GPU ACCELERATED HIGH-ORDER DISCONTINUOUS GALERKIN LEVEL SET METHODS FOR INCOMPRESSIBLE MULTIPHASE FLOWS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ALİ KARAKUŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF DOCTOR OF PHILOSOPHY
IN
MECHANICAL ENGINEERING

SEPTEMBER 2015

Approval of the thesis:

GPU ACCELERATED HIGH-ORDER DISCONTINUOUS GALERKIN LEVEL SET METHODS FOR INCOMPRESSIBLE MULTIPHASE FLOWS

submitted by **ALİ KARAKUŞ** in partial fulfillment of the requirements for the degree of **Doctor of Philosophy in Mechanical Engineering Department, Middle East Technical University** by,

Prof. Dr. Gülbin Dural Ünver
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Tuna Balkan
Head of Department, **Mechanical Engineering** _____

Prof. Dr. Mehmet Haluk Aksel
Supervisor, **Mechanical Engineering Dept., METU** _____

Assist. Prof. Dr. Cüneyt Sert
Co-supervisor, **Mechanical Engineering Dept., METU** _____

Examining Committee Members:

Prof. Dr. İsmail Hakkı Tuncer
Aerospace Engineering Dept., METU _____

Prof. Dr. Mehmet Haluk Aksel
Mechanical Engineering Dept., METU _____

Assoc. Prof. Dr. Mehmet Metin Yavuz
Mechanical Engineering Dept., METU _____

Assist. Prof. Dr. Barbaros Çetin
Mechanical Engineering Dept., BU _____

Assist. Prof. Dr. Özgür Uğraş Baran
Mechanical Engineering Dept., TEDU _____

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: ALİ KARAKUŞ

Signature :

ABSTRACT

GPU ACCELERATED HIGH-ORDER DISCONTINUOUS GALERKIN LEVEL SET METHODS FOR INCOMPRESSIBLE MULTIPHASE FLOWS

Karakuş, Ali
Ph.D., Department of Mechanical Engineering
Supervisor : Prof. Dr. Mehmet Haluk Aksel
Co-Supervisor : Assist. Prof. Dr. Cüneyt Sert

September 2015, 182 pages

This thesis study focuses on the development of GPU accelerated, high-order discontinuous Galerkin methods for the solution of unsteady incompressible and immiscible multiphase flows with level set interface representation. Nodal discontinuous Galerkin framework is used for Navier-Stokes, level set evolution and reinitialization equations on unstructured elements. Computations are localized mostly near the interface location with an adaptive method to reduce computational cost without sacrificing the accuracy. An artificial diffusion approach with a modal decay rate based regularity estimator is used to damp out high frequency solution components near kinks. For the computation of interface equations, a multi-rate Adams-Bashforth time integrator is designed to avoid time step restrictions resulting from artificial diffusion stabilization and local mesh refinement. Implicit systems arising from the semi-explicit time discretization of the flow equations are solved with a matrix-free p-multigrid preconditioned conjugate gradient method to minimize memory requirements and to increase overall runtime performance. The developed numerical scheme is accelerated

using modern GPUs and many-core CPUs. Platform independence of the solver is achieved with an extensible multi-threading programming API as common kernel language that allows runtime selection of different computing devices and threading interfaces. Efficiency, scalability, local high-order accuracy and mass conservation of the method are confirmed through distinct numerical test cases.

Keywords: Discontinuous Galerkin Method, Incompressible Navier-Stokes, Multiphase Flow, Level Set, Interface Capturing, Reinitialization, Adaptivity, GPU-CPU Parallelization

ÖZ

ÇOK FAZLI AKIŞLAR İÇİN YÜKSEK BAŞARIMLI YÜKSEK SEVİYELİ SÜREKSİZ GALERKİN METODLARI

Karakuş, Ali
Doktora, Makina Mühendisliği Bölümü
Tez Yöneticisi : Prof. Dr. Mehmet Haluk Aksel
Ortak Tez Yöneticisi : Yrd. Doç. Dr. Cüneyt Sert

Eylül 2015 , 182 sayfa

Bu tez çalışmasında, zamana bağlı çok fazlı akışlar için yüksek başarımlı, level set ara-yüzey modellemesine dayalı, yüksek-seviyeli süreksiz Galerkin metodları geliştirilmiştir. Çok fazlı Navier-Stokes, ara-yüzey ilerleme ve tekrar ilklendirme denklemlerinin çözümünde noktasal süreksiz Galerkin sistemi kullanılmıştır. Kullanılan yerel uzaysal ağ adaptasyonu ile hesaplama yükü, doğruluktan feregat etmeden, ara-yüzey çevresinde yoğunlaştırılarak önemli ölçüde azaltılmıştır. Yüksek frekanstaki çözüm bileşenlerini sönümlemek için, modal katsayiların azalma oranına bağlı düzensizlik tahmini temelli bir yapay difüzyon stabilizasyonu yaklaşımı kullanılmıştır. Arayüzey denklemlerinin çözümünde, lokal grid adaptasyonu ve yapay difüzyondan kaynaklı zaman adımı boyutundaki kısıtlamalar, geliştirilen çok adımlı Adams-Bashforth metoduna dayanan yerel zaman integrasyonu yöntemiyle engellenmiştir. Akış denklemlerinin yarı-açık zaman ayrıklaştırılmasından kaynaklanan kapalı sistemler, hafıza gereksinimlerini azaltmak ve toplam çözüm performansını arttıtmak için, matristen bağımsız p- çoklu

ağ ön koşullayıcı konjuge gradyan metodu ile çözülmüştür. Geliştirilen sayısal metod, modern ekran kartı ve ana işlemciler kullanılarak hızlandırılmıştır. Çözücünün platformdan bağımsız olaral çalışması, hesaplama araçlarının çalışma sırasında seçilmesine olanak veren bir çok-çekirdek programlama dili kullanılarak elde edilmiştir. Geliştirilen metodun verimliliği, paralel hızı, yerel yüksek-seviyeli doğruluğu ve kütle korunumu farklı sayısal testler ile gösterilmiştir.

Anahtar Kelimeler: Süreksiz Galerkin Metodları, Sıkıştırılamaz Akışlar, Çok Fazlı Akışlar, Level Set, Ara-yüzey Yakalama, Yeniden İlklendirme, Adaptasyon, GPU-CPU Paralel Hesaplama

To my wife Serap and my parents

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude to my advisors Prof. Haluk Aksel and Prof. Cüneyt Sert for their continuous support to my Ph.D study and for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing of this thesis.

My sincere thanks goes to Prof. Tim Warburton for being the best host during my studies at Rice University, giving me chance to work with his fantastic research team and for his continuous help, contributions and support in all parts of this study. Special thanks to my colleagues from CAAM, Rice University, Dr. David Medina for introducing me multi-threaded parallelization, Dr. Rajesh Gandham for providing me the insight of multigrid techniques and Dr. Jesse Chan for many fruitful discussions about spectral approximation. Thank you all for sharing good moments at work as well as at Houston social life.

I am forever thankful to my colleagues at fluid mechanics division of Mechanical Engineering Department,METU for their friendship, support and creating a cordial working environment.

I gratefully acknowledge the partial financial support from Scientific and Technological Research Council of Turkey (TUBITAK) International Doctoral Research Fellowship Program (2214).

I deeply thank to my family: my parents and to my brother and sister for their love, support, and sacrifices. Without them, this thesis would never have been written. I have saved the last words of acknowledgment for my dear wife Serap, who has been with me all these years and has made them the best years of my life.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

AB	Adams-Bashforth
CPU	Central Processing Unit
CFL	Courant-Friedrichs-Lewy criterion
DG	Discontinuous Galerkin
DGM	Discontinuous Galerkin Method
FE	Finite Element
FEM	Finite Element Method
Fr	Froude Number
GPU	Graphical Processing Unit
HJ	Hamilton-Jacobi
INS	Incompressible Navier-Stokes
LS	Level Set
LSF	Level Set Function
LSM	Level Set Method
MRAB	Multi-rate Adams-Bashforth
MRRK	Multi-rate Runge-Kutta
Re	Reynolds Number
RK	Runge-Kutta
SSP	Strong Stability Preserving
2D	Two Spatial Dimension
3D	Three Spatial Dimension

CHAPTER 1

INTRODUCTION

In the first part of this chapter, literature survey is presented to point out the current state of art. General approaches including interface capturing and interface tracking methods for the numerical solution of multiphase flows are presented in the first section. Second section covers the development of discontinuous Galerkin methods for first order hyperbolic equations emphasizing the theoretical investigation of convergence and stability properties. Special techniques to handle the second order operators which play an important role for modeling Navier-Stokes and Stokes equations are discussed in the next section. In the second part, motivation of the thesis study is summarized which is followed by the outline.

1.1 Multi-Phase Flows

Multiphase flow denotes the systems in which more then one phase are simultaneously present. In this definition, phases refer to fluids of the same substance such as a liquid and its vapor or fluids of different substances such as liquid and gas or fluid and solid, all coexisting in the same domain and at the same time. It is hard to define multiphase flows precisely but common to all, they are very complex due to mutual interaction of subsystems. In this thesis work, we investigate a specific multiphase flow i.e. incompressible and immiscible flow where the phases are separated with a well definite, sharp interface. These types of multiphase flows are generally called free surface flows.

Immiscible multiphase flows occur in many areas of practical importance such as phase change problems, flow-structure interactions, reacting flows, fluid-fluid inter-facial dynamics. Numerical prediction of the multiphase flow with sharp interfaces are challenging due to locating position of the dynamic interface, capturing topological changes, handling discontinuous material properties and varying range of scales. In multiphase flows, numerical methods can be classified as Lagrangian (front tracking) and Eulerian (interface capturing) based on the explicit or implicit representation of the interface. Also, it can be classified as surface and volume methods according to defining or reconstructing the interface position, respectively. Main properties of these methods are covered in subsequent sections.

1.1.1 Surface methods

In this class of methods, interface is marked with some special marker points, mesh itself or values of some continuous functions. The most important advantage of the method is that the location of interface can be known explicitly during the computation and interface remains sharp while convected. Fig. 1.1 illustrates the common surface methods used for representing interface dynamics.

The particle based surface methods [51, 86] use massless markers inserted on the interface as shown in the Fig. 1.1(a). Then, they are convected with a known velocity field to find their new locations. Interface and its geometric properties such as normal and curvature are approximated by the interpolation between subsequent particles generally using the linear polynomials. This approach is very sensitive to the spacing between markers. If the spacing is too large, interface can not be represented accurately. On the other hand, if it is too small, local fluctuations increase which destroy the accuracy of the approximation especially in curvature and surface tension computations. To retain the regularity of the marker spacing, it is required to add or remove particles dynamically during the simulation. This situation creates a new problem related with the renumbering of particles to compute geometric properties. Keeping markers in sequence is

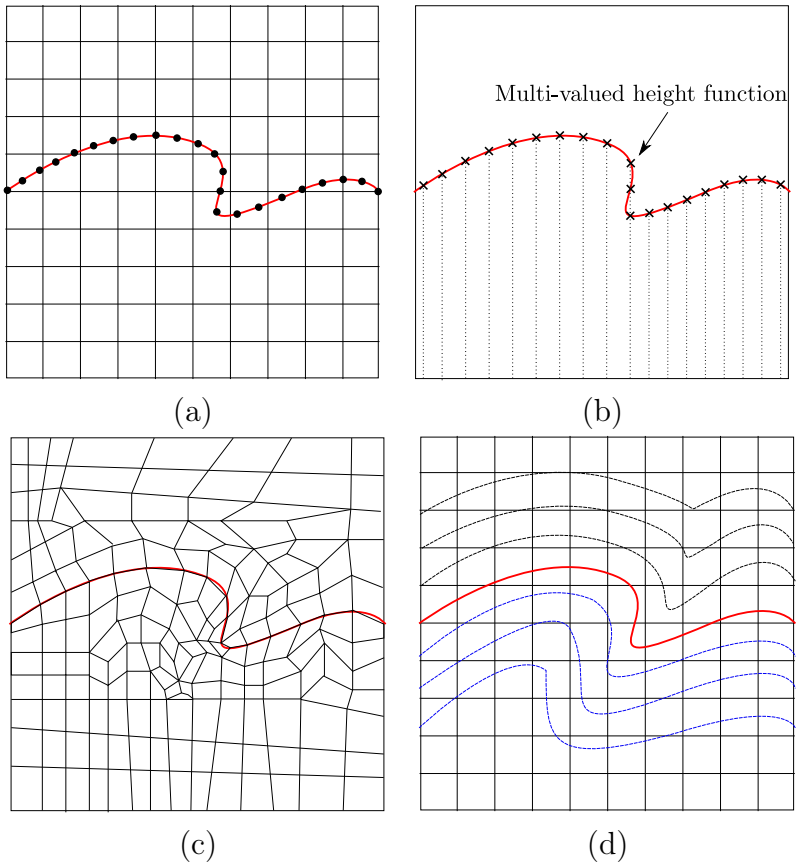


Figure 1.1: Common surface methods (a) Particle method (b) Height function approach (c) Surface fitted method (d) Level set method

the main drawback of the method because it puts restrictions when the interface encounters topological changes such as merging or breaking up. Also, extending the method to 3D problems requires sophisticated bookkeeping strategies to handle 2D connections on the interface which is practically impossible.

In [129], marker particles approach is extended by relating a point on the interface to a corresponding point on a reference plane. Interface is then represented by a height function defined from the reference plane. Fig. 1.1(b) illustrates a schematic description of the method. Each point on the reference plane can only represent a single value which is the main limitation of the method. In other words, it is impossible to approximate interface location when the height function is multivalued that is encountered frequently in the topological changes. But, the method is very efficient in terms of memory requirements, computational time and easy to generalize 3D problems of the non-complex interface

dynamics.

Another class is the surface fitted methods based on the attaching of mesh to interface [190, 144]. Fig. 1.1(c) shows the basic description of the method. As seen in the figure, the method provides a sharp interface profile enabling accurate application of boundary conditions at interface. However, interfaces subjected to large deformations cause highly distorted elements. In those cases, frequent mesh optimization or re-meshing procedures are required to obtain regular grids which lead to considerable computational burden in highly dynamic problems. Also, interpolation between old and new mesh structures destroys the optimal accuracy in high-order numerical methods.

Level set method (LSM) [133] is one of the most popular interface capturing method due its simplicity and efficiency. A continuous function defined on the whole computational domain is used to separate the phases. This function takes a constant value on the interface, which is generally zero. Smaller and larger function values indicate each phase. The scalar LS function is propagated with the flow velocity field to represent the dynamics of interface. Generally, signed distance is used as the LS function with zero iso-contour showing the current location of the interface. Because signed distance function is symmetric and regular around the interface, discontinuous material properties can be smoothed using the LS function leading to a continuous problem governing all the fluid flow. An LS function is illustrated in Fig. 1.1(d) with interface, negative and positive iso-contours shown with red, blue and black colors respectively. Although interface always remains at zero contour level, initial distance function loses its property and creates non-uniform gradients in the vicinity of the interface. Then, LS formulation uses reinitialization step to recover the signed distance function for numerical stability and accuracy [181]. LS interface modeling offers many advantages, such as straightforward extension from 2D to 3D, simple handling of topological changes and direct calculation of geometric properties due to implicit representation with continuous functions. The details of LS formulation will be given in the following chapters.

1.1.2 Volume Methods

Volume methods mark the fluid domains in both sides of the interface. Unlike surface methods, interface location is not known explicitly through the solution time. Because of this drawback, special techniques are required to capture interface shape and related geometric properties.

In the marker and cell method (MAC) [76, 75] massless particles are distributed over the one of the fluid domain. Empty elements indicate second fluid domain and first neighbors of the empty cells with marker particles contain the interface. Then, each particle is moved with its own velocity in the domain. MAC and its variants are very powerful in capturing complex interface dynamics and can be used in 3D problems easily. But, MAC methods require high computational effort and memory because of tracking of all particles and storing their coordinates in time. A simple illustration of MAC method can be found in Fig. 1.2(a).

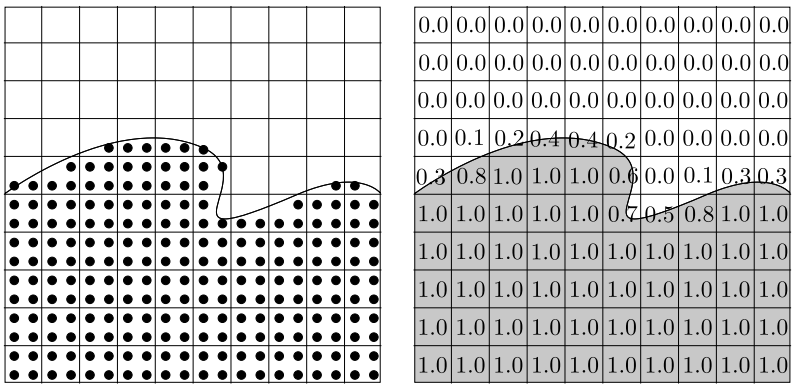


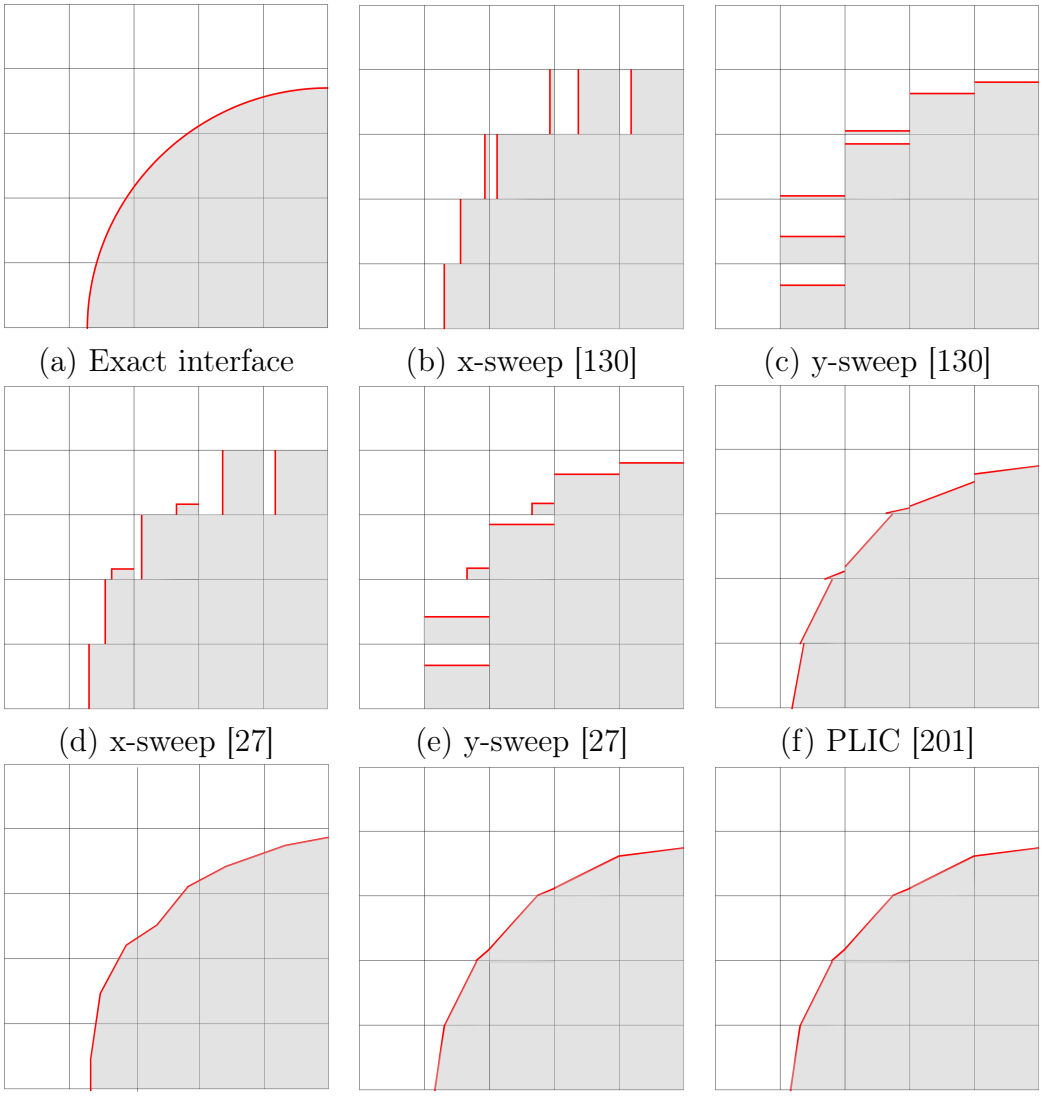
Figure 1.2: Common volume methods (a) marker and cell (b) volume of fluid element.

The volume of fluid element (VOF) [83] method uses volume of fractions as the indicator function to distinguish the phases. Volume fraction of zero indicates presence of one fluid and a value of unity indicates second fluid. Volume fraction between these two values shows the presence of interface as shown in Fig. 1.2(b). Method is very efficient in terms of computational time and memory because, only a scalar volume fraction should be stored and convected to evolve in time.

Also, VOF method directly enforces the mass conservation, unlike the LSM, in incompressible flows where volume conservation directly implies mass conservation. Due to its attractive properties, VOF methods are used widely in the multiphase flow simulations and will be investigated briefly.

Standard discretizations of VOF evolution lead to numerical dissipation and smearing off interface over the cells. Special discretization techniques are needed to overcome this problem. These techniques can be classified as donor-acceptor formulation and reconstruction [189]. The basic idea of the donor-acceptor formulation is to switch between downwinding and upwinding elements in flux calculation using slope information of the interface to ensure the boundness. Early VOF method [83] and some improved versions such as CICSAM [189], fall into this category. Reconstruction based algorithms predict the interface geometry using the volume fractions of some neighboring elements to evolve the volume fraction equation. The Simple Line Interface Calculation (SLIC) [130] is the first VOF reconstruction scheme where the interface is represented by a piecewise-constant line in each of the two fluid cells either vertically or horizontally. Then, this scheme is slightly improved by considering the direct neighbors of the elements called corner element concept in [27]. Youngs [201] introduced the piecewise linear interface calculation, (PLIC) and achieved a significant improvement. Interface representation is further improved by using flux line-segment model [5], least squares fit (LVIRA) [145], and parabolic (PROST) [152] reconstruction techniques. Fig. 1.3 shows schematic description of the different interface reconstruction algorithms. Detailed comparison of popular VOF advection and reconstruction techniques can be found in [69] and more recently in [2].

LS and VOF are the common methods used in multiphase flow simulations. Both methods have strengths and weaknesses. VOF methods are strictly mass conserving, can deal with topological changes but reconstruction of interface is complex and generalization to high order numerical methods and unstructured meshes are difficult. Because of that, VOF methods are second order accurate at most. On the other hand, LS method has great capability of geometric representation of interface and discontinuous properties. But, LS equation does



(a) Exact interface (b) x-sweep [130] (c) y-sweep [130]

(d) x-sweep [27] (e) y-sweep [27] (f) PLIC [201]

(g) FLAIR [5] (h) LVIRA [145] (i) PROST [152]

Figure 1.3: Some VOF interface reconstructions techniques.

not enforce the fluid discrete mass conservation principle even in the conservative formulations. Special care should be considered for the mass conservative, efficient solutions of the LS equations.

1.2 Discontinuous Galerkin Methods

Discontinuous Galerkin (DG) methods are a class of finite element methods that make use of completely discontinuous, piecewise polynomial approximations for spatial discretization and have excellent numerical dissipation properties which

is essential for obtaining mass conservative interface models. Due to relaxed strong elemental connectivity, DG methods are very flexible and well suited for local mesh and polynomial order based adaptations. In this section, basic properties of the DG method will be given, and advantages / disadvantages of the method compared with the well known classical methods will be pointed out. Then, a brief literature survey of the DG methods will be provided.

1.2.1 Overview of Basic Numerical Methods

All numerical methods differ in the sense of representation of approximate solution and satisfying the underlying partial differential equation. For example, in the oldest, finite difference method (FDM) spatial derivatives are approximated by simple difference equations and residual of the system is expected to vanish at each nodal point. This gives simple finite difference equations in the size of total nodal unknowns. The main advantages of the method are its simplicity, efficiency, robustness and potential to obtain high-order approximation. But, simple one dimensional approximation structure comes with the price that method is not suitable for complex geometries and discontinuous properties.

To obtain geometric flexibility, natural approach is to introduce element based discretization leading to well known finite volume method (FVM). In its simplest form, elemental unknowns are approximated with piecewise constant space and underlying PDE is satisfied such that cell average of residual vanishes at the center of element. Because, cell centered unknowns approximated by local piecewise constant space, global problem or communication of elements is obtained by fluxes which require reconstruction of boundary information from cell averaged values. FVM is easy to implement, robust and leads to pure local scheme but increasing the order requires to increase stencil size to reconstruct solution at element boundaries from cell center values, which is practically impossible.

A solution to obtain high-order approximation is to add more degrees of freedom to element. Finite element method (FEM) basically uses the global and symmetric polynomial space to approximate the unknowns. Residual is expected to be orthogonal with respect to all members of polynomial space. Because approx-

imation space is globally defined, boundary degrees of freedoms are shared by neighbors leading to a global problem. This implicit structure is the basic disadvantage of the method over FVM and FDM methods for the problems with explicit time dependency. Also, symmetry of the approximating polynomials may cause stability issues in highly wave dominated problems without considering special techniques. Note that directionality of the problem can be easily handled in FDM and FVM by using upwinding.

DG method can be considered as the intelligent combination of classical FEM and FVM where local high-order accuracy is achieved by using higher order approximation space. But, in this case, polynomial space is locally defined only on the element. Global problem is obtained weakly by utilizing numerical fluxes between elements as in FVM. As a result, DG method combines powerful properties of FEM and FVM. At the first glance, increase in the total degree of freedom, where boundary degrees of freedoms are doubled due to relaxed strong element connectivity, may be considered as a big disadvantage. However, these arguments are largely misleading. First, ratio of boundary degree of freedom to volume degree of freedom becomes smaller with increasing approximation order and it is well known that using higher approximation space scale more effectively then spatial refinement. Also, DG methods are well-suited for modern supercomputer architectures due to its locality and computational intensity (over memory access) offering more scalable solutions than any other numerical method. In Table 1.1, generic properties of common numerical methods are given. Note that, the table includes basic properties of methods and many special techniques are introduced in each context. For more information, please refer to [82].

Table 1.1: Generic properties of common numerical methods. + and x indicate succes and failure respectively, while (+) symbol indicates that method requires modications to be capable for solving the problem [82].

	FDM	FVM	FEM	DG
Complex geometries	x	+	+	+
High-order accuracy	+	x	+	+
Explicit semi-discrete form	+	+	x	+
Wave dominated problems	+	+	(+)	+
Elliptic problems	+	(+)	+	(+)

1.2.2 Development of Discontinuous Galerkin Methods

Discontinuous Galerkin Method is first introduced by [151] for the steady state neutron transport equation. First analysis of the method for linear problems with smooth solution is presented by [105], and showed that $O(h^N)$ convergence on triangular grids, $O(h^{N+1})$ optimal convergence on Cartesian grids for the polynomial approximation of order N. Later this result is improved in [89] to $O(h^{N+1/2})$ for general grids. The scalar hyperbolic equations with non-smooth solutions (discontinuous initial data) are studied in [110, 33]. For linear problems, to enhance accuracy to $O(h^{2N+1})$, post processing techniques are proposed over uniform meshes [41, 155] and non-uniform meshes [50]. A general discussion for the discretization of hyperbolic systems is given in [16, 58] and techniques for determination of numerical fluxes to minimize the dissipation are presented in [18, 29].

The method was first used to non-linear scalar conservation laws in [24] by using linear element for space discretization and Forward-Euler method for time discretization resulting unstable solution except for very limiting time step size [82]. This problem was resolved in [23] by introducing slope limiters and obtained stable solution but with low order approximations for both space and time. Order of approximation is increased in [43] by combining the previously developed Runge-Kutta method [164] with enhanced slope limiters [163]. Then, this Runge Kutta Discontinuous Galerkin (RKDG) method is extended to higher accuracy by introducing generalized slope limiter and non-linearly stable Runge-Kutta method in [28] and for one dimensional systems in [40]. Extension of the method to multi-dimensional cases are complex because of the multi-dimensional slope limiting which is first completed for the scalar problems in [35] and then non-linear conservation laws by introducing the slope limiters based on the local maximum principle in [44]. The RKDG uses the ideas from high order finite volume schemes such as exact or approximate Riemann solvers, Total Variable Diminishing (TVD) Runge-Kutta time discretization and limiters. In [146] fifth order Weighted Essentially Non-Oscillatory (WENO) schemes (which is developed for the finite difference type methods) based on the Hermite polynomials,

termed HWENO is applied to RKDG method as a limiter. This method is also applied to the two dimensional non-linear hyperbolic problems in [148]. A comparison of troubled cell (cell where limiting may be needed) indicator methods is given in [147]. WENO scheme is also used for the RKDG method in [149]. But all of the applications of *WENO* and HWENO schemes to RKDG are limited to simple grids only. An extension to 2D and 3D unstructured grids is presented in [114]. Recently in [206], WENO scheme is applied to Runge-Kutta Local Discontinuous Galerkin Method (RK-LDG) over unstructured meshes.

The DGM is originally developed for hyperbolic system of equation. Extension of method to problems with second order operators was initiated in [10] by using a Riemann solver for fluxes and mixed formulation (by rewriting second order operators into system of first order equations). The method is often termed as Bassi-Rebay flux (BR flux or central flux). This approach is revised in [45] and lead to introduction of Local Discontinuous Galerkin (LDG) method. At the same time, an alternative formulation is presented in [6, 11] which is known as Interior Penalty (IP) methods having been developed much earlier. An overview of methods, their differences and properties is given in [4]. In [203] a detailed comparison of these three methods is presented and close results are obtained for LDG and IP method by suitable selection of penalty parameter and lower accuracy in BR method. The methods further developed in terms of correct choice of penalty terms, stabilization methods and error analysis in [20, 68, 160, 15]. A new technique is presented in [139] which is termed as Compact Discontinuous Galerkin (CDG) method which is closely related to the LDG but increases the compactness in multi-dimensional cases. Recently for the elliptic problems, hybridized LDG (LDG-H) and super-convergent version of this method is presented in [31] and [30, 34], respectively.

1.2.3 Incompressible Navier-Stokes and Stokes Equations

These developments have opened up a new range of applications, specially compressible and incompressible fluid flows. The DG solution of incompressible Navier-Stokes (INS) equation is first suggested by [111]. They used vorticity

stream-function formulation for 2D incompressible flows and while momentum equation is treated explicitly utilizing the DG method the stream function is solved by continuous finite elements. There is a natural balance between continuous and discontinuous frameworks in 2D because normal components of velocity field is continuous through element boundaries but in 3D normal velocity is no longer continuous at element boundaries and the method cannot be used. The suggested methodology utilizes a second order Godunov type fluxes for convective terms and central flux (or BR flux) [10] for viscous terms. Later, Cockburn and coworkers proposed and analysed LDG method for Stokes [39], Oseen [37] equations and published a review of these works [38]. Then, they further developed these methods for stationary incompressible Navier-Stokes equations and suggested two strategies in [36]. In the first one, the main difficulty in enforcing incompressibility is overcame by discretization of Oseen equation where convective velocity is taken to be projection of approximated velocity into space of globally divergence-free functions and through an iterative procedure locally conservative velocity field is achieved. In the second one, the pressure is replaced with the Bernoulli pressure and locally conservativeness is attained. Main drawback of the first method proposed in [36] is locally conservativeness is achieved by an iterative procedure and second one is unphysical solution at outflow boundary conditions as stated in the context of continuous Galerkin approximation.

In [161] a DG method is presented for unsteady INS equations. The proposed method is based on semi-implicit temporal discretization with explicit treatment of convective terms and implicit treatment of viscous terms. For the discretization of viscous terms they used IP formulation and applied both $P^N - P^N$ and $P^N - P^{N-1}$ formulations for approximation of velocity and pressure. Nonlinear terms discretized in divergence form (yields local conservativeness immediately) and they used the local Lax-Friedrichs flux to obtain stable results. In [9] inviscid numerical fluxes in continuity and momentum equations are computed by the exact Riemann problem associated with the local artificial compressibility. Although artificial compressibility has been extensively used in finite volume and finite element approximation of INS equations, in this work artificial com-

pressibility is deployed only for the construction of interface fluxes leading the scheme independent from the amount of artificial compressibility added because interface flux reduces physical one for vanishing interface jumps. In this work viscous fluxes are computed by BR [10] method and time integration is established in a fully implicit method. In [60] a DG finite element solver is proposed for 2D incompressible Navier-Stokes equations. They have used second order stiffly stable method [93] to discretize the equation in time leading a non-linear convection equation, a Poisson equation for pressure and Helmholtz equation for viscous terms and for spatial discretization, Lesaint-Raviart fluxes for convective terms and Symmetric Interior Penalty (SIP) method [153] for elliptic equations. In their solver, a modal expansion hierarchical basis functions are used to approximate the solution.

The attractiveness of DG method is mainly due to its stability properties in convection dominated problems, its efficiency in high-order approximations which allowing *hp* adaptive refinement, its local conservativeness leading superior properties for parallelization. But computational cost of the DG method is higher than the continuous finite element and finite volume methods due to doubled degree of freedoms at element boundaries resulting from the relaxed inter-element continuity condition. To decrease the degrees of freedom in both velocity and pressure, different DG formulations are proposed in the literature. In [7] and [90], a DG formulation is proposed with a piecewise polynomial divergence-free velocity with optimal error bounds. This formulation uses the continuous pressure approximation and only Drichlet boundary conditions are applied because it is not easy to handle Neumann boundary condition in this formulation. Toselli [185] proposed a DG method for the Stokes equation with piecewise polynomial approximations without imposing the divergence free condition and obtained better stability properties than the continuous Galerkin approximations and uniform divergence stability is proven when velocity is approximated one or two degrees higher than pressure. Cockburn and co-workers [19] proposed a solenoidal piecewise polynomial approximation for the DG formulation of Stokes equation. The method is derived from LDG rationale on mixed formulation with velocity, vorticity and pressure. They used the concept of hybrid

pressure which is the pressure along the element sides and pressure inside the element is computed as a post-process of the *LDG* solution. In [124] a new DG formulation is developed with solenoidal polynomial velocity and hybrid pressure, in fact this method is very similar to method in [19] but derived from IP method rather than LDG and produce similar symmetric and coercive bilinear form for velocity. Montlaur et.al. [125] then extended this procedure for incompressible Navier-Stokes equations and developed the methodology in the framework of IP and CDG methods obtained compact formulations compared with the LDG method. Recently Nguyen et al. [128] suggested a HDG method for the incompressible Navier-Stokes equations by extending their methodology for Stokes system [32, 127, 42]. They showed that methodology have many advantages over the other DG methods in terms of reducing the globally-coupled degrees of freedom, in the convergence and accuracy properties of approximation and ability to handle wide range boundary conditions.

1.3 High Performance Computing

The history of super computers started when many single-core vector machines were connected to each other. These machine networks were coordinated from a single node by some special software. Starting from early 2000s, multi-core processors started to use as accelerators for super computers. After less than a decade of cell accelerators' development, graphics processors units currently dominated the co-processor cell architectures, except the Intel Xeon Phi processors. Although, these co-processors are not widely utilized on the Top500 Supercomputer list [184], the machines at the top of the list are equipped with them.

In this section, multi-core central processing units (CPU), graphics processing units (GPU) architectures, both are general-purpose parallel architectures, and programing approaches for these architectures will be briefly reviewed. In addition to this, an overview of OCCA which is a unified approach for heterogeneous computing will be given

1.3.1 Central Processing Units

Floating point units (FPU's) were used as co-processors to central processing units in order to make floating precision works until they became integrated with the CPU's. For many years, Moore's law accurately predicted that the number of transistors in a dense integrated circuit double every two years. In parallel with Moore's Law, peak floating points operations (FLOPS) increased exponentially for many years. Currents and voltages were scaled proportionally to transistor sizes also known as Dennard scaling to control power utilization and maintain the transistor density rate suggested by Moore's Law once power limitations became obvious [150]. After 2000's, performance scaling was slowed down and further architectural improvements were required to continue scaling.

Inspired from Beowulf clusters (small networks with a few computers), multi-core central processing units where each core containing an independent instruction scheduler were born. In years, processors with 2 to 16 cores became the industry standard. Increase in number of cores resulted that single instruction multiple data (SIMD) cores became common on general-purpose and performance based processing units. Although the idea of using the same instruction over and over on multiple data was already used since the CRAY-1 in 1976 [154] current SIMD vectorization units not only allow reuse of instructions but also decrease overall fetching latency.

Over the years, latency and bandwidth improved at different rates. Nowadays it came to a point that memory bandwidth is main bottleneck for most of the problems because of high rates of the processor clock speed development for many years. In order to cover the memory bandwidth problem, multilevel caches were introduced to predict and pre-fetch data directly for processors to avoid excessive data transfers between the RAM and processor.

Co-processors dedicated to specific computational tasks were built up, besides the advances in CPU architecture. For co-processors , frequent usages presently are graphics-processing and embarrassingly – parallel computations, whereas it is the central processor for all around computations. Famous co-processors have

graphics processing units (GPUs), field programmable gate arrays (FGPAs), Intel's Xeon Phi, and IBM's Cell architecture. On the other hand, architectures designed for performance with low-power rates are also presented like massively parallel processor arrays (MPPAs).

1.3.2 Graphical Processing Units

The main reason behind the fast development of graphics cards were demand for better graphics in video games. Graphics cards architectures were designed for linear algebra operations in order to do 3D rendering. They firstly utilized for assisting to general purpose CPU's in graphics related tasks then became a core component. Use of identical instructions on many pixels for graphics rendering practically proves that GPU's has an embarrassingly parallel architecture. Programming graphics cards are composed of a fixed pipeline, splitting work into various stages for updating vertices and managing single fragments. The Brook language was constructed [17] in order to cut down the complexity of the fixed pipeline usage for general purpose computations as a result of the potential of graphics card for that purpose. After Brook proposed that the graphics cards could be used for general purpose computing in 2007, NVIDIA developed CUDA for using their graphics cards in general purpose GPU's (GPGPUs). However usage of CUDA is restricted to boost hardware by NVIDIA's developers [61].

To solve this limitation, the Khronos group introduced OpenCL standard for heterogeneous programming including CPUs, CPUs and Intel's Xeon Phi. Then, Intel and AMD released their OpenCL in 2008. Currently, CUDA and OpenCL languages, or some applications utilizing these languages, are used for programming modern multi-threaded architectures.

1.3.3 OCCA

OCCA [120] is an abstracted programming model used to encapsulate native languages for many-core devices developed by D. Medina and T. Warburton from Rice University. OCCA creates an intermediate representation (IR) to inte-

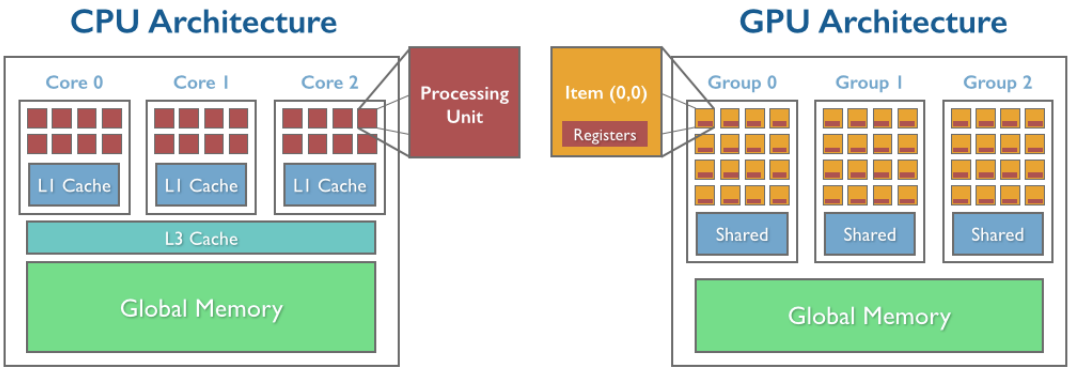


Figure 1.4: Modern CPU and GPU architectures.

grate different languages and standards such as serial code, Pthreads, OpenMP, CUDA, OpenCL, and COI. Using this programming approach therefore allows customized implementations of algorithms for several computing devices with a single code and offers flexibility to choose hardware architectures and programming model at run-time.

Fig. 1.5 represents the relation of OCCA application programming interface (OCCA API), kernel languages and intermediate representation (IR) with supported platforms and hardwares. As seen from the figure OCCA supports many programming languages, (frontends) such as C, C++, Phyton, etc. and all many-core architectures (backends) through standard CUDA, OpenCL and OCCA OKL (C type) and OFL (Fortran) kernel languages.

1.4 Motivation and Contributions

Discontinuous finite element methods have many distinct properties such as local conservativeness, stability, high-order accuracy and also handling of complex geometries, irregular meshes with hanging nodes and different order of approximations in different elements. These properties render the methods ideal to be used with adaptive strategies and parallelization. LS method has great capability of geometric representation of interface and discontinuous properties, but does not enforce discrete mass conservation principle. Although, many hybrid methods are proposed to improve this drawback, they cause the level set

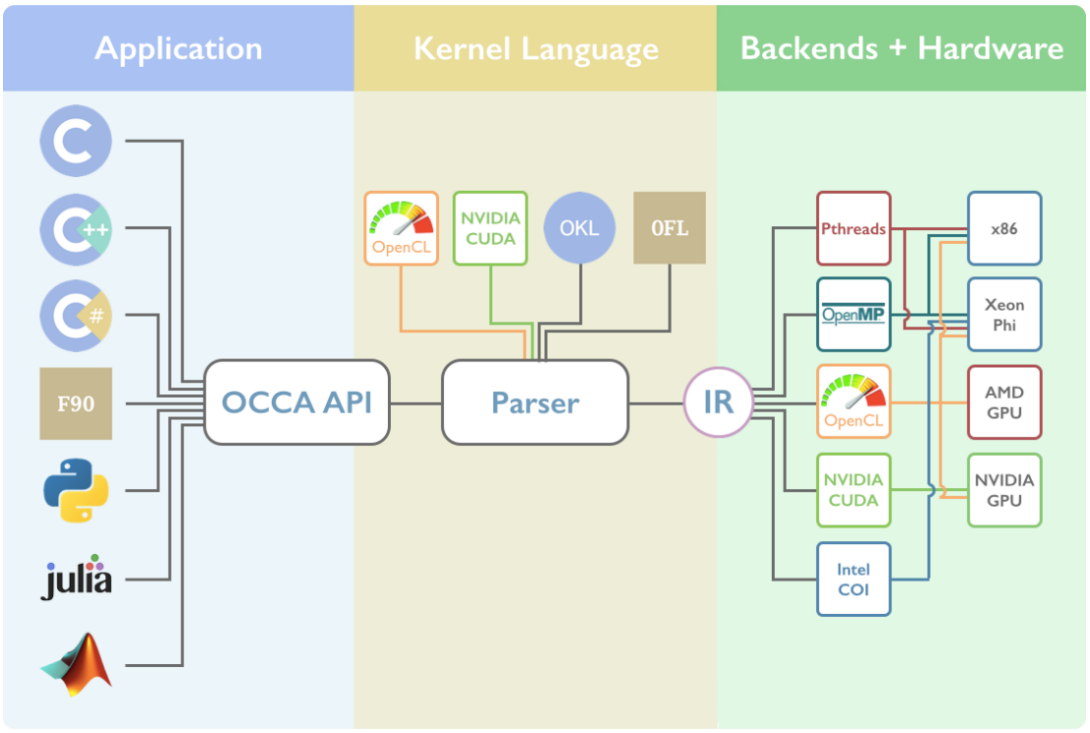


Figure 1.5: Structure of OCCA API with relationship between supported frontends and backends [120].

formulation to lose its simplicity and efficiency.

The aim of this research is to develop a robust, accurate and flexible numerical method combining the local high order approximation with an enhanced mass conserved level set method to predict the complex flow phenomena between two immiscible fluids separated by well-defined interface on unstructured triangular/tetrahedral elements. To meet these requirements, research comprises the followings,

- A high order discontinuous Galerkin level set method is developed on unstructured grids. The formulation preserves the simplicity of the original level set method while mass conservation properties are significantly improved. It is shown that mass loss problem of the method is not related with the formulation but with the discretization scheme used.

- To avoid the time step restriction resulting from explicit treatment of level set equations, a multi-rate Adams-Bashforth local time stepping technique

is proposed. Combining the local adaptive unstructured grid with hanging nodes, computational load is significantly reduced and optimal speedups are obtained.

- A numerical scheme for the high-order reinitialization of level set function is proposed in discontinuous framework. Stabilization of this high-order scheme is accomplished with artificial diffusion mechanism. Marking troubled elements rely on the modal decay rate based regularity detector which is modified from the 1D methodology proposed by [98]. Severe time step restriction due to artificial diffusion and high order mesh dependent parameters are avoided with the developed local time stepper. The proposed approach enables to achieve optimum convergence rates on both conformal and non-conformal discretizations and is suitable for parallelization on many-core and multi-threaded architectures.

- A fully discontinuous Galerkin multiphase flow model is presented. Combining with the efficient adaptive strategies, multiphase model handles high density/viscosity ratios, topological changes and mass loss problem. To increase the memory efficiency and decrease computational effort, matrix-free solution technique is used with a novel matrix-free p-multigrid preconditioner for dynamic implicit systems.

- All numerical schemes mentioned above are designed for the full parallelization. To the best knowledge of the author, this is the first fully discontinuous scheme developed for modern multi-threading and many-core processing units. The numerical method is implemented in a platform independent approach where low level parallelization is achieved by run-time code generation. High scaling is obtained in all graphic and central processing units.

1.5 Outline of Thesis

The remainder of the thesis study is structured as follows,

Chapter 2. Implementation details of discontinuous Galerkin method is in-

troduced. Starting from the node distribution and discontinuous approximation spaces, local operators are defined for triangular and tetrahedral elements. Then, source of aliasing errors and cubature integration are explained. Chapter is closed with efficient flux calculation on conformal and non-conformal face pairs.

Chapter 3. A discontinuous Galerkin level set method is introduced in this chapter. A multi-rate Adams-Bashforth time stepping is designed to overcome time step restriction introduced by the local mesh adaptivity. Efficient parallelization of the method is also given.

Chapter 4. This chapter is devoted to the high order reinitialization of the level set function. Artificial diffusion stabilization and troubled cell indicator are introduced. Then, efficient solution of the system utilizing the explicit local time stepping and massive parallelization is given.

Chapter 5. Multiphase flow solver on adaptive unstructured grids are presented. Matrix-free solution procedure and p-multigrid preconditioner are also explained. Then, modified reinitialization scheme for local level set formulation is presented. GPU-CPU hybrid parallelization and scaling of the numerical method developed is presented in this chapter. Also, accuracy and efficiency of the new method is tested for several distinct problems.

CHAPTER 2

BUILDING NODAL DISCONTINUOUS GALERKIN SCHEME

In this chapter, some well-known facts regarding the polynomial interpolation in a d-dimensional simplex, i.e. triangular and tetrahedral elements, are investigated. Optimal nodal points and orthogonal polynomials are discussed in the first section. Following this section, geometric and polynomial notations required for the rest of the thesis are included. Then, discontinuous Galerkin local operators are derived for efficient implementation on parallel architectures.

2.1 Interpolation Concepts

High quality interpolation of functions plays an essential role in the success of continuous/discontinuous spectral element methods. A good interpolation basically needs suitable orthogonal basis functions and appropriate interpolation points. On quadrilateral/hexahedral elements, spectral element methods employ a nodal approach based on the Legendre polynomials and tensor product of one dimensional Gauss-Lobatto-Legendre (GLL) points. Such a straightforward approach is no longer possible for non-tensorial triangular/tetrahedral domains [137].

2.1.1 Nodal Distribution

The first factor in the quality of high order polynomial approximation is the location of interpolation points. Interpolation using uniformly distributed points yield poor approximation quality as the polynomial degree increases even for the smooth analytical functions. This problem is solved for tensor product domains by recurring Gauss-Lobatto points, but for non-tonsorial domains it is still an open question.

2.1.1.1 Triangular Nodal Distribution

For triangular elements, a widely used approach is proposed in [92] based on the idea of changing coordinate system to transform quadrangle and its quadrature points into triangle. Main drawback of collapsed coordinate transformation approach is that interpolation points are not symmetrically distributed over the triangle and condensed around one vertex of the standard triangle. An early nodal set referred as Fekete points, which maximize the Vandermonde matrix is presented in [13] up to polynomial order of 7. Fekete points are one possible generalization of GLL points for triangle because it is known that GLL points are the Fekete points for any d dimensional cube [14]. This approach is further improved and extended to 13^{th} order in [25] and 18^{th} order in [183]. The same method is enhanced in [77] by using more sophisticated optimization techniques with revised algorithm to find nodal sets starting from Fekete points with minimization of Lebesgue constant. Definitions of Vandermonde matrix and Lebesgue constant will be presented in the following sections.

A different physically motivated approach based on the early observation in [170] is presented in [81] relying on the fact that equilibrium configuration of electric charges constrained to lie on the bi-unit interval coincides with the location of some Jacobi polynomials. This implies that GLL quadrature points coincide with the equilibrium position of electric charges. Hesthaven [81] extended this analogy to compute nodal distribution in a triangle by looking for the equilibrium position of charges distributed in a triangle by fixing line charges on the

boundary.

All the methods discussed so for result from optimizing interpolation quality of the nodes by varying their coordinates. On the contrary, [12] introduced an explicit, simple and easy to implement, purely geometric methodology where edge nodes constructed by the zeros of Lobatto polynomials and interior nodes were generated by barycentric coordinate lines inside the triangle and averaging the coordinates of vertices of some particular internal triangles. An alternative to this approach is presented in [65] by generalizing the recursive explicit formula to arbitrarily shaped domains. In [195], a simple explicit node construction method for arbitrary order approximations is introduced based on the philosophy of replacing the task of creating of a nodal distribution with closely related task of building a coordinate-warping transform for the triangle which is a familiar problem encountered in the curvilinear finite elements. Fig. 2.1 shows the different nodal distributions for the approximation order of 5.

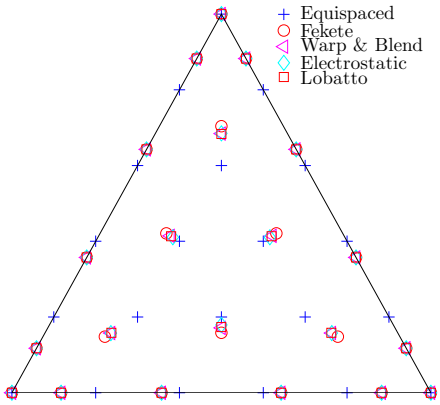


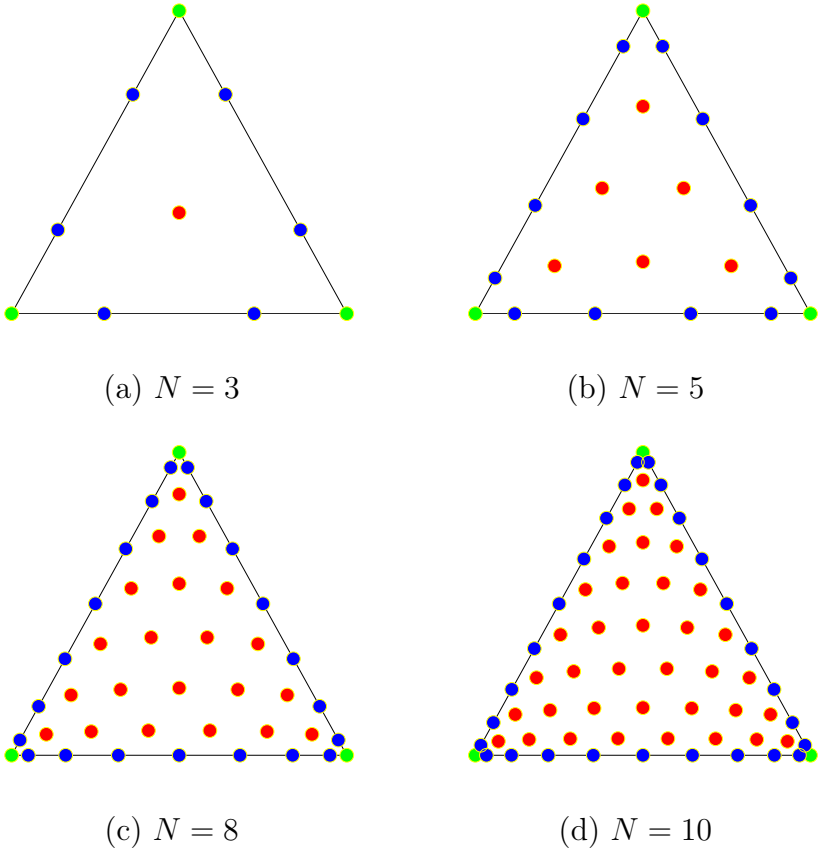
Figure 2.1: Different node distributions for triangular expansion of order 5.

The dimension of N order polynomial space, $\mathcal{P}_N$ for an triangle,

$$\dim \mathcal{P}_N = N_p = \binom{N+2}{2} = \frac{(N+1)(N+2)}{2} \tag{2.1}$$

which is the minimum dimension of the polynomial space to be complete. N_p is the number of elements in N-order Pascal triangle generally referred as the triangular truncation. There are $N+1$ nodes in each face, $N-1$ interior face nodes and 3 vertex nodes shared with neighbor faces leading totally to $3N$ face

nodes. Each face node coincides with the 1D GLL points which ensures the conformity of the global nodal distribution. Fig. 2.2 illustrates Warp & Blend [195] nodes for varying order of approximation on an equilateral triangle.



(a) $N = 3$ (b) $N = 5$

(c) $N = 8$ (d) $N = 10$

Figure 2.2: Warp & Blend nodes [195] on the equilateral triangle for different appropriation orders.

2.1.1.2 Tetrahedral Nodal Distribution

To prevent the oscillations and to increase the interpolation quality, nodes should be distributed through the tetrahedron such that the magnitude of the i^{th} interpolation function reaches its maximum value of 1 at the i^{th} node, and takes the values changing between 0 and 1. In other words, sum of the absolute value of the interpolation functions is bounded by N_p, number of points. The nodes satisfying this requirement are referred as the Fekete points as explained before. It is important to mention that Fekete points are uniquely defined, because changing the polynomial basis only multiples the determinant of the Vandermonde

matrix by a constant. Fekete sets are available for the 1D interval (GLL points), for the triangle [13], for the rectangle and for the hexahedron (tensor product of GLL points), but not for the tetrahedron. However, there are optimal nodal approximation points for tetrahedron.

Chen and Babuska [25] introduced a optimal distribution based on the maximizing the determinant of the Vandermonde matrix and minimizing L_2 norm of magnitude of approximating functions. They use two main approximations in the computations i.e. each face has the triangular optimal nodal distributions and interior nodes hold the geometric symmetry of the tetrahedron. Hesthaven and Teng [79] extended electrostatic node distribution relying the equilibrium position of charges distributed [81] to tetrahedral elements. Explicitly computed Lobatto distribution for triangle [12] is also generalized to the tetrahedral domains in [115]. Warp & Blend nodes of Warburton [195] is explicitly computed for triangle and tetrahedron in the same study. Also, Gassner et.al. [65] is introduced a simple nodal construction based on the recursively locating nodes using warping strategy [195] for polymorphic elements on hybrid grids.

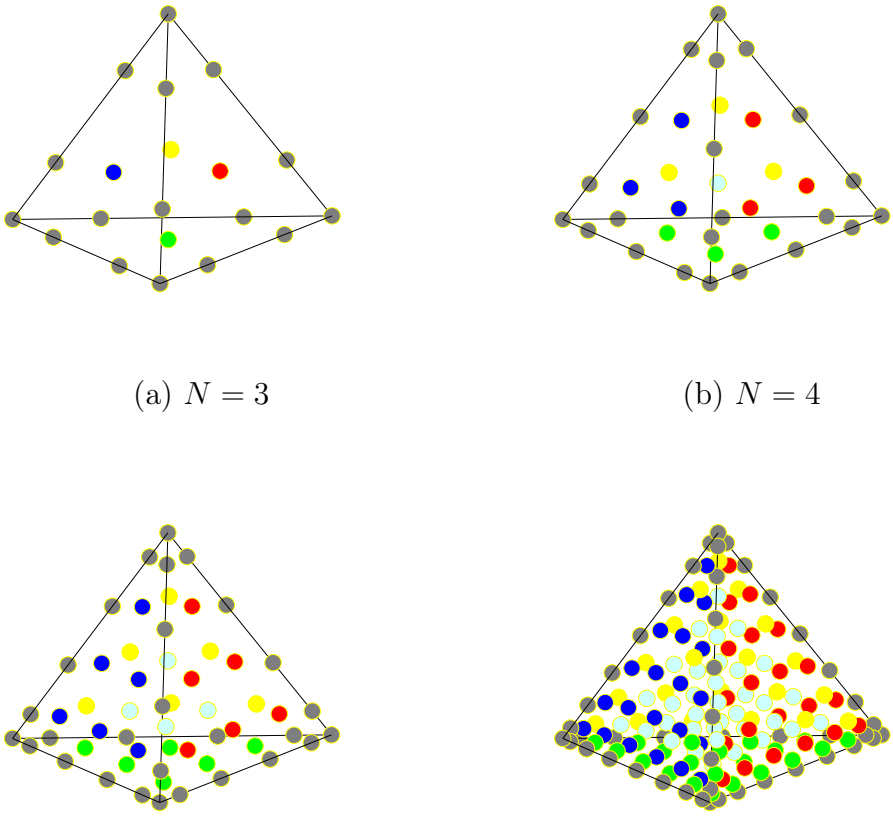
For a tetrahedron, required dimension of the N order polynomial space, $\mathcal{P}_N$ to be complete is given with

$$\dim \mathcal{P}_N = N_p = \binom{N+3}{3} = \frac{(N+1)(N+2)(N+3)}{6} \tag{2.2}$$

which can be recovered by N_p nodal points. Tetrahedral node distribution has $(N+1)(N+2)/2$ nodes on each face with $3(N+1)$ edge nodes including vertex nodes. This allows to construct complete N order, unique, one and two dimensional polynomial space in each edge and face, respectively. Also, subtracting the all face nodes from the total nodes, one gets

$$N_{p,I} = \binom{N-4}{3} = \frac{(N-1)(N-2)(N-3)}{6} \tag{2.3}$$

interior nodes for $N > 3$. Note that, number of interior nodes is equal to number of nodes in $N-4$ polynomial space. For $N \leq 3$, there is no volume nodes in the tetrahedron indicating number of face nodes are sufficient to construct N order, complete polynomial space. Warp & Blend nodes [195] are shown in the Fig. 2.3 for equilateral tetrahedron and different approximation orders.



(a) $N = 3$ (b) $N = 4$

(c) $N = 5$ (d) $N = 8$

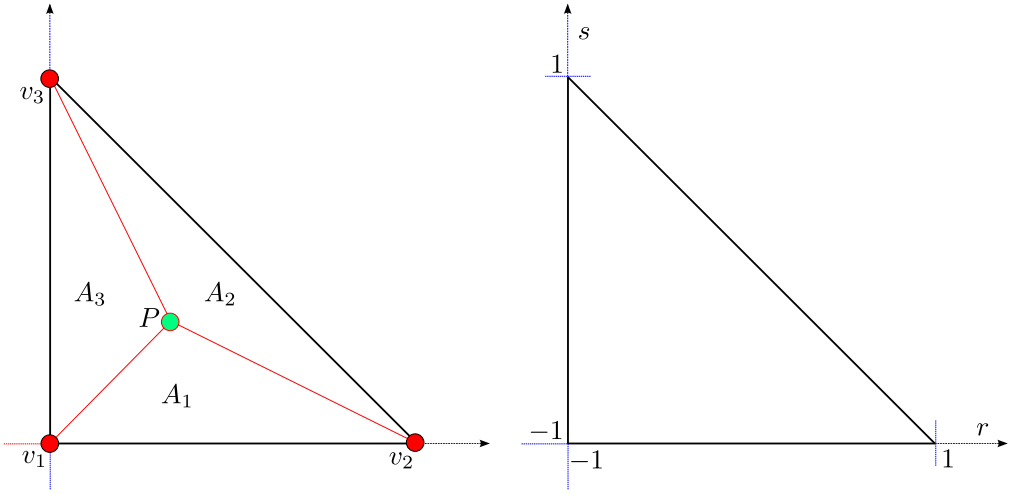
Figure 2.3: Warp & Blend nodes [195] node distributions for the tetrahedral expansion.

2.1.2 Local Coordinate Systems

For d-simplexes , Koornwinder reference or standard element is used with barycentric and collapsed coordinate systems to drive local operators, polynomial spaces and mapping between physical elements.

Because of the rotational symmetry property, barycentric coordinate system has been extensively used in unstructured domains. Maintaining rotational symmetry in a triangular element requires one dependent coordinate unlike the quadrilateral elements. This makes tensor product construction of expansions very difficult or impossible. However, barycentric coordinate system is useful in defining

rotationally symmetric, non-tensorial expansions [92].



(a) Barycentric Coordinate (b) Koornwinder Triangle

Figure 2.4: Barycentric coordinate frame and Koornwinder reference triangle.

Barycentric coordinate system is illustrated in Fig. 2.4(a) for the standard triangle. Any point, P in the triangle is uniquely defined by three coordinates, λ_1, λ_2 and λ_3, which are ratio of the areas A_1, A_2 and A_3, to the total area, $A = A_1 + A_2 + A_3$ and given by

$$\lambda_1 = \frac{A_1}{A}, \quad \lambda_2 = \frac{A_2}{A}, \quad \lambda_3 = \frac{A_3}{A} \tag{2.4}$$

Therefore, barycentric coordinates, λ_1, λ_2 and λ_3 take unit values at vertices and system satisfies $\lambda_1 + \lambda_2 + \lambda_3 = 1$.

As previously stated, polynomial space for structural elements may be formed by a tensor product of one-dimensional expansions based on the Cartesian coordinate system bounded by constant limits. Implicit assumption of the tensor extensions relies the fact that coordinates in two dimensional region has constant limits. The Koornwinder triangle shown in Fig. 2.4(b) has the following region,

$$\mathcal{T}^2 = \{(r,s)| \quad -1 \leq (r,s) \quad r+s \leq 0\} \tag{2.5}$$

To develop a tensorial or generalized tensorial type basis within a triangular region, we need to develop a coordinate system where the local coordinates have constant limits. A suitable coordinate system to describe a triangular region

between constant independent limits is collapsed coordinates or Duffy transform having the relation,

$$a = \frac{2(1+r)}{1-s} - 1, \quad b = s \tag{2.6}$$

With this transformation triangular region is expressed as

$$\mathcal{T}^2 = \{(a,b)| \quad -1 \leq a, b \leq 1\} \tag{2.7}$$

Definition of the triangular region with this coordinates system is identical to definition of quadrilateral region so that transformation can be interpreted as a mapping from quadrilateral region to triangular region. For this reason, coordinate system with (a, b) is referred as the collapsed coordinate system [167]. It is worthwhile to mention about the singularity of transformation at $r = -1, s = 1$ but r is bounded at that location similar to natural singularity in cylindrical and spherical coordinate systems [92]. Mapping of bi-unit rectangle to triangle and collapsing vertex C and D into single vertex C are shown in Fig. 2.5.

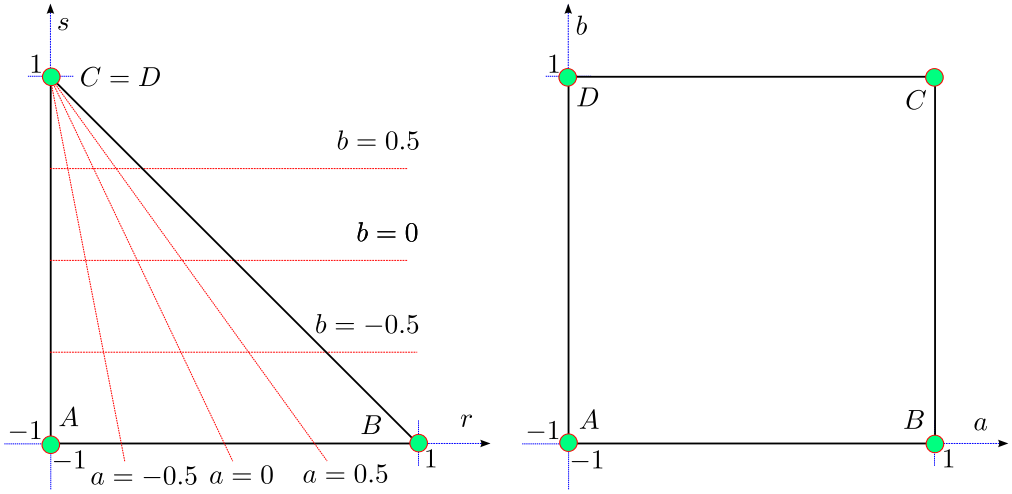


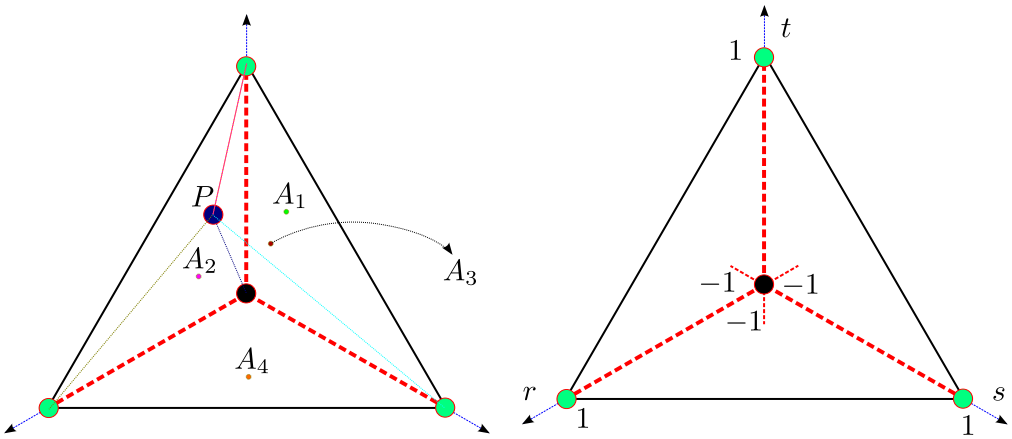
Figure 2.5: Mapping of bi-unit rectangle to triangle.

The same procedure for the triangular domain can be generalized to the tetrahedron. Any point in the tetrahedron can be defined by four, rotational symmetric coordinates, λ_1, λ_2,λ_3 and λ_4 which are the ratios of the areas A_1, A_2, A_3 and A_4 to total area $A = A_1 + A_2 + A_3 + A_4$ as shown in the Fig. 2.6(a). Obviously, $\lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1$ holds true.

Standard tetrahedral element is defined by,

$$\mathcal{T}^3 = \{(r,s,t)| \quad -1 \leq (r,s,t) \quad r+s+t \leq -1\} \tag{2.8}$$

and shown in Fig. 2.6(b).



(a) Barycentric Coordinate (b) Reference tetrahedron

Figure 2.6: Barycentric coordinate system and reference element for tetrahedron.

Tetrahedral element does not have constant limits to construct the tensor like polynomial space. Transformation between a bi-unit hexahedron to tetrahedron can be accomplished by successive rectangle to triangle transformations similar to triangle. Fig. 2.7 shows how to obtain a collapsed coordinate transformation for a tetrahedral element. No that, the same procedure can be used for pyramid or prismatic elements as shown in the sub-steps.

Collapsed coordinate transformation leads to the following relation,

$$a = \frac{2(1+r)}{-s-t} - 1, \quad b = \frac{2(1+s)}{1-t} - 1 \quad c = t \tag{2.9}$$

Tetrahedral region can be expressed with the constant limits as follow,

$$\mathcal{T}^3 = \{(a,b,c)| \quad -1 \leq (a,b,c) \leq 1\} \tag{2.10}$$

2.1.2.1 Orthogonal Expansions

Constructing polynomial expansions is critical to develop computationally efficient, high-order numerical schemes. Starting point for the multi-dimensional

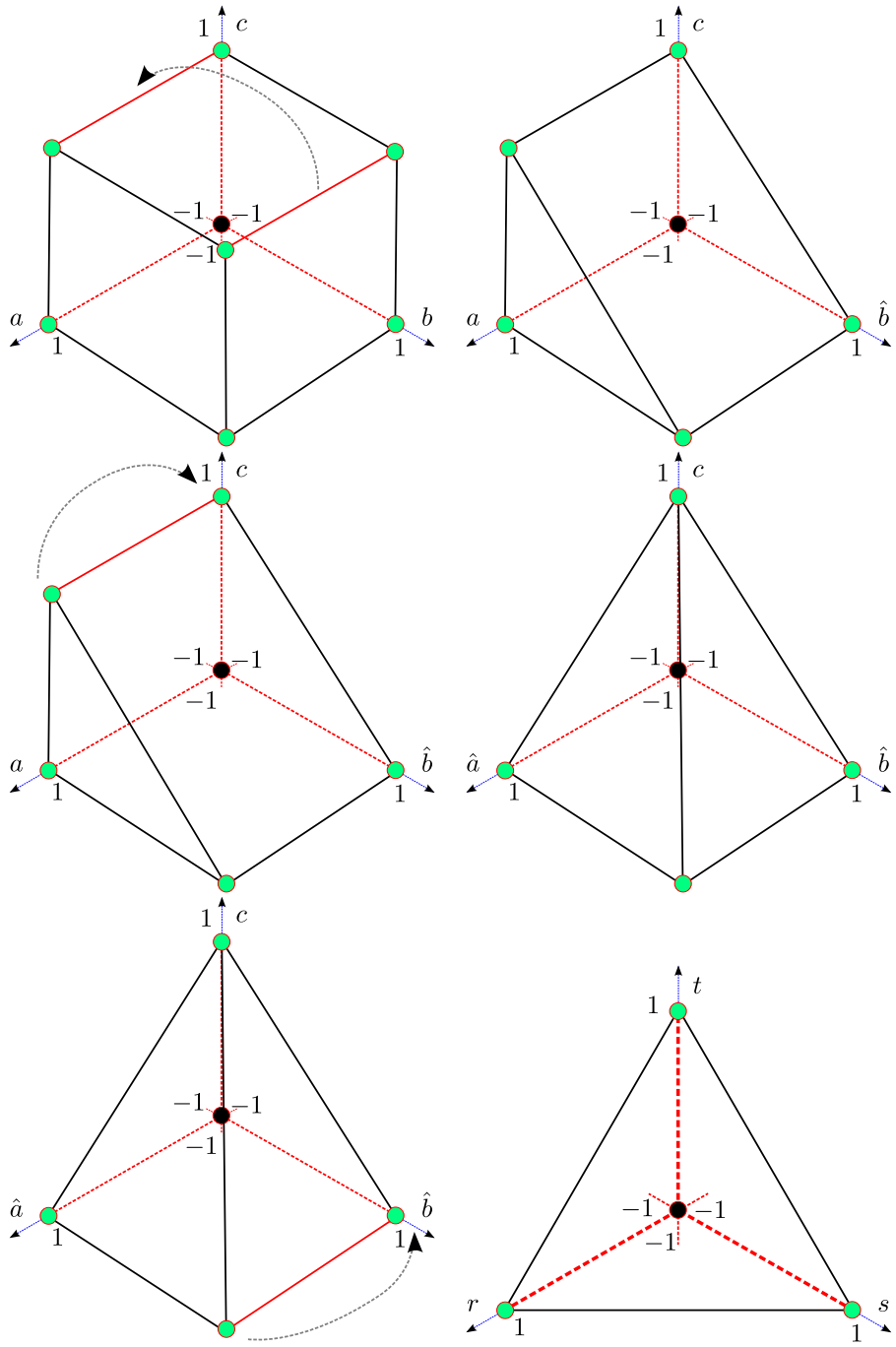


Figure 2.7: Collapsed coordinate transformation steps from bi-unit hexahedron to tetrahedron.

polynomial expansions is to ensure the orthogonality in the Legendre inner product norm (L_2 norm) over each elemental domain [92]. Different modal expansion spaces are introduced for the d simplex using the barycentric coordinate systems in non-symmetric [87] or symmetric [207] on the reference elements. Using ro-

tationally invariant barycentric coordinate system can destroy the numerical efficiency associated with the tensor product polynomials.

In this study, a generalized tensor product orthogonal basis is used in the computations. These polynomial spaces are introduced by Proriol [143], Koornwinder [101] and Dubiner [55] named as PKD polynomials. Sherwin [162] also presented a unified approach for hybrid elements based on the Dubiner's [55] orthogonal polynomials.

Using the collapsed coordinate transformation, tensor like (warped or generalized tensor product) polynomial space can be presented involving multiplication of one, two and three dimensional tensors. Unlike the structural elements, higher dimensional space can be constructed from the same one dimensional polynomials, a triangular expansion uses more general product in the form of

$$\Psi_{i,j}(a,b) = \psi_i(a)\psi_{i,j}(b) \tag{2.11}$$

where principal functions,$\psi_i, \psi_{i,j}$ are the Jacobi polynomials, $\mathcal{P}^{\alpha,\beta}$ of order i, j given with

$$\psi_i = \mathcal{P}_i^{0,0}(a) \quad \psi_{i,j} = \mathcal{P}_j^{2i+1,0}(b)(\frac{1-b}{2})^i \tag{2.12}$$

The special case, $\mathcal{P}_i^{0,0}$ represents the well known Legendre polynomials and polynomial, $\mathcal{P}_i^{\alpha,\beta}(x)$ is orthogonal with respect to weight functions, $w(x) = (1-x)^\alpha(1+x)^\beta$ which is the direct consequence of being solution of singular Sturm-Liouville eigenvalue problem [194]. It is convenient to normalize the basis space which can be achieved by normalizing the each polynomial function,

$$\Psi_{i,j}(r,s) = \left(\frac{\mathcal{P}_i^{0,0}(a)}{\sqrt{\frac{2}{2i+1}}}\right)\left(\frac{\mathcal{P}_j^{2i+1,0}(b)(\frac{1-b}{2})^i}{\sqrt{\frac{2}{2(i+j)+1}}}\right) \tag{2.13}$$

rearranging the terms leads,

$$\Psi_{i,j}(r,s) = c_{i,j}\mathcal{P}_i^{0,0}(a)\mathcal{P}_j^{2i+1,0}(b)(\frac{1-b}{2})^i \tag{2.14}$$

where $c_{i,j} = \sqrt{\frac{(2i+1)(i+j+1)}{2}}$ is the normalization parameter. Finally, writing the equation in the standard coordinate frame instead of collapsed coordinate, we arrive

$$\Psi_{i,j} = \sqrt{\frac{(2i+1)(i+j+1)}{2}}\mathcal{P}_i^{0,0}(\frac{2r+s+1}{1-s})(\frac{1-s}{2})^i\mathcal{P}_j^{2i+1,0}(s) \tag{2.15}$$

Complete polynomial space of maximum order N on the reference element, $\mathcal{P}_N(\mathcal{T}^2)$, can be defined using the PKD polynomials, $\Psi_{i,j}$ with $i, j \geq 0$, $i+j \leq N$. Without any ambiguity, single indexed notation, Ψ_k is also used for the polynomials such that k is any arbitrary bijection, $k = k(i, j)$.

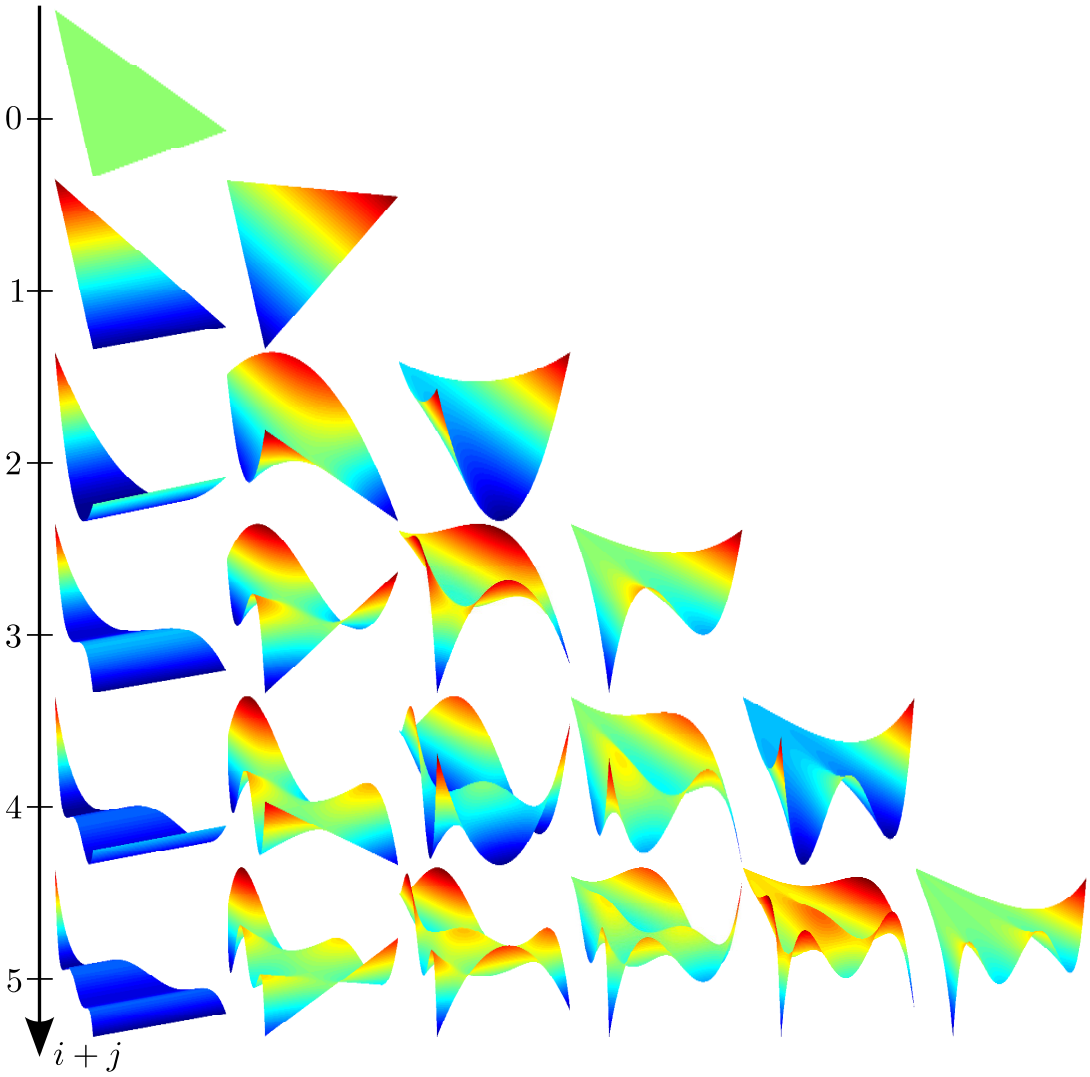


Figure 2.8: Orthonormal basis functions on standard triangle, $\mathcal{T}^2$ up to $N = 5$. Each row corresponds to all basis functions of the same order, $i + j \leq N$.

Fig. 2.8 shows all modal PKD polynomials for $N = 5$. Each row in the figure corresponds basis functions of the same order where hierarchical nature of the expansion space can be seen such that all polynomial functions of $N = 1$ is embedded into the $N = 2$ expansion space and so on.

Using the same approach with triangular expansion, orthonormal basis functions

of tetrahedron can be obtained. In this case, general tensor product involves three, one dimensional polynomials of orders i, j, k such that,

$$\Psi_{i,j,k}(r,s,t) = \psi_i(a)\psi_{i,j}(b)\psi_{i,j,k}(c) \tag{2.16}$$

where ψ_i and $\psi_{i,j}$ are defined similar to triangular expansion,

$$\psi_i = \frac{\mathcal{P}_i^{0,0}(a)}{\sqrt{\frac{2}{2i+1}}}, \quad \psi_{i,j} = \frac{\mathcal{P}_j^{2i+1,0}(b)(\frac{1-b}{2})^i}{\sqrt{\frac{2}{2(i+j)+1}}} \tag{2.17}$$

while $\psi_{i,j,k}$ can be written as,

$$\psi_{i,j,k} = \frac{\mathcal{P}_k^{2(i+j)+2,0}(c)(\frac{1-c}{2})^{i+j}}{\sqrt{\frac{2}{2(i+j+k)+3}}} \tag{2.18}$$

Generalized orthonormal expansion become,

$$\Psi_{i,j,k}(r,s,t) = c\mathcal{P}_i^{0,0}(a)\mathcal{P}_j^{2i+1,0}(b)(\frac{1-b}{2})^i\mathcal{P}_k^{2(i+j)+2,0}(c)(\frac{1-c}{2})^{i+j} \tag{2.19}$$

where $c = c(i,j,k)$ is the normalization parameter. Complete polynomial space of order N on the reference tetrahedron, $\mathcal{P}_N(\mathcal{T}^3)$, is constructed with the PKD polynomials, $\Psi_{i,j,k}$ with $i,j,k \geq 0$, $i+j+k \leq N$. Similar to triangular elements, single indexed notation, Ψ_m is given with arbitrary bijection, $m = m(i,j,k)$. The required one dimensional Jacobi polynomials can be easily computed with three-term recursion formula which completes the construction of modal basis functions on d-simplex.

2.1.3 Nodal Basis Functions and Vandermonde Matrix

Nodal basis functions do not have any closed form definition for general approximation orders through the arbitrary set of points in a simplex. It is necessary to express them in terms of another polynomials which have closed form definition. Modal polynomial space having explicit definition is given in the previous section. Construction of nodal Lagrange polynomials having the below Kronecker delta property will be presented following [196, 80]

$$L_i(\mathbf{r}_j) = \delta_{ij}, \quad \forall i,j = 1, \cdots, N_p \tag{2.20}$$

where $\mathbf{r}$ is the local coordinates with $\mathbf{r} = (r, s)$ for triangle, $\mathbf{r} = (r, s, t)$ for tetrahedron and N_p is the cardinality of the orthonormal basis space which is equal to number of nodes in the standard element as discussed in Sec. 2.1.1.

Let first consider the interpolation of a function, f in a simplex with a known function, Ψ which spans the same space, through N_p distinct points,

$$f(\mathbf{r}) = \sum_{j=1}^{N_p} \Psi_j(\mathbf{r}) c_j \tag{2.21}$$

Where j represents the each mode of Ψ and c_j is the corresponding coefficients associated with the expansion. These modal coefficients exhibit the relative importance of each mode in the interpolation. Because both functions spans the same polynomial space, any type of projection will recover the exact expansion coefficients, c_j. To obtain expansion coefficients, collocation projection at the nodal set, $\mathbf{r}_i$ will give

$$f(\mathbf{r}_i) = \sum_{j=1}^{N_p} \Psi_j(\mathbf{r}_i) c_j, \quad \forall i = 1, \cdots, N_p \tag{2.22}$$

or in matrix form,

$$f = Vc \tag{2.23}$$

where $V(i, j) = \Psi_j(\mathbf{r}_i)$. If Ψ denotes a monomial basis then V is the Vandermonde matrix and it is known as generalized Vandermonde matrix for general basis functions. Under the assumption of existence and uniqueness of the Lagrange interpolation polynomial, Eq. 2.22 can be written as,

$$\sum_{j=1}^{N_p} f(\mathbf{r}_i) L_j(\mathbf{r}_i) = \sum_{j=1}^{N_p} \Psi_j(\mathbf{r}_i) c_j \tag{2.24}$$

which establish the link between modal and nodal basis space,

$$L_i(\mathbf{r}) = \sum_{j=1}^{N_p} V_{i,j}^{-1} \Psi_j(\mathbf{r}) \tag{2.25}$$

where Lagrange polynomial, $L = \left[L_1(\mathbf{r}), \cdots, L_{N_p}(\mathbf{r})\right]$ can be computed once the solution of linear system given in Eq. 2.25 is found. Fig. 2.9 shows complete Lagrange basis constructed using the Warp & Blend nodes and PKD polynomials.

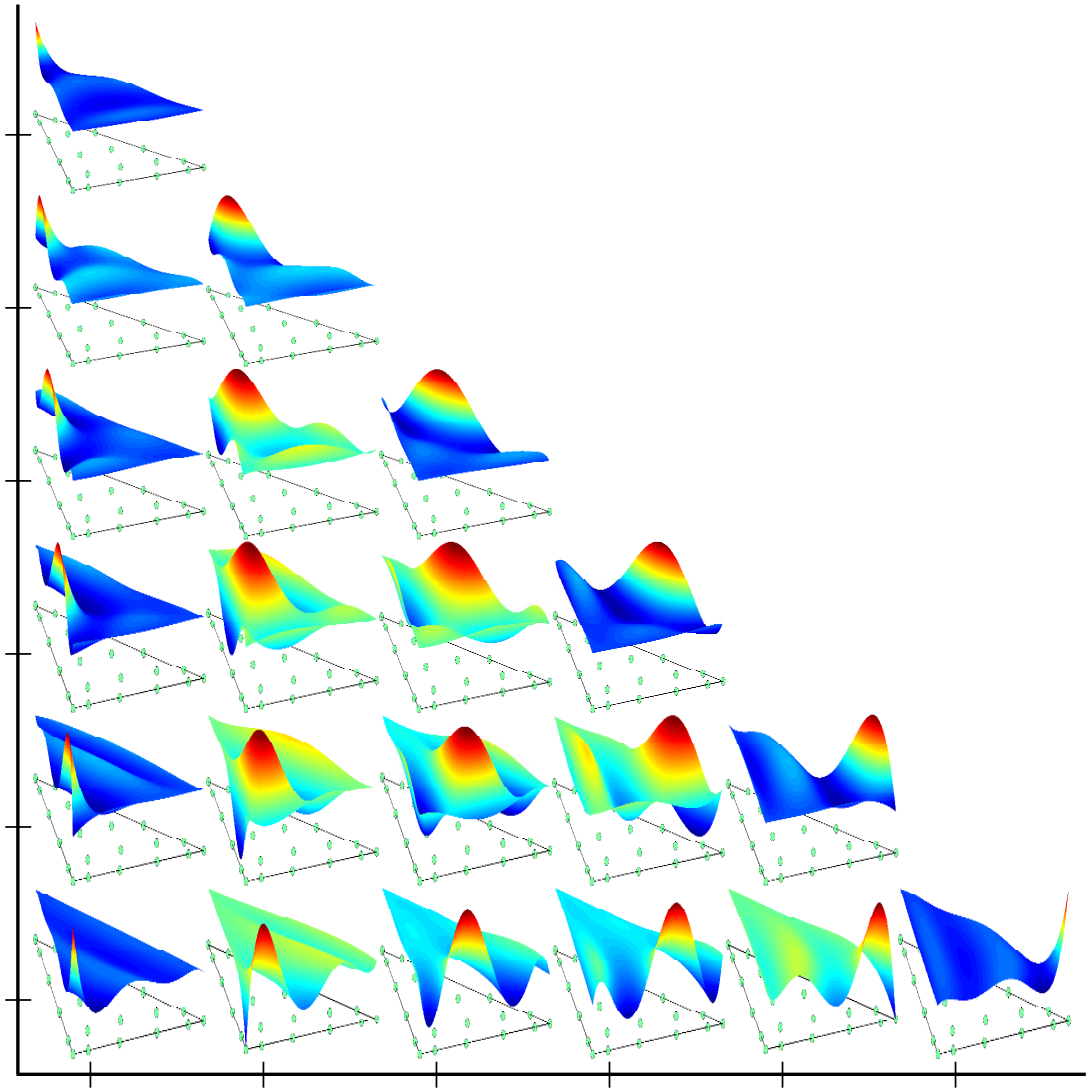


Figure 2.9: Lagrange interpolating polynomials constructed with Warp & Blend nodes and orthonormal PKD polynomials on $\mathcal{T}^2$ for $N = 5$.

Difficulty of solving the linear system is related with the conditioning of the generalized Vandermonde matrix to ensure the interpolation is well-behaved. Obviously, conditioning of the matrix is only dependent to the orthonormal basis forming the modal space and collocation points from its definition. To select the most appropriate combination, interpolation quality of the Vandermonde matrix should be measured, which is the subject of next section.

2.1.4 Interpolation Quality

Considering the same problem of interpolating a function $f(\mathbf{r})$ in the standard elements given in Fig. 2.4 and Fig. 2.6 for triangle and tetrahedron through a set of points, $\Theta = \{\mathbf{r}_1, \cdots, \mathbf{r}_{Np}\}$. Let assume another function in the same polynomial space that satisfies,

$$\begin{aligned} g(\mathbf{r}_i) &= f(r_i), \quad \forall i \quad 1 \leq i \leq N_p \\ g(\mathbf{r}) &= \mathcal{I}_N f(\mathbf{r}), \quad g, f \in \mathcal{P}_N \end{aligned} \tag{2.26}$$

where $\mathcal{I}_N$ is the interpolation operator. Lebesgue constant, Λ_N is a measure of the quality of $\mathcal{I}_N$ that shows how well $\mathcal{I}_N$ approximates f. Let consider a function f^* which spans the same space with f i.e. $f^*, f \in \mathcal{P}_N$ and best represents f in usual maximum norm, $\|.\|_\infty$ [25]. $f^* \neq \mathcal{I}_N f$ but $f^* = \mathcal{I}_N f^*$ and maximum norm can be written as,

$$\begin{aligned} \|f - \mathcal{I}_N f\|_\infty &= \|f - f^* + \mathcal{I}_N f - \mathcal{I}_N f^*\|_\infty \\ &\leq \|f - f^*\|_\infty + \|\mathcal{I}_N\|_\infty \|f - f^*\|_\infty \\ &\leq (1 + \|\mathcal{I}_N\|_\infty)\|f - f^*\|_\infty \end{aligned} \tag{2.27}$$

Using the Lagrange polynomials, interpolation of the function become

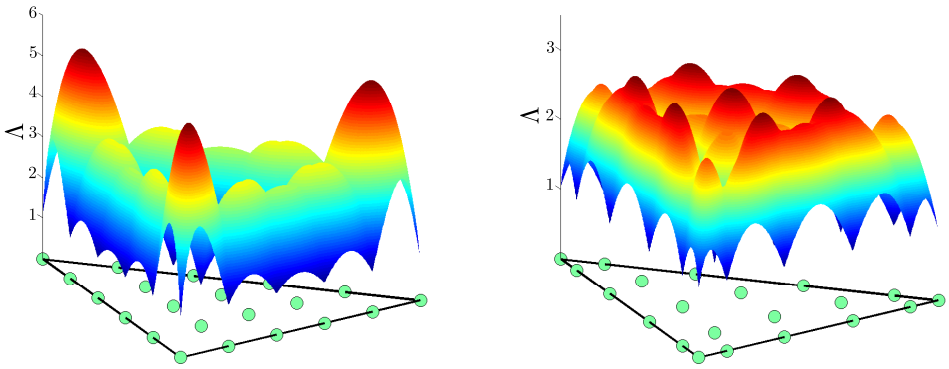
$$\mathcal{I}_N f = \sum_{1 \leq i \leq N_p} f(\mathbf{r}_i) L_i(\mathbf{r}) \tag{2.28}$$

And using Kronecker delta property of Lagrange polynomials

$$\Lambda_N = \|\mathcal{I}_N\|_\infty = \max_{\|f\|_\infty = 1} \|\mathcal{I}_N f\|_\infty = \max_{\mathcal{T}^2} \sum_{i=1}^{N_p} |L_i(\mathbf{r})| \tag{2.29}$$

Evaluating the sum of absolute values of Lagrange polynomials through the standard region gives the Lebesgue function and maximum norm of this function leads the Lebesgue constant. It is worthwhile to mention that, Lebesgue function is constructed with cardinal functions so that it is only dependent to the nodal distribution and independent from the polynomial basis space. Lebesgue functions on triangular domain are shown in Fig. 2.10 for equispaced and Warp & Blend nodes. Equispaced nodes introduces large values near the domain boundaries showing poor interpolation quality due to well-known Runge phenomena. On the other hand, sophisticated nodes are clustered at edges and

vertices which prevent the osculations and improve the interpolation quality. Note that, Lebesgue function takes unit values on nodes due to its definition which is the indication of exact interpolation.



(a) Equispaced nodes (b) Warp & Blend nodes

Figure 2.10: Lebesgue function over the standard triangle for $N = 5$.

Table 2.1 represents the comparison of the interpolation properties of different node distributions. Tabulated Lebesgue constants are computed very accurately with symbolic operations. As seen from the table, interpolation quality of nodes are very close the each other at low order approximations. Increasing the order Fekete points become most attractive one, but Warp & Blend nodes [195] have smaller values at practically applicable orders, $N \leq 10$, have closed form definition and remains a good alternative.

Table 2.1: Comparison of Lebesgue constant for different order of approximations and node distributions on triangle.

N	Uniform	Fekete	Lobatto	Warp&Blend	Electrostatic
3	2.2698	2.1125	2.1125	2.1125	2.1125
4	3.47481	2.72928	2.66195	2.66208	2.58801
5	5.45164	3.61081	3.13670	3.12115	3.19559
6	8.7476	4.1711	3.87430	3.7019	4.07490
9	40.9222	6.8016	7.39200	5.7352	6.88370
12	221.2266	9.6775	17.7789	9.3565	12.6325

Conditioning of the Vandermonde matrix plays a very critical role on building

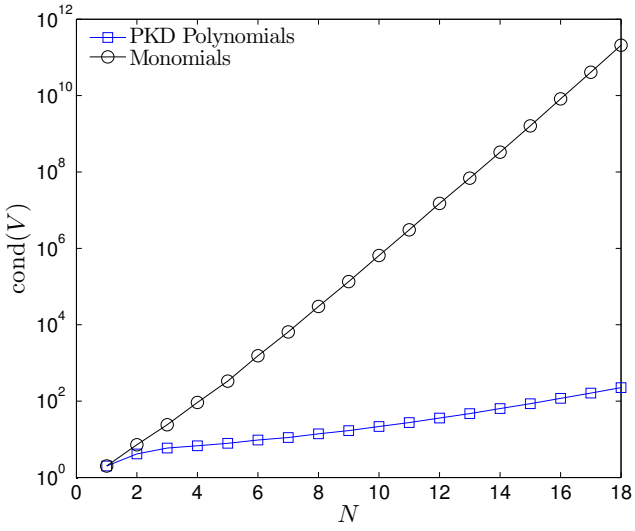


Figure 2.11: Condtion number of Vandermonde matrix constructed with PKD polynomials and monomials on Warp & Blend nodes.

the nodal scheme because inverse of this matrix is required to construct Lagrange polynomials. Furthermore, efficient computation of all volume and surface inner products of discontinuous Galerkin scheme relies on the good conditioning of Vandermonde matrix. Unlike the Lebesgue function, condition number is dependent to polynomial basis and node distribution so that it is not only related with node quality but also construction of the orthogonal polynomial basis. To show the PKD polynomials form a good basis space, let first check conditioning of Vandermonde matrices constructed also with the simple monomial polynomials,

$$\Psi_{i,j}(r,s) = r^i s^j, \quad \forall i,j, \quad i+j \leq N \tag{2.30}$$

Change of condition number of Vandermonde matrices constructed with Warp & Blend nodes is shown in Fig. 2.11. It is clear that monomial basis polynomials do not constitute orthogonal basis. Increasing the order, high order space will be already well presented by the low order space.

Table 2.2 represents the condition number of Vandermonde matrix at varying orders of approximation and different nodal sets. Warp & Blend nodes have excellent properties at practical orders of approximations.

The number of node distributions for tetrahedron is smaller than those for the triangle. In Table 2.3, condition numbers for Vandermonde matrix for the L_2

Table 2.2: Comparison of condition number of Vandermonde matrix constructed with PKD polynomials for different order of approximations and node distributions on triangle.

N	Uniform	Fekete	Lobatto	Warp&Blend	Electrostatic
3	5.8283	5.9028	5.9028	4.07804	5.9028
6	14.6583	9.7990	9.8423	11.4460	9.8294
9	59.9490	18.1216	18.0994	24.8176	18.8432
12	344.977	22.4680	43.3978	36.1323	54.0048
15	21,944.3821	29.4572	130.2558	85.6921	186.5085

nodes of Chen and Babuska [25], electrostatic nodes of Hesthaven and Teng [79], Lobatto grid of Luo and Pozrikidis [115], Warp & Blend nodes of Warburton [195] and equispaced node distribution are compared. Similar to Lebesgue constant, condition numbers for Warp & Blend nodes are competitive with the other node distributions.

Table 2.3: Comparison of condition number of Vandermonde matrices constructed with PKD polynomials for different order of approximations and node distributions on tetrahedron.

N	Uniform	L_2	Lobatto	Warp&Blend	Electrostatic
3	10.248	10.516	10.505	10.505	10.505
6	32.997	26.171	26.536	26.310	26.549
9	162.216	89.170	99.379	88.324	98.839
12	1064.722	-	568.921	421.013	-
15	7560.901	-	3847.701	2346.218	-

2.2 Discontinuous Galerkin Operators

In this section, basic notation required to develop DG scheme will be presented first. Then, global to local mapping, definition of basic inner products will be introduced. Efficient flux function evaluation will be discussed for both conformal and non-conformal discretizations. Scheme will be developed for the tetrahedron unless otherwise stated, result can be obtained for the triangular elements by inspection.

2.2.1 Notation

We assume that d-dimensional domain, $\Omega \subset R^d$ is well approximated by the computational domain, Ω_h having boundaries $\partial\Omega_h$ that can be Dirichlet type, $\partial\Omega_D$ or Neumann type, $\partial\Omega_N$. Ω_h is partitioned into non-overlapping, possibly nonconforming triangular or tetrahedral elements, $\Omega_h = \sum_{k=1}^{K} D_k$. Two element domains, D_k^- and D_k^+ of the triangulation Ω_h has a common face if $\partial D_k^- \cap \partial D_k^+ \neq \emptyset$, where ∂D_k denotes the element interface as illustrated in Fig. 2.12. Also, $\partial D_k = \sum_{f=1}^{N_f} \partial D_k^f$ with N_f is the total number of connections for an element which is equal to number of faces for conformal discretizations. It is obvious that superscripts $-$ and $+$ are meaningful only considering the face pairs to specify the right and left sides. $\mathbf{n}^- = -\mathbf{n}^+ = (n_1^-, \cdot, n_d^-)$ be the unit outward normal vector to ∂D_k.

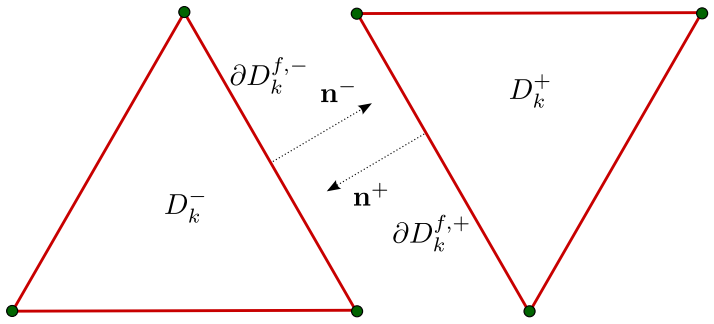


Figure 2.12: Basic geometrical notation used in the rest of the thesis.

ϕ_k^- and ϕ_k^+ denote the traces of a scalar function, ϕ when evaluated at ∂D_k^- and ∂D_k^+, respectively. According to this definition average and jump operators for a scalar can be defined as,

$$\{\varphi\} = \frac{\varphi^- + \varphi^+}{2}, \quad [\![\varphi]\!] = \varphi^- - \varphi^+ \tag{2.31}$$

In the DG method, inter-element continuity constraint is relaxed which enables the solution discontinuous through the element boundaries. Physical interpretation of the operators are illustrated in the Fig. 2.13 for a scaler valued function on 1D sample grid. If ϕ is a vector-valued function, the above operators act component wise on the function. On a Dirichlet or Neumann boundaries, normal vectors points outward to Ω_h, and average and jump operators are equal to

trace of function φ if otherwise stated explicitly.

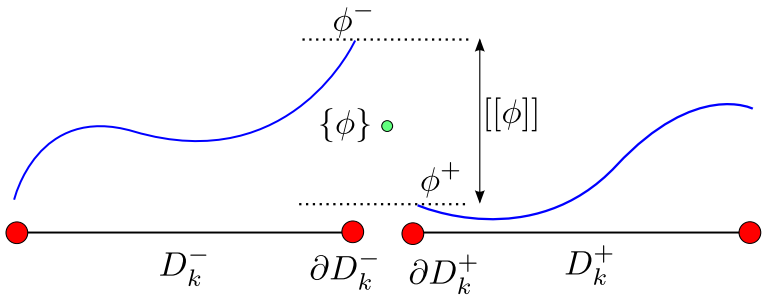


Figure 2.13: Illustration of the average and jump operators.

The discontinuous approximate spaces are defined as,

$$\mathcal{V}_N = \{v \in L_2(\Omega) | v|_{D_k} \in \mathcal{P}_N(D_k), \quad \forall D_k \in \Omega_h\} \tag{2.32}$$

and its vector version, $\mathcal{V}_N^d$. $\mathcal{P}_N(D_k)$ is the space of discontinuous piecewise polynomial functions of degree $N \geq 1$ on each elemental domain, D_k. Lagrange polynomials are constructed as the basis for this polynomial space using Warp & Blend nodes [195] and orthonormal PKD [143, 101, 55] polynomials. Also, let $(\cdot,\cdot)_{D_k}$ represent the inner product computed over the volume of the element k. Similarly, let $(\cdot,\cdot)_{\partial D_k}$ denote the inner product taken along the element boundaries.

2.2.2 Global to Local Mapping

Let consider a physical tetrahedral element abbreviated with D with straight sided faces and the reference element, $\mathcal{T}^3$ illustrated previously. Physical coordinates are shown with $\mathbf{x} = (x, y, z) \in D$ and vertices of the element are given as $V_1(\mathbf{x}) - V_4(\mathbf{x})$ with the faces of $F_1 - F_4$ oriented counterclockwise. Faces are named in a way that F_1 has the V_1 as the base vertex and correspondingly for the others. Similarly, vertices and faces are indexed for the $\mathcal{T}^3$ with $v_1 - v_4$ and $f_1 - f_4$, respectively as shown in the Fig. 2.14.

There exist a global to local map function, $\Psi : D_k \to \mathcal{T}^3$. Any coordinate on the straight sided triangle, $\mathbf{x} \in D_k$ can be represented by the convex combinations

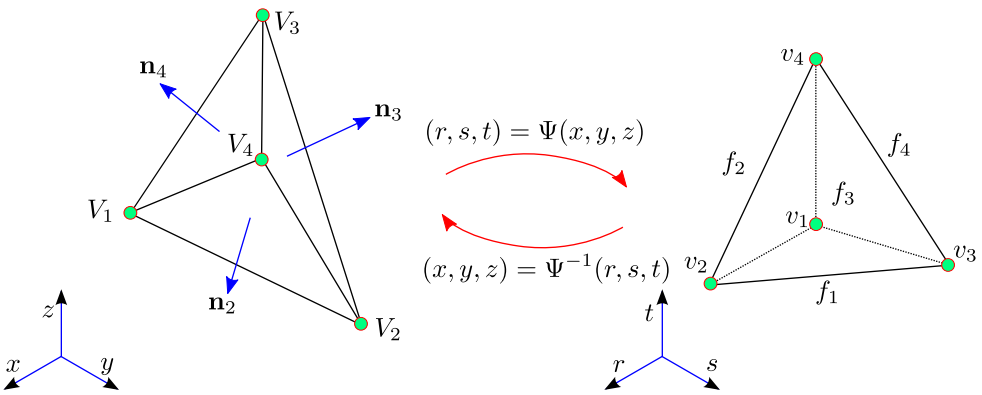


Figure 2.14: Mapping between physical straight sided tetrahedron, D and the standard element, $\mathcal{T}^3$.

of the barycentric coordinates,

$$\mathbf{x} = \lambda_3 V_1 + \lambda_4 V_2 + \lambda_2 V_3 + \lambda_1 V_4 \tag{2.33}$$

The same discussion holds for the standard element given with local coordinates of the vertices,

$$\mathbf{r} = \lambda_3 v_1 + \lambda_4 v_2 + \lambda_2 v_3 + \lambda_1 v_4 \tag{2.34}$$

where vertex coordinates of the Koornwinder tetrahedron are given with $v_1 = (-1,-1,-1)$, $v_2 = (1,-1,-1)$, $v_3 = (-1,1,-1)$ and $v_4 = (-1,-1,1)$, leading

$$\lambda_1 = \frac{t+1}{2}, \quad \lambda_2 = -\frac{s+1}{2}, \quad \lambda_3 = -\frac{r+s+t+1}{2}, \quad \lambda_4 = \frac{r+1}{2} \tag{2.35}$$

Combining with the Eq. 2.33, direct mapping, Ψ is obtained as,

$$\mathbf{x} = -\frac{r+s+t+1}{2} V_1 + \frac{r+1}{2} V_2 - \frac{s+1}{2} V_3 + \frac{t+1}{2} V_4 \tag{2.36}$$

Transformation is linear in local coordinates because any two straight-sided tetrahedrons are connected with an affine mapping. In other words, transformation has a constant Jacobian. The metric of transformation can be directly obtained through,

$$\frac{\partial \mathbf{x}}{\partial \mathbf{r}} \frac{\partial \mathbf{r}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial r} & \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} & \frac{\partial r}{\partial z} \\ \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} & \frac{\partial s}{\partial z} \\ \frac{\partial t}{\partial x} & \frac{\partial t}{\partial y} & \frac{\partial t}{\partial z} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.37}$$

where $\partial \mathbf{x}/\partial \mathbf{r}$ can be recovered from Eq. 2.36 leading,

$$\frac{\partial \mathbf{x}}{\partial r} = \frac{V_2 - V_1}{2}, \quad \frac{\partial \mathbf{x}}{\partial s} = \frac{V_3 - V_1}{2}, \quad \frac{\partial \mathbf{x}}{\partial t} = \frac{V_4 - V_1}{2} \tag{2.38}$$

Then, Jacobian of the transformation, which is the ratio of volumes of the physical element to the standard element, can be written as,

$$J = \det\left(\begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial s} & \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial s} & \frac{\partial y}{\partial t} \\ \frac{\partial z}{\partial r} & \frac{\partial z}{\partial s} & \frac{\partial z}{\partial t} \end{bmatrix}\right) = \frac{\partial x}{\partial r}\left(\frac{\partial y}{\partial s}\frac{\partial z}{\partial t} - \frac{\partial z}{\partial s}\frac{\partial y}{\partial t}\right) - \frac{\partial y}{\partial r}\left(\frac{\partial x}{\partial s}\frac{\partial z}{\partial t} - \frac{\partial z}{\partial s}\frac{\partial x}{\partial t}\right) + \frac{\partial z}{\partial r}\left(\frac{\partial x}{\partial s}\frac{\partial y}{\partial t} - \frac{\partial y}{\partial s}\frac{\partial x}{\partial t}\right) \tag{2.39}$$

and required metric entities follow directly,

$$\begin{aligned} \frac{\partial r}{\partial x} &= \frac{\frac{\partial y}{\partial s}\frac{\partial z}{\partial t} - \frac{\partial z}{\partial s}\frac{\partial y}{\partial t}}{J}, & \frac{\partial r}{\partial y} &= -\frac{\frac{\partial x}{\partial s}\frac{\partial z}{\partial t} - \frac{\partial z}{\partial s}\frac{\partial x}{\partial t}}{J}, & \frac{\partial r}{\partial z} &= \frac{\frac{\partial x}{\partial s}\frac{\partial y}{\partial t} - \frac{\partial y}{\partial s}\frac{\partial x}{\partial t}}{J} \\ \frac{\partial s}{\partial x} &= -\frac{\frac{\partial y}{\partial r}\frac{\partial z}{\partial t} - \frac{\partial z}{\partial r}\frac{\partial y}{\partial t}}{J}, & \frac{\partial s}{\partial y} &= \frac{\frac{\partial x}{\partial r}\frac{\partial z}{\partial t} - \frac{\partial z}{\partial r}\frac{\partial x}{\partial t}}{J}, & \frac{\partial s}{\partial z} &= -\frac{\frac{\partial x}{\partial r}\frac{\partial y}{\partial t} - \frac{\partial y}{\partial r}\frac{\partial x}{\partial t}}{J} \\ \frac{\partial t}{\partial x} &= \frac{\frac{\partial y}{\partial r}\frac{\partial z}{\partial s} - \frac{\partial z}{\partial r}\frac{\partial y}{\partial s}}{J}, & \frac{\partial t}{\partial y} &= -\frac{\frac{\partial x}{\partial r}\frac{\partial z}{\partial s} - \frac{\partial z}{\partial r}\frac{\partial x}{\partial s}}{J}, & \frac{\partial t}{\partial z} &= \frac{\frac{\partial x}{\partial r}\frac{\partial y}{\partial s} - \frac{\partial y}{\partial r}\frac{\partial x}{\partial s}}{J} \end{aligned} \tag{2.40}$$

In the DG scheme, geometric factors related with surfaces of the element i.e. normals and surface Jacobian are also required to compute flux functions efficiently on the standard element. Normals can be recovered from the properties of the mapping function, Φ. Observing the Fig. 2.14 reveals that unit outward normal vectors through the each face hold the following,

$$\mathbf{n}^1 = -\frac{\nabla t}{\|\nabla t\|_2}, \mathbf{n}^2 = -\frac{\nabla s}{\|\nabla s\|_2}, \mathbf{n}^3 = \frac{\nabla r + \nabla s + \nabla t}{\|\nabla r + \nabla s + \nabla t\|_2}, \mathbf{n}^4 = -\frac{\nabla r}{\|\nabla r\|_2} \tag{2.41}$$

where $\|\cdot\|_2$ denotes the standard L_2 norm or Euclidean length of the vector. Then, collecting each face normal vector for the physical element, $\mathbf{n} = (n_x, n_y, n_z) = [\mathbf{n}^1, \mathbf{n}^2, \mathbf{n}^3, \mathbf{n}^4]$, following relation will be recovered for the surface Jacobian,

$$J_{\partial D_k} = \|\mathbf{n}\|_2 J_{D_k} \tag{2.42}$$

which completes the derivation of mapping for tetrahedral element. Generalization to triangular element is straight forward and will not be covered here.

2.2.3 Evaluation of Inner Products

Let consider two scalar functions, f and g which are well approximated by $f_h, g_h \in \mathcal{V}_N$. Local Lagrange representations of the functions will be,

$$f_h(\mathbf{x}(\mathbf{r})) = \sum_{i=1}^{Np} f_h(\mathbf{x}_i) L_i(\mathbf{r}), \quad \text{and} \quad g_h(\mathbf{x}(\mathbf{r})) = \sum_{j=1}^{Np} g_h(\mathbf{x}_j) L_j(\mathbf{r}) \tag{2.43}$$

Similar to other numerical techniques DG formulation needs to evaluate volume inner products in the form of,

$$(f_h, g_h)_{D_k} = \int_{D_k} f_h g_h dx = \int_{\mathcal{T}^3} f_h g_h J_{D_k} dr \tag{2.44}$$

For a straight sided tetrahedron, where the Jacobian is constant,

$$(f_h, g_h)_{D_k} = J_{D_k} \int_{\mathcal{T}^3} \sum_{i=1}^{Np} \sum_{j=1}^{Np} f_h(\mathbf{x}_i) L_i(\mathbf{r}) L_j(\mathbf{r}) g_h(\mathbf{x}_j) dr \tag{2.45}$$

which can be written as,

$$(f_h, g_h)_{D_k} = J_{D_k} \left(f_i M_{i,j} g_j\right) \quad \forall i, j = 1, \ldots, N_p \tag{2.46}$$

where nodal function values are $f_i = f_h(\mathbf{x}_i)$ and $g_j = g_h(\mathbf{x}_j)$. M is the completely local mass matrix given with

$$M_{i,j} = \int_{\mathcal{T}^3} L_i L_j dr \quad \forall i, j = 1, \ldots, N_p \tag{2.47}$$

Using the definition of Lagrange polynomials, i.e.

$$L_i(\mathbf{r}) = \sum_{n=1}^{N_p} (V^T)^{-1}_{i,n} \Psi_n(\mathbf{r}) \quad \forall i = 1, \ldots, N_p \tag{2.48}$$

and inserting into Eq. 2.47,

$$\begin{aligned} M_{i,j} &= \int_{\mathcal{T}^3} \sum_{n=1}^{N_p} (V^T)^{-1}_{i,n} \Psi_n(\mathbf{r}) \sum_{m=1}^{N_p} (V^T)^{-1}_{j,m} \Psi_m(\mathbf{r}) dr \\ &= \sum_{n=1}^{N_p} \sum_{m=1}^{N_p} (V^T)^{-1}_{i,n} (V^T)^{-1}_{j,m} \int_{\mathcal{T}^3} \Psi_n(\mathbf{r}) \Psi_m(\mathbf{r}) dr \end{aligned} \tag{2.49}$$

Orthogonality of the basis space results with the identity matrix,

$$\int_{\mathcal{T}^3} \Psi_n(\mathbf{r}) \Psi_m(\mathbf{r}) dr = \delta_{n,m} \tag{2.50}$$

and combining with the previous equation,

$$M_{i,j} = \sum_{n=1}^{N_p} \sum_{m=1}^{N_p} (V^T)^{-1}_{i,n} \delta_{n,m} (V^T)^{-1}_{j,m} \tag{2.51}$$

In matrix notation, expression takes the following final form,

$$M = (V^T)^{-1} V^{-1} = \left(V V^T\right)^{-1} \tag{2.52}$$

This mass matrix is locally defined on the reference element and small in the size of $N_p \times N_p$. Elemental inner products can be computed efficiently using the local matrix-vector multiplication,

$$(f_h, g_h)_{D_k} = J_{D_k} f_h^T M g_h \tag{2.53}$$

It is very important to mention that the above integration is exact only for the integrand belonging to $\mathcal{P}_{2N}(\mathcal{T}^3)$. More precisely if $f, g \in \mathcal{P}_N$ and Jacobian is constant, which is encountered in the Galerkin type discretization of the linear problems, integration approach according to Eq. 2.53 is exact . For the quadratic and cubic nonlinearities as in incompressible and compressible Navier-Stokes equations, integrands are belonging to $\mathcal{P}_{3N}$ and $\mathcal{P}_{4N}$, respectively. In these circumstances, integrations will be sub-optimal leading polynomial aliasing and more accurate cubature integration is required for accuracy and/or stability of the numerical scheme. More sophisticated integration rules will be covered in the Sec. 2.2.5.

Stiffness matrices can be computed in a similar procedure. Using the same scalar function $f, g \in \mathcal{P}_N$,

$$\begin{aligned}
(\nabla f_h, \nabla g_h)_{D_k} &= \int_{D_k} \nabla f_h \cdot \nabla g_h dx = \int_{D_k} \left(\frac{\partial f_h}{\partial x}\frac{\partial g_h}{\partial x} + \frac{\partial f_h}{\partial y}\frac{\partial g_h}{\partial y} + \frac{\partial f_h}{\partial z}\frac{\partial g_h}{\partial z} \right) d\mathbf{x} \\
&= \sum_{i=1}^{N_p} \sum_{j=1}^{N_p} f_i g_j \int_{D_k} \left(\frac{\partial L_i}{\partial x}\frac{\partial L_j}{\partial x} + \frac{\partial L_i}{\partial y}\frac{\partial L_j}{\partial y} + \frac{\partial L_i}{\partial z}\frac{\partial L_j}{\partial z} \right) d\mathbf{x}
\end{aligned} \tag{2.54}$$

Derivatives in physical space can be written in the local coordinate frame,

$$\begin{aligned}
D_x = \frac{\partial L}{\partial x} &= \frac{\partial r}{\partial x}\frac{\partial L}{\partial r} + \frac{\partial s}{\partial x}\frac{\partial L}{\partial s} + \frac{\partial t}{\partial x}\frac{\partial L}{\partial t} = \frac{\partial r}{\partial x} D_r + \frac{\partial s}{\partial x} D_s + \frac{\partial t}{\partial x} D_t \\
D_y = \frac{\partial L}{\partial y} &= \frac{\partial r}{\partial y}\frac{\partial L}{\partial r} + \frac{\partial s}{\partial y}\frac{\partial L}{\partial s} + \frac{\partial t}{\partial y}\frac{\partial L}{\partial t} = \frac{\partial r}{\partial y} D_r + \frac{\partial s}{\partial y} D_s + \frac{\partial t}{\partial y} D_t \\
D_z = \frac{\partial L}{\partial z} &= \frac{\partial r}{\partial z}\frac{\partial L}{\partial r} + \frac{\partial s}{\partial z}\frac{\partial L}{\partial s} + \frac{\partial t}{\partial z}\frac{\partial L}{\partial t} = \frac{\partial r}{\partial z} D_r + \frac{\partial s}{\partial z} D_s + \frac{\partial t}{\partial z} D_t
\end{aligned} \tag{2.55}$$

All geometric factors are computed in the previous section and given in the Eq. 2.40. Local derivatives, D_r, D_s and D_t defined below are required to close the relation.

$$D_{r,(i,j)} = \frac{\partial L_j(\mathbf{r}_i)}{\partial r}, \quad D_{s,(i,j)} = \frac{\partial L_j(\mathbf{r}_i)}{\partial s}, \quad D_{t,(i,j)} = \frac{\partial L_j(\mathbf{r}_i)}{\partial t} \tag{2.56}$$

The connection of the modal and nodal expansion spaces is established before through the Vandermonde matrix i.e. $V^T L_i = \Psi_j$ and $L_j = V^{-1}\Psi_i$. Then, the following relations follow directly,

$$D_{r,(i,j)} = \sum_{n=1}^{N_p} V_{r,(i.n)} V^{-1}_{n,j}, \quad D_{s,(i,j)} = \sum_{n=1}^{N_p} V_{s,(i.n)} V^{-1}_{n,j}, \quad D_{t,(i,j)} = \sum_{n=1}^{N_p} V_{t,(i.n)} V^{-1}_{n,j} \tag{2.57}$$

where V_r, V_s and V_t are the Vandermonde derivative matrix given with,

$$V_{r,(i.j)} = \left.\frac{\partial \Psi_j(\mathbf{r})}{\partial r}\right|_{\mathbf{r}_i} \tag{2.58}$$

It is more convenient to write the derivatives in matrix form,

$$D_r = V_r V^{-1}, \quad D_s = V_s V^{-1}, \quad D_t = V_t V^{-1} \tag{2.59}$$

The entries of the $V_{\mathbf{r}}$ can be obtained directly from the differentiation of PKD polynomials in collapsed coordinate frame where it is defined as,

$$\begin{aligned}
\frac{\partial \Psi_j}{\partial r} &= \frac{\partial a}{\partial r}\frac{\partial \Psi_j}{\partial a} + \frac{\partial b}{\partial r}\frac{\partial \Psi_j}{\partial b} + \frac{\partial c}{\partial r}\frac{\partial \Psi_j}{\partial c} \\
\frac{\partial \Psi_j}{\partial s} &= \frac{\partial a}{\partial s}\frac{\partial \Psi_j}{\partial a} + \frac{\partial b}{\partial s}\frac{\partial \Psi_j}{\partial b} + \frac{\partial c}{\partial r}\frac{\partial \Psi_j}{\partial c} \\
\frac{\partial \Psi_j}{\partial t} &= \frac{\partial a}{\partial t}\frac{\partial \Psi_j}{\partial a} + \frac{\partial b}{\partial t}\frac{\partial \Psi_j}{\partial b} + \frac{\partial c}{\partial r}\frac{\partial \Psi_j}{\partial c}
\end{aligned} \tag{2.60}$$

Geometric factors of the transformation can be recovered from the Eq. 2.9 with purely local operations. After defining all entries of the differential matrices D_x, D_y and D_z, stiffness matrix can be computed efficiently with simple matrix -vector multiplications. Different from the mass matrix, stiffness matrix include locally defined matrices, $D_{\mathbf{r}}$ and element-wise defined geometric factors, $\mathbf{r_x}$. However ,this situation does not introduce any complexity because $\mathbf{r_x}$ is also constant for the straight sided tetrahedral elements similar to transformation Jacobian. Finally, Eq. 2.54 takes the following form,

$$\begin{aligned}
S_{D_k} &= D_x^T D_x + D_y^T D_y + D_z^T D_z, \\
(\nabla f_h, \nabla g_h)_{D_k} &= J_{D_k} f_h^T S_{D_k} g_h,
\end{aligned} \tag{2.61}$$

Unlike the continuous finite element or spectral element methods, inter-element continuity constraint is relaxed in the DG methods. Communication between the elements are provided by the numerical fluxes defined on the element boundaries. Boundary contribution resulting from the weak form DG scheme results with the surface integral in the form of,

$$(L_i, \mathbf{n} \cdot \mathbf{F}_h)_{\partial D_k} = \int_{\partial D_k} L_i \mathbf{n} \cdot \mathbf{F}_h d\mathbf{x} \tag{2.62}$$

where $\mathbf{F}_h \in \mathcal{P}_N^d$ is polynomial trace of the d dimensional vector function composed of the numerical fluxes with the arguments of jump and/or average of the field variable at the boundary. Using the Lagrange interpolation of the function and splitting the integration into connection pairs, f

$$(L_i, \mathbf{n} \cdot \mathbf{F}_h)_{\partial D_k} = \sum_{f=1}^{N_f} \left(\sum_{j=1}^{N_p^{d-1}} \mathbf{n} \cdot \mathbf{F}_j \int_{\partial D_k^f} L_i L_j d\mathbf{x} \right) \tag{2.63}$$

where $d-1$ is the co-dimension one geometric entity, which is a surface when $d = 3$ and an edge for $d = 2$. We use the term face and surface for all dimensions although it corresponds an edge for triangle, without ambiguity. Also, N_p^{d-1} denotes the number of nodes in the face. respectively. Turning back to tetrahedron where $d = 3$, $N_f = 4$ and $N_p^2 = (N+1)(N+2)/2$, Eq. 2.63 recast into,

$$(L_i, \mathbf{n} \cdot \mathbf{F}_h)_{\partial D_k} = \sum_{f=1}^{4} \left(\sum_{j=1}^{N_p^2} \mathbf{n} \cdot \mathbf{F}_j \int_{\partial D_k^f} L_i L_j d\mathbf{x} \right) \tag{2.64}$$

It is obvious that the last term in the form of mass matrix which can be considered to be full size of $N_p \times N_p^2$. However, the Lagrange polynomials, L_i are exactly zero at points except the ones lying on the face f. Which reduces the size of the face-mass matrix to $N_p^2 \times N_p^2$ on each face. Then, defining two dimensional Vandermonde matrix on each face of the element,

$$\int_{\partial D_k^f} L_i L_j d\mathbf{x} = J_{\partial D_k^f} \int_{\partial T^f} L_i L_j d\mathbf{r} = J_{\partial D_k^f} \left(V_f V_f^T \right)^{-1} \tag{2.65}$$

It can be observed that only nodal information of the face is required to evaluate surface integrals on that face. This is practically an important advantage of nodal scheme over the modal scheme where all information is needed to define a solution point wise.

2.2.4 Defining Interpolation Matrices

Interpolation of the functions between different nodal distributions, basis spaces, orders of the same space and grids are crucial to implement DG numerical schemes. For example, interpolating nodal solutions to equispaced nodes may be needed in post-processing. Decreasing the order of polynomial space is frequently encountered to stabilize non-linear equations via. limiting and filtering. Application of cubature rules also needs the interpolation of functions between high and low number of nodal points through the same polynomial orders. Also, p type multigrid solvers change the approximation orders between hierarchical levels. In this section, interpolation matrices will be constructed to handle these transforms efficiently.

Let first consider the function $f^p \in \mathcal{P}_N$ defined on the nodal set, $\Theta_1 = \{\mathbf{r}_1, \cdots, \mathbf{r}_{Np}\}$. The problem is to interpolate function to a nodal set, $\Theta_2 = \{\mathbf{r}_1, \cdots, \mathbf{r}_{N_c}\}$ given with $f^c \in \mathcal{P}_N$ with $N_p \neq N_c$ Interpolation between the nodal sets can be easily achieved using the interpolation properties of the Lagrange polynomials,

$$f_i^c = \sum_{k=1}^{Np} V_{i,k}^c \left(V_{k,j}^p\right)^{-1} f_j^p, \quad \forall i = 1, \cdots, N_c, \quad \forall j = 1, \cdots, N_p \tag{2.66}$$

where Vandermonde matrices are defined as $V_{i,j}^c = \Psi_j(\mathbf{r}_i),\ \mathbf{r}_i \in \Theta_2$ and $V_{i,j}^p = \Psi_j(\mathbf{r}_i), \mathbf{r}_i \in \Theta_1$. Then, interpolation matrix, $\mathcal{I}_p^c$ can be written as $\mathcal{I}_p^c = V^c V^{-1}$ and $f^c = \mathcal{I}_p^c f$. It is clear that interpolation operator is local and in the size of $N_c \times N_p$ which reduces the interpolation operators to simple matrix-vector multiplications in implementation.

The second problem is to interpolate the same function, $f^p \in \mathcal{P}_{N_1}$ defined on the nodal set, Θ_1 to the different order polynomials space, $f^c \in \mathcal{P}_{N_2}$ with $N_2 \leq N_1$. The cardinality of the expansions spaces are $N_c = (N_2+1)(N_2+2)(N_2+3)/6$ and $N_p = (N_1+1)(N_1+2)(N_1+3)/6$. Θ_1 is previously defined and $\Theta_2 = \{\mathbf{r}_1, \cdots \mathbf{r}_{N_c}\}$ is the nodal distribution for the $\mathcal{P}_{N_2}$. Let define two Vandermonde matrices,

$$\begin{aligned} V_{i,j}^{N_2} &= \Psi_j(\mathbf{r}_i), \quad \mathbf{r}_i \in \mathcal{P}_{N_2}, \quad \forall j = 1, \cdots, N_c \\ V_{i,j}^{N_1} &= \Psi_j(\mathbf{r}_i), \quad \mathbf{r}_i \in \mathcal{P}_{N_1}, \quad \forall j = 1, \cdots, N_p \end{aligned} \tag{2.67}$$

The previous discussion follows directly,

$$f_i^c = \sum_{k=1}^{Np} \left(V_{i,k}^{N_2}\right)^{-1} V_{j,k}^{N_1} f_j^p = \left(V^{N_1}\left(V^{N_2}\right)^{-1}\right)^T f^p = \mathcal{I}_{N_1}^{N_2} f^p, \tag{2.68}$$

Size of the $\mathcal{I}_{N_1}^{N_2}$ is $N_c \times N_p$ and local, independent from the physical element. Back transform can be obtained as $\mathcal{I}_{N_2}^{N_1} = \left(\mathcal{I}_{N_1}^{N_2}\right)^T = V^{N_1}\left(V^{N_2}\right)^{-1}$. These matrices play an crucial role in designing the matrix-free multigrid preconditioner discussed in Sec. 5.2.2.2.

Another important interpolation is related with the development of adaptive mesh refinement strategy and also accurate post-processing algorithms. In this interpolation problem, information between coarse and fine grids are transformed. Let consider a triangular element with regular uniform refinement to show the basic properties. Fig. 2.15 shows the regular refinement on the physical element, D_k by splitting edges leading 4 new elements, $D_k^1, D_k^2, D_k^3, D_k^4$ and three new vertices, V_1^n, V_2^n, V_3^n. The refinement can be represented in the reference element, $\mathcal{T}^2$ by the new elements $T_2^1, T_2^2, T_2^3, T_2^4$ and the new vertices v_1^n, v_2^n, v_3^n.

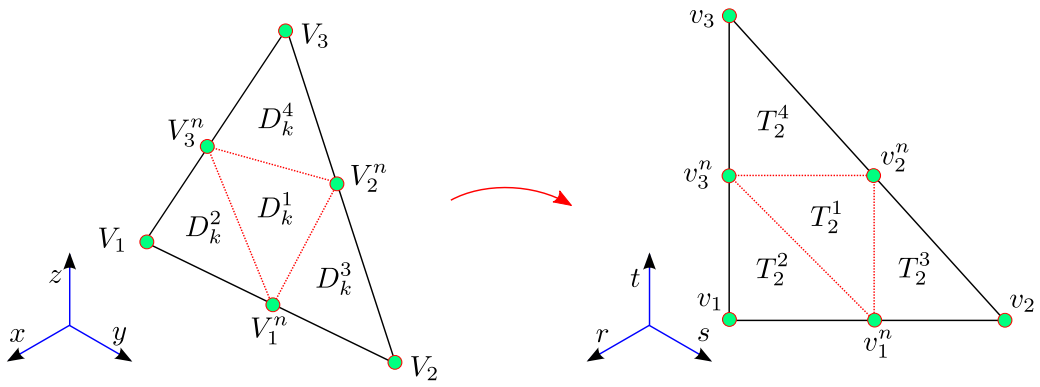


Figure 2.15: Sample uniform refinement on triangular element and corresponding structure on the reference element for the interpolation.

Any nodal points on the new elements can be mapped to local coordinate frame

as in the Fig. 2.15, using the coordinates of the new vertices,

$$\begin{aligned}
\mathbf{r}_{T_2^1} &= -\frac{r+s}{2}\mathbf{v}_1^n + \frac{r+1}{2}\mathbf{v}_2^n + \frac{s+1}{2}\mathbf{v}_3^n \\
\mathbf{r}_{T_2^2} &= -\frac{r+s}{2}\mathbf{v}_1 + \frac{r+1}{2}\mathbf{v}_1^n + \frac{s+1}{2}\mathbf{v}_3^n \\
\mathbf{r}_{T_2^3} &= -\frac{r+s}{2}\mathbf{v}_2 + \frac{r+1}{2}\mathbf{v}_2^n + \frac{s+1}{2}\mathbf{v}_1^n \\
\mathbf{r}_{T_2^4} &= -\frac{r+s}{2}\mathbf{v}_3 + \frac{r+1}{2}\mathbf{v}_3^n + \frac{s+1}{2}\mathbf{v}_2^n
\end{aligned} \tag{2.69}$$

Vandermonde matrices for each element can be defined using the created nodal sets on siblings such that,

$$V_{T_2^1,(i,j)} = \Psi_j(\mathbf{r}_{T_2^1,i}), \quad \forall i,j = 1,\cdots,N_p \tag{2.70}$$

other three Vandermonde matrices can be computed similarly. Finally, interpolation matrices from coarse to fine grid follows,

$$\mathcal{I}_{D_k}^{D_k^1} = V_{T_2^1}V^{-1}, \quad \mathcal{I}_{D_k}^{D_k^2} = V_{T_2^2}V^{-1}, \quad \mathcal{I}_{D_k}^{D_k^3} = V_{T_2^3}V^{-1}, \quad \mathcal{I}_{D_k}^{D_k^4} = V_{T_2^4}V^{-1} \tag{2.71}$$

where V is the standard Vandermonde matrix defined on the T_2. All interpolation matrices are $Np \times Np$ in size and local. For the implementation point of view , it is beneficial to combine them leading $4N_p \times N_p$ matrix. For the coarsening, where four sibling elements are connected to single element, interpolation operators will be the inverse of the matrices defined above i.e. $\mathcal{I}_{D_k^1}^{D_k} = VV_{T_2^1}^{-1}$ with the same notation for other matrices.

Generalization of this strategy to tetrahedron is straight forward and can be obtained by inspection. Regular refinement of the tetrahedral element results with eight siblings by connecting middle point of edges. This will give eight interpolation matrices having the same structure with triangular elements.

2.2.5 De-aliasing and Cubature Integration

Integration according to methodology presented in the previous section is exact only the integrands belonging to polynomial space, $\mathcal{P}_{2N}$, as explained before. If two functions, $f, g \in \mathcal{P}_N$ and transformation Jacobian constant, $(f,g)_{D_k}$ is exact and does not lead the polynomial aliasing. However, quadratic and cubic nonlinearities of the incompressible and compressible Navier-Stokes equations

require to integrate functions belonging to $\mathcal{P}_{3N}$ and $\mathcal{P}_{4N}$, respectively. This situation leads aliasing errors and even aliasing-driven instabilities especially in marginally- and under-resolved problems. One way to reduce those errors and instability is to choose sufficiently large number of nodes (and corresponding weights) to integrate the higher-order polynomial functions. Cubature integration for volume inner products and Gauss quadrature rules for surface integrals address this problem and will be covered in this section.

Implementation of the cubature integration can be achieved using the outputs of previous section. Let consider two functions f, g both are approximated by the $f_h, g_h \in \mathcal{P}_N$ defined on a nodal set, $\Theta_p = \{\mathbf{r}_1, \cdots, \mathbf{r}_{N_p}\}$. To perform the cubature integration, both functions need to be interpolated to the cubature nodes. $\Theta_c = \{\mathbf{r}_1, \cdots, \mathbf{r}_{N_c}\}$. Weighted discrete inner product the interpolated vectors will be computed using the cubature weights. Because, function f_h and corresponding interpolated function, f_c span the same polynomial space, interpolation matrix from standard node distribution to cubature nodes is as given in Eq. 2.66. Also, let $\mathcal{W} = \{w_1, \cdots, w_{N_c}\}$ represents the collection of the weights associated with the each cubature nodes. Discrete computation of the mass matrix takes the following form for straight-sided tetrahedral element.

$$(f_h, g_h)_{D_k} = J_{D_k} \sum_{i=1}^{N_c} f_{c,i} w_i g_{c,i} \tag{2.72}$$

Stiffness matrices on cubature nodes can be computed following the same approach. Because, only straight sided elements are considered here, metric identities, $\partial\mathbf{r}/\partial\mathbf{x}$ are also constant for the elements so only local derivative matrices, $D_{\mathbf{r}}$ need to be evaluated to compute $D^c_{\mathbf{x}}$. At cubature nodes,

$$\begin{aligned} D^c_r &= \sum_{n=1}^{Np} \frac{\partial \Psi_n(\mathbf{r}_i)}{\partial r} V^{-1}_{n,j}, \quad &= V^c_r V^{-1} \\ D^c_s &= \sum_{n=1}^{Np} \frac{\partial \Psi_n(\mathbf{r}_i)}{\partial s} V^{-1}_{n,j}, \quad &= V^c_s V^{-1} \\ D^c_t &= \sum_{n=1}^{Np} \frac{\partial \Psi_n(\mathbf{r}_i)}{\partial t} V^{-1}_{n,j}, \quad &= V^c_t V^{-1} \end{aligned} \tag{2.73}$$

$\forall i = 1, \cdots, N_c$ and $\forall j = 1, \cdots, N_p$. Then, differentiation on physical space can

be written as,

$$\begin{aligned} D_x^c &= \frac{\partial r}{\partial x} D_r^c + \frac{\partial s}{\partial x} D_s^c + \frac{\partial t}{\partial x} D_t^c \\ D_y^c &= \frac{\partial r}{\partial y} D_r^c + \frac{\partial s}{\partial y} D_s^c + \frac{\partial t}{\partial y} D_t^c \\ D_z^c &= \frac{\partial r}{\partial z} D_r^c + \frac{\partial s}{\partial z} D_s^c + \frac{\partial t}{\partial z} D_t^c \end{aligned} \tag{2.74}$$

Entries of the discrete stiffness matrix, $S = (\nabla f_h, \nabla g_h)_{D_k}$ are computed first evaluating the differentiation , $f^c = D_{\mathbf{x}}^c f_h$ and $g^c = D_{\mathbf{x}}^c g_h$, and then integrating the interpolated functions such that,

$$S_{D_k} = J_{D_k} \sum_{i=1}^{N_c} f_i^c w_i g_i^c \tag{2.75}$$

Cubature nodes on the faces of a triangle reduce to Gauss quadratures on the edges where surface integrals can be easily computed by following the same procedure.

As seen in the previous discussion, implementation of a cubature rule does not introduce any complexity and can be achieved locally similar to standard node operations by defining the suitable interpolation operators. Main questions are to decide when a cubature integration is needed, cubature order and hence the number of integration nodes to obtain the aliasing-free numerical solution. Because, unnecessary increase in the number of integration nodes obviously create computational burden and may destroy the efficiency of scheme. In fact, answer of the first question is clear that cubature integration for the nonlinearities are needed in under-resolved cases where aliasing adds high energy to modal coefficients leading instability.

Non-linearities may occur in varying orders. Only quadratic nonlinearities arising from the convective terms of the incompressible Navier-Stokes equitations are considered here. Galerkin type evaluation of the these non-linearities gives $\mathcal{P}_{3N}$ integrand. In this study, exact integration is used although many well-known solvers utilize under-integration i.e. integrating exactly the linear terms, $\mathcal{P}_{2N}$. Optimal cubature nodes for triangle and tetrahedron are given in [49, 48] up to some cubature orders. If the required $3N$ cubature rule exists, nodes and corresponding weights are extracted from the lookup table. If the required order is larger then tabulated orders, tensor product of Gauss-Lobatto-Legendre

distribution in r direction and Gauss-Radau-Jacobi distribution in s and t directions are utilized using collapsed coordinate frame [92]. Although, this approach results with unsymmetrical nodal distribution with condensing nodes on collapsed vertices, aliasing errors can be significantly reduced with enforcing conservation by keeping the mean mode untouched [96]. In this case, $(3N+3)/2$ cubature nodes are used in each directions to exactly integrate the quadratic non-linearities.

After setting the cubature integration rules for the d-dimensional simplexes, efficient flux evaluation through the non-conformal face pairs can be constructed. Let consider the non conformal discretization resulting from the one level local refinement as illustrated in Fig. 2.16, to simplify the computation.

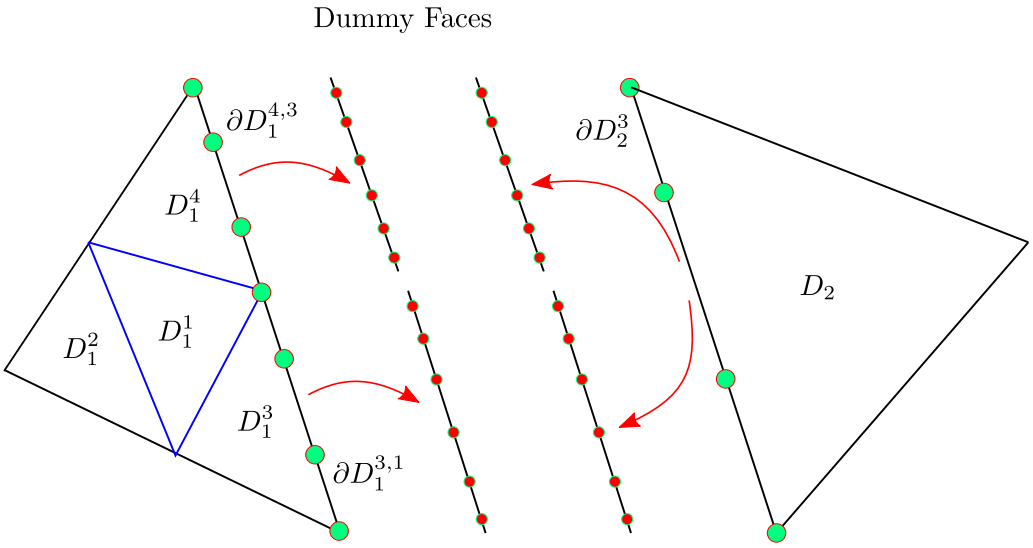


Figure 2.16: Evaluation of the flux function on non-conformal face pairs for one level local refinement and $N = 3$.

Let say D_1 and D_2 are initially connected through their second,∂D_1^2 and third, ∂D_2^3 faces, respectively. Then, D_1 is uniformly refined to the four sibling, D_1^1, D_1^2, D_1^3 and D_1^4. Due to the refinement strategy implemented, the siblings D_1^3, D_1^4 are connected to the ∂D_2^3 along the faces $\partial D_1^{3,1}$ and $\partial D_1^{4,3}$, and obviously ∂D_2^2 has a hanging node. Flux evaluation on this face needs special treatment to handle the nonconformity. As seen in the figure, a dummy face pair is created between the elements with different refinement levels. Dummy faces has the

sufficient Gauss quadrature points to integrate the rational flux functions, F_h. For the siblings, trace of field variables needs to be interpolated to the Gauss quadrature points using the following local matrices

$$\begin{aligned} \mathcal{I}_3 &= V_3 V^{-1}, \quad V_3 = \Psi_j(\mathbf{g}_i^3), \\ \mathcal{I}_1 &= V_1 V^{-1}, \quad V_1 = \Psi_j(\mathbf{g}_i^2), \end{aligned} \qquad \forall j = 1 \cdots N_p, \quad \forall i = 1 \cdots N_g \tag{2.76}$$

where N_g and $\mathbf{g}^f$ denote number of Gauss quadrature points and the corresponding local coordinates on the given face, f, respectively. For the ∂D_2^3, interpolation operators should be defined for the two dummy faces,

$$\begin{aligned} \mathcal{I}_{3,1} &= V_{3,1} V^{-1}, \quad V_{3,1} = \Psi_j(\mathbf{g}_i^{3,1}), \\ \mathcal{I}_{3,2} &= V_{3,2} V^{-1}, \quad V_{3,2} = \Psi_j(\mathbf{g}_i^{3,2}), \end{aligned} \qquad \forall j = 1 \cdots N_p, \quad \forall i = 1 \cdots N_g \tag{2.77}$$

where $\mathbf{g}_i^{3,1}$ and $\mathbf{g}_i^{3,2}$ can be computed easily using the vertices on the standard triangle and following the notation in Fig. 2.15.

$$\begin{aligned} \mathbf{g}^{3,1} &= 0.5\left(\left(1-\mathbf{g}^3\right) v_3 + \left(1+\mathbf{g}^3\right) v_3^n\right) \\ \mathbf{g}^{3,1} &= 0.5\left(\left(1-\mathbf{g}^3\right) v_3^n + \left(1+\mathbf{g}^3\right) v_1\right) \end{aligned} \tag{2.78}$$

with these operators, all required information is obtained for the non-conformal flux evaluation. It can be concluded that six local interpolation matrices should be constructed. This approach can be generalized to the tetrahedron and/or higher refinement levels. In the adaptive scheme, difference between the refinement levels of elements are set to two to reduce the algorithmic and coding complexity. In other words, if the refinement levels of two neighboring elements differs more than two, additional refinement is used for the element having the lower refinement level.

2.3 Massively Parallel Implementation

In DG discretizations, elements are only weakly connected to their first neighbors with numerical fluxes, as previously stated. This property render element-local computations and results with locality of memory access. In addition, the high order approximation space used in DG methods, increase the amount of computations per degree of freedom and hence, computational intensity. These

features are all well-suited for the parallelization on many-core/multi-thread architectures.

To show our massive parallel implementation, let consider a generic hyperbolic equation,

$$\frac{\partial Q}{\partial t} + \nabla \cdot \mathbf{F}(Q) = 0 \tag{2.79}$$

Multiplying the equation with smooth test functions, $\phi \in \mathcal{V}_N$ and integrating part parts, we obtain

$$\left(\phi, \frac{\partial Q_h}{\partial t}\right)_{D_k} = (\nabla\phi \cdot \mathbf{F}(Q_h))_{D_k} + (\phi, \mathbf{n} \cdot \mathbf{F}(Q_h))_{\partial D_k} \tag{2.80}$$

where Q is well approximated by $Q_h \in \mathcal{V}_N$. $\mathbf{F}(Q_h)$ is the flux function generally appriximated by monotone, consistent numerical flux which is a function of traces at element boundaries i.e. Q^- and Q^+ etc. The semi-discrete form can be written in terms of matrix form using previously defined operators,

$$\frac{dQ_h}{dt} = \frac{M_{\mathcal{T}}^{-1}}{J_{D_k}} D_{\mathbf{r}}^{c\,T} \cdot \left(\frac{\partial \mathbf{r}}{\partial \mathbf{x}} w_c J_{D_k} \mathbf{F}^c(Q_h)\right) - \frac{M_{\mathcal{T}}^{-1}}{J_{D_k}} M_{\partial\mathcal{T}} \cdot (J_{\partial D_k} \mathbf{n} \cdot \mathbf{F}^g(Q_h)) \tag{2.81}$$

with $\mathbf{F}^c(Q_h) = \mathcal{I}_c \times \mathbf{F}(Q_h)$ and $\mathbf{F}^g(Q_h) = \mathcal{I}_g(F_Q)$ are the volume and flux functions defined on volume and surface cubature points, respectively. There are three major computations required to solve the equation; volume integration, surface integration, and time-step update which are performed by Volume, Surface and Update kernels respectively as shown in the following generic form.

$$\underbrace{\frac{dQ}{dt}}_{\text{Update Kernel}} = \underbrace{\mathcal{V}(Q_h)}_{\text{Volume Kernel}} + \underbrace{\mathcal{S}(Q_h^-, Q_h^+)}_{\text{Surface Kernel}} \tag{2.82}$$

In our parallel implementation, a work group computes the integrals of one (or more) elements and a work item in the work group computes the contribution from each integration node in the kernels as illustrated in Fig. 3.6. For each element, contributions from volume and surface integrals are represented with multiplication of a local dense matrix defined on the reference element and a vector of volume terms and surface fluxes.

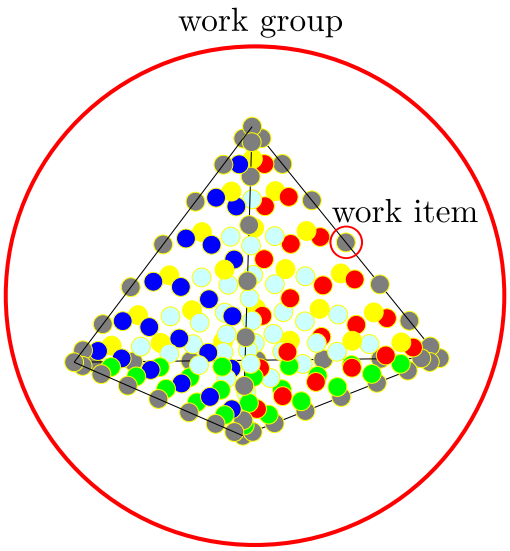


Figure 2.17: A work group and a work item for 3D kernels.

2.3.1 Volume Kernel

Volume integrals are computed in this kernel. $\mathcal{V}(Q_h)$ is in the size of N_p on each element D_k and can be written in the following generic form,

$$\mathcal{V}(Q) = P_r \times cF_1 + P_s \times cF_2 + P_t \times cF_3 \tag{2.83}$$

where P_r, P_s and the P_t are the projection operators defined on the reference element with the size of $N_p \times N_c$. Projection operators are pre multiplied with inverse mass matrix and cubature weights to accelerate computations, such that

$$\begin{aligned} P_r &= M_{\mathcal{T}}^{-1} \times D_r^{c\,T} \times \mathrm{diag}(w_c) \\ P_s &= M_{\mathcal{T}}^{-1} \times D_s^{c\,T} \times \mathrm{diag}(w_c) \\ P_t &= M_{\mathcal{T}}^{-1} \times D_t^{c\,T} \times \mathrm{diag}(w_c) \end{aligned} \tag{2.84}$$

cF_1, cF_2 and cF_3 denote the numerical volume vector defined on the cubature integration points of size N_c . These vectors for our generic equation read,

$$\begin{aligned} cF_1 &= \frac{\partial r_c}{\partial x} F_x^c(Q) + \frac{\partial r_c}{\partial y} F_y^c(Q) + \frac{\partial r_c}{\partial z} F_z^c(Q) \\ cF_2 &= \frac{\partial r_c}{\partial x} F_x^c(Q) + \frac{\partial r_c}{\partial y} F_y^c(Q) + \frac{\partial r_c}{\partial z} F_z^c(Q) \\ cF_3 &= \frac{\partial r_c}{\partial x} F_x^c(Q) + \frac{\partial r_c}{\partial y} F_y^c(Q) + \frac{\partial r_c}{\partial z} F_z^c(Q) \end{aligned} \tag{2.85}$$

First operation of kernel is to copy elemental field variables from global memory to shared memory. N_p work items are required for this operation. Then, stored shared memory variables are interpolated to the cubature integration points. This operation requires number of field variables matrix-vector multiplication all using the same interpolation matrix like $I_c \times Q_h$. Each work item multiples one row of I_c with the vectors to avoid memory conflicts. Resulting nodal values and geometric data are stored on the register memory for fast evaluation of future computations. N_c work items are assigned for this operation.

Second step in the kernel is calculation of volume flux terms, cF_1, cF_2 and cF_3, using the previously obtained values stored on register memory. Volume flux term is stored on the shared memory vector. N_c work items are used for this operation.

Finally, volume terms are interpolated to interpolation nodes. This operation involves three matrix-vector multiplication using three interpolation matrices and three vectors stored previously on shared memory. Similar to the interpolation to cubature points, each work item multiplies one vector of interpolation matrices with the flux vector to prevent the memory conflicts. N_p work item is used for this operation.

Number of required work items change within the kernel by accomplishing the tasks. To accommodate the number for work items required for the all computations, kernel request the $K_v \times \max(N_c, N_p)$ work items where K_v is the number of elements processed by single work group.

2.3.2 Surface Kernel

Surface contribution is computed in this kernel. $\mathcal{S}(Q_h^-, Q_h^+)$ is vector of length N_p for each field on element D_k. Surface integral can be written in the following generic form,

$$\mathcal{S}(Q_h^-, Q_h^+) = -L_{\mathcal{T}} \times \mathbf{F}_n^g(Q_h^-, Q_h^+) \tag{2.86}$$

where L is the lifting matrix [99] that interpolates the surface terms to interpolation nodes. The lifting operator is defined on the reference element and

pre multiplied with inverse mass matrix and surface cubature weights. In our hyperbolic equation system, the operator reads,

$$L_{\mathcal{T}} = -M_{\mathcal{T}}^{-1} \times M_{\partial\mathcal{T}} \times \mathrm{diag}(w_g) \tag{2.87}$$

$\mathbf{F}_n^g(Q_h^-, Q_h^+)$ represent the normal trace of flux function evaluated at surface cubature points and multiplied with face scale ,i.e. $J_{\partial D_k}/J_{D_k}$.

First operation of the surface kernel storing elemental field variables on shared memory. For internal degrees of freedoms, e.g. Q_h^-, a vector of size N_p is used for each variable. External field variables, e.g. are stored on the shared vector in size of $N_{\mathcal{S}} \times N_p$ where $N_{\mathcal{S}}$ is the number of connection for the work group. For conformal discretization and when single element processed by the work group, $N_{\mathcal{S}}$ denotes the number of faces i.e. 2 and 3 for triangle and tetrahedron. N_p work item is used for this operation.

Second operation is to interpolate the standard nodal values to surface cubature points. This operation includes multiplication pre stored shared vectors with the same interpolation matrix defined on the reference element, e.g. $\mathcal{I}_g \times Q_h$. Similar to volume kernel, each work item multiplies one row of I_g and the vectors. Interpolated values and geometric data are stored on the register memory. Then, flux function at cubature nodes, $\mathbf{F}_n^g(Q_h^-, Q_h^+)$ is computed using previously obtained register memory values. Flux function is stored on the shared memory vector of size $N_f \times N_g$ which is equal to assigned number of work items for this step.

Finally, flux function is lifted to the interpolation nodes. This step involves the matrix - vector multiplication given in Eq. 4.29. N_p work item is assigned for this operation. To accommodate all different tasks, surface kernel request $K_{\mathcal{S}} \times \max(N_f \times N_g, N_p)$ work items where $K_{\mathcal{S}}$ denote the elements processed by the work group.

2.3.3 Update Kernel

Update kernel computes time integration step which involves the global vector operations using computed right hand side vectors and some level of history

depending the integration method. $K_{\mathcal{U}} \times N_p$ work item is requested by the kernel, $K_{\mathcal{U}}$ represents the number of elements processed by each work group in the kernel.

2.3.4 Kernel Tuning

Performance of the mentioned kernels are highly dependent to the hardware, memory usage, tuning parameters etc. Some strategies for performance improvement are discussed below.

Coalescing: All elemental nodal values, solution field and geometric data, and local operators are read from the global memory continuously to maximize the bus utilization.
Unrolling: Loops are unrolled to reduce the number of instructions, end of loop checks etc. to minimize the overhead in memory read/write.
Padding: The size of vectors holding elemental nodal values are increased with the factor of 4 to align the access and minimize bank conflicts.
Memory Usage: All nodal values associated with the element and local projection/interpolation operators are stored on the shared memory vectors to efficiently reuse in the kernel. On the orher hand, shared memory is limited on GPU architectures and excessive usage of this memory cause the reduction in the number of active work groups. As a result, usage of shared-global memory for the kernels requires an optimization study. In our implementation, shared memory is used to hold local operators and nodal values for triangle but interpolation operators for tetrahedral elements are read from the global memory to reduce the shared-memory usage.
Element Number/Block: Multiple elements are processed by a single work group in the kernels to better align the data and utilize hardware resources. Optimum number of elements per work group depend hardware, approximation order and changes from kernel to kernel also. We use multiple elements only in low order approximation of 2D tests. In, high order 2D and all 3D problems, kernels uses single element per work group to reduce the shared memory usage. Determining optimum element number is beyond the scope of this study because

it is highly dependent to architectures and portability is our main concern for now.

CHAPTER 3

A GPU ACCELERATED ADAPTIVE DISCONTINUOUS GALERKIN METHOD FOR LEVEL SET EQUATION

This chapter presents a GPU accelerated nodal discontinuous Galerkin methods for the solution of two and three dimensional level set equation on unstructured adaptive meshes. Using local dynamic grid, computations are localized mostly near the interface location to reduce the computational cost. Small global time step size resulting from the local adaptivity is avoided by local time-stepping based on a multi-rate Adams-Bashforth scheme. Platform independence of the solver is achieved with an extensible multi-threading programming API as common kernel language that allows runtime selection of different computing devices (GPU and CPU) and different threading interfaces (CUDA, OpenCL and OpenMP). Overall, a highly scalable, accurate and mass loss free numerical scheme that preserves the simplicity of level set formulation is obtained. Efficiency, performance and local high-order accuracy of the method are confirmed through distinct numerical test cases.

3.1 Introduction

In free surface and multiphase flows, two commonly used approaches to dynamically represent the interface are interface tracking and capturing. Interface tracking methods are Lagrangian or semi-Lagrangian, where the mesh explicitly represents the interface. On the other hand interface capturing methods

are Eulerian, where the interface is represented by an implicit function defined on a fixed mesh. Interface tracking methods are generally accurate and robust but difficult to use when the interface encounters topological changes such as merging and breaking up.

Interface capturing methods can be classified as volume of fluid (VOF) [83] and level set (LS) [133]. VOF methods have generally excellent conservation properties, but high order reconstruction of interface from volume fraction information and obtaining geometry dependent properties such as surface normals and curvature are difficult to compute. Due to this limitation VOF methods are usually second order accurate at most. Details of VOF based methods, their strengths and weaknesses are introduced in Sec. 1.1.2. LS methods address the problems of VOF methods and are used extensively to study multiphase flows, computer vision, material science, and biology ([159, 135] and references therein).

In the LS method an interface, Γ is represented as the zero LS of an at least Lipschitz continuous function, ϕ [133]. Considering d-dimensional Cartesian plane, the function $\phi(\mathbf{x}, t)$ is positive in one region and negative in the other, with zero level contour of $\phi(\mathbf{x}, t)$ always representing the current location of the interface $\Gamma(t) = \{\mathbf{x} \mid \phi(\mathbf{x}, t) = 0\}$. Fig. 3.1 illustrates the level set function for a circular interface problem. Γ is shown with red solid line and the level set function takes negative and positive values above and below the interface, respectively. ϕ is a Lipschitz continuous function having a kink point at center of the circle.

If the motion of the interface is determined by the external velocity field $\mathbf{u}$, evolution of the interface is given with a first order hyperbolic equation as,

$$\phi_t + \mathbf{u} \cdot \nabla \phi = 0 \tag{3.1}$$

This implicit representation of interface motion offers many advantages, such as straightforward extension from 2D to 3D, simple handling of topological changes and easy calculation of geometric properties. The most popular traditional techniques to solve Eq.(3.1) are high order finite difference Hamilton-Jacobi essentially non-oscillatory (HJ-ENO) and weighted essentially non-oscillatory (HJ-WENO) schemes [134, 204, 88]. But generalization of these finite difference

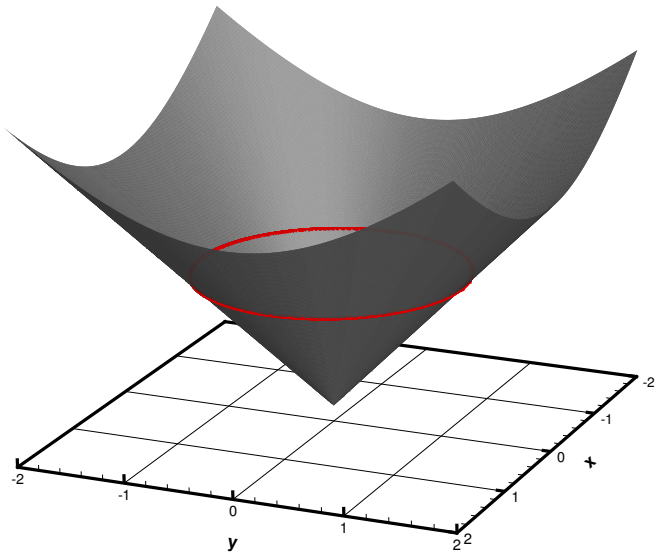


Figure 3.1: Level set function for 2D circular interface problem.

schemes on unstructured meshes are quite complicated [204].

The level set interface modeling does not enforce the discrete mass conservation as observed in Eq. 3.1. This problem is needed to be controlled especially in the regions of high curvature and long, thin filamentary structures due to excessive amount of numerical diffusion. Various techniques have been proposed to improve the conservation properties of LS method. Frequently used ones are the hybrid methods such as combining LS method with volume of fluid [180, 94], or with a Lagrangian method resulting in particle LS method [56, 57] and applying mass correction procedures [177, 178, 192]. Common to all these efforts is the fact that simplicity of the original LS method is lost.

There are also efficient adaptive methods that keep the computational work mainly in a small neighborhood of the interface. Sussman et al. [176] presented an adaptive LS approach for computing incompressible two-phase flows in two and three dimensions. Strain [172, 173] proposed an adaptive mesh re-

finement strategy by combining a tree structure with the LS method and a semi-Lagrangian time-stepping scheme. Sochnikov and Efrima [166] presented an algorithm using narrow band LS method on dynamically adaptive grids. Losasso et al. [113] introduced a particle LS method using octree data structure for free surface flows. Min and Gibou [122] presented a LS method on adaptive non-graded Cartesian grids using semi-Lagrangian scheme. Cecil et al. [21, 22] proposed an ENO adaptive generalized binary tree method. Herrmann [78] introduced a balanced force-refined LS method for two-phase flows on unstructured grids. Recently, a LS method for the motion of high codimensional objects (e.g., curves in 3D) on two level uniform adaptive Cartesian grid is presented by Wang and Xiang [191].

The discontinuous Galerkin (DG) method is a class of finite element methods that make used completely discontinuous, piecewise polynomial approximations for space discretization. Recently DG methods become highly attractive and popular, mainly because they are high-order accurate, nonlinear stable, highly parallelizable, can easily be used with complicated geometries and boundary conditions, and are capable of capturing discontinuities without spurious oscillations ([82] and references therein). The DG method can resolve the kinks with discontinuous derivatives even for long time integrations due to low numerical dissipation that can be achieved by the use of high order polynomial approximations [1]. Merchandise et al. [117] and Sussman et al. [179] showed that DG method gives more accurate results and exhibits less mass loss compared with the classical HJ-ENO/WENO schemes.

Increasing resolution near the interface substantially improves the conservation properties of standard LS formulation even for DG discretizations, local spatial mesh refinement reduces the allowable time step size when a global explicit time integrator is used. Multi-rate schemes use various time step sizes that satisfy the local CFL condition on different elements. Standard time integration schemes are applied to bulk groups with local time step sizes in a way that computational time can be drastically reduced. Different local time integration schemes are proposed in the literature. For example, multi-rate Adams-Bashforth methods are applied to electromagnetic wave propagation [74], damped wave problems [71]

and shallow water equations [63]. Also, multi-rate Runge-Kutta (MRRK) time steppers are proposed for general conservation laws [52, 47], geophysical flows [157] and coastal ocean modeling [53]. Different from the previous studies, we developed an efficient local time stepping strategy for the adaptive DG scheme which does not require additional storage or computational effort and discuss the efficient implementation on multi-threaded architectures.

Weak element connection and high-order approximation space in DG method lead local memory access and high arithmetic intensity. These properties make DG method well suited for multi-threaded architectures specially for GPUs. Recently, performance of the nodal DG methods on massively parallel architectures are demonstrated for several applications [63, 62, 123, 99]. Developed LS formulation is further accelerated using modern GPUs and many-core CPUs. Platform independence is achieved using OCCA [121] kernel language that abstracts common multi-threading languages (OpenCL, CUDA, pThreads and OpenMP) and offers flexibility to test the developed solver by choosing architecture and programming language at runtime.

In this chapter, we introduce a highly scalable, mass-loss free, multi-rate discontinuous Galerkin method on adaptive unstructured grids which preserves the simplicity of the level set formulation. The rest of the chapter is organized as follows: Sec. 3.2 provides the mathematical formulation, discretization of level set equation and basic properties adaptivity. In Sec. 3.2.2, multi-rate/multi-level local time stepping scheme and its efficient implementation is introduced. Then, parallelization of the method on many-core/multi-threaded architectures is discussed in Sec. 4.3, Finally, numerical results that demonstrate the accuracy, mass conservation and scalability of the method for two and three dimensional tests are given in Section Sec. 3.4.

3.2 Discretization

3.2.1 Level Set Equation

In this study, we mainly consider the externally known incompressible velocity fields that satisfy, $\nabla \cdot \mathbf{u} = 0$. Using the identity, $\nabla \cdot (\mathbf{u}\phi) = \phi \nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla \phi$, level set equation can be written in conservative form as

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\mathbf{u}\phi) = 0 \tag{3.2}$$

Then, multiplying Eq. 3.2 by a smooth test function, $v \in \mathcal{P}_N(D_k)$ and applying the divergence theorem leads to the following DG scheme in weak form

$$(\frac{\partial \phi}{\partial t}, v)_{D_k} - (\mathbf{u}\phi, \nabla v)_{D_k} + (F_n, v)_{\partial D_k} = 0 \tag{3.3}$$

where F_n is the normal trace of the flux function at element boundary. Value of the numerical flux depends on the two values of approximate solutions, ϕ^- and ϕ^+ at the common face. The numerical flux can be any two-point Lipschitz flux, which is monotone, consistent with the flux function, $\mathbf{n} \cdot \mathbf{u}\phi$ and conservative such that only one flux is defined at ∂D_k. We choose the upwind flux which is efficient in implementation on parallel architectures due to low conditional statements,

$$F_n(\phi_k^-, \phi_k^+) = (\mathbf{n} \cdot \mathbf{u}) \frac{\phi_k^- + \phi_k^+}{2} + \frac{|\mathbf{n} \cdot \mathbf{u}|}{2} (\phi_k^- - \phi_k^+) \tag{3.4}$$

Non-homogeneous boundary conditions are imposed weakly by incorporating boundary value in the numerical flux given in Eq. 3.4 as the external trace.

Then, semi-discrete form of the equation can be derived easily following the outputs of the Sec. 2.2.

$$\begin{aligned} J_{D_k} \left(I_c^T \cdot \mathrm{diag}(w_c) \cdot I_c \right) \frac{d\phi_k}{dt} &= J_{D_k} \left(D_{\mathbf{r}}^T \cdot \mathrm{diag}(\mathbf{r}_k^{\mathbf{x}} w_c) \cdot I_c \right) \mathbf{u}_k \phi_k \\ &\quad - \sum_{f=1}^{N_f} J_{D_k,f} \left(I_g^T \cdot \mathrm{diag}(w_g) \cdot I_g \right) F_{n,f} \end{aligned} \tag{3.5}$$

where ϕ_k holds N_p discrete nodal unknowns associated with element k. w_c and w_g are the cubature weights for volume and surface integrations. $\mathbf{r}_k^{\mathbf{x}} = \partial \Psi / \partial \mathbf{r}$ are geometric factors related with the affine mapping, Ψ and $J_{D_k} = \det(\partial \Psi / \partial \mathbf{r})$ is

the Jacobian of the transformation which is constant for straight sided elements. Also, each face of the physical element is mapped to reference element with Jacobian, $J_{D_k,f}$. Then, defining local mass matrix, $M_{\mathcal{T}} = I_c^T \cdot \text{diag}(w_c) \cdot I_c$ and inverting left side of the equation, we arrive,

$$\begin{aligned}\frac{d\phi_k}{dt} =& M_{\mathcal{T}}^{-1} \left(D_{\mathbf{r}}^T \cdot \text{diag}(\mathbf{r}_k^{\mathbf{x}} w_c) \cdot I_c(\mathbf{u}_k \phi_k) \right) - \\ & M_{\mathcal{T}}^{-1} \sum_{f=1}^{N_f} \frac{J_{D_k,f}}{J_{D_k}} \left(I_g^T \cdot \text{diag}(w_g) \cdot I_g \right) F_{n,f}\end{aligned} \tag{3.6}$$

Eq. 3.6 can be further simplified for efficient implementation on many-core parallel architectures. Details will be given in Sec. 4.3. For sake of clarity, we finally rewrite the semi-discrete equations as the following abstract system,

$$\frac{d\phi_k}{dt} = \mathcal{V}_k(\phi_k) + \sum_{n=1}^{N_f} \mathcal{S}_k^f(\phi_k^-, \phi_k^+) \tag{3.7}$$

3.2.2 Local Time Stepping

To relax the time step restriction resulting from local adaptivity, a two rate multi-step Adams-Bashforth (MRAB) scheme with different order base Adams-Bashforth (AB) methods are designed for the adaptive DG-LS formulation. To reduce complexity of the scheme, elements are grouped according to their refinement levels. It is assumed that elements belonging to the same group have the same characteristic time scale and each time scale differs by an integer ratio so that two groups can be synchronized in the largest time-step. Elements in the initial coarse level form the slow group, G_s and all refined elements (siblings) form the fast group, G_f without checking the refinement levels. This grouping strategy is reasonable because majority of the elements are in either coarsest level or the finest level due to locality of the adaptive scheme.

To clarify basic features and coupling between the groups, let start designing MRAB(m,l) scheme with m'th order base method and l sub-steps on the mesh shown in Fig. 3.2. The mesh is obtained by l_M level local refinement so that time scales of the G_f and G_s have the relation, $\Delta t_s = l\Delta t_f$. Communication between the groups occur only through the second face of D_2 and first face D_3. It is assumed that all the required history is known for all elements at $t = n$.

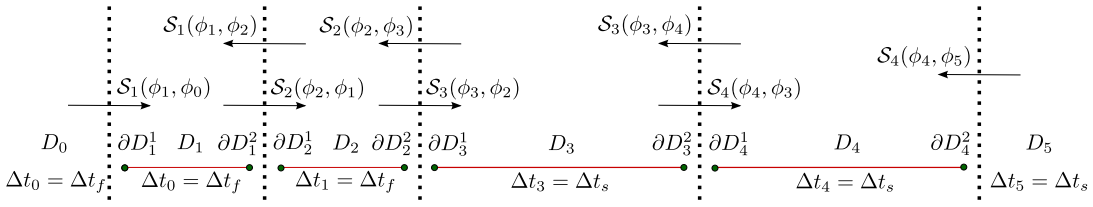


Figure 3.2: One dimensional unstructured sample grid for the local time stepping. Fast group includes D_0, D_1 and D_2, while slow group includes D_3, D_4 and D_5.

In the DG spatial discretization, elements are weakly connected with their first neighbors by fluxes. Once an element evolves in time with a stable time step size, it doesn't require additional information from the other time scales. In other words, coupling between the groups is also weak so that elements inside the groups but away from the slow-fast interface can be evolved with the base Adams-Bashforth integrator without any special treatment. We adapted fast is first approach [67] in the computations so that element 1 can be advanced from time level n to $n+l$ in l sub-steps with the time step size of Δt_f. For $i = 1, \cdots, l$

$$\phi_1^{n+i} = \phi_1^{n+i-1} + \Delta t_f \sum_{j=1}^{m} \beta_j (\mathcal{V}(\phi_1^{n+i-j}) + \mathcal{S}(\phi_1^{n+i-j}, \phi_0^{n+i-j}) + \mathcal{S}(\phi_1^{n+i-j}, \phi_2^{n+i-j})) \tag{3.8}$$

where β_j's denote the classical Adams-Bashforth coefficients. Similarly, element 4 is advanced to the same time level in a single step with Δt_s.

$$\phi_4^{n+l} = \phi_4^{n} + \Delta t_s \sum_{j=1}^{m} \beta_j (\mathcal{V}(\phi_4^{n-(j-1)l}) + \mathcal{S}(\phi_4^{n-(j-1)l}, \phi_3^{n-(j-1)l}) + \mathcal{S}(\phi_4^{n-(j-1)l}, \phi_5^{n-(j-1)l})) \tag{3.9}$$

The complexity of the MRAB comes from communication between the bulk groups while ensuring the accuracy. Element 2 requires ϕ_3^{n-1} to evolve from time level n to $n+1$ but it is not available because element 3 evolves in the slow time scale. We adapt a simple coupling approach based on using the newest available slow history [156]. This strategy results with the second order accuracy at the slow-fast interface [85, 156] and doesn't require any extra effort such as additional storage and interpolation of the slow history such as used in [171]. In our adaptive strategy, interface is always inside the finest level and away from

the slow-fast coupling as this coupling approach doesn't degrade the accuracy at groups interface. Then, element 2 is evolved in l sub-steps. For $i = 1, \cdots, l$

$$\phi_2^{n+i} = \phi_2^{n+i-1} + \Delta t_f \sum_{j=1}^{m} (\beta_j \mathcal{V}(\phi_2^{n+i-j}) + \mathcal{S}(\phi_2^{n+i-j}, \phi_1^{n+i-j}) + \mathcal{S}(\phi_2^{n+i-j}, \phi_3^{n-(j-1)l})) \tag{3.10}$$

Finally, slow-fast coupling is integrated from n to $n + l$ in a single step.

$$\phi_3^{n+l} = \phi_3^n + \Delta t_f \sum_{j=1}^{m} \beta_j (l\mathcal{V}(\phi_3^{n-(j-1)l}) + l\mathcal{S}(\phi_3^{n-(j-1)l}, \phi_4^{n-(j-1)l}) + \sum_{i=1}^{l} \mathcal{S}(\phi_3^{n-(j-1)l}, \phi_2^{n+i-j})) \tag{3.11}$$

A careful investigation of the 1D example reveals that the only extra effort compared with the single step AB scheme is the doubled evaluation of the boundary flux at faces located on the slow side of the coupling and it is minimal. Generalization to higher-order dimensions are quite straight-forward such that there are $l = 2^{l_M}$ sub-steps by assuming that each uniform refinement reduce characteristic length by half. Time step size of the G_f is defined, $\Delta t_f = \Delta t_s / l$ and $\Delta t_s = \min(Ch/|\mathbf{u}|)$ is the stable time step size for the initial coarse grid. $C = 1/(2N + 1)$ is the Courant number which is selected similar to those given in [46] for Runge-Kutta DG methods.

A sample buffer groups for 2D circular interface on $l_M = 2$ grid are illustrated in Fig. 3.3 showing pure slow (light gray elements), slow-fast buffer (medium gray elements) and pure fast (white elements) components. For the efficient implementation, only slow-fast buffer, where extra effort is required, is needed to be known.

Explicit Adams-Bashforth methods, as the MRAB scheme, are not self starting because they need $m + 1$ closest history. The initial values are provided with $(m + 1)l$ small time steps by Runge-Kutta methods which is one order higher than the base AB method to ensure that the temporal errors are only introduced by the MRAB scheme.

Efficiency of the two rate MRAB(m,l) scheme is approximated by assuming that the work load for each element,W_{D_k} is the same and dominates all the

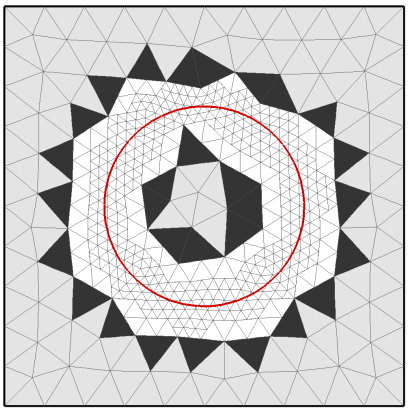


Figure 3.3: Multirate groups for an unstructured, locally refined grid ($l_M = 2$) for the circular interface problem.

computations. Also, it is assumed that slow-fast buffer is totally integrated with the slow time scale although, surface contribution coming from the fast side is computed in the fast time scale. Let the number of elements in the fast and slow time scales are K_f and K_s respectively. Under these assumptions, workload of the global single rate AB and MRAB time integrators for one synchronization level can be given as

$$W_{AB} = l \cdot K \cdot W_{D_k}, \quad W_{MRAB} \quad = l \cdot K_f \cdot W_{D_k} + K_s \cdot W_{D_k} \tag{3.12}$$

and the speed up will be

$$S_{th} = \frac{W_{AB}}{W_{MRAB}} = \frac{l}{1 + (l-1)K_f/K} \tag{3.13}$$

This speed-up approximation can be considered as the upper bound because of neglecting the extra flux computations at multi-rate interface and additional update stages. Efficiency of the MRAB increases when the number of fast elements decrease and number of sub-steps increase. It is expected that speed-up will be higher when the interface occupies a small portion of the problem and localized with the adaptivity as our scheme achieves.

3.2.3 Localizing Level Set Function

In most of the applications, the LS function are actually needed only near the zero LS. Far from the interface, only the sign of LS function is important to represent the interface dynamics. Therefore the LS function should be treated accurately within $|\phi| < \epsilon$ where ϵ is the band thickness. Focusing on the elements in a band around the interface has advantages in both lowering the computational work, preventing discontinuities away from the interface and imposing the boundary conditions. But errors at a distance slightly larger than ϵ may degrade the solution within the band while evolving the LS function [70]. To prevent this, LS function needs to be accurate in a slightly larger region, e.g. 1.5ϵ. Restricted LS function can be written as

$$\phi = \begin{cases} \phi & for \quad |\phi| \leqslant \epsilon \\ P_\epsilon(\phi) & for \quad \epsilon < |\phi| < 1.5\epsilon \\ \text{sgn}(\phi) 1.25\epsilon & else \end{cases} \tag{3.14}$$

where $P_\epsilon(\phi)$ is a polynomial with N order continuous derivatives which provides a smooth transition in the interval $\epsilon < |\phi| < 1.5\epsilon$ with a maximum value of 1.25ϵ. Fig. 3.4 shows the regularized LS function for circular problem on a $2D$ computational domain and $\epsilon = 0.1$. Here quite a large band thickness is used for illustration purposes. In calculations, band thickness is selected slightly larger than the minimum diameter of element in coarse level mesh unless otherwise is stated.

3.2.4 Mesh Adaptivity

DG discretizations are less diffusive comparing with the standard finite difference schemes, it is obvious that increasing resolution in the vicinity of the interface improves the accuracy. Adaptive mesh refinement (AMR) strategies are based on the conformal and non-conformal discretizations. In conformal discretizations, each face is shared by two elements so that AMR algorithm should handle the complicated mesh transition to make sure that mesh remains conformal after adaptation. This approach results with computational burden in refinement

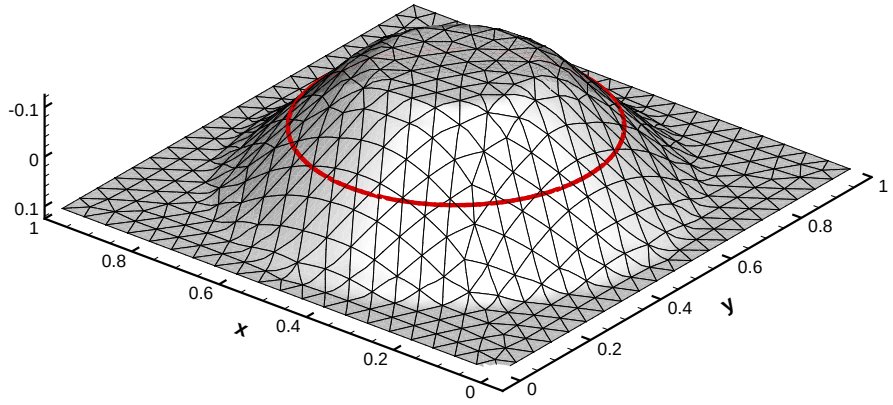


Figure 3.4: Regularized level set function for the circle centred at $(0.5, 0.5)$ with radius of 0.3 on computational domain of $[0, 1]^2$, $(\epsilon = 0.1, N = 3)$

step with easy calculation of fluxes. On the other hand, adaptation step in non-conformal AMR strategy can be simply obtained by dividing elements by pre defined levels. Then, numerical fluxes should be evaluated at non-conformal faces where more than two elements are connected. In this study, non-conformal discretizations on unstructured simplex elements is selected which enable us to get fast, more flexible and local adaptive grid.

In the adaptive scheme used here, the computational mesh consists of elements in a range of predefined levels with l_0 denoting initial coarse level and l_M being the maximum level. Refinement and coarsening are performed dynamically during the solution. The level of refinement and the elemental dependencies are stored in a hierarchical tree for efficient h type adaptivity.

Refinement is carried out in an isotropic way, i.e. a parent element is divided into siblings by connecting the mid-edges resulting with 4 and 8 children for triangle and tetrahedron, respectively. A threshold value,γ is selected to mark the elements for refinement. There is a flexibility for choosing the threshold value such as a predefined band width thickness or characteristic element length. If the $\min |\phi_k| \leqslant \gamma$ holds and refinement level of the element is smaller than the predefined maximum refinement level $(l_k < l_M)$, then the element is marked

for refinement and the approximation on the parent element is projected onto its siblings. If $\min |\phi_k| > \gamma$ holds and refinement level of all four siblings are larger than the initial coarse level, these elements are marked for coarsening and combined by removing all siblings and their newly created vertices. Then, solution of the siblings are projected to the parent element and level of refinement information is updated. Projection from and to siblings is totally local operation which can be achieved by local matrix, defined on reference element, and vector multiplications.

The DG method supports arbitrary number hanging nodes per face but it is restricted to decrease the computational complexity in flux evaluation. Adaptive mesh is $3:1$ and $2:1$ balanced for triangular and tetrahedral elements. In other words, a face of triangular and tetrahedral element can connect 4 and 3 elements at most, respectively. Any $1:1$ connection (conformal pair) is a member of $2:1$ balanced grid and so on. Fig. 3.5 shows mesh balancing to obtain $3:1$ grid. The mesh adaptation strategy is a propagation problem and always performed on CPU. Efficient, GPU accelerated adaptation is beyond the scope of this study and will be covered in a future work.

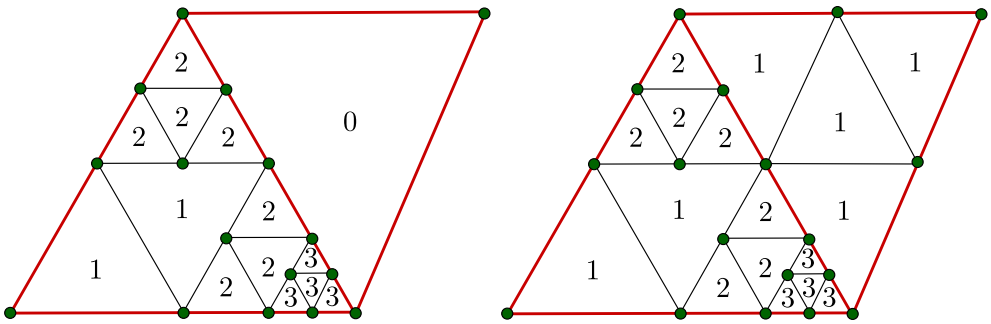


Figure 3.5: Mesh balancing for a triangular element. (a) Adaptation producing a connection not a member of $3:1$ balanced mesh. (b) Obtaining balanced grid with an extra refinement.

3.3 Parallel Implementation

Weak element connectivity of the DG discretizations enable element-local computations resulting with locality of memory access. In addition, the high order approximation space used, increase the amount of computations per degree of freedom and hence, computational intensity. These features are all well-suited for the parallelization on many-core/multi-threaded architectures.

The developed method is coded in C++ and OCCA [120] kernel language. OCCA is a abstracted programming model used to encapsulate native languages for parallel devices such as CUDA, OpenCL, Pthreads and OpenMP. Therefore, OCCA allows customized implementations of algorithms for several computing devices with a single code and offers flexibility to choose hardware architectures and programming model at run-time.

In our parallel model, a work group computes the integrals of one element while a work item in a work group computes the contribution from each integration node in the kernels as shown in Fig. 3.6. There are three major computa-

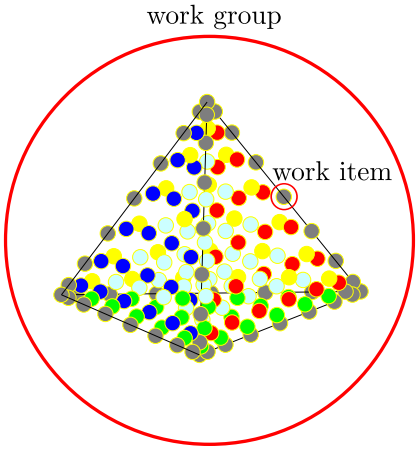


Figure 3.6: A work group and work item for 3D kernel.

tions required to solve the equation; volume integration, surface integration, and time-step update which are performed by Volume, Surface and Update kernels respectively. Rest of the section, we introduce the parallel implementation and basic performance improvement for the kernels.

3.3.1 Volume Kernel

Volume integrals are computed in this kernel. $\mathcal{V}(\phi_k)$ is in the size of N_p on each element and can be written in the following generic form,

$$\mathcal{V}(Q) = P_{\mathbf{r}} \cdot \mathbf{cF} = \sum_{i=1}^{d} P_{r_i} \times cF_i \tag{3.15}$$

where P_{r_i} are the projection operators of size of $N_p \times N_c$ defined on i'th direction of reference element. Projection operators are obtained pre multiplying local derivative matrices with inverse mass matrix and cubature weights to accelerate computations. cF_i denote the numerical volume vector defined on the cubature integration points. In terms of the previously defined operators,

$$P_{r_i} = M_{\mathcal{T}}^{-1} \times {D_{r_i}}^T \times \mathrm{diag}(w_c), \quad cF_i = \sum_{j=1}^{d} \frac{\partial r_i}{\partial x_j} \times I_c \times (u_j \phi_k) \tag{3.16}$$

First operation of kernel is to copy elemental field variables from global memory to shared memory i.e. obtaining $\mathbf{u}_k, \phi_k$ vectors. N_p work items are required for this operation. Then, stored shared memory variables are interpolated to the cubature integration points with matrix-vector multiplication all using the same interpolation matrix, $I_c \times (u_j \phi_k)$. Each work item multiples one row of I_c with the vectors to avoid memory conflicts. Resulting nodal values and geometric data, $\partial r_i / \partial x_j$ are stored on register memory for fast evaluation cF_i . N_c work items are assigned for this operation. After this operation, volume flux terms are computed and stored on the shared memory vectors. N_c work items are used for this operation. Finally, volume terms are interpolated to interpolation nodes, i.e. $P_{r_i} \times cF_i$ operation which involves d matrix-vector multiplication performed similarly previous one to prevent the memory conflicts. N_p work item is used for this operation. Number of required work items change within the kernel to perform individual tasks. To accommodate the number for work items required for the all computations, kernel request $\max(N_c, N_p)$ work items.

3.3.2 Surface Kernel

Surface contribution is computed in this kernel. $\mathcal{S}(\phi_k^-, \phi_k^+)$ is vector of length N_p for each field on element. Surface integral term can be written in the following generic form,

$$\mathcal{S}(\phi_k^-, \phi_k^+) = P_g \times F_n^g(\phi_k^-, \phi_k^+) \tag{3.17}$$

where P_g is the projection operator of $N_p \times (N_f \times N_g)$ that interpolates the surface terms to interpolation nodes. P_g is defined on the reference element and pre multiplied with inverse mass matrix and surface cubature weights such that,

$$P_g = -M_{\mathcal{T}}^{-1} \times I_g \times \text{diag}(w_g) \tag{3.18}$$

$F_n^g(\phi_k^-, \phi_k^+)$ represent the normal trace of flux function evaluated at surface cubature points and multiplied with face scale, $J_{\partial D_k}/J_{D_k}$.

First operation of the surface kernel storing elemental field variables on shared memory. For internal degrees of freedoms, e.g. ϕ_k^-, u_k^-, a vector of size N_p is used for each variable. External field variables, e.g. ϕ_k^+, u_k^+ are stored on the shared vector in size of $N_f \times N_p$ where N_f is the number of connection for the work group. For conformal discretization N_f denotes the number of faces i.e. 3 and 4 for triangle and tetrahedron. N_p work item is used for this operation.

Second operation is to interpolate the standard nodal values to surface cubature points. This operation includes multiplication pre stored shared vectors with the interpolation matrix defined on the reference element, e.g. $\mathcal{I}_g \times u_k$. Similar to volume kernel, each work item multiplies one row of I_g and the vectors. Interpolated values and geometric data are stored on the register memory. Then, flux function at cubature nodes, $F_n^g(\phi_k^-, \phi_k^+)$ is computed using previously obtained register memory values. Flux function is stored on the shared memory vector of size $N_f \times N_g$ which is equal to assigned number of work items for this step.

Finally, flux function is lifted to the interpolation nodes. This step involves the matrix - vector multiplication given in Eq. 4.29. N_p work item is assigned for this operation. To accommodate all different tasks, surface kernel request $\max(N_f \times N_g, N_p)$ work items.

3.3.3 Update Kernel

Update kernel computes local time integration step which involves the global vector operations using computed right hand side vectors and required level of history depending the order of integration method. N_p work item per element is requested by the kernel.

Performance of the kernels are highly dependent to the hardware, memory usage, tuning parameters etc. Some basic strategies for performance improvement are discussed below. Please note that these are not the only tuning strategies and many other method can be used such as multiple elements per work group, hardware dependent padding etc. which require an optimization study and will not covered here.

Coalescing: All elemental nodal values, solution field and geometric data, and local operators are read from the global memory continuously to maximize the bus utilization.
Unrolling: Loops are unrolled to reduce the number of instructions, end of loop checks etc. to minimize the overhead in memory read/write.
Padding: The size of vectors holding elemental nodal values are increased with the factor of 4 to align the access and minimize bank conflicts.
Memory Usage: All nodal values associated with the element and local projection/interpolation operators are stored on the shared memory vectors to efficiently reuse in the kernel.

3.4 Numerical Tests

We solved the LS evolution for 2D and 3D test problems on parallel CPU-GPU platforms. To evaluate the mass conservation of numerical scheme, volume is computed according to following formula,

$$V = \sum_{k=1}^{K} \int_{D_k} H(\phi) d\boldsymbol{x}, \tag{3.19}$$

where $H(\phi)$ is a indicator function with $H(\phi) = 1$ if $\phi \geqslant 0$ and $H(\phi) = 0$ otherwise. In all of the numerical experiments, MRAB scheme is used for the

$m = 2^{l_M}$ sub-steps with the second order base method if otherwise stated.

3.4.1 2D Zalesak's Rotating Disk

Zalesak [202] proposed a test to demonstrate how well the interface capturing method transports the interface. This problem is a good indicator of diffusion errors of the schemes and used frequently in the literature. Problem initial data is a slotted disk centered at $(0.5, 0.75)$ with a radius of 0.15, a width of 0.05 and a slot length of 0.25 in the computational domain of $[0, 1]^2$. The constant vorticity field to transport Γ is given by

$$u_1 = -2\pi(x_2 - 0.5), \quad u_2 = 2\pi(x_1 - 0.5) \tag{3.20}$$

so that the notched disk completes one revolution in $1s$.

Accuracy of the present numerical method in terms of area loss (or gain) is tested on the fixed grid. Numerical test are performed in two different mesh resolution with characteristic length of $h = 1/20$ and $h = 1/25$ where $K = 898$ and $K = 1400$ and notched disk is approximately represented by 40 and 55 elements respectively. Table. 3.1 compares the present numerical solutions with widely used fifth order HJ-WENO level set (LS) and semi-Lagrangian particle level set (PLS) solutions reported by Enrigth et al. [56, 57]. For the comparison with the other methods, degree of freedom for $N = 3$ and $N = 5$ elements are 10 and 21, respectively so that third order discretization leads total degree of freedom of 8980 and 14000 for $h = 1/20$ and $h = 1/25$ grids which approximately corresponds to $h = 1/95$ and $h = 1/120$ rectangular grid. Similarly $N = 5$ discretization approximately corresponds to $h = 1/138$ and $h = 1/170$ rectangular grids. Results show that area loss problem is solved even in relatively unresolved cases.

Effect of different refinement levels for the area loss after one revolution is presented in Table 3.2. $h = 1/20$ is used as initial coarse grid. Then, mesh is dynamically adapted so that zero level set contour always lies inside the finest level elements. Both $N = 3$ and $N = 5$ results are presented. Dynamic grid improved area loss problem significantly and very accurate results are obtained

Table 3.1: Area loss/gain for 2D Zalesak's disk problem on fixed mesh.

Method	N	h	Area	Area Loss %
Present	3	1/20	0.05777	0.771
	3	1/25	0.05800	0.369
	5	1/20	0.05813	0.158
	5	1/25	0.05819	0.053
LS [56, 57]	-	1/200	0.05791	0.540
PLS [56, 57]	-	1/200	0.05810	0.200
Exact	-	-	0.05822	-

with increasing refinement level.

Table 3.2: Area loss/gain in Zalesak's disk problem for different maximum refinement levels and order of approximations.

		l_M			
	N	0	1	2	3
Area	3	0.057771	0.058292	0.058232	0.058222
	5	0.058189	0.058202	0.058216	0.058220
% Area Loss	3	0.7713	-0.1242	-0.0211	-0.0032
	5	0.0529	0.031	0.007	-0.0005

Fig. 3.7(a-c) show the initial coarse level representation of the interface, Lagrange interpolation of the initial data to $l_M = 1$ grid and interface after one full rotation for $N = 3$. Fig. 3.7(d) illustrates zoomed view of upper right corner of notched disk for $N = 3, l_M = 0$ - $N = 3, l_M = 1$ - $N = 3, l_M = 2$ and $N = 5, l_M = 2$ cases. Spacing between the minor ticks are given as 5×10^{-3}. Increasing mesh resolution near the interface and high order of approximation significantly reduce the dissipation and smearing the high radius of curvature regions. There is no visible difference between the initial and final interface shapes of $N = 5, l_M = 2$ solution. Our coarsest ($N = 3, l_M = 0$) and finest ($N = 5, l_M = 3$) solutions after one full revolution are illustrated in Fig. 3.8(a). High radius of curvature regions (lower and upper corners of the notched disk) are smeared out as expected due to low resolution in coarsest level. Fig. 3.8(b) shows zoomed view of lower right corner of notched disk for $N = 3, l_M = 0$ - $N = 3, l_M = 1$ and $N = 5, l_M = 3$ cases. The figure illustrates that smeared out corners are quarter circles with different radius. Our results confirmed by the theoretical work of Ainsworth [1] about diffusive and dispersive properties of high-order DG methods. Radius of quarter circles scale like $\frac{\pi h}{4N+2}$ so that radius

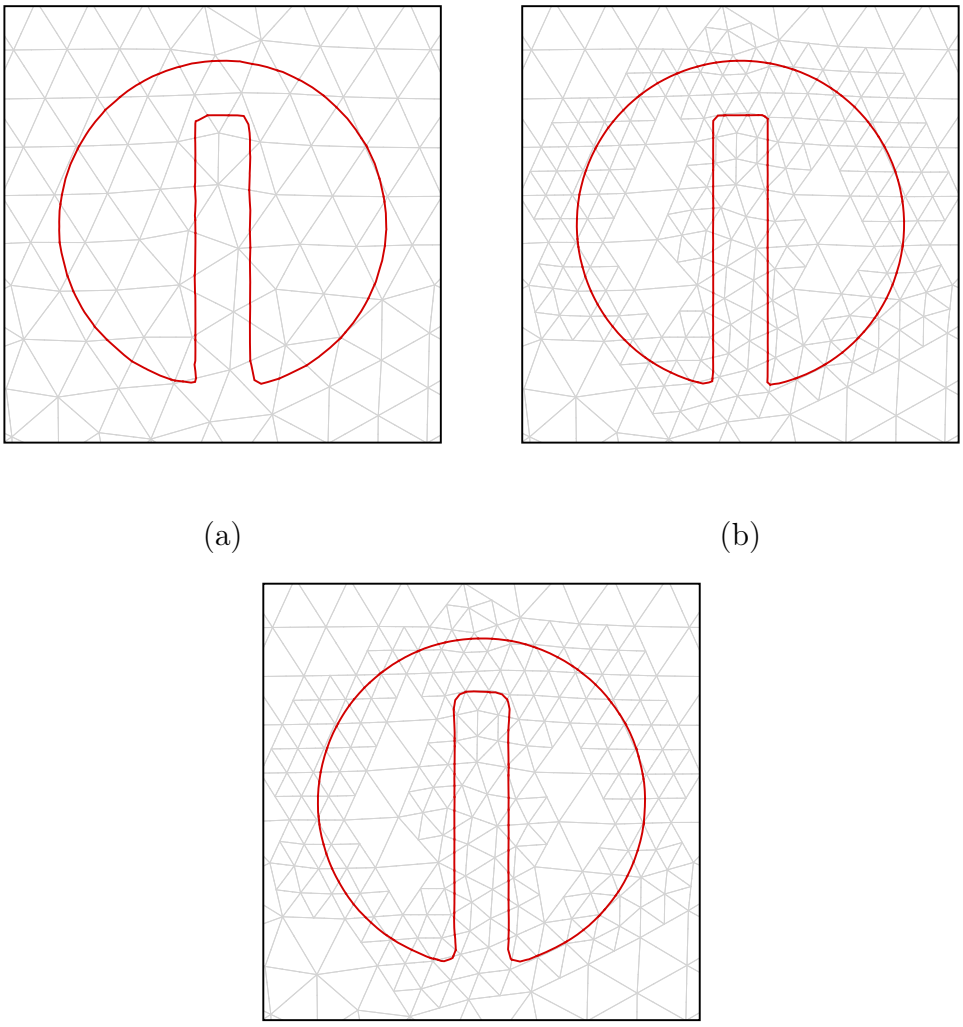


Figure 3.7: Interfaces for Zalesak's disk (a) Initial coarse level (b) Interpolated initial data (c) after one full rotation for ($N = 3, l_M = 1$) (d) comparison of N=5 and N=3 solution for $l_M = 1$.

of circles are reduced by a factor of 2 and 11/7 with each refinement level and increasing the order from 3 to 5, respectively.

To study the performance of MRAB scheme, 2 different initial mesh is used starting with the characteristic element length of $h = 1/20$. $h/2$ grid is generated with uniform global refinement. Number of elements in each group, size of sub-steps and distribution of elements, i.e. size of the slow-fast buffer are expected to effect predicted speedups. Using different initial grid enable us to obtain

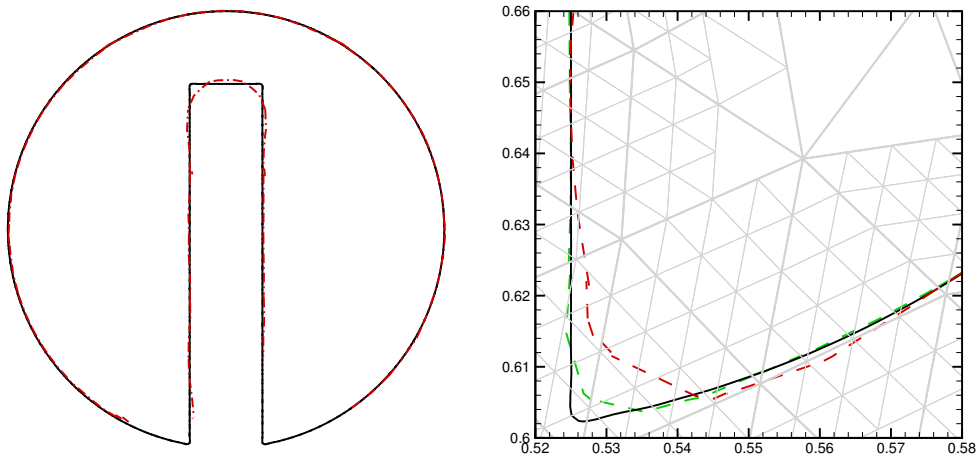


Figure 3.8: (a) Interfaces for $N = 3, l_M = 0$ and $N = 5, l_M = 3$ cases (b) Zoomed view of the lower right corner for $N = 3, l_M = 0$ - $N = 3, l_M = 1$ and $N = 5, l_M = 3$ cases from inside to outside, respectively.

various proportions of the fast elements with the same refinement strategy. For the calculation of theoretical speedup, time average of the fast group percentage is used. Numerical experiments shown in Table 3.3, reach the theoretical values computed according to the Eq. 4.27 for both $N = 3$ and $N = 5$. This is because all the fast elements are located together leading small slow-fast buffer size where it's workload contribution is neglected in the theoretical speedup calculations.

Table 3.3: Multi-rate time-stepping speedups for Zalesaks's disk test at different refinement levels and order of approximations.

		l_M								
		1			2			3		
	k	S	S_{th}	$\%\hat{G}_f$	S	S_{th}	$\%\hat{G}_f$	S	S_{th}	$\%\hat{G}_f$
h	3	1.38	1.40	43	1.33	1.37	64	1.17	1.20	81
	5	1.36	1.40	43	1.33	1.36	65	1.19	1.21	80
$h/2$	3	1.50	1.53	31	1.51	1.56	52	1.31	1.37	69
	5	1.52	1.53	31	1.46	1.49	56	1.26	1.29	74

Finally, we present the parallel performance of kernels on different architectures and multi-threading models. Fig. 3.9 shows the achieved GFLOPs of volume and surface kernels on NVIDIA Tesla C2075 GPU when kernels are cross-compiled with OpenCL and CUDA and on Intel Xeon E5-2670 CPU when kernels are compiled with OpenMP. Figure also illustrates the speedup of each model rela-

tive to serial CPU implementation. All performance numbers are obtained using the wall clock time from the beginning of one time step to the next one and averaged over a few hundred samples to minimize the timing transients. Similar performance between CUDA and OpenCL models are observed for both surface and volume kernels on GPU. GPU outperforms the CPU in all polynomial orders by a factor ranging from 25 to 100.

3.4.2 Vortex in a box

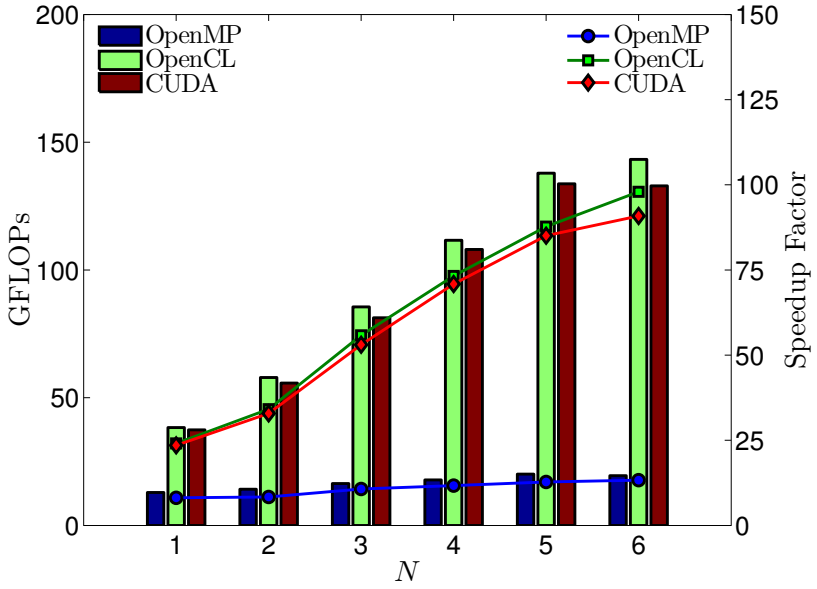
Single vortex in a box problem is solved to show the ability of present numerical scheme for resolving stretching interfaces. The problem initial data is a circle centered at $(0.5, 0.75)$ with a radius of 0.15. The deformation velocity field to transport the interface is given by

$$u_x = \sin^2(\pi x) sin(2\pi y) g(t), \quad u_y = -\sin^2(\pi y) sin(2\pi x) g(t) \tag{3.21}$$

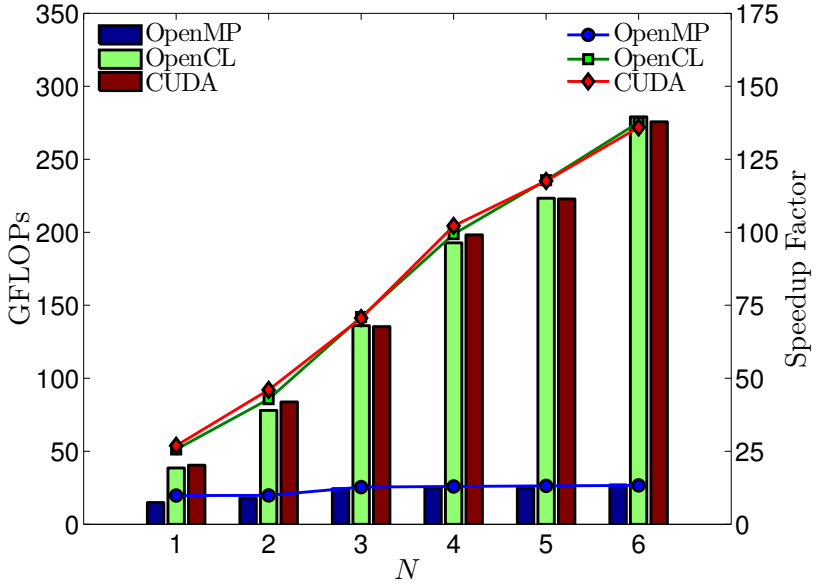
The function, $g(t) = \cos(\pi t/T)$ is used to reverse the flow field so that initial data should be recovered after one period, T. This makes the error analysis simple although flow field is quite complicated. Here $T = 8$ is used and interface reaches its most deformed form at half period, $t = 4$. To compare the numerical experiments, area loss as in Zalesaks disk problem and L_1 error, which is numerically evaluated by calculating initial data on very fine ($N = 5, l_M = 5$) grid and interpolating the computed $H(\phi)$ to the same fine grid, is used.

Table. 3.4 represents the area loss/gain and L_1 errors for different order of approximations and refinement levels for $h = 1/20$ initial mesh with fifth order HJ-WENO, LS and PLS solutions given in [56, 57] for $h = 128$. Results indicate that local high-order approximation and refinement significantly improves the area loss and interface shape even in the relatively unresolved solutions without any special treatment.

Ability of the present method to maintain thin filaments for different refinement levels is illustrated in Fig. 3.10. Contours in Fig. 3.10(a)-(c), show deformed interfaces for $N = 5$ at half period where maximum deformation occurs. Also recovered interfaces after one full rotation for different refinement levels and



(a) Volume Kernel



(b) Surface Kernel

Figure 3.9: Single precision GFLOPs and speedups of volume and surface kernels vs polynomial order on CPU and GPU using different multi-threading models. Speedups are computed according to serial CPU implementation.

$N = 3$ is represented in Fig. 3.10(d) where only $l_M = 0$ and $l_M = 1$ are shown because $l_M = 2$ solution nearly indistinguishable from the exact solution.

Similar to Zalesak's disk problem, 2 different initial mesh configurations, $h = 1/20$ and $h/2$ is used for the speedup tests. Computing the theoretical speedups for this case is not straightforward because element numbers change considerably

Table 3.4: Area loss/gain and L_1 Error for vortex in box problem

Method	N	l_M	Area	Area Loss%	L_1 Error
Present	3	0	0.06830	3.37	$2.5e^{-3}$
		1	0.07023	0.64	$4.8e^{-4}$
		2	0.07027	0.58	$4.3e^{-4}$
	5	0	0.07056	0.19	$1.5e^{-4}$
		1	0.07076	-0.10	$7.1e^{-5}$
		2	0.07067	0.02	$1.8e^{-5}$
LS [56, 57]	-	-	0.0425	39.8	$3.1e^{-2}$
PLS [56, 57]	-	-	0.0702	0.71	$1.0e^{-3}$
Exact	-	-	0.0707	-	-

during the analysis due to dynamic grid. Per time theoretical speedups are computed in each time-step and then time averages are calculated to get the global values. Numerical experiments confirm the theoretical upper bound but differences are slightly larger than the Zalesak's disk problem. Fast and slow groups are not compact unlike the previous problem which leads high fast-slow buffer region resulting slight deviation from the theoretical speedups.

Table 3.5: Speedups for Zalesaks's disk for different refinement levels for MRAB(3,m) with $m = 2^{l_M}$

		l_M					
		1			2		
	N	S	S_{th}	% G_f	S	S_th	% G_f
h	3	1.33	1.40	43	1.31	1.40	62
	5	1.34	1.41	42	1.31	1.38	63
$h/2$	3	1.45	1.54	30	1.55	1.64	48
	5	1.47	1.54	30	1.54	1.60	50

3.4.3 3D Deformation Field

Three-dimensional deformation field problem [106] is solved to show the ability of present numerical scheme for resolving stretching interfaces. The problem initial data is a sphere centered at $(0.35, 0.35, 0.35)$ with a radius of 0.15. The deformation velocity field to transport the interface is given by

$$\begin{aligned} u_1 &= 2\sin^2(\pi x_1)\sin(2\pi x_2)\sin(2\pi x_3)g(t) \\ u_2 &= -\sin(2\pi x_1)\sin^2(\pi x_2)\sin(2\pi x_3)g(t) \\ u_3 &= -\sin(2\pi x_1)\sin(\pi x_2)\sin^2(\pi x_3)g(t) \end{aligned} \tag{3.22}$$

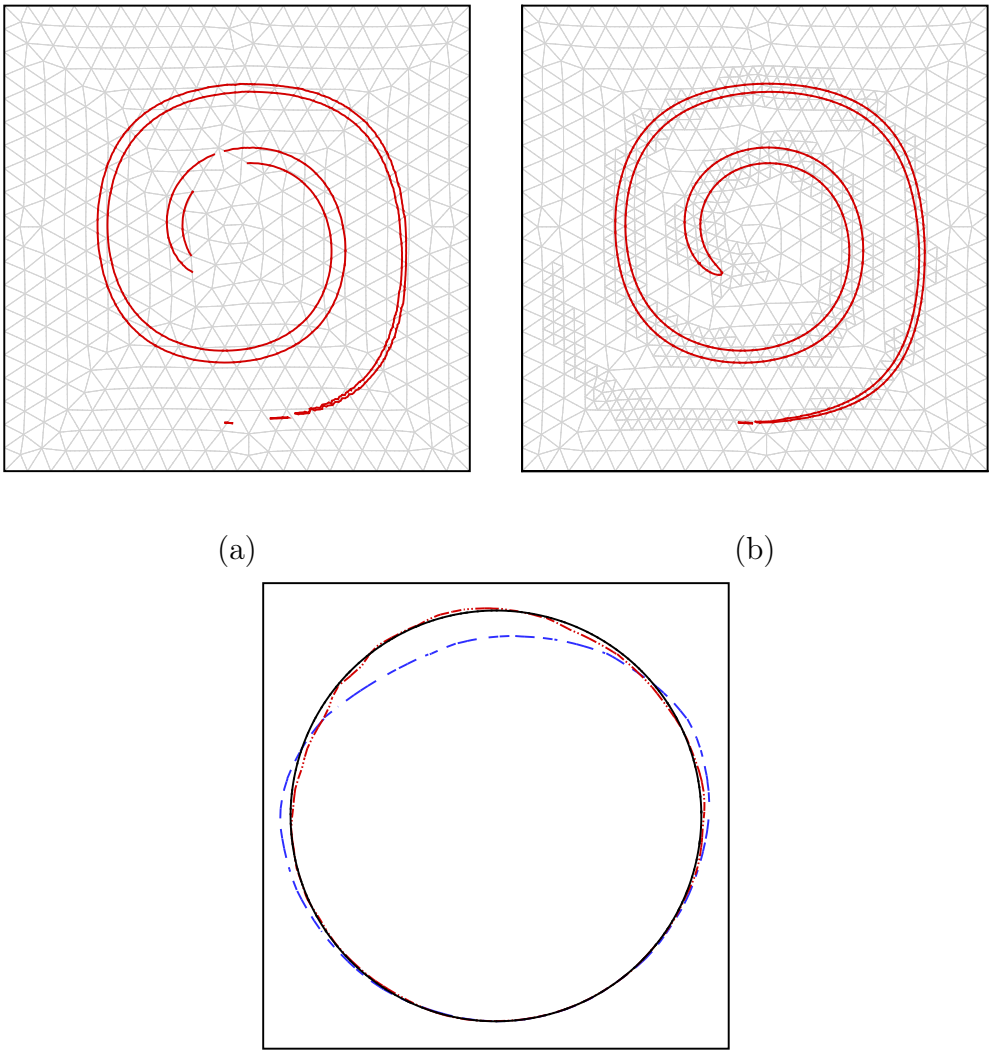
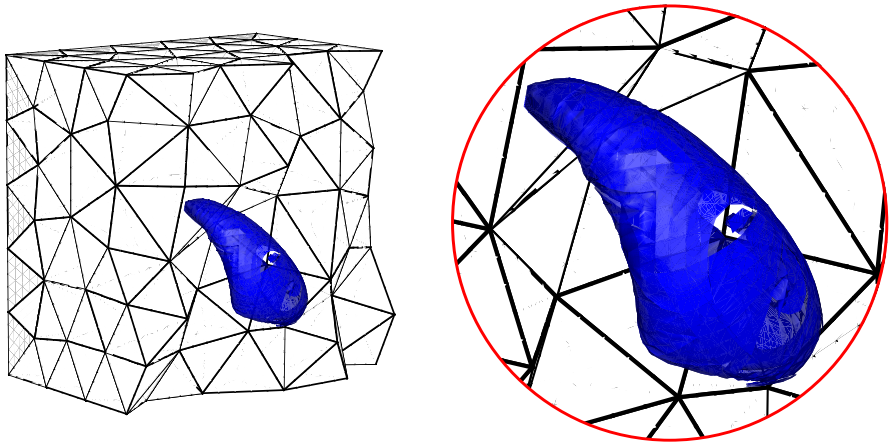


Figure 3.10: Vortex in a box problem deformed interfaces (a) $l_M = 0$ and (b) $l_M = 1$ and for $N = 5$ and at $t = T/2$ (c) recovered interfaces, exact solution (solid), $l_M = 0$ (dash-dot) and $l_M = 1$ (dashed) for $N = 3$ and at $t = T$

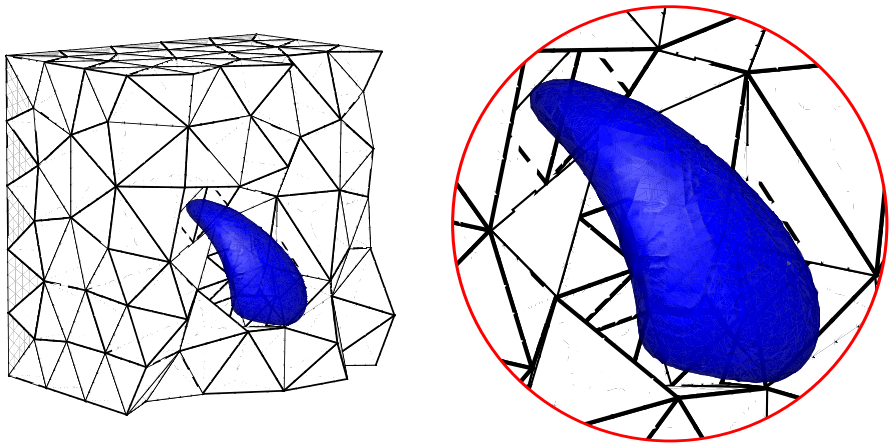
The function, $g(t) = \cos(\pi t/T)$ is used to reverse the flow field so that initial data should be recovered after one period, T. This makes the error analysis simple although flow field is quite complicated. Here, $T = 3$s is used as the interface reaches its most deformed form at half period, $t = 1.5$s.

Ability of the adaptive method to maintain thin filaments is illustrated in Fig. 3.11. We start with coarse grid with characteristic length of $h = 1/5$ ($K = 930$). An under-resolution is obvious in the fixed grid even at early times

of the simulation. Only one level refinement substantially improves the interface representation. The figure also shows how the non-conformal refinement leads locally refined grid where $K = 1280$ at that instanuskint.



(a) $l_M = 0$



(b) $l_M = 1$

Figure 3.11: Interface and mesh structure of fixed and one level locally refined grid for $N = 5$ at $t = 0.2$ s. Left: Domain part for $x < 0.35$. Right: Zoomed view of interface.

Fig. 3.12 shows evolution of the interface at times, $0, 0.4, 0.8, 1.2, 1.8, 2.2, 2.6$ and 3s for $l_M = 2$ and $N = 3$ starting with $h = 1/5$ coarse grid. Because flow is reversed at $t = 1.5$s, interface shape at time t should be the identical to the shape at $3 - t$. Slight under-resolution starts to be visible at $t = 1.2$ where filament became too thin. Comparing the LS and particle LS [56] and quadrature-free DG [117] solutions, our method well preserves the interface on relatively coarser grids. Note that even we use uniform refinement which gives

$h = 1/20$ ($K \approx 60000$), our resolution much lower then the previous solutions.

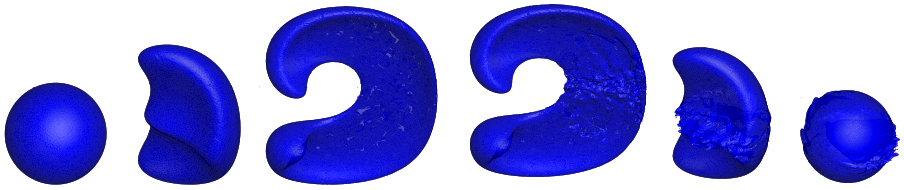
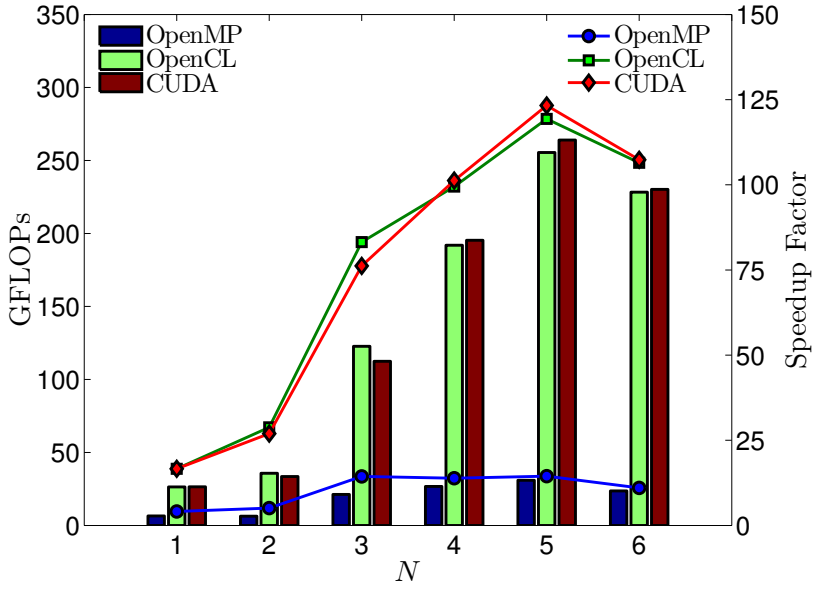


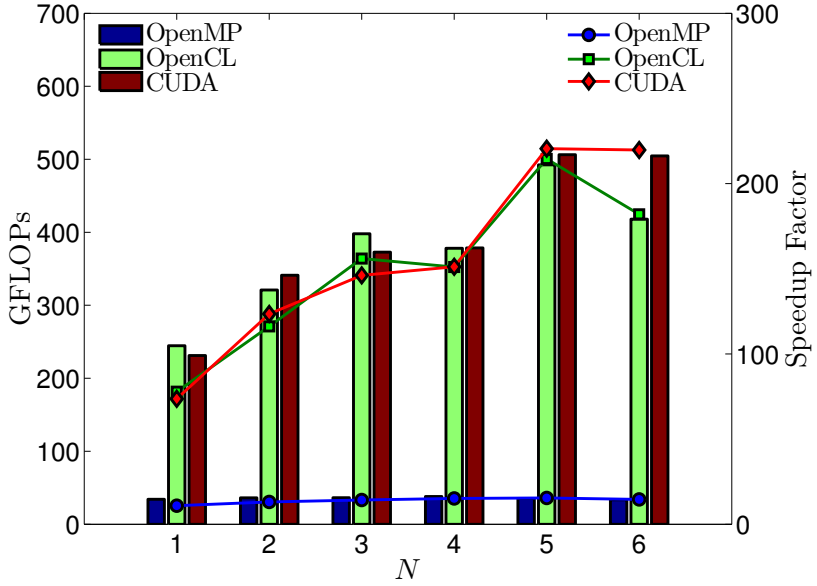
Figure 3.12: 3D deformation test case interfaces at $t = 0, 0.4, 0.8, 2.2, 2.6, 3.0$s for two level local adaptivity on $h = 1/5$ initial grid and $N = 3$.

Fig. 3.13 presents the floating point operations and speedups for 3D volume and surface kernels when cross-compiled with CUDA and OpenCL on Tesla C2075 GPU and with OpenMP on Intel Xeon E5-2670 CPU. Speedups are computed according to serial CPU implementation on the same machine. Volume kernels at polynomial orders of 1 and 2 give lower performance due to lower computational load. Surface kernel performs well at all orders due to good vectorization of upwinding flux on $\mathbf{u}$. Because CUDA ptx compiler can be hardware optimized, CUDA outperforms OpenCL on NVIDIA GPU specially at higher-order polynomial approximations. Very high speedups factors, around two orders of magnitude, are obtained on the GPU comparing the serial CPU code . For the practical order of approximations, $N = 3, 4$ speedup factors reach 100 and 150 for the volume and surface kernels, respectively.

Fig. 3.14 shows percentage of time spent for the main parts of the solver for polynomial orders 1,3 and 5 on one level adaptive grid starting with $h = 1/10$ initial mesh. Update kernel is not included in the figure due to its very low computational time. Adaptation time includes all mesh refinement/coarsening operations and data transform between CPU and GPU. As the polynomial order increases, percentage of time spent for the volume kernel increase while fraction of adaption step time decrease due to increased computational load. Resulting from the complexity of non-conformal flux evaluation, surface kernel dominates the all computations with a decreasing fraction as the polynomial order increases.



(a) Volume Kernel

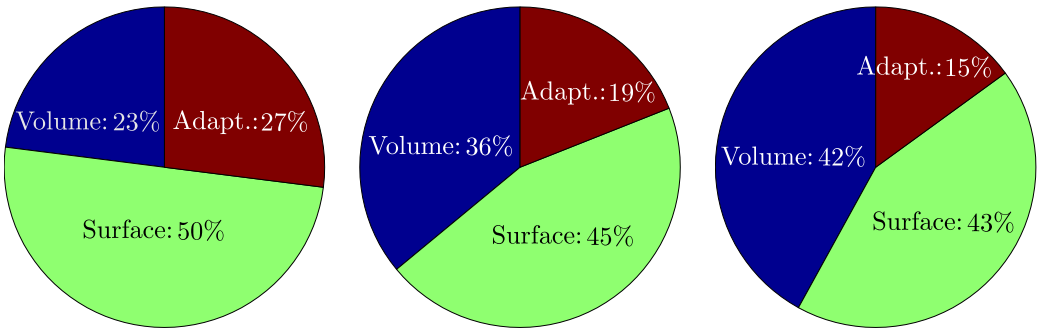


(b) Surface Kernel

Figure 3.13: Single precision GFLOPs and speedups of volume and surface kernels vs polynomial order on CPU and GPU using different multi-threading models. Speedups are computed according to serial CPU implementation.

3.5 Conclusion

Numerical results for the solution of the level set equation by high-order nodal discontinuous Galerkin methods on fully unstructured adaptive meshes are presented. Refinement and coarsening of the grid is performed only near the inter-



(a) $N = 1$ (b) $N = 3$ (c) $N = 5$

Figure 3.14: Percentage of time spent for the main parts of the solver at different orders.

face location to localize the computational effort. A two-rate multi-step Adams-Bashforth time integration is adopted for the method to avoid the severe time restriction resulting from the local refinement. Efficiency and high-order accuracy of the method for level set advection are confirmed by distinct test problems. Numerical results indicate that mass is well conserved without applying any special procedures. The numerical results indicate that the mass loss problem is not related with the level set formulation, but the numerical discretization scheme.

CHAPTER 4

A GPU ACCELERATED LEVEL SET REINITIALIZATION FOR AN ADAPTIVE DISCONTINUOUS GALERKIN METHOD

In this chapter, we consider a GPU accelerated high order reconstruction of signed distance functions in the level set formulation. The flow based reinitialization equation is discretized in space by using a nodal discontinuous Galerkin method on dynamic unstructured grids. An artificial diffusion approach with a modal decay rate based regularity estimator is used to damp out high frequency solution components near kinks where mesh adaptivity is applied. A multi-rate Adams-Bashforth time integrator is designed to avoid time step restrictions resulting from artificial diffusion stabilization and local mesh refinement. Platform independence of the solver is achieved with an extensible multi-threading programming API as common kernel language that allows runtime selection of different computing devices (GPU and CPU) and different threading interfaces (CUDA, OpenCL and OpenMP). Overall, a highly scalable numerical scheme which preserves the simplicity of the original level set method is obtained. Efficiency, performance and local high order accuracy of the method to construct signed distance function on highly disturbed initial data with smooth and non-smooth interfaces are confirmed through distinct two and three dimensional numerical test cases.

4.1 Introduction

In free surface and multiphase flows, level set (LS) methods [133] are commonly used to represent surface dynamics. Evaluation of LS equations in time often distorts the scalar LS function such that it generates flat or steep gradients near the interface. Reinitialization replaces LS function with the signed distance function, which has many advantages such as being regular and having uniform gradients. Basic algorithms for reinitialization can be categorized as fast sweeping [187] and fast marching [158] methods which are based on the solution of static boundary value problems, and flow based methods [181] which are based on defining an artificial flow to obtain signed distance function in steady-state.

Flow based methods are more flexible, accurate and easier to parallelize. The most popular flow based reinitialization equation is a pseudo time first order partial differential equation given by

$$\frac{\partial \phi}{\partial t} + \mathrm{sgn}(\phi_0)(|\nabla \phi| - 1) = 0, \quad \phi(\mathbf{x}, 0) = \phi_0 \tag{4.1}$$

In principle, interface location remains unchanged because $\mathrm{sgn}(\phi_0) = 0$, zero contour of ϕ and ϕ_0 are the same. Away from the interface, $|\nabla \phi|$ converges to 1 without explicitly locating the interface location. Except for the signum term, Eq. 4.1 is a Hamilton-Jacobi (HJ) equation. The general approach is to smear the signum term in a narrow band and treat Eq. 4.1 as a standard HJ equation with a smooth Hamiltonian. Finite difference Essentially Non-Oscillatory (ENO) and Weighted ENO (WENO) schemes have been developed to solve this equation on Cartesian grids [134, 88]. Although these schemes have been adapted to unstructured grids [204], they are complicated to implement and computationally inefficient. Additionally, these schemes result in mass loss problems in interface capturing applications [56].

Discontinuous Galerkin (DG) methods are a class of finite element methods that make use of completely discontinuous, piecewise polynomial approximations for spatial discretization and have excellent properties to overcome the problems mentioned above. DG can resolve kinks with discontinuous derivatives even over long time integrations due to the low numerical dissipation achieved by

the use of local high order polynomial approximations [1]. For these reasons, DG for LS advection gives more accurate results compared with standard HJ-ENO/WENO finite difference schemes [117, 179]. However it is rarely used in reinitialization, because the HJ equations can not be written in conservative form and it is hard to define suitable numerical fluxes in DG framework.

The first DG method for the solution of the general HJ equations is introduced in [84]. Accuracy and stability properties of this method are analyzed in [104] and reinterpreted for a simplified implementation with reduced computational cost in [107], though this method seems to have made the algorithm indirect, complicated and not optimal for reinitialization [179]. Then, the local discontinuous Galerkin (LDG) approach, which is first developed to discretize second order operators, is extended to solve HJ equations directly [199]. Recently, the DG method has found an application in multiphase-flow simulations with level set interface modeling in which reinitialization is required to maintain regularity of the LS function. In [70], LS function is reinitialized using a standard LDG method with an adaptive filtering and stream line diffusion approach for stabilization, but both the filter strength and diffusion coefficients remained as user defined parameters. In [179], a high order DG method for the reinitialization is introduced based on finding the closest distance between a constructed height function and a point thus, their algorithm is applicable only to structured cartesian grids and hard to implement in 3D. A fully implicit coupled DG scheme is used to solve unsteady incompressible multi-phase flow problems in [141]. They used a geometric reinitialization based on recursive contouring to localize the zero-level set but this simple reinitialization technique leads noise in interface and important mass losses without increasing the recursion level, hence the computational cost, significantly.

High order DG methods, like many other high-order methods, produce oscillations when the approximation space is inadequate to resolve the main features of the exact solution, such as kinks in the LS formulation. Different methods have been proposed to stabilize high-order DG discretizations. A successful approach is limiting, which is based on reducing polynomial order near discontinuous regions [44] or high order WENO type polynomial reconstruction [147]. On the

other hand, artificial diffusion, which relies on explicitly adding viscous terms to the governing equations in order to smooth solution near discontinuities, offers a stabilization mechanism for high-order discretizations. The amount and the region where viscosity should be added to avoid oscillations without excessive smearing requires sophisticated discontinuity detectors. An artificial viscosity stabilization for high-order DG method is introduced in [140] which resolves shocks on a sub-cell level. Their shock detector relies on the magnitude of the highest-order coefficients in an orthonormal representation of the solution. Later, this method is further improved in [98] by introducing a reliable and scaled smoothness detector that uses information from all modes instead of only the highest-order one.

Weak element connection and high-order approximation space in DG method lead local memory access and high arithmetic intensity. These properties make DG method well suited for multi-threaded architectures specially for GPUs. Recently, performance of the nodal DG methods on massively parallel architectures are demonstrated for several applications [63, 62, 123, 99]. Developed LS formulation is further accelerated using modern GPUs and many-core CPUs. Platform independence is achieved using OCCA [121] kernel language that abstracts common multi-threading languages (OpenCL, CUDA, pThreads and OpenMP) and offers flexibility to test the developed solver by choosing architecture and programming language at runtime.

Although the DG has become attractive for solution of multiphase flows problem with level set interface formulation, it is never used for constructing signed distance functions. In this chapter, we present a GPU accelerated explicit, flow based reinitialization scheme on unstructured dynamic meshes using high order nodal DG space discretization. An artificial diffusion mechanism with a reliable troubled cell detector is adapted to damp out the high frequency solutions. A multi-rate time integrator is designed to avoid the time step restriction resulting from the high order mesh dependent parameters. It is believed that this procedure can be effectively used in interfacial flow dynamics problems. The rest of the chapter is organized as follows: Sec. 4.2 provides mathematical formulation starting with the regularization of signum function and discretization

of reinitialization equation. Then, an artificial diffusion scheme, including irregularity detector, is given. This section is finished by introducing the time integrator and mesh adaptation strategy. Finally, in Sec. 4.4, numerical results that demonstrate the accuracy and flexibility of the method are given.

4.2 Mathematical Formulation

4.2.1 Regularize Sign Function

The general approach for solving Eq. 4.1 is to smear the signum term in a narrow band in the vicinity of the interface. Without regularization of the sign function, characteristics of the equation are emanating from the interface in the normal direction with the unit speed. However, the speed changes with the $\mathrm{sgn}_\alpha(\phi_0)$ term that selection of the regularization affects the steady state solution and the convergence rate/accuracy of the scheme.

Generally, regularized signum term is selected as $\mathrm{sgn}_\alpha(\phi_0) = \phi_0/\sqrt{\phi_0^2+\alpha^2}$, where α is the amplitude parameter chosen as a small, non-zero value related to the characteristic mesh size, h. Classical low order methods require to choose α at least to the order of characteristic mesh size, $\alpha = O(h)$. Due to the high order interpolation of the DG method, variations can be resolved and integrated stably in the smaller distance of $\alpha = O(h/N)$. Our numerical tests show that $\alpha = 2h/N$ offers sharp but smooth profile with good integrability of the reinitialization equation.

Regularization with only considering the mesh size, creates small coefficients resulting in small characteristic speeds, when the LS function becomes too flat near the interface. Also if the LS is too steep, it may change sign and lead to artificial movement of the interface position [138]. In this study, hyperbolic tangent function with modified amplitude parameter is used to regularize the signum term such as,

$$\mathrm{sgn}_\alpha(\phi_0) = \tanh(\frac{\pi\phi_0}{\alpha}) \tag{4.2}$$

where $\alpha = 2|\nabla\phi_0|h/N$, is scaled with the gradient of the initial level set function

to overcome the distorted initial data problem. This regularization function smooths the discontinuous coefficients in 2α neighborhood of the zero LS and offers smooth but sharp profile. Different regularization choices are compared in Sec. 4.4 in terms of convergence rate and accuracy on a problem with highly distorted initial data.

4.2.2 Discretization of Hamiltonian

After regularizing the signum term, the reinitialization equation can be written as the following standard HJ equation with smooth coefficients.

$$\frac{\partial \phi}{\partial t} + H(\frac{\partial \phi}{\partial \mathbf{x}}) = 0, \quad \phi(\mathbf{x}, 0) = \phi_0 \tag{4.3}$$

where $\mathbf{x} = (x_1, \cdot, x_d)$ are the Cartesian coordinates and Hamiltonian, $H(\frac{\partial \phi}{\partial \mathbf{x}})$ denotes $\text{sgn}_\alpha(\phi_0)(|\nabla \phi| - 1)$ with t being the fictitious time variable. Solving Eq. 4.3 requires an accurate approximation of solution derivatives which is calculated following the Local Discontinuous Galerkin (LDG) method [199]. The LDG formulation requires to define two new variables for each derivative component. Let $p^l_{x_i}$ and $p^r_{x_i}$ be auxiliary variables used to approximate $\frac{\partial \phi}{\partial x_i}$ when the fluxes are chosen from the left and right upwinding sides, respectively. Then, for the i^{th} component, we have two equations, such that

$$p^{l,r}_{x_i} - \frac{\partial \phi}{\partial x_i} = 0 \tag{4.4}$$

Let ϕ is approximated by $\phi_h \in \mathcal{V}_N$. Multiplying Eq. 4.4 with test functions, $v_h \in \mathcal{V}_N$, integrating over element domain and performing integration by parts leads to the following upwind DG scheme for in weak form,

$$(p^{l,r}_{x_i}, v_h)_{D_k} + (\phi_h, \frac{\partial v_h}{\partial x_i})_{D_k} - (\phi^{l,r}_{x_i} n_{x_i}, v_h)_{\partial D_k} = 0 \tag{4.5}$$

left and right upwind fluxes can be defined as,

$$\phi^l_{x_i} = \begin{cases} \phi^- & for \quad n_{x_i} \geqslant 0 \\ \phi^+ & for \quad n_{x_i} < 0 \end{cases} \quad \text{and} \quad \phi^r_{x_i} = \begin{cases} \phi^+ & for \quad n_{x_i} \geqslant 0 \\ \phi^- & for \quad n_{x_i} < 0 \end{cases} \tag{4.6}$$

It is obvious that volume terms are the same and only flux functions differ between the directional derivative approximations of each component. Finally,

Eq. 4.3 is integrated over the cell D_k to get,

$$(\frac{\partial \phi_h}{\partial t}, v_h)_{D_k} + \big(\bar{H}(p^l_{\mathbf{x}}, p^r_{\mathbf{x}}), v_h\big)_{D_k} = 0 \tag{4.7}$$

$\bar{H}(p^l_{\mathbf{x}}, p^r_{\mathbf{x}})$ is a monotone and consistent numerical Hamiltonian approximating $H(\frac{\partial \phi}{\partial \mathbf{x}})$. We use the Godunov Hamiltonian given below because it is efficient, easy to program and leads pure upwind scheme.

$$\bar{H}(p^l_{\mathbf{x}}, p^r_{\mathbf{x}}) = \mathrm{sgn}_\alpha(\phi_0) \begin{cases} \left(\sum_{i=1}^{d} \max((p^{l,m}_{x_i})^2, (p^{r,p}_{x_i})^2)\right)^{1/2} - 1, & \text{if} \quad \mathrm{sgn}_\alpha(\phi_0) \geqslant 0 \\ \left(\sum_{i=1}^{d} \max((p^{l,p}_{x_i})^2, (p^{r,m}_{x_i})^2)\right)^{1/2} - 1, & \text{else} \end{cases} \tag{4.8}$$

In Eq. 5.30, positive and negative parts of the approximate directional derivatives are defined as $p^{l,p}_{x_i} = \max(p^l_{x_i}, 0)$ and $p^{l,m}_{x_i} = -\min(p^l_x, 0)$.

It is important to mention that when the solution is smooth, $p^l_{x_i}$ is very close to $p^r_{x_i}$ while they differ significantly near the discontinuities. Thus, at discontinuous regions, ($p^l_{x_i}$, $p^r_{x_i}$) can capture the complete information of $\frac{\partial \phi}{\partial x_i}$. For a piecewise constant approximation, this scheme is monotone and converges to the entropy solution. However, stabilization is needed for higher order approximations. The use of artificial diffusion and regularity detector is discussed next.

4.2.3 Artificial Diffusion

Limiting prevents the oscillations near discontinuities by reducing the order of the approximation space and is used as a standard approach for stabilizing DG methods. Limiting can also be interpreted as introducing diffusion by explicitly omitting high order terms of the expansion basis and reconstructing the solution with lower degree polynomials. Reducing the order of the interpolating polynomial space degenerates the solution not only in elements having kinks but also in a wide surrounding region by adding diffusion of $O(h)$. It is also computationally expensive due to the reconstruction of the solution especially in non-conformal discretizations. Instead, we developed a fast and efficient artificial diffusion stabilization mechanism that introduces dissipation of $O(h/N)$ and resolves the

kinks in an element length. By adding the diffusion term, Eq. 4.1 can be recast into following form,

$$\phi_t + \mathrm{sgn}_\alpha(\phi_0)(|\nabla\phi|-1) = \nabla\cdot(\nu\nabla\phi) \tag{4.9}$$

where ν denotes the artificial diffusion coefficient. How much and where the viscosity should be added to avoid oscillation without excessive smearing out requires reliable and scalable troubled element detectors.

Klockner et.al. [98] introduced a modal regularity detector as an improvement to the method proposed in [140] by taking into account the all modes of expansion space for high-order DG solution of 1D transport equations. We generalized this approach to 2D simplex element which is challenging due to dependency on the adjacency of modal coefficients. Fig. 4.1 illustrates the dependency of the all modal coefficients in triangular element for the approximation order, $N = 3$. First, we grouped the same order modal coefficients and then obtain

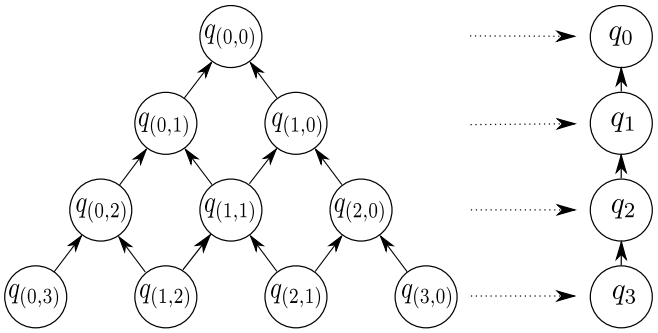


Figure 4.1: Relation of the modal coefficients for 2D and corresponding 1D polynomial expansions, $N = 3$

the corresponding 1D expansion space using a quadratic mean,

$$q_n = (\sum_{n=0}^{N} q_{(i,j)}^2)^{1/2} \quad \text{for} \quad i+j=n, \quad i,j \in \{0,\cdots,N\} \tag{4.10}$$

where double and single indexes show the 2D and computed 1D modal coefficients respectively. The coefficients in the expansion space are expected to decay fast for the smooth solutions. Rate of decay is decelerated when the solution is non-smooth. Thus, smoothness information can be approximated using a Fourier analogy, $|q_n| \sim cn^{-s}$. Taking the logarithm of the relation leads to

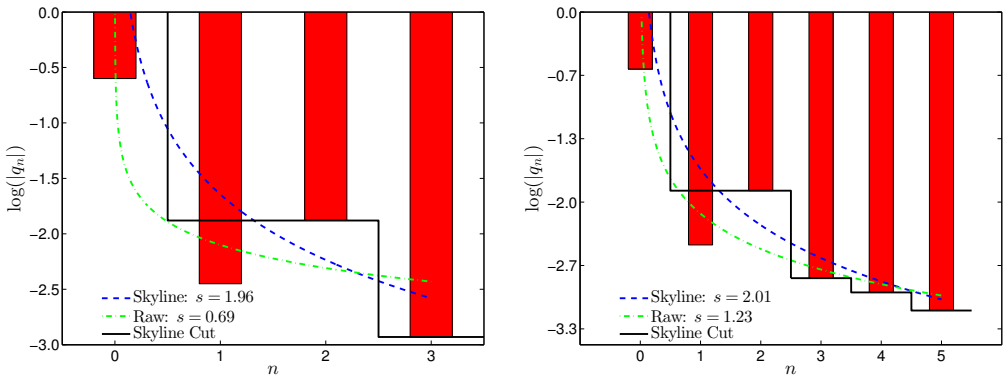
$$\log(|q_n|) \sim \log(c) - s\log(n) \tag{4.11}$$

The modal decay exponent, s and the coefficient, $\log(c)$ can be recovered from a least squares fitting. Analysis of Fourier mode decay reveals that $s \approx 1$ if solution is discontinuous, $s \approx 2$ if $\phi \in C^0$, $s \approx 3$ if $\phi \in C^1$ and so forth. However, in some circumstances, such as odd-even effect on modal coefficients, least squares fits mislead decay exponent. The skyline pessimization [98] recovers this issue by creating a new modal set, $\bar{q}_n$ such as

$$\bar{q}_n = \max_{i \in \{\min(n, N-1), \ldots, N\}} |q_i| \quad \text{for} \quad n \in \{1, 2, \ldots, N\} \tag{4.12}$$

In the new modal set, each modal coefficient is increased up to a higher-numbered coefficient, keeping the largest numbered coefficient larger than the second largest numbered one to eliminate non-monotone decay.

To show the performance of the detector, consider the signed distance function for the circle centered at origin with radius of 1.0. In this case, LS function is given as $\phi_0 = \sqrt{x^2 + y^2} - 1$ which is a non-differentiable C^0 function with a kink located at $(0, 0)$. In Fig. 4.2, the modal portrait, skyline cut procedure and fitted decay curves with and without skyline pessimization are investigated on the element including center point for $N = 3$ and $N = 5$. The expected decay exponent is 2 and it is approximated as 1.97 and 2.01 for $N = 3$ and $N = 5$, respectively. Even in the low order case where only 3 modal coefficients exist to approximate decay rate, the modal indicator accurately predicts the smoothness information for triangular elements.



(a) $N = 3$ (b) $N = 5$

Figure 4.2: Modal portrait and approximated modal decay profile for computed 1D expansion on an element having kink.

Having shown that modal decay fit with skyline pessimization is a robust and accurate smoothness indicator, we can construct the activation function, $\nu(s)$ which determines the magnitude of elemental artificial viscosity according the decay exponent. We prefer not to modify C^1 solutions similar to [140] and consider a smooth transition between C^1 and discontinuous solutions as follow,

$$\nu(s) = \nu_{\max} \begin{cases} 1 & s \in (-\infty, 1), \\ \frac{1}{2}(1 + \sin(-\frac{\pi(s-2)}{2})) & s \in [1, 3], \\ 0 & s \in (3, \infty). \end{cases} \tag{4.13}$$

The activation function returns the maximum viscosity, $\nu_{\max}$ if $s \leq 1$ and zero if $s \geq 3$. $\nu_{\max}$ should be chosen as a function of discretization resolution to localize discontinuity in an element length. Local characteristic velocity analysis reveals that $\nu_{\max} = \lambda_{\max} \frac{h}{N}$ propagates information the same distance as with the hyperbolic part of the equation. $\lambda_{\max}$ is the maximum characteristic speed of Eq. 4.1 and can be approximated as 1 although, it is slightly lower than 1 because of the regularization of the sign function.

The artificial viscosity is constant in each element, and possibly locally vanishing and discontinuous through the elements. The numerical solution of diffusion equations with discontinuous coefficients requires special attention especially with high order discretizations. To overcome this complexity, a diffusion equation is solved to smooth viscosity in time and space in [8] but this method is not suitable for explicit time integration and is computationally expensive. In [98], a post-processing technique is proposed to obtain globally continuous viscosity by using linear interpolation of vertex maximums. This approach may cause spreading of diffusivity to a stencil of the selected element and it is hard to implement on h nonconforming discretizations.

In this study, we use a direct approach to discretize the diffusion operator with discontinuous coefficients. The symmetric interior penalty method (SIP) with weighted averages, [54] is adapted to handle discontinuous and locally vanishing diffusivity directly. Introducing a new vector function, σ , diffusion operator can be re-written in the following mixed form,

$$\nabla \cdot (\nu \nabla \phi) = \nabla \cdot \sigma, \quad \sigma = \nu \nabla \phi \tag{4.14}$$

Then multiplying the mixed formulation of Eq. 4.9 with test functions, $v_h \in \mathcal{V}_N$ and $\psi_h \in \mathcal{V}_N^d$ and integrating by parts leads the following local form,

$$\begin{aligned}(\frac{\partial \phi_h}{\partial t}, v_h)_{D_k} + (\bar{H}(p^l_{\mathbf{x}}, p^r_{\mathbf{x}}), v_h)_{D_k} &= -(\sigma, \nabla v_h)_{D_k} + (\mathbf{n}\cdot\sigma^*, v)_{\partial D^k} \\ (\sigma, \psi)_{D_k} &= -(\nu\phi_h, \nabla\cdot\psi)_{D_k} + (\nu\phi^*, \mathbf{n}\cdot\psi)_{\partial D^k}\end{aligned} \tag{4.15}$$

Numerical fluxes that approximates the scalar and vector valued flux functions in the boundary contributions are given as,

$$\phi^* = \{\phi_h\}, \quad \sigma^* \ = \{\nu\nabla\phi_h\}_\omega - \tau[\phi_h] \tag{4.16}$$

with average and jumps operators of,

$$\{\phi\}_\omega = \omega^-\phi_h^- + \omega^+\phi_h^+, \quad [\phi] \ = \mathbf{n}^-\phi_h^- + \mathbf{n}^+\phi_h^+ \tag{4.17}$$

The viscosity weighted averages are introduced by defining double valued scalar function in internal faces,such as

$$\omega^- = \frac{\nu^+}{\nu^- + \nu^+} \quad \omega^+ = \frac{\nu^-}{\nu^- + \nu^+} \tag{4.18}$$

where weights satisfies, $\omega^- + \omega^+ = 1$ and $\omega^-, \omega^+ \geq 0$. Here, the $\nu^- + \nu^+ = 0$ case is not introduced because diffusion is only computed on the selected elements ($s < 3$) and on their first neighbors due to locality of artificial viscosity and compactness of discretization. By setting $\omega^- = \omega^+ = 1/2$, the regular average operators are recovered and subscripts are dropped. The penalty factor, τ plays an important role for the stability of the scheme and should be selected sufficiently large to enforce coercivity. We use the following definition of the penalty parameter,

$$\tau^k = \frac{2\nu^-\nu^+}{\nu^- + \nu^+} N(N+1) \frac{h_{\partial D^k}}{\min(h_{D^{k,-}}, h_{D^{k,+}})} \tag{4.19}$$

Here, $h_{\partial D_k}$ and h_{D_k} denote the surface area and volume of the element, respectively. On the non-conformal faces with hanging nodes, where more than two elements are connected, γ is computed using the collection of all adjacent elements which guarantees the coercivity [160]. Boundary conditions for the artificial diffusion equation are only needed on the troubled elements lying on the global boundary. In those cases, symmetry conditions are imposed by introducing exterior ghost states.

4.2.4 Mesh Adaptivity

Although, artificial diffusion is favorable in terms of the magnitude of dissipation to resolve kinks, it is obvious that using h adaptivity in the vicinity of troubled region is crucial to recover the accuracy. In the adaptive scheme used here, the computational mesh consists of elements in a range of predefined levels with l_0 denoting initial coarse level and l_M being the maximum level. Refinement and coarsening are performed dynamically during the solution. The level of refinement and the elemental dependencies are stored in a hierarchical tree for efficient and fast h type adaptivity.

Refinement is carried out in an isotropic way, i.e. a parent triangle is divided into four siblings by connecting the mid-edges. Elemental decay exponent, s is used as a threshold value to mark the elements for refinement. If $s \leqslant 3$, which indicates non-zero elemental viscosity, and refinement level of the element is smaller than the predefined maximum refinement level, then the element is marked for refinement and the approximation on the parent triangle is projected onto its four siblings.

For coarsening, elements are checked in two steps. In the first step, if $s > 3$ and refinement level of all four sibling elements are larger than the initial coarse level, these elements are marked for coarsening. The marked elements are temporarily combined by removing the center element and its three vertices. Solution of the sibling elements are projected onto the parent element. Then, a second check is performed on the temporarily combined element. If $s \leqslant 3$ for this element, we keep siblings untouched otherwise they are combined and level of refinement information is updated.

The DG method supports arbitrary number hanging nodes per face but it is restricted to decrease the computational complexity in flux evaluation. Adaptive mesh is $4:1$ and $2:1$ balanced for triangular and tetrahedral elements. In other words, a face of triangular and tetrahedral element can connect 4 and 3 elements at most, respectively. Any $1:1$ connection (conformal pair) is a member of $2:1$ balanced grid and so on as illustrated previously in Fig. 3.5.

4.2.5 Local Time Stepping

The spatial discretization of the pseudo time reinitialization equation with artificial diffusion is discussed in the previous sections. The semi-discrete form of the equation can be written in the following general form.

$$\frac{d\phi_k}{dt} = \mathcal{V}(\phi_k) + \sum_{n=1}^{N_f} \mathcal{S}^f(\phi_k^-, \phi_k^+) = L(\phi_k, t) \tag{4.20}$$

where $L(\phi, t)$ is the linear operator coming from the spatial discretization given in Eq. 4.15. Explicit time integration schemes are conditionally stable and time step size is limited by Courant-Friedrichs-Lewy (CFL) criterion as,

$$dt \approx \frac{1}{\lambda_{\max}\frac{N^2}{h_{\min}} + |\nu|_{L_\infty}\frac{N^4}{h_{\min}^2}} \tag{4.21}$$

For global time stepping, the smallest element length and approximation order determine the overall time step size. Explicit time integration schemes require to take into account an extension of imaginary axis of stability region responsible for convective scale, $h_{\min}/(\lambda_{\max}N^2)$. Also it requires an extension in negative real axis responsible for the diffusive scale, $h_{\min}^2/(|\nu|_{L^\infty}N^4)$ which leads to very small allowable time step size due to high order mesh dependent parameters when a standard global explicit integration is used.

In this section, we adopted the previously developed two rate multistep Adams-Bashforth (MRAB) scheme to reinitialization problem with artificial diffusion stabilization. Developed Adams-Bashforth (AB) method using different order base methods and substeps is modified to relax the time step restriction and to efficiently integrate this adaptive high-order DG level set reinitialization formulation. Theoretically, each element can be integrated with its time step size in a local time stepping technique but grouping the elements satisfying the local CFL condition in each group are required to obtain computationally efficient scheme.

In the MRAB scheme, elements are grouped according to their refinement levels if the maximum refinement depth is larger then zero. It is assumed that elements belonging to the same group have the same characteristic time scale and each

time scale differs by an integer ratio so that two groups can be synchronized in the largest time-step. Elements having smooth solution or in the initial coarse level form the slow group, G_s. All troubled or refined elements and their siblings form the fast group, G_f without checking the refinement levels. This grouping strategy is reasonable because in the adaptive scheme, troubled elements that artificial diffusion is required to stabilize, are in the highest level. The other refined elements having smooths solution include the very small proportion of the grid. So grouping those elements and increasing the rate number does not introduce any computational gain in our case. In Fig. 4.3, fast group that are marked for the stabilization, form the fast group and all other elements belong the slow group. The first slow neighbors of the fast group form the slow-fast buffer which need to be known for efficient implementation.

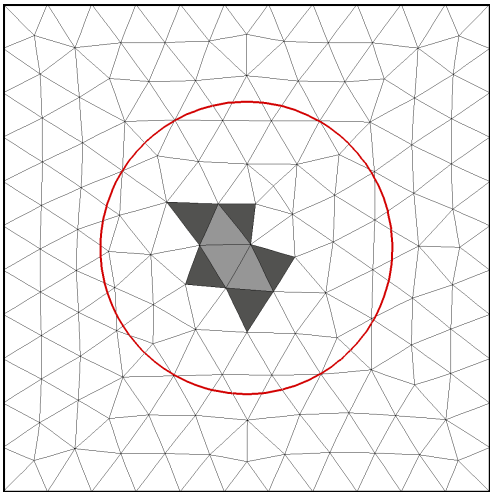


Figure 4.3: Multi-rate groups for the circular interface problem. Grey elements form the fast group, G_f and other elements form the slow group, G_s. Dark elements show the slow-fast buffer.

The features and coupling between the groups of the MRAB(l,m) scheme with base method of order l and sub-steps number of m is introduced in **??**. In the DG method, elements are weakly connected with their first neighbors by fluxes. Once an element evolves in time with a stable time-step size, it doesn't require additional information from the other time scales. In other words, coupling between the groups is also weak so that elements inside the groups but away from the slow-fast interface can be evolved with the base Adams-Bashforth integrator

without any special treatment. Previously adapted fastest first approach [67] is used in the computations so that fast elements away from the coupling interface can be advanced from time t^n to t^{n+m} in m substeps with the fast side time-step size, Δt_f as follow

$$\phi_f^{n+i} = \phi_f^{n+i-1} + \Delta t_f \sum_{j=1}^{l} \beta_j L(\phi_f^{n+i-j}) \quad \text{for} \quad i = 1 \cdots m \tag{4.22}$$

where β denotes the corresponding single rate AB coefficients. Similarly, pure slow components can be advanced to the same time level in a single step with the time step size of $m\Delta t_f$.

$$\phi_s^{n+m} = \phi_s^n + m\Delta t_f \sum_{j=1}^{l} \beta_j \left(L(\phi_s^{n-(j-1)m}) \right) \tag{4.23}$$

The complexity of the MRAB comes from communication between the bulk groups while satisfying the conservation and accuracy. Efficient coupling of the bulk group is explained in **??** . Only steady state solution of the reinitialization equation is needed, so that this coupling approach does not degrade the global accuracy. Fast elements at the transition region is evolved as follow,

$$\phi_f^{n+i} = \phi_f^{n+i-1} + \Delta t \sum_{j=1}^{l} \beta_j L(\phi_f^{n+i-j}, \phi_s^{n-(j-1)m}) \tag{4.24}$$

Slow-fast buffer is integrated in a single step by adding the known sub-level fast side flux contribution to the slow side.

$$\phi_s^{n+m} = \phi_s^n + \Delta t \sum_{j=1}^{l} \beta_j \left(\sum_{i=1}^{m} L(\phi_f^{n+i-j}, \phi_s^{n-(j-1)m}) \right) \tag{4.25}$$

It is obvious that the only extra effort compared with the single step AB scheme is the doubled evaluation of the flux function at the face located on the slow side of the buffer region so it is minimal. When the number of the sub-levels increase, elements in fast side and size of slow-fast buffer decrease, efficiency of the local time stepper increase. Number of sub-levels can be easily driven from the Eq. 4.21 such that $m = 1 + N \times 2^{l_M}$ by using $\lambda_{\max} = 1$ and $|\nu|_{L_\infty} = h \backslash N$ and assuming characteristic mesh size is reduced by half with each isotropic refinement level. AB methods are not self starting because they need $l + 1$

closest history. The initial values are provided with $(l+1)m$ small time-steps by Runge-Kutta methods which is one order higher than the base AB method to ensure that the temporal errors are only introduced by the MRAB scheme.

Finally, the efficiency of the local time stepper is estimated similar to **??** by assuming that the work load for each element, $\bar{L}_{D_k}$ is the same and dominates all the computations. Also, it is assumed that slow-fast buffer is totally integrated with the slow time scale although fluxes through the fast side are computed in the fast time scale. Let the number of elements in the fast and slow time scales are K_{G_f} and K_{G_s} respectively. Under these assumptions, workload, W of the global single rate AB and MRAB time integrators for one synchronization level can be given as

$$W_{AB} = mK\bar{L}_{D_k}, \quad W_{MRAB} = mK_{G_f}\bar{L}_{D_k} + K_{G_s}\bar{L}_{D_k} \tag{4.26}$$

and the theoretical speed up will be

$$S_{th} = \frac{W_{AB}}{W_{MRAB}} = \frac{m}{1+(m-1)K_{G_f}/K} \tag{4.27}$$

Again, this theoretical speedup calculation can be considered as an upper bound due to omitting additional workload such as residual update and buffer region computations. In the reinitialization scheme, only small portion of the grid form the fast group and they are located together leading small slow-fast buffer size so that efficiency of the local time stepping is expected to be high. At this point, locality of the mesh refinement is important to prevent the number of fast elements grow rapidly which is achieved with the non-conformal mesh adaptivity by letting the number of hanging nodes per face arbitrary.

4.3 Parallel Implementation

Weak element connectivity of the DG discretizations enable element-local computations resulting with locality of memory access. In addition, the high order approximation space used, increase the amount of computations per degree of freedom and hence, computational intensity. These features are all well-suited for the parallelization on many-core/multi-threaded architectures.

The developed method is coded in C++ and OCCA [120] kernel language. OCCA is a abstracted programming model used to encapsulate native languages for parallel devices such as CUDA, OpenCL, Pthreads and OpenMP. Therefore, OCCA allows customized implementations of algorithms for several computing devices with a single code and offers flexibility to choose hardware architectures and programming model at run-time.

In our parallel model, a work group computes the integrals of one element while a work item in a work group computes the contribution from each integration node in the kernels. There are three major computations required to solve the equations; volume and surface integration, and time-step update which are performed by Volume, Surface and Update kernels in solve and stabilization steps.

4.3.1 Volume Kernel

Volume integrals are computed in this kernel. $\mathcal{V}(\phi_k)$ is in the size of N_p on each element and can be written in the following generic form,

$$\mathcal{V}(Q) = \sum_{i=1}^{d} P_{r_i} \times cF_i \tag{4.28}$$

where P_{r_i} is the local projection operators of size of $N_p \times N_c$ defined on i'th direction of reference element. Projection operators are obtained pre multiplying local derivative matrices with inverse mass matrix and cubature weights to accelerate computations. cF_i denote the numerical volume terms defined on the cubature integration points.

First operation of kernel is to copy elemental field variables from global memory to shared memory such as ϕ_k vector. N_p work items are required for this operation. Then, stored shared memory variables are interpolated to the cubature integration points with matrix-vector multiplication all using the same interpolation matrix. Each work item multiples one row of cubature interpolation matrix with the vector to avoid memory conflicts. Resulting nodal values and all required geometric data, i.e. metric identities and Jacobian of local to global transformation, are stored on register memory for fast evaluation cF_i. N_c work

items are assigned for this operation. After this operation, volume flux terms are computed and stored on the shared memory vectors. N_c work items are used for this operation. Finally, volume terms are interpolated to interpolation nodes, i.e. $P_{r_i} \times cF_i$ operation which involves d matrix-vector multiplication performed similarly previous one to prevent the memory conflicts. N_p work item is used for this operation. Number of required work items change within the kernel to perform individual tasks. To accommodate the number for work items required for the all computations, kernel request $\max(N_c, N_p)$ work items.

4.3.2 Surface Kernel

Surface contribution is computed in this kernel. $\mathcal{S}(\phi_k^-, \phi_k^+)$ is vector of length N_p for each field on element. Surface integral term can be written in the following abstract form,

$$\mathcal{S}(\phi_k^-, \phi_k^+) = P_g \times F^g(\phi_k^-, \phi_k^+) \tag{4.29}$$

where P_g is the local projection operator of $N_p \times (N_f \times N_g)$ which is pre multiplied with inverse mass matrix and surface cubature weights. $F^g(\phi_k^-, \phi_k^+)$ represent the flux function evaluated at surface cubature points.

First operation of the surface kernel storing elemental field variables on shared memory. For internal degrees of freedoms, e.g. ϕ_k^-, a vector of size N_p is used for each variable. External field variables, e.g. ϕ_k^+ are stored on the shared vector in size of $N_f \times N_p$ where N_f is the number of connection for the work group. For conformal discretization N_f denotes the number of faces i.e. 3 and 4 for triangle and tetrahedron. N_p work item is used for this operation.

Second operation is to interpolate the standard nodal values to surface cubature points. This operation includes multiplication pre stored shared vectors with the cubature interpolation matrix defined on the reference element. Similar to volume kernel, each work item multiplies one row of interpolation matrix and the vectors. Interpolated values and geometric data are stored on the register memory. Then, flux function at surface cubature nodes, $F^g(\phi_k^-, \phi_k^+)$ is computed using previously obtained register memory values. Flux function is stored on the

shared memory vector of size $N_f \times N_g$ which is equal to assigned number of work items for this step.

Finally, flux function is lifted to the interpolation nodes. This step involves the matrix - vector multiplication i.e. $P_g \times F^g(\phi_k^-, \phi_k^+)$. N_p work item is assigned for this operation. To accommodate all different tasks, surface kernel request $\max(N_f \times N_g, N_p)$ work items.

4.3.3 Update Kernel

Update kernel computes local time integration step which involves the global vector operations using computed right hand side vectors and required level of history depending the order of integration method. N_p work item per element is requested by the kernel.

Performance of the kernels are highly dependent to the hardware, memory usage, tuning parameters etc. Here, we only use coalescing and unrolling which are very basic ways to improve the performance of kernels. Please note that these are not the only tuning strategies and many other method can be used such as multiple elements per work group, hardware dependent padding etc. which require an optimization study and will not be covered here.

4.4 Numerical Tests

Reinitialization tests are solved for both smooth and non-smooth interface problems. To evaluate the accuracy of the numerical scheme, the L_1 and L_∞ norms are determined according to the following relation,

$$L_1 = \frac{1}{N_T} \sum_{n=n_1 \cdots n_{N_T}} h_n |\phi_n - \phi_n^e|, \quad L_\infty = \max_{n=n_1 \cdots n_{N_T}} (h_n |\phi_n - \phi_n^e|) \tag{4.30}$$

where h_n represent the characteristic length of the element that node, n belongs and N_T denotes total number of nodes in the computational grid or in the predefined band thickness, ϵ depending on the location where the error is computed. ϕ^e denotes the exact solution or very accurate approximation of the exact solu-

tion if it is not explicitly known. Although local time stepper is formulated for the arbitrary orders, we used second order base methods if otherwise stated in the computations.

4.4.1 Circle

In the first test case, the reinitialization problem proposed by [178] is solved. The interface of interest is a circle centered at the origin with radius of 1.0. The signed distance function in a computational domain of $[-2, 2]^2$ is perturbed into the following initial level set function

$$\phi_0 = ((x_1 - 1)^2 + (x_2 - 1)^2 + 0.1)(\sqrt{x_1^2 + x_2^2} - 1) \tag{4.31}$$

The initial LS has a smooth interface with widely changing gradients which creates flat and steep regions in the vicinity of the interface. Fig. 4.4 shows the measured order of accuracy for $N = 3, 4, 5$ using L_∞ and L_1 error estimates. In panel (a), errors are measured near the interface with condition of $\epsilon = 0.1$ such that kink point at $(0, 0)$ is excluded. Panels (b) and (c) shows globally measured L_∞ errors including the kink region. Computations are performed on the sequence of meshes by dividing the initial coarse level grid having characteristic length of $h = 0.4$ and element number of $K = 226$ in panels (a) and (b). Locally adapted grid with different refinement depths are used for the errors in panel (c). Optimal $N+1$ convergence rate is obtained near the smooth interface region while second order global accuracy is obtained due to the kink point, as expected. Comparison of (b) and (c) reveals that non-uniform local and uniform global refinements give very close errors and convergence rates indicating the efficiency of local refinement strategy.

As mentioned in Sec. 4.2.1, the regularization of the signum term has an impact on the accuracy and convergence of the scheme. To test the behavior of regularization, $\text{sgn}_\alpha(\phi_0) = \phi_0/\sqrt{\phi_0^2 + (|\nabla\phi_0|h)^2}$ [138] is compared with $\text{sgn}_\alpha(\phi_0) = \phi_0/\sqrt{\phi_0^2 + |\nabla\phi_0|h}$ [26]. Also, our selection of the normalized hyperbolic tangent function, $\text{sgn}_\alpha(\phi_0) = \tanh(\frac{\pi\phi_0}{2|\nabla\phi_0|h})$, which is also C_∞ continuous over the domain, is used. Fig. 4.5 shows the local L_1 error decay for these regulated signum

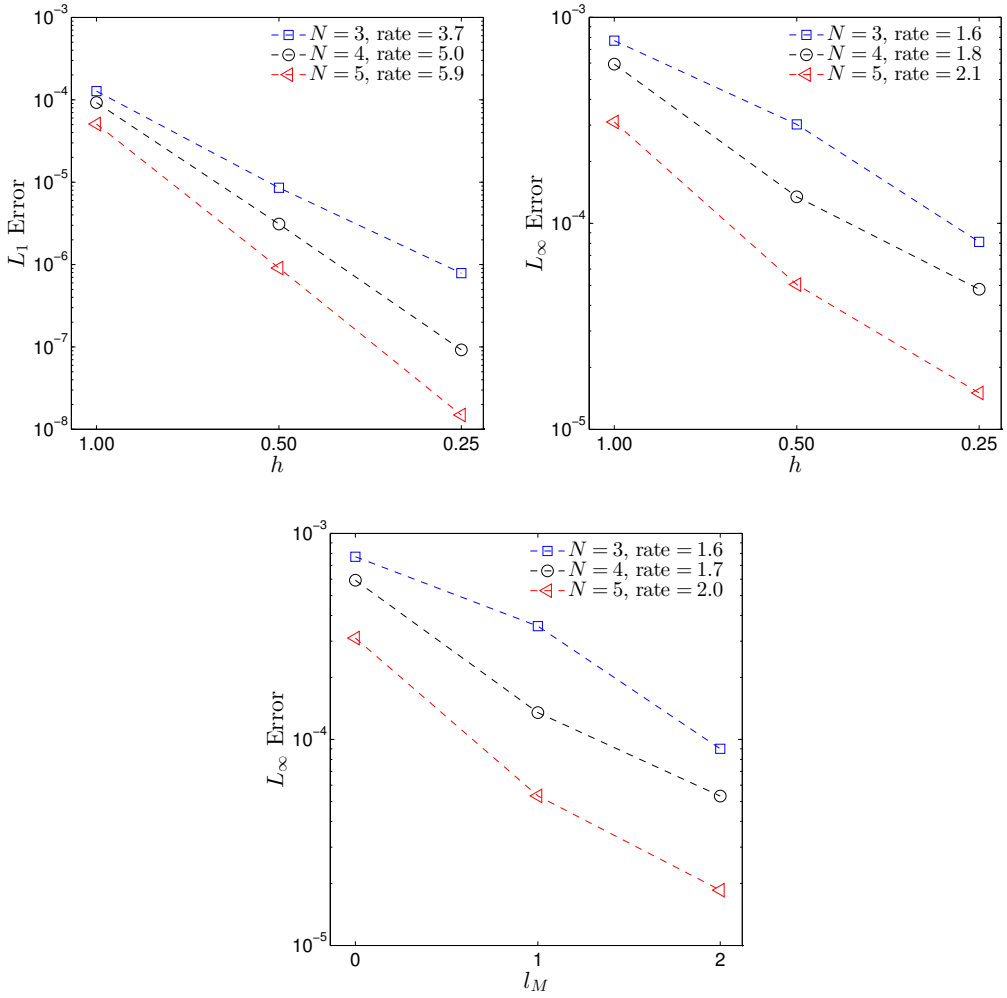


Figure 4.4: Accuracy of the scheme at constructing signed distance function for smooth circle test.

terms. The hyperbolic tangent function is sharp and leads to fast convergence but gives slightly larger errors in steady state. For the regularization given in [26], more accurate results can be obtained by sacrificing the speed substantially. The regularization proposed in [138] offers a mid point between accuracy and convergence speed. Hyperbolic tangent function gives very fast convergence and used rest of the papers.

Fig. 4.6 illustrates the time evolution of the level set function for $l_M = 2$ dynamic grid starting from the initial mesh with characteristic length of h and $N = 5$. A good recovery of the signed distance function is obtained for highly disturbed initial data and relatively coarse grid. Note that, proposed method does not cause the artificial movement of the interface and that its location

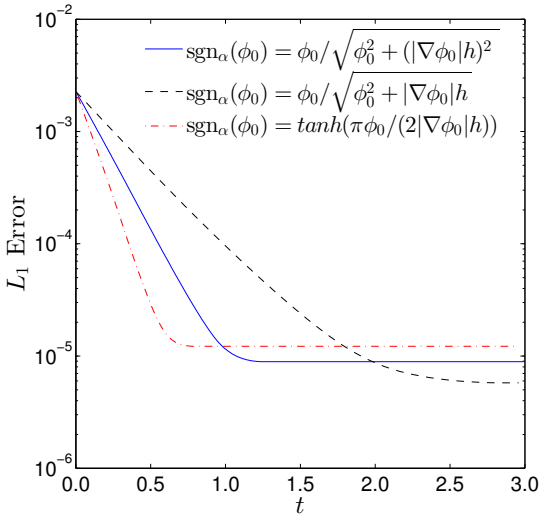


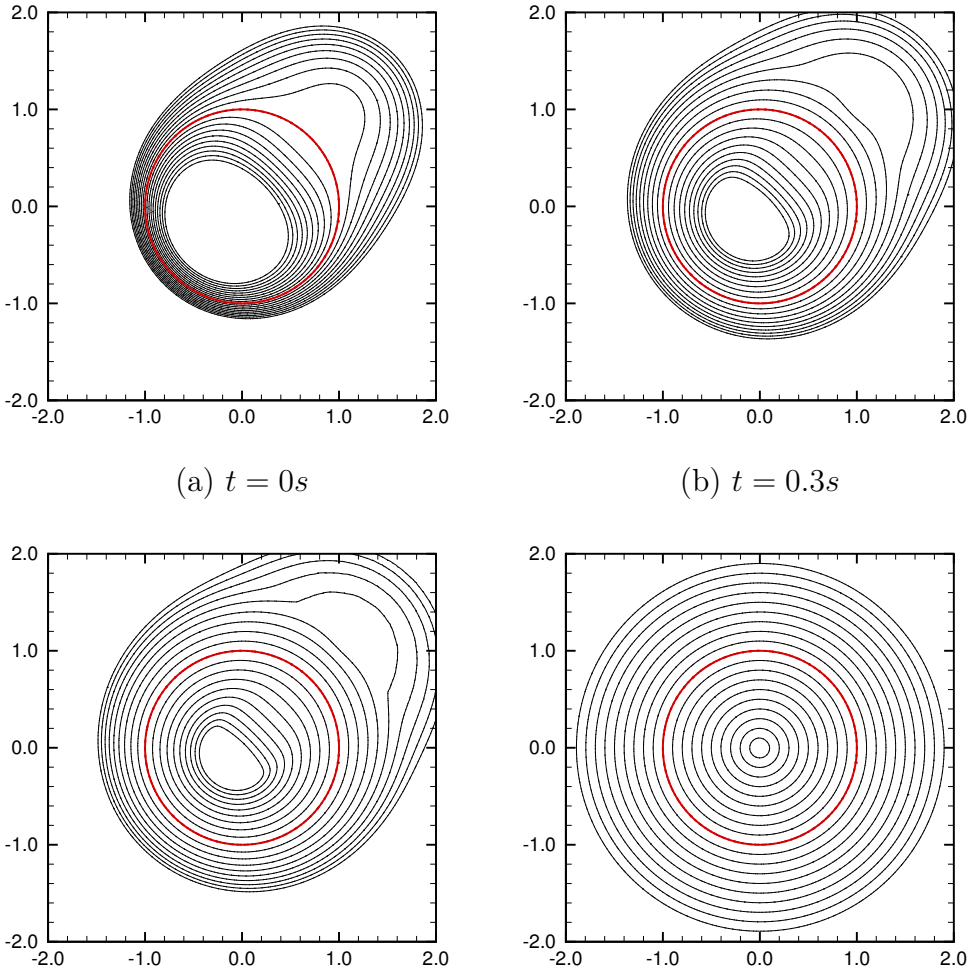
Figure 4.5: Speed of local L_1 error decay for different regulated signum terms at grid h and $N = 3$.

remains unchanged after reinitialization.

To study the speedup, initial mesh configurations with the characteristic element lengths of h is used. Number of elements in each group, size of substeps and distribution of elements, i.e. size of the slow-fast interface are expected to effect predicted speedups which changes with the depth of refinement. For the calculation of theoretical speedup, time average of the fast group percentage is used in adaptive studies. Numerical experiments shown in Table 4.1, reach the theoretical values computed according to the Eq. 4.27 for both $N = 3$ and $N = 5$. This is because all the fast elements are located together leading small slow-fast buffer size where its workload contribution is neglected in the theoretical speedup calculations.

Table 4.1: Speedups for circle test case for different refinement levels and order of approximations.

	$l_M = 0$			$l_M = 1$			$l_M = 2$		
N	S	S_{th}	$\%K_{G_f}$	S	S_{th}	$\%K_{G_f}$	S	S_{th}	$\%K_{G_f}$
3	3.85	3.89	0.89	5.29	5.36	5.11	6.48	6.54	8.23
5	5.72	5.75	0.89	7.23	7.28	5.11	7.89	7.94	8.23



(a) $t = 0s$ (b) $t = 0.3s$

(c) $t = 0.6s$ (d) $t = 0.9s$

Figure 4.6: Reinitialization of level set function for circle test for grid $h/2$ and $N = 5$. Drawn are contour levels from -0.9 to 0.9 with step size 0.1.

4.4.2 Ellipse

The signed distance function is computed from the reinitialization of ellipse starting with the following initial LS function,

$$\phi_0 = ((x_1 - x_{1,0})^2 + (x_2 - x_{2,0})^2 + 0.1)(\sqrt{\frac{x_1^2}{A^2} + \frac{x_2^2}{B^2}} - 1) \tag{4.32}$$

with $A = 1$,$B = 0.5$, $x_{1,0} = 0.875$ and $x_{2,0} = 0$. Similar to the circle test, initial LS has both small and large gradients near the interface with extended kink region between the line segment $(-A, A)$ on the x_1 axis. The closed form of the exact signed distance function is not known for the ellipse so it is accu-

rately approximated by creating a finite number of points on the interface with coordinates,

$$x_{1,p} = A\cos(2\pi p/N_p) \quad \text{and} \quad x_{2,p} = B\sin(2\pi p/N_p) \tag{4.33}$$

where p and N_p are the index and total number of the points inserted on the interface respectively. Then, the approximated distance function is defined as

$$\phi^e(x_{1,i}, x_{2,i}) = \min(\sqrt{(x_{1,i} - x_{1,p})^2 + (x_{2,i} - x_{2,p})^2})\text{sgn}(\phi_0(x_{1,i}, x_{2,i})) \tag{4.34}$$

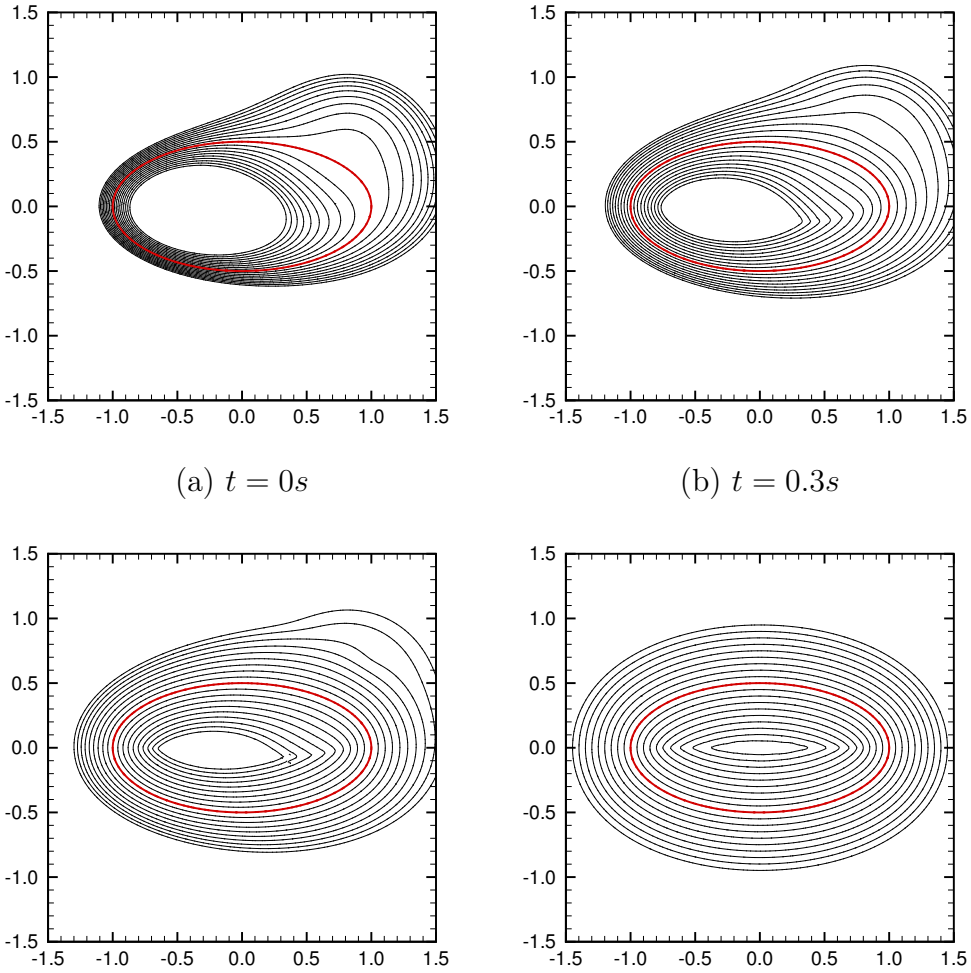
The same initial mesh configuration with the circle test case is used. Fig. 4.7 illustrates reinitialization of highly perturbed LS function at different solution times for $h/2$ grid and $N = 5$. As seen in the figure, the signed distance function is recovered from the initial data at both smooth and non-smooth regions. Also the position of the interface is well preserved leading to the small L_∞ errors near the interface.

Fig. 4.8 shows the marked troubled elements, the mesh structure and LS contours at the final time of $1.0s$. The irregularity detector successfully picks elements and adaptivity substantially improves the solution near the kink region. Away from the kink, the scheme achieves full convergence rate similar to the circle test. Also, pollution caused by the artificial diffusion does not spread out and degrade the solution outside of the kink.

Similar to circle problem, initial mesh of $h = 0.4$ is used for the speedup tests. Numerical experiments confirm the theoretical upper bound but differences are slightly larger than the circle test. In this case, fast and slow groups are not as compact as the previous problem which leads comparatively high fast-slow buffer region resulting slight deviation from the theoretical speedups as given in Table 4.2.

Table 4.2: Speedups for ellipse test case for different refinement levels and order of approximations.

	$l_M = 0$			$l_M = 1$			$l_M = 2$		
N	S	S_{th}	$\%K_{G_f}$	S	S_{th}	$\%K_{G_f}$	S	S_{th}	$\%K_{G_f}$
3	3.29	3.41	5.75	3.85	3.96	12.74	3.34	3.47	22.87
5	4.64	4.74	5.31	4.75	4.84	12.74	3.68	3.77	22.87



(a) $t = 0s$ (b) $t = 0.3s$

(c) $t = 0.6s$ (d) $t = 0.9s$

Figure 4.7: Reinitialization of LS function for ellipse at grid $h/2$ and $N = 5$. Drawn are contour levels from -0.45 to 0.45 with step size 0.05. Only part of the domain is shown.

4.4.3 Intersecting Circles

In this test, reinitialization of two intersecting circles of radii r centered at $(\pm a, 0)$ and $0 < a < r$ is considered. Because $0 < a < r$ holds, the circles intersect and the interface of interest is the union of the two circles. The signed distance function to the interface is given as,

$$d(x_1, x_2) = \begin{cases} \min(\sqrt{x_1^2 + (x_2 \pm \sqrt{r^2 - a^2})^2}) & \text{if } \frac{a - x_1}{\sqrt{(a-x_1)^2 + x_2^2}} \geqslant \frac{a}{r} \text{ and } \frac{a + x_1}{\sqrt{(a+x_1)^2 + x_2^2}} \geqslant \frac{a}{r} \\ \min(\sqrt{(x_1 \pm a)^2 + x_2^2} - r) & \text{else} \end{cases} \tag{4.35}$$

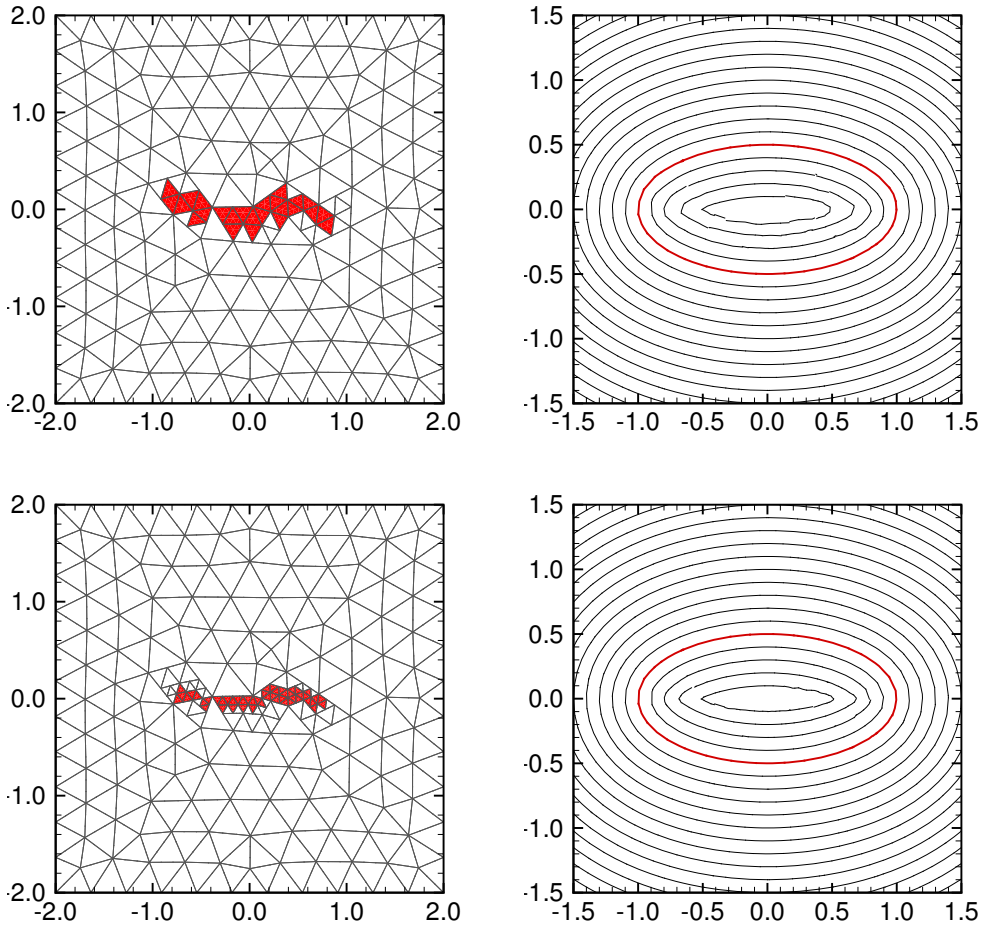


Figure 4.8: Marked elements and LS contours of the ellipse test case at the final time of $1.0s$ for different refinement levels, grid h and $N = 3$. In second column, only part of the domain is shown.

This test is more critical because the signed distance function has kinks on the whole x_2 axis and line segment $[-a, a]$ on the x_1 axis. The problem is solved for $r = 1$ and $a = 0.7$ on a computational domain of $[-2, 2]^2$. Similar to the previous tests, the initial LS function is defined by multiplying the signed distance with a perturbation function to create highly varying gradients near the interface as follows,

$$\phi_0 = ((x_1 - 1)^2 + (x_2 - 1)^2 + 0.1)d(x_1, x_2) \tag{4.36}$$

Adaptive mesh structure, activated elements and LS contours are shown in Fig. 4.9 for $N = 3$ and $h = 0.2$. Good recovery of the signed distance function from the perturbed initial LS data is obtained. Local refinement narrows

the kink region and improves the solution substantially.

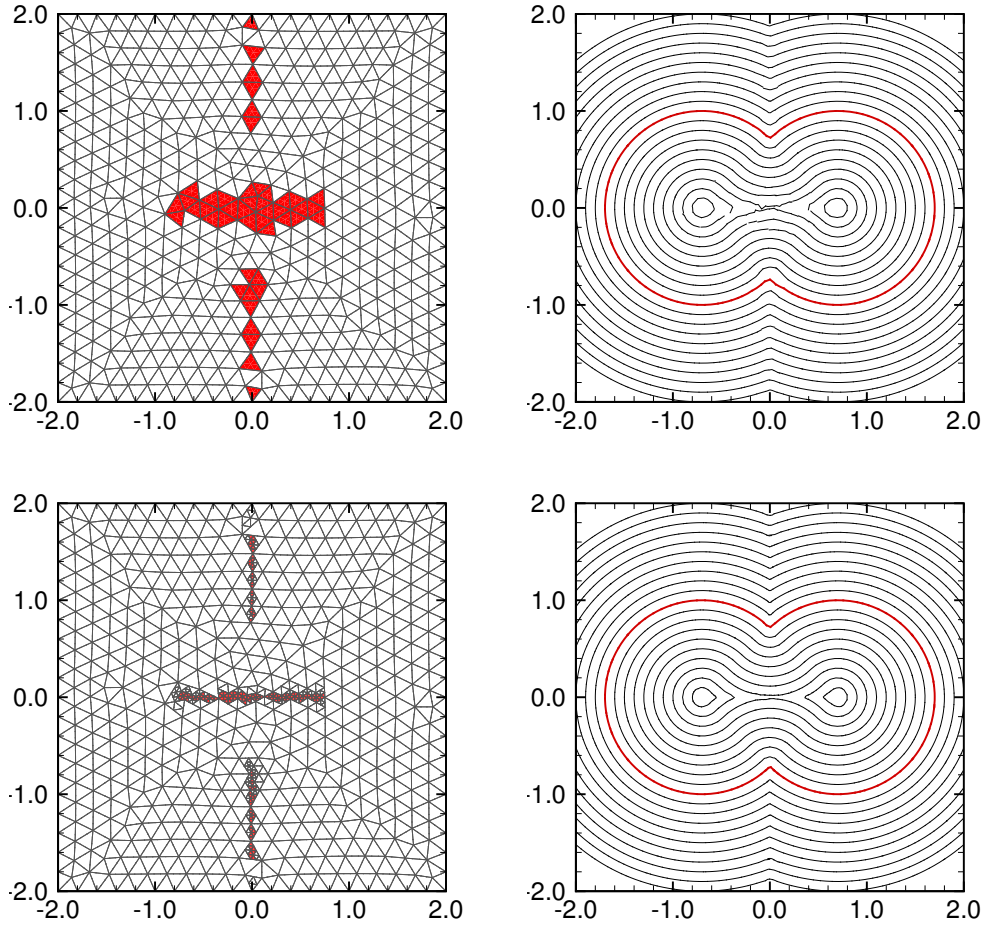


Figure 4.9: Marked elements and LS contours at different refinement levels for intersecting circle test, $h = 0.2$ and $N = 3$. Contours levels are drawn from -0.9 to 1.0 with step size 0.1. In third row, only part of the domain is shown.

4.4.4 Square

Constructing signed distance function is considered for a square inside the computational domain of $[-2, 2]^2$ starting with following initial LS function,

$$\phi_0(x_1, x_2) = \max(|x_1 - x_{1,c}| - w/2, |x_2 - x_{2,c}| - w/2) \tag{4.37}$$

For $w = 2$, $x_{1,c} = x_{2,c} = 0$, initial LS creates concentric squares centered at the origin with zero contour level of width 2. Obviously, ϕ_0 is not a signed

distance function and includes kinks along the diagonals of the domain. Exact distance function is approximated similar to the ellipse test and has kinks at the diagonals but only for $\phi \leq 0$. The LS function is reinitialized well for smooth and non-smooth regions with sharp corners increasing the local refinement level as illustrated in Fig. 4.10.

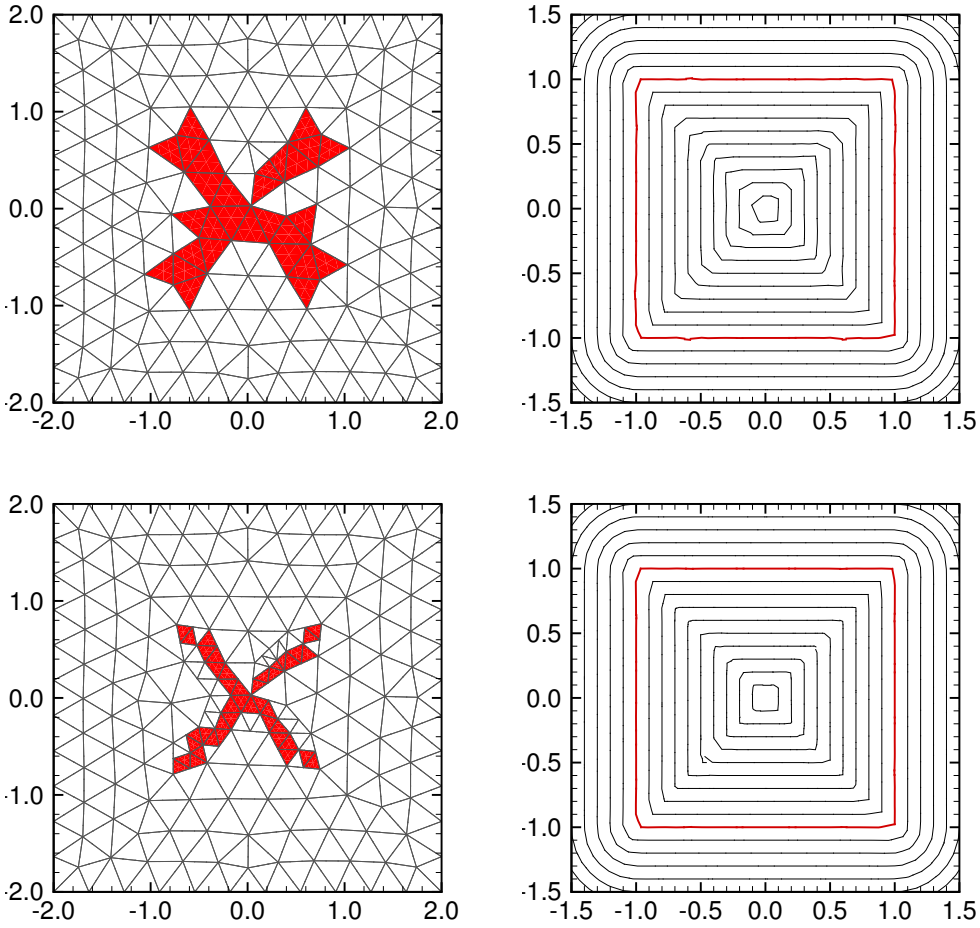


Figure 4.10: Marked elements and LS contours at different refinement levels for square test, $h = 0.4$ and $N = 3$. Contours levels are drawn from -0.9 to 1.0 with step size 0.1. In second column, only part of the domain is shown.

Parallel performance of kernels on different architectures and multi-threading models are shown in Fig. 4.11 in terms of achieved GFLOPs of HJ volume and surface kernels on NVIDIA Tesla C2075 GPU when cross-compiled with OpenCL and CUDA and on Intel Xeon E5-2670 CPU when compiled with OpenMP. Similar performances are obtained for the stabilization volume and surface kernels

and not included. Figure also illustrates the speedup of each model relative to serial CPU implementation. All performance numbers are obtained using the wall clock time from the beginning of one time step to the next one and averaged over a few hundred samples to minimize the timing transients. Similar performance between CUDA and OpenCL models are observed for both surface and volume kernels on GPU. GPU outperforms the CPU in all polynomial orders by a factor ranging from 25 to 125 and reach almost peak performance of the hardware.

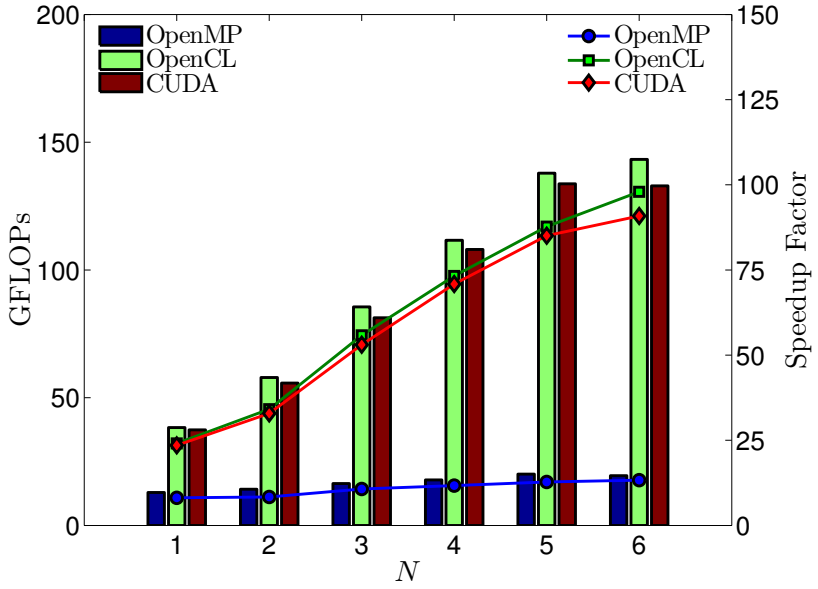
4.4.5 3D Smooth Interface

Circular interface problem is extended to three dimension by considering the following initial level set function in the domain of $[-2, 2]^3$,

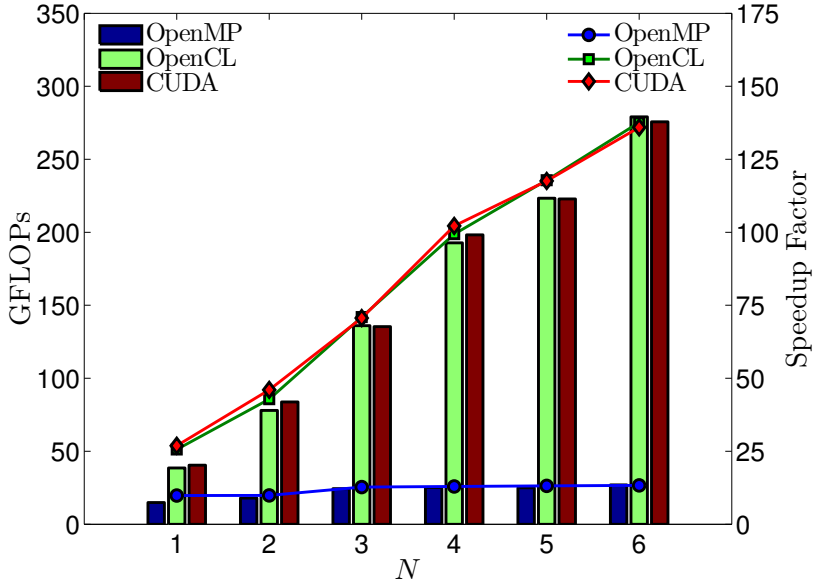
$$\phi_0 = ((x_1 - 1)^2 + (x_2 - 1)^2 + (x_3 - 1)^2 + 0.1)(\sqrt{x_1^2 + x_2^2 + x_3^2} - 1) \qquad (4.38)$$

which gives a highly perturbed spherical interface. Fig. 4.12 shows the initial and final level set functions for $N = 3$ on the domain part of $z < 0$. Initial grid characteristic length is $h = 0.4$ where $K \approx 7000$. One level refinement depth is used to recover accuracy near the kink point i.e. center of the sphere. Similar to the 2D test cases, good recovery of the signed distance function is obtained.

Performance of HJ volume and surface on number of single precision floating point operations are illustrated in Fig. 4.13 using different multi-threading models and the same hardware with the previous test case. Speedups factors are computed according to the serial CPU implementation. Similar performances are obtained for CUDA and OpenCL implementation on GPU. GPU outperforms CPU in all cases with speedup factors ranging between 20-80 for volume kernel and 50-150 for surface kernel. Higher speedups are obtained at higher order approximation due to increasing computational intensity. OpenCL seems to be more efficient on GPU except $N = 4$ for volume kernel and $N = 5$ for surface kernel where CUDA slightly outperforms OpenCL on NVIDIA GPU when the number of work items per work group are multiplicity of 8. So that



(a) Volume Kernel

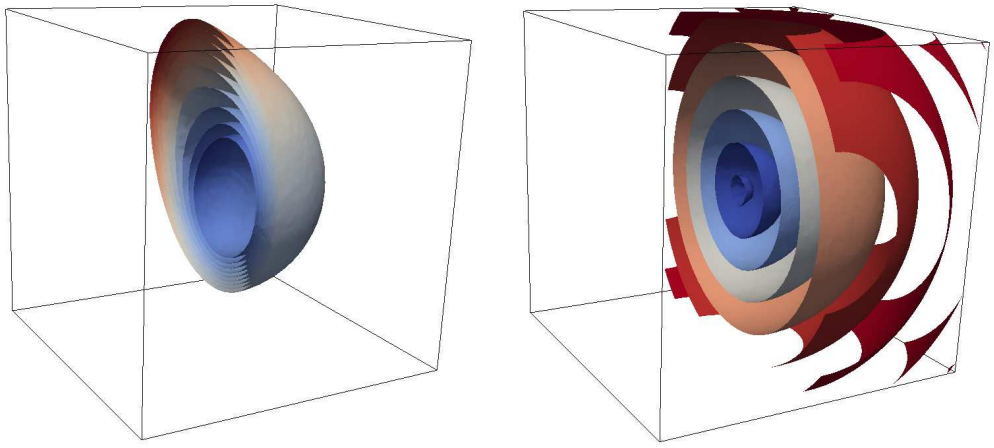


(b) Surface Kernel

Figure 4.11: Single precision GFLOPs and speedups of 2D volume and surface kernels vs polynomial order on CPU and GPU using different multi-threading models.

padding the number of work items with the factor of 8 could certainly improve the performances at all orders and multi-threading models.

Table. 4.3 represents the global performance of the solver using different models. Wall clock times are computed for $h = 0.4$, $(K \approx 7000)$, $l_M = 1$ and 1 s total analysis time where signed distance function is obtained on the whole domain. In



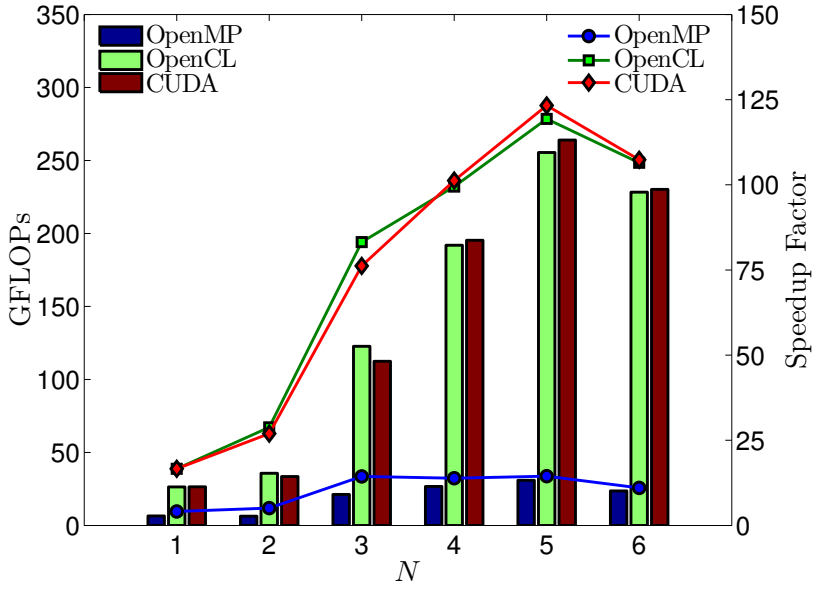
(a) $t = 0$ (b) $t = 1.0$

Figure 4.12: Reinitialization of level set function for spherical interface test. Drawn are contour levels from -0.8 to 2.4 with step size of 0.4. ($h = 0.4$, $l_M = 1$, $N = 3$)

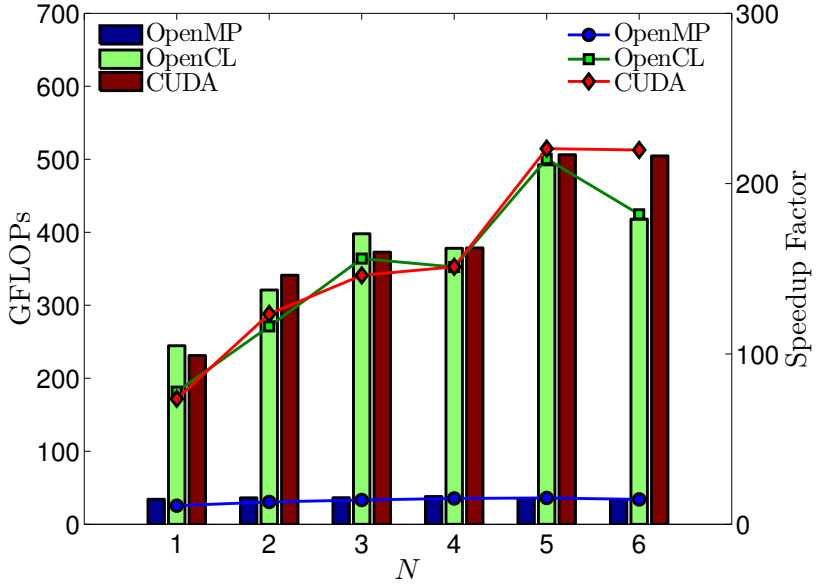
OpenMP model, 16 threads are used with the hyper-threading capacities of the 8-core processor. In all cases, percentage of stabilization time is low compared to the actual computation time indicating the efficiency of stabilization technique used. Update stage has a negligible cost thanks to performance of local time stepper which almost does not require any extra effort comparing single-rate counterpart. Although, OpenCL is slightly more efficient, similar performance of OpenCL and CUDA on GPU can be seen in overall simulation time. Also, percentage of stabilization time in GPU is higher then the CPU implementations because of the data transform between GPU and CPU in adaptation step.

Table 4.3: Overall timings for the solver on different multi-threading models and approximation orders.

				Solve		Stabilize		Update	
Model	N	Speedup	Total	Time (s)	%	Time (s)	%	Time (s)	%
CUDA (GPU)	3	204	2.00E0	1.76E0	87.96	2.35E-1	11.76	5.60E-3	0.28
OpenCL (GPU)		207	1.97E0	1.70E0	86.13	2.61E-1	13.26	1.21E-2	0.61
OpenMP (CPU)		15	2.70E1	2.45E1	90.79	2.34E0	8.69	1.41E-1	0.52
Serial (CPU)		-	4.08E2	3.70E2	90.73	3.66E1	8.96	1.24E0	0.30
CUDA (GPU)	5	218	3.81E1	3.61E1	94.96	1.90E0	4.99	1.73E-2	0.05
OpenCL (GPU)		210	3.95E1	3.77E1	95.47	1.75E0	4.44	3.53E-2	0.09
OpenMP (CPU)		15	5.46E2	5.26E2	96.19	1.97E1	3.60	1.12E0	0.21
Serial (CPU)		-	8.30E3	7.98E3	96.12	3.07E2	3.71	1.39E1	0.17



(a) Volume Kernel



(b) Surface Kernel

Figure 4.13: Single precision GFLOPs and speedups of 3D volume and surface kernels vs polynomial order on CPU and GPU using different multi-threading models.

4.5 Conclusion

A GPU accelerated discontinuous Galerkin algorithm on unstructured adaptive meshes is presented for high-order solution of two and three dimensional level set reinitialization. An artificial diffusion approach with a reliable irregularity

detector is used to stabilize the system. The stabilization mechanism does not reduce accuracy to first-order directly, does not prevent spreading of diffusion over the elements and is highly scalable in multi-threading architectures. The presented method is further accelerated with a multi-rate Adams-Bashforth time stepping to prevent severe time step sizes resulting from the explicit treatment of artificial diffusion stabilization. Accuracy of the solution is recovered by using local non-conformal adaptivity in troubled regions. Platform independence of the solver is achieved with an extensible multi-threading programming API as common kernel language. The developed solver is highly scalable on many-core architectures and considerable speedups factors are obtained comparing the serial CPU implementation.

CHAPTER 5

A DISCONTINUOUS GALERKIN LEVEL SET METHOD FOR INCOMPRESSIBLE MULTIPHASE FLOWS

This chapter focuses on the development of a high-order discontinuous Galerkin method for the solution of unsteady incompressible multiphase flows with level set interface formulation. Nodal discontinuous Galerkin discretization is used for incompressible Navier-Stokes, level set advection and reinitialization equations on adaptive unstructured elements. Implicit systems arising from the semi-explicit time discretization of the flow equations are solved with a matrix-free p-multigrid preconditioned conjugate gradient method which minimizes the memory requirements and increases overall runtime performance. Computation is localized mostly near the interface location in the adaptive method to reduce computational cost without sacrificing the accuracy. Efficiency, local high-order accuracy and mass conservation of the method are confirmed through distinct numerical test cases of sloshing, dam break and Rayleigh-Taylor instability.

5.1 Introduction

Multiphase flows occur in many areas of practical importance such as flow-structure interaction, air-water interfacial dynamics, phase change problems, reacting flows etc. Numerical prediction of the deformable interfaces in these complex flows are challenging due to discontinuity in the material properties, varying topology and range of scales.

Numerous numerical methods are proposed to represent the phase dynamics in incompressible flows. These methods can be classified as interface tracking and interface capturing. The former group are Lagrangian or semi-Lagrangian, where the mesh explicitly represents the interface [190] or particles define the interface by their locations (MAC) [76]. Interface tracking approaches are generally accurate and robust but difficult to use when the interface encounters topological changes. Interface capturing methods, Volume of Fluid (VOF) [83] and level set (LS) [133], are Eulerian i.e. an implicit function defined on a fixed grid represents the interface. Specifically, VOF methods are widely used in multiphase flow simulations due to its natural conservation properties and efficiency, but reconstruction of the interface from the volume fraction data and obtaining geometry dependent properties, such as interface normal and curvature, are difficult to compute.

LS method addresses the problems of interface tracking and VOF methods. In the LS method, an interface between the phases is represented as the zero contour of a continuous function which is positive in one region and negative in the other. This implicit representation of the interface offers many advantages such as straightforward extension from 2D to 3D, simple handling of topological changes and easy calculation of geometric properties. The main difficulty with use of the LS method is the need to control mass loss present in the method. Various techniques have been proposed to improve the conservation properties of the original method. Popular ones are combining LS method with VOF method [180, 174, 193, 108, 200] or with semi-Lagrangian particle method [56, 57, 109, 66, 100] and applying some additional mass correction procedures [177, 192]. Common to all these approaches is the fact that simplicity and computational efficiency of the original LS method are partly lost. Another modified approach is the Conservative Level Set (CLS) method [131, 132] based on replacing the signed distance function of the original LS formulation with sharp hyperbolic function and altering the equations appropriately. Although mass conservation can be improved with CLS method, geometric properties may not be computed as accurate as the LS method due to sensitivity of the normal calculation to small spurious errors [205]. In fact, the mass loss problem with the LS method

is more inherent in the discretization scheme used than the formulation itself [117].

The discontinuous Galerkin (DG) methods ([82] and references therein) are a class of finite element methods that make use of completely discontinuous spatial discretization. The DG methods have excellent dissipation properties achieved by local high-order polynomial approximations to overcome mass loss problem of the LS formulation. For this reason, DG for LS advection gives more accurate results compared with the widely used essentially non-oscillatory (ENO) type finite difference schemes [179]. DG level set modeling is also applied in incompressible multiphase flow simulations i.e. it is combined with stabilized finite element [118, 116] and conservative finite difference [136] flow solvers. These hybrid methods require interpolation of the velocity field obtained from the lower order flow solver to higher order discontinuous polynomial space and back interpolation of LS function to flow solver grid in each time step. Fine grids are needed for the lower-order flow solver which may became computationally inefficient when high-order interpolation orders are used for LS advection.

Multiphase flows are generally highly dynamic in such a way that location and topology of the interface change considerably during the simulation. Resolution should be increased in the whole domain, which creates unnecessary burden in computational time to capture the full physics of the interface. Adaptive mesh refinement (AMR) techniques decrease the computational effort by increasing the resolution in the vicinity of the interface with keeping coarse grid at less dynamic regions. There are some adaptive LS methods for multiphase flows based on the low order finite difference/volume schemes on structured Cartesian grid and tree type data structures [176, 175, 112]. Unstructured anisotropic mesh refinement is also coupled with the low order finite element flow - high order DG interface modeling [118]. These types of refinement strategies are based on the conformal discretization that each face can be shared by two elements. AMR algorithm should handle the complicated interpolation between grids and mesh transition to ensure the conformity [102]. Global re-meshing or iteratively optimized local modifications may become inefficient when frequent adaptation is needed as in the multiphase flow simulations. Due to relaxed strong elemental

connectivity, DG methods can be easily used on unstructured elements with hanging nodes [59, 102] enabling the fast adaptation with accurate interpolation between grids.

In this chapter, we present a high order fully discontinuous Galerkin method for immiscible, incompressible multiphase flows on adaptive unstructured grids. In the dynamically adapted grid, no restriction introduced on the number of hanging nodes per face to avoid complicated remeshing and interpolation steps. Computational complexity coming from the flux evaluation on non-conformal faces are handled with precomputed local operators according to the predefined maximum refinement level. Multiphase flow equations are discretized in time with a semi-implicit splitting scheme which enable to use equal order approximation for velocity and pressure. Divergence free constraint is explicitly enforced by interpolating the intermediate velocity to exact divergence free polynomial space. Implicit systems arising from the splitting scheme are solved in matrix-free form with efficient, matrix-free p-multigrid preconditioner to reduce the memory requirements and costly assembly procedures. Time evolution of interface is achieved with discontinuous Galerkin local level set method having the same other with flow solver that avoids the interpolation of solution fields to different polynomial spaces in each time step. To reinitialize the distorted level set function to signed distance function, a stabilized, local scheme based on the local discontinuous Galerkin method is proposed. To the best knowledge of authors, this is the first adaptive, fully discontinuous Galerkin method for multiphase immiscible flows. The proposed method allows to capture interface topology accurately in simulating wide range of flow regimes with high density/viscosity ratios and offers good mass conservation even in relatively coarse grids while keeping the simplicity of the level set interface modeling.

The outline of this chapter is as follows: the governing equations of the problem and notation used in the numerical scheme are described in Sec. 5.2. Then, time and high order discontinuous Galerkin spatial discretizations and matrix-free solution techniques are presented. At the end of the section, local interface model and reinitialization of the level set function are described. Sec. 5.2.4 is dedicated to adaptive mesh refinement strategy and computation of flux functions on non-

conformal faces. Finally, Sec. 5.3 gives distinct numerical test cases to show the accuracy and mass conservation of the method.

5.2 Numerical Method

5.2.1 Governing Equations

Incompressible, laminar flow of two immiscible, non-reacting fluids is governed by the Navier-Stokes equations defined on the domain $\Omega = \Omega_1 \cup \Omega_2 \subset R^2$. Here, subscripts 1 and 2 denote to sub-domains of the first and second fluids, respectively. Domain boundary is represented by $\partial\Omega$ while the $\Gamma = \Omega_1 \cap \Omega_2$ denotes the interface separating the fluid phases. Problem domain, Ω is fixed but two fluid domains, Ω_1 and Ω_2 and the interface, Γ change in time.

The non-dimensional incompressible Navier-Stokes and the continuity equations on the global domain, Ω can be written as,

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\,\mathbf{u} = -\frac{\nabla p}{\rho} + \frac{1}{\mathrm{Re}\rho}\nabla \cdot (2\mu S) + \frac{\mathbf{e}_g}{\mathrm{Fr}^2} + \frac{\kappa \mathbf{n}_\Gamma}{\mathrm{We}}, \quad \nabla \cdot \mathbf{u} = 0, \quad \forall \mathbf{x} \in \Omega \tag{5.1}$$

where $\mathbf{u}$ denotes the velocity field, p is the static pressure, ρ is the density, μ is the dynamic viscosity, $S = \frac{1}{2}\left(\nabla \mathbf{u} + \nabla \mathbf{u}^T\right)$ is the deformation rate tensor, $\mathbf{e}_g$ is the direction where gravitational acceleration acts, $\mathbf{n}_\Gamma$ is the unit normal of Γ and $\kappa = \nabla \cdot \mathbf{n}_\Gamma$ is the local curvature of the interface. Following parameters are used to get dimensionless quantities,

$$\mathbf{x} = \frac{\mathbf{x}^*}{L_R}, \quad t = \frac{t^*}{L_R/U_R}, \quad \mathbf{u} = \frac{\mathbf{u}^*}{U_R}, \quad p = \frac{p^*}{\rho_R U_R^2}, \quad \rho = \frac{\rho^*}{\rho_R}, \quad \mu = \frac{\mu^*}{\mu_R}, \quad \mathbf{e}_g = \frac{\mathbf{g}}{|\mathbf{g}|} \tag{5.2}$$

where superscript * denotes the dimensional parameter, the subscript R refers to corresponding reference value. The non-dimensional Reynolds, Froude and Weber numbers in the Eq. 5.1 are defined as $\mathrm{Re} = \rho_R U_R L_R/\mu_R$, $\mathrm{Fr} = U_R/\sqrt{\mathbf{g} L_R}$ and $\mathrm{We} = \rho_R U_R L_R/\sigma_R$, respectively. Hereafter, $\mathbf{F} = \frac{\mathbf{e}_g}{\mathrm{Fr}^2}$ is used for the non-dimensional body force term.

Interface continuum condition, i.e. no mass transfer between phases and kinematic condition i.e. jump in the normal component of stresses are balanced with

the surface tension forces so that the net stress vanishes along Γ, are implicitly included in the equation. Surface tension forces are neglected in this study but the method still governs many practical, large scale problem where curvature is too small to contribute to governing equations.

In the level set method, interface between the phases is represented by at least a Lipschitz continuous function, ϕ which is positive in one fluid domain and negative in the other. The zero contour of the implicit LS function, ie. $\phi(\mathbf{x}, t) = 0$ defines the current location of interface. Using this definition, ρ and μ can be globally defined.

$$\begin{aligned} \rho(\phi) &= \rho_1 H(\phi) + \rho_2 \left(1 - H(\phi)\right) \\ \mu(\phi) &= \mu_1 H(\phi) + \mu_2 \left(1 - H(\phi)\right) \end{aligned} \tag{5.3}$$

where $H(\phi)$ is the classical Heaviside step function. Time evolution of the interface can be obtained by simple advection of the level set function with the known velocity field .

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \tag{5.4}$$

In the interface model, LS is kept as signed distance function such that $|\nabla \phi| = 1$ is fulfilled. Unfortunately, evaluation of LS equation destroys the regularity of LS function and creates very large or small gradients near the interface. Reinitialization replaces the distorted LS with more desired signed distance function to enhance the accuracy of interface representation. The details of the high-order reinitialization solution is given in Sec. 5.2.3.3.

5.2.2 Discretization

A high-order splitting scheme [93] is used for the temporal discretization of the flow equations. The scheme is semi-implicit, in which non-linear term is integrated explicitly and linear term is treated implicitly. We adopted second order backward differentiation for unsteady term and second order extrapolation for non-linear term. With this implementation, Eq. 5.1 can be advanced from time levels, t_n to t_{n+1} by solving the following semi-discrete equations decomposed

into three fractional steps for $\mathbf{u}$.

$$\frac{\hat{\mathbf{u}} - \alpha_0 \mathbf{u}^n - \alpha_1 \mathbf{u}^{n-1}}{\Delta t} = \beta_0 \left(\left(\mathbf{u}^n \cdot \nabla \right) \mathbf{u}^n + \mathbf{F}^n \right) + \beta_1 \left(\left(\mathbf{u}^{n-1} \cdot \nabla \right) \mathbf{u}^{n-1} + \mathbf{F}^{n-1} \right) \tag{5.5a}$$

$$\frac{\hat{\hat{\mathbf{u}}} - \hat{\mathbf{u}}}{\Delta t} = -\frac{1}{\rho(\phi)} \nabla p^{n+1} \tag{5.5b}$$

$$\frac{\gamma_0 \mathbf{u}^{n+1} - \hat{\hat{\mathbf{u}}}}{\Delta t} = \frac{1}{\rho(\phi) Re} \nabla \cdot \mu(\phi) \nabla \mathbf{u}^{n+1} \tag{5.5c}$$

The coefficients, α,β and γ correspond to second order stiffly stable scheme and their values can be found in [93, 92]. In the splitting scheme, pressure is decoupled from velocity which avoids spurious pressure modes and enable to use equal order approximations for both flow fields.

In the first step, nonlinear terms and body forces are advanced explicitly by second order stiffly stable (SS) extrapolation scheme. Then, incompressibility constraint is enforced in Eq. 5.5(b) by requiring that the second intermediate velocity field, $\hat{\hat{\mathbf{u}}}$ is divergence-free. Finally, a modified Helmholtz equation is solved for the viscous terms to obtain next time level velocities. In [72], it is demonstrated that the splitting scheme presented preserves the optimal second order accuracy.

5.2.2.1 Nonlinear Treatment

The first step of the splitting scheme requires the approximation of nonlinear term which is written in divergence form i.e. $\left(\mathbf{u} \cdot \nabla \right) \mathbf{u} = \mathbf{u} \cdot \nabla \mathbf{u} + \mathbf{u} \nabla \cdot \mathbf{u} = \nabla \cdot \left(\mathbf{u} \otimes \mathbf{u} \right)$ where $\mathbf{u} \otimes \mathbf{v} = u_i v_j,\ i, j = 1, \cdots, d$. Let $\mathbf{u}$ is approximated by $\mathbf{u}_h \in V_N^d$. Multiplying the nonlinear term with a test function, $\mathbf{v}_h \in V_N^d$, integrating over element domain and performing integration by parts, we obtain the following local statement.

$$\left(\nabla \cdot \left(\mathbf{u}_h \otimes \mathbf{u}_h \right), \mathbf{v}_h \right)_{D_k} = - \left(\mathbf{u}_h \otimes \mathbf{u}_h, \nabla \cdot \mathbf{v}_h \right)_{D_k} + \left(\mathbf{n} \cdot [\![\mathbf{u}_h \otimes \mathbf{u}_h]\!], \mathbf{v}_h \right)_{\partial D_k} \tag{5.6}$$

Due to the discontinuous approximation space, the flux function, $\mathbf{n} \cdot [\![\mathbf{u}_h \otimes \mathbf{u}_h]\!]$, in the boundary contribution is not uniquely defined and hence replaced with

the local Lax-Friedrich numerical flux, $\tilde{F}_c$,

$$\tilde{F}_C = \mathbf{n} \cdot \{\mathbf{u}_h \otimes \mathbf{u}_h\} + \frac{1}{2}\Lambda_{\partial D_k}[\![\mathbf{u}_h]\!] \tag{5.7}$$

Here, $\Lambda_{\partial D_k}$ denotes the maximum eigenvalue of the flux Jacobian in absolute value which reads, $\Lambda_{\partial D_k} = \max_u |\mathbf{n} \cdot \frac{\partial(\mathbf{u}_h \otimes \mathbf{u}_h)}{\partial \mathbf{u}_h}|$. The first term of the flux function denotes central part and and the second one is the dissipative contribution. $\Lambda_{\partial D_k}$ determines the required artificial diffusion to stabilize the system. This flux choice leads to compact stencil size such that the degrees of freedom of an element couple only its immediate neighbors. Dirichlet boundary conditions is applied weakly by defining a exterior ghost state where average jump operators become $\{\mathbf{u}_h \otimes \mathbf{u}_h\}_{\partial D_k \cap \Omega_D} = 0.5\,(\mathbf{u}_h \otimes \mathbf{u}_h + \mathbf{u}_D \otimes \mathbf{u}_D)$ and $[\![\mathbf{u}_h]\!]_{\partial D_k \cap \Omega_D} = (\mathbf{u}_h - \mathbf{u}_D)$.

Explicit treatment of nonlinear terms introduces a Courant-Friedrichs-Lewy (CFL) type restriction on the time step size. CFL estimate for high order methods gives $\Delta t \approx O(h/UN^2)$ as reported in [92] for advection model problem. Here, U and h refer the characteristic velocity and mesh size, respectively.

5.2.2.2 Implicit Treatment

The time splitting scheme requires the solution of implicit pressure Poisson eqution for velocity projection and modified Helmholtz equation for velocity correction. Specifically, Poisson equation needs to special attention due to poor conditioning of the system for the efficient solution and obtaining divergence free velocity field for long term stability. In this section, we start with discretization and solution of velocity projection step and then give the details of the Helmholtz equation discretization based on the ideas developed in the first part.

Incompressibility constraint is enforced by taking the divergence of Eq. 5.5(b) which leads to the following variable density pressure Poisson equation.

$$\nabla \cdot \left(-\frac{\hat{\mathbf{u}}}{\Delta t}\right) = \nabla \cdot \left(-\frac{1}{\rho(\phi)}\nabla p^{n+1}\right) \tag{5.8}$$

Boundary conditions of this Neumann problem can be derived from the original equation. To preserve the temporal accuracy and fulfill the compatibility condition, Neumann boundary conditions have to be approximated to the same order

with time discretization scheme, which requires extrapolation of the Neumann boundary data,

$$\mathbf{n}\cdot\left(\frac{1}{\rho}\nabla p\right)^{n+1} = \mathbf{n}\cdot\Big[\beta_0\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\,\mathbf{u}-\frac{1}{\rho\mathrm{Re}}\nabla\cdot(2\mu S)-\mathbf{F}\right)^{n}+ \beta_1\left(\frac{\partial \mathbf{u}}{\partial t}+(\mathbf{u}\cdot\nabla)\,\mathbf{u}-\frac{1}{\rho\mathrm{Re}}\nabla\cdot(2\mu S)-\mathbf{F}\right)^{n-1}\Big] \tag{5.9}$$

There are different DG methods developed for the discretization of the equations with second (or higher) order operators generally based on the mixed formulation i.e. writing the equations in first order systems by defining new auxiliary variables (Please refer to [4] for the detailed analysis). In this study, Symmetric Interior Penalty (SIP) [3] method is preferred due to its simplicity, computational efficiency and compact stencil size. Let pressure, p is approximated by $p_h \in V_N$ and $v_h \in V_N$ be the test functions. SIP discretization of the Eq. 5.8 for an elemental domain reads as follow,

$$\begin{aligned}\left(\frac{1}{\rho}\nabla p_h,\nabla v_h\right)_{D_k} - \left(\mathbf{n}\cdot\{\frac{1}{\rho}\nabla p_h\},v_h\right)_{\partial D_k\setminus\Omega_N} - \left([\![p_h]\!],\mathbf{n}\cdot\frac{1}{\rho}\nabla v_h\right)+\\ (\gamma[\![p_h]\!],v_h)_{\partial D_k\setminus\Omega_N} = \left(-\frac{\nabla\cdot\hat{\mathbf{u}}}{\Delta t},v_h\right)_{D_k} - \left(p_D,\mathbf{n}\cdot\frac{1}{\rho}\nabla v_h\right)_{\partial D_k\cap\Omega_D}+\\ (\gamma p_D,v_h)_{\partial D_k\cap\Omega_D}+(p_N,v_h)_{\partial D_k\cap\Omega_N}\end{aligned} \tag{5.10}$$

where γ is the penalty parameter, p_D and p_N are Dirichlet and derived Neumann boundary conditions. In the equation, we dropped time dependence of the pressure for clarity. Penalty parameter plays an important role for the stability of the scheme and should be selected sufficiently large to enforce coercivity. On the other hand, selecting arbitrarily large γ values increases condition number of the system and degrades the performance of linear solvers. We use the following definition of the penalty parameter.

$$\gamma_{\partial D_k} = \frac{1}{2}(N+1)(N+2)\max\left(\frac{h_{\partial D_k^-}}{\rho_{\partial D_k^-}h_{D_k^-}},\frac{h_{\partial D_k^+}}{\rho_{\partial D_k^+}h_{D_k^+}}\right) \tag{5.11}$$

Here, $h_{\partial D_k}$ and h_{D_k} denote the surface area and volume of the element, respectively. On the non-conformal faces with hanging nodes, where more than two elements are connected, γ is computed using the collection of all adjacent elements which guarantees the coercivity of the bilinear form [160].

Solution of projection step is challenging and time consuming due to construction of global system matrices in each time step. Also, discretization matrix of the Poisson problem is poorly conditioned in high density ratio flows even with optimum penalty parameter selection. To show the spectral properties of resulting discretization matrices, condition numbers are computed in a computational domain of $[0,1]^2$ with 62 triangular elements using MATLAB built-in function *cond*() for the circular, triangular and two intersecting circular shape of the same characteristic lengths. We observed that condition number is highly sensitive to the density ratios not to interface shape and topology. Condition numbers are of the same order of magnitude for different interface topologies but grow rapidly with increasing density ratio as seen in Table 5.1. Special care should be considered to solve pressure Poisson problem, efficiently.

Table 5.1: Condition numbers of Pressure Poisson operator with SIP discretization for different interface topologies and density ratios. ($N = 3, K = 62$).

	Density Ratio			
Interface	1	10	100	1000
Circle		1.31×10^4	1.33×10^5	5.15×10^7
Triangle	7.03×10^3	1.51×10^4	4.70×10^5	8.37×10^7
Intersecting Circles		2.78×10^4	2.43×10^5	7.08×10^7

The pressure system arises from the SIP discretization can be written in the generic form, $Ax = b$ where b and x are the right hand side and unknown vectors, A is the symmetric and positive definite coefficient matrix so that the system can be solved efficiently with preconditioned conjugate gradient (PCG) method. To avoid costly set-up time and minimize the memory requirements, we newer construct the global matrices in CG solver instead matrix vector multiplications are performed explicitly by doing the calculations per-element as given in Eq. 5.17. Because the system is poorly conditioned in high-density ratio flows, efficient preconditioners are required to converge the system within small iteration numbers. Multigrid as the preconditioner for the CG method offers faster solutions on badly conditioned problems then full multigrid or incomplete LU preconditioners [182]. Among the other multigrid methods algebraic multigrid (AMG) requires only entries of the corresponding matrices. AMG methods have set-up and solve phases. Set-up phase includes construction of hierarchical grids

by aggregating fine grid nodes to coarse grid node and defining restriction and interpolation operators to transform data between fine to coarse and coarse to fine grids, respectively. Hierarchy of the different grids can be computed at the beginning of analysis and stored for fixed grid and constant coefficient problems where sparsity pattern and spectral properties of the matrix does not change in time. But this is not the case for dynamic problems as the pressure Poisson problem where set-up of the preconditioner creates computational burden and may suppress the actual solution time.

A p-multigrid preconditioner is designed for the dynamic pressure Poisson system to reduce the computational time and memory requirements. We avoid to construct full matrix and related hierarchical levels. Required residuals are computed with the matrix-free approach starting from the approximation order of N until reaching first order coarse system. Restriction operator, $\mathbf{I}_p^q$ that projects the errors from the polynomial space of order p, to a space of order q for $p > q$ are easily constructed due to discontinuous interpolating functions,

$$\mathbf{I}_p^q = (M^q)^{-1} M^{qp} \tag{5.12}$$

where corresponding mass matrices can be computed in index notation as,

$$M_{i,j}^p = (v_i^p, v_j^q)_R \quad \text{and} \quad M_{i,j}^{pq} = (v_i^p, v_j^q)_R \tag{5.13}$$

Note that the restriction operator is defined on the reference element and can be applied element-wise through the whole domain without considering the non-conformal face pairs, curved elements etc. The prolongation operator that transforms the state vectors from the low order polynomial space to higher one can be defined similarly leading $\mathbf{I}_q^p = (\mathbf{I}_p^q)^T$. With the use of Lagrange basis space, these operators are full but small matrices in the size of dimension of polynomial spaces, $N_p \times N_q$ implying the required matrix-vector multiplication can be performed efficiently for an element. Because residuals are computed in matrix-free form, we don't need to define residual restriction/prolongation operators where they can be different from the state vectors' operators. Instead, residual functions handling different approximation orders are constructed. Geometric mappings and metric identities are constant for straight sided elements enable to compute residuals without storing extra information.

After reducing approximation order to the first order, $q = 1$, the system matrix are assembled and related hierarchical grids are constructed at this level bases on the recently introduced aggregation (AMG) method [64]. One pre- and one post smoothing step, S_l are used to remove high frequency errors at the corresponding level, l. Damped Jacobi smoother given below is utilized due to its comparatively low set-up cost.

$$x = x + \omega D^{-1}\left(b - Ax\right), \quad \omega = \frac{4}{3}\frac{1}{\varrho(D^{-1}A)} \tag{5.14}$$

where D is the diagonal of A and $\varrho(D^{-1}A)$ is the spectral radius of the matrix $D^{-1}A$ estimated from the Arnoldi iterations. K-multigrid cycles are used as a preconditioner for the CG iterations as illustrated in Algorithm 1. Number of coarse level iterations increases with K-cycles but it is required to achieve grid independent convergence and improving overall runtime performance. Note that combining matrix-free p-multigrid with aggregation (AMG) reduce the storage and computational effort significantly. If the system is solved with matrix form, $N_N \times N_N \times (1 + N_f)$ non-zero entries have to be stored for an element only at the highest level, where N_f stands for the total face connection pairs. This requirement is reduced to $N_1 \times N_1 \times (1 + N_f)$ so that matrix free approach becomes more effective in non-conformal discretizations, $3D$ problems and high orders.

After obtaining the next time level pressure, p_h^{n+1} second intermediate velocity field is computed with the use of Eq. 5.5(b). Required pressure gradient, $\mathbf{G} = \nabla p_h^{n+1}$ is evaluated in the weak form,

$$\left(\mathbf{G}, v_h\right)_{D_k} = -\left(p_h^{n+1}, \nabla v_h\right)_{D_k} + \left(\mathbf{n} \cdot \{p_h^{n+1}\}, v_h\right)_{\partial D_k} \tag{5.15}$$

where the flux function in the surface integral contribution is replaced with the numerical central flux.

In the splitting scheme, divergence-free constraint is enforced only in weak and local sense by projecting the second intermediate velocity field to approximately divergence free space with setting $\nabla \cdot \hat{\hat{\mathbf{u}}} = 0$. This may cause compressibility artifacts at the element boundaries for long term, slightly viscous, undamped problems [169]. Numerical test demonstrated that deviation of the velocity field

Algorithm 1 *K-Cycle Matrix-free p-Multigrid Preconditioner*

$x_{l,p} \leftarrow$ K-Cycle$((l,p), b_{l,p}, x_{l,p})$
Input: initial guess x_0, order n, level l
Output: Updated Solution x_l^N
if $q \geq 1$ **then** { Operations in Matrix-free Form }
  $x_p \leftarrow S_p(b_p, A_p, x_p)$ { pre-smoothing }
  $r_q \leftarrow b_q - A_q x_q$ { compute residual for $1 < n \leq N$ }
  $r_q \leftarrow I_p^q r_q$ { restrict residual to coarse-grid }
  $x_q \leftarrow$ K-Cycle(q, r_q, x_q) { inner CG iteration }
else { Operations in Matrix Form }
  $x_l \leftarrow S_p(l_p, A_l, x_l)$ { pre-smoothing }
  $r_l \leftarrow b_l - A_l x_l$ { compute residual in matrix form }
  $r_{l+1} \leftarrow I_l^{l+1} r_l$ { restrict residual to coarse-grid }
  if $l+1 = L$ **then** { coarsest grid exact solution }
    $x_{l+1} \leftarrow A_{l+1}^{-1} r_{l+1}$
  else
    $c_{l+1} \leftarrow$ K-Cycle$(l+1, r_{l+1}, x_{l+1})$ { inner CG iteration }
    $v_{l+1} \leftarrow A_{l+1} c_{l+1}, \quad a_1 \leftarrow c_{l+1}^T v_{l+1}$
    $\alpha_1 \leftarrow c_{l+1}^T r_{l+1}, \quad \tilde{r}_{l+1} \leftarrow r_{l+1} - \frac{\alpha_1}{a_1} v_{l+1}$
    if $\|\tilde{r}_{l+1}\| \leq \text{TOL}\|r_{l+1}\|$ **then**
      $x_{l+1} \leftarrow \frac{\alpha_1}{a_1} c_{l+1}$
    else
      $d_{l+1} \leftarrow$ K-Cycle$(l+1, \tilde{r}_{l+1}, x_{l+1})$ { inner second CG iteration }
      $w_{l+1} \leftarrow A_{l+1} d_{l+1}, \quad \gamma \leftarrow d_{l+1}^T v_{l+1}$
      $\beta \leftarrow d_{l+1}^T w_{l+1}, \quad \alpha_2 \leftarrow d_{l+1}^T \tilde{r}_{l+1}, \quad a_2 \leftarrow \beta - \frac{\gamma^2}{a_1}$
      $x_{l+1} \leftarrow \left(\frac{\alpha_1}{a_1} - \frac{\gamma \alpha_2}{a_1 a_2}\right) c_{l+1} + \frac{\alpha_2}{a_2} d_{l+1}$
  if $l \neq 0$ **then** {Operations in Matrix Form }
    $r_l \leftarrow I_{l+1}^l r_{l+1}$ { prolongate }
    $r_l \leftarrow r_l - A_l x_l$ { compute residual }
    $x_l \leftarrow S_p(r_l, A_l, x_l)$ { post-smoothing }
  else {Operations in Matrix-Free Form }
    $r_p \leftarrow I_q^p r_p$ { prolongate }
    $r_p \leftarrow r_p - A_p x_p$ { compute residual }
    $x_p \leftarrow S_p(r_p, A_p, x_p)$ { post-smoothing }

from incompressibility is more severe at high density ratio flows in under-resolved cases and lead unphysical movement of the interface and numerical instability at long term analysis. To overcome the problem, a local post-processing technique is used to obtain exactly divergence free velocity field from the $\hat{\mathbf{u}}$ by projecting it to non-divergent space. A vector basis function, ψ is constructed such that $\psi = \nabla \times v_h$ where $v_h \in V_N$ are members of the N^{th} order orthogonal polynomial basis used. By construction, $\nabla \cdot \psi = 0$ is fulfilled in the machine precision. Then, divergence-free velocity field, $\hat{\underline{\mathbf{u}}}$ is expanded using the new basis space, $\hat{\underline{\mathbf{u}}} = \sum_{i=1}^{N_d} c_i \psi_i$. Required coefficients and hence the divergence-free velocity field can be recovered by the Galerkin projection on the reference element, T once the weakly divergence-free velocity field, $\hat{\mathbf{u}}$ is transformed to local element coordinate frame, i.e. $(\psi, \psi)_T \, c = \left(\psi, \hat{\mathbf{u}}\right)_T$. This operation is completely local and can be carried out element-by-element fashion.

Finally, time discretization of the flow equations are completed by applying viscous correction through the solution of a modified Helmholtz equation given in Eq. 5.5(c) which can be recast into following form,

$$\frac{\gamma_0 \rho Re}{\Delta t} \mathbf{u}^{n+1} - \nabla \cdot \mu \nabla \mathbf{u}^{n+1} = \frac{\rho Re}{\Delta t} \hat{\underline{\mathbf{u}}} \tag{5.16}$$

Actually, the equation is composed of scalar Helmholtz equations for each velocity component closed with the appropriate velocity boundary conditions at time, t^{n+1}. For the scalar form, this system is very similar to pressure Poisson equation with additional scaled $\mathbf{u}^{n+1}$ term on the left side. Let u is a component of the velocity vector approximated by $u_h \in V_N$ and $v_h \in V_N$ be the test functions. Similar to the pressure equation, SIP discretization of the for an elemental domain can be written as,

$$\left(\frac{\gamma_0 \rho Re}{\Delta t} u_h, v_h\right)_{D_k} + (\mu \nabla u_h, \nabla v_h)_{D_k} - (\mathbf{n} \cdot \{\mu \nabla u_h\}, v_h)_{\partial D_k \setminus \Omega_N} -$$

$$(\llbracket u_h \rrbracket, \mathbf{n} \cdot \mu \nabla v_h) + (\gamma \llbracket u_h \rrbracket, v_h)_{\partial D_k \setminus \Omega_N} = \left(\frac{\rho Re}{\Delta t} \hat{\underline{u}}, v_h\right)_{D_k} - (u_D, \mathbf{n} \cdot \mu \nabla v_h)_{\partial D_k \cap \Omega_D} +$$

$$(\gamma u_D, v_h)_{\partial D_k \cap \Omega_D} + (u_N, v_h)_{\partial D_k \cap \Omega_N} \tag{5.17}$$

where penalty parameters and boundary conditions are as defined before. The system is symmetric, positive definite and solved with CG method in matrix-free form. Because, solution of the velocity system is considerably easier then

the pressure equation due to scaled mass matrix, block-Jacobi preconditioner is used to reduce the computational effort and increase the runtime performance.

5.2.3 Interface Modeling

5.2.3.1 Discontinuous Fluid Properties

Discontinuous fluid properties are avoided by smoothing their variations in the vicinity of the interface with thickness of size, ϵ. There are many definitions of smoothing functions, such as piecewise continuous [70] or continuous trigonometric [176]. Here, hyperbolic tangent function is used as the regularized Heaviside function,

$$H_\epsilon(\phi) = \tanh(\frac{\pi\phi}{\epsilon}) \tag{5.18}$$

which smooths the variations over the distance 2ϵ. The Heaviside function is infinitely differentiable and more appropriate for high-order discretizations.

The interface thickness, ϵ should be selected as small as possible to get a sharp profile for accuracy but large enough to capture variations and to stabilize the system. Classical low order methods require to choose ϵ at least to the order of characteristic mesh size, $\epsilon = O(h)$. Due to the high order interpolation of the DG method, variations can be resolved and integrated stably in the smaller distance of $\epsilon = O(h/N)$. Our numerical tests show that $\epsilon = 2h/N$ offers sharp but smooth profile with good integrability of the equations.

After defining the smoothed Heaviside function, discontinuous material properties can be replaced with globally defined continuous counterparts. For the isothermal, incompressible flows only density and viscosity are to be smoothed such that,

$$\begin{aligned} \rho(\phi) &= \frac{1}{2}\rho_1\left(1 + H_\epsilon(\phi)\right) + \frac{1}{2}\rho_2\left(1 - H_\epsilon(\phi)\right) \\ \mu(\phi) &= \frac{1}{2}\mu_1\left(1 + H_\epsilon(\phi)\right) + \frac{1}{2}\mu_2\left(1 - H_\epsilon(\phi)\right) \end{aligned} \tag{5.19}$$

which gives, $\rho(\phi) = \rho_1$ for $\phi < 0$ and $\rho(\phi) = \rho_2$ for $\phi > 0$ with smooth transition around $\phi = 0$. Level set function has to be symmetric to the zero level set to accurately represent the material properties, which requires constant gradient

at least around the ϵ-neighborhood of the interface which is achieved by the use of signed distance function.

5.2.3.2 Level Set Advection

In this study, exactly divergence-free flow field, that satisfy $\nabla\cdot\mathbf{u}=0$, is obtained by the local post-processing. Using the identity, $\nabla\cdot(\mathbf{u}\phi)=\phi\nabla\cdot\mathbf{u}+\mathbf{u}\cdot\nabla\phi$, level set equation can be written in the conservative form leading the following linear advection equation,

$$\phi_t+\nabla\cdot(\mathbf{u}\phi)=0 \tag{5.20}$$

Similar to previous section, let ϕ is approximated by $\phi_h\in V_N$ and $v_h\in V_N$ are the smooth test functions. Weak form of the Eq. 5.20 can be obtained by integration by parts,

$$(\frac{\partial\phi_h}{\partial t},v_h)_{D_k}-(\mathbf{u}_h\phi_h,\nabla v_h)_{D_k}+(\tilde{F}_L,v_h)_{\partial D_k}=0 \tag{5.21}$$

where $\tilde{F}_L(\phi_h,\mathbf{u}_h)$ is the numerical flux to approximate the normal trace of non-uniquely defined flux function,$\mathbf{n}\cdot[\![\mathbf{u}_h\phi_h]\!]$. For $\tilde{F}_L$, we choose the upwinding on the value of ϕ_h on ∂D_k, such that

$$\tilde{F}_L(\phi_h,\mathbf{u}_h)=(\mathbf{n}\cdot\mathbf{u}_h)\{\phi_h\}+\frac{1}{2}|\mathbf{n}\cdot\mathbf{u}_h|[\![\phi_h]\!] \tag{5.22}$$

The level set equation, hence the interface location is evolved in time with a explicit TVD Runge-Kutta (RK) method. Stability region of the RK method is slightly larger than the Adams-Bahforth method used in the explicit integration of the flow equations. This guarantees the stability of the LS advection under the time-step restriction of the semi-explicit flow solution. DG method has excellent mass conservation properties due to the low numerical dissipation achieved by the use of local high-order polynomial approximations [1, 117]. The formulation, presented here enables to obtain accurate interface presentation without using any special treatments or modifications of the original LS formulation

5.2.3.3 Reinitialization

While evolving the Eq. 5.21 in time, level sets adjacent to interface may move with velocities different from the zero level set which distorts the scalar LS function. Loosing the regularity and symmetry of the LS function around the interface may cause numerical instabilities, errors in the computation of fluid properties and precisely locating interface location. Thus, LS function should be reinitialized to signed distance having the property of $|\nabla\phi| = 1$ at least around the interface. But, it is not required to apply reinitialization at the end of each time step to improve the mass loss unlike the standard level set methods [181, 126, 165], due to high-order, conservative interface modeling. In this study, LS function is reinitialized only when it produces too step or flat profile near the interface based on the indicator relying $|\nabla\phi|$.

There are several ways to reinitialize level set function such as fast marching [187], fast sweeping [158] and flow based method [181]. The flow based reinitialization is used here due to its flexibility and accuracy.

$$\frac{\partial\phi}{\partial\tau} + \mathrm{sgn}(\phi_0)(|\nabla\phi|-1) \ = 0, \quad \phi(\mathbf{x},0) = \phi_0 \tag{5.23}$$

where τ is the pseudo time which is not related with the physical time, t. Eq. 5.23 defines an artificial flow to obtain signed distance function in steady-state. In principle, interface location remains unchanged because $\mathrm{sgn}(\phi_0) = 0$, zero contour of ϕ and ϕ_0 are the same. Away from the interface, $|\nabla\phi|$ converges to 1 without explicitly locating the interface location. The general approach for solving Eq. 5.23 is to regularize the signum term in a narrow band in the vicinity of the interface instead of using sharp signum function. A careful investigation of the equation reveals that characteristics are emanate from the interface in normal direction with a speed of 1 without regularization term. However, the speed changes with the $sgn_\alpha(\phi_0)$ term that selection of the regularization affects the steady state solution and the convergence rate/accuracy of the scheme [91]. We choose to regularize sign function similar to Heaviside function,

$$\mathrm{sgn}_\epsilon(\phi_0) = \tanh(\frac{\pi\phi_0}{|\nabla\phi_0|\epsilon}) \tag{5.24}$$

where band thickness, ϵ has the same value with the previous section. Because, LS function is not reinitialized in each time step, it is not exactly signed distance function in the whole computation time. The scaling factor, $|\nabla\phi_0|$ is added to avoid very small coefficients resulting in small characteristic speeds, when the LS function becomes flat or to improve the accuracy when the LS function is step around the interface.

After regularizing the signum term, the reinitialization equation can be written as the following Hamilton-Jacobi form with smooth coefficient,

$$\frac{\partial\phi}{\partial\tau} + H(\nabla\phi, \mathbf{x}) = 0, \quad \phi(\mathbf{x}, 0) = \phi_0 \tag{5.25}$$

where the Hamiltonian, $H(\nabla\phi, \mathbf{x})$ denotes $\text{sgn}_\epsilon(\phi_0)(|\nabla\phi| - 1)$. In [91], we introduced a direct, adaptive, high-order DG method for the LS reinitialization with local time stepping and artificial diffusion stabilization which preserves the optimal accuracy . Here, this technique is modified for accounting the local interface modeling for efficient multiphase flow simulations. Solution of the HJ equation requires accurate approximation of derivatives. We followed the local DG scheme [199] to approximate $\nabla\phi$ by defining two new variables for each derivative component and applied the standard upwind DG method. Let p_x^1 and p_x^2 be auxiliary variables used to approximate $\frac{\partial\phi}{\partial x}$ when the fluxes are chosen from the left and right upwinding sides, respectively.

$$p_x^1 - \phi_x = 0 \quad \text{and} \quad p_x^2 - \phi_x = 0 \tag{5.26}$$

which gives the following upwind DG scheme in weak form,

$$(p_x^{1,2}, v_h)_{D^k} + (\phi_h, \frac{\partial v_h}{\partial x})_{D^k} - (F_R^{1,2} n_x, v_h)_{\partial D^k} = 0 \tag{5.27}$$

the left, F_R^1 and right, F_R^2 upwind fluxes are defined as

$$F_R^1 = \begin{cases} \phi^+ & for \quad n_x \geqslant 0 \\ \phi^- & for \quad n_x < 0 \end{cases} \quad \text{and} \quad F_R^2 = \begin{cases} \phi^- & for \quad n_x \geqslant 0 \\ \phi^+ & for \quad n_x < 0 \end{cases} \tag{5.28}$$

Auxiliary variables p_y^1 and p_y^2, which are used to approximate $\frac{\partial\phi}{\partial y}$, can be computed in similar way. To complete the discretization, Hamiltonian given in Eq. 5.25 is replaced with monotone and consistent numerical Hamiltonian,$\widehat{H}$

and integrated over the element D_k to get

$$(\phi_t, v)_{D^k} + (\widehat{H}(p_x^1, p_x^2, p_y^1, p_y^2), v)_{D^k} = 0 \tag{5.29}$$

In this study, following Godunov Hamiltonian is used because it is easy to implement, efficient and pure upwind scheme,

$$\widehat{H}(p_x^1, p_x^2, p_y^1, p_y^2) = \begin{cases} sign_\alpha(\phi_0)(\sqrt{\max((p_x^{1,m})^2, (p_x^{2,p})^2) + \max((p_y^{1,m})^2, (p_y^{2,p})^2)} - 1), & sign_\alpha(\phi_0) \geqslant 0 \\ sign_\alpha(\phi_0)(\sqrt{\max((p_x^{1,p})^2, (p_x^{2,m})^2) + \max((p_y^{1,p})^2, (p_y^{2,m})^2)} - 1), & sign_\alpha(\phi_0) < 0 \end{cases} \tag{5.30}$$

where $p_x^{1,p} = \max(p_x^1, 0)$ and $p_x^{1,m} = -\min(p_x^1, 0)$ with the same notation for all other variables. For a piecewise constant approximation, this scheme is monotone and converges to the entropy solution. However, stabilization is needed for higher order approximations. An artificial diffusion approach is utilized to damp out the high frequencies in the solution. Different from the limiting techniques, the artificial diffusion method does not directly reduce the accuracy to first order and avoids to reconstruction operations which is costly especially in non-conformal discretizations. How much and where the diffusion should be added to avoid oscillation without excessive smearing out requires reliable and scalable detectors. We used regularity detector based on the modal decay rate. For details, we refer to our recent work [91].

Characteristic velocities of the reinitialization equation, as mentioned before, point outwards from the interface in the direction of normals. In other words, LS function is reinitialized to signed distance starting from the interface. Multiphase flows requires the LS function to be signed distance function around the interface only. This means that we don't need to solve reinitialization equation to steady state over the hole domain instead over the small neighborhood of the interface. We kept the signed distance function on the thickness of 2ϵ which requires approximately $2\epsilon/\Delta\tau$ time steps. Because characteristic speed slightly lower than 1 due to regularization of the signum function, we use more time steps than estimated value. Numerical tests shown that ten percent extra time step is satisfactory to obtain the signed distance function in the band thickness. Constant level set function is used on the outside the interface band. In practice, a few explicit RK step is satisfactory to get the signed distance function around the interval so that the reinitialization does not create any computational burden

in the total simulation time.

5.2.4 Mesh Adaptivity

Although, DG discretizations are less diffusive comparing with the standard schemes, it is obvious that increasing resolution with adaptivity in the vicinity of the interface improves the accuracy. Adaptive mesh refinement (AMR) strategies are based on the conformal and non-conformal discretizations. In conformal discretizations, each face is shared by two elements so that AMR algorithm should handle the complicated mesh transition to make sure that mesh remains conformal after adaptation. This approach results with computational burden in refinement step with easy calculation of fluxes because of the conformity. On the other hand, adaptation step in non-conformal AMR strategy can be simply obtained by dividing elements. Then numerical fluxes should be evaluated at non-conformal faces where more than two elements are connected. In this study, non-conformal discretizations on unstructured triangular elements is selected. Although, it seems that added complication and cost related with the flux computation at non-conformal faces are more than creating conformal grid, this approach enable us to use more flexible and local adaptive grid.

In the adaptive scheme used here, the computational mesh consists of elements in a range of predefined levels with l_0 denoting initial coarse level and l_M being the maximum level. Refinement and coarsening are performed dynamically during the solution. The level of refinement and the elemental dependencies are stored in a hierarchical tree for efficient h type adaptivity.

Refinement is carried out in an isotropic way, i.e. a parent triangle is divided into four siblings by connecting the mid-edges. A threshold value,γ is selected to mark the elements for refinement. There is a flexibility for choosing the threshold value such as a predefined band width thickness or characteristic element length. If the $\min |\phi_k| \leqslant \gamma$ holds and refinement level of the element is smaller than the predefined maximum refinement level ($l_k < l_M$), then the element is marked for refinement and the approximation on the parent triangle is projected onto its four siblings.

If $\min|\phi_k| > \gamma$ holds and refinement level of all four sibling elements are larger than the initial coarse level, these elements are marked for coarsening. Marked elements are combined by removing center element and its three vertices. Then, solution of the siblings are projected to the parent element and level of refinement information is updated. The DG method supports arbitrary number hanging nodes per face so it is not restricted in order to keep the adaptive scheme flexible and local.

5.3 Numerical Tests

Verification problems are solved to investigate convergence properties and mass conservation of the our numerical framework in multiphase flows with varying density/viscosity ratios and interface topology. All numerical results presented here are obtained using high order fully DG method on locally adaptive triangular grids in two spatial dimensions.

5.3.1 Sloshing in Rectangular Tank

In this test case, water column oscillates in a rectangular tank. The driving forces are gravitational acceleration and viscous dissipation causing the water column eventually reach its lowest potential energy level from the initially perturbed shape which is given by the following sinusoidal function,

$$\Gamma = d + a_0 \sin\left(\pi(0.5d - x)\right) \tag{5.31}$$

where $d = 1$ is the mean water depth, a_0 is the amplitude of initial wave. This test case is good indicator to show the ability of numerical scheme to transfer potential energy into kinetic energy and the mass loss introduced by the discretization.

The problem is first solved in the computational domain of $[0, d] \times [0, 2d]$ for the small amplitude, $a_0 = 0.01$ where an analytical solution based on the linearized Navier-Stokes equations is reported by Wu et.al. [197]. The fixed grid used in this simulation is obtained starting with the coarse grid of characteristic element

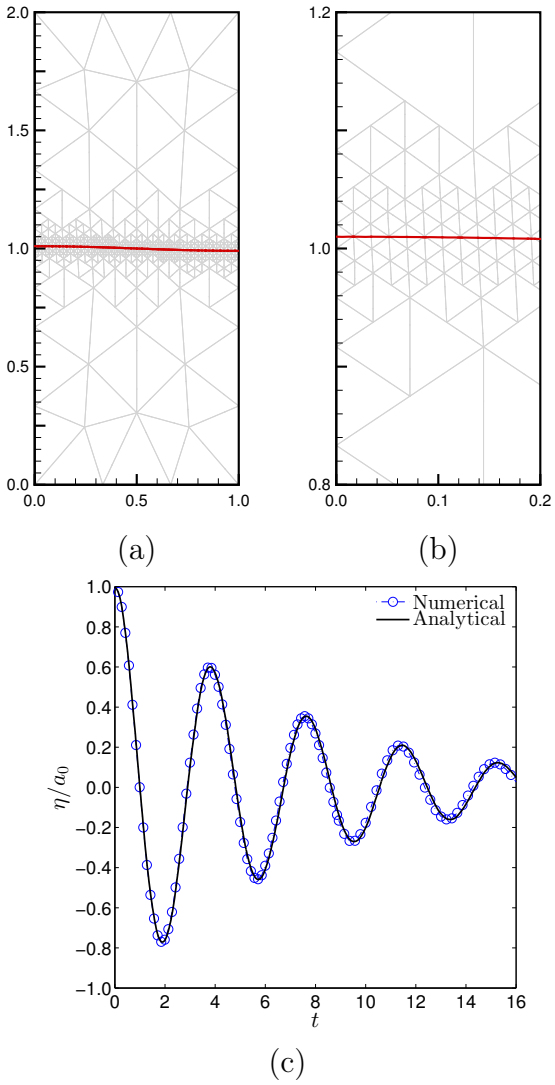
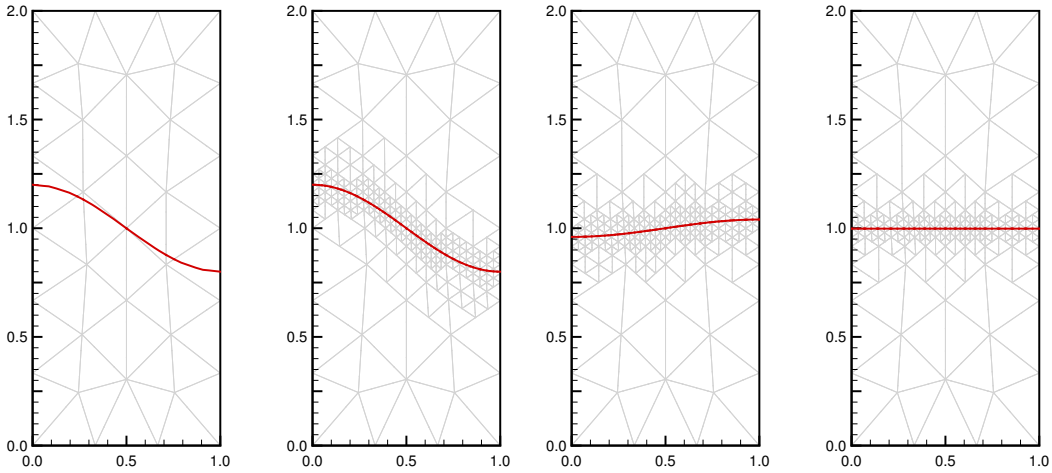


Figure 5.1: Sloshing problem for the small amplitude case, (a) Computational grid and initial interface shape for $l_M = 4$ (b) Zoomed view near the left wall. (c) Comparison of the computed wave amplitude with analytical solution [197]. ($\rho_l/\rho_g = 100$, $\mu_l/\mu_g = 100$, $Re = 100$, $Fr = 1$, $N = 3$, $a_0 = 0.01$)

length, $h = 1/3$, ($K = 50$). Then, high resolution near $y = 1$ is obtained with $l_M = 4$ local refinement as shown in Fig. 5.1(a-b). Slip boundary condition are assigned to the bottom and side walls while zero pressure is imposed at the top wall. Zero velocities and hydrostatic pressure distribution are used as the initial condition. The superimposed fluids have the density and viscosity ratios of 1/100. The non-dimensional Froude and Reynolds numbers are taken as 1 and 100, respectively. Fig. 5.1(c) illustrates the normalized wave elevation, η/a_0 against the time for computed and analytical solutions. The numerical results

match well with the analytical solution.

Fig. 5.2 illustrates the $l_M = 3$ locally adapted mesh structures and interfaces at different simulation times for the high initial amplitude, $a_0 = 0.2$ and local polynomial order of 5 for the same parameters with the previous case. Time integration is carried out until $t = 40$ where water column comes to rest. Mesh adaptivity used here always keeps interface in the highest level elements. Letting number of hanging nodes per face unconstrained, enable us to get local mesh structure.



(a) $t = 0$ (b) $t = 0$ (c) $t = 20.5$ (d) $t = 40$

Figure 5.2: Standing wave problem $l_M = 3$ locally adapted grid and the interface at different simulation times. ($\rho_l/\rho_g = 100$, $\mu_l/\mu_g = 100$, $Re = 100$, $Fr = 1$, $N = 5$, $a_0 = 0.2$)

Table. 5.2 shows the maximum area fluctuations introduced by the scheme for long term analyses. Different refinement levels and order of approximations are used in the numerical experiments. Similar problem parameters and initial mesh configurations with small amplitude case are used. Mass fluctuations are computed very accurately by the adaptive contouring algorithm and with the formula, $A_f = 100 \times (\max(A_t) - \min(A_t))/A_{exact}$, where A_t is the time history of the total area of the liquid. Mass is well conserved for this long integration time and increasing resolution in the vicinity of interface significantly improves the mass loss problem.

Table 5.2: Percentage mass fluctuations in sloshing in a rectangular tank problem for different refinement levels and order of approximations.

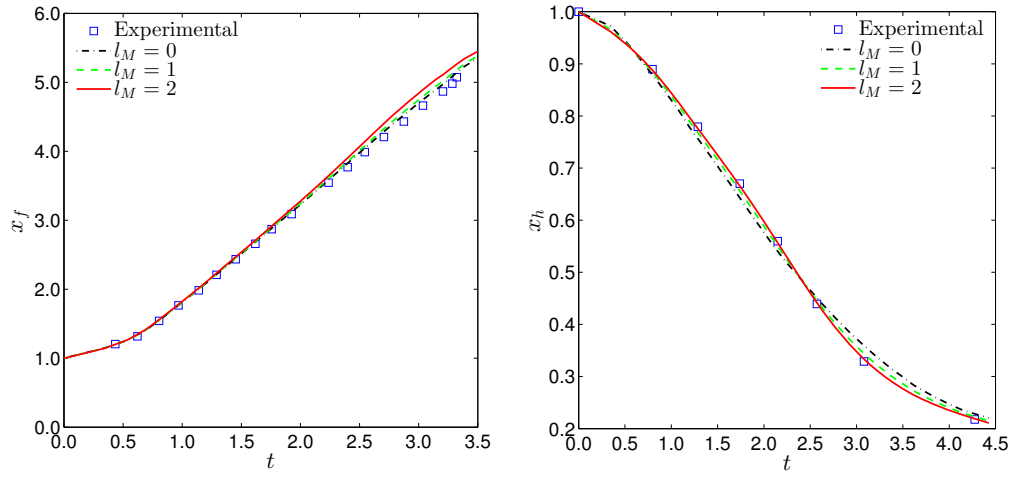
		l_M		
	N	0	1	2
% Mass Fluctuation	3	4.52×10^{-1}	1.11×10^{-1}	2.12×10^{-2}
	4	1.51×10^{-1}	8.27×10^{-2}	6.60×10^{-3}
	5	6.33×10^{-2}	1.55×10^{-2}	2.01×10^{-3}

5.3.2 Dam Break Problem

Dam break problem consists of sudden collapse of a rectangular liquid column into a horizontal plane under the action of gravitational acceleration. Flow field is highly unsteady and interface encounters strong deformations. Due to viscosity, water column eventually comes to rest and occupy the bottom of the tank. Measurements of the exact interface shape and analytical solution for the viscous case are not available but some secondary data such as column height reduction and wave front speeds are reported in literature.

All computations are conducted on the $[0, 1.5a] \times [0, 6a]$ domain with $a = 1$. Square water column with the length of a is released at $t = 0$. Slip boundary conditions are applied to the bottom and side walls of the tank. Similar to the experimental study, top wall is modeled as open boundary with zero atmospheric pressure and zero normal gradients on the velocities. Water and air properties are assigned to the liquid and gas phases as $\rho_l = 1000$ kg m^{-3}, $\rho_g = 1$ kg m^{-3} and $\mu_l = 10^{-3}$ Pa s, $\mu_g = 10^{-5}$ Pa s. Selecting the characteristic length, $L_R = 0.05715$ m and using the properties given, corresponding non-dimensional Reynolds and Froude numbers are 42792 and 1, respectively.

Fig. 5.3 shows the comparison between the numerically computed surge front position and water column height and experimental results reported by Martin and Moyce [119]. Initial coarse grid has the characteristic length, $h = 1/4$ which gives $K = 172$. Local polynomial order of 3 is used in all simulations. Increasing the adaptive level, non-dimensional height of the water column corresponds very well with the experimental data. Numerical results show that surge front position moves faster when the resolution near the interface increased. A thin fluid layer forms at the bottom wall just in front of the bulk flow and adaptive

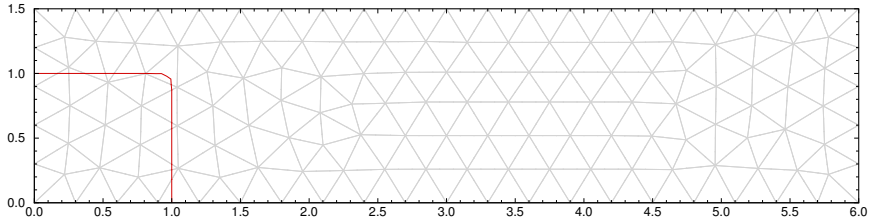


(a) Surge front position (b) Water column height

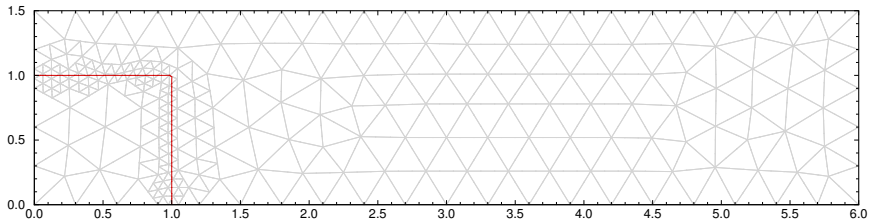
Figure 5.3: Comparison of the present numerical method with different refinement levels and experimental results from Martin and Moyce [119]. ($Re = 42792$, $Fr = 1$, $N = 3$)

contouring algorithm accurately locate the position of the front according to the thin layer. Difficulty to determine the exact location of the leading edge position is confirmed in experimental study [119] and the same tendency is shown in other numerical works [188, 95].

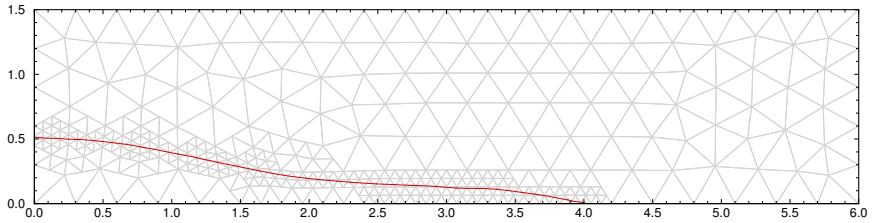
Fig. 5.4 illustrates the interface shapes and mesh structures. (a) and (b) show initial coarse grid and interface for $K = 242$, $N = 3$ and Lagrange interpolation of the initial data to $l_M = 2$ locally adapted grid, respectively. (c-d) represents the interface at $t = 2.5$ for adaptive simulation and 2 level globally refined fixed grid. A careful investigation of the figure reveals the consistency of adaptive simulation where varying mesh resolution and non-conformal discretization do not degenerate the accuracy near the interface. It is worthwhile to mention that total number of elements in fixed grid is 3904 and adaptive grid has 610 elements at that instant. Computational effort is significantly reduced to obtain the same accuracy with fixed grid.



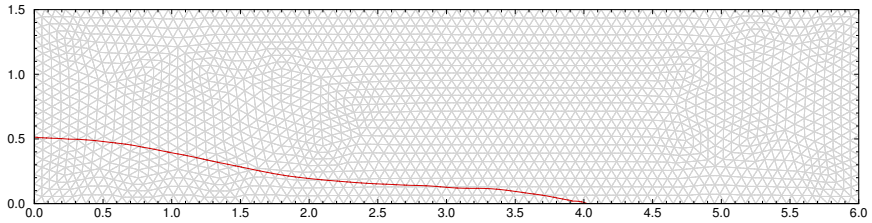
(a) Coarse grid, $t = 0$



(b) $l_M = 2$ adaptive grid, $t = 0$



(c) $l_M = 2$ adaptive grid, $t = 2.5$



(d) Fixed grid, $t = 2.5$

Figure 5.4: Dam break problem interface locations and mesh structures. ($Re = 42792$, $Fr = 1$, $N = 3$)

5.3.3 Rayleigh-Taylor instability

Rayleigh-Taylor instability is a common test problem to show to performance of the numerical methods in complex flows. In the problem, a heavy fluid overlies a layer of light fluid. Instability arises when the initial interface is perturbed. Due to vertical gravitational field, fluids start penetrating into each other with increasing amplitude in time.

Problem is solved in rectangular domain of $[0, d/2] \times [0, 4d]$. Boundary conditions

are slip at side walls, no-slip at bottom wall and prescribed zero pressure at top wall. Initial conditions are taken as zero velocity field and hydrostatic pressure distribution. Interface between the fluids is initially perturbed with a cosine function of amplitude 0.1 , $\eta = 2.0 + 0.1\cos(2\pi x)$. Dynamic viscosity of the fluids are equal and density difference is represented by the Atwood ratio, $\mathrm{At} = (\rho_l - \rho_g)/(\rho_l + \rho_g)$. To use the same notation with [186], reference time is chosen as $t_r = \sqrt{d/(\mathrm{At}g)}$.

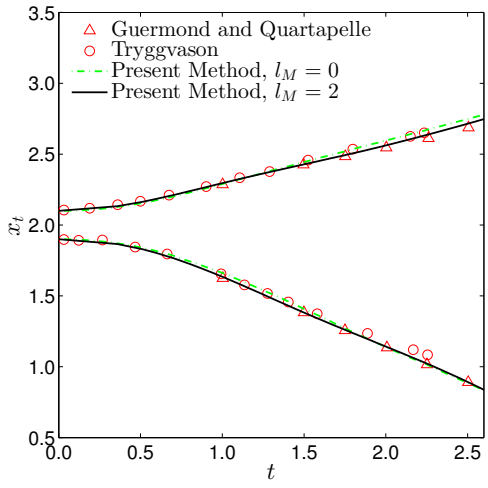


Figure 5.5: Tip positions of the rising and dropping fluids for Rayleigh Taylor instability. ($\mathrm{Re} = 42792$, $\mathrm{At} = 0.5$, $N = 3$)

Non-dimensional tip position of the rising and dropping fluid columns, x_t are represented in Fig. 5.5 for $\mathrm{Re} = \rho_l d^{3/2} g^{1/2}/\mu_l = 3000$ and $\mathrm{At} = 0.5$. Numerical results are carried out for fixed grid having characteristic length, $h = 1/3$ ($K = 212$) and $l_M = 2$ locally adapted grid. For the comparison, results obtained by the Lagrangian-Eulerian vortex method [186] and the variable density finite element projection method [73] are also included. Inspection of the figure reveals that present results compare well with the previous works and increasing resolution near the interface improves the accuracy.

Fig. 5.6 illustrate the interface shapes for $l_M = 2$ locally adapted grid at different simulation times. Numerical results compare well with those reported by Puckett et al. [145], Popinet and Zaleski [142] and Xie et al.[198]. Coarse grid, $l_M = 1$ and $l_M = 2$ solutions at $t = 1.163$ are also shown in Fig. 5.7 . Comparison of the figure reveals that the present numerical approach with local

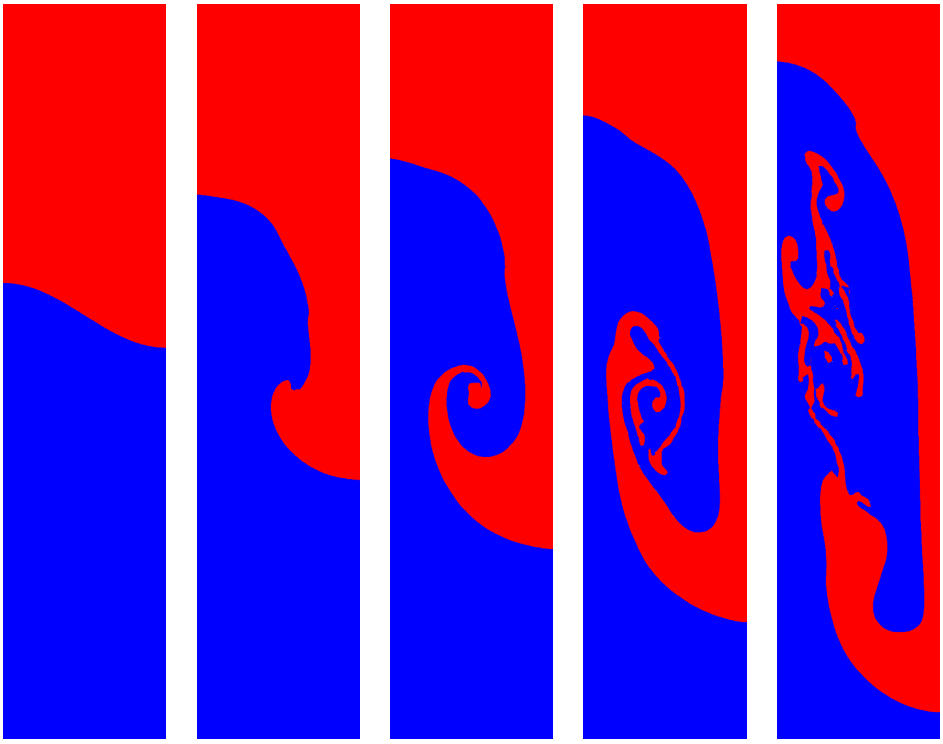


Figure 5.6: Interface evolution for the Rayleigh-Taylor instability for $l_M = 2$ adaptive solutions at $t = 0$, 1.28, 1.71, 2.18, 2.70. (Re = 3000, At = 0.5, $N = 5$, Only a part of the domain is shown.)

unstructured adaptivity can capture complex interfaces efficiently without increasing the number of elements significantly. At that instant, element numbers in $l_M = 2$ and $l_M = 1$ grids are 425 and 305, respectively.

Percentage mass fluctuations in Rayleigh-Taylor instability is summarized in Table 5.3. Mass loss is computed as in the sloshing problem. Starting with the same initial coarse grid , simulations are conducted for different order of approximations and refinement levels up to 2. Results obtained for the same problem with front tracking method [142], VOF [145] and stabilized finite element-discontinuous level set formulation (FEM-DG-LS) [118] are also included in the table. In adaptive simulations, time average of the element numbers are used for the comparison with the other methods. Dynamic grid improved the mass loss problem significantly and very accurate results are obtained with increasing refinement level. It is worthwhile the mention that degree of freedom for the finest solution ($l_M = 2, N = 5$) is very close to reference works but lower mass

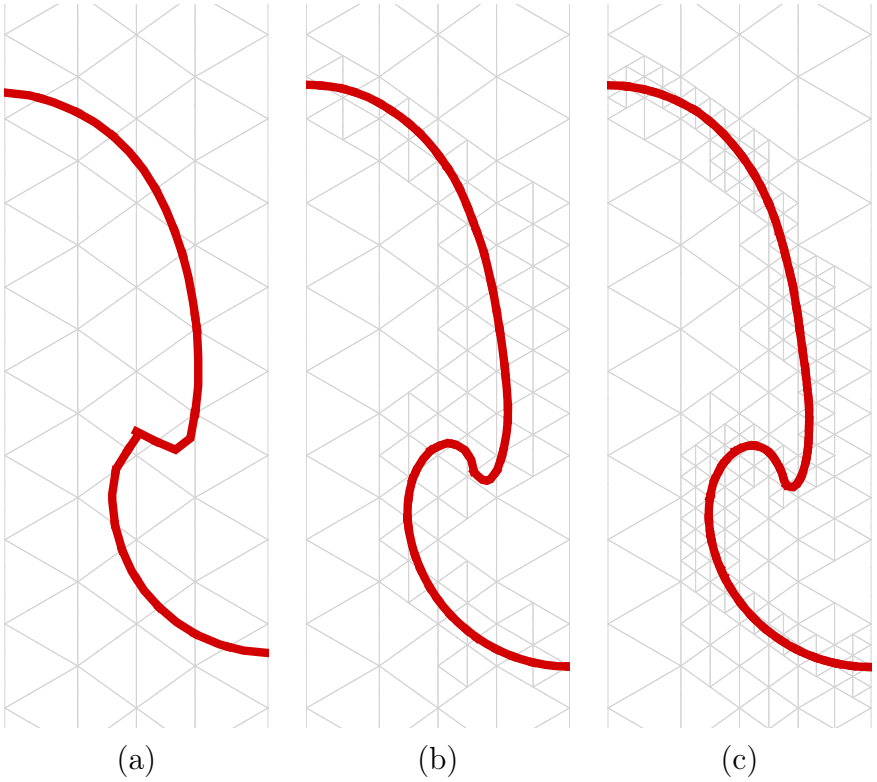


Figure 5.7: Interface shapes for the Rayleigh-Taylor instability at (a) fixed grid (b) $l_M = 1$ and (c) $l_M = 2$ locally adapted grids. ($t = 1.63$, Re $= 500$, At $= 0.5$, $N = 3$, Only a part of the domain is shown.)

loss is achieved.

5.3.4 3D Dam Break with Obstacle

To show the ability of numerical method on 3D, complex interface problems, a dam break problem with rectangular obstacle is solved. For direct comparison with the experimental study of Maritime Research Institute Netherlands (MARIN) [168], computational domain is described with the size of 3.22 $m \times 1\ m \times 1\ m$. Rectangular obstacle having the dimensions of 0.161 $m \times 0.403\ m \times 0.161\ m$ is placed 2.476 m downstream the water column. Rectangular water column with height and width of 0.55 m and 1.228 m is released at $t = 0$. Rectangular liquid column collapses under the action of gravitational acceleration which creates highly unsteady flow field and complex interface deformations. Fig. 5.8 illustrates the problem geometry and initial configuration.

Table 5.3: Percentage maximum mass fluctuations in Rayleigh-Taylor instability for different refinement levels and order of approximations.(Re = 500, At = 0.5)

Method	l_M	N	Element Number	% Mass Loss
Present Method	0	3	212	0.259
		5	212	0.148
		3	836	0.093
		5	836	0.032
	1	3	296	0.096
		5	294	0.037
	2	3	490	0.025
		5	482	0.0091
Front Tracking [142]	-	-	32×265	0.14
FEM-DG-LS [118]	-	1	32×265	0.17

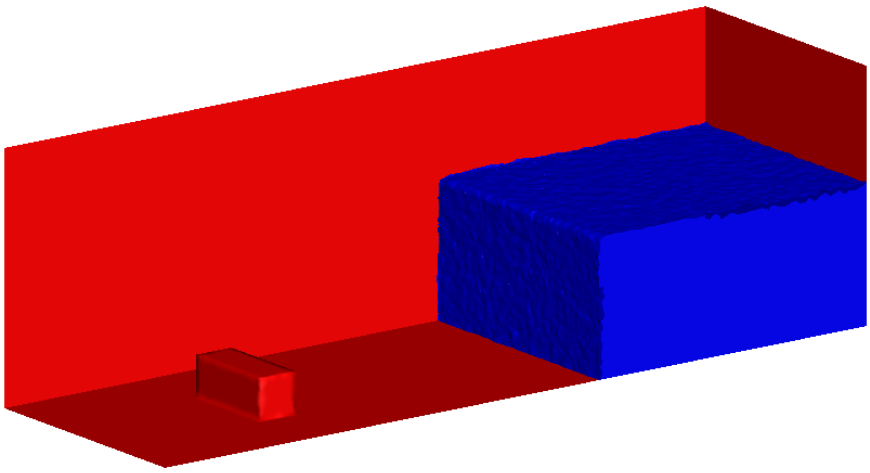
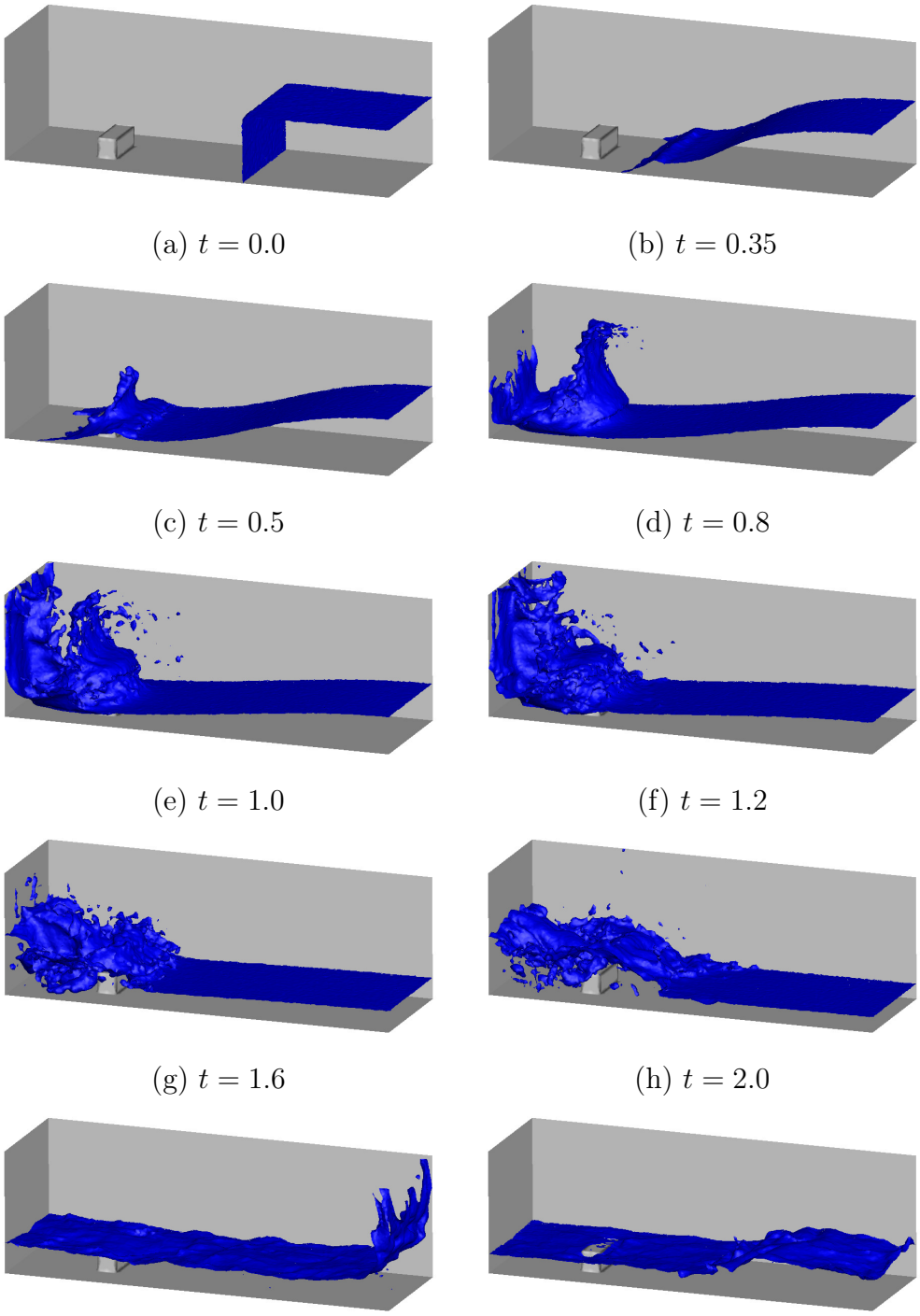


Figure 5.8: Problem domain and initial water column for 3D dam break test.

Slip boundary conditions are applied to the bottom and side walls of the domain. Top wall is modeled as open boundary with zero normal gradients on the velocities. Water and air properties are assigned to the liquid and gas phases as $\rho_l = 1000$ kg m^{-3}, $\rho_g = 1$ kg m^{-3} and $\mu_l = 10^{-3}$ Pa s, $\mu_g = 10^{-5}$ Pa s. Problem is solved on fixed tetrahedral grid with characteristic length of 0.1 where $K \approx 40000$.

Fig. 5.9 shows snapshots of interface at different solution times. The figure illustrates capability of the present numerical method in capturing highly deformable interfaces having strong topological changes.

In Fig. 5.10 interface shape at $t = 0.56$ s, just after first impact with the obstacle, is shown. For the qualitative comparison, volume of fluid solution [97] and



(a) $t = 0.0$ (b) $t = 0.35$

(c) $t = 0.5$ (d) $t = 0.8$

(e) $t = 1.0$ (f) $t = 1.2$

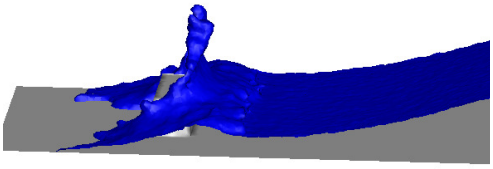
(g) $t = 1.6$ (h) $t = 2.0$

(i) $t = 3.6$ (j) $t = 4.4$

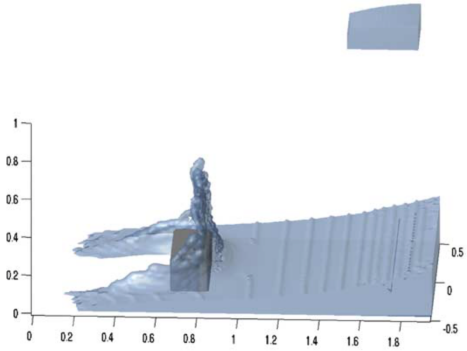
Figure 5.9: Snopshots of interface topolgy for 3D broken dam problem for $N = 3$ and $h = 0.1$.

experimental snapshot at the same instant of time are also included. There is a good agreement between present study and the reference works in terms

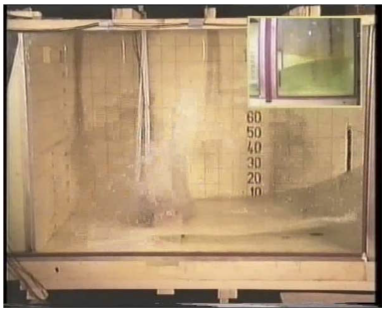
of interface shape and impact time. In the VOF solution, however, there are some ripples, which are not shown in the present study and experiment. This is probably due to low order reconstruction scheme used in the VOF formulation of the reference work [97].



(a) Present solution.



(b) Kleefsman's VOF solution [97].

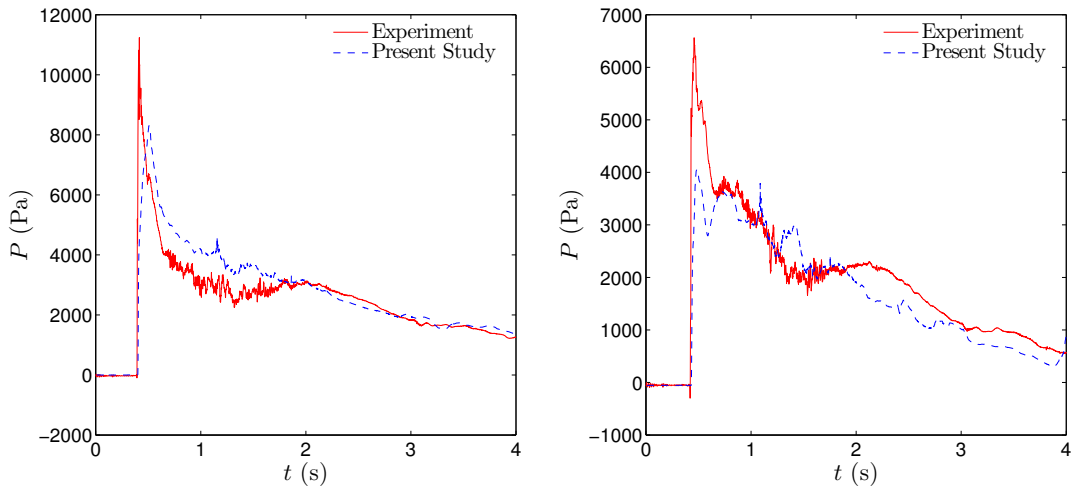


(c) Experiment [168].

Figure 5.10: Comparasion of interface shape with Kleefsman's VOF solution [97] and experimental study [168] at $t = 0.56\ s$.

In the experimental work of MARIN [168], pressure histories on the rectangular obstacle are measured by eight sensors, four located on the front wall and four on the top wall . Please refer to [168], for the details of experimental setup. Fig. 5.11

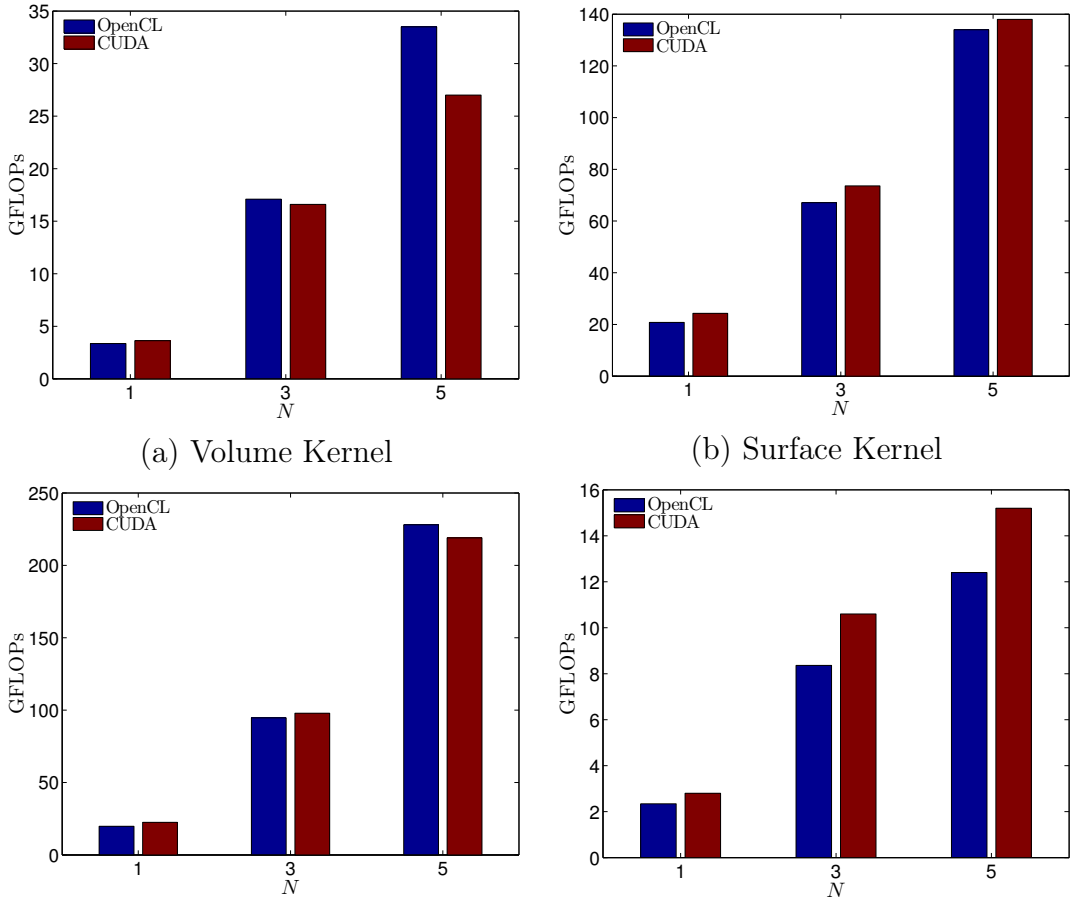
presents computed pressure history on the first and third sensor location. The instant that water wave hits the obstacle is measured as $t = 0.4\ s$ which is well approximated by the numerical method. Simulation and the experiment show very good agreement except the magnitude of impact pressure which is under-predicted by the numerical method. This difference is observed in other simulations using different numerical techniques such as VOF [97] and Smooth Particle Hydrodynamics (SPH) [103].



(a) Prope location 1 (b) Prope location 3

Figure 5.11: Comparasion of computed pressure histories with experimental results [168].

Fig. 5.12 presents double precision floating point operations for 3D advection step kernels when cross-compiled with CUDA and OpenCL multi-threading models on Tesla C2075 GPU. Advection step includes volume, surface, source and time step update computations which are computed with corresponding kernels. All performance numbers are obtained using the wall clock time from the beginning of one time step to the next one and averaged over a few hundred samples to minimize the timing transients. Main computational load of advection step comes from the volume and surface term computations. All kernels at polynomial order of 1 gives lower performance due to lower computational load. Similar performance results are obtained for both multi-threading models and for all kernels.



(a) Volume Kernel

(b) Surface Kernel

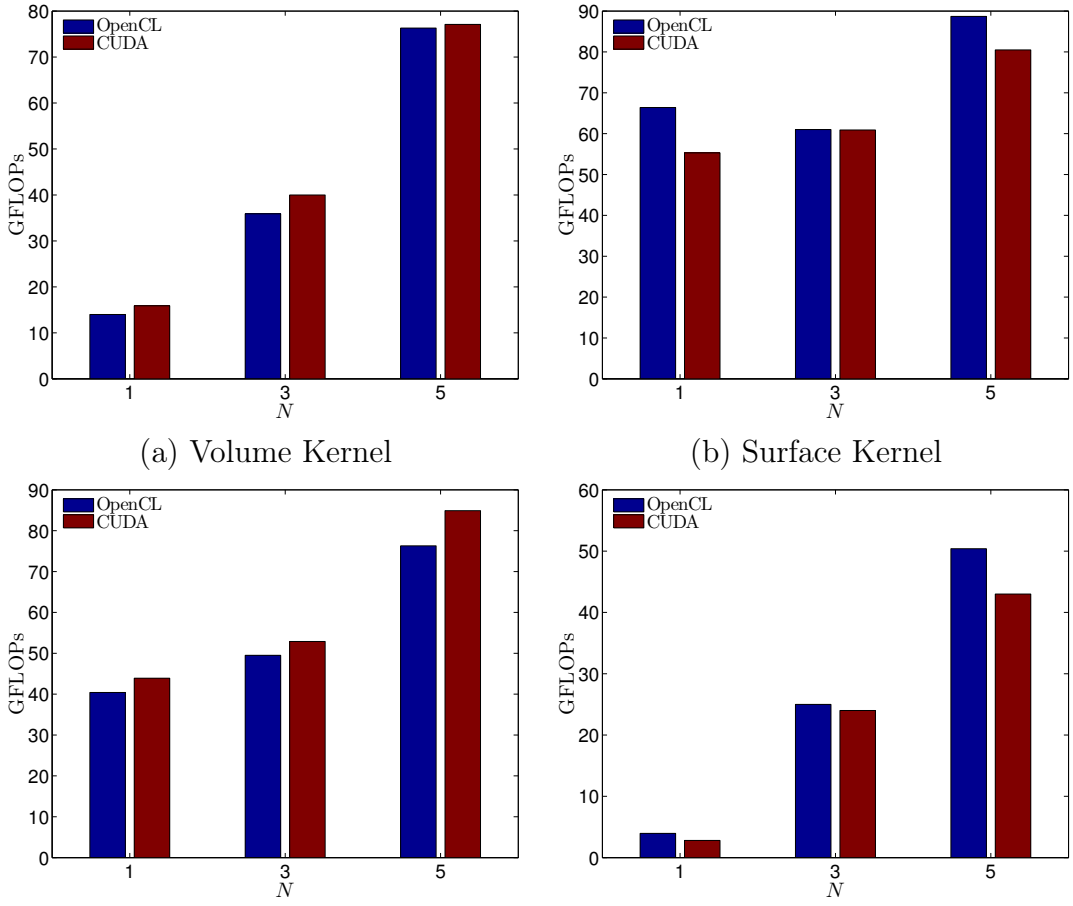
(c) Source Kernel

(d) Update Kernel

Figure 5.12: Double precision GFLOPs of advection step kernels vs polynomial order on Tesla C2075 GPU using CUDA and OpenCL multi-threading models.

Fig. 5.13 shows double precision floating point operations for pressure step kernels including volume, surface, right-hand side and update computations. Similar to the advection step, kernels are compiled with CUDA and OpenCL multi-threading models on Tesla C2075 GPU. Volume kernel at polynomial order of 1 gives low performance due to low computational load. Surface and right-hand side kernels perform well in all orders due to higher computational load. OpenCL slightly outperforms CUDA on update kernel for all orders.

Finally, performance of velocity step kernels are illustrated in Fig. 5.14. Structure of velocity step computations are similar to pressure step leading to comparable kernel performances. Computational load of the velocity volume kernel is higher than the pressure volume kernel due to additional scaled mass matrix.



(a) Volume Kernel

(b) Surface Kernel

(c) RHS Kernel

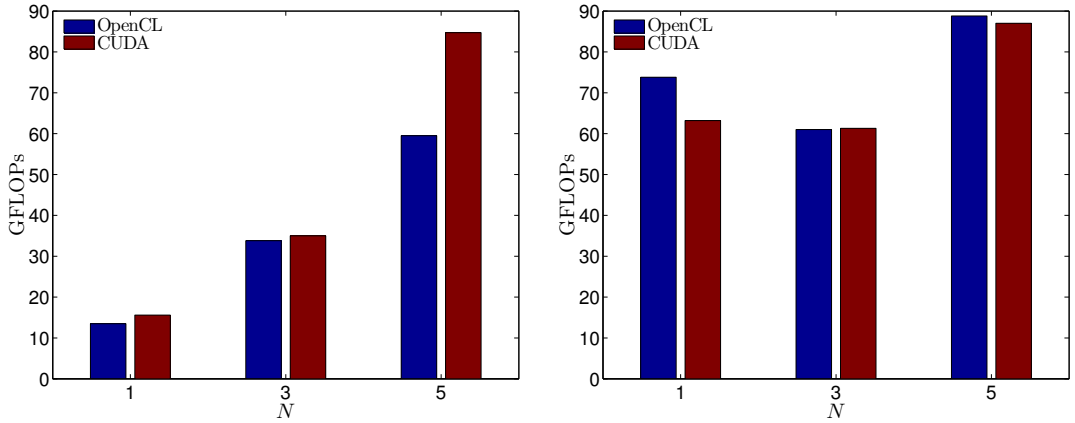
(d) Update Kernel

Figure 5.13: Double precision GFLOPs of pressure step kernels vs polynomial order on Tesla C2075 GPU using CUDA and OpenCL multi-threading models.

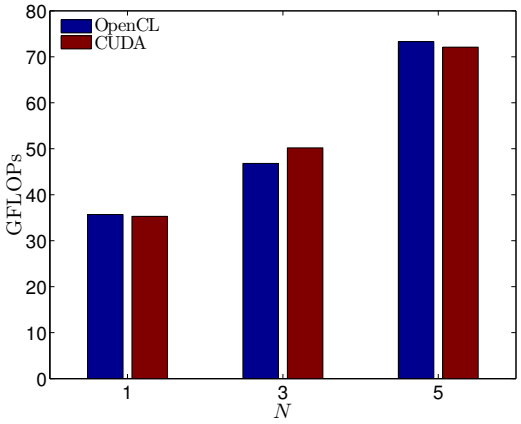
CUDA seems to be more efficient on volume kernel unlike the pressure step. Because CUDA ptx compiler can be hardware optimized, CUDA outperforms OpenCL on NVIDIA GPU specially at higher-order polynomial approximations.

5.4 Conclusion

We presented a high-order, fully discontinuous Galerkin method for the incompressible multiphase flows on unstructured adaptive meshes. With the proposed numerical framework, the mass is well conserved even in the coarse grid solutions without introducing any special treatment or modification of the level set formulation. Velocity, pressure and interface modeling uses the same high order



(a) Volume Kernel (b) Surface Kernel



(c) RHS Kernel

Figure 5.14: Double precision GFLOPs of velocity step kernels vs polynomial order on Tesla C2075 GPU using CUDA and OpenCL multi-threading models.

polynomial space preventing the interpolation of the field variables and increase the efficiency. All implicit systems are solved with a matrix-free, p-multigrid approach which reduces the memory requirements. In the adaptive method, the computation is localized mostly near the moving objects; thus, the computational cost is significantly reduced compared with the uniform mesh over the whole domain with the same resolution. Platform independence of the solver is achieved with an extensible multi-threading programming API as common kernel language. The developed solver is highly scalable on many-core architectures.

CHAPTER 6

CONCLUSIONS AND FUTURE WORKS

This study focuses on development of high-order discontinuous Galerkin methods for the solution of incompressible multiphase flows with sharp interfaces. Developed methods are designed for efficient solutions on massively parallel modern architectures, namely many-core CPUs and multi-threaded GPUs. Accuracy and efficiency of the method is further increased using a local non-conformal adaptivity combined with a multi-rate time integration.

In multiphase flows, most of the complexities are encountered around the interface. Rapid fluid property changes lead to spurious velocity oscillations. Strong vorticity is concentrated in the vicinity of interface. Under-resolution in the interface model may lead to unphysical mass loss. Those problems become more critical when density/viscosity ratios are high and the interface has topological changes. To overcome the mentioned problems, an adaptive local interface model is presented based on discontinuous elements. Transient adaptivity uses non-conformal discretizations which enable to get fast refinement/coarsening operations in a truly local manner where parallelization follows. Another important property of the adaptive scheme is to preserve high-order accuracy in interpolation of field properties between coarse to fine and fine to coarse grids. It is well known that, this is one of the main drawbacks of conformal and anisotropic refinement methods. Also, forcing conformity and optimizing mesh after adaptation increases computational time and decreases parallelization i.e. these operations require propagation of local information through the domain. However, non-conformal refinement shifts computational load from adaptation

step to solver by increasing the complexity in flux evaluations. An efficient and local non-conformal flux evaluation technique is also developed which decreases computational time substantially.

Level set method is very powerful to represent interface dynamics due its implicit nature. However, evolution of level set equations in time often distorts the scalar level set function and generates flat or steep gradients near the interface. Losing the regularity of level set function may lead to numerical instabilities, large errors in evaluation of material properties and geometric information etc. , so that level set function has to be replaced with more regular functions. This process is called as reinitialization and plays an important role in level set interface modeling. In this study, a high-order, fully explicit level set reinitialization algorithm is introduced. Stabilization of the system is achieved with artificial diffusion mechanism which does not directly reduce the approximation to first order. Stabilization technique uses constant, possibly discontinuous and locally vanishing viscosity through elements. Diffusion equation is discretized directly and handles discontinuous viscosity without introducing any regularization. This treatment accelerates the computations and requires no additional information. For the detection of troubled elements that stabilization is needed for, the modal coefficient based regularity detector proposed in [98] is generalized to higher-order dimensions. Numerical test shows that this detector is quite successful in tetrahedron and triangle even for the low order approximations where small number of modes exist to predict the regularity within an element. Proposed reinitialization algorithm preserves the optimum convergence rates. More specifically, $(N+1)^{th}$ and 2^{nd} order accuracy are achieved for smooth and non-smooth solutions for the N order discontinues approximation space. The main advantage of the proposed scheme relies on its potential of parallelization on multi-threaded architectures.

It is clear that increasing resolution near the interface improves the capability of model even for high-order and well resolved problems. Artificial diffusion stabilization offers a stable and local mechanism to damp out the oscillations around troubled regions. But, both local adaptivity and artificial diffusion restrict time step size of the method when a global explicit integrator is used. To avoid this

strict time limitation, a multi-rate Adams-Bashforth time integrator is designed for the explicit treatment of level set advection and reinitialization equations. The local time-stepper does not require extrapolation of field histories and any additional storage requirements unlike many other proposed methods in literature. All these properties make the scheme fast, memory efficient and suitable for multi-threaded architectures. Numerical tests have shown that scheme preserves the base Adams-Bashforth order away from the multi-rate coupling and second order accuracy at groups interface.

A fully discontinuous Galerkin level set based multiphase flow solver on adaptive triangular/tetrahedral grids are proposed. A semi-implicit method is used for time discretization which decouples velocity and pressure fields. Numerical test and eigenvalue analysis indicates that standard discretization schemes do not enforce discrete incompressibility for high density ratios. Divergence-free velocity field is obtained by projecting intermediate velocity to $H(\mathrm{div})$ polynomial space. This enables getting a stable numerical scheme even in the very high material property ratios up to 1000. Another important aspect of the fully discontinuous scheme is to solve bad conditioned implicit pressure system. Condition numbers are highly dependent to density ratios between the fluids, not to interface topology. This complex system is solved efficiently with a developed matrix-free algebraic multigrid scheme derived from the full matrix based method proposed in [64]. The most important contribution of the multigrid method is reducing memory requirement and set-up cost considerably which are critical for the efficient implementation on GPUs for highly dynamic problems where spectral properties of the pressure system changes fast. Combined with the local adaptivity and previously developed interface model, a mass loss free numerical scheme preserving the simplicity and efficiency of level set formulation is obtained without using any special treatment or tricks.

All numerical algorithms mentioned so far are trivially parallelizable on modern CPU and GPU architectures. Developed solver is coded in C++ and OCCA multi-threading API. OCCA is an abstracted programming model used to encapsulate native languages for parallel devices such as CUDA, OpenCL, Pthreads and OpenMP. Therefore, OCCA allows customized implementations of algo-

rithms for several computing devices with a single code and offers flexibility to choose hardware architectures and programming model at run-time. It is shown that the developed solver is highly scalable on all platforms and programming models, and reach almost peak performances of hardwares. Speedups factors reaching two-order of magnitude are obtained comparing the serial CPU implementations.

As future works, efficient adaptation techniques on GPUs will be considered to accelerate refinement-coarsening time and to avoid the data transfer between host and device. Generally speaking, multiphase flows need finer grids for the interface model and coarser resolution for flow solver. An ungraded dual grid method based on separating fluid and interface grids will be developed to increase accuracy of the interface representation without using finer grids for flow equations and/or mesh adaptation. This method may also utilize different numerical techniques for the discretization of flow and interface equations such as continuous spectral or finite elements for flow evolution and discontinuous Galerkin for interface evolution with different approximation orders. Accurate high-order computation of surface tension forces on discontinuous framework will also be a future work.

REFERENCES

[1] M. Ainsworth. Dispersive and dissipative behaviour of high order discontinuous Galerkin finite element methods. *Journal of Computational Physics*, 198(1):106–130, 2004.

[2] W. Aniszewski, T. Ménard, and M. Marek. Volume of Fluid (VOF) type advection methods in two-phase flow: A comparative study. *Computers & Fluids*, 97:52–73, 2014.

[3] D. Arnold. An Interior Penalty Finite Element Method with Discontinuous Elements. *SIAM J. Numer. Anal.*, 19(4):742–760, 1982.

[4] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified Analysis of Discontinuous Galerkin Methods for Elliptic Problems. *SIAM Journal on Numerical Analysis*, 39(5):1749–1779, 2002.

[5] N. Ashgriz and J. Poo. FLAIR: flux line-segment model for advection and interface reconstruction. *Journal of Computational Physics*, 93(2):449–468, 1991.

[6] I. Babuska, C. E. Baumann, and J. T. Oden. A discontinuous hp finite element method for diffusion problems: 1-D analysis. *Computers & Mathematics with Applications*, 37(9):103–122, 1999.

[7] G. A. Baker, W. N. Jureidini, and O. A. Karakashian. Piecewise solenoidal vector fields and the stokes problem. *SIAM Journal on Numerical Analysis*, 27(6):1466–1485, 1990.

[8] G. E. Barter and D. L. Darmofal. Shock capturing with PDE-based artificial viscosity for DGFEM: Part I. Formulation. *Journal of Computational Physics*, 229(5):1810–1827, 2010.

[9] F. Bassi, A. Crivellini, D. Di Pietro, and S. Rebay. An artificial compressibility flux for the discontinuous Galerkin solution of the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 218(2):794–815, 2006.

[10] F. Bassi and S. Rebay. A High-Order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *Journal of Computational Physics*, 131(2):267–279, 1997.

[11] C. E. Baumann and J. T. Oden. A discontinuous hp finite element method for convection–diffusion problems. *Computer Methods in Applied Mechanics and Engineering*, 175(3-4):311–341, 1999.

[12] M. G. Blyth and C. Pozrikidis. A lobatto interpolation grid over the triangle. *IMA Journal of Applied Mathematics*, 71(1):153 –169, 2006.

[13] L. Bos. Bounding the lebesgue function for lagrange interpolation in a simplex. *Journal of Approximation Theory*, 38(1):43–59, 1983.

[14] L. Bos, M. A. Taylor, and B. A. Wingate. Tensor product Gauss-Lobatto points are fekete points for the cube. *Mathematics of Computation*, 70(236):1543–1547., 2000.

[15] F. Brezzi, B. Cockburn, L. Marini, and E. Süli. Stabilization mechanisms in discontinuous Galerkin finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 195(25-28):3293–3310, 2006.

[16] F. Brezzi, L. D. Marini, and E. Süli. Discontinuous Galerkin methods for first-order hyperbolic problems. *MATH. MODELS METHODS APPL. SCI*, 14, 2004.

[17] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: Stream Computing on Graphics Hardware. *ACM TRANSACTIONS ON GRAPHICS*, 23:777–786, 2004.

[18] E. Burman and B. Stamm. Minimal stabilization for discontinuous Galerkin finite element methods for hyperbolic problems. *Journal of Scientific Computing*, 33(2):183–208, 2007.

[19] J. Carrero, B. Cockburn, and D. Schötzau. Hybridized globally divergence-free LDG methods. i. the stokes problem. *Mathematics of Computation*, 75:533–563, 2006.

[20] P. Castillo. Performance of discontinuous Galerkin methods for elliptic PDEs. *SIAM Journal on Scientific Computing*, 24(2):524, 2002.

[21] T. C. Cecil, S. J. Osher, and J. Qian. Simplex free adaptive tree fast sweeping and evolution methods for solving level set equations in arbitrary dimension. *Journal of Computational Physics*, 213(2):458–473, 2006.

[22] T. C. Cecil, S. J. Osher, and J. Qian. Essentially non-oscillatory adaptive tree methods. *Journal of Scientific Computing*, 35(1):25–41, 2008.

[23] G. Chavent and B. Cockburn. The local projection $P_0 - P_1$-discontinuous-Galerkin finite element method for scalar conservation laws, 1989.

[24] G. Chavent and G. Salzano. A finite-element method for the 1-D water flooding problem with gravity. *Journal of Computational Physics*, 45(3):307–344, 1982.

[25] Q. Chen and I. Babuška. Approximate optimal points for polynomial interpolation of real functions in an interval and in a triangle. *Computer Methods in Applied Mechanics and Engineering*, 128(3-4):405–417, 1995.

[26] L.-T. Cheng and Y.-H. Tsai. Redistancing by flow of time dependent eikonal equation. *Journal of Computational Physics*, 227(8):4002–4017, 2008.

[27] A. J. Chorin. Flame advection and propagation algorithms. *Journal of Computational Physics*, 35(1):1–11, 1980.

[28] B. Cockburn. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II : General framework. *Math. Comput.*, 52:411–435, 1989.

[29] B. Cockburn and B. Dong. An analysis of the minimal dissipation local discontinuous Galerkinn method for Convection–Diffusion problems. *Journal of Scientific Computing*, 32(2):233–262, 2007.

[30] B. Cockburn, B. Dong, and J. Guzman. A superconvergent LDG-hybridizable Galerkin method for second-order elliptic problems. *Mathematics of Computation*, 77:1887–1916, 2008.

[31] B. Cockburn, J. Gopalakrishnan, and R. Lazarov. Unified hybridization of discontinuous Galerkin, mixed, and continuous Galerkin methods for second order elliptic problems. *SIAM Journal on Numerical Analysis*, 47(2):1319, 2009.

[32] B. Cockburn, J. Gopalakrishnan, N. C. Nguyen, J. Peraire, and F. Sayas. Analysis of HDG methods for stokes flow. *Mathematics of Computation*, 80(274):723–723, 2011.

[33] B. Cockburn and J. Guzmán. Error estimates for the Runge-Kutta discontinuous Galerkin method for the transport equation with discontinuous initial data. *SIAM Journal on Numerical Analysis*, 46(3):1364–1398, 2008.

[34] B. Cockburn, J. Guzmán, and H. Wang. Superconvergent discontinuous Galerkin methods for Second-Order elliptic problems. *Math. Comput.*, 78:1–24, 2009.

[35] B. Cockburn, S. Hou, and C. Shu. The Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws. IV: the multidimensional case. *Mathematics of Computation*, 54(190):545–581, 1990.

[36] B. Cockburn, G. Kanschat, and D. Schotzau. A locally conservative LDG method for the incompressible Navier-Stokes equations. *Mathematics of Computation*, 74:1067–1095, 2005.

[37] B. Cockburn, G. Kanschat, and D. Schötzau. The local discontinuous Galerkin method for the oseen equations. *Mathematics of Computation*, 73(246):569–593, 2004.

[38] B. Cockburn, G. Kanschat, and D. Schötzau. The local discontinuous Galerkin method for linearized incompressible fluid flow: a review. *Computers & Fluids*, 34(4-5):491–506, 2005.

[39] B. Cockburn, G. Kanschat, D. Schötzau, and C. Schwab. Local discontinuous Galerkin methods for the stokes system. *SIAM Journal on Numerical Analysis*, 40(1):319–343, 2002.

[40] B. Cockburn, S. Lin, and C. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems. *Journal of Computational Physics*, 84(1):90–113, 1989.

[41] B. Cockburn, M. Luskin, C. Shu, and E. Suli. Enhanced accuracy by post-processing for finite element methods for hyperbolic equations. *Mathematics of Computation*, 72:577–606, 2003.

[42] B. Cockburn, N. Nguyen, and J. Peraire. A comparison of HDG methods for stokes flow. *Journal of Scientific Computing*, 45(1-3):215–237, 2010.

[43] B. Cockburn and C. Shu. The Runge-Kutta local projection discontinuous-Galerkin finite element method for scalar conservation laws, 1991.

[44] B. Cockburn and C. Shu. The Runge-Kutta discontinuous Galerkin method for conservation laws v: Multidimensional systems. *Journal of Computational Physics*, 141(2):199–224, 1998.

[45] B. Cockburn and C.-W. Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, 1998.

[46] B. Cockburn and C.-W. Shu. Runge-Kutta discontinuous Galerkin methods for convection dominated problems. *Journal of Scientific Computing*, 16(3):173–261, 2001.

[47] E. M. Constantinescu and A. Sandu. Multirate timestepping methods for hyperbolic conservation laws. *J Sci Comput*, 33(3):239–278, 2007.

[48] R. Cools. Monomial cubature rules since “Stroud”: a compilation — part 2. *Journal of Computational and Applied Mathematics*, 112(1–2):21–27, 1999.

[49] R. Cools and P. Rabinowitz. Monomial cubature rules since "Stroud": a compilation. *Journal of Computational and Applied Mathematics*, 48(3):309–326, 1993.

[50] S. Curtis, R. M. Kirby, J. K. Ryan, and C. Shu. Post-Processing for the discontinuous Galerkin method over Non-Uniform meshes. *SIAM Journal on Scientific Computing*, 30:272—289, 2007.

[51] B. J. Daly. A technique for including surface tension effects in hydrodynamic calculations. *Journal of Computational Physics*, 4(1):97–117, 1969.

[52] C. Dawson and R. Kirby. High resolution schemes for conservation laws with locally varying time steps. *SIAM J. Sci. Comput.*, 22(6):2256–2281, 2001.

[53] C. Dawson, C. J. Trahan, E. J. Kubatko, and J. J. Westerink. A parallel local timestepping runge–kutta discontinuous galerkin method with applications to coastal ocean modeling. *Computer Methods in Applied Mechanics and Engineering*, 259:154–165, 2013.

[54] D. Di Pietro, A. Ern, and J. Guermond. Discontinuous galerkin methods for anisotropic semidefinite diffusion with advection. *SIAM J. Numer. Anal.*, 46(2):805–831, 2008.

[55] M. Dubiner. Spectral methods on triangles and other domains. *Journal of Scientific Computing*, 6:345–390, 1991.

[56] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[57] D. Enright, F. Losasso, and R. Fedkiw. A fast and accurate semi-Lagrangian particle level set method. *Computers and Structures*, 83(6-7):479–490, 2005.

[58] A. Ern and J. L. Guermond. Discontinuous Galerkin methods for friedrichs' systems. i. general theory. *SIAM Journal on Numerical Analysis*, 44(2):753, 2006.

[59] C. Eskilsson. An hp-adaptive discontinuous Galerkin method for shallow water flows. *International Journal for Numerical Methods in Fluids*, 67(11):1605–1623, 2011.

[60] E. Ferrer and R. Willden. A high order discontinuous Galerkin finite element solver for the incompressible Navier-Stokes equations. *Computers & Fluids*, 46(1):224–230, 2011.

[61] S. B. François Bodin. Heterogeneous multicore parallel programming for graphics processing units. *Scientific Programming*, 17(4):325–336, 2009.

[62] M. Fuhry, A. Giuliani, and L. Krivodonova. Discontinuous Galerkin methods on graphics processing units for nonlinear hyperbolic conservation laws. *International Journal for Numerical Methods in Fluids*, 76(12):982–1003, 2014.

[63] R. Gandham, D. Medina, and T. Warburton. GPU Accelerated Discontinuous Galerkin Methods for Shallow Water Equations. *Communications in Computational Physics*, 18(01):37–64, 2015.

[64] R. Gandham, D. S. Medina, and T. Warburton. GPU accelerated discontinuous galerkin methods for shallow water equations. *arXiv:1403.1661 [math]*, 2014.

[65] G. J. Gassner, F. Larcher, C. Munz, and J. S. Hesthaven. Polymorphic nodal elements and their application in discontinuous Galerkin methods. *Journal of Computational Physics*, 228(5):1573–1590, 2009.

[66] D. Gaudlitz and N. A. Adams. On improving mass-conservation properties of the hybrid particle-level-set method. *Computers & Fluids*, 37(10):1320–1331, 2008.

[67] C. W. Gear and D. R. Wells. Multirate linear multistep methods. *BIT Numerical Mathematics*, 24(4):484–502, 1984.

[68] E. H. Georgoulis and E. Süli. Optimal error estimates for the hp-version interior penalty discontinuous Galerkin finite element method. *IMA Journal of Numerical Analysis*, 25(1):205 –220, 2005.

[69] D. Gerlach, G. Tomar, G. Biswas, and F. Durst. Comparison of volume-of-fluid methods for surface tension-dominant two-phase flows. *International Journal of Heat and Mass Transfer*, 49(3-4):740–754, 2006.

[70] J. Grooss and J. Hesthaven. A level set discontinuous Galerkin method for free surface flows. *Computer Methods in Applied Mechanics and Engineering*, 195(25-28):3406–3429, 2006.

[71] M. J. Grote and T. Mitkova. High-order explicit local time-stepping methods for damped wave equations. *Journal of Computational and Applied Mathematics*, 239:270–289, 2013.

[72] J. L. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195(44–47):6011–6045, 2006.

[73] J. L. Guermond and L. Quartapelle. A projection FEM for variable density incompressible flows. *Journal of Computational Physics*, 165(1):167–188, 2000.

[74] N. Gödel, S. Schomann, T. Warburton, and M. Clemens. GPU Accelerated Adams-Bashforth Multirate Discontinuous Galerkin FEM Simulation of High-Frequency Electromagnetic Fields. *IEEE Transactions on Magnetics*, 46(8):2735–2738, 2010.

[75] F. H. Harlow, A. A. Amsden, and J. R. Nix. Relativistic fluid dynamics calculations with the particle-in-cell technique. *Journal of Computational Physics*, 20(2):119–129, 1976.

[76] F. H. Harlow and J. E. Welch. Numerical Calculation of Time Dependent Viscous Incompressible Flow of Fluid with Free Surface. *Physics of Fluids (1958-1988)*, 8(12):2182–2189, 1965.

[77] W. Heinrichs. Improved lebesgue constants on the triangle. *Journal of Computational Physics*, 207(2):625–638, 2005.

[78] M. Herrmann. A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids. *Journal of Computational Physics*, 227(4):2674–2706, 2008.

[79] J. Hesthaven and C. Teng. Stable Spectral Methods on Tetrahedral Elements. *SIAM Journal on Scientific Computing*, 21(6):2352–2380, 2000.

[80] J. Hesthaven and T. Warburton. Nodal High-Order methods on unstructured grids: I. Time-Domain solution of maxwell's equations. *Journal of Computational Physics*, 181(1):186–221, 2002.

[81] J. S. Hesthaven. From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. *SIAM Journal on Numerical Analysis*, 35:655, 1998.

[82] J. S. Hesthaven and T. Warburton. *Nodal discontinuous Galerkin methods: algorithms, analysis, and applications*. Springer, 2008.

[83] C. Hirt and B. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39(1):201–225, 1981.

[84] C. Hu and C.-W. Shu. Discontinuous Galerkin finite element method for hamilton-jacobi equations. *SIAM Journal on Scientific Computing*, 21(2):666–690, 1999.

[85] W. Hundsdorfer, S. Ruuth, and R. Spiteri. Monotonicity-preserving linear multistep methods. *SIAM J. Numer. Anal.*, 41(2):605–623, 2003.

[86] J. M. Hyman. Numerical methods for tracking interfaces. *Physica D: Nonlinear Phenomena*, 12(1-3):396–407, 1984.

[87] R. A. J. P. Webb. Hierarchical triangular elements using orthogonal polynomials. *International Journal for Numerical Methods in Engineering*, 38(2):245 – 257, 1995.

[88] G.-S. Jiang and D. Peng. Weighted ENO schemes for hamilton-jacobi equations. *SIAM Journal on Scientific Computing*, 21(6):2126–2143, 2000.

[89] C. Johnson and J. Pitkäranta. An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation. *Mathematics of Computation*, 46(173):1–26, 1986.

[90] O. Karakashian and W. Jureidini. A nonconforming finite element method for the stationary navier-stokes equations. *SIAM Journal on Numerical Analysis*, 35(1):93–120, 1998.

[91] A. Karakus, T. Warburton, M. Aksel, and C. Sert. Level set reinitialization for a high order h adaptive discontinuous galerkin method. 2015. Submitted to International Journal of Numerical Methods in Fluids.

[92] G. Karniadakis and S. J. Sherwin. *Spectral/hp element methods for CFD*. Oxford University Press, 2005.

[93] G. E. Karniadakis, M. Israeli, and S. A. Orszag. High-order splitting methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 97(2):414–443, 1991.

[94] C. Kees, I. Akkerman, M. Farthing, and Y. Bazilevs. A conservative level set method suitable for variable-order approximations and unstructured meshes. *Journal of Computational Physics*, 230(12):4536–4558, 2011.

[95] F. J. Kelecy and R. H. Pletcher. The development of a free surface capturing approach for multidimensional free surface flows in closed containers. *Journal of Computational Physics*, 138(2):939–980, 1997.

[96] R. Kirby and S. Sherwin. Aliasing errors due to quadratic nonlinearities on triangular spectral element discretisations. *Journal of Engineering Mathematics*, 56(3):273–288, 2006.

[97] K. M. T. Kleefsman, G. Fekken, A. E. P. Veldman, B. Iwanowski, and B. Buchner. A Volume-of-Fluid based simulation method for wave impact problems. *Journal of Computational Physics*, 206(1):363–393, 2005.

[98] A. Klockner, T. Warburton, and J. S. Hesthaven. Viscous shock capturing in a time-explicit discontinuous Galerkin method. *Mathematical Modelling of Natural Phenomena*, 6(3):57–83, 2011.

[99] A. Klöckner, T. Warburton, J. Bridge, and J. S. Hesthaven. Nodal discontinuous Galerkin methods on graphics processors. *Journal of Computational Physics*, 228(21):7863–7882, 2009.

[100] C. G. Koh, M. Gao, and C. Luo. A new particle method for simulation of incompressible free surface flow problems. *International Journal for Numerical Methods in Engineering*, 89(12):1582–1604, 2012.

[101] T. Koornwinder. Two-variable analogues of the classical orthogonal polynomials. In *Theory and application of special functions*, pages 435–495. Academic Press, 1975.

[102] M. A. Kopera and F. X. Giraldo. Analysis of adaptive mesh refinement for IMEX discontinuous Galerkin solutions of the compressible Euler equations with application to atmospheric simulations. *Journal of Computational Physics*, 275:92–117, 2014.

[103] E.-S. Lee, D. Violeau, R. Issa, and S. Ploix. Application of weakly compressible and truly incompressible SPH to 3-D water collapse in waterworks. *Journal of Hydraulic Research*, 48(sup1):50–60, 2010.

[104] O. Lepsky, C. Hu, and C.-W. Shu. Analysis of the discontinuous Galerkin method for hamilton-jacobi equations. *Applied Numerical Mathematics*, 33(1):423–434, 2000.

[105] P. Lesaint and P. Raviart. On a finite element method to solve the neutron transport equation. *Mathematical Aspects of Finite Elements in Partial Differential Equations*, pages 84–145, 1974.

[106] R. J. Leveque. High-Resolution Conservative Algorithms for Advection in Incompressible Flow. *SIAM Journal on Numerical Analysis*, 33(2):627–665, 1996.

[107] F. Li and C.-W. Shu. Reinterpretation and simplified implementation of a discontinuous Galerkin method for hamilton-jacobi equations. *Applied Mathematics Letters*, 18(11):1204–1209, 2005.

[108] Q. Li, J. Ouyang, B. Yang, and X. Li. Numerical simulation of gas-assisted injection molding using CLSVOF method. *Applied Mathematical Modelling*, 36(5):2262–2274, 2012.

[109] Z. Li, F. A. Jaberi, and T. I.-P. Shih. A hybrid Lagrangian–Eulerian particle-level set method for numerical simulations of two-fluid turbulent flows. *International Journal for Numerical Methods in Fluids*, 56(12):2271–2300, 2008.

[110] Q. Lin and A. Zhou. Convergence of the discontinuous Galerkin method for a scalar hyperbolic equation. *Acta Math.Sci*, 13:207–210, 1993.

[111] J. Liu and C. Shu. A High-Order discontinuous Galerkin method for 2D incompressible flows. *Journal of Computational Physics*, 160(2):577–596, 2000.

[112] F. Losasso, R. Fedkiw, and S. Osher. Spatially adaptive techniques for level set methods and incompressible flow. *Computers & Fluids*, 35(10):995–1010, 2006.

[113] F. Losasso, F. Gibou, and R. Fedkiw. Simulating water and smoke with an octree data structure. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, page 457–462, New York, NY, USA, 2004. ACM.

[114] H. Luo, J. D. Baum, and R. Löhner. A hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids. *Journal of Computational Physics*, 225(1):686–713, 2007.

[115] H. Luo and C. Pozrikidis. A Lobatto interpolation grid in the tetrahedron. *IMA Journal of Applied Mathematics*, 71(2):298–313, 2006.

[116] E. Marchandise, P. Geuzaine, N. Chevaugeon, and J.-F. Remacle. A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics. *Journal of Computational Physics*, 225(1):949–974, 2007.

[117] E. Marchandise, J. Remacle, and N. Chevaugeon. A quadrature-free discontinuous Galerkin method for the level set equation. *Journal of Computational Physics*, 212(1):338–357, 2006.

[118] E. Marchandise and J.-F. Remacle. A stabilized finite element method using a discontinu-

ous level set approach for solving two phase incompressible flows. *Journal of Computational Physics*, 219(2):780–800, 2006.

[119] J. C. Martin and W. J. Moyce. An experimental study of the collapse of liquid columns on a rigid horizontal plane. *Phil. Trans. R. Soc. Lond. A*, 244(882):312–324, 1952.

[120] D. S. Medina. *OKL: A Unified Language for Parallel Architectures*. PhD thesis, Rice University, Houston, TX, 2015.

[121] D. S. Medina, A. St-Cyr, and T. Warburton. OCCA: a unified approach to multi-threading languages. *arXiv:1403.0968 [cs]*, 2014.

[122] C. Min and F. Gibou. A second order accurate level set method on non-graded adaptive cartesian grids. *Journal of Computational Physics*, 225(1):300–321, 2007.

[123] A. Modave, A. St-Cyr, W. A. Mulder, and T. Warburton. Nodal Discontinuous Galerkin Simulations for Reverse-Time Migration on GPU Clusters. *arXiv:1506.00907 [physics]*, 2015.

[124] A. Montlaur, S. Fernandez-Mendez, and A. Huerta. Discontinuous Galerkin methods for the stokes equations using divergence free approximations. *International Journal for Numerical Methods in Fluids*, 57(9):1071–1092, 2008.

[125] A. Montlaur, S. Fernandez-Mendez, J. Peraire, and A. Huerta. Discontinuous Galerkin methods for the Navier–Stokes equations using solenoidal approximations. *International Journal for Numerical Methods in Fluids*, 64(5):549–564, 2010.

[126] S. Nagrath, K. Jansen, and R. Lahey Jr. Computation of incompressible bubble dynamics with a stabilized finite element level set method. *Computer Methods in Applied Mechanics and Engineering*, 194(42-44):4565–4587, 2005.

[127] N. Nguyen, J. Peraire, and B. Cockburn. A hybridizable discontinuous Galerkin method for stokes flow. *Computer Methods in Applied Mechanics and Engineering*, 199(9-12):582–597, 2010.

[128] N. Nguyen, J. Peraire, and B. Cockburn. An implicit high-order hybridizable discontinuous Galerkin method for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 230(4):1147–1170, 2011.

[129] B. D. Nichols and C. W. Hirt. Calculating three-dimensional free surface flows in the vicinity of submerged and exposed structures. *Journal of Computational Physics*, 12(2):234–246, 1973.

[130] W. F. Noh and P. Woodward. SLIC (simple line interface calculation). In *Proceedings of the 5th International Conference on Fluid Dynamics*, volume 59, pages 330–340, 1976.

[131] E. Olsson and G. Kreiss. A conservative level set method for two phase flow. *Journal of Computational Physics*, 210(1):225–246, 2005.

[132] E. Olsson, G. Kreiss, and S. Zahedi. A conservative level set method for two phase flow II. *Journal of Computational Physics*, 225(1):785–807, 2007.

[133] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[134] S. Osher and C. Shu. High-order essentially nonoscillatory schemes for hamilton-jacobi equations. *SIAM Journal on Numerical Analysis*, 28(4):907–922, 1991.

[135] S. J. Osher and R. P. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 1 edition, 2002.

[136] M. Owkes and O. Desjardins. A discontinuous Galerkin conservative level set scheme for interface capturing in multiphase flows. *Journal of Computational Physics*, 249:275–302, 2013.

[137] R. Pasquetti and F. Rapetti. Spectral element methods on unstructured meshes: Comparisons and recent advances. *Journal of Scientific Computing*, 27:377–387, 2006.

[138] D. Peng, B. Merriman, S. Osher, H. Zhao, and M. Kang. A PDE-Based fast local level set method. *Journal of Computational Physics*, 155(2):410–438, 1999.

[139] J. Peraire and P. Persson. The compact discontinuous Galerkin (CDG) method for elliptic problems. *SIAM J. Numer. Anal.*, 30(4):1806–1824, 2007.

[140] P.-o. Persson and J. Peraire. Sub-cell shock capturing for discontinuous Galerkin methods. *Proc. of the 44th AIAA Aerospace Sciences Meeting and Exhibit*, AIAA-2006-113, 2006.

[141] F. Pochet, K. Hillewaert, P. Geuzaine, J.-F. Remacle, and T. Marchandise. A 3D strongly coupled implicit discontinuous Galerkin level set-based method for modeling two-phase flows. *Computers and Fluids*, 87:144–155, 2013.

[142] S. Popinet and S. Zaleski. A front-tracking algorithm for accurate representation of surface tension. *International Journal for Numerical Methods in Fluids*, 30(6):775–793, 1999.

[143] J. Proriol. Sur une famille de polynomes ‘a deux variables orthogonaux dans un triangle. *C. R. Acad. Sci.*, 257(1):2459–2461, 1957.

[144] A. Prosperetti and G. Tryggvason, editors. *Computational Methods for Multiphase Flow*. Cambridge University Press, Cambridge ; New York, 1 edition edition, 2009.

[145] E. Puckett, A. Almgren, J. Bell, D. Marcus, and W. Rider. A high-order projection method for tracking fluid interfaces in variable density incompressible flows. *Journal of Computational Physics*, 130(2):269–282, 1997.

[146] J. Qiu and C. Shu. Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method: one-dimensional case. *Journal of Computational Physics*, 193(1):115–135, 2004.

[147] J. Qiu and C. Shu. A comparison of Troubled-Cell indicators for Runge–Kutta discontinuous Galerkin methods using weighted essentially nonoscillatory limiters. *SIAM Journal on Scientific Computing*, 27(3):995, 2005.

[148] J. Qiu and C. Shu. Hermite WENO schemes and their application as limiters for Runge-Kutta discontinuous Galerkin method II: two dimensional case. *Computers & Fluids*, 34(6):642–663, 2005.

[149] J. Qiu. and C. Shu. Runge–Kutta discontinuous Galerkin method using WENO limiters. *SIAM Journal on Scientific Computing*, 26(3):907, 2005.

[150] F. H. G. R. H. Dennard. Design of Ion-Implanted MOSFET'S with very small physical dimensions. *Solid-State Circuits, IEEE Journal of*, 9(5):256 – 268, 1974.

[151] W. H. Reed and T. R. Hill. *Triangular mesh methods for the neutron transport equation*. Los Alamos Scientific Laboratory Report LA-UR-73-479, 1973.

[152] Y. Renardy and M. Renardy. PROST: A parabolic reconstruction of surface tension for the volume-of-fluid method. *Journal of Computational Physics*, 183(2):400–421, 2002.

[153] B. Rivière. *Discontinuous Galerkin methods for solving elliptic and parabolic equations: theory and implementation*. SIAM, 2008.

[154] R. M. Russell. The CRAY-1 Computer System. *Commun. ACM*, 21(1):63–72, 1978.

[155] J. Ryan, C. Shu, and H. Atkins. Extension of a post processing technique for the discontinuous Galerkin method for hyperbolic equations with application to an aeroacoustic problem. *SIAM Journal on Scientific Computing*, 26(3):821, 2005.

[156] A. Sandu and E. Constantinescu. Multirate explicit adams methods for time integration of conservation laws. *J Sci Comput*, 38(2):229–249, 2009.

[157] B. Seny, J. Lambrechts, R. Comblen, V. Legat, and J.-F. Remacle. Multirate time stepping for accelerating explicit discontinuous galerkin computations with application to geophysical flows. *Int. J. Numer. Meth. Fluids*, 71(1):41–64, 2013.

[158] J. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591–1595, 1996.

[159] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 2 edition, 1999.

[160] K. Shahbazi. An explicit expression for the penalty parameter of the interior penalty method. *Journal of Computational Physics*, 205(2):401–407, 2005.

[161] K. Shahbazi, P. Fischer, and C. Ethier. A high-order discontinuous Galerkin method for the unsteady incompressible Navier-Stokes equations. *Journal of Computational Physics*, 222(1):391–407, 2007.

[162] S. Sherwin. Hierarchical hp finite elements in hybrid domains. *Finite Elements in Analysis and Design*, 27(1):109–119, 1997.

[163] C. Shu. TVB uniformly High-Order schemes for conservation laws. *Mathematics of Computation*, 49(179):105–121, 1987.

[164] C. Shu. Total-Variation-Diminishing time discretizations. *SIAM Journal on Scientific and Statistical Computing*, 9(6):1073, 1988.

[165] A. Smolianski. Finite-element/level-set/operator-splitting (FELSOS) approach for computing two-fluid unsteady flows with free moving interfaces. *International Journal for Numerical Methods in Fluids*, 48(3):231–269, 2005.

[166] V. Sochnikov and S. Efrima. Level set calculations of the evolution of boundaries on a dynamically adaptive grid. *International Journal for Numerical Methods in Engineering*, 56(13):1913–1929, 2003.

[167] S. Spencer J. Hierarchical hp finite elements in hybrid domains. *Finite Elements in Analysis and Design*, 27(1):109–119, 1997.

[168] SPHERIC: 3D schematic dam break and evolution of the free surface. URL: `https://wiki.manchester.ac.uk/spheric/index.php/Test2`, Last visited: 05/07/2015.

[169] D. T. Steinmoeller, M. Stastna, and K. G. Lamb. A short note on the discontinuous Galerkin

discretization of the pressure projection operator in incompressible flow. *Journal of Computational Physics*, 251:480–486, 2013.

[170] T. Stieltjes. Sur quelques théorèmes d'algèbre. *Comptes Rendus De L'Academie Des Sciences*, 100:439–440, 1885.

[171] A. Stock. *Development and application of a multirate multistep AB method to a discontinuous Galerkin method based particle in cell scheme.* diploma thesis, Institut für Aerodynamik und Gasdynamik Universität Stuttgart and Brown University, 2009.

[172] J. Strain. Tree methods for moving interfaces. *Journal of Computational Physics*, 151(2):616–648, 1999.

[173] J. Strain. A fast modular semi-lagrangian method for moving interfaces. *Journal of Computational Physics*, 161(2):512–536, 2000.

[174] M. Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *Journal of Computational Physics*, 187(1):110–136, 2003.

[175] M. Sussman. A parallelized, adaptive algorithm for multiphase flows in general geometries. *Computers & Structures*, 83(6–7):435–444, 2005.

[176] M. Sussman, A. S. Almgren, J. B. Bell, P. Colella, L. H. Howell, and M. L. Welcome. An adaptive level set approach for incompressible two-phase flows. *Journal of Computational Physics*, 148(1):81–124, 1999.

[177] M. Sussman and E. Fatemi. Efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. *SIAM Journal on Scientific Computing*, 20(4):1165–1191, 1999.

[178] M. Sussman, E. Fatemi, P. Smereka, and S. Osher. An improved level set method for incompressible two-phase flows. *Computers and Fluids*, 27(5-6):663–680, 1998.

[179] M. Sussman and M. Hussaini. A discontinuous spectral element method for the level set equation. *Journal of Scientific Computing*, 19(1-3):479–500, 2003.

[180] M. Sussman and E. Puckett. A coupled level set and Volume-of-Fluid method for computing 3D and axisymmetric incompressible Two-Phase flows. *Journal of Computational Physics*, 162(2):301–337, 2000.

[181] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational Physics*, 114(1):146–159, 1994.

[182] O. Tatebe. The Multigrid Preconditioned Conjugate Gradient Method. 1995. 00000.

[183] M. A. Taylor, B. A. Wingate, and R. E. Vincent. An algorithm for computing fekete points in the triangle. *SIAM Journal on Numerical Analysis*, 38:1707, 2000.

[184] TOP500 Supercomputer Site. URL: `http://www.top500.org/`, Last visited: 06/08/2015.

[185] A. Toselli. hp discontinuous Galerkin approximations for the stokes problem. *Mathematical Models and Methods in Applied Sciences*, 12(11):1565–1597, 2002.

[186] G. Tryggvason. Numerical simulations of the rayleigh-taylor instability. *Journal of Computational Physics*, 75(2):253–282, 1988.

[187] Y.-H. Tsai, L.-T. Cheng, S. Osher, and H.-K. Zhao. Fast sweeping algorithms for a class of hamilton-jacobi equations. *SIAM Journal on Numerical Analysis*, 41(2):673–694, 2003.

[188] O. Ubbink and R. Issa. A method for capturing sharp fluid interfaces on arbitrary meshes. *Journal of Computational Physics*, 153(1):26–50, 1999.

[189] O. Ubbink and R. I. Issa. A Method for Capturing Sharp Fluid Interfaces on Arbitrary Meshes. *Journal of Computational Physics*, 153(1):26–50, 1999.

[190] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of Computational Physics*, 100(1):25–37, 1992.

[191] H. Wang and Y. Xiang. An adaptive level set method based on two-level uniform meshes and its application to dislocation dynamics. *International Journal for Numerical Methods in Engineering*, pages 573–597, 2013.

[192] Y. Wang, S. Simakhina, and M. Sussman. A hybrid level set-volume constraint method for incompressible two-phase flow. *Journal of Computational Physics*, 231(19):6438–6471, 2012.

[193] Z. Wang, J. Yang, B. Koo, and F. Stern. A coupled level set and volume-of-fluid method for sharp interface simulation of plunging breaking waves. *International Journal of Multiphase Flow*, 35(3):227–246, 2009.

[194] T. Warburton. *Spectral/hp Methods on Polymorphic Multi-Domains: Algorithms and Applications*. PhD thesis, Brown University, Providence RI, 1999.

[195] T. Warburton. An explicit construction of interpolation nodes on the simplex. *Journal of Engineering Mathematics*, 56:247–262, 2006.

[196] T. Warburton, L. Pavarino, and J. Hesthaven. A pseudo-spectral scheme for the incompressible Navier–Stokes equations using unstructured nodal elements. *Journal of Computational Physics*, 164(1):1–21, 2000.

[197] G. X. Wu, R. E. Taylor, and D. M. Greaves. The effect of viscosity on the transient free-surface waves in a two-dimensional tank. *Journal of Engineering Mathematics*, 40(1):77–90, 2001.

[198] Z. Xie, D. Pavlidis, J. R. Percival, J. L. M. A. Gomes, C. C. Pain, and O. K. Matar. Adaptive unstructured mesh modelling of multiphase flows. *International Journal of Multiphase Flow*, 67, Supplement:104–110, 2014.

[199] J. Yan and S. Osher. A local discontinuous Galerkin method for directly solving Hamilton-Jacobi equations. *Journal of Computational Physics*, 230(1):232–244, 2011.

[200] K. Yokoi. A practical numerical framework for free surface flows based on CLSVOF method, multi-moment methods and density-scaled CSF model: Numerical simulations of droplet splashing. *Journal of Computational Physics*, 232(1):252–271, 2013.

[201] D. Youngs. Time-dependent multi-material flow with large fluid distortion. *Numerical Methods for Fluids Dynamics*, pages 273–285, 1982.

[202] S. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of Computational Physics*, 31(3):335–362, 1979.

[203] M. Zhang and C. Shu. An analysis of three different formulations of the discontinuous Galerkin method for diffusion equations. *Mathematical Models and Methods in Applied Sciences*, 13(3):395–413, 2003.

[204] Y.-T. Zhang and C.-W. Shu. High-order WENO schemes for hamilton-jacobi equations on triangular meshes. *SIAM Journal on Scientific Computing*, 24(3):1005–1030, 2003.

[205] L. Zhao, J. Mao, X. Bai, X. Liu, T. Li, and J. J. R. Williams. Finite element implementation of an improved conservative level set method for two-phase flow. *Computers & Fluids*, 100:138–154, 2014.

[206] J. Zhu and J. Qiu. Local DG method using WENO type limiters for convection-diffusion problems. *Journal of Computational Physics*, 230(11):4353–4375, 2011.

[207] G. W. Zumbusch. Symmetric hierarchical polynomials for the hp-version of finite elements. *Houston Journal of Mathematics*, page 529, 1995.

CURRICULUM VITAE

CONTACT INFORMATION

Name: Ali Karakus

Address: Mechanical Engineering Department/METU

Phone: +90-506-456-9344/+90-312-210-5243

Fax.: +90-312-210-2536

E-mail: akarakus@metu.edu.tr

EDUCATION

Degree	Institution	Year of Graduation
Ph.D.	METU - Mechanical Engineering	2015
M.S	MU - Mechanical Engineering	2009
B.S.	METU - Mechanical Engineering	2005

RESEARCH INTERESTS

- High Order Numerical Methods
- Discontinuous/Continuous Spectral Element Methods
- Multiphase, Incompressible and non-Newtonian Flows
- High Performance Computing, CPU-GPU Parallelization
- Efficient Linear Solvers, Local Time Stepping Techniques

PUBLICATIONS

Refereed Journals (Submitted or in Preperation)

- **Karakus, A.**, Warburton T., Aksel H., and Sert, C. An Adaptive Fully

Discontinuous Galerkin Level Set Method for Incompressible Free Surface Flows. Appear in *International Journal of Numerical Methods in Fluids*.

- **Karakus, A.**, Warburton T., Aksel H., and Sert, C., A GPU Accelerated Discontinuous Galerkin Scheme for Interface Capturing in Incompressible Multiphase flows. Appear in *Journal of Computational Physics*.
- **Karakus, A.**, Warburton T., Aksel H., and Sert, C., 2015. Level Set Reinitialization for a h-Adaptive Discontinuous Galerkin Method. Appear in *Computer and Mathemetaics with Applications*.
- **Karakus, A.**, Warburton T., Aksel H., and Sert, C.,2015. An adaptive Discontinuous Galerkin Level Set Method with Local Time Stepping. Submitted to *Journal of Computational Fluid Dynamics*.

Refereed Journals

- Tutar, M., **Karakus, A.**, 2014. Numerical Study of Polymer Melt Flow in a Three Dimensional Sudden Expansion: Effect of Viscous Dissipation. *Journal of Polymer Engineering*, 135(1), pp:1-16.
- Tutar, M., **Karakus, A.**, 2013. Computational Modeling of the Effects of Viscous Dissipation on Polymer Melt Flow Behavior During Injection Molding Process in Plane Channels. *Journal of Manufacturing Science and Engineering* (ASME) 135(1), pp:1-16.
- Tutar, M., **Karakus, A.**, 2013. A Numerical Study of Solidification and Viscous Dissipation Effects on Polymer Melt Flow in Plane Channels. *Journal of Polymer Engineering*, 33(1), pp:355-384.
- Tutar, M., **Karakus, A.**, 2010. Computational Study of the Effect of Governing Parameters on a Polymer Injection Molding Process for Single-Cavity and Multicavity Mold Systems. *Journal of Manufacturing Science and Engineering* (ASME), 132(1), pp:1-12.
- Tutar, M., **Karakus, A.**, 2009. A Numerical Simulation Study of Compressible Filling Process of Mold Insert Polymer Injection Molding. *Journal of Polymer Engineering*, 29(6), pp:355-384.
- Tutar, M., **Karakus, A.**, 2009. 3-D Computational Modeling of Process Condition Effects on Polymer Injection Molding. *International Polymer*

Processing, 2009(5), pp: 384-398.

Conferences

- **Karakus, A.**, Warburton T., Aksel H., and Sert, C., An Adaptive Fully Discontinuous Galerking Level Set Method for Incompressible Free Surface Flows. Appear in *European Conference on Numerical Mathematics and Advanced Applications, ENUMATH*, September 14-18 2015, Ankara, Turkey
- Ongut, A. E., Aksel, M. H., Turan, R., **Karakus, A.**., Erkursun, B., Cicek, E., Demircioglu, O. and Kilkis, B.,A , Numerical Study For The Improvement of Automobile Cabin Confort Conditions Using External Cooling Units Operated by Phovoltaic Solar Panels. *6th Automotive Technologies Congress, OTEKON*, June 4-5 2012, Bursa, Turkey
- Tutar, M., **Karakus, A.**, 3D Numerical Simulation of Polymer Injection Molding. *in Proc. of 6th ICCHMT-International Conference on Computational Heat and Mass Transfer*, May 18-21 2009, Guangzhou, China
- Tutar, M., **Karakus, A.**, 3D Computational Modeling of Effects of Geometric Conditions on Injection Molding. *in Proc. Of Twelfth International Materials Symposium*, October 15-18, 2008, Denizli, Turkey
- Tutar, M., **Karakus, A.**, Computational Modeling of Three-dimensional Compressible Filling Process. *in Proc. of ICHMT International Symposium on Advances in Computational Heat Transfer*, May 11-16, 2008, Marrakech, Morocco

PROJECTS

- Fast, high-order numerical methods on many-core and GPU-based architectures for interface capturing in incompressible multiphase flows, **Primary Researcher**, Computational and Applied Mathematics, Rice University, Houston, TX , July, 2013-July, 2014.
- Design of an Unmanned Underwater Vehicle Operating Under High Speed, ASELSAN, **Research Engineer**, September, 2009 - July, 2013.

- Automotive Solar Cooling: Research and Development and Prototype Manufacturing for Electronic-Thermal-Mechanical-Photovoltaic Systems, **Research Engineer**, Turkish Automobile Factories (TOFAS), Project No: METU-BİLTİR- T-2008-0804-C-20, September, 2012 - May, 2013.
- Development of a Two-dimensional Laminar Navier-Stokes Solver for Cartesian Grids, **Research Engineer**, Engineering Research Committee, The Scientific and Technical Research Council of Turkey, Project No: 109M510, Aug, 2011 - June, 2012.
- A Design Project of an Ideal Metal-Insert Leakproof Fitting Operating Under Very High Pressures, **Research Engineer**, Engineering Research Committee, The Scientific and Technical Research Council of Turkey, Project No: 106M465, September, 2007 - June, 2009.
- Producing Electrical Power from the Sea Environment, **Research Engineer**, Engineering Research Committee, Turkish Ministry of Industry and Trade, Project No: 00280.STZ.2008-1, June, 2008 - July, 2009.