvement on the Positivity Results for 2-stage Explicit Runge-Kutta Methods. *Journal of Computational and Applied Mathematics 235* (2010), 137–143.

[88] Meng, Q., Luitjens, J., and Berzins, M. Dynamic task scheduling for the uintah framework. *In Proceedings of the 3rd IEEE Workshop on Many-Task Computing on Grids and Supercomputers (MTAGS10) (Washington DC, USA,2010), IEEE Computer Society.*

[89] Monaghan, J. J., and Gingold, R. A. Shock Simulation by the Particle Method SPH. *Journal of Computational Physics 52* (1983), 374–389.

[90] Monaghan, J. J., and Pongracic, H. Artificial Viscosity for Particle Methods. *Applied Numerical Mathematics 1* (1985), 187–194.

[91] Moon, K.-S., Szepessy, A., Tempone, R., and Zouraris, G. E. Adaptive Approximation of Differential Equations Based on Global and Local Errors. *TRITA-NA-0006 (2000), NADA, KTH, Sweden.*

[92] Moon, K.-S., Szepessy, A., Tempone, R., and Zouraris, G. E. A Variational Principle for Adaptive Approximation of ordinary Differential Equations. *Numerische Mathematik 93* (2003), 131–152.

[93] Moon, K.-S., Szepessy, A., Tempone, R., and Zouraris, G. E. Convergence Rates for Adaptive Approximation of ordinary Differential Equations. *Numerische Mathematik 93* (2003), 99–129.

[94] Nairn, J. A. Numerical simulations of transverse compression and densification in wood. *Wood and Fiber Science 38*, 4 (2006), 576–591.

[95] Parker, S. G. A component-based architecture for parallel multi-physics pde simulation. *Future Generation Computer Systems 22*, 1 (2006), 204–216.

[96] Parker, S. G., Guilkey, J., and Harman, T. A component-based parallel infrastructure for the simulation of fluid structure interaction. *Engineering with Computers 22*, 3-4 (2006), 277–292.

[97] Parker S. G. A Component-Based Architecture for Parallel Multi-physics PDE Simulation. *In International Conference on Computational Science (ICCS2002) Workshop on PDE Software April 21-24 2002. The Netherlands. Proceedings, Part III MA. Sloot, CJ Kenneth Tan, JJ Dongarra, AG Hoekstra(Eds). Lecture Notes in Computer Science 2002; 2331 Springer-Verlag GmbH, ISSN: 0302-9743.*

[98] Pember, R., Bell, J., Colella, P., Crutchfield, W., and Welcome, M. L. An adaptive Cartesian grid method for unsteady compressible flow in irregular regions. *J. Comput. Phys. 120* (1995), 278–304.

[99] Poinsot, T. J., and Veynante, D. *Theoretical and Numerical Combustion.* Edwards, 2001; ISBN 1-930217-05-6.

[100] Qiu, J., and Shu, C.-W. A Comparision of Troubled-Cell Indicators for Runge-Kutta Discontinuous Galerkin Methods using Weighted Essentially Nonoscillaroty Limiters. *SIAM J. Sci. Comput. 27*, 3 (2005), 995–1013.

[101] Quirk, J. An alternative to unstructured grids for computing gas dynamic flows around arbitrarily complex twodimensional bodies. *Comput. & Fluids 23* (1994), 125–142.

[102] Reinelt, D., Laurs, A., and Adomeit, G. Ignition and Combustion of a packed bed in a Stagnation Point Flow. *Combustion and Flame 99*, 2 (1994), 395–403.

[103] Roe, P. L. Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes. *J. Comput. Phys. 43* (1981), 357-372.

[104] Shampine, L. F. Error Estimation and control for ODEs. *Journal of Sci. Comput. 25*, 1 (2005), 3–16.

[105] Shampine, L. F. Local Error Estimation by Doubling. *Computing 34* (1985), 179–190.

[106] Shu, C.-W., and Osher, S. J. Efficient Implementation of Essentially Non-Oscillatory Shock Capturing Schemes. *J. Comput. Phys. 77* (1988), 439–471.

[107] Shu, C.-W., and Osher, S. J. Efficient Implementation of Essentially Nonoscillatory Shock Capturing Schemes II. *J. Comp. Phys. 83* (1989), 32–78.

[108] Sod, G. A. A survey of several difference methods for systems of nonlinear hyperbolic conservation laws. *Journal of Computational Physics 27* (1978), 1–31.

[109] Steffen, M. *Analysis-guided Improvements of the Material Point Method.* PhD dissertation, University of Utah, 2008.

[110] Steffen, M., Kirby, R. M., and Berzins, M. Analysis and reduction of quadrature errors in the material point method (MPM). *International Journal for Numerical Methods in Engineering 76*, 6 (2008), 922–948. DOI:10.1002/nme.2360.

[111] Steffen, M., Wallstedt, P. C., Guilkey, J. E., Kirby, R. M., and Berzins, M. Examination and Analysis of Implementation Choices within the Material Point Method (MPM). *In Computer Modeling in Engineering and Sciences 32* (2008), 107–127.

[112] Steffen, M., Kirby, R. M., and Berzins, M. Decoupling and Balancing of Space and Time Errors in the Material Point Method (MPM). *International Journal for Numerical Methods in Engineering 82*, 10 (2010), 1207–1243.

[113] Sulsky, D., Chen, Z., and Schreyer, H. L. A particle method for history-dependent materials. *Computer Methods in Applied Mechanics and Engineering 118* (1994), 179-196.

[114] Sulsky, D., Zhou, S.-J., and Schreyer, H. L. Application of a particle-in-cell method to solid mechanics. *Computer Physics Communications 87* (1995), 236–252.

[115] Sulsky, D., Schreyer, H., Peterson, K., Kwok, R., and Coon, M. Using the material point method to model sea ice dynamics. *J. Geophys. Res. 112* (2007). C02S90. DOI:10.1029/2005JC003329.

[116] Sweby, P. K. High Resolution Schemes using Flux-Limiters for Hyperbolic Conservation Laws. *SIAM J. Num. Anal. 21* (1984), 995–1011.

[117] Swensen, D. A., Denison, M. K., Harman, T., Guilkey, J., and Goetz, R. A Software Framework for Blast Event Simulation. *Reaction Engineering International, Salt Lake City, UT.*

[118] Thompson, K. W. Time Dependent Boundary Conditions for Hyperbolic Systems. *J. Comput. Phys. 68*, 1 (1987), 1–24.

[119] Titarev, V. A., and Toro, E. F. ADER schemes for three-dimensional nonlinear hyperbolic systems. *Journal of Computational Physics 204* (2005), 715–736.

[120] Toro, E. F. *Riemann Solvers and Numerical Methods for Fluids Dynamics: A Practical Introduction,* third ed. Springer, 2008; ISBN 978-3-540-25202-3.

[121] Toro, E. F., Hidalgo, A., and Dumbser, M. FORCE schemes on unstructured meshes I: Conservative Hyperbolic Systems. *Journal of Computational physics 228* (2009), 3368-3389.

[122] Tran, L.-T., and Berzins, M. Improved Production Implicit Continuous-fluid Eulerian Method for Compressible Flow Problems in Uintah. *International Journal For Numerical Methods In Fluids 69*, 5 (2012), 926–965.

[123] Tran, L.-T., Kim, J., and Berzins, M. Solving Time-Dependent PDEs using the Material Point Method, A Case Study from Gas Dynamics. *International Journal for Numerical Methods in Fluids 62*, 7 (2009), 709–732.

[124] Tucker, P. G., and Pan, Z. A Cartesian Cut Cell Method for Incompressible Viscous Flow. *Applied Mathematics Modelling 24* (2000), 591-606.

[125] Vallis, G. K. *Atmospheric and Oceanic Fluid Dynamics: Fundamentals and Large-scale Circulation.* Cambridge University Press 2006; ISBN 978-0-521-84969-2.

[126] Van Albada, G. D., Van Leer, B., and Roberts, W. W. A Comparative Study of Computational Methods in Cosmic Gas Dynamics. *Astron. Astrophysics 108* (1982), 76–84.

[127] VanderHeyden, W. B., and Kashiwa, B. A. Compatible Fluxes for Van Leer Advection. *Journal of Computational Physics 146* (1998), 1–28.

[128] van der Heul, D. R., Vuik, C., and Wesseling, P. A Conservative Pressure-Correction Method for Flow at All Speeds. *Computers and Fluids 3* (2003), 1113–1132.

[129] Van Leer, B. Towards the Ultimate Conservative Difference Scheme II. Monotonicity and Conservation Combined in a Second Order Scheme. *J. Comp. Phys. 14* (1974), 361–370.

[130] Van Leer, B. Towards the Ultimate Conservative Difference Scheme III. Upstream-Centered Finite-Difference Schemes for Ideal Compressible Flow. *J. Comp. Phys. 23* (1977), 263–275.

[131] Van Leer, B. Towards the Ultimate Conservative Difference Scheme V. A Second Order Sequel to Godunov's Method. *J. Comp. Phys. 32* (1979), 101–136.

[132] Van Leer, B. On the Relation Between the Upwind-Differencing Schemes of Godunov, Enguist-Osher and Roe. *SIAM J. Sci. Stat. Comput. 5* (1985), 1–20.

[133] Venditti, D. A, and Darmofal, D. L. Adjoint Error Estimation and Grid Adaptation for Functional Outputs: Application to Quasi-One-Dimensional Flow. *J. Comput. Phys. 164* (2000), 204–227.

[134] Versteeg, H., and Malalasekera, W. *An introduction to Computational Fluid Dynamics: The Finite Volume Method,* second ed. Prentice-Hall, 2007.

[135] Vshivkov, V. A. The approximation properties of the particles-in-cells method. *Computational Mathematics and Mathematical Physics 36*, 4 (1996), 509–515.

[136] Waterson, N. P., and Deconinck, H. A Unified Approach to the Design and Application of Bounded Higher-Order Convection Schemes. *In Numerical Methods in Laminar and Turbulent Flows, Proceedings of the Ninth International Conference* 1995; **9**(1):203-214.

[137] White, F. M. *Fluid Mechanics,* fifth ed. McGraw-Hill, 2003; ISBN 0-07-119911-X.

[138] Woodward, P., and Colella, P. The Numerical Simulations of Two-Dimensional Fluid Flow with Strong Shocks. *J. Comput. Phys. 54* (1984), 115–173.

[139] Ye., T., Mittal, R., Udaykumar, H. S., and Shyyy., W. An Accurate Cartesian Grid Method for Viscous Incompressible Flows with Complex Immersed Boundaries. *Journal of Computational Physics 156* (1999), 209–240.

[140] York, A. R., Sulsky, D., and Schreyer, H. L. Fluid-membrane interaction based on the material point method. *International Journal for Numerical Methods in Engineering 48* (2000), 901–924.