SANDIA REPORT

SAND2016-9800
Unlimited Release
Printed September 2016

A Mesh-Free Method to Predictively Simulate Solid-to-Liquid Phase Transitions in Abnormal Thermal Environments

Lindsay C. Erickson, Karla Morris, Jeremy A. Templeton

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories

SAND2016-9800
Unlimited Release
Printed September 2016

A Mesh-Free Method to Predictively Simulate Solid-to-Liquid Phase Transitions in Abnormal Thermal Environments

Lindsay C. Erickson
Thermal/Fluids Science & Engineering Department
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94551-0969
lcerick@sandia.gov

Karla Morris
Digital and Quantum Info Systems Department
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94551-0969
knmorri@sandia.gov

Jeremy A. Templeton
Thermal/Fluids Science & Engineering Department
Sandia National Laboratories
P.O. Box 969
Livermore, CA 94551-0969
jatempl@sandia.gov

Abstract

Particle methods in computational physics are useful for modeling the motion of fluids and solids subject to large deformations. Under these conditions, mesh-based approaches often fail due to decreasing element quality leading to inaccuracy and instability. The developed software package called Moab investigates and prototypes next-generation particle methods, focusing on rigorous error analysis and active error minimization strategies during the computation. The present work discusses examples calculations representative of real engineering problems with quantified and maximized accuracy while demonstrating the potential for meeting engineering performance requirements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Metal melting and encapsulant decomposition significantly impact weapon systems safety in abnormal thermal environments. Currently, Sandia has no production-ready simulation capability to predict system failure due to complex thermal-mechanical processes or provide the computational verification and validation (V&V) required to confidently assess associated nuclear safety issues. Most existing finite element codes are also unable to adequately capture massive geometry changes and liquid relocation. Alternatively, Smooth-Particle Hydrodynamics has been applied to molten aluminum flows [43], but its formulation precludes phase changes and uncertainty quantification. Other particle formulations, such as the Reproducing Kernel Particle Method [35, 33], are amenable to rigorous error analysis and preserve physical quantities, e.g. viscosity, by retaining the integral form, but have difficulties maintaining high-order numerical quadrature as particles advect. Given the promise of mesh-free methods to accurately model melting and relocation, we have developed several novel capabilities to make these methods relevant to the needs of several Sandia mission spaces.

For any computational method to be used in support of the nuclear weapons mission, it must be amenable to V&V such that the numerical errors inherent in the method can be quantified and controlled to the necessary level. For many mesh-free formulations, this is problematic because of the difficulty demonstrating solution convergence with increased particle count as well as their reliance on *ad hoc* models, such as artificial compressibility, with affects which are not well described. Without being able to add particles to converge to the true solution of the partial-differential equations (PDE), the error can not be taken down such that the quantities of interest can be predicted with sufficient fidelity for the application. Similarly, unphysical models like the compressibility used in many SPH formulations make it challenging to compare with experiments to show that the physics models are of sufficient quality to produce useful simulation results.

To address these issues, we have developed a meshfree formulation in which consistency is enforced at the equation level rather than for differential operators. The result is that the governing equations are consistently transformed by integration against a kernel function and so the solution field can be directly compared to experimental measurements. More importantly, consistency conditions are computed on the fly so that more accurate computations are possible. Many meshfree methods are very sensitive to the length scale parameters used in the kernels. To avoid this problem, the particle topology is used to determine the length scale and consistency conditions used to ensure the quadrature weights can recover the volume of the kernel function. As a result, increasing particle density decreases the error of the solution.

While all these affects are significant, the most important one concerning this work are the behavior of interfaces. The Sandia applications in which meshfree methods are likely to be the most relevant are abnormal thermal environments and additive manufacturing. In both applications, the flow of molten metals is critical to predicting quantities of interest. In abnormal thermal environments, structural metals go through a melting process after which they can flow and relocate, changing the environment in which the thermal safety scenario must be evaluated. To capture melting, the interface between the solid and solidus must be correctly tracked in order to understand how the structure will change shape. In addition, oxide skins form on the surface of molten metal and are hypothesized to govern the rate at which the system deforms and how stable it is prior to deformation. In the case of additive manufacturing, the metal is flows while in the liquid phase and then solidifies to form a part. Like the abnormal thermal case, understanding the timing of this process is necessary to understand the thermal history of the part which leads to residual stresses.

While nearly all meshfree methods are capable of having an interface, their behavior is often unphysical or highly parameter dependent. Surface stresses are either a by-product of the internal model, which is not correct, or interfaces are identified by under-coordinated particles and empirical surface tension models are added. The authors are unaware of a capability documented in the literature to adequately model the physics of the interface. Therefore, a significant contribution of this work involve developing the new interface tracking techniques and a new particle movement strategy which are able to precisely represent interfaces. Both couple with the particle representation chosen to represent the problem, the former through a novel particle-level set method, and the latter by using the motion of interface particles to adapt the positions of internal particles.

In order to take advantage of these numerical methods, a Stefan problem [45] model has been implemented and tested. The Stefan condition is used to determine the speed at which a phase change interface moves based on the discontinuity of the heat flux across the interface. In the current implementation, particles are retained on the interface and also represent the solid and liquid. As the interface moves, particles switch between being solid or liquid. Using this technique, the temperature history of the solid can be measured.

This report is organized as follows. Chapter 2 describes the evaluation of many quadrature schemes and the proposed adaptive quadrature method to achieve consistent equation discretization. Another part of the meshfree method, interface tracking, is overviewed in Chapter 4 in which a new particle-level set method is presented. To take advantage of an accurately captured interface and the new adaptive quadrature method, Chapter 5 describes a way to move particles using methods inspired by molecular dynamics. Chapter 6 presents the Stefan problem formulation, and some concluding thoughts are offered in Chapter 7.

Chapter 2

Reproducing Kernel Particle Methods

Particle (meshfree) methods are suitable for problems that undergo large deformation and high gradients [31]. Meshfree methods require an efficient approach to approximate functions using a set of disordered particles. These particles serve as a discrete representation of the continuum problem. The solution is thus obtained by accounting for the weighted contribution of neighboring particles falling within a target smoothing length. This smoothing length defines a finite support of the so-called kernel function, which allows us to improve the accuracy of the solutions without sacrificing computational efficiency [32].

Smoothed Particle Hydrodynamics (SPH) methods are one of the most commonly-used particle methods, but lack accuracy near boundaries, or for configurations with small number of particles [32, 40]. Element-free Galerkin methods (EFGM) improve the accuracy of SPH methods by adding more accurate derivatives in the formulation [35]. Reproducing Kernel Particle Methods (RKPM) further improve accuracy by defining correction functions to account for boundary contributions [36]. These methods use a weak-form reformulation of the equations of interest, and exploit the weaker consistency on the approximate function. In general, these integral formulations produce a set of discretized system of equations that is both stable and produces accurate results [31].

In this chapter, we describe the differences between different RKPM algorithms, and discuss one example analyzing the behavior with respect to different kernel functions.

2.1 Numerical Approach

In this section, we first concentrate on describing properties and different types of kernel functions, which are the key component of meshfree methods. Then, we provide a high-level description of different RKPM algorithms.

2.1.1 Smoothing Kernels Function

These functions define the support domain of the particles used in the approximation, and at the same time determine the consistency and accuracy of the functions approximations [32]. A general expression for a (normalized) kernel function suitable for RKPM takes this form

$$\psi_{d_x}(x-s) = \frac{1}{d_x}\psi\left(\frac{x-s}{d_x}\right), \tag{2.1}$$

where d_x is the so-called dilation parameter, which sets the smoothing length, and ψ is the actual kernel. The above equation is such that all kernel functions, $\psi_{d_x}(x-s)$, satisfy the following requirements [32]:

Unity: To ensure that the integral of the kernel over the support domain is always equal to the unity, kernels must be normalized over their support domain.

Compact support: To allow local approximations to accurately capture global approximations kernels must have compact support. The locality of the operations contributes to the overall scalability of the methods. The support domain for any given particle x is determined by the dilation parameter, d_x.

Positivity: The kernel must be positive for any point within the support domain of a particle x. Following equation 2.1, the kernel will be positive for any point s that is within the support d_x. This property ensures that no physical meaning is lost in the approximation process.

Decay: The kernel value for a particle monotonically decreases with increasing distance away from the particle. This will guarantee that nearest neighbors have the largest influence on the particle of interest.

Delta function: The kernel satisfies the Dirac delta function condition as the smoothing length approaches zero.

Symmetry: The kernel must be an even function. The symmetry will allow for the contributions from points within the support to be defined based on the distance and independent of the position.

Smoothness: The kernel must be sufficiently smooth to ensure that the approximations are not sensitive to the distribution of the particles.

There are several kernel functions available in the literature, some of the most common ones are shown in figure 2.1.

2.1.2 Reproducing Kernel Particle Methods

There are several RKPM algorithms that have been published in the literature, see [4] and references therein.

The main idea behind a RKPM is to approximate a function $u(x,y)$ using a convolution integral with a corrected kernel as follows

$$u^a(x,y) = \int_\Omega C(x,y,s,t)\psi_{d_x}(x-s,y-t)u(s,t)dsdt = \int_\Omega N(x,y,s,t)u(s,t)dsdt, \tag{2.2}$$

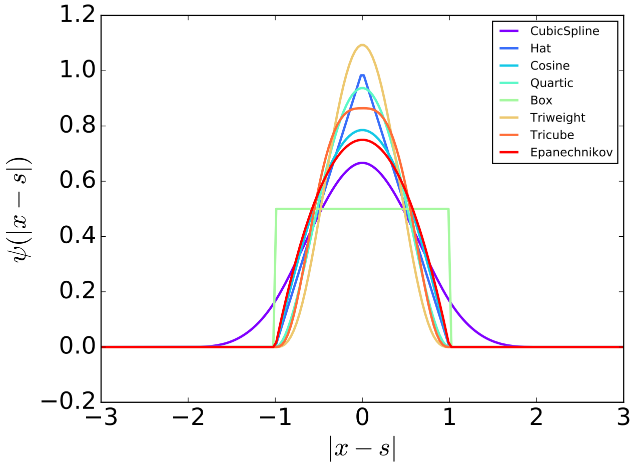


Figure 2.1: Commonly-used smoothing kernel functions, with $d_x = 1$ for all cases.

where $C(x,y,s,t)$ is the correction function, $\psi_{d_x}(x-s,y-t)$ is a kernel function, and $N(x,y,s,t) = C(x,y,s,t)\psi_{d_x}(x-s,y-t)$ is the shape function. In [3], the correction function is written as a polynomial expansion of the form

$$C(x,y,s,t) = c_0(x,y) + c_1(x,y)(x-s) + c_2(x,y)(y-t) + c_3(x,y)(x-s)(y-t) + \dots \tag{2.3}$$

where $c_0(x,y)$, $c_1(x,y)$... are the unknown correction function coefficients. The number of coefficients depends on the order of the polynomial expansion, which in turns is tied to the maximum order of the derivative present in the target PDE to solve. The coefficients of the correction function are obtained by solving a local linear system of equations (moment matrix) within the compact support of the kernel.

For the remainder of this section, we will focus on the following [4] four methods: point collocation, fixed point reproducing kernel, moving reproducing kernel, and multiple fixed reproducing kernel. All these methods are special cases of the general formulation described above. There are advantages and disadvantages associated with each of these.

Point Collocation Method The point collocation algorithm is based on

$$u^a(x,y) = \int_\Omega C(x-s,y-t)\psi_{d_x}(x-s,y-t)u(s,t)dsdt = \int_\Omega N(x-s,y-t)u(s,t)dsdt. \tag{2.4}$$

Both the correction function and kernel depend on the location x,y of the particle of interest and its neighbors s,t. Hence, for every particle ones needs to solve a separate linear problem to obtain the correction function coefficients for each derivative term in the target PDE. This increases the overall complexity of the algorithm, making this method computationally expensive [3].

Fixed Point Reproducing Kernel Method This method reduces the complexity of the point collocation method above by fixing the point (x_K, y_K) for the kernel evaluation [4]. The approximation of a function u is given by

$$u^a(x,y) = \int_\Omega C(x,y,s,t)\psi(x_K - s, y_K - t)u(s,t)dsdt = \int_\Omega N(x,y,x_K,y_K,s,t)u(s,t)dsdt. \quad (2.5)$$

Since x,y do not appear in the kernel expression $\psi(x_K - s, y_K - t)$, the representation for the derivatives of u is thus independent of the kernel itself. This eliminates the need to formulate a separate linear system for each derivative to compute the correction function coefficients. One drawback is that this leads to multi-valued shaped functions, which can affect the accuracy of the approximation [4]. To overcome this obstacle, for each point of reference (x_K, y_K), only neighboring particles (i.e. those within a predefined radius of influence) have a defined shape function, while all other particles have zero-valued shape function.

Moving Reproducing Kernel Method The approximation formula is

$$u^a(x,y) = \int_\Omega C(x,y,s,t)\psi(x-s,y-t)u(s,t)dsdt = \int_\Omega N(x,y,s,t)u(s,t)dsdt. \quad (2.6)$$

The shows that the correction function $C(x,y,s,t)$ is now constructed using the actual position s,t rather than the distances $x-s, y-t$. The evaluation of the kernel is done by moving the point of reference. The moment matrix and the kernel are now function of x,y, thus requiring the recomputation of the correction coefficients for approximating all the derivatives. This makes this method more expensive that the fixed point described before.

Multiple Fixed Reproducing Kernel Method The approximation takes the form

$$u^a(x,y) = \int_\Omega C(x,y,s,t)\psi_{s,t}(s-x,t-y)u(s,t)dsdt = \int_\Omega N(x,y,s,t)u(s,t)dsdt. \quad (2.7)$$

Here, the kernel is centered at s,t rather than being centered at x,y. The shape function is not multivalued, and both the moment matrix and the kernels are function of x,y, so that the method is more expensive than the simple fixed point RKM. This method, however, is more flexible than the moving RKM above, because one can define different fixed point kernels (shapes) at different locations.

Derivative Free RKPM The differential reproducing kernel interpolation (DRK) [54] is a derivative-free approach. This method has the potential to speed up RKPM calculations and increase the accuracy of our approximation to the derivatives compared to Aluru [3] and the classical RKPM method [35], where one must take the derivative of the moment matrix when obtaining derivative on the shape functions. For RKPM, one solves to u_i values so that the solution can be reconstructed as

$$u^a(x) = \sum_{i=1}^{N} C(x,s_i)\psi_{d_x}(x-s_i)u_i \quad (2.8)$$

where ψ_{d_x} is the kernel function, and $C(x,s_i) = \mathbf{p}^T(x-s_i)\mathbf{x}$ is the correction function. To obtain $C(x,s_i)$, we want to satisfy reproducing conditions such that the approximation exactly matches the function up to some order of polynomial, n.

$$\sum_{l=1}^{N} C(x,x_l)\psi_{d_x}(x-x_l)x_l^r = x^r \text{ for } r \leq n \tag{2.9}$$

The reproducing conditions in terms of the shape function, $\phi_l(x) = C(x,x_l)\psi_{d_x}(x-x_l)$, are

$$\sum_l \phi_l(x)x_l^n = x^n \tag{2.10}$$

and we want to show that

$$\sum_l \phi_l(x)\mathbf{p}(x-x_l) = \mathbf{p}(0) \tag{2.11}$$

satisfies the same constraint. To show this we will make heavy use of the binomial theorem,

$$(x-s)^n = \sum_{k=0}^{n} \binom{n}{k} x^k(-s)^k. \tag{2.12}$$

$$\begin{aligned}
\sum_l \phi_l(x)\mathbf{p}(x-x_l) &= \sum_l \phi_l(x)(x-x_l)^n & (2.13)\\
&= \sum_l \phi_l(x) \sum_{k=0}^{n} \binom{n}{k} x^k(-x_l)^{n-k} & (2.14)\\
&= \sum_{k=0}^{n} \binom{n}{k} x^k(-1)^{n-k} \sum_l \phi_l(x)x_l^{n-k} & (2.15)\\
&= \sum_{k=0}^{n} \binom{n}{k} x^k(-1)^{n-k}x^{n-k} & (2.16)\\
&= x^n \sum_{k=0}^{n} \binom{n}{k} (-1)^{n-k} & (2.17)\\
&= x^n(1-1)^n & (2.18)\\
&= 0^n = \mathbf{p}(0) & (2.19)
\end{aligned}$$

We want to show that

$$\sum_l \phi_l^{(x)}\mathbf{p}(x-x_l) = -\frac{\partial \mathbf{p}(0)}{\partial x} \tag{2.20}$$

is the correct constraint to place on the shape function and is equivalent to $\phi_l(x)$ satisfying the derivative reproducing conditions

$$\phi_l^{(x)}x_l^n = nx^{n-1}. \tag{2.21}$$

Again, we make use the binomial theorem,

$$\begin{aligned}
\sum_l \phi_l^{(x)} \mathbf{p}(x-x_l) &= \sum_l \phi_l^{(x)}(x)(x-x_l)^n & (2.22)\\
&= \sum_l \phi_l^{(x)}(x) \sum_{k=0}^{n} \binom{n}{k} x^k (-x_l)^{n-k} & (2.23)\\
&= \sum_{k=0}^{n} \binom{n}{k} x^k (-1)^{n-k} \sum_l \phi_l(x) x_l^{n-k} & (2.24)\\
&= \sum_{k=0}^{n} \binom{n}{k} x^k (-1)^{n-k} (n-k) x^{n-k-1} & (2.25)\\
&= x^{n-1} \sum_{k=0}^{n} \binom{n}{k} (n-k)(-1)^{n-k} & (2.26)\\
&= x^{n-1} \sum_{k=0}^{n} \binom{n-1}{k} n(-1)^{n-k} & (2.27)\\
&= -nx^{n-1} \sum_{k=0}^{n} \binom{n-1}{k} (-1)^{(n-1)-k} & (2.28)\\
&= -nx^{n-1}(1-1)^{n-1} & (2.29)\\
&= 0^n = -\frac{\partial \mathbf{p}(0)}{\partial x} & (2.30)
\end{aligned}$$

2.2 Error Analysis for the Fixed Point RKM

In this section, we evaluate the accuracy of the fixed point RKPM by exploring the effect of the order of the correction function, the kernel used, and particle spacing. We tested different PDE's and types of boundary conditions, and compare to the analytical solution to quantify the accuracy. To this end, we rely on the root mean square error (RMSE) defined as

$$\varepsilon = \frac{1}{|u^{exact}|_{max}} \sqrt{\frac{1}{N} \sum_{i=1}^{N} (u^a{}_i - u^{exact}{}_i)^2} \qquad (2.31)$$

where u^{exact} represents the exact solution.

Poisson Equation with Dirichlet Boundary Conditions The target problem is

$$\begin{aligned}
\frac{\partial^2 u}{\partial x^2} &= 2, & (2.32)\\
u(x=0) &= 0,\\
u(x=1) &= 1.
\end{aligned}$$

with analytical solution being $u(x) = x^2$. Figure 2.2 show the RMSE as a function of the kernel type. The results obtained using 2nd order correction functions and 3rd order correction functions are shown in red and blue respectively. The date shown with square markers are obtained using twice as many particles as those reported with the circle markers. For the explored kernels, the errors in all cases are below 10^{-10}.

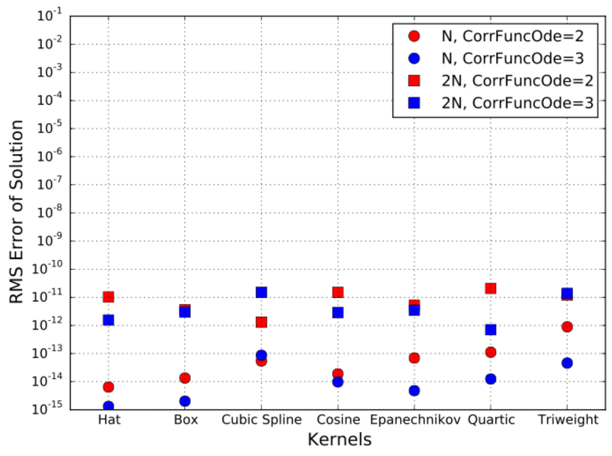


Figure 2.2: Root mean square error (RMSE) for 2nd order PDE 2.32. Function approximations using 2nd order and 3rd order correction functions are shown in red and blue respectively. Square markers are runs with double the number of particles compared to the run with circle markers.

Poisson Equation with Mixed Boundary Conditions The problem is defined as

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= 2, \\ u(x=0) &= 0, \\ \frac{\partial^2 u(x=1)}{\partial x^2} &= 2. \end{aligned} \tag{2.33}$$

with analytical solution being $u(x) = x^2$. Figure 2.3 show the RMSE for the solution of the PDE defined in Eq. 2.33. The cubic spline kernel shows consistently better results, but the approximations do not benefit from incrementing the order of the correction function. Even for the cubic spline, higher order correction functions degrade the accuracy such that the error is comparable to the results obtained with all the other kernels.

Poisson Equation with Non-Constant Source Term In this test, we chose the kernel to be the cubic spline, since it is the one that showed the best accuracy in the results above. We focus on a convergence test with respect to particle spacing for a Poisson equation with a non-constant forcing

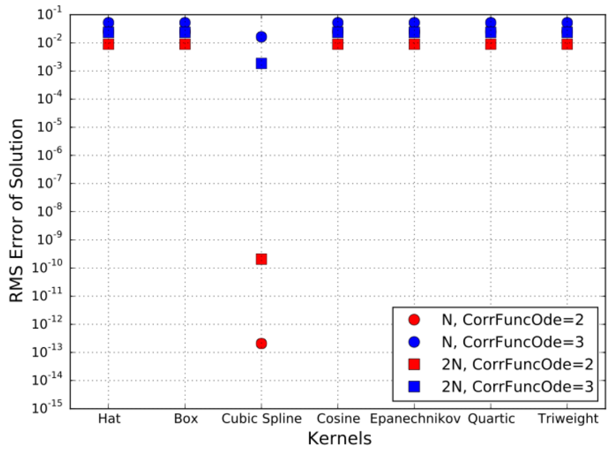


Figure 2.3: Root mean square error (RMSE) for 2nd order PDE with Neumann boundary conditions. Function approximations using correction functions of 2nd order and 3rd order are shown in red and blue respectively. Square markers are runs with double the number of particles compare to the run with circle markers.

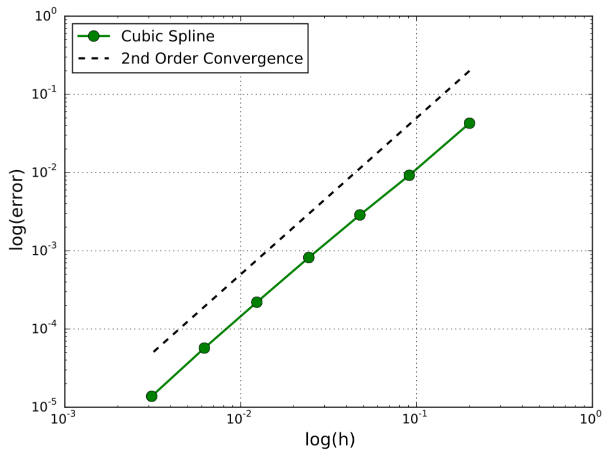


Figure 2.4: Convergence for the PDE in Eq.2.34. A fixed point method is used in combination with a cubic spline kernel. The black dashed line shows the theoretical 2nd order convergence.

term. The PDE used for this test is

$$\begin{aligned} \frac{\partial^2 u}{\partial x^2} &= \frac{105}{2}x^2 - \frac{15}{2}, \\ u(x=0) &= 3/8, \\ u(x=1) &= 17/8. \end{aligned} \tag{2.34}$$

with analytical solution $u(x) = \frac{35}{8}x^4 - \frac{15}{4}x^2 + \frac{3}{8}$. Figure2.4 shows the convergence results, which reveal that the method is second-order accurate with respect to particle spacing.

2.3 Conclusion

The different RKPM methods discussed above try to obtain a target level of accuracy while keeping a low computational cost. This combination is not always feasible. In some cases, the computational cost has to be balanced with the desired accuracy, which can result in more complex algorithms.

For the methods explored above, theoretically the accuracy should improve by increasing the order of the correction function. However, in the test cases evaluated here, we did not find this to be consistent across types of PDEs and boundary conditions. For instance, problems with Dirichlet boundary conditions in general meet this expectation, and a solution within a desired tolerance is obtained with all tested kernels. On the contrary, problems with Neumann boundary conditions do not have the same behavior. In this case, only the cubic spline kernel returned the correct solution using a second-order correction function. Higher-order correction functions did not improve the accuracy of the solution.

Chapter 3

Moab Particle Method

We have investigated the construction of a new particle method using a different PDE reformulation via integration by parts.

3.1 Integral Formulation for Poisson Equation

Consider the Poisson equation:

$$\nabla_{\mathbf{x}}^2 u(\mathbf{x}) = g(\mathbf{x}). \tag{3.1}$$

Taking the convolution with respect to a kernel function $\psi(\mathbf{x})$, we obtain

$$\int_\Omega \psi(\mathbf{y}-\mathbf{x}) \nabla_{\mathbf{y}}^2 u(\mathbf{y})\, d\mathbf{y} = \int_\Omega \psi(\mathbf{y}-\mathbf{x}) g(\mathbf{y})\, d\mathbf{y}, \tag{3.2}$$

where Ω is the domain of interest. We assume that ψ has compact support. The kernel ψ must have a sufficient number of continuous derivatives such that derivatives acting on the function u can be transferred to it. One integration by parts (IBP) yields

$$\int_{\partial\Omega} \psi(\mathbf{y}-\mathbf{x}) \nabla_{\mathbf{y}} u(\mathbf{y}) \cdot \mathbf{n}\, dS_{\mathbf{y}} - \int_\Omega \nabla_{\mathbf{y}} \psi(\mathbf{y}-\mathbf{x}) \cdot \nabla_{\mathbf{y}} u(\mathbf{y})\, d\mathbf{y} = \int_\Omega \psi(\mathbf{y}-\mathbf{x}) g(\mathbf{y})\, d\mathbf{y}, \tag{3.3}$$

with $\mathbf{n}$ being the outward normal to the boundary $\partial\Omega$. A second IBP provides an alternative formulation

$$\int_{\partial\Omega} \psi(\mathbf{x}-\mathbf{y}) \nabla_{\mathbf{y}} u(\mathbf{y}) \cdot \mathbf{n}\, dS_{\mathbf{y}} - \int_{\partial\Omega} u(\mathbf{y}) \nabla_{\mathbf{y}} \psi(\mathbf{x}-\mathbf{y}) \cdot \mathbf{n}\, dS_{\mathbf{y}} + \int_\Omega u(\mathbf{y}) \nabla_{\mathbf{y}}^2 \psi(\mathbf{x}-\mathbf{y})\, d\mathbf{y} = \int_\Omega \psi(\mathbf{x}-\mathbf{y}) g(\mathbf{y})\, d\mathbf{y}. \tag{3.4}$$

For the non-surface integrals, a numerical approximation is

$$\int_\Omega \psi(\mathbf{x}-\mathbf{y}) g(\mathbf{y})\, d\mathbf{y} \approx \sum_{n=1}^{N} \psi(\mathbf{x}-\mathbf{y}_n) g(\mathbf{y}_n) \Delta V_n, \tag{3.5}$$

$$\int_\Omega u(\mathbf{y}) \nabla_{\mathbf{y}}^2 \psi(\mathbf{x}-\mathbf{y})\, d\mathbf{y} \approx \sum_{n=1}^{N} u(\mathbf{y}_n) \nabla_{\mathbf{y}}^2 \psi(\mathbf{x}-\mathbf{y}_n) \Delta V_n. \tag{3.6}$$

If one can assume Ω to be infinite, then the boundary integrals drop out of the formulation. If this is not the case, then the numerical approximation of those boundary integrals depends on the dimensionality of the problem. For 1D, it reduces to a point evaluation, for 2D it becomes a line integral, and 3D is a surface integral.

In these approximations, functions or their derivatives are evaluated only at a discrete set of points $\mathbf{y}_n$ and summed against a quadrature weight ΔV_n. However, the problem is still continuous in that these integrals produce functions at every point in space $\mathbf{x}$.

The next step is then to discretize the problem in space. A mesh-based method would do this by providing a set of nodes at which variables (u,g) were stored and quadrature locations and weights for each $\mathbf{y}_n$ at which ψ and its derivatives are known and to which nodal variables can be interpolated. For integrals in which the variables appear directly, it is convenient to have the quadrature points and nodes to have coincident locations, denoted "particles", at which the integrals are evaluated

$$\int_\Omega \psi(\mathbf{y}-\mathbf{x}_m)g(\mathbf{y})\,d\mathbf{y} \approx \sum_{n=1}^{N} \psi(\mathbf{y}_n-\mathbf{x}_m)g(\mathbf{y}_n)\Delta V_{m,n}, \quad \forall m \in \{1,\ldots,N\} \tag{3.7}$$

$$\int_\Omega u(\mathbf{y})\nabla^2_{\mathbf{y}}\psi(\mathbf{x}_m-\mathbf{y})\,d\mathbf{y} \approx \sum_{n=1}^{N} u_n\nabla^2_{\mathbf{y}}\psi(\mathbf{x}_m-\mathbf{y}_n)\Delta V_{m,n}, \quad \forall m \in \{1,\ldots,N\}. \tag{3.8}$$

As the number of unknowns u_n is then the same as the number of equations, a well posed linear system governs the approximated system

$$A\mathbf{u} = \mathbf{f}. \tag{3.9}$$

Entries in each term are such that

$$\mathbf{u}_i = u_i, \tag{3.10}$$

$$\mathbf{f}_i = \sum_{n=1}^{N} \psi(\mathbf{y}_n-\mathbf{x}_i)f(\mathbf{y}_n)\Delta V_{i,n} \tag{3.11}$$

$$A_{ij} = \nabla^2_{\mathbf{y}}\psi(\mathbf{y}_n-\mathbf{x}_i)\Delta V_{i,j}. \tag{3.12}$$

Assuming a quadrature scheme providing ψ and $\Delta V_{i,n}$ (the weights in each equation need not be the same), this provides a discrete approximation to the full set of equations.

3.1.1 Methods for Determining Quadrature Weights

There are several methods that can be applied for quadrature weight estimation. The first is to take uniform weights. The weight for nodal integrals is $\Delta V_n = V/N$ where $V = \int_\Omega d\mathbf{x}$ is the total volume of the domain of interest. Similarly for stress points, $\Delta V_n = V/Q$. These are the same quadrature schemes as in SPH. However, there is no reason why each quadrature volume cannot be different, or even different depending on the quadrature pair. The RKPM method, for example, introduces a correction function for each quadrature pair, and as it is widely explored in the literature, will not be considered here.

An alternative would be to impose constraints on the system. If each quadrature volume is unique, there can be at most one constraint per quadrature volume. A logical constraint is that each quadrature can exactly integrate a constant function, e.g.,

$$\sum_{n=1}^{N} \psi(\mathbf{x}_m - \mathbf{y}_n)\Delta V_{m,n} = \int_{\Omega} \psi(\mathbf{x}_m - \mathbf{y})\, d\mathbf{y}, \quad \forall m \in \{1,\ldots,N\}, \tag{3.13}$$

but any function for which the convolution can be analytically evaluated is a possibility and derivative functions might best be tested against functions which they should exactly operate on, e.g., the first derivative should exactly recover the derivative of linear functions. This linear system can be solved directly. However, there is the issue of consistency amongst the terms in the equation as term multiplying u will contain either a first or second derivative of the kernel function. There is no reason why the quadrature weights must be the same, so it is possible to use different weights per term, each of which satisfies an equation similar to the one above.

We can also use the discrete approximations to the integrals to construct auxiliary conditions, for example, between the zeroth and second derivatives

$$\sum_{n=1}^{N} \psi(\mathbf{x}_m - \mathbf{y}_n)\nabla^2 p_k \Delta V_{m,n} = \sum_{n=1}^{N} p_n \nabla^2_{\mathbf{y}} \psi(\mathbf{x}_m - \mathbf{y}_n)\Delta V_{m,n}, \quad \forall m \in \{1,\ldots,N\}, \tag{3.14}$$

with p_k being the evaluation at quadrature location n of a polynomial of order k. Due to the terms in the equations, this can be viewed as either choosing a derivative stencil for ψ or selecting a different quadrature weight for each derivative order.

3.1.2 Boundary Conditions

Neumann or natural boundary conditions arise when the normal derivative of the solution at the boundary is specified. In this case, the prescribed value is simply substituted into the boundary integral terms, and then the entire term is moved to the right-hand side.

By contrast, application of Dirichlet or essential boundary conditions is often problematic in particle methods. For evaluation of zeroth-order integrals, the domain can be extended to infinity with zeros for the values outside of the domain by introducing the Heaviside function. This is equivalent to computing convolutions against the kernel function over Ω but without having to explicitly determine the integration over part of the kernel function support for the continuous integral when used as constraints on the discrete quadrature weights. As the primal variable in this method is u, Dirichlet conditions can be explicitly imposed by removing any particles at the boundary from the linear system and using the prescribed values as they appear.

To deal with the problem of the derivative at the boundary, one can use a one-sided finite difference for particles adjacent to the wall. There is no need to explicitly compute the values for particle at the wall because they are known to correspond to the prescribed data, hence a simple first order finite difference approximation would be:

$$\nabla u(\mathbf{y}) \cdot \mathbf{n} \approx \frac{u_n - u|_{\partial\Omega}}{d}, \tag{3.15}$$

where d is the distance of particle n from the wall. If higher order approximations are needed, then explicit estimation of the convolved field in the extended domain can be computed (see Templeton & Shoeybi). Note that the term proportional to u at the boundary can either be incorporated onto the left-hand or right-hand sides if the boundary conditions are specified as Neumann or Dirichlet, respectively.

3.2 Poisson Equation Results

To explore the accuracy of the method described above, we solved a set of test PDEs. Here, we present two second order PDEs. The first one is the following

$$\frac{\partial^2 u}{\partial x^2} = 2, \quad u(x=-\pi)=0, \quad u(x=\pi)=0, \tag{3.16}$$

solved over the interval $(-\pi, \pi)$ with analytical solution $u(x) = x^2$. The second one has a sinusoidal forcing term

$$\frac{\partial^2 u}{\partial x^2} = -\sin x, \quad u(x=-\pi)=0, \quad u(x=\pi)=0, \tag{3.17}$$

solved over $(-\pi, \pi)$ with analytical solution $u(x) = \sin(x)$.

Figure 3.1 shows the results obtained for these two PDEs using the particle method described in § 3.1. The first row shows the results obtained for Eq. 3.16, while the second row shows those obtained for Eq. 3.17. The first column shows the approximation of the function at the particle positions with blue markers, and the superimposed analytical solution of the PDE with black solid line. As a representative case we show results for $N = 30$. The right column shows the RMS error plotted as a function of the particle spacing.

For Eq. 3.16, Figure 3.1 (a,b) show that the particle method is able to accurately approximate the function as the results for RMS error are within what can be consider as noise. The polynomial bases functions in Eq 3.14 used to calculate the quadrature volumes needed to approximate the functions according to Eq 3.10 include the term with the same for of the analytical solution of the PDE. This similarity can explain why the error results are just noise. The approximation of Eq. 3.16 serves as a consistency check for the method, such that quadrature volumes calculated for the polynomial bases satisfy the integral approximation of PDEs with similar solution. Although not shown here this is the case for PDE's with analytical solutions for polynomials of various orders.

For Eq. 3.17, Figure 3.1 (c,d) show that the method successfully approximates the function of interest. In this case the form of the analytical solution of the PDE (sinusoidal function) is different than the polynomial basis used for the calculation of the quadrature volumes. Even in the absence of that similarity the method is able to approximate the integral of the PDE. Although the magnitude of the errors is larger in this case, we are able to improve the results by decreasing the particle spacing.

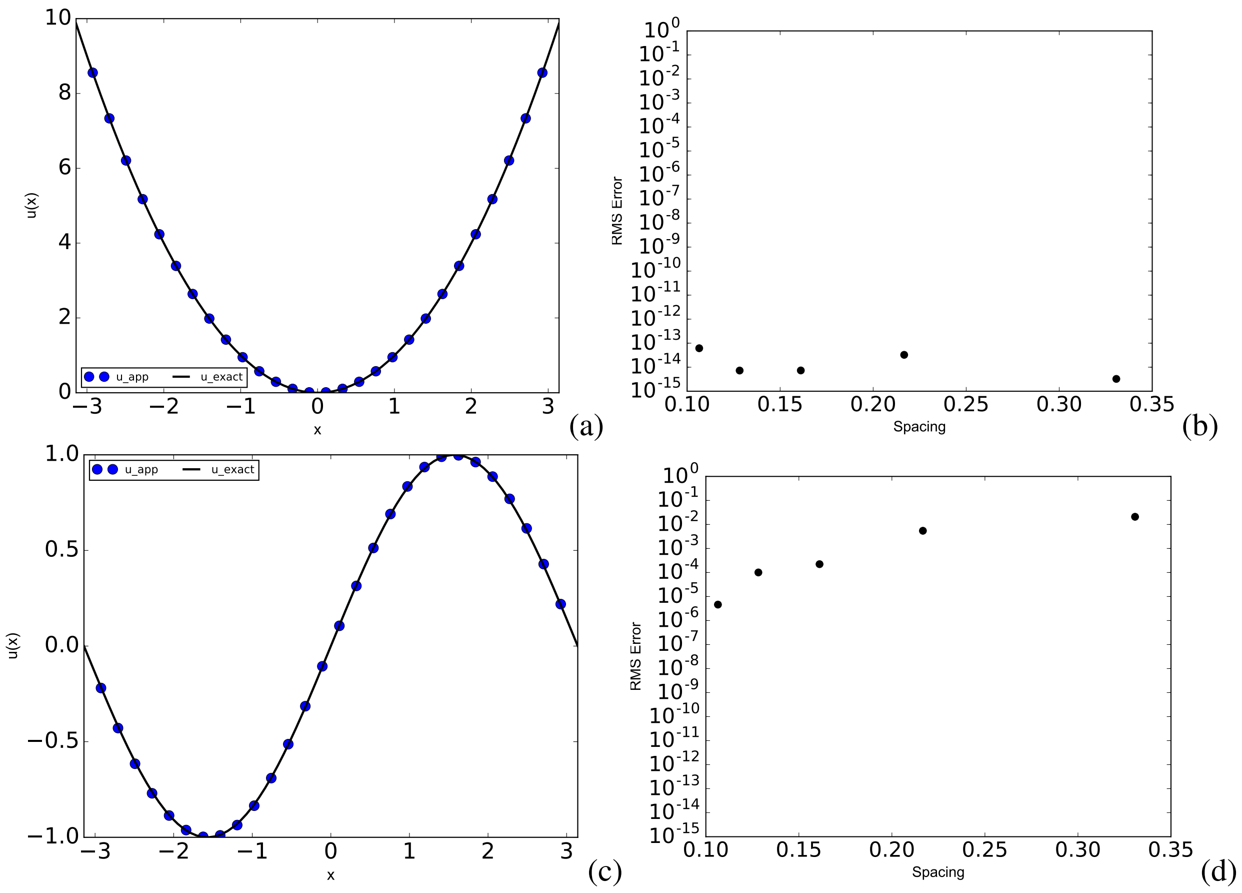


Figure 3.1: Top row: (a) function approximation superimposed to the analytical solution obtained for $N = 30$ for Eq. 3.16; (b) RMS error as function of the particle spacing for Eq. 3.16. Bottom row shows similar results obtained for Eq. 3.17.

All results shown are obtained using the Epanechnikov kernel, which proved to give the most accurate solution. The errors are sensitive to the dilation parameter and it possible that a more extensive analysis can be perform to identify the appropriate parameters required to increase the accuracy of the other kernels.

3.3 Conclusion

This method was developed in an attempt to reduce the computational cost with respect to existing RKPM methods. Similarly to other RKPM methods, this one also uses an integral formulation of the target PDE. The method relies on finding a systematic way to compute the quadrature volumes used for correcting the approximation of the integral PDE. This is done by leveraging a polynomial basis in a formulation that is consistent with the PDE of interest.

Local quadrature volumes can be calculated by solving a system of equations that includes constraints such as the sums of the quadrature volumes equals the total volume of the local domain, or the integral of the kernel within that local domain, or combinations thereof.

We explored 1D Poisson equations with different source terms. We tested multiple kernels, and found that only the Epanechnikov one worked suitably. We observed the solution to be sensitive to the dilation parameter, which required a systematic process to find appropriate value for the Epanechnikov kernel. Follow-up analysis will be needed to identify the appropriate value for the other kernels.

Chapter 4

An Interpolative Particle Level Set Method

4.1 Introduction

There exist a wide range of applications for solutions to multiphase flow problems with moving interfacial dynamics. These include engineering, fluid mechanics, melting metals, geophysical, medical, computer graphics and image processing. Over the years there have been a large effort in the numerical method community to solve these types of problems. Capturing topological changes with physical accuracy remains a challenge.

The two main computational approaches for simulating moving interfaces can be categorized as interface capturing (most notably volume of fluid (VOF) [27] and the level set method [41]) and interface tracking methods (Lagrangian methods [53]). Interface capturing methods are typically Eulerian methods, requiring a grid, an N dimensional implicit function (e.g. a color function or a signed distance function) to define an $N-1$ dimensional interace and an advection-type governing equation to evolve the implicit function. On the other hand, interface tracking methods use Lagrangian marker particles to distinguish the interface. These particles evolve with some prescribed velocity for interfacial motion.

Advantages of the level set method include natural merging and pinch-off behavior as well as straightforward calculation for the interface normal vector and the radius of curvature. However, mass conservation due to numerical diffusion is a problem that plagues this approach. Reinitialization of the signed distance function is typically necessary for the level set to retain its signed distance property and to limit mass loss. Reinitialization procedures are also prone to numerical diffusion and without careful implementation have the tendency to move the zero level set interface, which is not desired. Another downside of the level set method is that it is limited by the grid size - finer features of the interface or regions of high curvature cannot be resolved if they are thinner than the local grid width.

Lagrangian particle methods conserve mass by nature and are excellent at resolving fine scales and curvatures of the interface in flow regimes that do not cause major deformation or stretching of the interface. The downside is that a large number of points are needed to create the interface and a special approach must be in place to back out the surface geometry (e.g. the surface normal and curvature) since there is no connectivity between particles. These methods fail the shrinking square test [41] and cases with merging fronts, but this is due to how the velocity gets interpolated from a background mesh [19]. Reseeding is also necessary as the interface gets stretched, since the particles can get spread out and fine scale resolutin gets lost. A self organizing particle method [44] has been developed, where particles move to adapt to local resolution requirements. As holes and particle clustering form, particles get essentialy remeshed using pseudo-forces and dynamic insertion and removal. In addition, it is worth noting that topological changes must be specially handled in Lagrangian particle methods, again since there is no measure of connectivity between particles. A Lagrangian particle level set method was developed by Hieber *et al.* [26] using techniques from vortex methods and particles as essentially quadrature points. This paper develops an approach to cutting and reconnecting the interface.

Much progress has been made by combining the advantages of both the level set method and Lagrangian particle methods, i.e. hybrid-type methods. A self-adaptive particle level set method was developed by Ianniello *et al.* [28], with the idea of oriented Lagrangian points (that include the normal and tangential vectors as part of their owned data). The particles, normals, and tangent vectors each have their own governing equations and do not require a signed distance function to calculate or integrate the interfacial dynamics. Particles here lie exactly on the interface, and the side of the interface information is carried by the normal vector direction. This purely Lagrangian approach circumvents any numerical diffusion. The level set is later postprocessed for fast reconstruction of the interface using the Eikonal equation. This approach switches the roles of the level set and particles used in the hybrid particle level set method developed by Enright *et al.* [18]. In this context particles play the main role of tracking the interface and use a high order Runge Kutta integration scheme to recreate a continuous interface, while the level set function acts as a post-processing tool. The method requires a re-seeding algorithm to manage the particle distribution and to check the accuracy of the interface constructed by the Lagrangian particles.

In a foundational paper, the hybrid particle level set method was developed by Enright *et al.* [18]. Lagrangian particles are placed near the interface and are used to correct the level set function for mass loss (in addition to a traditional reinitialization approach) when "escaped" particles are detected. Adjacent to both sides of the interface defined by the level set equation, massless marker particles of randomly varying size are initially placed. They are given a sign (positive or negative) and move with same velocity field used for advection of the signed distance function. When these particles end up on the wrong side of the interface due to numerical error, the particles are used to correct the signed distance field using the radius of the marker particle as a measure of the local level set. In this method, a 5th order WENO scheme for the computation of the spatial term $\nabla\phi$ is combined with a 3rd order TVD Runge Kutta procedure for time integration [29]. In a following paper [19], they show that this correction procedure makes high order integration schemes for the level set funciton unnecessary. Instead, a semi-Lagrangian method [50] coupled with a first order fast marching method [47] for reinitialization is used as a faster alternative (and the resulting numerical diffusion is effectively mitigated with the incorporation of the particle correction procedure.

A version of the particle level set method also later formulated by Wang *et al.* [55]. This work accounts for particle distances on the other side of the level set, which were originally getting set incorrectly (and ultimately not contributing to the method). This approach expands on the number of grid points that an escaped particle can update, but also further limits the particles that are used to update the interface by not using the local level set of particles that are too far off from the normal to the interface at a grid point.

Most particle-level set hybrid methods methods use a large number of particles to preform their calculations (64 per cell in 2D in [18]), and most of these particles do not even contribute to the correction procedure since only the escaped particles contribute to updating the level set function. In this work we suggest a different approach, using all particles adjacent to the level set and within one grid spacing, and are able to get a smooth and accurate method with only 12 particles per cell close to the interface. We are able to accomplish this by instead using an interpolation scheme to update grid points near the interface using the distances of nearby particles (escaped or not). Using this approach we do not have to check the escaped status of a particle or calculate the projection of the distance between particle and grid point to see if it is normal or tangent to the interface.

4.2 Numerical Approach

4.2.1 The Level Set Method

The level set method relies on a signed-distance function field

$$\phi(\mathbf{x},t) > 0 \quad \text{for} \quad \mathbf{x} \in \Omega \tag{4.1}$$

$$\phi(\mathbf{x},t) \leq 0 \quad \text{for} \quad \mathbf{x} \notin \Omega \tag{4.2}$$

where the iso-contour $\phi = 0$ defines the zero level set $\Gamma(t) = \{x \in \Omega : \phi(x,t) = 0\}$, also called the zero level set. The level set function is initialized as a signed distance function,

$$\phi(\mathbf{x},0) = \phi_0(\mathbf{x}), \tag{4.3}$$

where $|\nabla\phi_0| = 1$ everywhere. An advection equation is typically used to project the level set function in time

$$\frac{\partial\phi}{\partial t} + \mathbf{u}\cdot\nabla\phi = 0 \tag{4.4}$$

where $\mathbf{u}$ is the velocity field that the interface is subject to. The unit normal gets computed directly from ϕ,

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{4.5}$$

and the curvature is defined as

$$\kappa = \nabla\cdot\mathbf{n} = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{4.6}$$

Both of these quantities can be easily backed out directly from the level set field, whereas this could be a much more difficult task for a particle-only approach as there is generally no connectivity information. The surface curvature is important for calculating the surface tension and other physical parameters for a variety of applications.

Reinitialization

The level set suffers from mass loss over time due to numerical diffusion, and typically this is corrected for by periodically reinitializating $\phi(\mathbf{x})$ to be a signed distance function

$$\frac{\partial \phi}{\partial \tau} + S(\phi_0)(|\nabla \phi| - 1) = 0 \tag{4.7}$$

where $S(\phi_0)$ is the sign of the initial level set variable prior to reinitialization. This function typically gets smeared out

$$S(\phi_0) = \frac{\phi_0}{\sqrt{\phi_0^2 + (\Delta x)^2}}. \tag{4.8}$$

Instead of simply advecting $\phi(\mathbf{x})$ according to equation 4.4, we periodically take the zero level set as gold, and match the surrounding values of ϕ to be the signed distance to the zero level set using Equation 4.7, using pseudo-time integration. This can be done using a 5th order WENO scheme [37] or alternatively one could use first order fast marching methods for reinitialization procedure [1, 47].

4.2.2 Lagrangian Particle Motion

Tracker or marker particles can be used to define an interface. We use two sets of marker particles near the interface to define either side of it (positive and negative particles). Their motion is governed by the Lagrangian equation,

$$\frac{\partial \mathbf{x}_p}{\partial t} = \mathbf{u}(\mathbf{x}_p). \tag{4.9}$$

where $\mathbf{x}_p$ is the spatial location of particle p.

We initialize particles near the initial zero level set field in a pseudo-random fashion, on both sides of the interface with a normal distribution of $2\Delta x$. The radii for these particles depends on their initial distance from the interface, and once they are placed, that information is backed out and owned by the particles. Particles are also given a sign (plus or minus, denoted s_p) depending on which side of the interface they lie on. The side of the interface they are on remain fixed; only their location in space changes according to Equation 4.9.

4.2.3 The Particle Level Set Method

In the particle level set method, each particle is initialized with a sign and a radius. The particle radii are defined based on the level set function (the distance from the zero level set),

$$r_p = \begin{cases} r_{max} & \text{if } s_p\phi(\mathbf{x}_p) > r_{max} \\ s_p\phi(\mathbf{x}_p) & \text{if } r_{min} \le s_p\phi(\mathbf{x}_p) \le r_{max} \\ r_{min} & \text{if } s_p\phi(\mathbf{x}_p) < r_{min} \end{cases} \tag{4.10}$$

where s_p is the sign of the particle (± 1 depending on the side of the interface they reside on). The cutoff radii, r_{max} and r_{min}, depend on the grid spacing. The cutoff radii suggested in [18] are $r_{max} = 0.5\Delta x$ and $r_{min} = 0.1\Delta x$, where $d\Delta x$ is the average grid spacing.

The calculation of the local levet set at adjacent grid points per particle p, is $\phi_p(\mathbf{x})$, and is used in the reinitialization procedure:

$$\phi_p(\mathbf{x}) = s_p(r_p - |\mathbf{x} - \mathbf{x}_p|) \tag{4.11}$$

the zero level set of ϕ_p is the boundary of the marker particle sphere. In the correction particle level set method [19], there is a slight change here

$$\phi_p(\mathbf{x}) = s_p(r_p \pm |\mathbf{x} - \mathbf{x}_p|) \tag{4.12}$$

with the $\pm$ sign corresponding to what side of the interface an escaped particle is on (need better description here).

4.2.4 Error Reduction Procedure

To limit the error of mass loss suffered by the traditional level set method, the marker particles in the particle level set method [18] are used to correct the level set field for errors near the interface. This is done using the escaped particles - that is particles that have appeared to switch sides of the interface due to numerical errors in the level set equation advancement or by movement of the zero level set by the reinitialization procedure. The details of the method are described here. Fields $\phi^{\pm}$ are defined using positive and negative escaped marker particles,

$$\phi^{+}(\mathbf{x}) = \max_{p \in E^{+}}(\phi_p, \phi). \tag{4.13}$$

$$\phi^{-}(\mathbf{x}) = \min_{p \in E^{-}}(\phi_p, \phi). \tag{4.14}$$

By choosing the maximum (minimum), the $\phi_p(\mathbf{x})$ that trumps the other values is going to be the one that is closest to the normal of the interface at a given grid point.

$$\phi(\mathbf{x}) = \begin{cases} \phi^{+}(\mathbf{x}) & \text{if} \quad |\phi^{+}(\mathbf{x})| \leq |\phi^{-}(\mathbf{x})| \\ \phi^{-}(\mathbf{x}) & \text{if} \quad |\phi^{+}(\mathbf{x})| > |\phi^{-}(\mathbf{x})| \end{cases} \tag{4.15}$$

4.2.5 Distinctions for our approach

In our approach, $r_p = \phi_p = \phi(\mathbf{x}_p)$, the signed distance from the zero level set and we use bilinear interpolation at each grid point to update the "coarse" grid level set function with the information from the "finer" set of particles near the interface (see Figure 4.1). Our Lagrangian particles do not get reinitialized since they reside near the zero level set (which, within Δx, remains fixed during a reininitialization event). This is not necessary for a rigid rotation problem (Zalesk's disk), but we may need a way to handle this for problems where there is shear flow and stretching near the interfacial region (like that in the circle in a vortex flow problem, for instance).

4.2.6 Particle reseeding procedure

As an interface deforms and stretches, the original set of particles also gets distorted and particle re-seeding becomes appropriate. Particles need to be added in regions where the interface gets stretched and particles should be deleted if they move too far away from the interface. This is done here by keeping track of the number of particles in each grid cell near the interface and keeping the number of those particles up to a given threshold (e.g. 12 or 64 particles per cell). In grid cells far enough away (e.g. $3\Delta x$) from the zero level set, particles will be deleted.

4.3 Results

4.3.1 Pseudo 1D Test

In order to compare methods, we tested our method with the original particle level set method [18] and the corrected version [55]. We looked at a pseudo one dimensional test case in which the particles and the level set defined on the (2D) grid were given a linear profile ($\phi(\mathbf{x}) = x - 0.5$). Then the level set field was given a constant error by shiting it by $\Delta x/2$, so the particles and the level set field differ by $\Delta x/2$. We assume that the particles are "correct" and that there is error in the level set field, and use each of the three methods to attempt to correct the zero level set. The results are shown in Figures 4.2a, 4.2b, and 4.3.

Other particle correction methods require a large number of particles per cell (64 used in both paper), yet only a small fraction (the "escaped" particles that have crossed from the positive side to the negative side of the level set field, or vice versa) get used. Our interpolative approach works better when the original hybrid particle level set method fails. The original approach assumes that the closer particle correction function is more accurate (a good assumption in general), but there

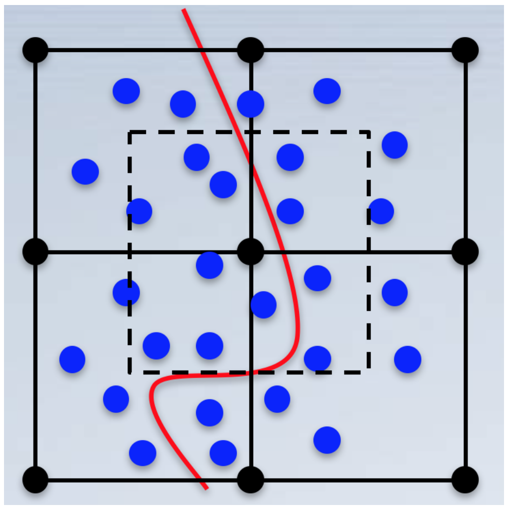
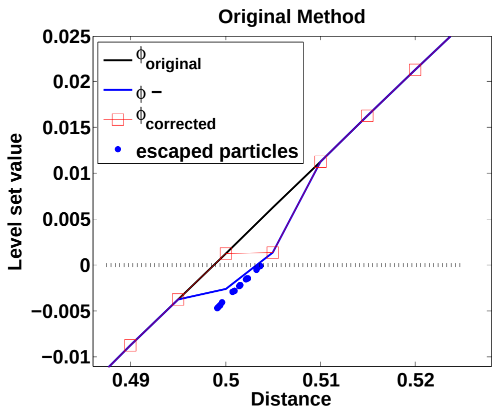
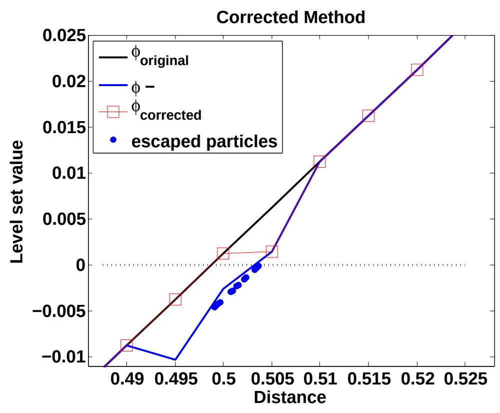


Figure 4.1: Illustrating our particle correction method



(a) Original method [18].



(b) Corrected method [55].

Figure 4.2: Illustrating particle correction for a psuedo 1D problem in which the level set field is off by $\Delta x/2$ and there is no error in the particle positions and radii. In the original method (a), the location of the zero level set does not get updated, and the same is true for the corrected method (b).

are some cases where this breaks down. If a particle is used to correct the interface is farther than the grid point it is correcting, the level set field does not get updated. This is more likely to occur when larger time steps are taken, closer to the CFL limit. Therefore our method can work with larger time steps, whereas the original method needed further time-step restriction.

4.3.2 Offset Circle Test

A similar test was used to compare these methods in 2D on a circle. The circle's level set was offset by Δx in both the x and y directions for the grid nodes, while the particle signs, positions and radii are correct. The initial condition is shown in Figure 4.4a and the results of the three methods compared are shown in Figures 4.4b, 4.4c and 4.4d. The interpolation method outperforms the other methods for this case in giving a smooth solution since it uses information from all particles within $\Delta x/2$, instead of just the escaped particles. If the level set is only off by a small fraction of the grid spacing, all methods perform well. This demonstrates that one should be able to evolve for longer time periods using the interpolation method between particle correction steps as well as using fewer particles.

4.3.3 Slotted Disk Test: Rigid Rotation

The slotted disk problem (Zalesak's disk [56]) is a classic test problem for checking for error due to numerical diffusion. Figure 4.5 shows the results of this test. We use a 80 by 80 grid to solve the level set equation on all three approaches. The level set method is solved using a 5th order WENO scheme with reinitialization events occurring every 10 time steps. For the level set equation solver alone, we see numerical diffusion taking over, and the slotted disk loses volume and shape. Both particle correction procedures alleviate this problem and limit numerical diffusion, however our interpolative particle correction method provides a smooth interface, whereas the original particle level set method has a bumpy interface that varies over time in a way that is non-conducive to calculations of the normal vector and curvature on the interface.

4.3.4 Circle Under Vortex Flow Test

Our new particle correction method is further tested for a circle in a vortex flow. This problem tests the method ability to resolve the formation of thin filaments. We use a 80 by 80 grid to solve the level set equation for all three approaches. The level set method is solved using a 5th order WENO scheme with reinitialization events occurring every 10 time steps. Figure 4.6 shows the results of this test. The level set method alone loses resolution quickly as the circle is stretched out thinner than the mesh underlying mesh resolution. Both particle correction approaches attempt to capture the thin filament as it forms, but our interpolative method performs better since we are able to use more of the surrounding particles (instead of just the escaped particles that have crossed the zero level set), and is able to resolve the thinning circle as it revolves around the domain.

4.4 Conclusions

A novel interpolative particle level set method has been developed and has shown improvement on the original particle level set method for a variety of practical test problems. A particle correction approach to minimizing error due to numerical diffusion caused by the level set method is shown to be an effective approach, while maintaining the benefits of the level set method, such as the ability to calculate the normal vector and the curvature at the interface as well as naturally handling merging and pinch-off of the interface as it evolves in time.

This approach is meant to be coupled to adaptive particle motion based on error minimization (Chapter 5) for the interior particles. The background gird and finite difference method used here for solving the level set equation will be replaced with the method and we hope to see further improvements for larger systems of coupled equations containing interfacial dynamics.

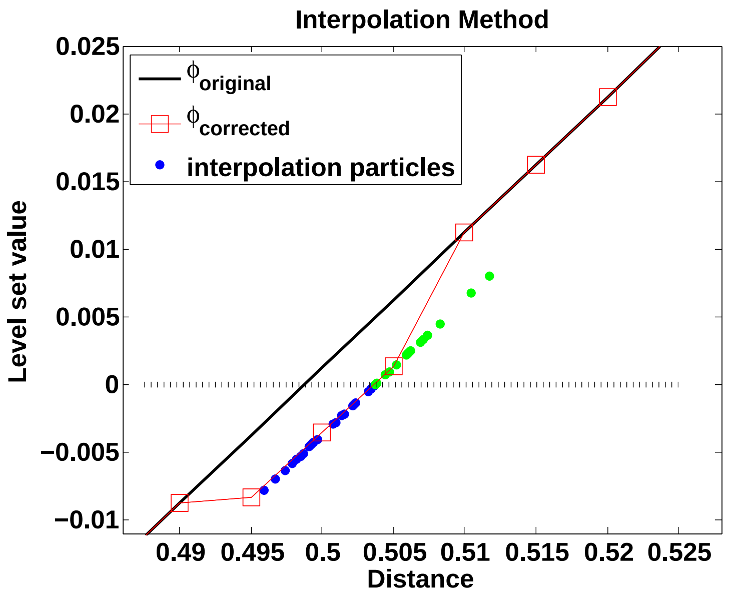
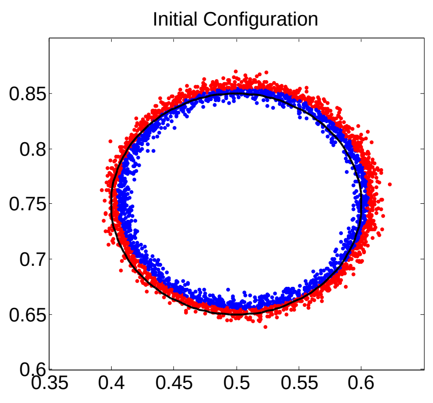
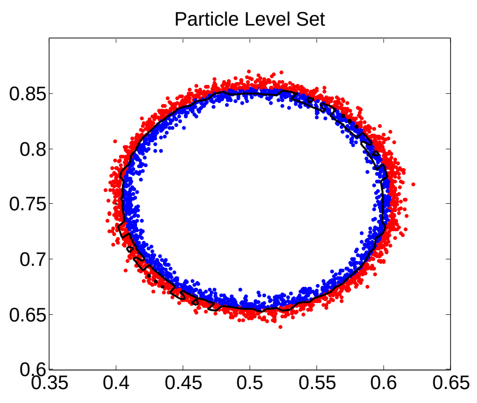


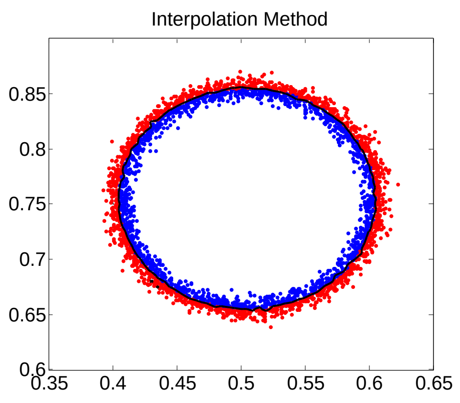
Figure 4.3: Illustrating particle correction for a psuedo 1D problem in which the level set field is off by $\Delta x/2$ and there is no error in the particle positions and radii for the new interpolation method proposed in this paper.



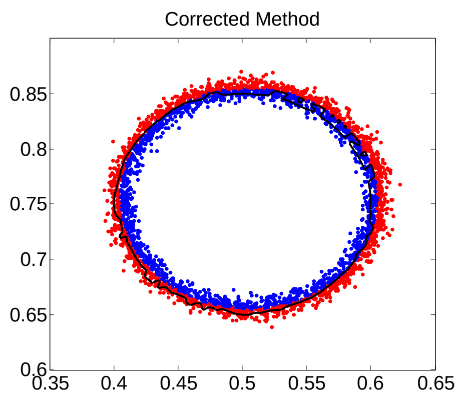
(a) Initial Condition



(b) Particle level set method [18].



(c) Interpolation method



(d) Corrected method [55]

Figure 4.4

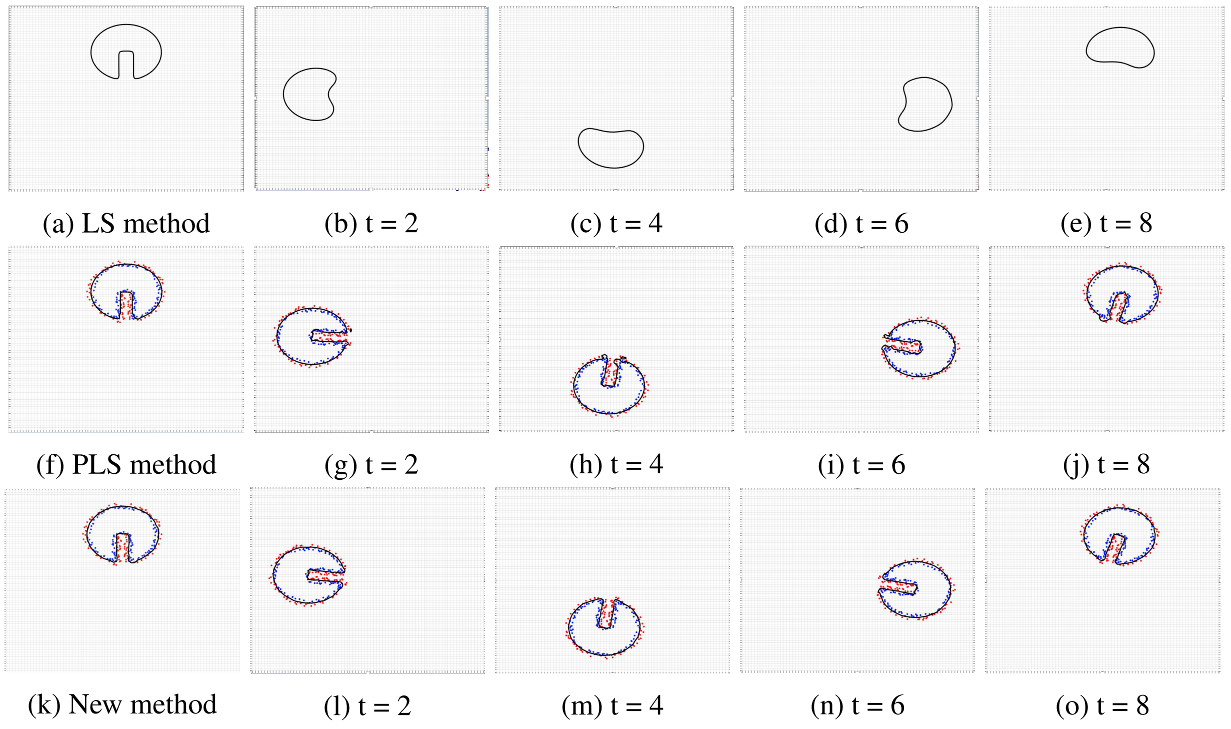
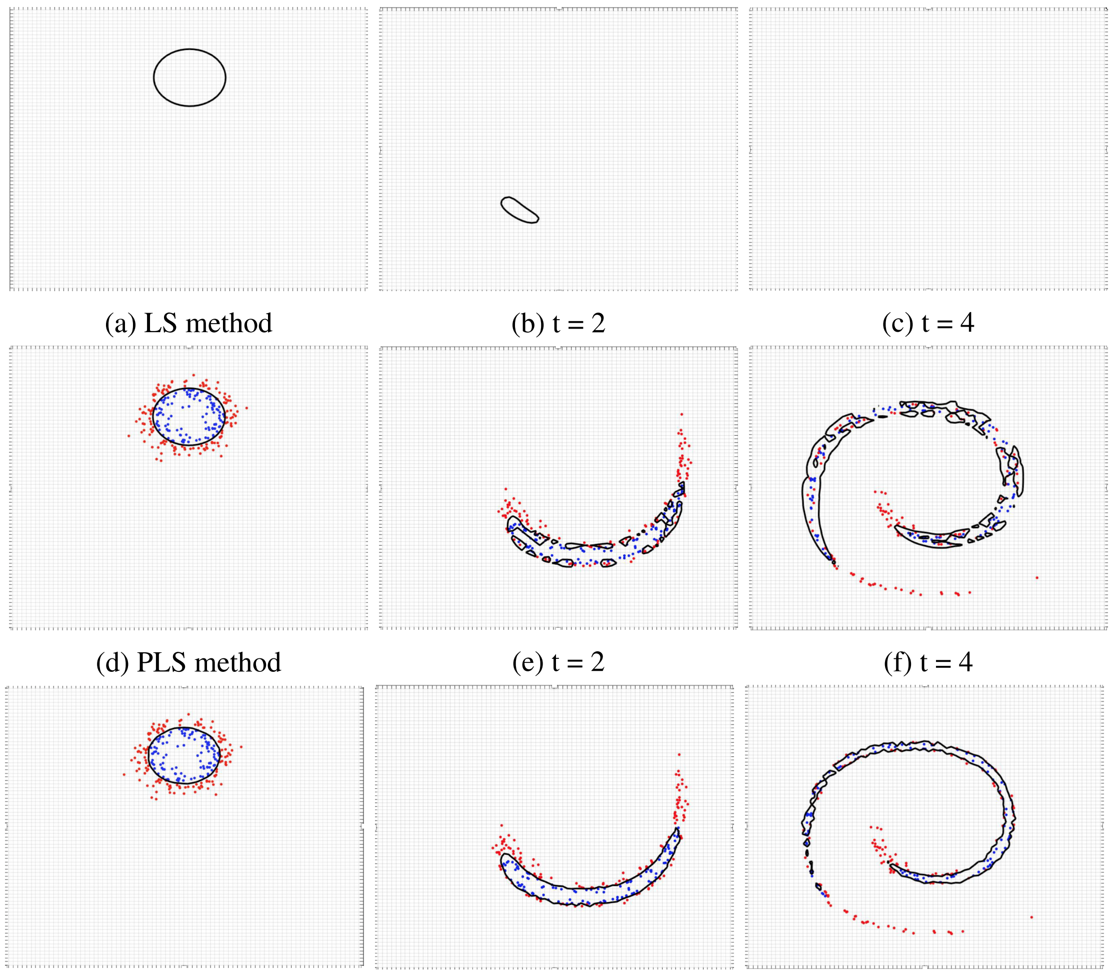


Figure 4.5: Illustrating particle correction for a slotted disk, rigid rotation flow field test problem. All tests are on an 80 by 80 grid and both particle correction techniques use the same number of particles. (a)-(e) is the level set (LS) method for a 5th order WENO scheme with reinitialization. Numerical diffusion results in loss of the high curvature details of the shape. (f)-(j) show the original particle level set method and how the max/min correction procedure can result in jumpy, non-smooth edges near the corners. (k)-(o) show our new interpolative particle level set method in practice. Our correction procedure is able to keep numerical diffusive due to the LS method at bay, while keeping the interface smooth.



(a) LS method (b) t = 2 (c) t = 4

(d) PLS method (e) t = 2 (f) t = 4

(g) New method (h) t = 2 (i) t = 4

Figure 4.6: Illustrating particle correction for the circle in a vortex flow problem. This problem tests the method ability to resolve the formation of thin filaments. All tests are on an 80 by 80 grid and both particle correction techniques use the same number of particles. (a)-(c) is the level set (LS) method for a 5th order WENO scheme with reinitialization. (d)-(f) show the original particle level set method and how the restriction to only use escaped particles can limit the method's effectiveness. (g)-(i) show our interpolative particle level set method and how using all surrounding particles we are able to better capture thin filaments below the grid resolution level.

Chapter 5

Adaptive Particle Motion for Error Minimization

5.1 Introduction

Meshfree, or particle, methods, are useful for capturing moving regions of fluid or largely deforming regions of solids because the particles, which often serve as nodes, can move as the topology of the system moves. Particle motion is based on the local velocity such that a particle follows the local streamlines. Because of this property, meshfree approaches can integrate the advection term in partial differential equations (PDEs) with high accuracy while any other terms in the equation, such as diffusion, are approximated using the particle layout and values. It is these terms which often suffer in a particle method due to large errors associated with disordered particle arrangements [51, 24, 20, 58].

While initially conceived for advection dominated problems in astrophysics [23, 38], particle methods are widely used in problems in both fluid (e.g., [7, 48]) and solid (e.g., [9]) mechanics. In addition, they can be formulated for elliptic and parabolic problems as well which do not contain advection by approximating derivative operators as quadratures on point clouds [36, 2, 11]. However, in these cases the advantage of the particle formulation is lost because a general quadrature approach will not be as accurate as a more structured one, for example finite elements, where quadrature points are specifically placed. While not as relevant to this class of problems, meshfree methods do have a potentially important role to play in these applications when the boundary moves, i.e., the flow is dominated by the interface while within the interface the physics is governed by elliptic or parabolic equations. Such flows are commonly found in manufacturing applications [43].

In this work, the focus is on identifying an approach that can efficiently move particles in a meshfree scheme when advecting them will lead to large numerical errors in the differential operators. The main motivator are elliptic problems driven by moving interfaces, but even established meshfree schemes can benefit by having alternate ways to realign particles to reduce numerical errors incurred by random particle placement. Initializing and creating new particles in meshfree schemes have been of recent interest, for example to set initial conditions [15] or to split particles for refinement [13]. It is also noted that exactly solving this problem would be impractical. In order to do so, the Green's function associated with a source at each particle would have be computed along with an error estimate owing to the discretization at each point. Then the total error in the solution is the linear combination of the Green's functions weighted by the error estimates. In order to reduce this error, a minimization algorithm would be needed, which would need to recompute each Green's function at every iteration. While possible, performing $O(N^2)$ work, where N is the number of particles, per iteration would be prohibitive in meaningful applications.

If an exact solution is not viable, an approximate solution must be sought. For an elliptic problem, the primary sources of error will be associated with the derivative approximation (e.g., order of accuracy) and the particle arrangement. Since many meshfree approaches exist which can formulate discrete derivatives to high order [10, 17], the focus here will be on reducing particle disorder. Taking a cue from the field of atomic physics, it is observed that many solid materials are arranged in ordered lattices at the atomic level. It is the hypothesis of this work that emulating those physics on particles in a meshfree simulation can reduce error in the approximations to PDE solutions by enabling particles to be arranged in a more advantageous layout.

This work will explore the Molecular Dynamics (MD) [21] framework as a field which can supply an approximate solution to the particle arrangement problem. MD models the interactions between atoms as a short range interaction, often of a pairwise nature between nearby particles. It is both efficient and scalable [42], and relevant to the present problem because it can reproduce the ordered atom lattices observed in physical systems. For the purposes of this work, it contains two components which will be useful: energy functions which represent interatomic interactions as sums of pairwise interactions, and minimization algorithms which reduce the energy to its lowest level to identify equilibrium (with respect to forces) atomic configuration.

To test the hypothesis, MD approaches will be used to move particles by considering the interatomic potential as a surrogate for the error in the PDE solution. If correct, by moving the particles to their equilibrium configuration as identified by MD, the resulting approximation to the PDE should be more accurate. Because of the locally approximated error surrogate, the cost per iteration will be only $O(N)$, which is much more tractable for large problems. To investigate the usefulness of this approach, several possible MD functions, and the methods by which they can be minimized, will be presented in Section 5.2. The equation sets and associated discretization strategies will be described in Section 5.3. In Section 5.4, these two pieces will be combined and evaluated on how accurately they can solve elliptic PDEs. Some concluding thoughts will be offered in Section 7.

5.2 Error Surrogate Function

One of the primary sources of error in particle methods is disorganized arrangements of particles. Many methods have been considered to correct integral and derivative approximations when particle locations lack the structure commonly found associated with nodes in mesh-based methods. Here we consider an alternative approach: moving particles in order to minimize numerical errors, which naturally captures configurational error as well as errors associated with integral evaluation. The approach taken is to emulate molecular dynamics (MD) because solutions for particle arrangements using this method are often regular.

The function considered for the error surrogate is the commonly used Lennard-Jones (LJ) potential

$$e_{ij} = \varepsilon \left[\left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right]. \tag{5.1}$$

This function is repulsive at short range ($r_{ij} < 2^{1/6}\sigma$) and attractive thereafter. The magnitude of ε determines the size of the gradient, but if ε is uniform across all particles, then its value does not affect the final configuration (but can affect performance of minimization algorithms). Figure 5.1 shows the value of this function over distance.

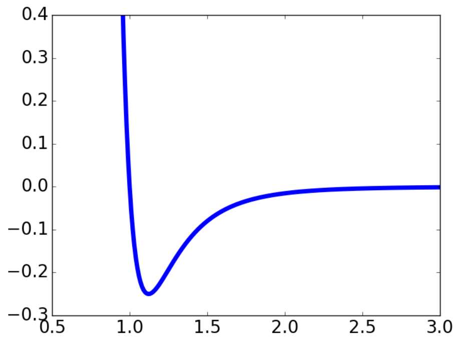


Figure 5.1: The Lennard-Jones potential with $\varepsilon = 1$ and $\sigma = 1$. The well-depth is $\varepsilon/4$ and its location is $2^{1/6}\sigma$.

LJ potentials are used to approximate the interactions of many types of systems, including solids and liquids. For the purposes here, its use to represent the ordered arrangement of solid atoms is relevant, suggesting the length scale used be near the average inter-particle spacing. Within the MD community, many other potentials of varying complexity exist. It is possible that other potentials might better reduce error, although we performed some preliminary calculations using other potentials with the LAMMPS [42] code, the PDE solutions did not significantly vary. Therefore, we have pursued the LJ potential given its effectiveness at reducing error associated with particle arrangements, as will be shown, as well its simplicity and computational efficiency.

A large body of experience exists with using the LJ potential to model solids. In the present application for elliptic problems, the particle configuration will not move based on the PDE solution. Instead, the LJ error surrogate must be minimized prior to solving the PDE. MD codes use conjugate gradient (CG) [49] methods with line search (the Polak-Ribière formulation was used here) to solve this non-linear optimization problem. In general, the CG algorithm computes the search direction without the use of additional parameters, but often the line search algorithm requires some heuristics to effectively converge. It this application, the backtrack line search method was found to be effective if the maximum step size was limited less than the mean interparticle distance. The reason is that for the random initial conditions, some particles start very close and as a result have a very large LJ interaction. Slowly relaxing such an initial condition is required for convergence.

Minimizing an LJ error surrogate can also be useful for non-linear and time-varying equations. In a non-linear equation, it is likely that a homogeneous distribution of particles will no longer be optimal. In that case, the LJ length scale can vary (indeed, it can be an arbitrary scaling and rotation tensor) based on local error estimates from the PDE to perform a variable resolution approach, which has been previously performed using domain decomposition [6]. In such a case, an iterative solution between the PDE and the particle positions can be used to adapt the particle configuration to the problem. For parabolic PDEs, the minimization procedure can be performed as a pre-solution step, after which the time-dependent PDE can be solved. If the problem contains some particles which must advect, for example in the case of particles representing a moving interface, the remaining particles' positions can be minimized after each advection step. The result is a quasi-static particle configuration driven by the particles whose motion must be physical.

5.3 Numerical Method

The Poisson equation used in this work is

$$\nabla^2 u(\mathbf{x}) = f(\mathbf{x}) \tag{5.2}$$

on the finite domain Ω subject to Dirichlet boundary conditions on the boundary $\partial\Omega$. The solution in space, $u(\mathbf{x})$ is forced by a spatially varying source term $f(\mathbf{x})$. Meshfree discretizations begin by convolving the PDE with a kernel function. In this work for simplicity, a piecewise constant kernel is used. However, it is emphasized that the results of the method should be applicable to most meshfree methods because error associated with particle disorganzation is ubiquitous [], and this one is merely chosen as an example for convenience. Upon convolution with a constant function which has support over the volume of space $V_\mathbf{x}$ with boundary $S_\mathbf{x}$ having unit normal $\mathbf{n}$, and one application of integration by parts, Eq. (5.2) becomes:

$$-\int_{S_\mathbf{x}} \nabla u(\mathbf{x}) \cdot \mathbf{n} \, dS = \int_{V_\mathbf{x}} f(\mathbf{x}) \, dV. \tag{5.3}$$

The above equation is still defined everywhere in space and requires spatial integrals to be evaluated at each location. In order to discretize the equations, volume integral is approximated as the value at the ith particle multiplied by its associated volume, i.e.,

$$\int_{V_{\mathbf{x}_i}} f(\mathbf{x}_i) \, dV \approx f(\mathbf{x}_i)\Delta V_i \tag{5.4}$$

for a finite number of particles, with $\Delta V_i = \int_{V_{\mathbf{x}_i}} dV$. A Voronoi tesselation, commonly used to analyze atomic configurations from MD (as well as in the recent approaches on meshfree particle distributions cited above [15, 13]), on the particles is computed which provides this volume, as well as connectivity with neighbor volumes defined by shared faces (the code in this work uses the voro++ package for this [46]). Surface integrals are discretized by evaluating the function at the surface centroid and multiplying by the surface's area. In this case, at a surface quadrature point S_{ij} can be approximated as

$$\nabla u|_{ij} \approx \frac{u_i - u_j}{\mathbf{x}_i - \mathbf{x}_j} \tag{5.5}$$

because each surface is defined by a pair of particles consisting of the i particles and one of its neighbors j. Normal vectors at these locations are merely the unit vector from $\mathbf{x}_i$ to $\mathbf{x}_j$, so the face integral can be approximate as

$$-\int_{S_{\mathbf{x}}} \nabla u(\mathbf{x}) \cdot \mathbf{n}\, dS \approx \sum_j \Delta S_{ij} \frac{u_i - u_j}{|x_i - x_j|}. \tag{5.6}$$

When fully discretized, the PDE is approximated by a linear system

$$\sum_j \Delta S_{ij} \frac{u_i - u_j}{|x_i - x_j|} = f(\mathbf{x}_i)\Delta V_i \tag{5.7}$$

for the unknown solution u_i for all particles. To solve the linear system, the Trilinos package [25] is used to provide linear solvers and preconditioners. For the systems of equations in this work, the GMRES algorithm using an ILUT preconditioner was found to be robust and able to converge the linear system to a relative tolerance of $1.e-8$. The same solver set was used for each solution shown in the next section.

5.4 Poisson Equation Results

When solving elliptic equations such as the Poisson equation, it is desirable to have a uniform and orthogonal set of points. Uniformity arises from the linear solver such that points at any location affect the solution everywhere else, as can be shown using a eigenvector decomposition of the Poisson matrix (for some cases in which the eigenvectors contained only a subset of particles, or for hyperbolic problems, non-uniformity can be advantageous and will be the subject of future work). The orthogonal set of points facilitates accurate discrete derivative operators. Results from the ideal orthogonal particle arrangement and a random arrangement are shown in Fig. 5.2. Both sets discretize the interior domain with 100 particles and each boundary edge with 10 particles, while both formulations have a particle at each corner to specify the geometry. Constant Dirichlet boundary conditions are used around the edge, while a forcing function

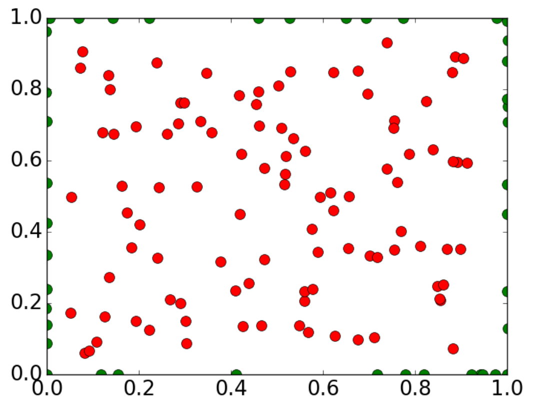
$$f(x,y) = \sin(5\pi x) + \sin(5\pi y)$$

is used to generate a non-trivial solution.

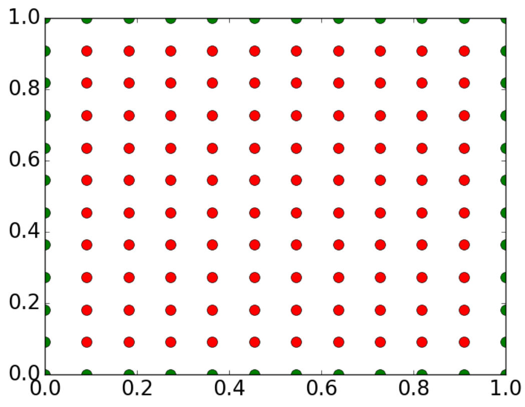
Visually, the results in Fig. 5.2 show that the solution with randomly arranged particles contains significantly larger errors than that with orthogonally arranged particles. This observation is made precise by comparing the solution against a highly refined solution using 10000 interior and 404 boundary particles in the orthogonal configuration. The L_2 error between the computed and refined solutions is 0.634 for the random arrangement and 0.166 for the orthogonal arrangement. Since the underlying method and software is identical, the error is solely attributable to particle arrangement.

The observations in the previous paragraphs are not new, but present the context for the use of the proposed method for elliptic equations. For the first example, considering minimizing the LJ error surrogate for the random initial configuration in Fig. 5.2 using the CG method described in Section 5.2. As noted previously, the use of the LJ surrogate introduces three new parameters: the length scale, which is set to the average inter-particle distance (a sufficient value to produce well-organized configurations that effectively eliminates one parameter), the cut-off distance which is set to 2.5 times the length scale such that the truncated interactions are small, and and error level. Minimized configurations are independent of the error level value, but it can influence convergence of the CG algorithm. Empirically it was set to $1e-2$ in this work. The minimization algorithm converged to a $1e-12$ tolerance after 290 iterations. The initial and final configurations, as well as some intermediate snapshots, are shown in Fig. 5.3.

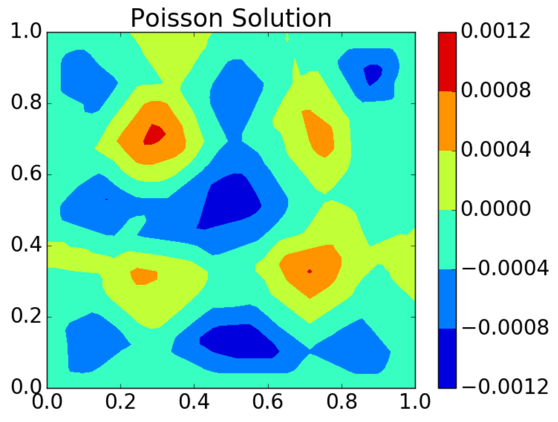
To assess the usefulness of the method, it must be determined how the actual error changes as the surrogate function is minimized. Figure 5.4 presents the solution for each configuration and its error relative the refined solution, showing indeed the solution does improve. The surrogate



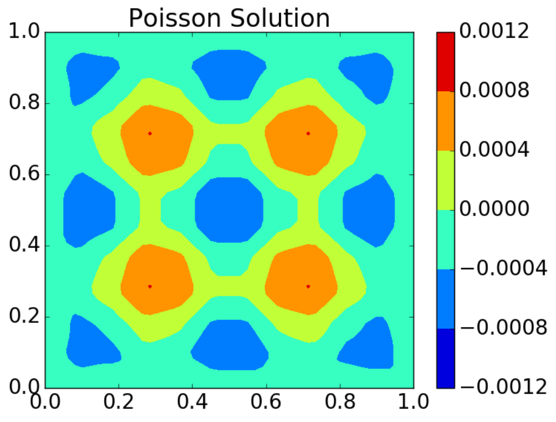
(a)



(b)



(c)



(d)

Figure 5.2: Comparison of using a) randomly and b) orthogonally arranged particles (red are interior particles, green are boundary particles) to solve a Laplace equation: c) the solution on the random particles (relative error: 0.634), d) the solution on the orthogonal particles (relative error: 0.166).

and error are correlated in this range, which is the reason this method is affective. However, the correlation is not perfect; the initial configuration generates a relative error in the solution approximately 4 times higher than the final configuration, the error surrogate is 14 orders of magnitude higher. As the minimum error surrogate configuration is approached, the gains saturate around the level of the orthogonal configuration. These observations demonstrate the limits of the method, but within a certain range, the error surrogate is an effective approximation to the relative L^2 error.

An important property of discretization approximations to PDEs is that they converge to the true solution as the discretization is refined; in this case as more particles are added. Therefore, additional arrangements consisting of mean inter-particle spacing of $2X$ and $4X$ the original configuration are considered. For each configuration, the particles where initially distributed randomly and minimized as previously described. The only change was to modify the LJ length scale consistently with the inter-particle spacing. Figure 5.5 presents the original and final particle configurations.

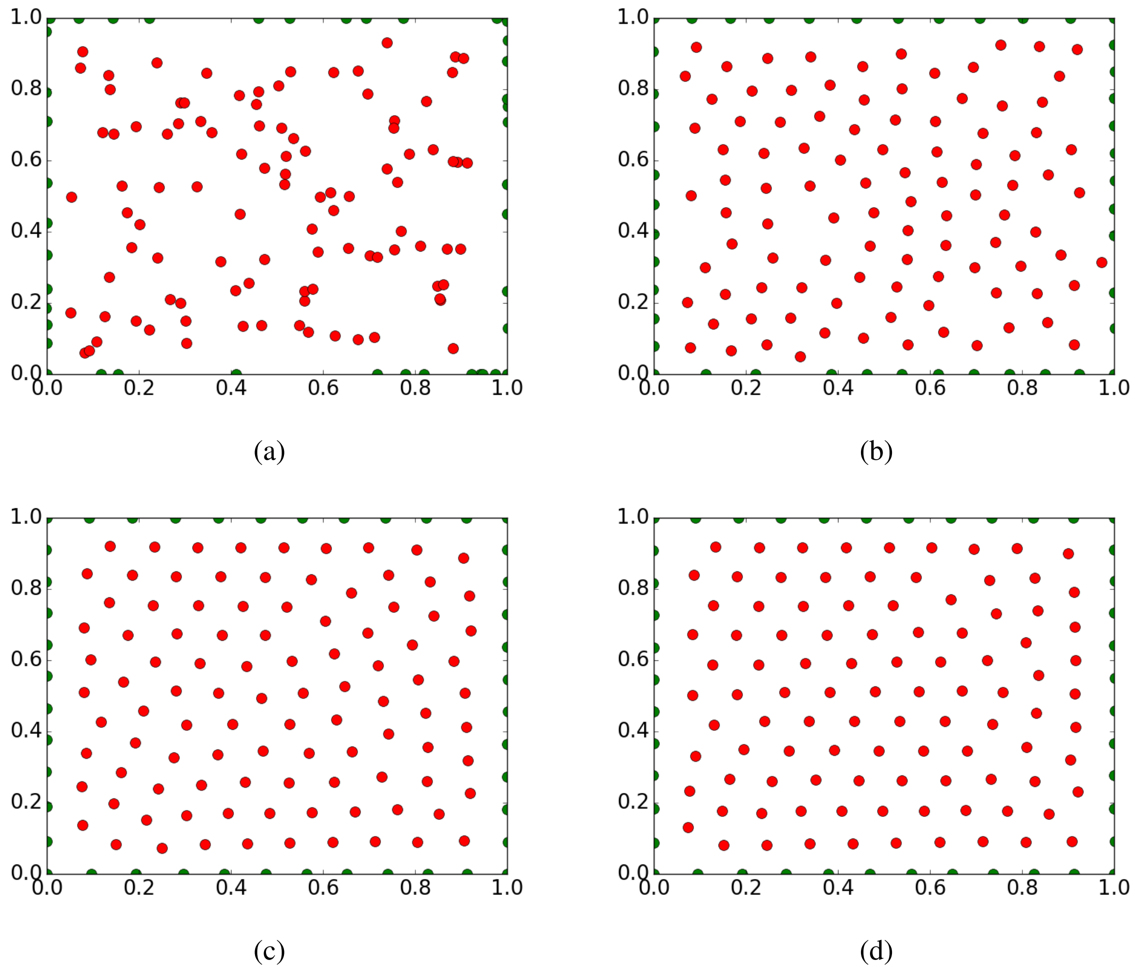


Figure 5.3: Particle arrangements during the minimization procedure (red are interior particles, green are boundary particles): a) initial random configuration, b) after 50 iterations, c) after 100 iterations, d) converged configuration (290 iterations).

While different random configurations will lead to different final configurations, the results are representative of the improvements likely to be obtained. The initial $2X$ particle layout results in a relative error of 0.162, which is reduced to 0.044 after minimization. Similarly, the $4X$ particle layout has a relative error 0.117 while post-minimization, the relative error is 0.015. In all cases, reducing the error surrogate made the solution at least as accurate as a four times more expensive configuration of random particles. Finally, it is noted that the configurations generated above were also used to solve a transient heat equation. In that application, because there is no advection, the error surrogate minimization step serves as a pre-processor for the simulation. When used in this mode, the results from the heat equation are virtually identical to those reported above.

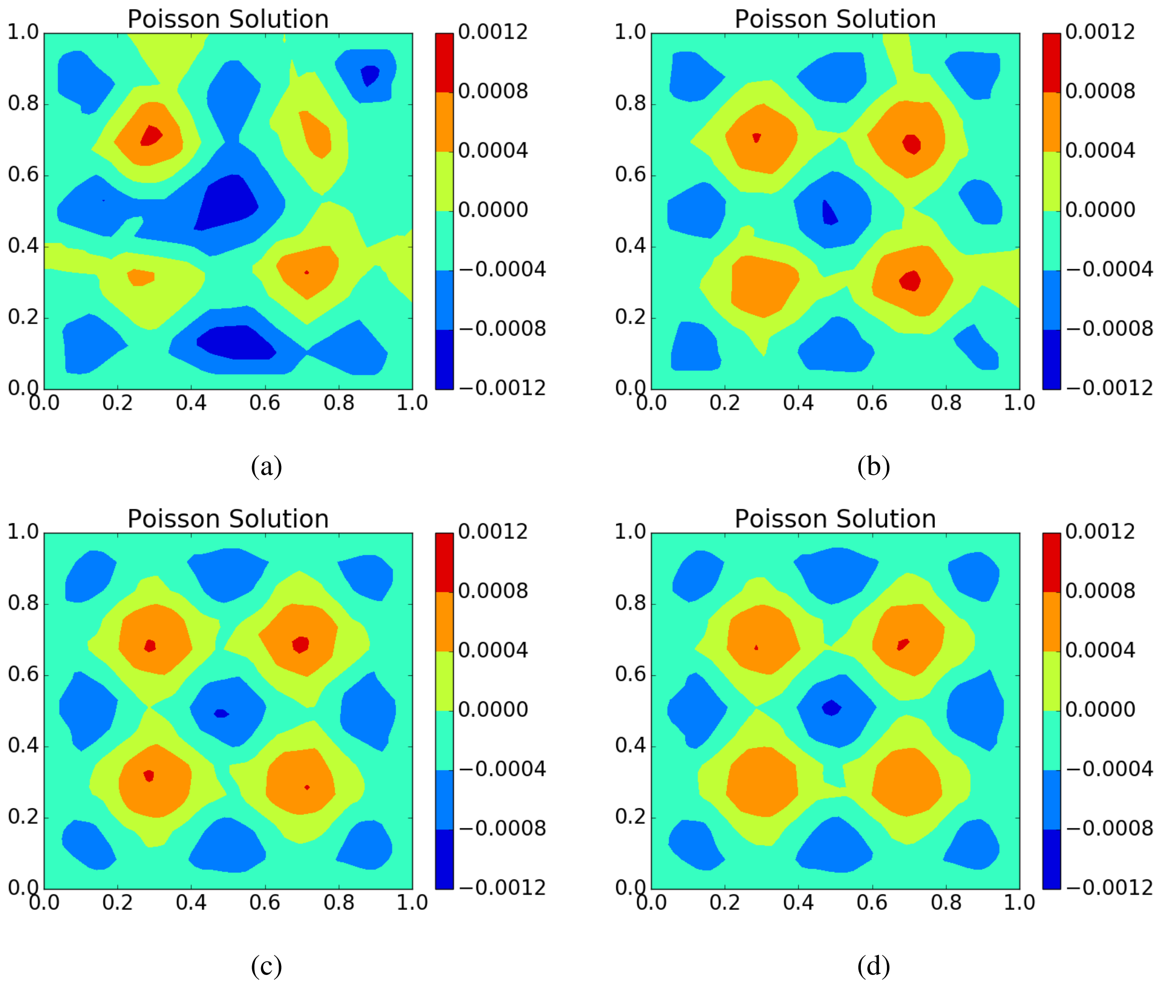


Figure 5.4: Solution for the particle configurations in Fig. 5.3: a) 0.634 relative error, b) 0.210 relative error, c) 0.160 relative error, d) 0.160.

5.5 Conclusions

This work has explored the idea of using functions and techniques inspired by MD to prescribe the location of particles within a meshfree method. While there are no expectations that these methods will produce the minimal possible error when the particle arrangement is used as the basis for solving PDEs, the empirical evidence presented herein shows that they can significantly reduce the error in meshfree schemes. The reason is that the errors associated with disordered particle arrangements is mitigated, which is expected because MD simulations of solids typically produce ordered particle arrangements consistent with ordered atom arrangements in real materials. Because particle disorder is known to introduce error in meshfree methods due to the difficulty discrete differential operators to represent the associated analytical operators.

The first results considered the use of the standard LJ potential, which is a commonly used

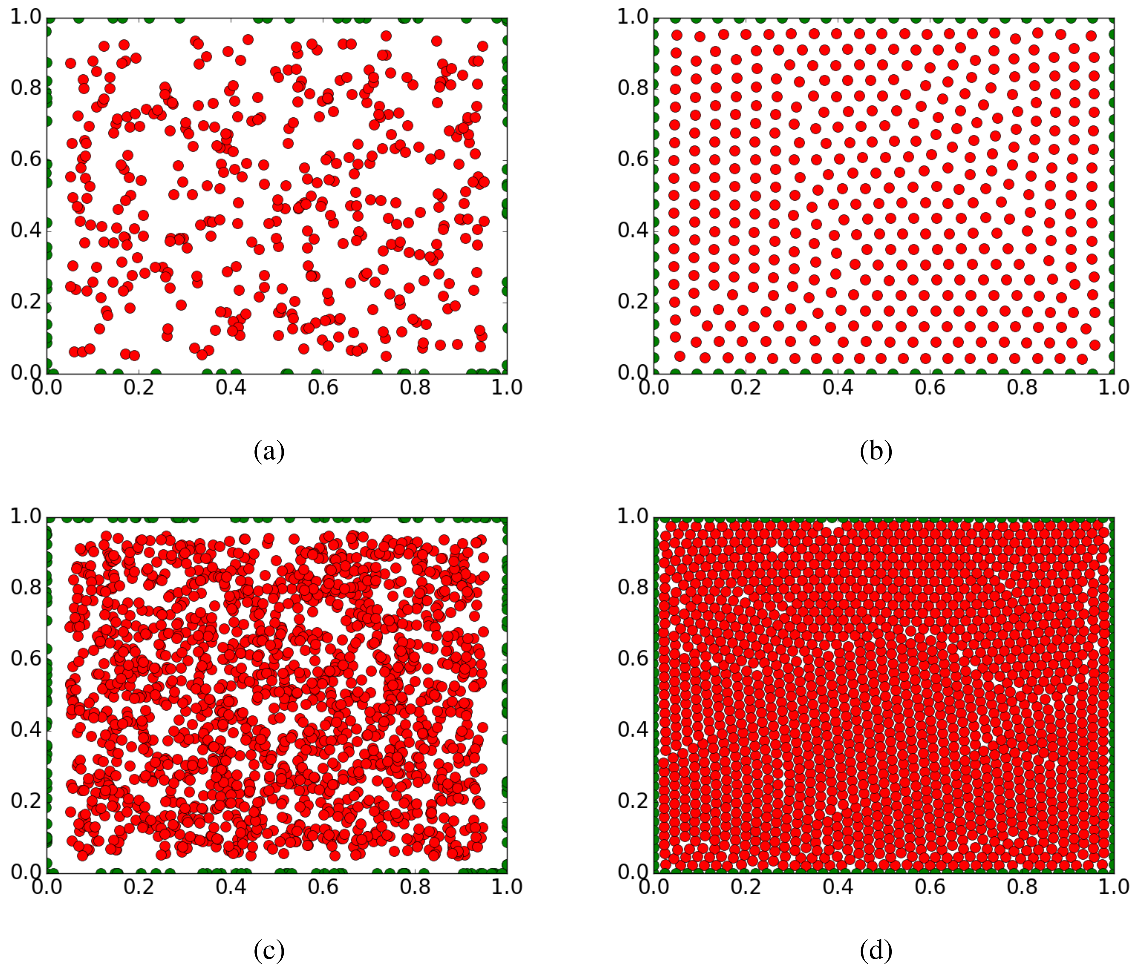


Figure 5.5: Initial and minimized configurations for increasing numbers of particles: a) $2X$ initial configuration, b) $2X$ minimized configuration, c) $4X$ initial configuration, d) $4X$ minimized configuration.

interatomic potential that can represent both solids and liquids. Using it requires the introduction of two parameters, a length scale and an error scale. The length scale can be readily set using the average inter-particle distance, while the error scale does not affect the final configuration and should be considered as a parameter for the CG minimization algorithm. For the equations used in this work, homogeneous values for both were optimal, although other types of equations may benefit from changing the length scale inhomogeneously and anisotropically to better resolve regions of high gradients or other complex effects. Results for the Poisson equation showed that the minimized configuration resulted in substantially less error in the solution as opposed to a random configuration. Hence, the LJ potential can be a useful error surrogate function.

As there is no reason to expect the LJ potential to be a particularly optimal form, several different parameterizations of both it and the centro-symmetry function were examined. The results suggest that while the different functions produce slightly different arrangements and different errors, the error in the solution is always well below that of a random arrangement. This implies that the results are relatively independent of the choice of error function assuming the function favors ordered particle configurations. However, some caveats are required. First, the error surrogates are only correlated with the error globally, not locally, and only over a regime near the minimum error which can be obtained by the functions. A more highly ordered, orthogonal particle layout, which will not be realized using these functions, produces less error. Thus, a direction for future research would be to find functions which favor this kind of layout without increasing the expense of the minimization algorithm.

Despite the above caveats, using error surrogate functions to position particles in meshfree method appears to be an efficient error control mechanism. For the surrogates considered here, the only data structure required is a neighbor list, which likely already exists in a meshfree code. As demonstrated here, the CG and line minimization algorithms widely used in MD are sufficient and are available from many open source packages. For elliptic equations, minimizing the error surrogate can be performed as a pre-computation step. While the expense does add to the cost of a computation, the results demonstrate that even a few iterations of the CG algorithm produce noticeable error reductions in the PDE solution. When used with time-dependent PDEs, particle configuration minimization can be performed at each step at modest cost after the first step, and like MD, potentially need only be performed every few time steps. Therefore, using an error surrogate function based on MD offers a straightforward way to reduce the error when using a meshfree method.

Chapter 6

Solid-to-Liquid Phase Change

6.1 Introduction

In the melting metal application space, we care about accurately modeling the solid-to-liquid phase transition and related dynamics. One of the major challenges to phase change computations is how to handle the jump condition at the melt front interface when the interface location itself is unknown a priori and is inherent to the solution of the problem. More traditional approaches such as the enthalpy method and the heat source method treat the phase transition by smearing out the interface and avoid resolving it as a sharp interface [45]. It should be noted that both of these approaches make the assumption that the density in the solid and the liquid phases are the same, which is too severe of a limitation for our simulation needs.

The heat transfer equations for temperature T in the solid and liquid regions are

$$\rho_s c_p \frac{\partial T}{\partial t} = \nabla \cdot (k_s \nabla T) + f_s \tag{6.1}$$

$$\rho_f c_p \frac{\partial T}{\partial t} = \mathbf{u}_f \cdot \nabla T + \nabla \cdot \left(k_f \nabla T\right) + f_f \tag{6.2}$$

where ρ is the density, c_p is the specific heat, $\mathbf{u}$ is the fluid velocity (the solid region is assumed to be stationary), k is the diffusion coefficient, and f is any external source terms (heat fluxes). The heat equation for the liquid and solid regions can be solved for separately with an interfacial boundary condition (the Stefan condition for heat balance) or monolithically by using a Heaviside function for velocity such that it is zero in the solid domain and by smoothly varying the diffusion and density parameters that depend on the region they are solved. When this approach is taken, one can back out the location of the interface after a temperature solve and implicitly setting the location to be the $T = T_{melt}$ temperature contour as the interface (see Figure 6.1). For example, in the heat source approach, the interface is implicitly defined at the melt temperature and a smoothed out Dirac delta function is used to account for the jump in heat flux is placed at the interface when solving the system of equations. In the enthalpy method, the conservative form of enthalpy is used instead of directly solving for the temperature, and the jump is removed by integration. The heat of formation then gets lumped into the specific heat, which then depends on temperature and has a spike near the melt temperature.

If we take a more accurate approach and solve the solid and liquid regions separately, we need to impose a heat balance boundary condition at the the phase change boundary. This heat balance relation is called the Stefan condition, where we have the requirements that

$$T_f|_X = T_s|_X \tag{6.3}$$

$$\rho L \frac{\partial X}{\partial t} = k_f \frac{\partial T_f}{\partial n}\bigg|_X - k_s \frac{\partial T_s}{\partial n}\bigg|_X \tag{6.4}$$

where $X(t)$ is the spatial position of the phase change boundary, T_f and T_s are the temperature fields for the liquid and solid phases, respectively, ρ is the density of the material, L is the latent heat. The Stefan condition provides us with an equation for determining the phase boundary's velocity. The boundary condition for the solid and liquid regions can use a Dirichlet boundary condition $T = T_{melt}$.

In order to relax the restriction that the density between the solid and liquid are distinct, the problem becomes more complex. A normal mass balance for the velocity normal to the interface must be solved for

$$\rho_f \mathbf{n} \cdot (\mathbf{u}_f - \mathbf{u}^*) = -\rho_s \mathbf{n} \cdot \mathbf{u}^* \tag{6.5}$$

where $\mathbf{u}^* = \frac{\partial X}{\partial t}$. This velocity condition requires us to solve explicitly for the phase change velocity and to resolve the phase change boundary during the computational solve.

There do exist alternative approaches to solving the solid-to-liquid phase transition problem explicitly using a sharp interface technique. For example, in [52] they use a finite difference approximation with stencil readjustments and demonstrate second order accuracy for the temperature field solve and first order accuracy for the interface location.

6.2 Numerical Approach

For our approach, we track the interface explicitly and treat the interface as a discontinuity in the heat flux. The interface is discretized into particles that hold the melt temperature for all time and

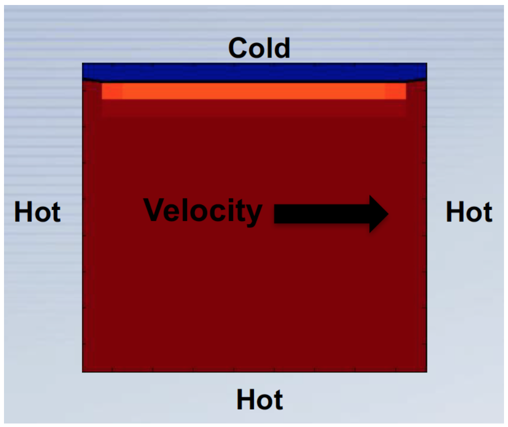


Figure 6.1: Implicitly tracking the interface by solving the heat equation on the full domain and using the resulting melt-front temperature to find the location of the melt front.

move with the normal speed of the interface. We directly solve for the interfacial velocity $\frac{\partial X}{\partial t}$ of the melting front using the Stefan condition

$$\rho_s L \frac{\partial X(t)}{\partial t} = k_f \frac{\partial T}{\partial n}|_X - k_s \frac{\partial T}{\partial n}|_X \tag{6.6}$$

where ρ_s is the density of the solid, L is the latent heat, k_f and k_s are the diffusion coefficients for the fluid and solid phases, respectively.

We explicitly track the interface with a set of particles on the interface itself that move with the interfacial velocity and hold the melt temperature for all time (the heat equation is not solved on the interface). The interface is then used to solve the heat equation in the solid and liquid regions, using the melt front location and melt temperature as a Dirchlet boundary condition. The interface is projected forward in time using a backward Euler method (1st and 3rd order). The heat fluxes are computed using our RKPM and Kernel-based approaches for the fixed RKPM method for the Laplacian operator in the heat equations and one-sided gradient operators (for the flux at the boundary in the liquid and solid domains) for the particle velocity solution. These calculations are performed on the set of nodes that either make up the solid interior or the liquid interior, and including the interface particles in both solves.

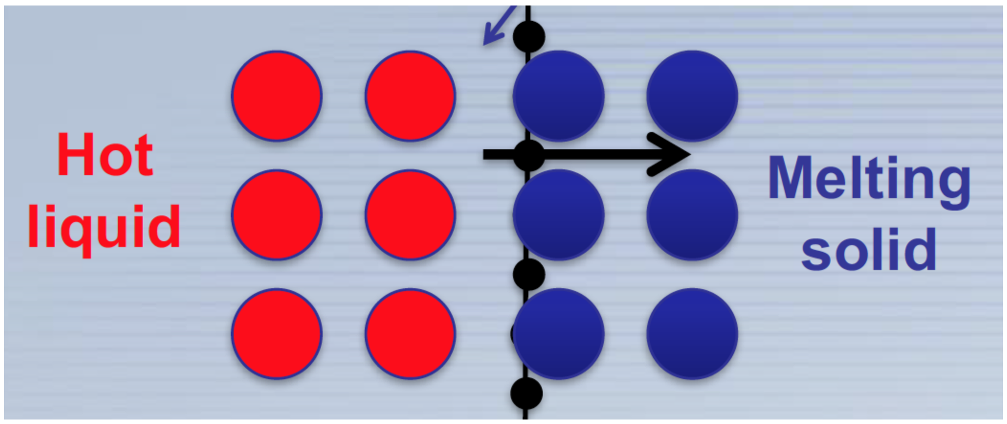


Figure 6.2: Illustration of melt-front interface tracking: the liquid region is solved for on the set of red “interior particles” and the solid region is solved for on the blue set of“interior particles.” The black set of particles represent the interface and their motion is governed by the Stefan condition. The set of black particles is used as a marker to switch the interior particles from solid to liquid or liquid to solid.

6.3 Results

Figure 6.3 and Figure 6.4 show an example simulation of a phase boundary set of particles passing through background nodes and the temperature solve on the liquid and solid domains. We impose Dirichlet boundary conditions at both ends of the domain $T(0) = 500$ on the LHS for the liquid and $T(1) = 380$ for the solid portion of the domain. The phase boundary is initially placed at

$X(0) = 0.06$ instead of at $X(0) = 0$ since Moab is not yet able to handle a new phase that is not initially set up (the liquid phase in the domain must be initialized with a non-zero number of interior particles). As the background nodes get passed over by the phase boundary interface particles, a group change occurs on the background or interior particles, where these nodes change from the liquid group set to the solid group set within our Moab framework and then get handled within their respective solves.

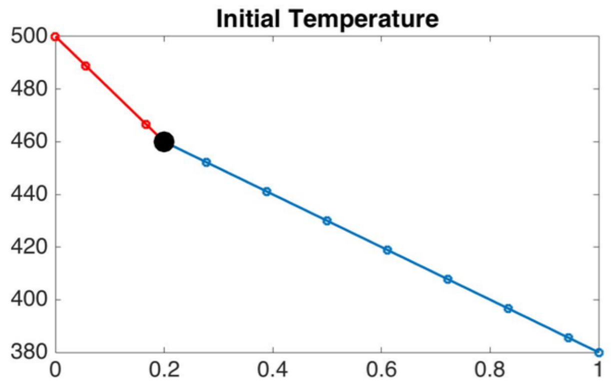


Figure 6.3: Temperature profile for the melt problem.

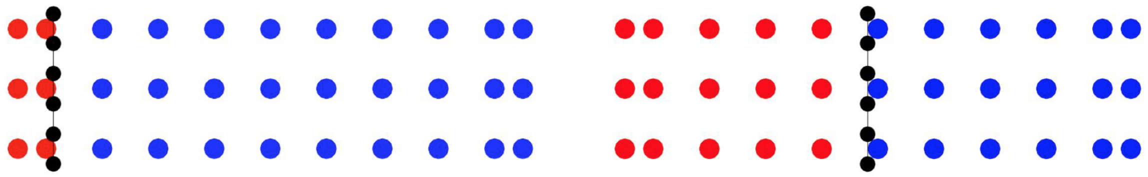


Figure 6.4: As the interface moves past the red liquid nodes, they change from being part of the liquid solution to part of the solid solution within Moab, by making a "group" change.

6.4 Future Directions

We have described a new approach to solving the phase transition problem using the Stefan condition. By explicitly tracking the melt front and solving for the particle velocity and location explicitly, we are able to perform calculations on materials that have different densities in their solid and liquid forms. This gives our approach an advantage over more traditional methods, such as the heat source or enthalpy methods. A formal convergence study is needed to quantify this method's accuracy. Combining this interface-tracking approach with the adaptive particle motion procedure for minimizing error (Chapter 5) by adjusting the interior particles for the liquid and solid temperature solves will improve the accuracy of these calculations.

Chapter 7

Conclusion

This report has described the development of several capabilities to move the state of the art forward towards predicting complex, low Reynolds number flows with phase change using meshfree methods. Two primary areas were explored. First, quantifying and minimizing numerical errors in a meshfree scheme was investigated by developing new adaptive quadrature methods and moving internal particles according to an error surrogate function. Second, accurate and efficient interface tracking was developed using an extension of the particle level set coupled with interface particles which drive the topology of the internal particles. The synthesis of these two key areas lays the groundwork for a meshfree simulation approach targeting mission-relevant problems.

Codifying these developments is a new software package, called MOAB, which supports the definition of kernel functions, their corrections, differential equations, and particle motion. MOAB is written in C++ on top of the Trilinos [] package of linear and non-linear solvers. While not designed for particle methods, Trilinos enables the governing equations to be readily solved following interface motion, updated particle positions, and adapted quadrature schemes. Many meshfree methods for fluids do not form complex matrix equations, so this is a distinguishing feature of MOAB.

MOAB is well-documented using Doxygen as a basis, and functionality is tested with nearly 100 tests thus far. In addition, more complex examples illustrating solution of relevant PDEs are included. In order to be more widely useful, MOAB is structured as a series of packages which build upon each other using a library approach. A MOAB application then pulls from these libraries to construct and solve the relevant linear or non-linear equations. Building problems is facilitated using a dependency management framework so that most of the needed calculations are performed without being explicitly called by the developer. Because MOAB uses the most recent features of Trilions, such as Tpetra, Kokkos, and Zoltan2, it should be portable to a wide range of current and emerging machines.

The MOAB package is designed to be used to prototype solutions to Sandia problems in which the meshfree framework is necessary or advantageous. These include assessing safety in abnormal thermal environments, qualifying parts produced with additive manufacturing, and modeling nuclear reactor core meltdowns, to name a few. Developing the necessary models, and taking them through the V&V process, will be within the capability range of the code and help drive forward the ability to simulate these challenging problems. As such, MOAB and the methods within it represent a new way to approach several important problems in Sandia's mission areas.

References

[1] David Adalsteinsson and James A Sethian. A fast level set method for propagating interfaces. *Journal of computational physics*, 118(2):269–277, 1995.

[2] N.R. Aluru. A point collocation method based on repro cing kernel approximations. *International Journal of Numerical Methods in Engineering*, 47:1083–1121, 2000.

[3] NR Aluru. A point collocation method based on reproducing kernel approximations. *International Journal for Numerical Methods in Engineering*, 47(6):1083–1121, 2000.

[4] NR Aluru and Gang Li. Finite cloud method: a true meshless technique based on a fixed reproducing kernel approximation. *International Journal for Numerical Methods in Engineering*, 50(10):2373–2410, 2001.

[5] I. Babuška, U. Banerjee, J.E. Osborn, and Q. Zhang. Effect of numerical integration on meshless methods. *Computer Methods in Applied Mechanics and Engineering*, 198:2886–2897, 2009.

[6] X. Bian, Z. Li, and G.E. Karniadakis. Multi-resolution flow simulations by smoothed particle hydrodynamics via domain decomposition. *Journal of Computational Physics*, 297:132–155, 2015.

[7] Y. Bourgault, M. Picasso, F. Alauzet, and A. Loseille. On the use of anisotropic a posteriori error estimators for the adaptative solution of 3D inviscid compressible flows. *International Journal of Numerical Methods in Fluids*, 59:47–74, 2009.

[8] Jiun-Shyan Chen and Youcai Wu. Stability in Lagrangian and semi-Lagrangian reproducing kernel discretizations using nodal integration in nonlinear solid mechanics. In *Advances in Meshfree Techniques*, pages 55–76. Springer, 2007.

[9] Jiun-Shyan Chen and Youcai Wu. *Stability in Lagrangian and Semi-Lagrangian Reproducing Kernel Discretizations Using Nodal Integration in Nonlinear Solid Mechanics, eds. Leitão, V. M. A. and Alves, C. J. S. and Armando Duarte, C.*, pages 55–76. Springer Netherlands, Dordrecht, 2007.

[10] J.S. Chen, M. Hillman, and M. Rüter. An arbitrary order variationally consistent integration for galerkin meshfree methods. *International Journal of Numerical Methods in Engineering*, 95:387–418, 2013.

[11] Rongjun Cheng and K. M. Liew. The reproducing kernel particle method for two-dimensional unsteady heat conduction problems. *Computational Mechanics*, 45(1):1–10, 2009.

[12] S.W. Chi, C.H. Lee, J.S. Chen, and P.C. Guan. A level set enhanced natural kernel contact algorithm for impact and penetration modeling. *International Journal for Numerical Methods in Engineering*, pages –, 2014.

[13] G. Chiaki and N. Yoshida. Particle splitting in smoothed particle hydrodynamics based on voronoi diagram. *Monthly Notices of the Royal Astronomical Society*, 451(4):3955–3963, 2015.

[14] L Cueto-Felgueroso, I Colominas, G Mosqueira, F Navarrina, and M Casteleiro. On the Galerkin formulation of the smoothed particle hydrodynamics method. *International journal for numerical methods in engineering*, 60(9):1475–1512, 2004.

[15] S. Diehl, G. Rockefeller, C. L. Fryer, D. Riethmiller, and T. S. Statler. Generating optimal initial conditions for smoothed particle hydrodynamics simulations. *Publications of the Astronomical Society of Australia*, 32, 12 2015.

[16] John Dolbow and Ted Belytschko. An introduction to programming the meshless Element Free Galerkin method. *Archives of Computational Methods in Engineering*, 5(3):207–241, 1998.

[17] Q. Du, R.B. Lehoucq, and A.M. Tartakovsky. Integral approximations to classical diffusion and smoothed particle hydrodynamics. *Computer Methods in Applied Mechanics and Engineering*, 286:216–229, 2015.

[18] D. Enright, R. Fedkiw, J. Ferziger, and I. Mitchell. A hybrid particle level set method for improved interface capturing. *Journal of Computational Physics*, 183(1):83–116, 2002.

[19] Douglas Enright, Frank Losasso, and Ronald Fedkiw. A fast and accurate semi-Lagrangian particle level set method. *Computers & structures*, 83(6):479–490, 2005.

[20] R. Fatehi and M.T. Manzari. Error estimation in smoothed particle hydrodynamics and a new scheme for second derivatives. *Computers & Mathematics with Applications*, 61(2):482–498, 2011.

[21] D. Frenkel and B. Smit. *Understanding Molecular Simulation: From algorithms to applications*. Academic Press, 2002.

[22] F. Gibou, L. Chen, D. Nguyen, and S. Banerjee. A level set based sharp interface method for the multiphase incompressible Navier-Stokes equations with phase change. *Journal of Computational Physics*, 222:536–555, 2007.

[23] R.A. Gingold and J.J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, 1977.

[24] D.I. Graphm and J.P. Hughes. Accuracy of SPH viscous flow models. *International Journal of Numerical Methods in Fluids*, 56(8):1261–1269, 2008.

[25] Michael Heroux, Roscoe Bartlett, Vicki Howle Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[26] S.E. Hieber and P. Koumoutsakos. A Lagrangian particle level set method. *Journal of Computational Physics*, 210:342–367, 2005.

[27] Cyril W Hirt and Billy D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of computational physics*, 39(1):201–225, 1981.

[28] S. Ianniello and A. Di Mascio. A self-adaptive oriented particles level set method for tracking interfaces. *Journal of Computational Physics*, 229:1353–1380, 2010.

[29] Guang-Shan Jiang and Danping Peng. Weighted ENO schemes for Hamilton–Jacobi equations. *SIAM Journal on Scientific computing*, 21(6):2126–2143, 2000.

[30] Shaofan Li and Wing Kam Liu. Meshfree and particle methods and their applications. *Applied Mechanics Reviews*, 55(1):1–34, 2002.

[31] Gui-Rong Liu and Y. T. Gu. *An Introduction to Meshfree Methods and their Programming*. Springer, 2005.

[32] Gui-Rong Liu and M. B. Liu. *Smoothed Particle Hydrodynamics: A Meshfree Particle Method*. World Scientific, 2003.

[33] Wing Kam Liu, Sukky Jun, Shaofan Li, Jonathan Adee, and Ted Belytschko. Reproducing kernel particle methods for structural dynamics. *International Journal for Numerical Methods in Engineering*, 38(10):1655–1679, 1995.

[34] Wing Kam Liu, Sukky Jun, Dirk Thomas Sihling, Yijung Chen, and Wei Hao. Multiresolution reproducing kernel particle method for computational fluid dynamics. *International Journal for Numerical Methods in Fluids*, 24(12):1391–1415, 1997.

[35] Wing Kam Liu, Sukky Jun, and Yi Fei Zhang. Reproducing kernel particle methods. *International journal for numerical methods in fluids*, 20(8-9):1081–1106, 1995.

[36] W.K. Liu, Y. Chen, S. Jun, J.S. Chen, T. Belytschko, C. Pan, R.A. Uras, and C.T. Chang. Overview and application of the reproducing kernel particle methods. *Archives of Computational Methods in Engineering*, 3(1):3–80, 1996.

[37] Xu-Dong Liu, Stanley Osher, and Tony Chan. Weighted essentially non-oscillatory schemes. *Journal of computational physics*, 115(1):200–212, 1994.

[38] L.B. Lucy. A numerical approach to the testing of fusion process. *The Astronomical Journal*, 88:1013–1024, 1977.

[39] MA Martinez, E Cueto, I Alfaro, M Doblare, and F Chinesta. Updated Lagrangian free surface flow simulations with natural neighbour Galerkin methods. *International Journal for Numerical Methods in Engineering*, 60(13):2105–2129, 2004.

[40] J.J. Monaghan. An introduction to sph. *Computer Physics Communications*, 48(1):89 – 96, 1988.

[41] Stanley Osher and Ronald Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. Springer, 2007.

[42] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117, 1995.

[43] Mahesh Prakash, Paul Cleary, and John Grandfield. Modelling of metal flow and oxidation during furnace emptying using smoothed particle hydrodynamics. *journal of materials processing technology*, 209(7):3396–3407, 2009.

[44] Sylvain Reboux, Birte Schrader, and Ivo F Sbalzarini. A self-organizing Lagrangian particle method for adaptive-resolution advection–diffusion simulations. *Journal of Computational Physics*, 231(9):3623–3646, 2012.

[45] Junuthula Narasimha Reddy and David K Gartling. *The finite element method in heat transfer and fluid dynamics*. CRC press, 2010.

[46] C.H. Rycroft. Voro++: A three-dimensional Voronoi cell library in C++. *Chaos*, 19, 2009.

[47] James A Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[48] S. Shao and E.Y.M. Lo. Incompressible sph method for simulating newtonian and non-newtonian flows with a free surface. *Advances in Water Resources*, 26:787–800, 2003.

[49] J.R. Shewchuck. An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/ quake-papers/painless-conjugate-gradient.pdf, 1994.

[50] John Strain. Semi-Lagrangian methods for level set equations. *Journal of Computational Physics*, 151(2):498–533, 1999.

[51] N. Trask, M. Maxey, K. Kim, M. Perego, M.L. Parks, K. Yang, and J. Xu. A scalable consistent second-order SPH solver for unsteady low Reynolds number flows. *Computer Methods in Applied Mechanics and Engineering*, 289:155–178, 2015.

[52] HS Udaykumar, Rajat Mittal, and Wei Shyy. Computation of solid–liquid phase fronts in the sharp interface limit on fixed grids. *Journal of computational physics*, 153(2):535–574, 1999.

[53] Salih Ozen Unverdi and Grétar Tryggvason. A front-tracking method for viscous, incompressible, multi-fluid flows. *Journal of computational physics*, 100(1):25–37, 1992.

[54] Yung-Ming Wang, Syuan-Mu Chen, and Chih-Ping Wu. A meshless collocation method based on the differential reproducing kernel interpolation. *Computational Mechanics*, 45(6):585–606, 2010.

[55] Zhaoyuan Wang, Jianming Yang, and Frederick Stern. An improved particle correction procedure for the particle level set method. *Journal of Computational Physics*, 228(16):5819–5837, 2009.

[56] Steven T Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. *Journal of computational physics*, 31(3):335–362, 1979.

[57] L.L. Zheng and H. Zhang. An adaptive level set method for moving boundary problems: application to droplet spreading and solidification. *Numerical Heat Transfer, Part B*, 37:437–454, 2000.

[58] Q. Zhu, L. Hernquist, and Y. Li. Numerical convergence in smoothed particle hydrodynamics. *The Astrophysical Journal*, 800, 2015.

DISTRIBUTION:

1	MS	0359	D. Chavez, LDRD Office, 1911
1	MS	0899	Technical Library, 9536 (electronic copy)

v1.40

Sandia National Laboratories