Stony Brook University

Scalable Particle and Mesh Algorithms for Elliptic Components of Multiphase Problems

A Dissertation Presented

by

Tongfei Guo

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

July 2013

Stony Brook University

The Graduate School

Tongfei Guo

We, the dissertation committee for the above candidate for the
Doctor of Philosophy degree, hereby recommend
acceptance of this dissertation.

Roman V. Samulyak - Dissertation Advisor
Associate Professor, Department of Applied Mathematics and Statistics

James Glimm - Chairperson of Defense
Distinguished Professor, Department of Applied Mathematics and Statistics

Xiangmin Jiao - Member
Associate Professor, Department of Applied Mathematics and Statistics

Alan C. Calder - Outside Member
Assistant Professor, Department of Physics and Astronamy

This dissertation is accepted by the Graduate School.

Charles Taber
Interim Dean of the Graduate School

Abstract of the Dissertation

Scalable Particle and Mesh Algorithms for Elliptic Components of Multiphase Problems

by

Tongfei Guo

Doctor of Philosophy

in

Applied Mathematics and Statistics

Stony Brook University

2013

New mesh and meshless algorithms for elliptic boundary and elliptic interface problems have been developed. By utilizing the embedded boundary method, a mesh based algorithm to solve elliptic interface problem is implemented as an extension of hybrid Eulerian-Lagrangian hydrodynamic library FronTier which employs the method of front tracking for interface propagation and this implementation is parallelized for distributed memory clusters. The use of embedded boundary method supports arbitrary discontinuities of density and other physics properties across the interfaces and significantly improves methods that smear interface discontinuities across several grid cells.

This code has been applied to process simulation for heat transfer problem, stefan problem and magnetohydrodynamics at low magnetic Reynolds number. To handle problems brought by the complexity of interfaces, algorithms for solving elliptic boundary and elliptic interface problems have been proposed based on meshless particle-based method. The typical feature of the elliptic interface problem is the presence of a geometrically complex internal boundary across which material properties or solutions rapidly change. The main motivation for the development of particle-based methods for elliptic problems is to carry out numerical simulation of free surface of multiphase systems described by coupled hyperbolic and elliptic equations. A Lagrangian particle technique, smoothed particle hydrodynamics(SPH) has been implemented and tested. To overcome the drawbacks of poor numerical accuracy of SPH, another Lagrangian particle technique with local polynomial fitting has been developed and implemented. All the implementation is fully parallelized. The current work deals with methods for elliptic components of coupled systems. And, the developed elliptic methods, if used independently, also favorably compare to unstructured finite element methods that require mesh generation and depend on the mesh quality. Currently, a second order accurate algorithm has been used and higher order discretization is also possible. *Key Words:* front tracking, embedded boundary method, MHD, smoothed particle hydrodynamics, Lagrangian particle method, local polynomial fitting

To my father

Shaopeng Guo

and mother
Xiaolian Lu

Table of Contents

List of Figures

List of Tables

Acknowledgements

This disseration is one of the most important works that has been done and I would like to say thank you to people who help me to make this through and I will cherish this experience forever. I would like to express my sincere gratutide to my advisor Professor Roman Samulyak for his support of my Ph.D study. It was more than five years ago when I first talked with him and I made my decision to work with him ever since. It is with his insightful and prospective ideas that I can go across all the obstacles in the way to this disseration. I would like to thank Professor Xiangmin Jiao. His professional and pratical suggestions help me to overcome lots of difficulties in reseach. I would also like to thank Professor James Glimm and Alan Calder for being on my dissertation committee. I would like to thank all my friends during my years of study as a graduate student at Stony Brook for their friendship and encouragement. It is you who made my graduate student life full of happiness. Last but not the least, I would like to thank my family. The unconditional love of my parents always gave me the strength to move forward.

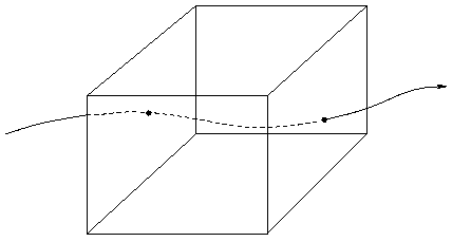
Chapter 1

Introduction

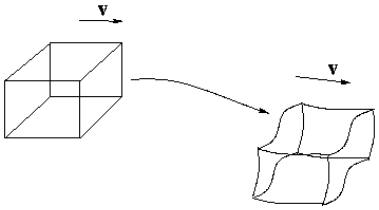
1.1 Overview and Motivation

The coupled elliptic-hyperbolic system is a fundamental partial differential equation system that governs a large number phenomena, such as the heat transfer problem, the phase transition problem as well as magnetohydrodynamic(MHD) problems. The study of numerical algorithms to solving such system provides stronger and stronger mathematical tools to understand the physics nature of problems mentioned above.

Significant part of this dissertation will be focused on hydrodynamics related problems. There are two approaches to deal with fluid dynamics, Eulerian method and Lagrangian method (Figure 1.1). In the Eulerian method, both coordinate frame(or computational mesh) is fixed to laboratory frame and the flows moves with respect to the coordinate frame. With Eulerian method, the evaluation of fluid states are relatively simpler and easier to be extended to higher dimension. A major drawback of Eulerian method is that the difficulty to capture dynamic interfaces. In the Lagrangian method, the



(a)



(b)

Figure 1.1: Mesh of Eulerian and Lagrangian method

moving substance is represented by material parcels that move together with the flow. It is a natural resolution of the interface but the material parcels are severely distorted by the flow. In high dimension, the untangling of material parcels becomes unrealistic. Lagrangian method is ideal for solid dynamics cause the deformation of material parcel is relatively small.

Mesh and meshless algorithms have been developed for elliptic boundary and elliptic interface problems. A hydrodynamics library named FronTier[9] is utilized to be the base for extension of elliptic solver. FronTier library employs hybrid Eulerian-Lagrangian approach to perform hydrodynamics simulation. The free surface or interior interface between two components is

represented by Lagrangian mesh which moves independently to the Cartesian mesh background. The propagation of the Lagrangian mesh is performed by Front Tracking method. On the other hand, the interior states of the fluid is solved on the Cartesian mesh. Both compressible and incompressible solvers are implemented in the FronTier library. The FronTier code always keeps discontinuities sharp and eliminates or strongly restricts numerical diffusion across material interfaces. It supports large number of geometrically complex interfaces in two- and three-dimensional spaces and robustly resolves their topological changes. FronTier has been widely used for variety of fundamental science (turbulent fluid mixing [30]) and applied problems (liquid targets for particle accelerators [45], pellet ablation in tokamaks [48], and plasma jet liners for magneto-inertial fusion [26]). The compressible FronTier-MHD code [47] is an extension of FronTier and it is the first implementation of embedded boundary method for FronTier. The code is well suited for free surface MHD phenomena driven by hydro waves, for instance in the case of matter interacting with strong energy sources. The obvious limitation of this code for the simulation of slow flows of liquid metals is the restriction of time steps by the CFL condition due to acoustic waves. Shuqiang Wang extended the embedded boundary method for elliptic interface problem[54] and applied such method for incompressible fluid solver[53], which eliminates all acoustic waves and allows big time steps.

In order to overcome this limitation but keep all advantages of front tracking, a sharp interface MHD algorithm for incompressible multiphase MHD flows in the low magnetic Reynolds number approximation has been devel-

oped. As the method is dependent on the quality of the Navier-Stokes equation solver, we would like to comment first on the hydrodynamic component of the code. Front tracking has already been used for the simulation of incompressible Navier-Stokes equations [52, 15]. But unlike the front tracking method for compressible flows [9] which always keeps the density discontinuity sharp, previous implementations of the front tracking for incompressible flows employed the smoothing of density similar to the level set method. Other methods such as the ghost fluid method [32, 25] and the immersed interface method [28, 27] also have difficulties with large density ratios across the interface. A front tracking algorithm for incompressible Navier-Stokes approximations that successfully deals with the large density discontinuity problem has been recently proposed by collaborators [53] by using the embedded boundary method [23] .

The pure Lagrangian method attracts a lot interests again, because many free surface problems would benefit from Lagrangian methods. The Smoothed Particle Hydrodynamics is a widely used Lagrangian particle technique[35, 33]. In the Smoothed Particle Hydrodynamics method, material parcel is replaced by a geometric point, called particle, which carries all physics properties of the corresponding material parcel. A Smoothed Particle Hydrodynamics code has been implemented and couple of testing cases have been performed. Because of poor derivatives calculation, the governing equations of the motion of fluid is not solved correctly. however the final physics picture is similar to the reality. It is argued[41] that this self-contradiction outcome is due to the

conservative property of the Smooth Particle Hydrodynamics technique.

In order to overcome the weakness of Smoothed Particle Hydrodynamics technique, a different approach, local polynomial fitting, is employed to approximate the derivatives. The local polynomial fitting has long been used to construct estimators for function value and derivatives. By choosing local particles with certain criteria and preprocessing the particle positions information, the estimation of derivatives can reach high accuracy. However, because of the unstructured distribution property of particles, the computational cost of acquiring estimation for derivatives with particles is much higher than acquiring such estimation with structured Cartesian grid. A method proposed by Robert D. Richtmyer and K. W. Morton[44] is employed to estimate the change of specific volume of flow.

1.2 Dissertation Organization

In Chapter 2, the technical detail of Embedded Boundary Method and the physics background of the problem will be present. And, the discretization governing equations of the MHD system as well as the accuracy of the simulation will be discussed. Chapter 3 presents the mathematical background of Smoothed Particle Hydrodynamics technique. Also, the implementation of the code and several specific technical problems are involved. Chapter 4 is about the approach to obtain approximation of derivatives with local polynomial fitting and the verification as well validation of such technique is present. The discretization of Lagrangian equations is also discussed.

Chapter 2

Finite Volume Elliptic Algorithms for Frontier

The FronTier library employs hybrid Eulerian-Lagranian method to solve CFD problems. By utilizing Front tracking[16], a Lagrangian mesh algorithm, to propagate the interface and grid based method to solve interior states of fluid, FronTier library is capable to simulate multi-components problem(Figure 2.1) with interfaces accurately. In this chapter, a front tracking MHD algorithm for free surface / multiphase flows at the low magnetic Reynolds numbers, coupled with the incompressible hydrodynamic solver, as well as validation and verification tests will be presented[19]. With the advantage of incompressible hydrodynamic solver, the code can deal with the simulation of large time scales, in particular, with flows of free surface liquid metals in magnetic fields. Meanwhile, the application on heat transfer problems will also be discussed.

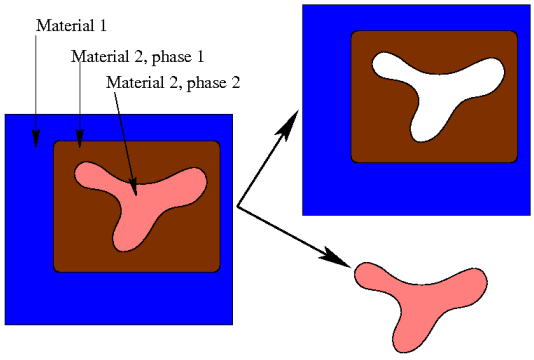


Figure 2.1: Multiphase Domain

2.1 Embedded Boundary Algorithm for Elliptic Problems in Irregular Domains

The governing equation of elliptic boundary problem is

$$\nabla\beta\nabla\Phi = f \tag{2.1}$$

where f is the source term. With the divergence theorem, the governing equation can be written in integral form 2.2 in a given control volume Ω.

$$\int_{\Omega} \nabla\cdot(\beta\nabla\Phi)\,dv = \oint_{\partial\Omega} \beta\nabla\Phi\cdot\mathbf{n}\,ds = \int_{\Omega} f\,dV \tag{2.2}$$

Further more, if the source term can be expressed by the divergence of a vector field,

$$f = \nabla\cdot(\mathbf{V}) \tag{2.3}$$

the equation can be written as 2.4

$$\oint_{\partial\Omega} \beta \nabla\Phi \cdot \mathbf{n}\, ds = \int_{\Omega} f\, dV = \oint_{\partial\Omega} \mathbf{V} \cdot \mathbf{n}\, ds \tag{2.4}$$

There are many methods for the solving of elliptic problem in the support domain (where the governing equation is defined) of regular boundaries. However, for geometric complex support domain, many methods will fail or be unable to achieve satisfying accuracy, because of the inaccurate discretization of boundary conditions. Here, the embedded boundary method is adopted to overcome this problem. The embedded boundary method([54, 23, 34, 49]) is a finite volume method for an irregular domain embedded on a Cartesian grid. By utilizing the embedded boundary method, the governing equation as well as boundary condition are discretized in each control cell. There are two types of cells, one is interior cell, and the other one is boundary cell. The boundary cell is split by the boundary to be two parts while only one of the parts belongs to the computational domain and it is called partial cell. On the contrary, all interior cells are full cells. When embedded boundary method is used to solve the elliptic boundary value problem, unknowns are defined at the computational cell centers for both interior cells (Full cells), and boundary cells (partial cells) which intersect with the interior boundary.

For the purpose of visually simplicity, a 2D example is presented to describe embedded boundary method. For interior cells, the discretization is similar to 5 point stencil second order finite difference, as shown in 2.2. For cell (i,j),

$$F_{i+\frac{1}{2},j} + F_{i-\frac{1}{2},j} + F_{i,j+\frac{1}{2}} + F_{i,j-\frac{1}{2}} = S * f_{i,j} \tag{2.5}$$

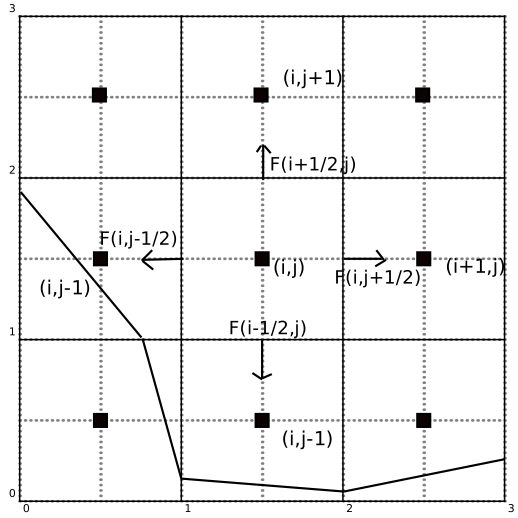


Figure 2.2: Stencil for the the interior flux

where S is the area of the corresponding cell.

Flux $F_{i+\frac{1}{2},j}$ can be expressed by

$$F_{i+\frac{1}{2},j} = \Delta y \beta_{i+\frac{1}{2},j} \frac{\Phi_{i+1,j} - \Phi_{i,j}}{\Delta x} \tag{2.6}$$

and $F_{i-\frac{1}{2},j}, F_{i,j+\frac{1}{2}}, F_{i,j-\frac{1}{2}}$can also be expressed in the similar way.

For boundary cells, fluxes can be divided into three types:full edge flux, partial edge flux(flux F(i+1,j) and flux F(i,j+1) in Figure 2.3) and boundary flux(flux F_f in Figure 2.3). The approximation of full edge flux is the same as interior cells. The approximation of partial edge flux is obtained by interpolation of two nearest full edge fluxes that

$$F_{i+\frac{1}{2},j} = a\Delta y \beta_m [\frac{(1+a)}{2} \frac{\Phi_{i+1,j} - \Phi_{i,j}}{\Delta x} + \frac{(1-a)}{2} \frac{\Phi_{i+1,j+1} - \Phi_{i,j+1}}{\Delta x}] \tag{2.7}$$

and

$$F_{i,j+\frac{1}{2}} = a\Delta x\beta_m[\frac{(1+a)}{2}\frac{\Phi_{i,j+1}-\Phi_{i,j}}{\Delta y} + \frac{(1-a)}{2}\frac{\Phi_{i+1,j+1}-\Phi_{i+1,j}}{\Delta y}] \qquad (2.8)$$

Here, a is the proportion of partial edge to the full edge. With equation(2.7), second order approximation of the partial edge flux can be obtained. For boundary flux, the approximation for Neumann boundary condition can be obtained directly from the boundary condition. While with Dirichlet boundary condition, approximation is obtained by interpolation in the normal direction[Fig 2.3]. To do this, the first pair of parallel cell center lines that intersect with the line in the normal direction of the interface while not passing through the current cell are selected. First, the interpolation of two values in the intersections of the line in the normal direction and two cell center lines are obtained. Then, boundary flux are acquired with interpolation of these two interpolated values together with the boundary value given by the boundary condition.

To obtain the approximation of gradient with second order accuracy, quadratic polynomial interpolation along grid lines is used and the gradient formula is applied as in one dimension:

$$q^f = \frac{1}{d_2-d_1}(\frac{d_2}{d_1}(\Phi^f - \Phi_1^I) - \frac{d_1}{d_2}(\Phi^f - \Phi_2^I)) \qquad (2.9)$$

Here,Φ^f is the value of function in the boundary which is given by the Dirichlet boundary condition at the boundary segment midpoint. With interpolation along grid lines, we can obtain Φ_1^I and Φ_2^I at the points distance d_1 and d_2

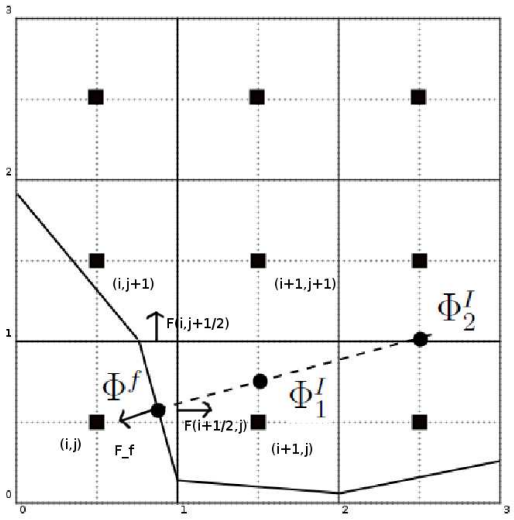


Figure 2.3: Stencil for the the boundary flux

away from the interface. Finally, the boundary flux is evaluated by

$$F^f = \beta^f A^f q^f \tag{2.10}$$

where β^f is the β value in the midpoint of boundary segment and A^f is the length of the boundary segment. The discretization of governing equation in each control cell forms a linear system. The resulting matrix will be a large sparse matrix and the number of unknowns is big.

2.2 Embedded Interface Algorithm for Multiphase problems

The embedded boundary method has been extended to solve elliptic and parabolic interface problems in this section. In these problems, there are more than one components. For each component i, the governing equation is

different and jump conditions are imposed on the interior interface(2.12).

$$\nabla \beta_i \nabla T = f_i \tag{2.11}$$

and the jump condition in the interface

$$[T] = J_1 \left[\beta \frac{\partial T}{\partial n} \right] = J_2 \tag{2.12}$$

The feature that defining unknowns in cell center for all cells of EBM is retained in the algorithm for the elliptic interface problem. Instead of define one unknown in each cell center for elliptic boundary value problem, more unknowns are needed in order to discretize the elliptic equation consistent with two interface jump conditions for internal partial cells. In 2D, (Figure 2.4) shows the placement of unknowns in cell with interior interface. The whole cell contains two partial cells representing two components(A and B) which are separated by the interior interface. Two unknowns are defined in the cell center for each part of the cell. In order to comply the jump conditions (equations 2.12 and 2.12), two more unknowns are defined in the geometry center of the segment of the interior interface which intersects with the cell for both partial cell A, B.

A schematic of corresponding stencil for the interpolation of boundary unknowns are shown in (Figure 2.5). Two unknowns T_a and T_b are defined respectively at the segment of interior boundary for both parts in cell (i,j). The direction of normal of the interior boundary is defined as pointing from A to B. The discretization of jump condition (2.12) is simply

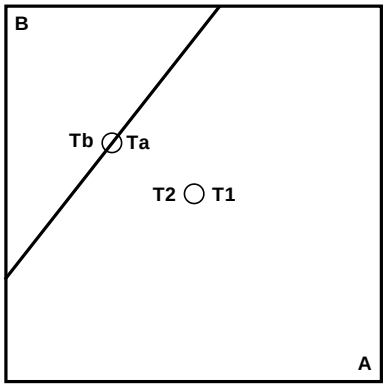


Figure 2.4: Placement of unknowns in a cell containing the interface

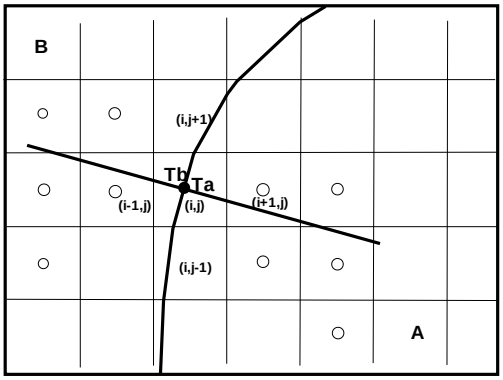


Figure 2.5: Stencil for the interface unknowns for the jump condition

$$T_A - T_B \;=\; J_1 \tag{2.13}$$

The discretization of jump condition (2.12) is more complicated that the normal derivatives of unknowns in both sides of the interior boundary have to be evaluated. This is carried out by fitting a quadratic polynomial with two variables[23, 34]. The main idea is use one unknown in cell (i,j), two unknowns in the first layer neighbours of cell (i,j) and three unknowns in the second layer neighbours of cell (i,j) that six unknowns in total are given for six coefficients of the quadratic polynomial. For example, in order to construct the quadratic polynomial to evaluate the flux at the interior boundary segment center where T_a is defined for component A, the six unknowns are T_a, $T_{i+1,j}$, $T_{i+1,j-1}$, $T_{i+2,j}$, $T_{i+2,j-1}$, and $T_{i+2,j-2}$. Similarly, we can construct the quadratic polynomial for component B. Taking the normal derivatives of the fitted polynomials for two components to get $\frac{\partial T}{\partial \mathbf{n}}\big|_A, \frac{\partial T}{\partial \mathbf{n}}\big|_B$, respectively, and using the jump condition (2.12), we obtain,

$$\beta|_B \left.\frac{\partial T}{\partial \mathbf{n}}\right|_B - \beta|_A \left.\frac{\partial T}{\partial \mathbf{n}}\right|_A = J_2 \tag{2.14}$$

The embedded boundary method is used to setup two equations for two unknowns in the cell center separately. For partial cell cdef, a similar stencil as the previous paragraph is used to discretize the elliptic operator(Figure 2.6). With equation (2.2), we obtain

$$\int_{cdef} \nabla \cdot (\beta \nabla T)\, ds = \oint_{\partial(cdef)} \beta \nabla T \cdot \mathbf{n}\, dl \;=\; \int_{cdef} f\, ds \tag{2.15}$$

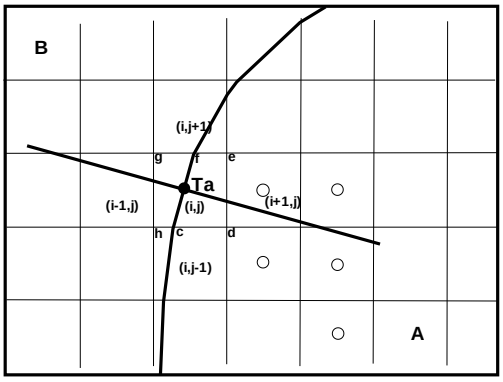


Figure 2.6: Stencil for the cell center unknowns

which is

$$\int_{cd} \beta \nabla T \cdot \mathbf{n}\, dl + \int_{de} \beta \nabla T \cdot \mathbf{n}\, dl + \int_{ef} \beta \nabla T \cdot \mathbf{n}\, dl + \int_{fc} \beta \nabla T \cdot \mathbf{n}\, dl \quad = \quad \int_{cdef} (2f16)\, ds$$

and the discretized form is

$$\begin{aligned} & l_{cd} \cdot Flux_{cd} + l_{de} \cdot Flux_{de} + l_{ef} \cdot Flux_{ef} \qquad (2.17) \\ & + l_{fc} \cdot Flux_{fc} = T(i,j) \int_{cdef} ds \end{aligned}$$

where l_{xy} is the length of the segment between x and y. Thus only fluxes across the center of segment of interior boundary and the center of cell edge shall be evaluated. For $Flux_{cd}$, a second order derivative is calculated by using linear interpolation of $\frac{T_{i,j-1}-T_{i,j}}{\Delta x}$ and $\frac{T_{i+1,j-1}-T_{i+1,j}}{\Delta x}$ in the center of segment cd and multiplied by β[54]. For $Flux_{de}$, we simply use central difference $\frac{T_{i+1,j}-T_{i,j}}{\Delta y}$ to calculate the derivative and multiply β. $Flux_{ef}$ is obtained in the similar way as $Flux_{cd}$ by evaluating linear interpolation of $\frac{T_{i,j+1}-T_{i,j}}{\Delta x}$ and $\frac{T_{i+1,j+1}-T_{i+1,j}}{\Delta x}$

in the center of ef and multiplied by β. $Flux_{fc}$ is evaluated in the previous paragraph which is $\beta|_A \frac{\partial T}{\partial \mathbf{n}}\big|_A$. In the same way, fluxed of other partial cells are obtained. In the implementation, it is not necessary to have two unknowns in the segment center of interior boundary that these unknowns can be represented by corresponding cell center unknowns by solving jump conditions (2.12),(2.12). And the resulting system of equations will only have unknowns defined at cell centers.

In 3D cases, the evaluation is similar. For linear interpolation of two edge center fluxes(e.g $Flux_{cd}$ in the previous paragraph), a bi-linear interpolation is used. Also, a 3-D quadratic polynomial has 10 coefficients thus 10 unknowns are required to construct such polynomial. Details of such interpolation and construction mentioned above can be found in [49].

2.3 Applications

The embedded boundary parabolic interface technique has been applied to several physics phenomenons

2.3.1 Stefan problem and phase transitions in incompressible media

Numerical algorithm for the Stefan problem

Let us consider the melting and solidification processes in multi-material systems. Let Ω be a bounded domain containing multiple materials in the solid and liquid states. We define the material interface as a contact surface

between different materials and the phase boundary a contact surface between different phases of the same material. We denote as $\partial\Omega$ the collection of interfaces, phase boundaries, and the external boundary of the domain Ω. The equation of the heat transport in $\Omega \setminus \partial\Omega$ is [2]

$$\frac{\partial \rho C_P T}{\partial t} = \nabla \cdot k \nabla T + Q, \tag{2.18}$$

where T is the temperature, ρ is the density, C_P is the heat capacity at constant pressure, k is the thermal conductivity, and Q is the external heat source. If equation 2.18 is applied to a single phase, ρ, C_P, and k are typically constants. But since the material properties change with the change of temperature, we do not restrict the algorithm by assuming constant values for these variables. The continuity of temperature and heat flux is satisfied at material interfaces:

$$[T]_{\text{material interface}} = 0, \tag{2.19}$$

$$\left[k\frac{\partial T}{\partial n}\right]_{\text{material interface}} = 0, \tag{2.20}$$

where [.] denotes the jump of quantity across the interface and n is the normal to the interface. At the phase transition boundary, the following conditions are satisfied:

$$[T]_{\text{phase boundary}} = 0, \quad T_{\text{phase boundary}} = T_M, \tag{2.21}$$

$$\left[k\frac{\partial T}{\partial n}\right]_{\text{phase boundary}} = -\rho L v, \tag{2.22}$$

where T_M is the melting temperature, L is the latent heat, and v is the velocity of the phase boundary. In the last equation, we assumed that the density does not change during the phase transition. It is the standard approximation in the theory of melting and solidification [2].

In addition, the Gibbs-Thomson condition must be satisfied at the phase boundary. The Gibbs-Thomson effect is the dependence of T_M on the interface curvature. Its thermodynamic foundation can be found in [2]. The equation of the Gibbs-Thomson effect is

$$T_M = T_{M,0} - \frac{K\gamma}{\Delta S_f}, \tag{2.23}$$

where $T_{M,0}$ is the melting temperature at the flat interface, K is the interface curvature, γ is the interfacial tension, and ΔS_f is the fusion entropy. Since the change of T_M is small at moderate surface curvatures, it is usually negligible during melting. The unsteady solidification which occurs in undercooled liquids (when the liquid temperature is lower then T_M) results in the growth of dendrites of very complex shape. The inclusion of the Gibbs-Thomson effect is necessary for the accurate resolution of the dynamics of unsteady solidification. Finally, various Dirichlet or Neumann or mixed type boundary conditions can be applied on the outside boundaries of the multi-phase domain.

We call the heat conduction problem in multi-material system *without* phase transitions, described by equations (2.18-2.20), the *parabolic interface problem*. Phase boundaries described by the jump condition (2.22) decouple the problem in a set of parabolic problems with geometrically complex and

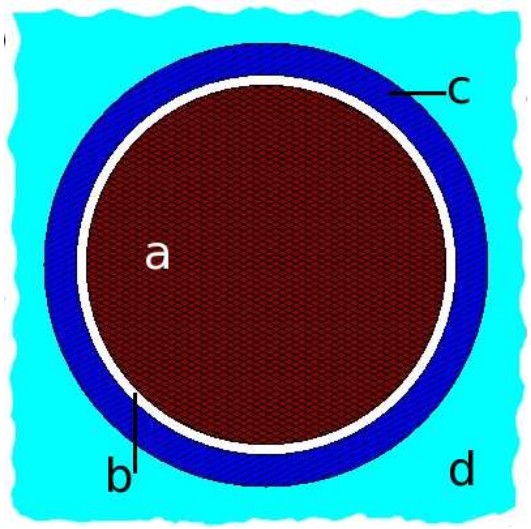


Figure 2.7: Schematic of the nuclear fuel rod cross-section: (a) is the fuel pellet, (b) is the gas gap, (c) is the stainless steel cladding, and (d) is the liquid sodium.

moving external boundaries (phase boundaries) with the Dirichlet boundary condition $T_{\text{boundary}} = T_M$

Simulations of failure of nuclear fuel rods during accidents

This application is part of DOE Nuclear Energy Research Initiative for coupled multiscale simulation of nuclear rod failure consortium of RPI, SBU, Columbia and BNL. NPHASE-CMFD code, using Reynolds-Averaged Navier Stokes (RANS, e.g. k-e model) approach to multiphase modeling, will simulate flow of liquid sodium coolant and fission gas around reactor fuel rods. PHASTA code, using direct numerical simulation (DNS) with Level Set method to track the interface between gas and liquid phases, will simulate high pressure fission gas entering coolant channels. Our FronTier code, which is a front tracking code capable of simulating multiphase compressible fluid dy-

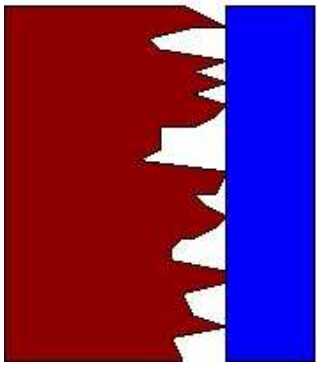


Figure 2.8: Shape of the gas gap between the fuel and cladding.

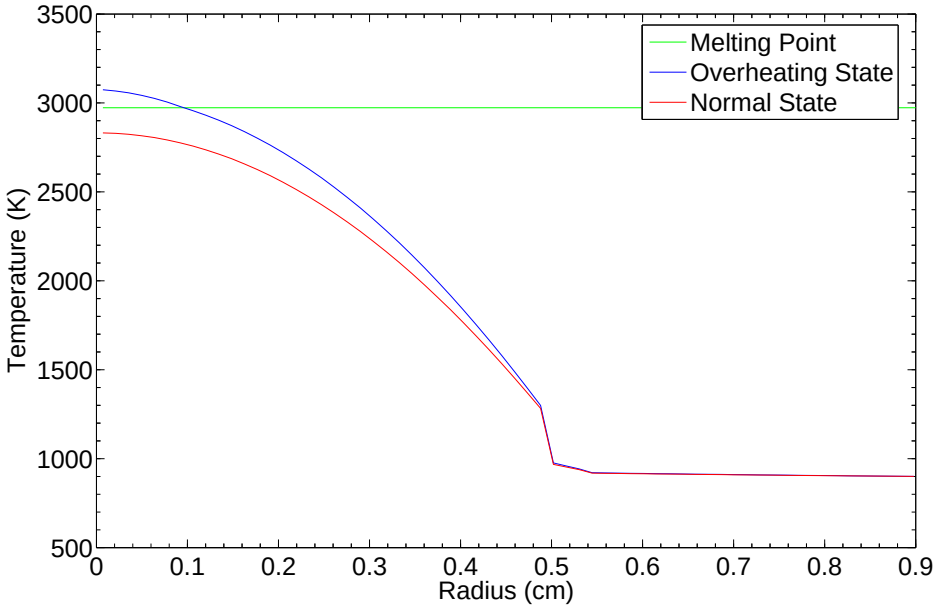


Figure 2.9: Temperature across the fuel rod at normal operating conditions and transient overheating.

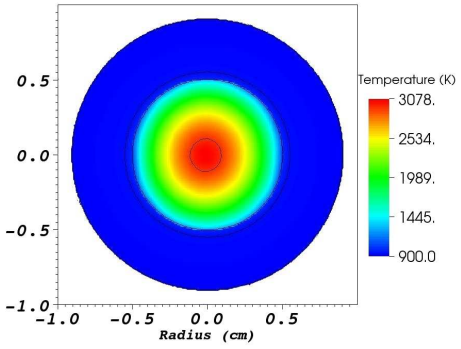


Figure 2.10: Temperature distribution across the fuel rod during transient overheating and the surface of the molten fuel.

namics, will simulate fuel rod overheating and melting of cladding in case of coolant-blockage accident. Code using molecular dynamics approach analysis the irradiated fuel properties will give prediction of fuel properties evolution.

A nuclear fuel rod contains 8 mm diameter pellets of metallic or oxide uranium fuel in a stainless still cladding of 0.5 mm thickness. The total length of the fuel rod is about 2.5 m. A narrow (0.1 mm), irregular gap between the fuel and the cladding is used to transport fission gas from the fuel to the gas plenum on the top of the reactor core. Fuel rods are assembled in hexagonal structures and placed into liquid sodium coolant that circulates in the reactor core and transports the thermal energy from fuel rods to the heat exchangers. The cross-section of the fuel rod is shown schematically in Figure 2.7.

The temperature distribution across the fuel rod at normal operating conditions is shown in Figure 2.9. The power production of about 60 kW/m in the fuel rod is balanced by the heat removal, achieved by the sodium coolant, so that the temperature is in the steady state. The picture shows rapid changes of the temperature gradients across the fuel rod due to sharp changes of the

heat conductivity in the fuel, gap, and cladding. After years of operation, the surface of the fuel erodes and comes to a point-wise contact with the clad as shown in Figure 2.8. The interior of the gas gap is not resolved on the mesh level and an empirical formula for the temperature jump across the gap [11] is applied for the heat transport calculations. For the related problem of the movement of fission gases in the gas gap, the equations of flow in porous media are solved.

During either transient overheating or loss of coolant accidents, the heat transfer balance is changed and the temperature in the fuel rod increases. It can cause melting of the fuel rod and even stainless steel cladding. However, because of the change of cladding properties as the temperature increases, the cladding usually fails before melting, causing the ejection of molten fuel and fission gases into the coolant reservoir. Figures 2.10 and 2.9 depict the increase of temperature and melting of the fuel in the transient overheating accident.

2.3.2 Incompressible magnetohydrodynamics

Governing Equations

We are interested in the description of multiphase or multi-material systems involving conducting fluids interacting with neutral fluids or gases in the presence of magnetic fields. Interfaces of the phase or material separation are assumed to be sharp (the thickness of the interface is negligible) and, in general, geometrically complex. The numerical simulation of liquid metals, liquid salts, and weakly ionized plasmas, which are relatively weak electrical conduc-

tors, is difficult using the standard full systems of MHD equations [14]. Fast diffusion of the magnetic field, caused by low value of electrical conductivity, introduces unwanted small time scales into the problem. If the time scale of the diffusion of the magnetic field is small compared to hydrodynamic time scale, the magnetic Reynolds number [42]

$$Re^M = \frac{4\pi u \sigma L}{c^2},$$

where L is the typical length scale, u is the fluid velocity, and σ is the electric conductivity, is small. If, in addition, the eddy-current-induced magnetic field δB is small compared to the external field B, the full system of MHD equations can be simplified by neglecting the time evolution of the magnetic field. In this case, the generalized Ohm's law is used for the evaluation of the current-density distribution instead of the Maxwell equation $\mathbf{J} = \frac{c}{4\pi}\nabla \times \mathbf{H}$, where the magnetic field $\mathbf{H}$ and the magnetic induction $\mathbf{B}$ are related by the magnetic permeability coefficient μ: $\mathbf{B} = \mu\mathbf{H}$. The governing equations of incompressible conductive fluids in the low magnetic Reynolds number approximation are

$$\rho\left(\frac{\partial}{\partial t} + \mathbf{u}\cdot\nabla\right)\mathbf{u} = \mu\Delta\mathbf{u} - \nabla P + \rho g + \frac{1}{c}(\mathbf{J}\times\mathbf{B}) \tag{2.24}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2.25}$$

$$\mathbf{J} = \sigma\left(-\nabla\varphi + \frac{1}{c}\mathbf{u}\times\mathbf{B}\right) \tag{2.26}$$

$$\nabla\cdot\mathbf{J} = 0 \tag{2.27}$$

Taking the divergence of both sides of equation (2.26) together with equation (2.27), an elliptic equation for the electric potential is obtained:

$$\nabla \cdot (\sigma \nabla \varphi) \quad = \quad \nabla \cdot \frac{\sigma}{c}(\mathbf{u} \times \mathbf{B}) \tag{2.28}$$

If a conductive fluid interfaces a neutral fluid or gas, the current density vector is tangential to the material interface. This statement is expressed by the following Neumann boundary condition for the Poisson equation (2.28).

$$\left.\frac{\partial \varphi}{\partial \mathbf{n}}\right|_{\Gamma} \quad = \quad \left.\frac{1}{c}(\mathbf{u} \times \mathbf{B}) \cdot \mathbf{n}\right|_{\Gamma} \tag{2.29}$$

where Γ is the boundary of conductive fluid.

Hydro- and MHD Algorithms

The system of MHD equations $(2.24) - (2.27)$, a coupled parabolicelliptic system in a geometrically complex domain, is solved using operator splitting and front tracking. The propagation and redistribution of the interface using the method of front tracking ([9],[17]) is performed at the beginning of each time step. Interfaces are represented by triangle meshes that are propagated in each time step. The topology issues of the interface are resolved by the FronTier library and the only information required by the FronTier library is the discretized velocity filed in the computational domain, which is stored in the center of each computation grid. Velocity of each vertex in the interface mesh is the result of interpolation of nearby cell center velocities. Then interior states are updated by the incompressible hydro solver.

The magnetic source term ($\frac{1}{c}(\mathbf{J} \times \mathbf{B})$) is evaluated first. The discretization of equation (2.28) is similar to that in section (2.2), while the boundary condition is much simpler. Similarly, integrating equation (2.28) together with divergence theorem, we obtain.

$$\int_v \nabla \cdot (\nabla \varphi)\, dv = \oint_{\partial v} \nabla \varphi \cdot \mathbf{n}\, ds = \oint_{\partial v} \frac{1}{c}(\mathbf{u} \times \mathbf{B}) \cdot \mathbf{n}\, ds, \tag{2.30}$$

which is

$$\oint_{\partial v} \frac{\partial \varphi}{\partial \mathbf{n}}\, ds \;\; = \;\; \oint_{\partial v} \frac{1}{c}(\mathbf{u} \times \mathbf{B}) \cdot \mathbf{n}\, ds. \tag{2.31}$$

With the boundary condition (2.29), we can see that the integral along the boundary of conductive fluid in each partial cell is canceled in both sides of (2.31), and the discretization equation is greatly simplified.

After solving equation (2.28), the gradient of the electric potential φ is substituted into equation (2.26) and the current density $\mathbf{J}$ is obtained. Secondly, we deal with the equation (2.24), without regarding the divergence constraint, for an intermediate velocity $\mathbf{u}^*$. Employing an operator splitting technique, we resolve the step

$$\frac{\mathbf{u}^{'} - \mathbf{u}^n}{\Delta t} \;\; = \;\; -(\mathbf{u}^n \cdot \nabla)\mathbf{u}^n. \tag{2.32}$$

Only for the advection step, the density jump across the interface of two fluid components is smoothed with a certain smoothing radius of computation cells [52]. The advection part, equation (2.32), is evaluated explicitly, with a

second order Godunov type scheme ([6]). For the diffusion part, we employ the implicit Crank-Nicolson method. Two fluid components are solved together, disregarding the interface.

Thirdly, the diffusion step and the source term are resolved

$$\frac{\mathbf{u}^* - \mathbf{u}^{'}}{\Delta t} + \nabla q = \rho g + \frac{1}{c}(\mathbf{J} \times \mathbf{B}) + \frac{\mu}{2}\nabla^2(\mathbf{u}^* + \mathbf{u}^{'}), \tag{2.33}$$

where q is the pressure of the previous time step. Finally, we perform the projection step. Applying the divergence operator in both sides of equation

$$\mathbf{u}^* = \mathbf{u}^{n+1} + \frac{\Delta t}{\rho}\nabla\phi^{n+1} \tag{2.34}$$

and using the divergence constraint $\nabla \cdot \mathbf{u}^{n+1} = 0$, we obtain the following elliptic equation for pressure

$$\nabla^2\phi^{n+1} = \frac{\rho}{\Delta t}\nabla \cdot \mathbf{u}^*. \tag{2.35}$$

The projection step is an elliptic interface problem discussed in section (2.2). Two jump conditions for pressure are

$$[p] = \sigma\kappa \tag{2.36}$$

$$\left[\frac{1}{\rho}\frac{\partial p}{\partial \mathbf{n}}\right] = 0 \tag{2.37}$$

where κ is the curvature and σ is the surface tension coefficient. A matrix to solve such elliptic interface problem is set according to the algorithm described

in section (2.2).

The pressure is updated using the solution of the projection step:

$$p = q + \phi^{n+1} \tag{2.38}$$

The described algorithm achieves the second order convergence.

The implementation is carried out with C++ and MPI for the communication between processors. FronTier's hyperbolic solvers demonstrate good scalability on large machines of the IBM BlueGene series. The scalability of elliptic solvers is determined by the scalability of commonly used parallel libraries for sparse linear system of equations (preconditioned Krylov subspace iterative solvers of the PETSc library have been used in our MHD code).

2.3.3 Verification and Validation

Verification and validation tests for the three-dimensional FronTier-MHD code have been performed using experimental and theoretical studies of liquid mercury jets in magnetic fields. Experimental studies of a mercury jet entering a magnetic field with the magnitude satisfying the hyperbolic tangent profile have been performed in [39]. An asymptotic theoretical analysis has also been done by the same group. The experiment setup is as follows. A mercury jet with the initial diameter of 8 mm is shot horizontally into a transverse magnetic field with the initial velocity of 2.1 m/s. The amplitude of the

transverse magnetic field satisfies the following equation

$$(\frac{B_y}{B_{max}})^2 = \frac{1}{2}[1 - \tanh(\frac{z - z_0}{L_m})], \tag{2.39}$$

where z_0 is the center and L_m is the characteristic length of the magnetic field. In our simulations, $z_0 = 1.5$ cm and $L_m = 0.62$ cm.

As predicted in [39], the magnitude of expansion of the jet depends on the z value:

$$\begin{aligned} g^*_{s2}(z'^*) &= \beta sin(\alpha z'^*) \int_{-\infty}^{z'^*} \frac{cos(\alpha t)}{cosh^2(\varepsilon_m t)}\, dt \\ &\quad -\beta cos(\alpha z'^*) \int_{-\infty}^{z'^*} \frac{cos(\alpha t)}{cosh^2(\varepsilon_m t)}\, dt \end{aligned} \tag{2.40}$$

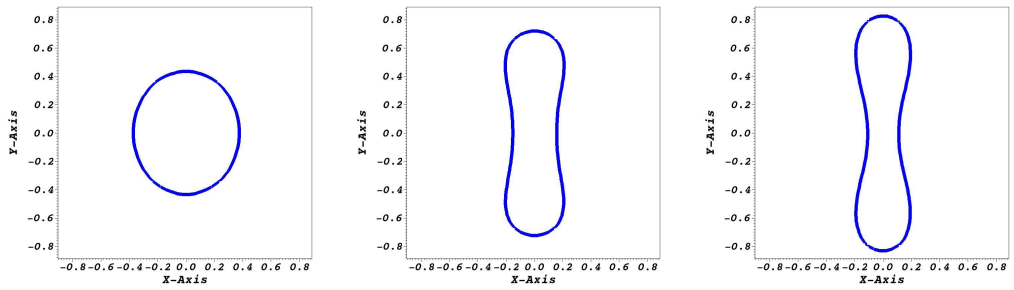
where $\alpha = \sqrt{6/W_a}$ and $\beta = \varepsilon_m N_a / 8\alpha$. And

- $N_a = \sigma_e B^2_{max} a/\rho_f \omega_0$ is the Stuart number of the jet, σ_e is the electric conductivity of mercury, a is the radius of the cross-section of the jet, ρ_f is the density of mercury and ω_0 is the main flow velocity which is 2.1m/s.

- $W_a = \rho_f a \omega_0^2/\sigma$ is the Weber number of the jet, σ is the surface tension of mercury.

- $\varepsilon_m = a/L_m$ and $z'^* = z/a$.

Numerical simulation was performed using the magnetic filed strength B_{max} of 1.41 T and 1.88 T. In order to save computation time, simulations



(a) (b) (c)

Figure 2.11: FronTier-MHD simulation of jet deformation in magnetic field. Cross sections of the jet are shown at observation points located at 0, 3.5, and 5.5 cm.

were performed in a frame moving with the initial jet velocity. In the asymptotic analysis of [39], the jet was assumed to extend infinitely and reach the steady state. To simulate similar conditions, initially long cylindrical jet was moving through the magnetic field rather then being ejected from the nozzle. Also, the jet is assumed to be in the vacuum while in the simulation, the vacuum was substituted with light gas, with the density 10^4 times smaller than the density of mercury. With such a large density ratio, the influence of gas on the momentum of the mercury jet can be ignored. In order to obtain accurate profile of the electric current density, the computational mesh contained approximately 20 cells across the cross-section of the mercury jet.

Experimental results of the jet deformation from [39], results of asymptotic analysis, and numerical simulations are plotted in Figure 2.12 for 1.88 T magnetic field and in Figure 2.13 for 1.41 T field. We observe a very good agreement of simulations with asymptotic calculations at small distances from the magnetic field center corresponding to smaller jet deformations. The ex-

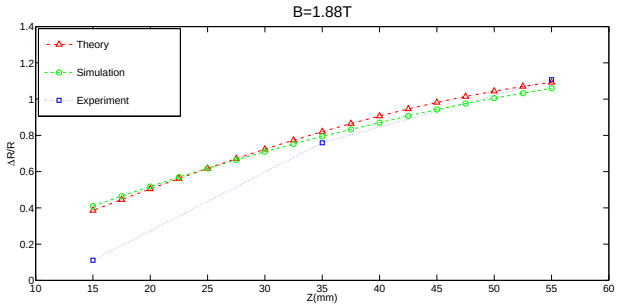


Figure 2.12: Mercury jet deformation as function of the distance from the magnetic field center for 1.88 T magnetic field. Results of simulations (green dashed line), asymptotic calculations (red dash-dotted line), and experiments (blue dotted line) are shown.

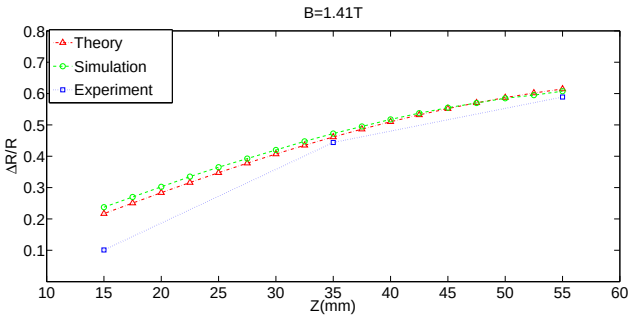


Figure 2.13: Mercury jet deformation as function of the distance from the magnetic field center for 1.41 T magnetic field. Results of simulations (green dashed line), asymptotic calculations (red dash-dotted line), and experiments (blue dotted line) are shown.

pected disagreement with experimental results at small distances can be explained by the fact that experiments were carried out using a cylindrical nozzle located at $z = 0$ that reduced jet deformations compared to long free jets. But at larger distances from the nozzle corresponding to larger jet deformations, numerical simulations, theoretical calculations, and experiments are all in agreement.

We would like to comment on the importance of maintaining a sharp

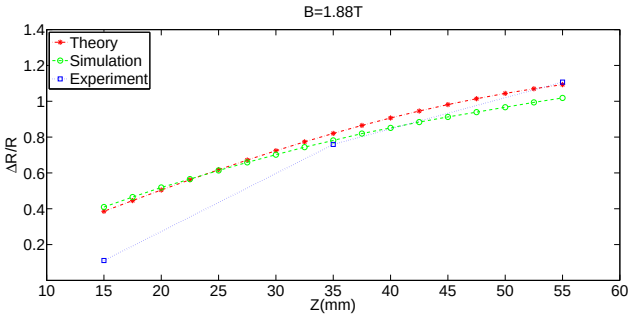


Figure 2.14: Degradation of accuracy without sharp density discontinuity. Mercury jet deformation as function of the distance from the magnetic field center for 1.88 T magnetic field. Results of untracked simulations (green dashed line), asymptotic calculations (red dash-dotted line), and experiments (blue dotted line) are shown.

density discontinuity via the front tracking and embedded boundary methods. Without the embedded boundary method, the density ratio that interior solving can handle is limited by high condition number of the corresponding matrix of projection step. In order to perform the simulation without the embedded boundary method, we artificially increased the density of ambient gas so that the density ratio dropped to 10. Figures 2.14 and 2.15 demonstrate the degradation of accuracy of simulations if the correct density ratio and sharp density discontinuity are not resolved. Keeping the discontinuity sharp is even more important for applications involving more extreme flow regimes.

Further Applications

Both compressible and incompressible fluid FronTier-MHD code are used for the simulation of processed relevant to energy research and accelerator applications. Simulations of the mercury target for the Muon Accelerator Project (http://map.fnal.gov) is among the most important applications of the code.

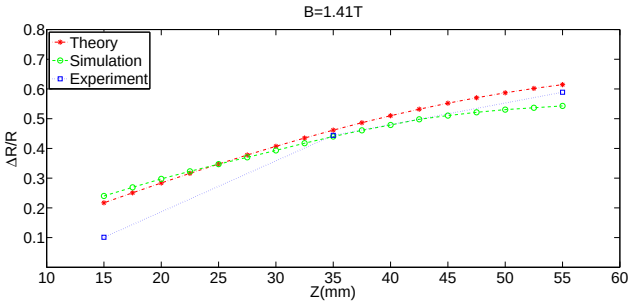


Figure 2.15: Degradation of accuracy without sharp density discontinuity. Mercury jet deformation as function of the distance from the magnetic field center for 1.41 T magnetic field. Results of untracked simulations (green dashed line), asymptotic calculations (red dash-dotted line), and experiments (blue dotted line) are shown.

The target will contain a series of 30-cm-long and 1-cm-diameter mercury jets entering a strong ($\sim$ 15 Tesla) magnetic field at a small angle to the solenoid axis. When each jet reaches the center of the solenoid, it interacts with a powerful proton pulse penetrating the jet and depositing energy[46] of the order of 100 J/g into mercury. The purpose of our numerical simulations is to evaluate states of the target before and after the interaction with protons to optimize the target design. The compressible code deals with the jet instabilities due to external energy deposition and their partial stabilization by the magnetic field [45]. The incompressible FronTier MHD code is used for the simulation of liquid metal jets in magnetic fields of different configurations prior to the interaction with proton pulses. Other applications involve pellet ablation in tokamaks [48], and plasma jet liners for magneto-inertial fusion [26].

Chapter 3

Smoothed Particle Hydrodynamics

3.1 Main Equations and Approximations

Although the hybrid Lagrangian-Eulerian fluid algorithm has been developed for years, it still faces huge difficulties when solving problems involving complex interfaces. In this chapter and the following chapter, the pure Lagrangian techniques will be discussed. The particle techniques are Lagrangian techniques that discretize continuous volume into small elements called particles and solves the underlying governing equations based on such discretization. For flows, the particles will be Lagrangian material parcels forming the discretization of Lagrangian specification of the flow field. There are several advantages of the particle technique[37]. First, the technique can exactly process pure convection because of the intrinsic Lagrangian property. Second, it is a mass conservative technique. Each particle has unchanged mass and such quantity is carried along with the movement of particle. Third, by replacing the material parcels with particles, it is easy to handle free surface problems.

3.1.1 Approximation of a field function

Smoothed particle hydrodynamics[37, 38, 35, 31, 21] method is one of Lagrangian particle techniques. It has been utilized to study astrophysics[51] in the beginning and later extended to study fluid dynamics[50] and solid mechanics[29]. It replaces the material parcels to be particles and there is no need to consider the deformation of material parcels. By using the δ function, the value a function can be written in integral form(3.1)

$$f(\mathbf{x}) = \int f(\xi)\delta(\mathbf{x} - \xi)\, d\xi \tag{3.1}$$

Using a bell shaped kernel function $W(\mathbf{x}-\xi, h)$ to approximate the δ function with second order accuracy in terms of h, we obtain (3.2)

$$f(\mathbf{x}) = \int_{\Omega(\mathbf{x},h)} f(\xi)W(\mathbf{x} - \xi, h)\, d\xi \tag{3.2}$$

Here, $\Omega(\mathbf{x}, h)$ is a domain described by $\mid \mathbf{x} - \xi \mid \leq h$. By descretizing Ω, we obtain (3.3)

$$f_i = \sum_j f_j W(\mathbf{x_i} - \mathbf{x_j}, h)V_j \tag{3.3}$$

Replacing the volume by division of mass and density

$$f_i = \sum_j m_j \frac{f_j}{\rho_j} W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{3.4}$$

This is the kernel estimation of a function value. As an example of the use of kernel estimation, suppose f is the density ρ. The interpolation formula then

gives the following estimation for the density at a point $\mathbf{x}$

$$\rho_i = \sum_j m_j W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{3.5}$$

showing that the mass is smeared to estimate the density.

3.1.2 First derivatives

With the theorem of integration by parts, if f is a differentiable function,

$$(\nabla f)_i = \sum_j m_j \frac{f_j}{\rho_j} \nabla W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{3.6}$$

In smoothed particle hydrodynamics, the derivative is evaluated by the exact derivative of an approximate function. However, this formula is not equal to zero when f is a constant. One way to guarantee that the derivative vanishes is rewriting

$$\nabla f = \frac{1}{g}(\nabla(gf) - f\nabla g) \tag{3.7}$$

where g is a differentiable function. The corresponding smoothed particle hydrodynamics form is

$$(\nabla f)_i = \frac{1}{g_i} \sum_j m_j \frac{g_j}{\rho_j}(f_j - f_i) \nabla W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{3.8}$$

which vanishes if f is a constant. if choosing $g = 1$, (3.8) becomes

$$(\nabla f)_i = \sum_j \frac{m_j}{\rho_j}(f_j - f_i) \nabla W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{3.9}$$

and, if choosing $g = \rho$, (3.8) becomes

$$(\nabla f)_i = \frac{1}{\rho_i} \sum_j m_j (f_j - f_i) \nabla W(\mathbf{x_i} - \mathbf{x_j}, h) \tag{3.10}$$

3.1.3 Kernel function

The kernel function plays an important role in the smoothed particle hydrodynamics method and has significant impact on the performance of such method. Basically, kernel function should satisfy several conditions, such as positivity, compact support, and unity and monotonicity. Positivity means that the kernel function will always be positivity in the interior area of the support domain. Compact support means that the kernel function should vanish in the boundary of support domain. Unity condition is that the integral of kernel function in the support domain is one. Monotonicity is that the kernel function will decrease with the increasing of distance. There are many kernel functions and how to choose a proper kernel function is quite empirical and problem dependent. Here, four common kernel functions are listed. And, for simplicity, let $r = \mid \mathbf{x} - \mathbf{x}' \mid$, $q = \frac{q}{h}$

Gaussian

$$W(r, h) = \alpha_D exp(-q2) \tag{3.11}$$

where α_D is $\frac{1}{\pi h^2}$ in 2D and $\frac{1}{\pi^{\frac{3}{2}} h^2}$ in 3D.

Gaussian function is smooth and infinite differentiable. The only drawback

of Gaussian function is that it vanishes in the positive and negative infinity. Usually, the support domain is $2h$, thus all the above mentioned conditions do not strictly hold.

Quadratic

$$W(r,h) = \alpha_D \left[\frac{3}{16}q^2 - \frac{3}{4}q + \frac{3}{4}\right] \qquad 0 \le q \le 2 \tag{3.12}$$

where α_D is $\frac{2}{\pi h^2}$ in 2D and $\frac{4}{4\pi h^3}$ in 3D.

Cubic spline

$$W(r,h) = \alpha_D \begin{cases} 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3 & 0 \le q \le 1 \\ \frac{1}{4}(2-q)^3 & 1 \le q \le 2 \\ 0 & q \ge 2 \end{cases} \tag{3.13}$$

where α_D is $\frac{10}{7\pi h^2}$ in 2D and $\frac{1}{\pi h^3}$ in 3D.

Cubic spline function is the most common used kernel function.

Quintic

$$W(r,h) = \alpha_d (1 - \frac{q}{2})^2 (2q+1) \qquad 0 \le q \le 2 \tag{3.14}$$

where α_D is $\frac{7}{4\pi h^2}$ in 2D and $\frac{21}{16\pi h^3}$ in 3D.

3.1.4 Analysis for the accuracy [31]

Approximation of a field function

With smoothed particle hydrodynamics method, the approximation of a field function can be obtained by the form of integral representation (3.2). Gui-Rong Liu and M. B. Liu offer an analysis of the accuracy of kernel approximation in their book [31]. If $f(\mathbf{x})$ is sufficiently smooth, by applying the Taylor series expansion of $f(\mathbf{x}')$ in the vicinity of $\mathbf{x}$, we can obtain

$$\begin{aligned} f(\mathbf{x}') &= f(\mathbf{x}) + f'(\mathbf{x})(\mathbf{x}' - \mathbf{x}) + \frac{1}{2} f''(\mathbf{x})(\mathbf{x}' - \mathbf{x})^2 \\ &= \sum_{k=0}^{n} \frac{(-1)^k h^k f^{(k)}(\mathbf{x})}{k!} (\frac{\mathbf{x}' - \mathbf{x}}{h})^k + O((\frac{\mathbf{x}' - \mathbf{x}}{h})^{n+1}) \end{aligned} \tag{3.15}$$

where $O((\frac{\mathbf{x}'-\mathbf{x}}{h})^{n+1})$ is the remainder of the Taylor series expansion. Substituting equation (3.15) into (3.2) yields

$$\begin{aligned} f(\mathbf{x}) &= \int_\Omega \sum_{k=0}^{n} \frac{(-1)^k h^k f^{(k)}(\mathbf{x})}{k!} (\frac{\mathbf{x}' - \mathbf{x}}{h})^k W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' + r_n(\frac{\mathbf{x}' - \mathbf{x}}{h}) \\ &= \sum_{k=0}^{n} \frac{(-1)^k h^k f^{(k)}(\mathbf{x})}{k!} \int_\Omega (\frac{\mathbf{x}' - \mathbf{x}}{h})^k W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' + r_n(\frac{\mathbf{x}' - \mathbf{x}}{h}) \\ &= \sum_{k=0}^{n} C_k f^{(k)}(\mathbf{x}) + r_n(\frac{\mathbf{x}' - \mathbf{x}}{h}) \end{aligned} \tag{3.16}$$

where

$$C_k = \frac{(-1)^k h^k}{k!} \int_\Omega (\frac{\mathbf{x}' - \mathbf{x}}{h})^k W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \tag{3.17}$$

By comparing two sides of equation (3.16), in order for $f(\mathbf{x})$ to have n-th order approximation, we can notice that the coefficients C_k must equal to the

counterparts for $f^{(k)}(\mathbf{x})$ in the left hand side of equation(3.16). Thus, we can obtain the following conditions for kernel function $W(\mathbf{x}' - \mathbf{x}, h)$

$$
\begin{aligned}
C_0 &= \int_\Omega W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 1 \\
C_1 &= -h \int_\Omega (\frac{\mathbf{x}' - \mathbf{x}}{h}) W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0 \\
C_2 &= \frac{h^2}{2!} \int_\Omega (\frac{\mathbf{x}' - \mathbf{x}}{h})^2 W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0 \\
&\vdots \\
C_n &= \frac{(-1)^n h^n}{n!} \int_\Omega (\frac{\mathbf{x}' - \mathbf{x}}{h})^n W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0
\end{aligned} \tag{3.18}
$$

"These conditions could be further written in the following simplified expressions in terms of the k-th moments M_k of kernel function"[31]

$$
\begin{aligned}
M_0 &= \int_\Omega W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 1 \\
M_1 &= \int_\Omega (\mathbf{x}' - \mathbf{x}) W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0 \\
M_2 &= \int_\Omega (\mathbf{x}' - \mathbf{x})^2 W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0 \\
&\vdots \\
M_n &= \int_\Omega (\mathbf{x}' - \mathbf{x})^n W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0
\end{aligned} \tag{3.19}
$$

We can notice that the first equation in (3.19) is the unity condition, while the second equation is the symmetric condition for the kernel function[31]. Imposing these two conditions ensures the first order consistency for smoothed particle hydrodynamics kernel approximation of a function.

Approximation of the derivatives

In many physics systems, the governing equations are up to second order. Thus, the analysis of approximation of the first and second derivatives of a field function will be focused on.

For First derivative, the approximation can be obtained by substituting the function $f(\mathbf{x})$ in equation(3.2) with its first derivative $f'(\mathbf{x})$,

$$f'(\mathbf{x}) = \int_{\Omega} f'(\mathbf{x}')W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \tag{3.20}$$

Withe the theorem of integral by parts

$$f'(\mathbf{x}) = \int_{S} f(\mathbf{x}')W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS - \int_{\Omega} f(\mathbf{x}')W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \tag{3.21}$$

Replacing $f(\mathbf{x}')$ with equation(3.15)

$$\begin{aligned} f'(\mathbf{x}) &= \int_{S} f(\mathbf{x}')W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS \\ &\quad - \int_{\Omega} \sum_{k=0}^{n} \frac{(-1)^k h^k f^{(k)}(\mathbf{x})}{k!} (\frac{\mathbf{x}' - \mathbf{x}}{h})^k W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' + r_n(\frac{\mathbf{x}' - \mathbf{x}}{h}) \\ &= \int_{S} f(\mathbf{x}')W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS \\ &\quad - \sum_{k=0}^{n} \frac{(-1)^k h^k f^{(k)}(\mathbf{x})}{k!} \int_{\Omega} (\frac{\mathbf{x}' - \mathbf{x}}{h})^k W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' + r_n(\frac{\mathbf{x}' - \mathbf{x}}{h}) \\ &= \int_{S} f(\mathbf{x}')W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS + \sum_{k=0}^{n} C'_k f^{(k)}(\mathbf{x}) + r_n(\frac{\mathbf{x}' - \mathbf{x}}{h}) \end{aligned} \tag{3.22}$$

where

$$C'_k = \frac{(-1)^{k+1} h^k}{k!} \int_{\Omega} (\frac{\mathbf{x}' - \mathbf{x}}{h})^k W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \tag{3.23}$$

Similar to the previous section, if the following equations hold, the approximation of $f'(\mathbf{x})$ is n-th order consistent.

$$\begin{aligned}
M'_0 &= \int_\Omega W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 1 \\
M'_1 &= \int_\Omega (\mathbf{x}' - \mathbf{x}) W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0 \\
M'_2 &= \int_\Omega (\mathbf{x}' - \mathbf{x})^2 W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0 \\
&\vdots \\
M'_n &= \int_\Omega (\mathbf{x}' - \mathbf{x})^n W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' = 0
\end{aligned} \tag{3.24}$$

and

$$W(\mathbf{x}' - \mathbf{x}, h)|_S = 0 \tag{3.25}$$

For second derivative, similarly, we have

$$f''(\mathbf{x}) = \int_\Omega f''(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \tag{3.26}$$

Integrating by parts, where S is the boundary of Ω we obtain

$$\begin{aligned}
f''(\mathbf{x}) &= \int_S f'(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS - \int_\Omega f'(\mathbf{x}') W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \\
&= \int_S f'(\mathbf{x}') W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS - \int_S f(\mathbf{x}') W'(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS \\
&+ \int_\Omega f(\mathbf{x}') W''(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}'
\end{aligned} \tag{3.27}$$

Substituting $f(\mathbf{x}')$ with equation(3.15) again

$$\begin{aligned}
f''(\mathbf{x}) &= \int_S f'(\mathbf{x}')W(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS - \int_S f(\mathbf{x}')W'(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS \\
&+ \int_\Omega \sum_{k=0}^{n} \frac{(-1)^k h^k f^{(k)}(\mathbf{x})}{k!}(\frac{\mathbf{x}'-\mathbf{x}}{h})^k W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}' + r_n(\frac{\mathbf{x}'-\mathbf{x}}{h}) \\
&= \int_S f'(\mathbf{x}')W(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS - \int_S f(\mathbf{x}')W'(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS \\
&+ \sum_{k=0}^{n} \frac{(-1)^k h^k f^{(k)}(\mathbf{x})}{k!}\int_\Omega (\frac{\mathbf{x}'-\mathbf{x}}{h})^k W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}' + r_n(\frac{\mathbf{x}'-\mathbf{x}}{h}) \\
&= \int_S f'(\mathbf{x}')W(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS - \int_S f(\mathbf{x}')W'(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS \\
&+ \sum_{k=0}^{n} C_k'' f^{(k)}(\mathbf{x}) + r_n(\frac{\mathbf{x}'-\mathbf{x}}{h})
\end{aligned} \tag{3.28}$$

where

$$C_k'' = \frac{(-1)^{k+1}h^k}{k!}\int_\Omega (\frac{\mathbf{x}'-\mathbf{x}}{h})^k W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}' \tag{3.29}$$

We can see that if the following equations hold, the approximation of $f''(\mathbf{x})$ is n-th order consistent.

$$\begin{aligned}
M_0'' &= \int_\Omega W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}' = 1 \\
M_1'' &= \int_\Omega (\mathbf{x}'-\mathbf{x})W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}' = 0 \\
M_2'' &= \int_\Omega (\mathbf{x}'-\mathbf{x})^2 W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}' = 0 \\
&\vdots \\
M_n'' &= \int_\Omega (\mathbf{x}'-\mathbf{x})^n W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}' = 0
\end{aligned} \tag{3.30}$$

and

$$W(\mathbf{x}' - \mathbf{x}, h)|_S = 0 \tag{3.31}$$

$$W'(\mathbf{x}' - \mathbf{x}, h)|_S = 0 \tag{3.32}$$

Conditions for n-th order approximations of field function and the first two derivatives

Using the theorem of integral by parts, given equation (3.25), we can obtain first expression in equation (3.24).

$$\begin{aligned}\int_\Omega W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' &= \int_S 1 \cdot W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS - \int_\Omega (1)' \cdot W(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \\ &= \int_S W(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS = 0\end{aligned} \tag{3.33}$$

Similarly, given equation (3.32)

$$\begin{aligned}\int_\Omega W''(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' &= \int_S 1 \cdot W'(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS - \int_\Omega (1)' \cdot W'(\mathbf{x}' - \mathbf{x}, h)\, d\mathbf{x}' \\ &= \int_S W'(\mathbf{x}' - \mathbf{x}, h) \cdot \mathbf{n}\, dS = 0\end{aligned} \tag{3.34}$$

If equations (3.25) and (3.32) are satisfied, equations (3.24) and (3.30) except the first equations can be obtained with equations(3.19) by using the following

integration by parts

$$\begin{aligned}\int_\Omega (\mathbf{x}'-\mathbf{x})^k W(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x} &= -\frac{1}{k+1}\int_\Omega \left[(\mathbf{x}'-\mathbf{x})^{k+1}\right]' W(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x} \\ &= -\frac{1}{k+1}\Big(\int_S (\mathbf{x}'-\mathbf{x})^{k+1} W(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS \\ &\quad -\int_\Omega (\mathbf{x}'-\mathbf{x})^{k+1} W'(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x}\Big) \\ &= \frac{1}{k+1}\int_\Omega (\mathbf{x}'-\mathbf{x})^{k+1} W'(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x} \quad (3.35)\end{aligned}$$

and

$$\begin{aligned}\int_\Omega (\mathbf{x}'-\mathbf{x})^k W'(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x} &= -\frac{1}{k+1}\int_\Omega \left[(\mathbf{x}'-\mathbf{x})^{k+1}\right]' W'(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x} \\ &= -\frac{1}{k+1}\int_S (\mathbf{x}'-\mathbf{x})^{k+1} W'(\mathbf{x}'-\mathbf{x},h)\cdot\mathbf{n}\,dS \\ &\quad -\int_\Omega (\mathbf{x}'-\mathbf{x})^{k+1} W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x} \\ &= \frac{1}{k+1}\int_\Omega (\mathbf{x}'-\mathbf{x})^{k+1} W''(\mathbf{x}'-\mathbf{x},h)\,d\mathbf{x} \quad (3.36)\end{aligned}$$

In conclusion, in order to ensure the approximation of field function and its first two derivatives to be n-th order consistent, the kernel function should satisfy equations (3.19), (3.25) and (3.32). Many kernel functions can satisfy such equations.

3.2 Discretization of Lagrangian fluid equations and related algorithms

A parallelized smoothed particle hydrodynamics code has been developed with C++ and MPI. The code structure is inspired by the ideas of SPHyics code developed by Gmez-Gesteira et al. [20]. This code is capable to simulate compressible flow of free surface by solving the Euler equations that govern the system.

3.2.1 Algorithms of the code

The spatial discretization of the governing equations is undertaken by smoothed particle hydrodynamics method. And, the time integral is performed by a second order Runge-Kutta scheme. There is a main loop in the code to control the total physics time of the code as well as the maximum time steps. In each time step, the states of each particle are updated through the solving of corresponding equations. To ensure the stability, artificial viscosity is introduced to the code.

Governing equations of the system

The basic form of Euler equations for compressible flow (omitting the diffusion term) is

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \tag{3.37}$$

$$\frac{\partial (\rho \mathbf{u})}{\partial t} + \nabla p = 0 \tag{3.38}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot (\mathbf{u}(E+p)) = 0 \tag{3.39}$$

To close this system, an equation of state is required

$$E = E(\rho, p) \tag{3.40}$$

By substituting the Nabla(∇) operator with smoothed particle hydrodynamics method representation, and including artificial viscosity as well as some other particle related modification terms into the equations, the following equations are obtained

$$\frac{d\rho_a}{dt} = \sum_b m_b \mathbf{v}_{ab} \nabla_a W_{ab} \tag{3.41}$$

$$\frac{d\mathbf{v}_a}{dt} = -\sum_b m_b \left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} + \Pi_{ab}\right) \nabla_a W_{ab} + \mathbf{g} \tag{3.42}$$

$$\frac{de_a}{dt} = \frac{1}{2} \sum_b m_b \left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} + \Phi_{ab}\right) \mathbf{v}_{ab} \nabla_a W_{ab} \tag{3.43}$$

The Π_{ab} is the artificial viscosity[38]

$$\Pi_{ab} = \alpha_D \begin{cases} \frac{-\alpha \bar{C_{ab}} \mu_{ab}}{\bar{\rho_{ab}}} & \vec{v}_{ab}\vec{r}_{ab} < 0 \\ 0 & \vec{v}_{ab}\vec{r}_{ab} > 0 \end{cases} \tag{3.44}$$

with $\mu_{ab} = \frac{h\vec{v}_{ab}\vec{r}_{ab}}{\vec{r}_{ab}^2+\eta^2}$, $\vec{r}_{ab} = \vec{r}_a - \vec{r}_b$, $\vec{v}_{ab} = \vec{v}_a - \vec{v}_b$, $\bar{C_{ab}} = \frac{C_a+C_b}{2}$, $\bar{\rho_{ab}} = \frac{\rho_a+\rho_b}{2}$, and $\eta^2 = 0.01h^2$. α is a free parameter for problem. The stiffened polytropic EOS model is

$$e = \frac{p + \gamma p_{inf}}{(\gamma - 1)\rho} - e_{inf} \tag{3.45}$$

A feature of Lagrangian particle is that it moves with the flow, the updating formula for the particle position is in need. In the code, particles are moved with XSPH variant[36].

$$\frac{d\mathbf{r}_a}{dt} = \mathbf{v}_a + \epsilon \sum_b \frac{m_b}{\bar{\rho_{ab}}} \mathbf{v}_{ba} W_{ab} \tag{3.46}$$

Here, $\epsilon = 0.5$, $\bar{\rho_{ab}} = (\rho_a + \rho_b)/2$. By utilizing this method, particles are moved with a velocity close to the average velocity in it vicinity.

Time integral scheme

Second order Runge-Kutta scheme and symplectic Verlet scheme have been implemented. Write the Euler equations and the position updating for-

mula in the following form

$$\begin{aligned}
\frac{d\mathbf{v}_a}{dt} &= \mathbf{A}_a \\
\frac{\rho_a}{dt} &= F_a \\
\frac{e_a}{dt} &= E_a \\
\frac{d\mathbf{r}_a}{dt} &= \mathbf{V}_a
\end{aligned} \tag{3.47}$$

The second order Runge-Kutta explicit scheme

$$y_{n+1} = y_n + hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(t_n, y_n)) \tag{3.48}$$

Substituting equations (3.47) individually into equation (3.48), we obtain

$$\begin{aligned}
\mathbf{v}_a^{n+1} &= \mathbf{v}_a^n + \Delta t\mathbf{A}(t_n + \frac{\Delta t}{2}, \mathbf{v}_a^n + \frac{\Delta t}{2}\mathbf{A}(t_n, \mathbf{v}_a^n)) \\
\rho_a^{n+1} &= \rho_a^n + \Delta tF(t_n + \frac{\Delta t}{2}, \rho_a^n + \frac{\Delta t}{2}F(t_n, \rho_a^n)) \\
e_a^{n+1} &= e_a^n + \Delta tE(t_n + \frac{\Delta t}{2}, e_a^n + \frac{\Delta t}{2}E(t_n, e_a^n)) \\
\mathbf{r}_a^{n+1} &= \mathbf{r}_a^n + \Delta t\mathbf{V}(t_n + \frac{\Delta t}{2}, \mathbf{r}_a^n + \frac{\Delta t}{2}\mathbf{V}(t_n, \mathbf{r}_a^n))
\end{aligned} \tag{3.49}$$

There is a trade off between order of accuracy and space complexity. The higher the order is, the more temporary space will be needed.

The predictor-corrector scheme is not a symplectic scheme thus the energy is not conservative. In order to simulate some real world phenomena, the conservation of energy is critical to ensure the stability of the system. The

Verlet Scheme is

$$\begin{aligned}
\mathbf{v}_a^{n+1} &= \mathbf{v}_a^{n-1} + 2\Delta t \mathbf{A}_a^n \\
\rho_a^{n+1} &= \rho_a^{n-1} + 2\Delta t D_a^n \\
e_a^{n+1} &= e_a^{n-1} + 2\Delta t E_a^n \\
\mathbf{r}_a^{n+1} &= \mathbf{r}_a^{n-1} + \Delta t \mathbf{V}_a^n + 0.5\Delta t^2 \mathbf{A}_a^n
\end{aligned} \tag{3.50}$$

3.2.2 Neighbour searching

In the development of the code, one of the crucial part is the neighbour searching method The performance of neighbour searching algorithm will greatly affect the performance of the code. The kernel function is effective in a radial domain with finite radius, in many cases, which is two times the smoothing radius h.

Bucket Searching

In the code, the bucket searching method is employed. Since each particle only interacts with adjacent neighbour particles, by dividing the whole computation domain with searching bucket (square in 2D and cube in 3D) and leveraging the locality of searching buckets, it is trivial to build up the neighbour list of a certain particle.

For visually simplicity, a 2D example is presented here. The searching radius is $2h$, and, if the searching bucket edge length is $2h$, it is obvious that

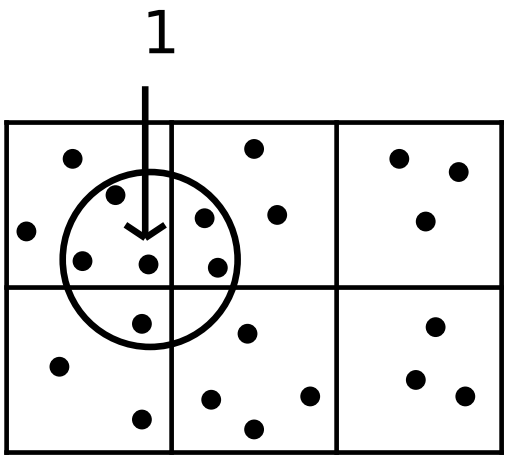


Figure 3.1: Bucket Searching

all neighbours of a particle will be in the same bucket that the center particle lies in and one layer of neighbour searching bucket because the minimum distance between center particle and the boundary of this searching stencil is $2h$. From Figure 3.1, particle 1 is the particle of interest, and particles lying in the searching circle are neighbour particles. It is always that the union of searching buckets does not exactly overlap but is larger than the computational domain.

With bucket searching method, the time complexity is $O(N)$ when building the buckets and $O(1)$ when searching for neighbours for most cases. Given the edge length of bucket and the starting point, it costs $O(1)$ time to find out in which bucket a certain particle lies. In addition, bucket searching method is relatively easy to be parallelized, because of the spatial domain decomposition nature of such method. The bucket searching method is very efficient when the distances between particles are relatively a constant.

Octree Searching

The octree searching[8] routine is also implemented(for 2D, it is quadtree) by our group. The idea is building up an 2^n-tree data structure to divide the spatial domain, storing all the particle position in such data structure and searching the neighbours via the tree. The depth of the tree is very important. As the increasing of the depth, building time is also increasing. An empirical number is 4 or 5. The tree searching method costs $O(NlogN)$ time to build up the tree and roughly O(logN) time to obtain the neighbour list. Generally speaking, octree searching routine is not efficient in some circumstances. The details of octree searching algorithm will be discussed in the next chapter.

3.2.3 Parallelization

In many simulation, it is necessary to process with large number of particles. If the particle number is too large, there could be two problems. One is that the memory of a single computer can not store all the data structure, and the other one is that the simulation time is very long. Thus, it is critical for the code to be parallelized for real world simulation.

Based on the bucket searching method, the computation domain can be divided into many subdomains. For each subdomain, there is a buffer layer. The buffer is crucial for calculating the interaction of particles. The thickness of the buffer is the same as the edge length of searching bucket ensuring that each particle will obtain the same neighbour list in both serial case and parallel case with same initial settings. In the beginning of each time step,

the latest particle information is put into the buffer layer. In the end of each time step, because particles can move into the buffer layer, a particle synchronization procedure is carried out to update the particle information in each subdomain. This is a major difference between particle based parallelization and structure grid based parallelization. Such procedure is sometimes called particle management.

The load balance and scaling of this parallelization depend on the distribution of particles. When particles concentrate to a relatively small cubic (in 2D it is square) domain with almost constant distance, the parallelization will have good performance.

3.3 Applications of SPH

Two simulations which are in weakly compressible regime are performed by the code. The stiffened polytropic EOS(Equation of State) model as well as Tait's EOS model are employed to simulate the weakly compressibility of the fluid flow.

3.3.1 Mercury jet entrance into mercury pool

Physics Background[46]

The main driver for computational magnetohydrodynamics is the research in magnetically conned nuclear fusion which has recently been boosted by the International Thermonuclear Experimental Reactor project (ITER, [1]). Numerical algorithms for nuclear-fusion-simulation research are optimized for

highly conductive, fully ionized plasmas. In simplied studies, even the innite-conductivity approximation is widely used (ideal MHD approximation [13]). But even in thermonuclear fusion devices such as tokamaks, low-conductivity, weakly ionized plasma may still be present under special conditions despite very high temperature (the nuclear fusion ignition temperature is of the order of 10 keV or 108 K). In tokamaks, weakly ionized plasma is found in the ablated clouds of cryogenic fuel pellets. The injection of such frozen deuterium - tritium pellets is considered the most ecient technique for the fueling of tokamaks[40]. In order to visualize the movement of mercury, a level function liked approach is adopted. This approach allows approximated interfaces to be shown and calculating interpolated interior state values on a Cartesian mesh. With the state values in a regular grid, better analysis of the simulation result can be processed.

Simulation result

Simulation has been done with smoothed particle hydrodynamics code, as shown in Figures 3.2,3.3,3.4.

3.3.2 Mercury jet interacting with proton beams

Physics Background and simulation

As mentioned in the previous section, the mercury target is proposed to be collided with proton beams. This collision will bring forth high energy density in the mercury as well as trigger the mercury to blow up. In order to

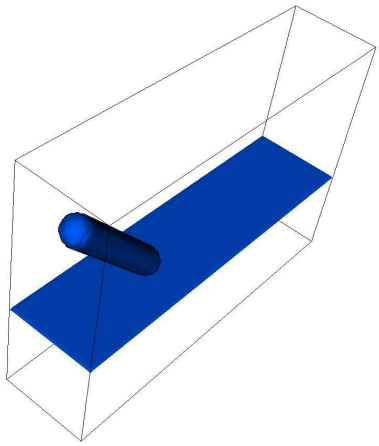


Figure 3.2: Initial state of the jet

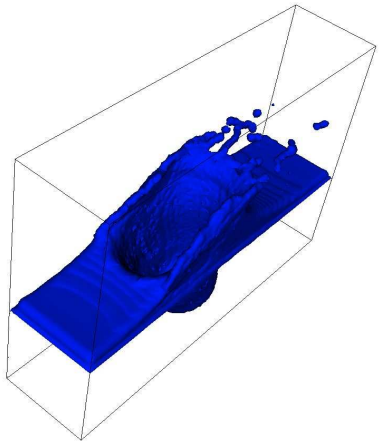


Figure 3.3: The state in the middle of simualtion

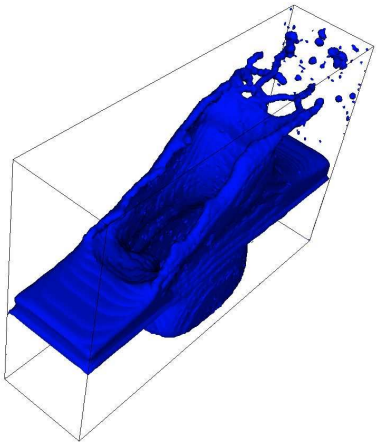


Figure 3.4: The state in end of simulation

study such phenomenon, an experiment was designed. Mercury was placed in a thimble(Figure 3.5) and bombarded by proton beams. As the increasing of energy deposition, the mercury will erupt from the thimble.

Simulation result

A collaborator Hsin-Chiang Chen made some simulation with smoothed particles hydrodynamics code. By comparing the simulation result and experiment result(Figures 3.6, 3.7, 3.8), we can see that the pressure, velocity and height values obtained by simulation is similar to experiment results.

3.4 Analysis of Accuracy and Deficiency of SPH

Smoothed Particle Hydrodynamics is a conserved method and has great stability over time which comes from the conservative nature of such technique. However, there is also a significant drawback preventing a broader application

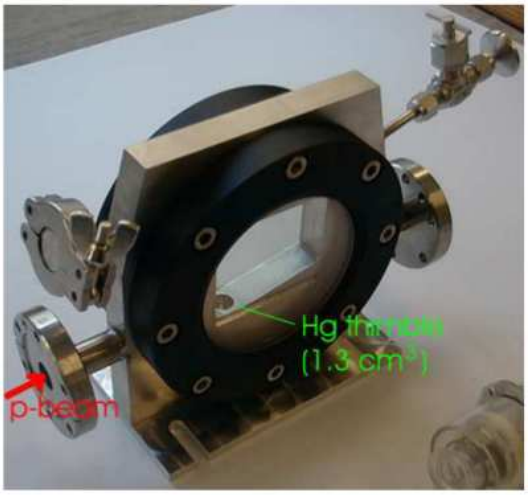


Figure 3.5: Mercury thimble

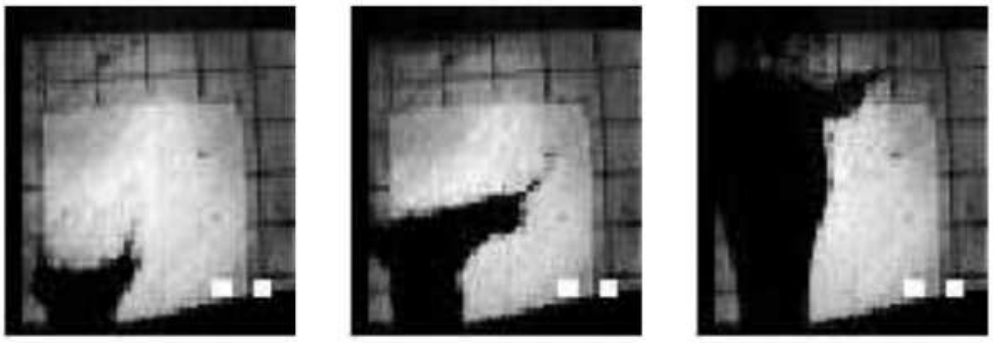


Figure 3.30: mercury splash at $t = 0.88, 0.125, 0.7$ ms after proton impact of $3.7\,10^{12}$ protons (thimble)

Figure 3.6: Mercury splash (thimble)

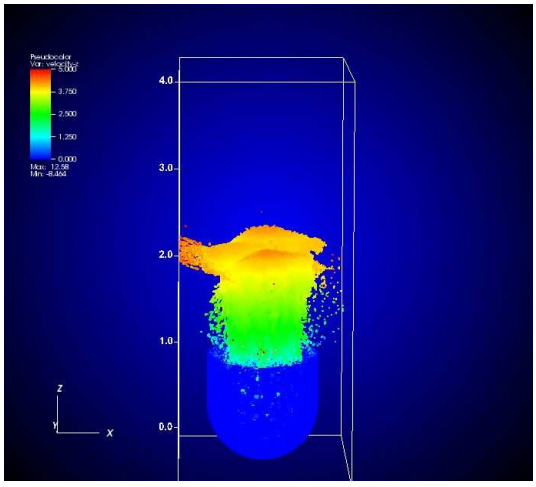


Figure 3.7: Mercury splash (thimble) at time 0.5 ms produced by Hsin-Chiang Chen

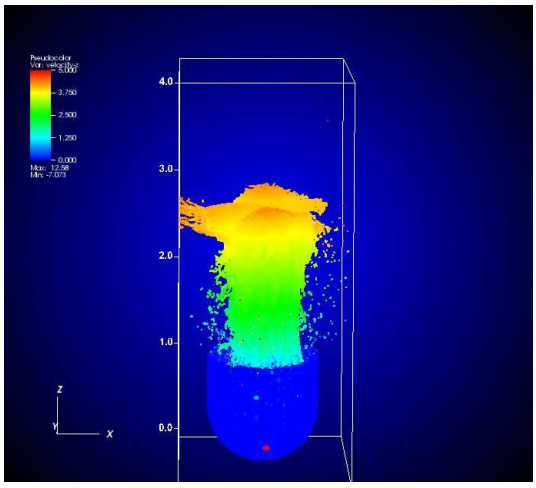


Figure 3.8: Mercury splash (thimble) at time 0.7 ms produced by Hsin-Chiang Chen

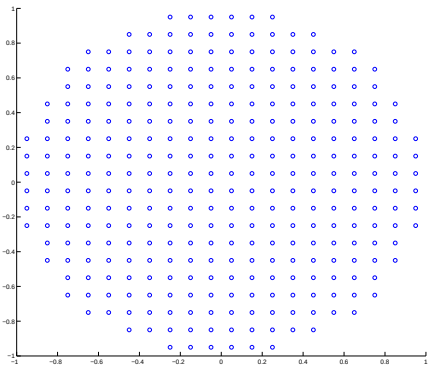


Figure 3.9: The testing particles for Smoothed Particle Hydrodynamics

of such approach, the accuracy. The error of approximation of derivatives by smoothed particle hydrodynamics is neither with small absolute value comparing with result obtained by finite difference, nor with acceptable convergence rate. In the discretized form, the summation(integral) of kernels no longer satisfied the previous conditions(3.19), (3.25) and (3.32) for consistency. A test for the approximation of gradient obtained by smoothed particle hydrodynamics has been carried out. The test function is chosen as $f(r) = exp(r^2)$ both in 2D and 3D($r = x^2 + y^2$ in 2D, $r = x^2 + y^2 + z^2$ in 3D). The test domain is $r < 1$ and all particles are placed with structured square packing(Figure 3.9). Because of the symmetry, L_2 norm of error in all directions will be the same, and only the result in x direction is shown. From the following tables we can see that the absolute value of errors are very large and there is only zero-th convergence rate.

This brings the result that Smoothed Particles Hydrodynamics is not a

particle distance	error of $\frac{\partial U}{\partial x}$ with SPH	error of $\frac{\partial U}{\partial x}$ with FD
0.1	1.0497	1.32×10^{-2}
0.05	1.0224	3.2×10^{-3}
0.025	1.0107	7.79×10^{-4}

Table 3.1: The mesh refinement test for SPH in 2D

particle distance	error of $\frac{\partial U}{\partial x}$ with SPH	error of $\frac{\partial U}{\partial x}$ with FD
0.1	1.1995	1.35×10^{-2}
0.05	1.1044	3.3×10^{-3}
0.025	1.0401	7.98×10^{-4}

Table 3.2: The mesh refinement test for SPH in 3D

good option for study phenomenon govern by linear waves. Meanwhile, for phenomenon govern by nonlinear waves, Smoothed Particle Hydrodynamics may perform better[41].

Chapter 4

Lagrangian Particle Method with Local Polynomial Fitting

In this chapter, an Lagrangian Particle Method with local Polynomial Fitting will be discussed. The motivation of developing such method is that poor derivatives accuracy strongly limits the applications of SPH. This approach utilizes local polynomial fitting to obtain estimators for derivatives, and solves the underlying Lagrangian equation for fluid. Here we keep only one idea of SPH, replacing material parcel to be particle. Some novel features including

- The method is free of artificial parameter: smoothing length.
- The change of density is evolved using Jacobian of particle position
- Local polynomial fitting is employed to perform derivative approximation

4.1 Lagrangian Equations

The Lagrangian equations with artificial viscosity for flow are

$$\frac{\partial \mathbf{R}}{\partial t} = \mathbf{u} \tag{4.1}$$

$$\frac{\partial \mathbf{u}}{\partial t} = V_0(\frac{\mathbf{R}(\mathbf{r},t)}{\mathbf{r}})^{\alpha-1}\nabla_{\mathbf{r}}(p+q) \tag{4.2}$$

$$\frac{\partial E}{\partial t} = -(p+q)\frac{\partial V}{\partial t} \tag{4.3}$$

where $V_0 = \frac{1}{\rho_0}$, and two auxiliary equations

$$V = V_0(\frac{\mathbf{R}(\mathbf{r},t)}{\mathbf{r}})^{\alpha-1}\frac{\partial \mathbf{R}}{\partial \mathbf{r}} \tag{4.4}$$

$$E = EOS(p,V) \tag{4.5}$$

A discretization of such equations in 3D(1D formula can be found in[44], 3D induction provided by Wei Li) is here. Let $\mathbf{R} = (x,y,z)$ and $\mathbf{u} = (u,v,w)$, we

have

$$x^{n+1} = x^n + \Delta t u^{n+1} \tag{4.6}$$

$$y^{n+1} = y^n + \Delta t v^{n+1} \tag{4.7}$$

$$z^{n+1} = z^n + \Delta t w^{n+1} \tag{4.8}$$

$$u^{n+1} = u^n - \Delta t V^n (\frac{\partial p}{\partial x} + \frac{\partial q}{\partial x})^n \tag{4.9}$$

$$v^{n+1} = v^n - \Delta t V^n (\frac{\partial p}{\partial y} + \frac{\partial q}{\partial y})^n \tag{4.10}$$

$$w^{n+1} = w^n - \Delta t V^n (\frac{\partial p}{\partial z} + \frac{\partial q}{\partial z})^n \tag{4.11}$$

$$E^{n+1} = E^n - (\frac{p^{n+1} + p^n}{2} + q^{n+1})(V^{n+1} - V^n) \tag{4.12}$$

$$V^{n+1} = V^n \mathbf{J}(\frac{\partial \mathbf{R^n + 1}}{\partial \mathbf{R^n}}) \tag{4.13}$$

$$p^{n+1} = eos(E^{n+1}, V^{n+1}) \tag{4.14}$$

The artificial viscosity term is

$$q^n = \begin{cases} \frac{2(a\Delta x)^2}{V^{n-1}(V^n + V^{n-1})}(\frac{V^n - V^{n-1}}{\Delta t})^2 & V^n - V^{n-1} < 0 \\ 0 & otherwise \end{cases} \tag{4.15}$$

The Jacobian is

$$\mathbf{J}(\frac{\partial \mathbf{R^{n+1}}}{\partial \mathbf{R^n}}) == \begin{vmatrix} \frac{\partial x^{n+1}}{\partial x^n} & \frac{\partial x^{n+1}}{\partial y^n} & \frac{\partial x^{n+1}}{\partial z^n} \\ \frac{\partial y^{n+1}}{\partial x^n} & \frac{\partial y^{n+1}}{\partial y^n} & \frac{\partial y^{n+1}}{\partial z^n} \\ \frac{\partial z^{n+1}}{\partial x^n} & \frac{\partial z^{n+1}}{\partial y^n} & \frac{\partial z^{n+1}}{\partial z^n} \end{vmatrix} \tag{4.16}$$

4.2 Approximation for derivatives and discretization of governing equation

4.2.1 Local Polynomial Fitting

To discretize the partial differential equations, local polynomial fitting will be employed. The local polynomial fitting has long been used to obtain the estimators for original function and corresponding derivatives[12, 22, 7], and the accurracy of it has also been discussed. Normally, with $n-th$ order polynomial, for $\nu-th$ order derivative, an estimator with $(n-\nu+1)-th$ to $(n-\nu+2)-th$ order of accuracy can be obtained. For visually simplicity, a 2D example is discussed here. As shown in Figure **??**, in the vicinity of point 0, the function value in point i can be expressed by

$$U_i = U_0 + h_i \left.\frac{\partial U}{\partial x}\right|_0 + k_i \left.\frac{\partial U}{\partial y}\right|_0 + \frac{1}{2}(h_i^2 \left.\frac{\partial^2 U}{\partial x^2}\right|_0 + k_i^2 \left.\frac{\partial^2 U}{\partial y^2}\right|_0 + 2h_i k_i \left.\frac{\partial^2 U}{\partial x \partial y}\right|_0) + \dots \tag{4.17}$$

where, U_i and U_0 are the corresponding function values in points i and 0. A polynomial can be used to approximate the original function and let us employ second order polynomial for example.

$$\tilde{U} = U_0 + h_i\theta_1 + k_i\theta_2 + \frac{1}{2}h_i^2\theta_3 + \frac{1}{2}k_i^2\theta_4 + h_i k_i \theta_5 \tag{4.18}$$

And the variables θ_1, θ_2, θ_3, θ_4 and θ_5 can be the estimators for $\frac{\partial U}{\partial x}$, $\frac{\partial U}{\partial y}$, $\frac{\partial^2 U}{\partial x^2}$, $\frac{\partial^2 U}{\partial y^2} and \frac{\partial^2 U}{\partial x \partial y}$ respectively. In order to obtain these variables, we can do local polynomial fitting with some points in the vicinity of center point 0. Suppose there are n

points in the neighbourhood of point 0 and they are called neighbours of point 0. A linear system can be built respectively

$$\begin{bmatrix} h_1 & k_1 & \frac{1}{2}h_1^2 & \frac{1}{2}k_1^2 & h_1k_1 \\ h_2 & k_2 & \frac{1}{2}h_2^2 & \frac{1}{2}k_2^2 & h_2k_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_n & k_n & \frac{1}{2}h_n^2 & \frac{1}{2}k_n^2 & h_nk_n \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \begin{bmatrix} U_1 - U_0 \\ U_2 - U_0 \\ \vdots \\ U_n - U_0 \end{bmatrix}$$

And, by solving this linear system, the variables can be obtained. In order to guarantee that a meaningful result to be acquired, there are some technical details about neighbour searching and will be discussed later.

To solve the linear system, the QR decomposition with pivoting is employed. An optimum solution is a solution x that minimize the L_2 norm of residual

$$min\|Ax - b\|_2 \tag{4.19}$$

and QR decomposition with column pivoting is employed to obtain x. Suppose $m \geq n$(m is the number of rows and n is the number of columns),

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix} P^T, m \geq n \tag{4.20}$$

where Q is an orthonomal matrix, R is an upper triangle matrix and P is a permutation matrix, chosen(in general) so that

$$|r_{11}| \geq |r_{22}| \geq \cdots \geq |r_{nn}| \tag{4.21}$$

and moreover, for each k,

$$|r_{kk}| \geq \|R_{k:j,j}\|_2 for j = k+1, \cdots, n. \tag{4.22}$$

In numerical computation, by determine an index k, such that the leading submatrix R_{11} in the first k rows and columns is well conditioned and R_{22} is negligible:

$$R = \begin{bmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{bmatrix} \simeq \begin{bmatrix} R_{11} & R_{12} \\ 0 & 0 \end{bmatrix} \tag{4.23}$$

Then k is the effective rank of A. Discussion about the numerical rank determination can be found in Golub and Van Loan[18]. A simple way to do numerical rank determination is to set a tolerance ϵ such that

$$\epsilon R_{11} \geq R_{kk} \tag{4.24}$$

and an appropriate choice of ϵ could be 10^{-8}. But, this approach is not stable. The solution for linear system (4.19) can be obtained as

$$x = P \begin{bmatrix} R_{11}^{-1} c_1 \\ 0 \end{bmatrix} \tag{4.25}$$

where c_1 is the first k elements of $c = Q^T b$. This can also be written as

$$x = A^+ b \tag{4.26}$$

particle distance	error of $\frac{\partial U}{\partial x}$ with LPF	error of $\frac{\partial U}{\partial x}$ with FD
0.1	2.08×10^{-2}	1.32×10^{-2}
0.05	5.4×10^{-3}	3.3×10^{-3}
0.025	1.4×10^{-3}	7.79×10^{-4}

Table 4.1: The mesh refinement test for local polynomial in 2D, LPF stands for local polynomial fitting and FD stands for finite difference

particle distance	error of $\frac{\partial U}{\partial x}$ with LPF	error of $\frac{\partial U}{\partial x}$ with FD
0.1	2.64×10^{-2}	1.35×10^{-2}
0.05	7.1×10^{-3}	3.3×10^{-3}
0.025	1.8×10^{-3}	7.98×10^{-4}

Table 4.2: The mesh refinement test for local polynomial in 3D, LPF stands for local polynomial fitting and FD stands for finite difference

where

$$A^{+} = P \begin{bmatrix} R_{11}^{-1} & 0 \\ 0 & 0 \end{bmatrix} Q^{T} \tag{4.27}$$

is the pseudoinverse of matrix A.

A test for first derivative in structured grid is carried out. The test function is $f = e^{x^2+y^2}$ in a unit circle. The result of second order finite difference is used to be the control. Test results are listed on tables 4.2.1 and 4.2.1. We can see that local polynomial fitting approximation has good convergence rate and the absolute value of error is comparable to the result of second order finite difference.

4.2.2 Neighbour searching and boundary particle finding

As mentioned in the previous section, the neighbour searching algorithm may greatly affect the result. With too small number of neighbours, the accuracy of estimators may be very low. With too large number of neighbours, the accuracy may also be very low and the computational cost will be very high. Meanwhile, the distribution of neighbours also affects the result. If all neighbours gather in a small area near the center point, we can expect relative bad outcome to be obtained. We suggest that number of neighbours should be greater of equal to twice of number of variables. For example, with second order polynomial, 10 neighbours will be a good choice, while slightly smaller number may still be doable. Also, the number of neighbours can be different for each center particle. The distribution of particles is a even difficult issue to be control cause the distribution of points are already fixed. We suggest that the distances of neighbours to the center point should be differs less than 2 times and the neighbours should be relatively equally distributed in the vicinity of center particle. A viable approach in 2D will be dividing the vicinity of center particle to 4 quadrants and selecting 2 to 3 neighbours in each quadrant, and in 3D, it will be dividing the vicinity to be 8 octants and selecting neighbours respectively.

For elliptic problem, the imposing of boundary condition is very important and in order to impose the boundary condition, boundary particles need to be found. With sufficient resolution of boundary, we can expect that the

number of points in a circle with fix radius to the center point will be smaller in the boundary point than interior point. By setting a threshold and checking all the points, all the boundary points can be located. For Dirichlet boundary condition, knowing the position of boundary points is enough. For Neumann boundary condition, the normal direction in the boundary point has to be calculated. In 2D, the boundary is a curve. By selecting adjacent boundary points of a given boundary points, a polynomial can be obtained by fitting. After obtaining the polynomial, the normal direction of can be calculated analytically. In 3D, same approach can be applied to surface.

4.2.3 Elliptic Boundary problem and discretization

The governing equation of elliptic boundary problem is

$$\nabla\beta\nabla\Phi = f \tag{4.28}$$

with Neumann boundary condition

$$\left.\frac{\partial\Phi}{\partial\mathbf{n}}\right|_{\Gamma} = \mathbf{g} \tag{4.29}$$

or Dirichlet boundary condition

$$\Phi|_{\Gamma} = h \tag{4.30}$$

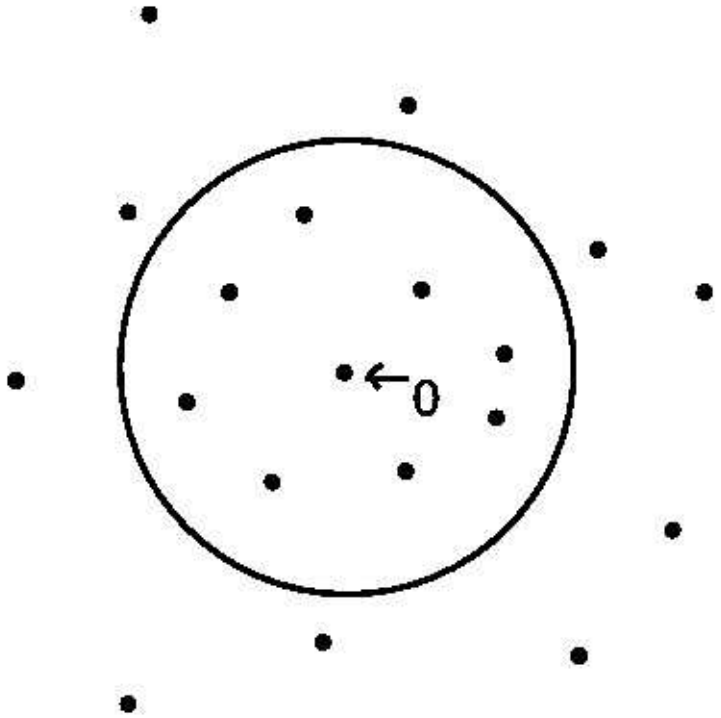


Figure 4.1: The stencil for boundary particle

Neumann Boundary Condition

The governing equation of elliptic Neumann boundary problem is

$$\nabla \cdot (k \nabla \varphi) = f \tag{4.31}$$

with boundary condition

$$\left. \frac{\partial U}{\partial n} \right|_{\partial \Omega} = g \tag{4.32}$$

Let's consider the discretization of governing equation for interior particles first. As shown in Figure 4.1, particle 0 is the particle of interest and there are n neighbour particles in the interaction domain. A linear system $Ax = b$

can be obtained. As described before, the solution x can be solved as $A^{+}b$. Donoting

$$A^{+} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ \cdots & \cdots & \ddots & \cdots \\ d_{51} & d_{52} & \cdots & d_{5n} \end{bmatrix} \tag{4.33}$$

the estimator for derivatives can be written as

$$\frac{\partial U}{\partial x} = \sum_{i=1}^{n} d_{1i}U_i - c_1 U_0 \tag{4.34}$$

$$\frac{\partial U}{\partial y} = \sum_{i=1}^{n} d_{2i}U_i - c_2 U_0 \tag{4.35}$$

$$\nabla^2 U = \sum_{i=1}^{n} (d_{3i} + d_{4i})U_i - (c_3 + c_4)U_0 = \frac{1}{k} f \tag{4.36}$$

where

$$c_i = \sum_{j=1}^{n} d_{ij} \tag{4.37}$$

For boundary particles(Figure 4.2), both the governing equation and boundary condition need to be discretized. In order to couple both the boundary condition and governing equation, a ghost particle is introduced. Also, as mentioned in the previous chapter, the approximation of the normal direction is obtained by the summation of all directed edges from surrounding particles to the central particle, $\mathbf{n} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{e_j}$. To scale the length of the normal vector, it is divided by n. The ghost particle is placed in $(x_0, y_0) + \mathbf{n}$, where (x_0, y_0) is the position of particle 0. Together with ghost particle, there are $m = n + 1$ neighbour particles, while particle m is the ghost particle. Let $\mathbf{n} = n_x\mathbf{e_x} + n_y\mathbf{e_y}$, while n_x and n_y are the x and y components of $\mathbf{n}$ correspondingly, $\mathbf{e_x}$ and $\mathbf{e_y}$

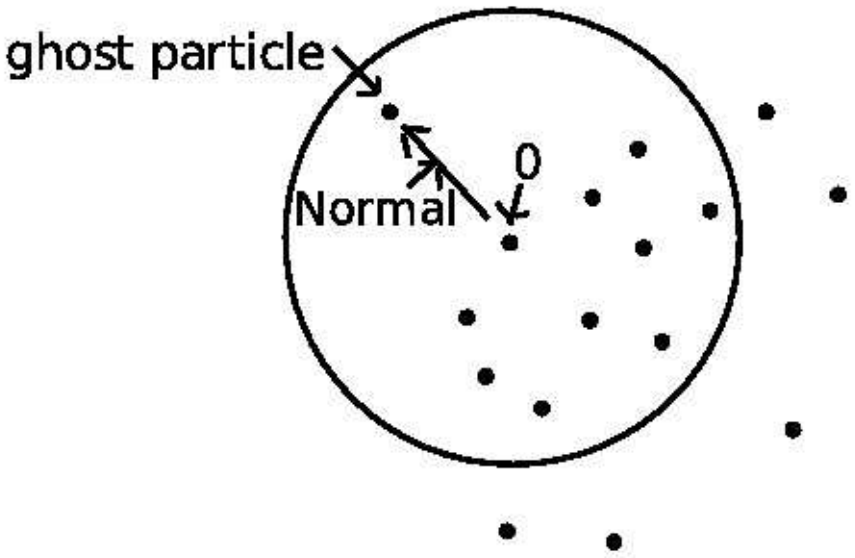


Figure 4.2: The stencil for boundary particle

are the base vector in x and y directions correspondingly. n_x and n_y can be further expressed by h_j and k_j

$$\begin{aligned} n_x &= -\frac{1}{n}\sum_{j=1}^{n} h_j \\ n_y &= -\frac{1}{n}\sum_{j=1}^{n} k_j \end{aligned}$$

And, obviously, $h_m = n_x$, $k_m = n_y$. So, a similar linear system $Ax = b$ can be constructed and by discretizing the governing equation, and we obtain

$$\nabla^2 U = \sum_{j=1}^{m}(c_{3j} + d_{4j})U_j - (c_3 + c_4)U_0 = \frac{1}{k}f \tag{4.38}$$

The boundary condition can be written as

$$\frac{\partial U}{\partial x}n_x + \frac{\partial U}{\partial y}n_y = g \tag{4.39}$$

and we obtain

$$\sum_{j=1}^{m}(d_{1j}n_x + d_{2j}n_y)U_j - (c_1n_x + c_2n_y)U_0 = g \tag{4.40}$$

thus

$$U_m = \frac{g + (c_3n_x + c_4n_y)U_0 - \sum_{j=1}^{l}(d_{1j}n_x + d_{2j}n_y)U_j}{d_{1m}n_x + d_{2m}n_y} \tag{4.41}$$

Substitute equation (4.41) into equation (4.38)

$$\begin{aligned} &\sum_{j=1}^{l}\left[d_{3j} + d_{4j} - \frac{d_{3m} + d_{4m}}{d_{1m}n_x + d_{2m}n_y}(d_{1j}n_x) + d_{2j}n_y\right]U_j \\ &- \left[c_3 + c_4 - \frac{d_{3m} + d_{4m}}{d_{1m}n_x + d_{2m}n_y}(c_1n_x + c_2n_y)\right]U_0 \\ = \;& \frac{1}{k}f - \frac{d_{3m} + d_{4m}}{d_{1m}n_x + d_{2m}n_y}g \end{aligned} \tag{4.42}$$

With equation (4.34) and equation (4.42), the governing equation on all particles can be discretized. By solving the linear system consisting of all the discretization, the values of φ on each particle can be obtained. Since the boundary condition is Neumann boundary condition, the solution can be differ up to a constant. If we are interested in the gradient of φ, it can be calculated in a similar way of discretizing the governing equation. For interior

particles,

$$\nabla\varphi = \sum_{j=1}^{n}(d_{1j}\mathbf{e_x} + d_{2j}\mathbf{e_y})U_j - (c_1\mathbf{e_x} + c_2\mathbf{e_y})U_0 \tag{4.43}$$

for boundary particles, the value of ghost particle should be calculated from the discretization of governing equation first

$$U_m = \frac{\frac{1}{k}f + (c_3 + c_4)U_0 - \sum_{j=1}^{n}(d_{3j} + d_{4j})U_j}{d_{3m} + d_{4m}} \tag{4.44}$$

then the gradient can be obtained by,

$$\begin{aligned}
\nabla\varphi \;=\; & \sum_{j=1}^{n}((d_{1j} - \frac{d_{1m}(d_{3j} + d_{4j})U_j)}{d_{3m} + d_{4m}})\mathbf{e_x} \\
& +(d_{2j} - \frac{d_{2m}(d_{3j} + d_{4j})U_j)}{d_{3m} + d_{4m}})\mathbf{e_y})U_j \\
& -((c_1 - \frac{d_{1m}(c_3 + c_4)}{d_{3m} + d_{4m}})\mathbf{e_x} \\
& +(c_2 - \frac{d_{2m}(c_3 + c_4)}{d_{3m} + d_{4m}})\mathbf{e_y})U_0 \\
& +\frac{f}{k(d_{3m} + d_{4m})}(d_{1m}\mathbf{e_x} + d_{2m}\mathbf{e_y})
\end{aligned} \tag{4.45}$$

Dirichlet Boundary Condition

The discretization of Dirichlet boundary condition is relatively simpler. All the function values of boundary particle are given so that linear system will be built only on interior particles. When a boundary particle is included in the neighbours of an interior particle, supposing the boundary particle is

particle 1, equation

$$\sum_{i=1}^{9}(d_{3i}+d_{4i})U_i-(c_3+c_4)U_0=\frac{1}{k}f \tag{4.46}$$

can be rewrite as

$$\sum_{i=2}^{9}(d_{3i}+d_{4i})U_i-(C_3+c_4)U_0=\frac{1}{k}f-(d_{31}+d41)U_1 \tag{4.47}$$

4.2.4 Solving of the global system

After building the unknown matrix and corresponding right-hand-side, a global linear system is obtained. For Dirichlet boundary condition, this matrix is with full rank, while for Neumann Boundary condition, the matrix is with rank deficiency one. LSQR method can be adopted to solve both these two cases, and theoretically, LSMR method will be with better performance.

4.2.5 Verification

In this part, 2D and 3D verification for both Neumann and Dirichlet boundary condition will be discussed.

The distribution of particle is constructed in the following way in 2D. First, particles were evenly placed in a domain $\Omega=[-1:1]\times[-1:1]$ with certain interval in x direction Δx and y direction Δy and $\Delta x=\Delta y$. Particles in circle $x^2+y^2\leq 1$ were selected. Then, random perturbation was applied to each particle. The perturbation vector is $\mathbf{p}=(p_x,p_y)$ where $|\mathbf{p}|=0.2\Delta x$. One distribution of particles is shown in Fig (4.3). For Neu-

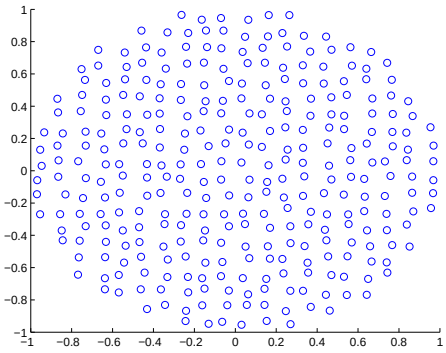


Figure 4.3: The distribution of particles

Mesh Size	error for Neumann Boundary condition	rate	error for Dirichlet Boundary condition	rate
0.1	4.30×10^{-2}		9.66×10^{-3}	
0.05	1.23×10^{-2}	1.81	2.70×10^{-3}	1.83
0.025	3.16×10^{-3}	1.96	7.22×10^{-4}	1.9

Table 4.3: The mesh refinement test in 2D

mann boundary condition, the normal vector of boundary particle is obtained as described in section (4.2.3,**??**). Simulation was run for 100 times for Neumann boundary condition and 50 times for Dirichlet boundary condition. For Neumann boundary condition, the error is a vector and the relative error is $\frac{|\mathbf{numerical\_gradient} - \mathbf{exact\_gradient}|}{|\mathbf{exact\_gradient}|}$. For Dirichlet boundary condition, relative error is $\frac{|numerical\_solution - exact\_exact\_solution|}{|exact\_solution|}$. Neumann Boundary condition results are shown in the Figures 4.4,4.5,4.6. The horizontal line in each figure is the average error of 40 different random events.

The table(4.2.5) shows the summary of such results In 3D, the particles inside the ball $x^2 + y^2 + z^2 \leq 1$ were selected and similar perturbation as 2D

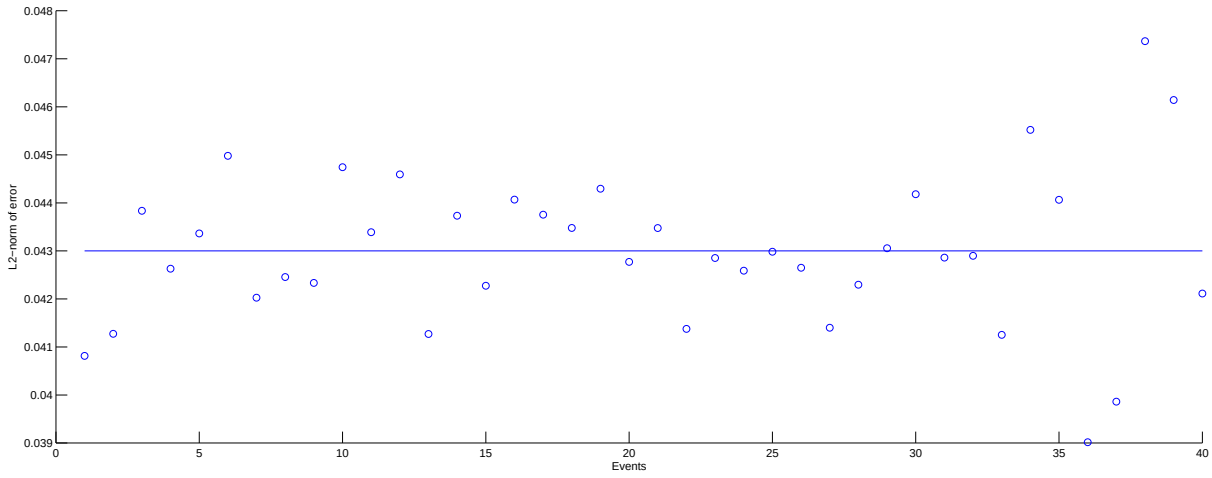


Figure 4.4: The RMS relative error of 2D Neumann Boundary Condition test with 0.1 average shortest particle distance

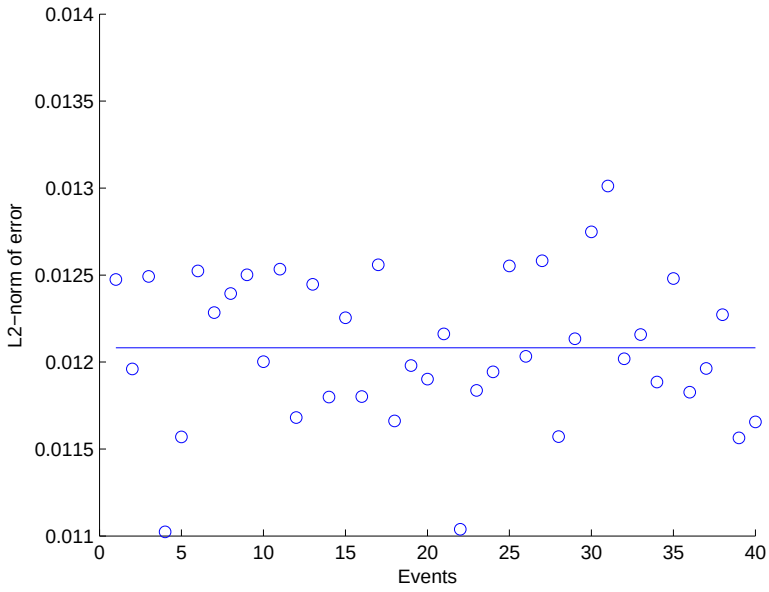


Figure 4.5: The RMS relative error of 2D Neumann Boundary Condition test with 0.05 average shortest particle distance

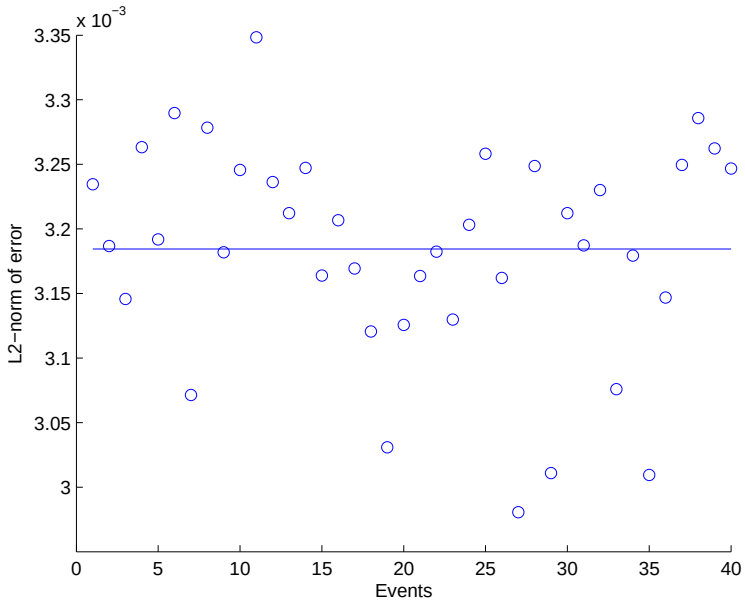


Figure 4.6: The RMS relative error of 2D Neumann Boundary Condition test with 0.025 average shortest particle distance

was applied:$\mathbf{p} = (p_x, p_y, p_z)$ where $|\mathbf{p}| = 0.2\Delta x$. Figures 4.7, 4.8, 4.8 show the results and the horizontal line in each figure is the average error of 10 events.

The table (4.2.5) shows the summary of the results we can see approximate second order convergence rate from the above tables.

Verification on complex interfaces(Figures 4.10, 4.11) has also been carried out with Neumann Boundary Condition and same testing function. 2D has been done with interface formula

$$r = 1 + 0.2sin(5\theta)$$

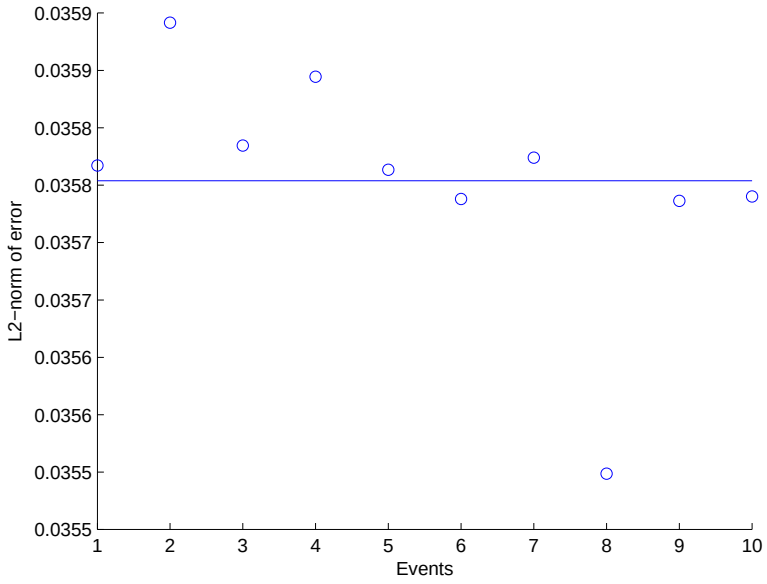


Figure 4.7: The RMS relative error of 3D Neumann Boundary Condition test with 0.1 average shortest particle distance

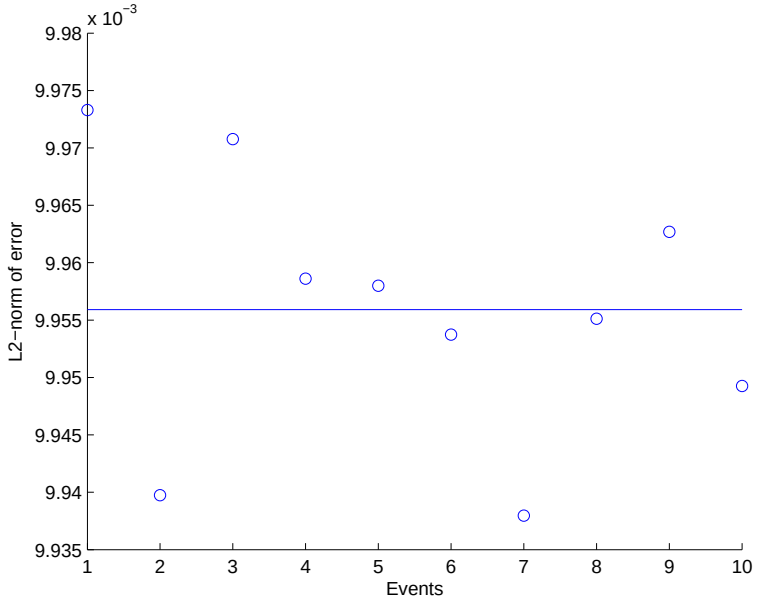


Figure 4.8: The RMS relative error of 3D Neumann Boundary Condition test with 0.05 average shortest particle distance

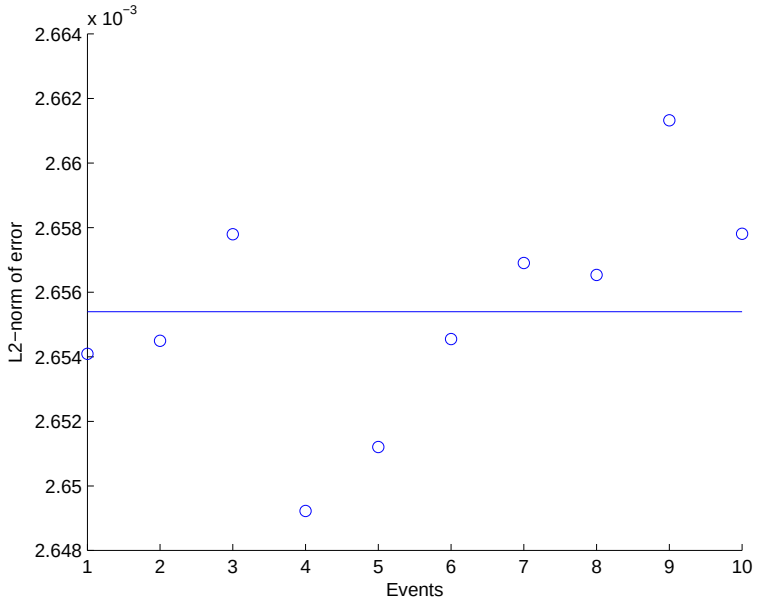


Figure 4.9: The RMS relative error of 3D Neumann Boundary Condition test with 0.025 average shortest particle distance

Mesh Size	error for Neumann Boundary condition	rate	error for Dirichlet Boundary condition	rate
0.1	3.58×10^{-2}		1.05×10^{-2}	
0.05	9.96×10^{-3}	1.85	2.96×10^{-3}	1.83
0.025	2.66×10^{-3}	1.9	7.9×10^{-4}	1.91

Table 4.4: The mesh refinement test in 3D

Mesh Size	2D Test	convergence rate	3D Test	convergence rate
0.1^3	0.1064		0.1773	
0.05^3	0.0318	1.74	0.0517	1.78
0.025^3	0.0085	1.9	0.0142	1.86

Table 4.5: The mesh refinement test for complex domain

so that, in 2D Cartesian coordinate system, the formula would be

$$\begin{aligned} x &= (1 + 0.2sin(5\theta))cos(\theta) \\ y &= (1 + 0.2sin(5\theta))sin(\theta) \end{aligned}$$

3D has been done with interface formula

$$r = 1 + 0.03e^{-|tan(\varphi)|}sin(5\theta)$$

and the formula in Cartesian coordinate system is

$$\begin{aligned} x &= (1 + 0.03e^{-|tan(\varphi)|}sin(5\theta))cos(\varphi)cos(\theta) \\ y &= (1 + 0.03e^{-|tan(\varphi)|}sin(5\theta))cos(\varphi)sin(\theta) \\ z &= (1 + 0.03e^{-|tan(\varphi)|}sin(5\theta))sin(\varphi) \end{aligned}$$

Boundary particles are put exactly on the surface(curve in 2D) and the normal direction is analytically given. In 3D test, Spherical Centroidal Voronoi Tesselation(SCVT)[10, 24, 43] is used to initialize the positions of boundary particles.

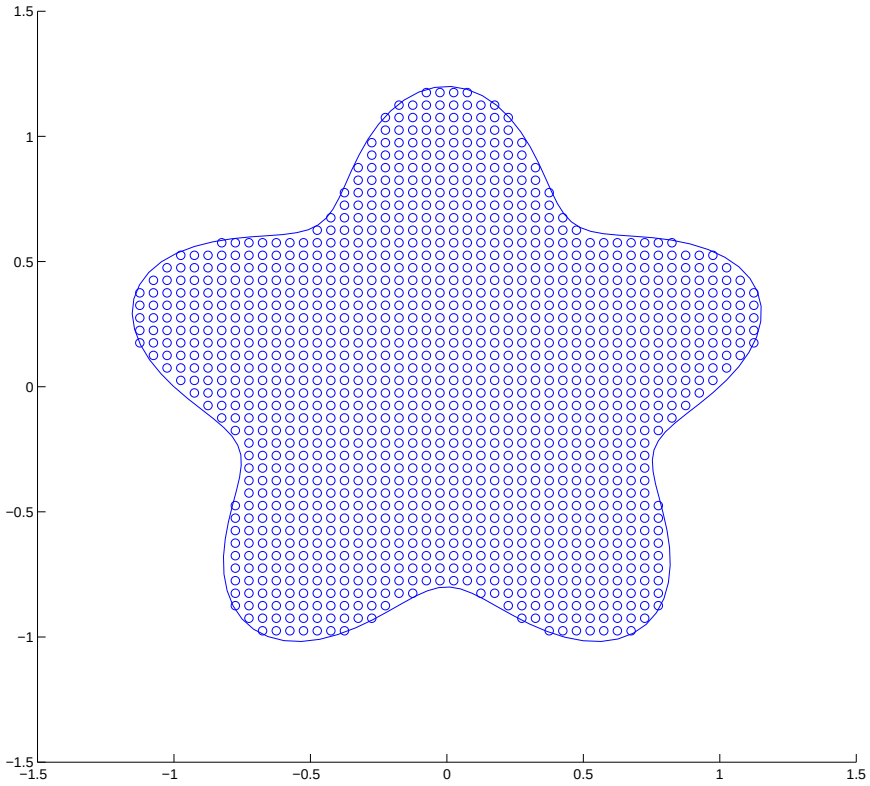


Figure 4.10: Testing domain for 2D

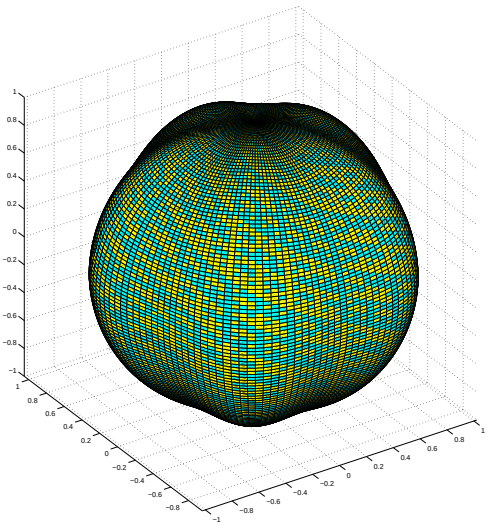


Figure 4.11: Testing domain for 3D

The electric potential problem mentioned mentioned in chapter **??** is used to validate this approach.

$$\nabla^2\varphi = \nabla \cdot (\frac{1}{c}\mathbf{u} \times \mathbf{b}) \tag{4.48}$$

$$\left.\frac{\partial\varphi}{\partial n}\right|_{\Omega} = \frac{1}{c}(\mathbf{u} \times \mathbf{b}) \cdot \mathbf{n} \tag{4.49}$$

Figure 4.12 shows the results obtained. The absolute values of current density and the direction of the current density are close to that obtained by mesh technique.

4.3 Development of Parallel Lagrangian Particle Code

A parallel Lagrangian particle code employing local polynomial fitting method is implemented. The code consists of the following classes

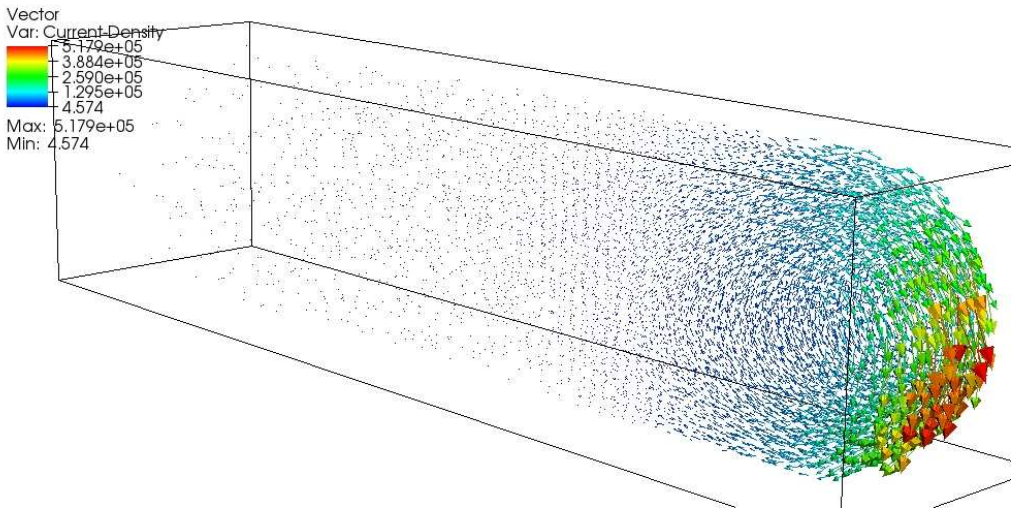


Figure 4.12: The current density obtained by local polynomial fitting method

1. Controller module
2. Particle management and physics module
   (a) Visualization module
   (b) Equation of state model
3. parallel communication module
4. Neighbour searching module
5. Local polynomial fitting coefficient calculation module
6. Time integral module
7. Spatial discretization module
8. Elliptic solver

9. Matrix solver

4.3.1 Controller Module

The controller is designed for controlling the work flow of the code. Controller reads initial settings from input file first. After that, the initialization module set initial states for all the particles. For pure elliptic problem, the elliptic solver will then be called, solve the corresponding problem store the result. For hyperbolic problem, the selection of temporal and spatial discretization schemes is read from the input file and a loop is create to carry out the time integral. For elliptic problem, there is only one set of output information, while for hyperbolic problem, customized output time interval can be set from the input file.

4.3.2 Particle Management and Physics Module

This module is designed for the management of Lagrangian particles and applying corresponding physics for each particle. A particle has the following states

1. velocity
2. density
3. specific energy(for compressible assumption)
4. pressure
5. position

Also, for each particle, the neighbour list is built in the beginning of each step, and the corresponding coefficients of all neighbours for the approximation of derivatives are also calculated. Equation of state module is for particle with compressible assumption. Currently, polytropic and stiffed-polytropic equation of state model are implemented. The visualization of interior states is critical in research. Both scalar field and vector field of interior states are printed into VTK files. To do mesh based visualization of states, a visualization mesh is built and states are deposit into the mesh.

4.3.3 Parallel Communication Module

The parallelization of local polynomial fitting Lagrangian code is similar to that of smoothed particle hydrodynamics code. It includes two main steps, the moving of particle between subdomains and the synchronization of buffer zone. For weakly compressible code, the distance between particles is varying in a relatively small range. The size of the buffer zone is determined by the maximum distance between central particle and neighbours to ensure that all interior particles have sufficient neighbour information. In the first step of communication, information of particles moving into the buffer zones is transfered to the corresponding adjacent subdomains. Next, information of particles in the area corresponding to the buffer zone of adjacent subdomains is transfered.

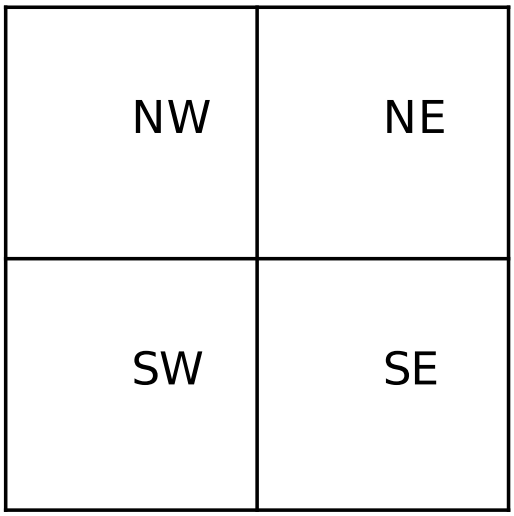


Figure 4.13: The partition of quadtree

4.3.4 Neighbour Searching Module[8]

Octree searching algorithm is employed to perform neighbour searching. For visual simplicity, the corresponding algorithm in 2D, quadtree searching, is discussed here. M De Berg, O Cheong, M van Kreveld, M Overmars offer a nice description of the quatree searching algorithm in their book[8]. A quadtree is a rooted tree that every subtreenode has four children. Every node in the quadtree is corresponding to a square in the searching domain. If a node n has children, then the corresponding squares of such children are the four quadrants of the square of n. The four quadrants of the subtree nodes together form the square of the parent(Figure 4.13). The children can be labeled as NE, NW, SW, and SE indicating the corresponding quadrants. Here is a definition[8] of quadtree for a set P of points inside a square Ω. Let $\Omega = [x_\Omega : x'_\Omega] \times [y_\Omega : y'_\Omega]$.

- If size(P) ≤ 1, then the quadtree consists of a single leaf where the set P and the square Ω is stored.

- Otherewise, let Ω_{NE}, Ω_{NW}, Ω_{Sw} and Ω_{SE} denote the four quadrants of Ω. Let $x_{mid} = \frac{x_\Omega + x'_\Omega}{2}$ and $y_{mid} = \frac{y_\Omega + y'_\Omega}{2}$.

Define

$$\begin{aligned} P_{NE} &= p \in P : p_x > x_{mid} \; and \; p_y > y_{mid} \\ P_{NW} &= p \in P : p_x \leq x_{mid} \; and \; p_y > y_{mid} \\ P_{SW} &= p \in P : p_x \leq x_{mid} \; and \; p_y \leq y_{mid} \\ P_{SE} &= p \in P : p_x > x_{mid} \; and \; p_y \leq y_{mid} \end{aligned}$$

The quadtree now consists of root node n where the square Ω is stored, and the square stored at n is denoted by $\Omega(n)$.

In practice, it is not necessary for each node to store its corresponding square as along as traversing the tree, the squares can be calculated.

The quadtree can be built recursively. Firstly, split the current square into four quadrants, and split the point set accordingly. Then, construct quadtree for each quadrants with its associated point set. The stopping criterion for the recursion is that the size of point set is smaller than two[8].

A basic operation on quadtrees is neighbour finding: given a node n and a direction, find a node n' such that $\Omega(n')$ is adjacent to $\Omega(n)$ in the given direction. This corresponds to finding an adjacent square of a given square. Let's take the finding of south neighbour as example. The pseudocode is as follows.

SOUTHNEIGHBOUR(n,T) input: a node in a quadtree T. output: The deepest node n' whose depth is at most the depth of n such that $\Omega(n')$ is a south-neighbour of $\Omega(n)$, and NULL if there is no such node.

1. if n is root(T) then return NULL
2. if n is the NW-child of parent(n) then return SW-child of parent(n)
3. if n is the NE-child of parent(n) then return SE-child of parent(n)
4. let μ be SOUTHNEIGHBOUR(parent(n),T)
5. if μ is NULL or μ is a leaf then return μ
6. else if n is the SW-child of parent(n) then return NW-child of μ
7. else return NE-child of μ

In the implementation, several things are different. The minimum area of a square corresponding to a leaf is determined by the searching radius, thus more than one points are allowed to reside in the smallest square and the stopping criterion is the depth of the quadtree. To obtain all the particles with distance smaller than certain value to a certain particle, the searching will be in eight direction, that is east, north-east, north, north-west, west, south-west, south and south-east for adjacent square. By calculating the distance between all points in the same square as well as adjacent squares to the point of interest, a neighbour list can be obtained. There is a trade off between the depth of the tree and point number in each square. If the depth is smaller, the constructing

time of the tree will be smaller while the comparison time for each adjacent square will be larger. As mentioned in the previous chapter, normally, the max depth is set to be 6 in our application.

4.3.5 Derivatives Estimator coefficient calculation module

For a certain distribution of particles, the coefficients of approximation for both first and second derivatives of all particles can be obtained together by the solving of corresponding local linear system of least square problems. Hyperbolic solver may use both first and second derivatives and elliptic solver uses the second derivative. In order to reuse the approximation information, the coefficients are evaluated after the parallel communication(for parallel run) and neighbour list building for each particle. The coefficients of all first and second partial derivatives are stored along side the neighbour list, meaning that for each particle there is a set of coefficients associating with it. In the current code, LAPACK is utilized to obtain the pseudoinverse.

4.3.6 Time Integral and Hyperbolic solver module

In the current code, forward Euler and second order Runge-Kutta schemes are implemented.

For partial differential equation

$$\frac{\partial y}{\partial t} = f(t, y) \tag{4.50}$$

The forward Euler scheme is

$$y_{n+1} = y_n + hf(t, y) \tag{4.51}$$

The spatial derivatives in f is approximated by the local polynomial fitting. y_n is the value of an interior state of a particle in the current time step, y_{n+1} is the value of the next time step and h is the time stepping size. With forward Euler scheme, the solution is with first order in time and the time stepping size is constrained by CFL condition. The second order Runge-Kutta scheme is also called predictor-corrector scheme in some resource. It is

$$y_{n+1} = y_n + hf(t_n + \frac{1}{2}h, y_n + \frac{1}{2}hf(t_n, y_n)) \tag{4.52}$$

This is a two stages scheme with midpoint method. The solution is with second order accuracy in time. The implementation of the second order Runge-Kutta scheme costs more memory space than forward Euler scheme, because that an extra particle management module instance is used to store the information of the intermediate step besides two instances storing the information for the current step and the next step.

4.3.7 Elliptic solver module

The elliptic solver can be called alone or together with the hyperbolic solver if there is elliptic component in the corresponding problem. The elliptic equation together with the boundary condition are discretized by local poly-

nomial fitting. A linear system for all the discretizing equations of all particles is constructed after that. To solve such linear system, PETSc[4, 3, 5] is employed. In the current code, a wrapper for the Krylov Subspace Methods are implemented. There are two linear solvers in the wrapper, which are BiCGSL solver and GMRES solver.

4.4 Applications

Lagrangian fluid solver can perform the same simulation discussed in chapter one while with relatively less computational cost since only one type of particle will be used. For phenomena discussed in chapter 2, by using local polynomial fitting, the accuracy of solution will be greatly improved while keep the basic physics facts. Currently, the hyperbolic solver is under development and both verification and validation for the code will be carried out soon.

Chapter 5

Conclusion

Mesh based as well as meshless algorithms have been developed and implemented in parallel which are capable of solving elliptic boundary and elliptic interface problem. Embedded interface method is implemented as a part of FronTier library to solve incompressible MHD problems. This code is capable of dealing with sharp interface with high jump of physics states. Simulation for MHD jet and stefan problems have been carried out and there is good agreement between numerical results and experiment results as well as theoretical results. The stefan problem benefits from the capability of preserving sharp interface. Meanwhile, for MHD jet simulation, the code allows more realistic conditions. Without embedded interface method, the sharp interface may need to be smeared into a band. We can see much better simulation results with the code employing embedded interface method.

A Smoothed Particle Hydrodynamics code is implemented and tested. The test results show that SPH method can only obtain very poor derivatives approximation. This fact limits the application of SPH method and it is

actually solving the Hamiltonian system of particles. For acoustic wave driven problems, the SPH method will not work. Also, SPH method highly depends on artificial and ad-hoc parameters. These drawbacks of SPH became the motivation of the developing of a new Lagrangian particle technique with local polynomial fitting. With local polynomial fitting, accurate approximation of derivatives can be obtained and there is no need any artificial parameters. We can see very good convergence rate for approximation of derivatives by using local polynomial fitting. The elliptic problems on complex boundary are also tested and the results are with good accuracy.

Bibliography

[1] Iter: International thermonuclear experimental reactor, http://www.iter.org/.

[2] V. Alexiades and A. D. Solomon. *Mathematical modeling of melting and freezing processes*. McGraw-Hill, 1993.

[3] Satish Balay, Jed Brown, , Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.4, Argonne National Laboratory, 2013.

[4] Satish Balay, Jed Brown, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2013. http://www.mcs.anl.gov/petsc.

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

[6] J. B. Bell, P. Colella, and H. M. Glaz. A second order projection method for the incompressible navier-stokes equations. *J. Comput. Phys.*, 1989.

[7] J.J. Benito, F. Urea, and L. Gavete. Influence of several factors in the generalized finite difference method. *Applied Mathematical Modelling*, 25(12):1039–1053, 2001.

[8] M De Berg, O Cheong, M van Kreveld, and M Overmars. *Computational geometry*. Springer Berlin Heidelberg, 2008.

[9] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, and L. Wu. A simple package for front tracking. *J. Comput. Phys.*, 213, 2006.

[10] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[11] J. Duderstadt and L. Hamilton. *Nuclear Reactor Analysis*. John Wiley & Sons, Inc., 2nd ed. edition, 1976.

[12] Jianqing Fan, Theo Gasser, Irene Gijbels, Michael Brockmann, and Joachim Engel. Local polynomial fitting: A standard for nonparametric regression, 1993.

[13] J. Freidberg. *Ideal Magnetohydrodynamics (Modern Perspectives in Energy),*. Springer, 1987.

[14] J. P. FREIDBERG. *Rev. Mod. PHys.*, 1982.

[15] B. Bunner G. Tryggvason and et. al. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169, 2001.

[16] J. Glimm, M.J. Graham, J. Grove, X.L. Li, T.M. Smith, D. Tan, F. Tangerman, and Q. Zhang. Front tracking in two and three dimensions. *Computers and Mathematics with Applications*, 35(7):1–11, 1998.

[17] J. Glimm, J. Grove, X. Li, K. L. Shyue, Q. Zhang, and Y. Zeng. Three dimensional front tracking. *SIAM J. Sci. Comput.*, 19, 1998.

[18] G. GOLUB and C. F. VAN LOAN. *Matrix Computations*. Johns Hopkins University Press, Baltimore, MD, 1996.

[19] Tongfei Guo, Shuqiang Wang, and Roman Samulyak. Sharp interface algorithm for large density ratio incompressible multiphase magnetohydrodynamic flows. *Procedia Computer Science*, 18, 2013.

[20] M. Gmez-Gesteira, A.J.C. Crespo, B.D. Rogers, R.A. Dalrymple, J.M. Dominguez, and A. Barreiro. Sphysics - development of a free-surface fluid solver- part 2: Efficiency and test cases. *Computers & Geosciences*, 48, 2012.

[21] M. Gmez-Gesteira and R.A. Dalrymple. Using a 3d sph method for wave impact on a tall structure. *J. Waterway, Port, Coastal, Ocean Engineering*, 130, 2004.

[22] X. Jiao and H. Zha. Consistent computation of first- and second-order differential quantities for surface meshes. *ACM Solid and Physical Modeling Symposium*, 2008.

[23] H. Johansen and P. Colella. A cartesian grid embedding boundary method for poisson's equation on irregular domains. *J. Comput. Phys.*, 147, 1998.

[24] Lili Ju, Qiang Du, and Max Gunzburger. Probabilistic methods for centroidal voronoi tessellations and their parallel implementations. *Parallel Computing*, 28:1477–1500, 2002.

[25] M. Kang, R. Fedkiw, and X.-D. Liu. A boundary condition capturing method for multiphase incompressible flow. *J. Comput. Phys.*, 15, 2000.

[26] H. Kim, L. Zhang, R. Samulyak, and P. Parks. On the structure of plasma liners for plasma jet induced magnetoinertial fusion. *Phys. Plasmas*, 2013.

[27] L. Lee and R. LeVeque. An immersed interface method for incompressible navier-stokes equations. *SIAM J. Sci. Comput.*, 25, 2003.

[28] Z. Li and K. Ito. The immersed interface method: Numerical solutions of pdes involving interfaces and irregular domains. *Frontiers Appl. Math. 33, SIAM, Piladelphia*, 2006.

[29] L.D. Libersky, A.G. Petschek, A.G. Carney, T.C. Hipp, J.R. Allahdadi, and F.A. High. Strain lagrangian hydrodynamics: a three-dimensional sph code for dynamic material response. *J. Comput. Phys.*, 109:67–75, 2010.

[30] H. Lim, J. Iwerks, Y. Yu, J. Glimm, and D. H. Sharp. Verification and validation of a method for the simulation of turbulent mixing. *Physica Scripta*, 2010.

[31] Gui-Rong Liu and M. B. Liu. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific, 5 Toh Tuck Link, Singapore 596224, Jan 1, 2003.

[32] X.-D. Liu, R. Fedkiw, and M. Kang. A boundary condition capturing method for poisson's equation on irregular domains. *J. Comput. Phys.*, 160, 2000.

[33] L.B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astron. J.*, 82, 1977.

[34] P. McCorquodale, P. Colella, and H. Johansen. A cartesian grid embedded boundary method for the heat equation on irregular domains. *J. Comput. Phys.*, 173, 2001.

[35] J J Monaghan. Smoothed particle hydrodynamics-theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181, 1977.

[36] J J Monaghan. On the problem of penetration in particle methods. *J. Comput. Phys.*, 82, 2005.

[37] J J Monaghan. Smoothed particle hydrodynamics. *Rep. Prog. Phys.*, 68, 2005.

[38] J J Monaghan. Smoothed particle hydrodynamics. *Annual review of astronomy and astrophysics*, 30, 2005.

[39] S. Oshima, R. Yamane, Y. Moshimaru, and T. Matsuoka. The shape of a liquid metal jet under a non-uniform magnetic field. *JCME Int. J.*, 30, 1987.

[40] B. Pegourie. *Plasma Phys. Control. Fusion*, 49, 2007.

[41] Daniel J. Price. Smoothed particle hydrodynamics and magnetohydrodynamics. *J. Comput. Phys.*, 231, 2012.

[42] C. B. Reed and S. Molokov. Review of free-surface mhd experiments and modeling. Technical Report Technical Report ANL/TD/TM99-08, Argonne National Laboratory, April 1999.

[43] Robert Renka. Algorithm 772: Stripack: Delaunay triangulation and voronoi diagram on the surface of a sphere. *ACM Transactions on Mathematical Software*, 23(3):416–434, 1997.

[44] Robert D. Richtmyer and K. W. Morton. *Difference methods for initial-value problems*. Interscience Publishers, 1967.

[45] R. Samulyak, W. Bo, X. Li, H. Kirk, and K. McDonald. Computational algorithms for multiphase magnetohydrodynamics and applications. *Condensed Matter Physics*, 2010.

[46] R. Samulyak, W. Bo, Xiaolin Li, H. Kirk, and K. McDonald. Computational algorithms for multiphase magnetohydrodynamics and applications. *Condensed Matter Physics*, 13(4):43402: 1–12, 2010.

[47] R. Samulyak, J. Du, J. Glimm, and Z. Xu. A numerical algorithm for MHD of free surface flows at low magnetic reynolds numbers. *J. Comput. Phys.*, 226, 2007.

[48] R. Samulyak, T. Lu, and P. Parks. A hydromagnetic simulation of pellet ablation in electrostatic approximation. *Nuclear Fusion*, 47, 2007.

[49] P. Schwartz, M. Barad, P. Colella, and T. Ligocki. A cartesian grid embedded boundary method for the heat equation and poisson's equation in three dimensions. *J. Comput. Phys.*, 211, 2006.

[50] B. Solenthaler and R. Pajarola. Predictive-corrective incompressible sph. In *SIGGRAPH '09 ACM SIGGRAPH 2009 papers*.

[51] Volker Springel. Smoothed particle hydrodynamics in astrophysics. *Annual Review of Astronomy and Astrophysics*, 48:391–430, 2010.

[52] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, imcompressible, multi-fluid flows. *J. Comput. Phys.*, 100, 1992.

[53] S. Wang, J. Glimm, R. Samulyak, and X. Jiao. Embedded boundary method for two phase incompressible flow. *Submitted to J. of Computational and Applied Mathematics, available at arxiv.ogr*, 2012.

[54] S. Wang, R. Samulyak, and T. Guo. An embedded boundary method for elliptic and parabolic problems with interfaces and application to multimaterial systems with phase transitions. *Acta Mathematica Scientia*, 30, 2010.