Smoothed Particle Hydrodynamics and Its Application Towards Extrusion Based Additive Manufacturing

by

Chang Yoon Park

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Mechanical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Summer 2019

Smoothed Particle Hydrodynamics and Its Application Towards Extrusion Based Additive Manufacturing

Abstract

Smoothed Particle Hydrodynamics and Its Application Towards Extrusion Based Additive Manufacturing

by

Chang Yoon Park
Doctor of Philosophy in Mechanical Engineering

University of California, Berkeley

Professor Tarek I. Zohdi, Chair

The possibility of using the mesh-free method Smoothed Particle Hydrodynamics (SPH) towards analyzing modern additive manufacturing techniques, extrusion based ones in particular, is investigated in this report. The mathematical foundations and a number of corresponding topics regarding SPH is introduced, ranging from incompressible SPH and implicit timestepping methods to total Lagrangian formulations: displaying its robust capabilities for solving multiphyiscal mechanical problems. Finally, we demonstrate a swarm-intelligence based optimization scheme coupled with our SPH solver to optimize extrusion topologies for DIW (Direct Ink Writing) applications.

To my loving wife, Sujung.

Contents

List of Figures **vi**

List of Tables **viii**

1 Brief Review of Continuum Mechanics **1**

1.1 Preliminary Mathematical Notations 2
1.1.1 Vectors, Scalars and Matrices 2
1.1.1.1 Vectors 2
1.1.1.2 Matrices 2
1.1.1.3 Basic Matrix Algebra 3
1.1.1.4 Eigenvalues and Eigenvectors 3
1.1.1.5 Determinant and Condition Numbers 4
1.1.2 Tensors 4
1.1.2.1 Definition 4
1.1.2.2 Einstein Notation 5
1.1.2.3 Tensor Calculus 5
1.2 Kinematics 6
1.2.1 Deformation Gradient 6
1.2.2 Jacobian 8
1.2.3 Decomposition of the Deformation Gradient 8
1.2.4 Strain 8
1.2.5 Stress Measures 9
1.2.5.1 Cauchy Stress Tensor 9
1.2.5.2 First Piola-Kirchhoff Stress Tensor 9
1.2.5.3 Second Piola-Kirchhoff Stress Tensor 9
1.2.5.4 Stress Rates 9
1.3 Balance Laws 10
1.3.1 Conservation of Mass 10
1.3.2 Balance of Linear Momentum 10
1.3.3 Field Equations: Balance of Linear Momentum 11
1.3.4 Field Equations: Thermodynamics 12

2 Topics in Smoothed Particle Hydrodynamics 13
2.1 Introduction to SPH 14
2.1.1 Concept 14
2.1.2 Operators 14
2.1.3 On Solving the Navier Stokes Equation with SPH 16
2.1.4 Boundary Condition Enforcement 18
2.1.5 Free-Surface Tracking 18
2.1.6 Particle Shifting 19
2.2 Advanced SPH Solvers 19
2.2.1 Review of Implicit Methods in SPH 20
2.2.1.1 Cummins, Rudman (1999) 20
2.2.1.2 IISPH - Implicit Incompressible SPH, Ihmsen et al (2014) 21
2.2.1.3 DFSPH - Divergence Free SPH (Bender and Koschier, 2017) 24
2.2.2 Viscosity Solvers 26
2.2.2.1 Takahashi et al. (2015) 27
2.2.2.2 Weiler et al. (2018) 28
2.3 Heat Transfer 28
2.3.1 Heat Equation 28
2.3.2 SPH Discretization 29
2.3.3 Ghost Particle Boundary for the Heat Equation 29
2.3.3.1 Method 29
2.3.3.2 Boundary Conditions 30
2.4 Thermal Stress occurring from Hot Droplet Impacting Surface 30
2.4.1 Introduction 30
2.4.2 Governing Equations 32
2.4.2.1 Navier-Stokes Equation for Newtonian Fluids 32
2.4.2.2 Thermo-Elasticity 34
2.4.3 Discretization 35
2.4.3.1 Navier-Stokes 35
2.4.3.2 Heat Equation 37
2.4.3.3 Finite Element Method 40
2.4.4 Numerical Algorithm 41
2.4.4.1 SPH-FEM 1-way Coupling 41
2.4.4.2 Neighbor Searching 41
2.4.4.3 Particle Interaction 42
2.4.4.4 SPH-FEM Coupling and Thermoelasticity 42
2.4.4.5 Initial Placement 42
2.4.5 Convergence Study 44
2.4.5.1 Heat Equation with Consistent Operators 44
2.4.5.2 SPH-FEM Momentum Coupling 45
2.4.6 Numerical Example 46
2.4.6.1 Thermal Droplet Impact Case 46

2.4.6.2 Fluid Properties . . . 49
2.4.6.3 Solid Properties . . . 49
2.4.6.4 General Parameters . . . 50
2.4.6.5 Analysis . . . 50
2.4.7 Discussion . . . 56
2.4.7.1 Summary . . . 56
2.4.7.2 Limitations . . . 56
2.5 Finite Deformation Elasticity: A Total Lagrangian Approach . . . 57
2.5.1 Introduction . . . 57
2.5.2 SPH Discretization in Current Configuration . . . 58
2.5.3 Total Lagrangian Formalism for Finite Deformation Thermoelasticity 59
2.5.3.1 Constitutive Relations . . . 59
2.5.4 Total Lagrangian SPH discretization for solid mechanics . . . 61
2.5.4.1 The choice of divergence operators . . . 62
2.5.5 Implicit Timestepping . . . 63
2.5.6 Numerical Expamples . . . 65
2.5.7 Transient Heat Transfer on an Expanding Rod . . . 67
2.5.8 Expansion Induced Curvature in a Free Standing Bimaterial . . . 69
2.5.9 Analysis of a Metamaterial Structure with Negative Thermal Expansion . . . 72
2.5.10 Discussion . . . 72
2.6 Non-Newtonian Fluids . . . 73
2.6.1 Introduction . . . 73
2.6.2 Problem Formulation . . . 73
2.6.3 Viscous Forces: Semi-Implicit Time Stepping . . . 75
2.6.4 Boundary Conditions . . . 77
2.6.5 Numerical Example: Herschel-Bulkley 2D Poiseuille Flow . . . 78
2.6.6 Numerical Results . . . 80
2.6.7 Discussion . . . 82

3 Computational Optimization of Nozzle Extrusion Paths in Ink Writing Manufacturing Procedures 83
3.1 Introduction . . . 84
3.2 Modeling of Inks . . . 86
3.2.1 Mechanics of Inks . . . 86
3.2.2 Numerical Method - Smoothed Particle Hydrodynamics (SPH) . . . 88
3.2.2.1 Brief Introduction . . . 88
3.2.2.2 Discretization of the Navier-Stokes Equation . . . 89
3.2.2.3 Solution of the Pressure Poisson Equation (Incompressible SPH) . . . 90
3.2.2.4 Implicit Viscosity Solver . . . 93
3.3 Computational Optimization of Extrusions in DIW . . . 94

3.3.1 Overview . . . 94
3.3.2 Waypoint generation . . . 95
3.3.3 Particle Swarm Optimization (PSO) . . . 98
3.4 Optimization Examples . . . 100
3.4.1 90 Degree Bend . . . 101
3.4.1.1 Non-Optimized Extrusion Case . . . 101
3.4.1.2 Optimized Extrusion Case . . . 101
3.4.2 180 Degree Bend (Small Radius) . . . 102
3.4.2.1 Non-Optimized Extrusion Case . . . 102
3.4.2.2 Optimized Extrusion Case . . . 104
3.4.3 180 Degree Bend (Large Radius) . . . 105
3.4.3.1 Non-Optimized Extrusion Case . . . 105
3.4.3.2 Optimized Extrusion Case . . . 105
3.5 Conclusion . . . 109
3.5.1 Discussion . . . 109
3.5.2 Limitations . . . 109
3.5.2.1 Ink Model . . . 109
3.5.2.2 Parameter Space . . . 109

Bibliography **110**

List of Figures

1.1 Mechanical Potato . . . 7

2.1 Operator comparison . . . 16
2.2 Droplet impacting a surface. . . . 31
2.3 Algorithmic procedures of the code. . . . 35
2.4 Left : Particle Configuration, Right : Laplacian @ $y = 0.5$, $h = 3\Delta x$ . . . 38
2.5 SPH particles placed on top of Finite Element nodes . . . 43
2.6 1D Transient Heat Transfer Problem Schematic . . . 43
2.7 1D Transient Heat Transfer Problem @ $t = 0.03$ . . . 44
2.8 Poiseuille flow shear transfer test . . . 45
2.9 Average Shear Stress Error . . . 46
2.10 Evolution of a fluid droplet impacting the substrate. . . . 48
2.11 Temperature distribution on the substrate surface at $t/t_c = 1.75$. . . . 49
2.12 Stress (Von-Mises) distribution on the substrate surface. . . . 51
2.13 Stress on the substrate with respect to depth $(t/t_c = 1.75, x = 0)$. . . . 53
2.14 Temperature evolution on substrate surface, with respect to time. . . . 54
2.15 Stress Components , $t/t_c = 1.75$ . . . 55
2.16 Initial Neighbors $(\mathcal{N}_0)$ are stored within memory and used throughout the simulation. Note that the initial neighbors may leave the kernel support radius in the current configuration. . . . 60
2.17 Transient Danilovskaya Problem $\lambda = 5 \times 10^5, \mu = 2.5 \times 10^5$. w represents the displacement of the tip. . . . 66
2.18 Bilayer Bending Radius (Ideal theory vs SPH Simulation) . . . 68
2.19 Comparison of FEM and SPH for the Bilayer Bending problem. Shown: Final deformed state for data point corresponding to $E_1/E_2 = 0.5$, $h_1/h_2 = 1$ in Figure 2.18. . . . 68
2.20 A 2D version of the metamaterial structure proposed in [124]. Material 1 is a more compliant material with a smaller thermal expansion coefficient, while material 2 is a stiffer material with a larger thermal expansion coefficient. . . . 70

2.21 Numerical example of the 2D microstructure proposed in [124]. Dotted line corresponds to the original volume of the microstructure. The structure is suddenly exposed to an elevated background temperature of $\Delta\theta_{ext} = 100\ K$. At $\sim 8s$, the structure nearly reaches thermal equilibrium, exhibiting a negative overall thermal expansion coefficient. The right 2 figures show the ghost particles that were placed by our algorithm (purple markers). 71
2.22 No-slip boundary conditions for Newtonian fluids and Non-Newtonian fluids. 78
2.23 Physical parameters: $\frac{\partial p}{\partial x} = 10\ [Pa/m],\ \rho = 1\ [kg/m^3],\ H = 0.5\ [m], \boldsymbol{u}(y = 0) = 0,\ \boldsymbol{u}(y = H) = 0,\ k = 0.5\ [Pa \cdot s],\ \tau_o = 0.5\ [Pa],\ n = 0.5,\ \gamma_o = 0.001.$ 79
2.24 Steady state velocity profile compared to analytical solution. 80
2.25 The method showed a convergence order of 1.6 in space. 80
2.26 Red region is the "unyielded" fluid where the effective viscosity is significantly larger. At startup (left), the region is significantly larger compared to steady state (right). 81

3.1 Non-Newtonian Fluid Models 87
3.2 Ink being "dragged" by the nozzle . The ink may inhibit very-high viscosity values and large surface tension effects, or non-Newtonian effects ([64, 65, 130, 59]). In effect, this becomes an obstacle for obtaining a desired deposition, which can involve many experimental iterations. 95
3.3 Computational Simulation of Ink Deposition with SPH. Blue object represents the nozzle, whilte the white particles are the SPH particles representing the ink. 96
3.4 Structure of our optimization framework, including the PSO optimizer and SPH based fitness value solver. The multiple evaluations corresponding to each evaluation point can be done simultaneously by invoking multiple solvers. 97
3.5 Toolpath Generation Procedure (From Top Left to Bottom Right) 99
3.6 Naive Deposition (Dotted red: desired deposition path), Fitness=2, 021, 340.0 101
3.7 Evolution of Fitness Value, 90 Degree Bend 102
3.8 Naive Deposition (Dotted red: desired deposition path, Dotted blue: nozzle path) 103
3.9 Naive Deposition (Dotted red: desired deposition path), Fitness=631, 201.0 104
3.10 Evolution of Fitness Value, 180 Degree Small Radius Bend 105
3.11 Optimized Deposition (Dotted red: desired deposition path, Dotted blue: nozzle path) 106
3.12 Naive Deposition (Dotted red: desired deposition path), Fitness=870, 687.0 107
3.13 Evolution of Fitness Value, 180 Degree Large Radius Bend 107
3.14 Optimized Deposition (Dotted red: desired deposition path, Dotted blue: nozzle path) 108

List of Tables

2.1 Physical Parameters . . . 70

3.1 Non-Newtonian Fluid Models . . . 86
3.2 Fluid Properties . . . 101
3.3 Bounds for Randomization (90 Degree Bend) . . . 102
3.4 Randomization Parameters(Small Radius 180 Degree Bend) . . . 104
3.5 Bounds for Randomization (Large Radius 180 Degree Bend) . . . 106

Preface

In no doubt, the manufacturing is among one of the most significant fields that have massively benefited from the development of efficient numerical computational methods. For example, the production of a modern automobile will be close to impossible without modern simulation technologies, most of which are based on the tried-and-true Finite Element Method (FEM) ([14, 131]); covering everything from sheet metal stamping process simulations, structural stiffness optimizations, engine combustion simulations to virtual crash tests. This is also true for nearly every conventional manufacturing based industries. I firmly believe that it is not an exaggeration that modern technological marvels were made possible due to the power of computational methods in general.

Apart from conventional subtractive / deformation based manufacturing methods, a completely different type of manufacturing method has been emerging throughout the last couple of decades: Additive Manufacturing, also known as 3D printing. The term additive manufacturing encompasses a large set of manufacturing methods where materials are combined to form a meaningful object; this includes methods such as FDM (Fused Deposition Modeling), DIW (Direct Ink Writing), SLM (Selective Laser Melting), SLA (Stereolithography Additive Manufacturing). Although the concept itself dates back as early as the 1980s, the modern advancements in microprocessor technologies and open-source software made the method widely available in recent years. Unlike conventional methods, such manufacturing methods involve many multi-physical phenomena that are sometimes extremely difficult to model with conventional numerical methods. For example, FDM involves the deposition of an amorphous polymer that is heated up until its glass-transition temperature, where it will start bonding to a pre-existing structure via crosslinking. Such glassy polymers are often extremely difficult to model ([8, 7]), and may exhibit complex mechanical responses due to thermal conditions. It can be said that, in a practical point of view, there are not many readily available solvers that have the capability to effectively provide a computational model for such applications.

In the field of computational mechanics, conventional methods such as FEM (Finite Element Method) have provided incredibly accurate simulations throughout its invention in the early 1940s. One main advantage is its capability to provide accurate computations without having “millions” of nodes. However, the advancement of the power of microprocessors following Moore’s Law ([106]) has allowed modern computational simulations to use many more unknowns. I believe that one major reason that popular mesh-free methods such as SPH (Smoothed Particle Hydrodynamics) or MPM (Material Point Method) have become widely adopted is because of the processing power of modern computers have finally allowed them to be useful.

SPH first started out as a numerical model to model astrophysical phenomenon ([94, 88, 89, 95]) by J.J. Monaghan. Due to it’s simplicity, it quickly evolved into a numerical method with its own category. Modern SPH solvers have become highly advanced compared to its original version, boasting features such as incompressible pressure solvers ([126, 37, 61, 16]), implicit viscous solvers ([16, 122, 113]), shock capturing pressure solvers

([62, 71, 70, 13, 11]), kernel correction methods ([44, 43, 118, 6, 18]) and more.

In this report, the possibility of employing the popular mesh-free method, Smoothed Particle Hydrodynamics (SPH), towards analyzing extrusion based deposition fabrication methods is explored. One major benefit of SPH is that it can trivially handle free-surface boundaries for fluids. Compared to mesh-based methods such as FEM, this eliminates major problems involved with the complex evolving topology of the deposited fluid. This makes SPH an ideal candidate to model such deposition problems, although there exists difficulties that one must overcome. One interesting problem that we look into is the discrepencies between the nozzle path and the actual deposition shape ([64, 130, 59, 65]). This often results in a situation where fine-tuning of the nozzle must be done in order to obtain desirable results. Coupled with a PSO (Particle Swarm Optimization) optimization algorithm, we prove that we can simulate and optimize the nozzle toolpaths in order to produce a deposition much closer to the original desired shape.

To end, I would like to thank my academic advisor Professor Zohdi for the many years of advising me (which I assume must have took great patience). I would also like to thank professor Souto-Iglesias from Universidad PolitÃ©cnica de Madrid, who teached me how to be better at SPH. I would not have been able to present the results in this paper without them.

Acknowledgments

I would like to thank professor Tarek Zohdi for advising me throughout the years. He patiently waited for me to mature as a researcher and an engineer.

I am also thankful to professor Antonio Souto-Iglesias for introducing me to the world of smoothed particle hydrodynamics. I would not have been able to write this dissertation without his training.

Chapter 1

Brief Review of Continuum Mechanics

1.1 Preliminary Mathematical Notations

Continuum mechanics is the foundation of virtually every fluid / solid model, and is critical in understanding our computational model that will be introduced later on. We will briefly go over mathematical preliminaries regarding the subject.

1.1.1 Vectors, Scalars and Matrices

1.1.1.1 Vectors

A N-dimensional vector is a collection of N scalars, and is written in bold (usually lower case):

$$\boldsymbol{a} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \tag{1.1}$$

a_i represents the ith component of the vector $\boldsymbol{a}$.

A "vector function" is a function where the result is a vector. This can be thought of as a collection of 3 scalar functions:

$$\boldsymbol{a}(\cdots) = \begin{bmatrix} a_1(\cdots) \\ a_2(\cdots) \\ a_3(\cdots) \end{bmatrix} \tag{1.2}$$

1.1.1.2 Matrices

A matrix is a collection of scalars presented in rows and columns, and is usually written in uppercase bold symbols:

$$\boldsymbol{A} = \begin{bmatrix} A_{11} & A_{12} & & \\ A_{21} & \cdots & & \\ & & \cdots & \\ & & & A_{nm} \end{bmatrix} \tag{1.3}$$

where A_{ij} represents the entry in the ith row and jth column. A matrix can be multiplied with another matrix:

$$\underbrace{\boldsymbol{A}}_{n\times m} \cdot \underbrace{\boldsymbol{B}}_{m\times n} = \underbrace{\boldsymbol{C}}_{n\times n} \tag{1.4}$$

Note: The dot $(\cdot)$ can be left out so that $\boldsymbol{AB} = \boldsymbol{C}$.

Where the components of $\boldsymbol{C}$ is defined as

$$C_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj} \tag{1.5}$$

The transpose of a matrix $\boldsymbol{A}$ is defined as:

$$\boldsymbol{A}^T = \begin{bmatrix} A_{11} & A_{21} & & \\ A_{21} & \cdots & & \\ & & \cdots & \\ & & & A_{mn} \end{bmatrix} \tag{1.6}$$

with components $\left(\boldsymbol{A}^T\right)_{ij} = A_{ji}$.
The inverse of a matrix $\boldsymbol{A}$, is defined as

$$\boldsymbol{A}^{-1} \cdot \boldsymbol{A} = 1 \tag{1.7}$$

Note: The existance of the inverse matrix is not guaranteed.

1.1.1.3 Basic Matrix Algebra

We state the following properties without proof:

$$\boldsymbol{A}\left(\boldsymbol{B} + \boldsymbol{C}\right) = \boldsymbol{A} \cdot \boldsymbol{B} + \boldsymbol{A} \cdot \boldsymbol{C} \tag{1.8}$$

$$\left(\boldsymbol{A} + \boldsymbol{B}\right)^T = \boldsymbol{A}^T + \boldsymbol{B}^T \tag{1.9}$$

$$\left(\boldsymbol{A}\boldsymbol{B}\right)^T = \boldsymbol{B}^T\boldsymbol{A}^T \tag{1.10}$$

$$\left(\boldsymbol{A}\boldsymbol{B}\right)^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1} \text{ (If the inverse exists for both } \boldsymbol{A}, \boldsymbol{B}) \tag{1.11}$$

$$\left(\boldsymbol{A}^{-1}\right)^T = \left(\boldsymbol{A}^T\right)^{-1} \text{ (If the inverse exists for } \boldsymbol{A}) \tag{1.12}$$

1.1.1.4 Eigenvalues and Eigenvectors

$\boldsymbol{\phi}$ is an eigenvector of the matrix $\boldsymbol{A}$ if:

$$\boldsymbol{A}\boldsymbol{\phi} = \lambda\boldsymbol{\phi} \tag{1.13}$$

where ϕ is the corresponding eigenvector. In general, they can be found by considering the following:

$$\left(\boldsymbol{A} - \lambda\boldsymbol{1}\right)\phi = 0 \tag{1.14}$$

Since we are interested in non-zero eigenvectors,

$$det\left(\boldsymbol{A} - \lambda\boldsymbol{1}\right) = 0 \tag{1.15}$$

In which we solve for the eigenvalues by solving the resulting characteristic equation.

1.1.1.5 Determinant and Condition Numbers

The following identities apply for the determinant of a matrix:

$$det(\boldsymbol{AB}) = det(\boldsymbol{A})det(\boldsymbol{B}) \tag{1.16}$$

$$det(\boldsymbol{A}^{-1}) = \frac{1}{det\,(\boldsymbol{A})} \tag{1.17}$$

The condition number is defined as:

$$Condition\ Number = \gamma = \sqrt{\frac{\text{Largest Eigenvalue}}{\text{Smallest Eigenvalue}}} \tag{1.18}$$

1.1.2 Tensors

1.1.2.1 Definition

In mathematical terms, a tensor is an object that can operate on another scalar, vector or tensor. For a tensor with two free indicies, it can be considered to be a "matrix". The same properties of a matrix will apply in this case:

$$\boldsymbol{Ta} = \boldsymbol{b} \tag{1.19}$$

where $\boldsymbol{a}$ and b are $n \times 1$ vectors and $\boldsymbol{T}$ is a $n \times n$ tensor. Such tensors are classified as "second order tensors". It can also operate on other second order tensors to produce another second order tensor:

$$\boldsymbol{T}_1 \cdot \boldsymbol{T}_2 = \boldsymbol{T}_3 \tag{1.20}$$

Sometimes, the order of a tensor is represented by the corresponding number of underbars:

$$\underline{\boldsymbol{T}} \text{ (Second Order Tensor)} \tag{1.21}$$

Higher order tensors has more than 2 free indicies, and can no longer be represented as a row/column matrix. For example, consiter a 4th order tensor $\underline{\boldsymbol{C}}$ operating on a second order tensor $\underline{\boldsymbol{\epsilon}}$ to produce a second order tensor $\underline{\boldsymbol{\sigma}}$:

$$\underline{\boldsymbol{C}} : \underline{\boldsymbol{\epsilon}} = \underline{\boldsymbol{\sigma}} \tag{1.22}$$

Where the i, j, k, lth component defined as:

$$(\underline{\boldsymbol{C}})_{ijkl} = C_{ijkl} \tag{1.23}$$

1.1.2.2 Einstein Notation

While dealing with tensor calculus, Einstein developed a convinient way to write them. For example, consider a second order tensor operating on a 3-by-1 vector:

$$\boldsymbol{T} \cdot \boldsymbol{n} = \boldsymbol{t} \tag{1.24}$$

In component form, this can be written as:

$$\sum_{j=1}^{3} T_{ij} n_j = t_i \tag{1.25}$$

In Einstein notation, the summation notation ($\sum$) is simply left out, and the repeated indicies (in this case, j) is assumed to be summed over the number of dimensions (in this case, 3). Thus,

$$\boldsymbol{T} \cdot \boldsymbol{n} = \boldsymbol{t} \overset{\text{Equivalent}}{\rightarrow} T_{ij} n_j = t_i \tag{1.26}$$

For a second order tensor operating on another second order tensor:

$$\boldsymbol{A} \cdot \boldsymbol{B} = \boldsymbol{C} \overset{\text{Equivalent}}{\rightarrow} A_{ik} B_{kj} = C_{ij} \tag{1.27}$$

Notice that operating a tensor on a vector or a different tensor eliminates the free index by summing over it. This is called “contraction”. Sometimes, the number of indicies being summed over is represented via the number of dots betwwen the two symbols (as seen above). For example,

$$\underline{\boldsymbol{C}} : \underline{\boldsymbol{\epsilon}} = \underline{\boldsymbol{\sigma}} \rightarrow C_{ijkl} \epsilon_{kl} = \sigma_{ij} \tag{1.28}$$

is considered a double-contraction.

1.1.2.3 Tensor Calculus

The true convenience of the Einstein notation comes with doing calculus with it.

Gradient The spatial gradient of a scalar field will result in a vector field:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \frac{\partial f}{\partial x_3} \end{bmatrix} \tag{1.29}$$

Or equivalently,

$$\nabla f = \frac{\partial f}{\partial x_i} \boldsymbol{e}_i \text{ (Vector Form)} \overset{\text{Equivalent}}{\rightarrow} (\nabla f)_i = \frac{\partial f}{\partial x_i} \text{ (Component Form)} \tag{1.30}$$

Divergence The divergence of a vector field is written as

$$\nabla \cdot \boldsymbol{u} = \frac{\partial u_i}{\partial x_i} \tag{1.31}$$

Tensor Product The tensor product of two vectors produces a tensor, and is written as

$$a_i b_j \tag{1.32}$$

$$\boldsymbol{a} \otimes \boldsymbol{b} = \begin{bmatrix} a_1 b_1 & a_1 b_2 & a_1 b_3 \\ a_2 b_1 & a_2 b_2 & a_2 b_3 \\ a_3 b_1 & a_3 b_2 & a_3 b_3 \end{bmatrix} \tag{1.33}$$

In Einstein notation, it is much more concise:

$$\boldsymbol{a} \otimes \boldsymbol{b} = a_i b_j \boldsymbol{e}_i \otimes \boldsymbol{e}_j \text{ (Vector Form)} \overset{\text{Equivalent}}{\rightarrow} (\boldsymbol{a} \otimes \boldsymbol{b})_{ij} = a_i b_j \text{ (Component Form)} \tag{1.34}$$

Differentiation Differentiating a vector field by another vector field, we obtain a tensor:

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}} = \begin{bmatrix} \frac{\partial a_1}{\partial b_1} & \frac{\partial a_1}{\partial b_2} & \frac{\partial a_1}{\partial b_3} \\ \frac{\partial a_2}{\partial b_1} & \frac{\partial a_2}{\partial b_2} & \frac{\partial a_2}{\partial b_3} \\ \frac{\partial a_3}{\partial b_1} & \frac{\partial a_3}{\partial b_2} & \frac{\partial a_3}{\partial b_3} \end{bmatrix} \tag{1.35}$$

In Einstein notation,

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}} = \frac{\partial a_i}{\partial b_j} \boldsymbol{e}_i \otimes \boldsymbol{e}_j \text{ (Vector Form)} \overset{\text{Equivalent}}{\rightarrow} \left(\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}} \right)_{ij} = \frac{\partial a_i}{\partial b_j} \text{ (Component Form)} \tag{1.36}$$

1.2 Kinematics

1.2.1 Deformation Gradient

In solid mechanics, the deformation is one of the most important tensors that are virtually used in every material model to describe its mechanical response. Physically, its represents the mathematical gradient of the deformation.

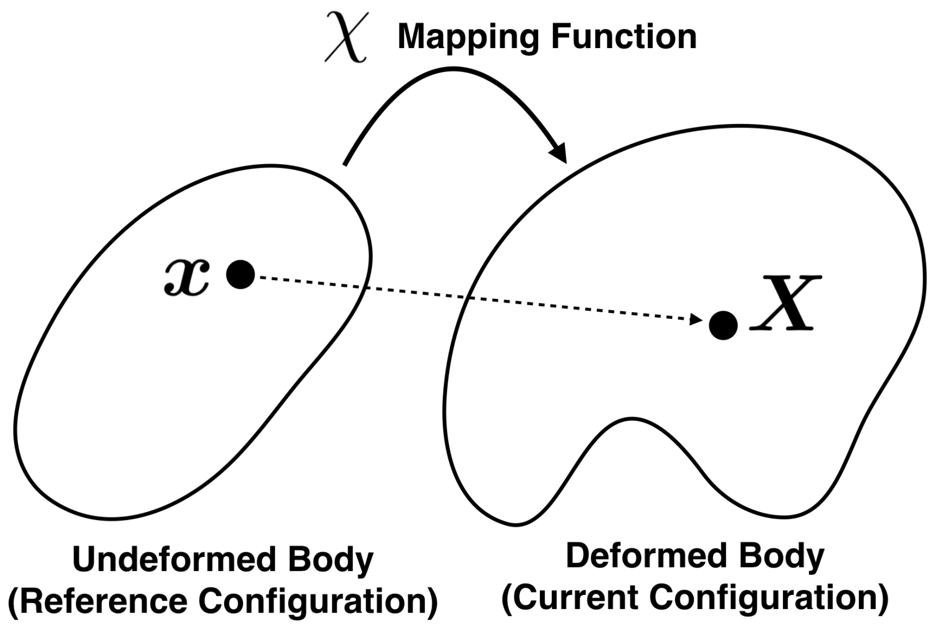


Figure 1.1: Mechanical Potato

The coordinate of the point on the deformed "mechanical potato" ($\boldsymbol{x}$) also corresponds to a coordinate ($\boldsymbol{X}$) on the undeformed, original "potato". Let us call this mapping function $\chi(\cdot)$:

$$\boldsymbol{x}(t) = \chi(\boldsymbol{X}, t) \tag{1.37}$$

which is a function of time and the original position vector on the undeformed body. The displacement is computed by (easy to get confused by the velocity field in fluid mechanics, since they use the same symbol)

$$\boldsymbol{u}(t) = \boldsymbol{x}(t) - \boldsymbol{X} \tag{1.38}$$

Finally, the deformation gradient is otained by taking the spatial gradient of $\boldsymbol{u}$:

$$\boldsymbol{F}(t) = \frac{\partial \boldsymbol{x}(t)}{\partial \boldsymbol{X}(t)} \tag{1.39}$$

The resulting tensor occupies the symbol $\boldsymbol{F}$, and is a second order tensor. In component form:

$$F_{ij} = \frac{\partial x_i}{\partial X_j} \tag{1.40}$$

1.2.2 Jacobian

The Jacobian is defined as

$$J = det\left(\boldsymbol{F}\right) \tag{1.41}$$

Physically, it represents the amount of "stretch" occuring due to the deformation:

$$dV_{defomred} = J \cdot dV_{undeformed} \tag{1.42}$$

1.2.3 Decomposition of the Deformation Gradient

The polar decomposition theorem states that it is always possible to decompose am admissible deformation gradient into a rotational part and a stretching part:

$$\boldsymbol{F} = \boldsymbol{R} \cdot \boldsymbol{U} = \boldsymbol{V} \cdot \boldsymbol{R} \tag{1.43}$$

where $\boldsymbol{R}$ is the rotational tensor and $\boldsymbol{U}, \boldsymbol{V}$ is the stretch tensor. Here, $\boldsymbol{R}$ is a proper orthogonal tensor and $\boldsymbol{U}, \boldsymbol{V}$ is a symmetric, positive definite tensor. $\boldsymbol{U}$ and $\boldsymbol{V}$ can both be decomposed via spectral decomposition:

$$\boldsymbol{U} = \sum_i \lambda_i \boldsymbol{r}_i \otimes \boldsymbol{r}_i \tag{1.44}$$

$$\boldsymbol{V} = \sum_i \lambda_i \boldsymbol{l}_i \otimes \boldsymbol{l}_i \tag{1.45}$$

The λ_is are known as the principal stretches.

1.2.4 Strain

Strain is a measurement of deformation, and is used directly to determine the mechanical response of the material in most solid models. The widely adopted Green-Strain is written as

$$\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{F}^T \boldsymbol{F} - \mathbf{1}\right) \tag{1.46}$$

An approximation for "small strains" for green strain can also be made:

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\boldsymbol{F}^T + \boldsymbol{F}\right) - \mathbf{1} \tag{1.47}$$

Note that the above approximation works well for "small strains", only under the condition that rotations are not present. If rotations are included in the deformation, the approximation no longer works even the strain itself might be considered "small'.

1.2.5 Stress Measures

1.2.5.1 Cauchy Stress Tensor

The Cauchy stress is a second order tensor that can be related to the traction vector:

$$\boldsymbol{\sigma}\left(\boldsymbol{x}\right)\cdot\boldsymbol{n}=\boldsymbol{t} \tag{1.48}$$

Here, $\boldsymbol{\sigma}$ is the Cauchy stress and $\boldsymbol{n}$ is the normal vector of an arbitrary imaginary plane cutting across the small "stress cube" located in point $\boldsymbol{x}$. The traction vector, $\boldsymbol{t}$, is the amount of force that will be acting on this plane. This tensor is defined in the current configuration.

1.2.5.2 First Piola-Kirchhoff Stress Tensor

The Piola-Kirchhoff stress tensor is defined as

$$\boldsymbol{S}=J\boldsymbol{\sigma}\boldsymbol{F}^{-T} \tag{1.49}$$

and is defined in the current configuration.

1.2.5.3 Second Piola-Kirchhoff Stress Tensor

The Second Piola-Kirchhoff stress tensor is one of the most widely used stress measures to represent the material response for a given strain. This tensor is defined as

$$\boldsymbol{S}^{(2)}=J\boldsymbol{F}^{-1}\boldsymbol{\sigma}\boldsymbol{F}^{-T} \tag{1.50}$$

For example, the stress response of a Saint Venant-Kirchhoff model is given as

$$\boldsymbol{S}^{(2)}=\lambda tr\left(\boldsymbol{E}\right)\mathbf{1}+2\mu\boldsymbol{E} \tag{1.51}$$

where $\boldsymbol{E}$ is the Green strain and μ is the shear modulus. We will review material models later on.

1.2.5.4 Stress Rates

In many cases regarding models used for viscoelastic / viscoplastic fluids, it is much more convenient to model stress rates, instead of finding the absolute stress. One attempting to model viscoelastic / viscoplastic fluids will find such rates extremely useful. We mention some popular stress rates below:

$$\overset{\triangle}{\boldsymbol{\sigma}}=\dot{\boldsymbol{\sigma}}+\boldsymbol{L}^T\boldsymbol{\sigma}+\boldsymbol{\sigma}\boldsymbol{L}\text{ (Convected Rate)} \tag{1.52}$$

$$\overset{\triangledown}{\boldsymbol{\sigma}}=\dot{\boldsymbol{\sigma}}-\boldsymbol{L}\boldsymbol{\sigma}-\boldsymbol{\sigma}\boldsymbol{L}^T\text{ (Oldroyd Rate)} \tag{1.53}$$

$$\overset{\circ}{\boldsymbol{\sigma}} = \dot{\boldsymbol{\sigma}} - \boldsymbol{W}\boldsymbol{\sigma} + \boldsymbol{W}\boldsymbol{\sigma} \text{ (Jaumann Rate)} \tag{1.54}$$

$$\overset{\otimes}{\boldsymbol{\sigma}} = \dot{\boldsymbol{\sigma}} + \boldsymbol{\sigma}\boldsymbol{\Omega} - \boldsymbol{\Omega}\boldsymbol{\sigma} \text{ (Green-Naghdi Rate)} \tag{1.55}$$

Where $\boldsymbol{L}$ is the velocity gradient $\left(\boldsymbol{L} = \frac{\partial \boldsymbol{v}}{\partial \boldsymbol{x}}\right)$ and $\boldsymbol{W}$ is the spin tensor $\left(\boldsymbol{W} = \frac{1}{2}\left(\boldsymbol{L} - \boldsymbol{L}^T\right)\right)$ and $\boldsymbol{\Omega} = \dot{\boldsymbol{Q}}\boldsymbol{Q}^T$. It is also worth noting that the integration of stress rates over a long period of time does cause accumulation of errors in many cases. Thus, it is beneficial to avoid when possible, in general.

1.3 Balance Laws

Essential concepts of balance laws in the context of continuum mechanics are presented. To be concise, many proofs and details are left out; most of them are outside the scope of this report. For a detailed review on continuum mechanics, it is strongly suggested for one to go through excellent textbooks such as [52, 111].

1.3.1 Conservation of Mass

For a region Ω within a body, the following must hold:

$$\frac{d}{dt}\int_\Omega \rho\left(\boldsymbol{x}, t; \chi\right) dV = 0 \tag{1.56}$$

From the transport equations, the above becomes

$$\int_\Omega \left(\dot{\rho} + \rho\nabla\cdot\boldsymbol{v}\right) dV = 0 \tag{1.57}$$

Since Ω is arbitrary, the integrand must satisfy:

$$\dot{\rho} + \rho\nabla\cdot\boldsymbol{v} = 0 \tag{1.58}$$

1.3.2 Balance of Linear Momentum

For a region Ω within a body, the resulting forces equals the rate of increase of its linear momentum:

$$\int_\Omega \rho\boldsymbol{b}dV + \int_{\partial\Omega} \boldsymbol{t}dA = \frac{d}{dt}\int_\Omega \rho\boldsymbol{v}dV \tag{1.59}$$

Where $\partial\Omega$ represents the boundary of the region Ω. Assuming the fields are smooth, using the fact that (without proof)

$$\frac{d}{dt}\int_{\Omega}\rho\phi dV=\int_{\Omega}\rho\dot{\phi}dV \tag{1.60}$$

we may write

$$\int_{\Omega}\rho\boldsymbol{b}dV+\int_{\partial\Omega}\boldsymbol{t}dA=\int_{\Omega}\rho\dot{\boldsymbol{v}}dV \tag{1.61}$$

1.3.3 Field Equations: Balance of Linear Momentum

We will state without proof (refer to [52] for details regarding the Cauchy's tetahedron and the symmetry of the stress tensor) that the surface integral of the traction vector can also be represented as

$$\int_{\partial\Omega}\boldsymbol{t}dA=\int_{\partial\Omega}\boldsymbol{T}^{T}\cdot\boldsymbol{n}dA \tag{1.62}$$

where $\boldsymbol{T}$ is the Cauchy stress tensor where its components are defined by

$$T_{ij}=\boldsymbol{t}\left(-\boldsymbol{e}_{i}\right)\otimes\left(-\boldsymbol{e}_{j}\right) \tag{1.63}$$

This allows to rewrite the balance of linear momentum:

$$\int_{\Omega}\rho\boldsymbol{v}dV=\int_{\partial\Omega}\boldsymbol{T}^{T}\boldsymbol{n}dA+\int_{\Omega}\rho\boldsymbol{b}dV=\int_{\partial\Omega}\boldsymbol{T}^{T}\boldsymbol{n}dA+\int_{\Omega}\rho\boldsymbol{b}dV \tag{1.64}$$

Thus;

$$\int_{\Omega}\left(\nabla\cdot\boldsymbol{T}+\rho\boldsymbol{b}-\rho\dot{\boldsymbol{v}}\right)dV \tag{1.65}$$

Again, since Ω is arbitrary, the above can be localize to give

$$\nabla\cdot\boldsymbol{T}^{T}+\rho\boldsymbol{b}=\rho\dot{\boldsymbol{v}} \tag{1.66}$$

Or in component form:

$$\frac{\partial T_{ji}}{\partial x_{j}}+\rho b_{i}=\rho\dot{v}_{i} \tag{1.67}$$

From Cauchy's theorem, we have

$$\boldsymbol{T}^{T}=\boldsymbol{T} \tag{1.68}$$

Thus we have

$$\nabla\cdot\boldsymbol{T}+\rho\boldsymbol{b}=\rho\dot{\boldsymbol{v}} \tag{1.69}$$

The Cauchy stress tensor $\boldsymbol{T}$ is also commonly written as $\boldsymbol{\sigma}$:

$$\nabla\cdot\boldsymbol{\sigma}+\rho\boldsymbol{b}=\rho\dot{\boldsymbol{v}} \tag{1.70}$$

1.3.4 Field Equations: Thermodynamics

Following the first law of thermodynamics, the rate of work and heating being applied to a region of the body Ω must equal the total rate of energy change occuring within the body. This can be written as

$$\int_{\partial\Omega} \boldsymbol{t}\cdot\boldsymbol{v}dA + \int_{\Omega}\rho\boldsymbol{b}\cdot\boldsymbol{v}dV + \int_{\partial\Omega}\boldsymbol{q}\cdot\boldsymbol{n}dA + \int_{\Omega}\rho r dV = \frac{d}{dt}\left(\int_{\Omega}\frac{1}{2}\rho\boldsymbol{v}\cdot\boldsymbol{v}dV + \int_{\Omega}\rho\varepsilon dV\right) \tag{1.71}$$

Here, the terms correspond to

$$\int_{\partial\Omega}\boldsymbol{t}\cdot\boldsymbol{v}dA \rightarrow \text{Mechanical power} \tag{1.72}$$

$$\int_{\Omega}\rho\boldsymbol{b}\cdot\boldsymbol{v}dV \rightarrow \text{Work done by body forces} \tag{1.73}$$

$$\int_{\partial\Omega}\boldsymbol{q}\cdot\boldsymbol{n}dA \rightarrow \text{Heat flux entering boundary} \tag{1.74}$$

$$\int_{\Omega}\rho r dV \rightarrow \text{Internal Heat Generation} \tag{1.75}$$

$$\int_{\Omega}\frac{1}{2}\rho\boldsymbol{v}\cdot\boldsymbol{v}dV \rightarrow \text{Kinetic Energy} \tag{1.76}$$

$$\int_{\Omega}\rho\varepsilon dV \rightarrow \text{Internal Energy} \tag{1.77}$$

We also state the work-energy identity without further proof:

$$\int_{\partial\Omega}\boldsymbol{t}\cdot\boldsymbol{v}dA + \int_{\Omega}\rho\boldsymbol{b}\cdot\boldsymbol{v}dV = \frac{d}{dt}\left(\int_{\Omega}\frac{1}{2}\rho\boldsymbol{v}\cdot\boldsymbol{v}dV\right) + \int_{\Omega}\boldsymbol{T}:\boldsymbol{D}dV \tag{1.78}$$

where $\boldsymbol{T}$ is the Cauchy stress and $\boldsymbol{D} = \nabla\boldsymbol{v}$ is the velocity gradient. Substituting the identity into equation 1.71,

$$\int_{\partial\Omega}\boldsymbol{q}\cdot\boldsymbol{n}dA + \int_{\Omega}\rho r dV + \int_{\Omega}\boldsymbol{T}\cdot\boldsymbol{D}dV = \frac{d}{dt}\left(\int_{\Omega}\rho\varepsilon dV\right) = \int_{\Omega}\rho\dot{\varepsilon}dV \tag{1.79}$$

Note that we have again used the identity $\frac{d}{dt}\int_{\Omega}\rho\phi dV = \int_{\Omega}\rho\dot{\phi}dV$. Applying the divergence theorem,

$$\int_{\Omega}\left(\boldsymbol{T}\cdot\boldsymbol{D} + \nabla\cdot\boldsymbol{q} + \rho r - \rho\dot{\varepsilon}\right)dV = 0 \tag{1.80}$$

Since Ω is an arbitrary region within the body, we can localize (like earlier) into

$$\boldsymbol{T}\cdot\boldsymbol{D} + \nabla\cdot\boldsymbol{q} + \rho r = \rho\dot{\varepsilon} \tag{1.81}$$

Chapter 2

Topics in Smoothed Particle Hydrodynamics

2.1 Introduction to SPH

2.1.1 Concept

SPH was first introduced by JJ. Monghan to solve astrophysics problems ([94, 89, 88, 95]). In SPH, a kernel function that centers around a SPH "particle" is defined as $W(\boldsymbol{r}, h)$, where $\boldsymbol{r}$ is the position vector to the center of the kernel (the position of the SPH particle), and h is the smoothing length. There are many possible selections of such kernel functions, although it is known that the Wendland kernel provides the best overall numerical properties ([77]).

2.1.2 Operators

A physical property ψ at the location of particle a then can be written as an interpolation of the properties of the surrounding particles (neighbors)

$$\langle\psi\rangle_a = \sum_{b\in\mathcal{N}_a} \psi_b v_b W(\boldsymbol{r}_{ab}, h) \tag{2.1}$$

where $W_{ab} = W(\boldsymbol{r}_{ab}, h)$, and $\boldsymbol{r}_{ab} = \boldsymbol{r}_a - \boldsymbol{r}_b$. Also, for convenience, we write the angled brackets $\langle\cdot\rangle$ to represent the value obtained from the SPH operators.

As first introduced by Monaghan, the gradient of a physical property $\nabla\psi_a$ can be written as:

$$\langle\nabla\psi\rangle_a = \sum_{b\in\mathcal{N}_a} \psi_b v_b \nabla W_{ab} \tag{2.2}$$

although this is not the only way one can write the gradient operator. One can write the above so that conservation properties are obtained:

$$\langle\nabla\psi\rangle_a = \sum_{b\in\mathcal{N}_a} (\psi_a + \psi_b) v_b \nabla W_{ab} \tag{2.3}$$

Note that this operator does not satisfy 0-order consistency, which means that it is not capable of representing a constant gradient (such as $\nabla(x+y+z) = [1 \quad 1 \quad 1]^T$). Another operator that will give us 0-order consistency for the gradient is

$$\langle\nabla\psi\rangle_a = \sum_{b\in\mathcal{N}_a} (\psi_b - \psi_a) v_b \nabla W_{ab} \tag{2.4}$$

Which is generally more widely used in SPH. Recently, in [44], a first-order consistent operator was given:

$$\langle\nabla\psi\rangle_a = \boldsymbol{B}_a \sum_{b\in\mathcal{N}_a} (\psi_b - \psi_a) v_b \nabla W_{ab} \tag{2.5}$$

Where $\boldsymbol{B}_a$ is the renormalization matrix defined as

$$\boldsymbol{B}_a = -\left[\sum_{b\in\mathcal{N}_a} \boldsymbol{r}_{ab} \otimes \nabla W_{ab} v_b\right]^{-1} \tag{2.6}$$

The divergence of a vector field can also be written in a similar manner:

$$\left\langle \nabla \cdot \boldsymbol{\psi} \right\rangle_a = \sum_{b\in\mathcal{N}_a} \left(\boldsymbol{\psi}_b - \boldsymbol{\psi}_a\right) v_b \cdot \nabla W_{ab} \tag{2.7}$$

$$\left\langle \nabla \cdot \boldsymbol{\psi} \right\rangle_a = \boldsymbol{B}_a \sum_{b\in\mathcal{N}_a} \left(\boldsymbol{\psi}_b - \boldsymbol{\psi}_a\right) v_b \cdot \nabla W_{ab} \tag{2.8}$$

These operator gives first order consistency for the gradient regardless of the particle orientation. One caviat is the fact that $\boldsymbol{B}_a$ may not always be invertible.

The Laplacian operator is another operator that frequently arises in PDEs as diffusive terms. A naive implementation would be to iteratively apply first derivative operators twice, such as

$$\left\langle \nabla^2 \psi \right\rangle = \left\langle \nabla \cdot \nabla \psi \right\rangle = \sum_{b\in\mathcal{N}_a} \left(\left\langle \nabla \psi_b \right\rangle - \left\langle \nabla \psi_a \right\rangle\right) v_b \cdot \nabla W_{ab} \tag{2.9}$$

This type of operators suffer from the well-known oscillatory issues regarding "extended stencils", since the evaluation makes use of the values of the neighbors of the neighbors. In [21], an operator that retains the compact stencil was proposed:

$$\left\langle \nabla^2 \psi \right\rangle = \sum_{b\in\mathcal{N}_a} 2 v_b \frac{\psi_b - \psi_a}{r_{ab}} \boldsymbol{e}_{ab} \nabla W_{ab} \tag{2.10}$$

Recently, in [44], a first-order consistent operator was introduced:

$$\left\langle \nabla^2 \psi \right\rangle = \hat{\boldsymbol{B}}_a : \sum_{b\in\mathcal{N}_a} 2 v_b \boldsymbol{e}_{ab} \otimes \nabla W_{ab} \left(\frac{\psi_a - \psi_b}{r_{ab}} - \boldsymbol{e}_{ab} \cdot \sum_{b\in\mathcal{N}_a} v_b \left(\psi_b - \psi_a\right) \boldsymbol{B}_a \cdot \nabla W_{ab}\right) \tag{2.11}$$

Where $\hat{\boldsymbol{B}}_a$ is another renormalization tensor is defined as the solution of the following:

$$\underbrace{\left[\left(\sum_{b\in\mathcal{N}_a} 2 v_b \boldsymbol{r}_{ab} \otimes \boldsymbol{e}_{ab} \otimes \boldsymbol{e}_{ab} \otimes \nabla W_{ab}\right) + \left(\sum_{b\in\mathcal{N}_a} v_b \boldsymbol{e}_{ab} \otimes \boldsymbol{e}_{ab} \otimes \nabla W_{ab}\right) \otimes \left(\sum_{b\in\mathcal{N}_a} v_b \boldsymbol{r}_{ab} \otimes \boldsymbol{r}_{ab} \otimes \nabla W_{ab}\right) : \boldsymbol{B}_a\right]}_{\boldsymbol{A}_a} : \hat{\boldsymbol{B}} \tag{2.12}$$

$$\hat{\boldsymbol{B}}_a = -\boldsymbol{A}_a^{-1} \tag{2.13}$$

Similar to the earlier renormalization tensor $\boldsymbol{B}_a$, $\boldsymbol{A}_a$ may be singular and a correction might not exist. One necessary condition for $\boldsymbol{A}_a$ to be invertible, is that particle a has a

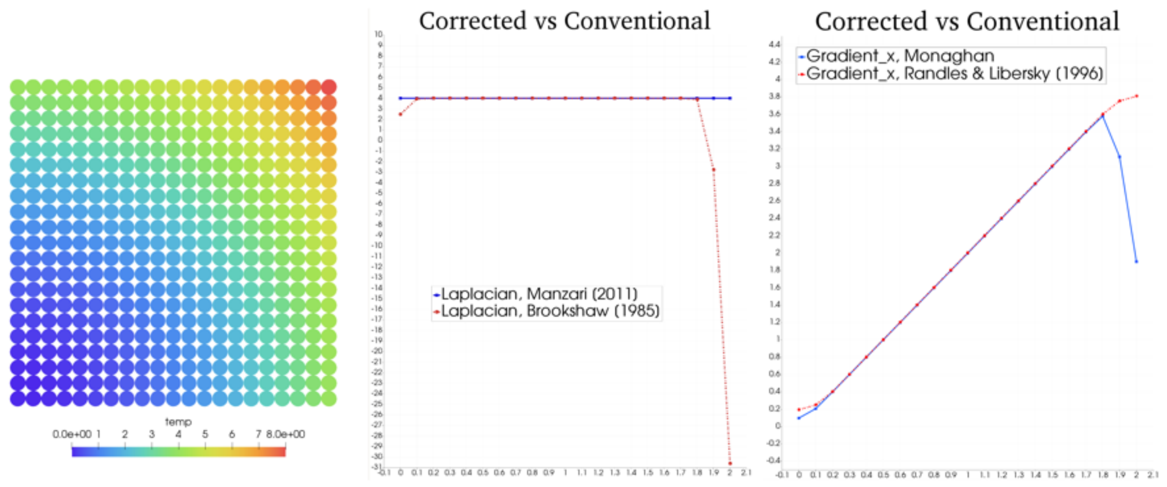


Figure 2.1: Operator comparison

neighbor in each of it's quadrants for a coordinate system centered around particle a. For example, in 3D, a particle must have at least 8 neighbors in each quadrants. In practice, an algorithm searches for the 8 particles and the attempt to invert $\boldsymbol{A}_a$ is only performed if they all exist.

2.1.3 On Solving the Navier Stokes Equation with SPH

First, we state the Navier-Stokes equation:

$$\rho \frac{D\boldsymbol{u}}{Dt} = -\nabla p + \mu \nabla^2 \boldsymbol{u} \tag{2.14}$$

the continuity equation must also be satisfied:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{2.15}$$

We start by discretizing the abobve equations using the SPH operators stated above:

$$\frac{D\mathbf{u}}{Dt} = -\left\langle \frac{1}{\rho} \nabla p \right\rangle + \left\langle \frac{\mu}{\rho} \nabla^2 \mathbf{u} \right\rangle + \mathbf{g} \tag{2.16}$$

$$\frac{D\rho}{Dt} = \langle \rho \nabla \cdot \mathbf{u} \rangle \tag{2.17}$$

$$p = f_{EOS}(\rho) \tag{2.18}$$

Where $< \cdot >$ represents the discretized version of the expression using SPH, and f_{EOS} represents the equation of state. An example of an equation of state is Tait's equation of state ([90]), where the pressure is directly computed from the density deviation:

$$p = \frac{1}{\gamma}\rho_0 c_0^2 \left(\left(\frac{\rho}{\rho_0} \right)^{\gamma} - 1 \right) \tag{2.19}$$

Although γ represents a physical material parameter associated with the fluid and c_0 represent the speed of sound, in practice, c_0 and γ are usually chosen so that the SPH model is "incompressible enough" to avoid unreasonably small timesteps. Using the gradient operator found in [96], Equation 3.10 can be written in the following symmetric form:

$$-\left\langle \frac{1}{\rho}\nabla p \right\rangle_i = -\sum_{j\in\mathcal{N}_i} \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W_{ij} m_j \tag{2.20}$$

Where the index i represents the ith SPH particle, while $j \in \mathcal{N}_i$ represents the set of neighbors of the ith particle. W_{ij} is the kernel function centered around particle i, $\nabla_i W_{ij}$ represents the gradient of the kernel, p_i is the pressure of particle i, p_j is the pressure of particle j, ρ_i is the density of particle i, ρ_j is the density of particle j, and m_i and m_j represent the mass of particle i and j, respectively. The viscosity contribution is discretized as

$$\left\langle \frac{\mu}{\rho}\nabla^2 \boldsymbol{u} \right\rangle_i = \sum_{j\in\mathcal{N}_i} \frac{4m_j(\mu_i+\mu_j)\boldsymbol{r}_{ij}\cdot\nabla_i W_{ij}}{(\rho_i+\rho_j)^2(\boldsymbol{r}_{ij}^2+\epsilon)}\boldsymbol{u}_{ij} \tag{2.21}$$

For weakly compressible SPH approaches, the following discretization for the continuity equation is employed. This is known asδ-SPH, where a numerical diffusion term is added onto the divergence of $\boldsymbol{u}$:

$$\langle \rho \nabla \cdot \boldsymbol{u} \rangle_i = -\rho_i \sum_{j\in\mathcal{N}_i} (\boldsymbol{u}_j - \boldsymbol{u}_i)\cdot\nabla_i W_{ij} V_j + \delta h c_0 \mathcal{D}_i \tag{2.22}$$

where $\mathcal{D}_i$ is the "diffusion" term, defined as

$$\mathcal{D}_i = 2\sum_{j\in\mathcal{N}_i} \psi_{ij} \frac{\boldsymbol{r}_{ji}\cdot\nabla W_{ij}}{\boldsymbol{r}_{ij}^2} V_j, \tag{2.23}$$

where δ is a tunable constant (usually chosen to be 0.1), h is the SPH smoothing length, c_0 is the speed of sound, and ψ_{ij} is defined as :

$$\psi_{ij} = (\rho_j - \rho_i) - \frac{1}{2}\left(\langle\nabla_1\rho_i\rangle + \langle\nabla_1\rho_j\rangle\right)\cdot\mathbf{r}_{ji}, \tag{2.24}$$

where $\langle\nabla_1(\,\cdot\,)\rangle$ represents the renormalized gradient operator. $\langle\nabla_1\rho_i\rangle$ can then be written as :

$$\langle\nabla_1\rho_i\rangle = \mathbf{B}_i \sum_{j\in\mathcal{N}(i)} (\rho_j - \rho_i)\nabla_i W_{ij} V_j \tag{2.25}$$

The above set of equations is a great starting point for simulating Newtonian Fluids with δ-SPH. We will introduce more advanced methods further along in the report.

Algorithm 2.1 Weakly Compressible SPH Timestep (Explicit Forward Euler)

```
for $Particle_i$ in all particles do
    Compute $-\left\langle \frac{1}{\rho(t)}\nabla p \right\rangle_i = -\sum_{j\in\mathcal{N}_i} \left( \frac{p_i}{\rho(t)_i^2} + \frac{p_j}{\rho(t)_j^2} \right) \nabla_i W_{ij} m_j$
    Compute $\mathcal{D}_i = 2 \sum_{j\in\mathcal{N}_i} \psi_{ij} \frac{\boldsymbol{r}(t)_{ji}\cdot\nabla W_{ij}}{\boldsymbol{r}(t)_{ij}^2} V_j,$
    Compute $\langle \rho(t)\nabla\cdot\boldsymbol{u}(t)\rangle_i = -\rho(t)_i \sum_{j\in\mathcal{N}_i} (\boldsymbol{u}(t)_j - \boldsymbol{u}(t)_i)\cdot\nabla_i W_{ij} V_j + \delta h c_0 \mathcal{D}_i$
end for
for $Particle_i$ in all particles do
    $\rho(t+\Delta t)_i = \rho(t)_i + (\Delta t)(\langle \rho(t)\nabla\cdot\boldsymbol{u}(t)\rangle_i)$
    $\boldsymbol{r}_i(t+\Delta t) = \boldsymbol{r}_i(t) + (\Delta t)(\boldsymbol{u}_i(t))$
    $\boldsymbol{u}_i(t+\Delta t) = \boldsymbol{u}_i(t) + (\Delta t)(-\left\langle \frac{1}{\rho(t)}\nabla p \right\rangle_i + \langle \rho(t)\nabla\cdot\boldsymbol{u}(t)\rangle_i)$
end for
```

2.1.4 Boundary Condition Enforcement

Enforcing proper boundary conditions in SPH is rather tricky. Conditions such as slip / no-slip conditions are usually achieved by using ghost particles ([81, 93, 82, 90]). This involves a layer of static SPH particles that lie on the boundary surface. During the position / velocity update phase during each timestep for the boundary particles, instead of updating them, the velocity / poisitions are enforced to be an appropriate value.

2.1.5 Free-Surface Tracking

In order to detect free-surface particles, the algorithm first filters out potential free surface particles. This can be done by computing the particle density for particle a:

$$c_a = \sum_{b\in\mathcal{N}_a} v_b W_{ab}$$

If $c_a < 0.95$, we mark particle a as a potential free-surface particle. Using this creterion exclusively usually produces undesirable results, and further filtering is required. In [83], a versitile free surface detection algorithm was implemented, and uses the following criterion to further determine whether it actually belong to the free surface :

$$\text{Condition 1:} \quad |\boldsymbol{x}_{ab}| \geq \alpha\sqrt{2}h \quad \text{and} \quad |\boldsymbol{x}_{bT}| < \alpha h$$

$$\text{Condition 2:} \quad |\boldsymbol{x}_{ab}| < \alpha\sqrt{2}h \quad \text{and} \quad |\boldsymbol{n}_a\cdot\boldsymbol{x}_{bT}| + |\boldsymbol{s}_a\cdot\boldsymbol{x}_{bT}| < \alpha h$$

Where $\boldsymbol{x}_T = \boldsymbol{x}_i + \Delta x\cdot\boldsymbol{n}_i$, and is a tuning parameter such that $0 < \alpha < 1$. In the original nomenclature in [83], α was chosen to be $\frac{1}{3}$ (as they use the term "smoothing length" as one third of the support radius of the kernel).

2.1.6 Particle Shifting

Another modern technique worth mentioning is particle shifting, and is usually used in conjunction with WCSPH methods. The main purpose of particle shifting is to provide a workaround regarding tensile instability. Shifting particles naively produces artificial defects on the free-surface. Thus, it is required that one correctly identifies the free surface and applies appropriate shifting methods accordingly. Various shifting algorithms have been proposed with varying performance. Particle "Shifting" schemes in SPH can be seen as a re-sampling scheme. It was shown that this can lead to significant improvements in accuracy and stability ([126],[104],[128]). We follow the recent work by [128] where a fine-tuned shifting methodology was successfully developed. In [104], the following normal vector computation was used with successful results:

$$\boldsymbol{n}_a = \frac{\boldsymbol{M}_a \cdot \nabla C_a}{|\boldsymbol{M}_a \cdot \nabla C_a|}$$

Now, following For the bulk of the particles that are not free-surface particles, we use

$$\delta \boldsymbol{r}_a = -\frac{5}{3} h V_a \nabla C_a \cdot \Delta t$$

For the free-surface ones, we use

$$\delta \boldsymbol{r}_a = -\frac{5}{3} h V_a \left(\mathbf{1} - \boldsymbol{n}_a \otimes \boldsymbol{n}_a\right) \nabla C_a \cdot \Delta t$$

After the position update phase during each time step, we now apply the shifting vector to the particle positions:

$$\hat{\boldsymbol{x}}_a = \boldsymbol{x}_a + \delta \boldsymbol{r}_a$$

Then, it is required that the variables associated with the particle must be advected accordingly:

$$\begin{cases} \hat{\rho}_a = & \rho_a + \left(\nabla \rho\right)_a \cdot \delta \boldsymbol{r}_a \\ \hat{\boldsymbol{v}}_a = & \boldsymbol{v}_a + \left(\nabla \boldsymbol{v}\right)_a \delta \boldsymbol{r}_a \\ \hat{\boldsymbol{\tau}}_a = & \boldsymbol{\tau}_a + \left(\nabla \boldsymbol{\tau}\right)_a \delta \boldsymbol{r}_a \\ \hat{\boldsymbol{\theta}}_a = & \boldsymbol{\theta}_a + \left(\nabla \theta\right)_a \delta \boldsymbol{r}_a \end{cases}$$

2.2 Advanced SPH Solvers

Since its development ([94]), SPH has been mostly an explicit method. By "explicit", we imply the following:

- The pressure Poisson equation (PPE) that keeps the fluid "incompressible" is only "approximated" by an equation of state.

- The viscous forces acting on each particle is computed according to the positions / velocities computed in the earlier timestep.

Explicit formulations tend to work well for low-viscosity Newtonian fluid flow applications, but for highly viscous fluids, they are usually very hard to stabilize even with tiny timesteps. Methods such as δ-SPH ([80]) are greatly beneficial for stabiblizing the pressure step, but large viscosities can still cause stability issues. This was covered in detail in an earlier section. Since the ultimate goal of this report is to devise a method that can simulate highly viscous printing inks for additive manufacturing methods, it is a requirement that an implicit timestepping scheme is used for our SPH framework.

In this section, we introduce several approaches regarding implicit timestepping with SPH.

2.2.1 Review of Implicit Methods in SPH

Enforcing incompressibility in SP

2.2.1.1 Cummins, Rudman (1999)

To our best knowledge, the earliest development of implicit pressure solvers that enforce incompressibility based on operator splitting can be traced back to the work of Cummins and Rudman ([38]). Following the original notation in the paper (note that particle indicies are a and b instead of i and j), an operator splitting methods was employed:

$$\boldsymbol{r}_a^* = \boldsymbol{r}_a^n + \Delta t(\boldsymbol{u}_a^n) \tag{2.26}$$

Where $\boldsymbol{r}^*$ represents the intermediate position computed by only applying the viscous operator:

$$\boldsymbol{u}_a^* = \boldsymbol{u}_a^n - \Delta t \left(\sum_b m_b \chi_{ab}^n (\boldsymbol{r}^*) + \frac{\boldsymbol{g}}{Fr^2} \right) \tag{2.27}$$

Here, χ_{ab}^n represents the viscous interation between particle a and b and Fr is the Froude number. The following PPE (Pressure Poisson Equation) is then solved to acquire the pressure values that each particle needs to have in order to enforce incompressibility:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla P \right)_a = \frac{\nabla \cdot \boldsymbol{u}_a^*}{\Delta t} \tag{2.28}$$

The computed pressure values are now used to obtain a divergence-free velocity field:

$$\boldsymbol{u}_a^{n+1} = \boldsymbol{u}_a^n - \Delta t \sum_b m_b \left(\frac{P_b}{\rho_b^2} + \frac{P_a}{\rho_a^2} \right) \nabla_a W_{ab} \tag{2.29}$$

The trapezoidal rule is then applied to update the particle positions:

$$\boldsymbol{r}_a^{n+1} = \boldsymbol{r}_a^n + \Delta t \left(\frac{\boldsymbol{u}_a^{n+1} + \boldsymbol{u}_a^n}{2} \right) \tag{2.30}$$

The PPE operator that was used in this paper follows as:

$$\nabla \cdot \left(\frac{1}{\rho} \nabla P \right)_a = \sum_b \frac{m_b}{\rho_b} \left(\frac{4}{\rho_a + \rho_b} \right) \frac{P_{ab} \boldsymbol{r}_{ab} \cdot \nabla_a W_{ab}}{|\boldsymbol{r}_{ab}^2| + \eta^2} \tag{2.31}$$

This operator is symmetric, and can be solved with a matrix solver such as conjugate gradient. One must note that the free surface boundary particles must be enforced a free-surface condition by precribing the pressure P_a to be zero. Thus, an accurate free-surface detection algorithm must be executed before attempting to solve the PPE.

However, in our experience, solving the above PPE resulted in oscillating pressure profiles (even with direct solvers), redering the simulation unstable for any practical usage. Cummin's also reported oscillations in their research.

2.2.1.2 IISPH - Implicit Incompressible SPH, Ihmsen et al (2014)

In 2014, an extremely robust pressure solver which has spawned many variants was developed by Ihmsen et al. ([61]). We would like to give a brief review of the method, since the original article (although excellent in content) can be somewhat confusing to follow.

The timestep is first splitted into a viscous part and the pressure part. Similar to conventional approaches, the acceleration acting on a particle is computed by:

$$\boldsymbol{a}_i^p(t) = -\sum_j m_j \left(\frac{P_i(t)}{\rho_i^2(t)} + \frac{P_j(t)}{\rho_j^2(t)} \right) \nabla W_{ij}(t) \tag{2.32}$$

No equation of state is used, so the pressure values are unknown and must be solved for. To obtain the pressure for the particles, a PPE is derived starting from the continuity equation:

$$\frac{D\rho}{Dt} = -\rho \nabla \cdot \boldsymbol{u} \tag{2.33}$$

using a forwards difference for the density:

$$\frac{\rho_i(t + \Delta t) - \rho_i(t)}{\Delta t} = \sum_j m_j \boldsymbol{v}_{ij}(t + \Delta t) \nabla W_{ij}(t) \tag{2.34}$$

Note that here we have the unknown densities $\boldsymbol{v}_i(t + \Delta t)$ and $\boldsymbol{v}_j(t + \Delta t)$. Then, the intermediate velocity after the application of viscous forces is written as:

$$\boldsymbol{v}^* = \boldsymbol{v}_i(t) + \Delta t \boldsymbol{a}_i^{adv}(t) \tag{2.35}$$

$\boldsymbol{a}_i^{adv}(t)$ represents the acceleration from the viscous forces / surface tension / gravity, etc, and are assumed to be known. We then compute an intermediate density field from equation 2.34:

$$\rho_i^* = \rho_i(t) + \Delta t \sum_j m_j \boldsymbol{v}_{ij}^* \nabla W_{ij}(t) \tag{2.36}$$

This density field clearly does not represent an "incompressible" fluid, since the viscous forces will not give a divergence-free velocity field. This needs to be fixed at the pressure step. Since we know that the density of the fluid after the pressure step needs to be ρ_0 (rest density), the velocity field from the pressure step must provide a density change to match:

$$\frac{\rho_0 - \rho_i^*}{\Delta t} = \sum_j m_j \left(\boldsymbol{v}_i \left(t + \Delta t\right) - \boldsymbol{v}_j \left(t + \Delta t\right)\right) \nabla W_{ij}(t) \tag{2.37}$$

Since $\boldsymbol{v}_i \left(t + \Delta t\right) = \boldsymbol{v}_i^* + \Delta t \boldsymbol{a}_i^p \left(t\right),\ \boldsymbol{v}_j \left(t + \Delta t\right) = \boldsymbol{v}_j^* + \Delta t \boldsymbol{a}_j^p \left(t\right)$:

$$\frac{\rho_0 - \rho_i^*}{\Delta t} = \sum_j m_j \left(\boldsymbol{v}_i^* - \boldsymbol{v}_j^*\right) \nabla W_{ij}(t) + \Delta t \sum_j m_j \left(\boldsymbol{a}_i^p - \boldsymbol{a}_j^p\right) \nabla W_{ij}(t) \tag{2.38}$$

Ihmsen further assumed that $\sum_j m_j \left(\boldsymbol{v}_i^* - \boldsymbol{v}_j^*\right) \nabla W_{ij}(t) \approx 0$ (although not explicitly stated in his paper). Note that since the intermediate velocity field is not completely divergence free, possible errors might arise (although for practical applications, the approximation yields satisfactory results). Thus, we have

$$\rho_0 - \rho_i^* = (\Delta t)^2 \left(\sum_j m_j \left(-\sum_j m_j \left(\frac{P_i(t)}{\rho_i^2\left(t\right)} + \frac{P_j(t)}{\rho_j^2\left(t\right)}\right) \nabla W_{ij}(t) - \sum_k m_k \left(\frac{P_j(t)}{\rho_j^2\left(t\right)} + \frac{P_k(t)}{\rho_k^2\left(t\right)}\right) \nabla W_{jk}(t)\right) \nabla W_{ij}(t)\right) \tag{2.39}$$

The above set of equations forms a set of linear equations that can be expressed as:

$$\boldsymbol{A} \cdot \boldsymbol{p}(t) = \boldsymbol{b} \tag{2.40}$$

Where $\boldsymbol{A}$ and $\boldsymbol{b}$ are the corresponding matrix and sourceterms, and $\boldsymbol{p}(t)$ is the unknown pressure vector.

Notice the index k, which refers to the **neighbors of the neighbors.** The only unknown values in the above expression are thre pressure values associated with each particle.

The most unique feature of this pressure solver originates from how it actually solves the PPE. Equation 2.39 is a set of linear equations, and is symmetric due to how the pressure gradient is calculated. In theory, a matrix solver such as conjugate gradient with the appropriate free-surface boundary conditions should be able to determine the pressure field (which was also mentioned in Ihmsen's original paper). However, this turned out to be difficult in our experience. Attempts were made to locate the free-surface boundary particles and implement proper free-surface boundary conditions, and then solve it via various matrix solvers (Direct / Conjugate Gradient, etc). Results showed unstable oscillations within the pressure field, similar to the results obtained via the method proposed by Cummins ([36, 38]).

In Ihmsen's original paper, he instead proposed an efficient iterative Jacobi-based iterative solver for the purpose of solving 2.39:

$$P_i^{l+1} = (1-\omega)\,P_i^l + \omega\frac{1}{a_{ii}}\left(\rho_0 - \rho_i^* - \sum_j m_j \left(\sum_j \boldsymbol{d}_{ii}P_j^l - \boldsymbol{d}_{jj}P_j^l - \sum_{k\neq i}\boldsymbol{d}_{jk}P_k^l\right)\nabla W_{ij}\right) \tag{2.41}$$

where

$$\boldsymbol{d}_{ii} = -\left(\Delta t\right)^2\left(\sum_j \frac{m_j}{\rho_i^2}\nabla W_{ij}\right),\ \boldsymbol{d}_{ij} = -\left(\Delta t\right)\frac{m_j}{\rho_j^2}\nabla W_{ij} \tag{2.42}$$

Also, during each iteration, **the pressure values where pinned to be positive:**

$$P_i^{l+1} = max(0, (1-\omega)\,P_i^l + \omega\frac{1}{a_{ii}}\left(\rho_0 - \rho_i^* - \sum_j m_j \left(\sum_j \boldsymbol{d}_{ii}P_j^l - \boldsymbol{d}_{jj}P_j^l - \sum_{k\neq i}\boldsymbol{d}_{jk}P_k^l\right)\nabla W_{ij}\right)) \tag{2.43}$$

this effectively elminates the requirement to locate the free-surface particles and enforce zero-pressure values to them. When implemented, one can observe that the free-surface particles automatically exhibit 0 pressure values. The downsides to this approach are:

- The method is *ad-hoc* in a sense. It makes assumptions to the solution while the boundary conditions are not rigorously applied.
- The method will not produce negative pressure values. Nevertheless, one can also argue that for free-surface fluid flows, effects from negative pressures are negligible (the fluid would simply break apart and will experience atmospheric pressure instead of exhibiting negative pressure).

Still, the method itself produces excellent, stable results that tolerates larger timesteps. Note that the methods was developed for computer graphics applications, and the requirements are clearly different compared to computational physics. Although we employ the above modified Jacobi solver in our numerical simulations as well, rigorous verification and validation of the proposed method in engineering solutions is something we would like to look into in the future.

2.2.1.3 DFSPH - Divergence Free SPH (Bender and Koschier, 2017)

In 2017, Bender and Koschier published a revised pressured solver based on Ihmsen's previous work ([16]). As noted earlier, Ihmsen's assumption that $\sum_j m_j \left(\boldsymbol{v}_i^* - \boldsymbol{v}_j^*\right) \nabla W_{ij}(t) \approx 0$ generates velocity fields that were not completely divergence free. DFSPH attempts to resolve this by using a different formulation for the PPE and performing a second Jacobi iteration process separately for the density values during the timestep. This results in an extremely fast and robust pressure solver. We breakdown how the algorithm stated in Bender's original manuscript works, since it may be confusing for beginning SPH practitioners.

Algorithm 2.2 DFSPH Timestep (Using notation from Bender's original paper)

```
1. Compute non-pressure forces $\boldsymbol{a}_i^v$
2. Compute $\boldsymbol{v}_i^*(t) = \boldsymbol{v}_i + \Delta t \boldsymbol{a}_i^v$
3. CorrectDensityError()
4. Update Positions $\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \Delta t \boldsymbol{v}_i^*$
4. UpdateDensities $\rho_i$
5. Compute $\alpha_i$
6. CorrectDivergenceError()
7. UpdateVelocities $\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i^*$
```

First, the viscous forces are applied and the system takes a half-step towards the intermediate state $(\boldsymbol{r}^*, \boldsymbol{v}^*, \rho^*)$. Now, the step *CorrectDensityError* and *CorrectDivergenceError* attempts to update $\boldsymbol{v}^*$ so that the updated velocity field represents a divergence-free vector field. To be explicit, the $\boldsymbol{v}^*$ in step 2. (line 12 in Algorithm 1. in the original paper) and step 4. (line 15 in Algorithm 1. in the original paper) are not the same (It is implied that the function *CorrectDensityError* and *CorrectDivergenceError* updates $\boldsymbol{v}^*$ to a divergence-free velocity field)!

Enforcing Zero Divergence We will now go in detail to explain what *CorrectDivergenceError* does to correct the divergence. The acceleratrion that will be applied due to pressure forces is computed with:

$$\boldsymbol{a}_i^p(t) = -m_i \nabla P_i(t) \tag{2.44}$$

Then, instead of the classic pressure gradient operator (which Ihmsen also adopted earlier) it is assumed that the pressure gradient ∇P_i can be represented as

$$\nabla P_i = \kappa_i^v \nabla \rho_i = \kappa_i^v \sum_j m_j \nabla W_{ij} \tag{2.45}$$

where κ_i^v represents the stiffness parameter which we need to determine.
Remark

Conceptually, this approach is similar to the SHAKE algorithm that was proposed by Ellero et al. ([42]) where Lagrange multipliers that enforce incompressibility is solved for. The noticable difference is that DFSPH takes advantage of Ihmsen's pressure-clamping Jacobi solver.

After updating the velocity field to $\boldsymbol{v}^*$ (Step 2), the divergence of the velocity field is no longer zero. This results in a $\frac{D\rho_i}{Dt}$ that is non-zero:

$$\frac{D\rho_i}{Dt} = -\rho_i \nabla \cdot \boldsymbol{v}_i^* = \sum_j m_j \left(\boldsymbol{v}_i^* - \boldsymbol{v}_j^*\right) \nabla W_{ij} \tag{2.46}$$

If we could add-on the pressure force induced velocity changes to $\boldsymbol{v}^*$ so that $\frac{D\rho_i}{Dt}$ is small, that would be exactly what we are looking for. Let us call the total pressure force "density" acting on particle i to be $\boldsymbol{f}_i^p$ and the pressure "density" acting on i coming from particle j to be $\boldsymbol{f}_{i\leftarrow j}^p$,. Then the velocity change of particle i, and the velocity change of j counting only the contrubition coming from particle i can be written as

$$\delta \boldsymbol{v}_i = \Delta t \frac{\boldsymbol{f}_i^p}{\rho_i},\ \delta \boldsymbol{v}_{j\leftarrow i} = \Delta t \frac{\boldsymbol{f}_{j\leftarrow i}^p}{\rho_i} \tag{2.47}$$

The term "Pressure force density" is a term that was used throughout the original paper, defined as $\boldsymbol{f}_i^p = -\nabla P_i$ with the unit $\left[\frac{N}{m^3}\right]$. The resulting density change resulting from this new velocity field should yield a density change that should cancel out $\frac{D\rho_i}{Dt}$:

$$\sum_j m_j \left(\delta \boldsymbol{v}_i - \delta \boldsymbol{v}_j\right) \nabla W_{ij} = \Delta t \sum_j m_j \left(\frac{\boldsymbol{f}_{j\leftarrow i}^p}{\rho_i} - \frac{\boldsymbol{f}_i^p}{\rho_i}\right) \nabla W_{ij} = -\frac{D\rho_i}{Dt} \tag{2.48}$$

substituting $\boldsymbol{f}_i^p = \nabla P_i = \kappa_i^v \nabla \rho_i = \kappa_i^v \sum_j m_j \nabla W_{ij}$,

$$\Delta t \sum_j m_j \left(\frac{\kappa_i^v m_j \nabla W_{ij}}{\rho_i} - \frac{\kappa_i^v \sum_j m_j \nabla W_{ij}}{\rho_i}\right) \nabla W_{ij} = -\frac{D\rho_i}{Dt} \tag{2.49}$$

Expanding out,

$$\frac{D\rho_i}{Dt} = \kappa_i^v \frac{\Delta t}{\rho_i} \left(\left|\sum_j m_j \nabla W_{ij}\right|^2 + \sum_j \left|m_j \nabla W_{ij}\right|^2\right) \tag{2.50}$$

We then can solve the system above wrt. κ:

$$\kappa_i^v = \frac{1}{\Delta t} \frac{D\rho_i}{Dt} \cdot \frac{\rho_i}{\alpha_i} \tag{2.51}$$

where $\alpha_i = \left|\sum_j m_j \nabla W_{ij}\right|^2 + \sum_j \left|m_j \nabla W_{ij}\right|^2$. Bender and Koschier observed that using Jacobi iterations, along with the clamping technique introduced by Ihmsen worked well for solving the above system of equations:

Algorithm 2.3 DFSPH - CorrectDivergenceError

```
while $\left(\left(\frac{D\rho}{Dt}\right)_{ave} > TOL_{div}\right)$ do
    for $Particle_i$ in all particles do
        Compute $\frac{D\rho_i}{Dt} = -\rho_i \nabla \cdot \boldsymbol{v}_i^*$
    end for
    for $Particle_i$ in all particles do
        $\kappa_i^v = \frac{1}{\Delta t}\frac{D\rho_i}{Dt} \cdot \frac{\rho_i}{\alpha_i}$, $\kappa_j^v = \frac{1}{\Delta t}\frac{D\rho_i}{Dt} \cdot \frac{\rho_j}{\alpha_j}$
        $\boldsymbol{v}^* \leftarrow \boldsymbol{v}^* - \Delta t \sum_j m_j \left(\frac{\kappa_i^v}{\rho_i} + \frac{\kappa_j^v}{\rho_j}\right) \nabla W_{ij}$
    end for
end while
```

Enforcing Zero Density Change On top of the zero divergence enforcement, Bender et al., also included a second step for the pressure solver. It is possible to consider this as a separate solver that can be applied on its own. In Bender et al., this solver was applied once more independantly after the zero divergence solver to deal with the residual errors. Ignoring the zero divergence solver step, after The non-pressure forces, the intermediate density can be written as

$$\rho_i^* = \rho_i + \Delta t \frac{D\rho_i}{Dt} = \rho_i + \Delta t \cdot \sum_j m_j \left(\boldsymbol{v}_i^* - \boldsymbol{v}_j^*\right) \nabla W_{ij} \tag{2.52}$$

the pressure forces that will enforce zero density change can written as ()

$$\rho_i^* - \rho_0 = \Delta t^2 \sum_j m_j \left(\frac{\boldsymbol{f}_i^p}{\rho_i} - \frac{\boldsymbol{f}_{j\leftarrow i}^p}{\rho_i}\right) \nabla W_{ij} \tag{2.53}$$

defining (note that this is the same as the zero divergence solver, only with $\frac{D\rho_i}{Dt} = \sum_j m_j \left(\boldsymbol{v}_i^* - \boldsymbol{v}_j^*\right) \nabla W_{ij}$ replaced with $\frac{\rho_i^* - \rho_0}{\Delta t}$, we have the following stiffness value:

$$\kappa_i = \frac{1}{\Delta t^2} \left(\rho_i^* - \rho_0\right), \tag{2.54}$$

where we could use for Jacobi iterations.

2.2.2 Viscosity Solvers

For large viscosities, implicit timestepping can be regarded as a requirement. Although work regarding implicit visocosity solvers were not abundant until this point, there were many recent advancements regarding the topic. We will introduce several types of implicit viscosity solvers that we have been investigating to use for our application. **Remark** Our own viscosity solver that can resolve large visocisity differences are also covered in later in the document.

Algorithm 2.4 DFSPH - CorrectDensityError

```
while $(\rho_{avg} - \rho_0)_{ave} > TOL_{dense}$ do
    for $Particle_i$ in all particles do
        Compute $\rho_i^*$
    end for
    for $Particle_i$ in all particles do
        $\kappa_i = \frac{\rho_i^* - \rho_0}{\Delta t^2}\alpha_i,\ \kappa_j = \frac{\rho_j^* - \rho_0}{\Delta t^2}\alpha_j$
        $\boldsymbol{v}^* \leftarrow \boldsymbol{v}^* - \Delta t \sum_j m_j \left(\frac{\kappa_i}{\rho_i} + \frac{\kappa_j}{\rho_j}\right) \nabla W_{ij}$
    end for
end while
```

2.2.2.1 Takahashi et al. (2015)

In [113], Takahashi formulated an implicit viscous step for the simulation of highly viscous fluids. Starting from the Navier-Stokes Equation:

$$\rho_i \frac{D\boldsymbol{u}_i}{Dt} = -\nabla P_i + \nabla \cdot \boldsymbol{\tau}_i + \frac{\rho_i}{m}\boldsymbol{F}_i^{ext} \tag{2.55}$$

$$\boldsymbol{\tau}_i = \mu_i \left(\nabla \boldsymbol{u}_i + (\nabla \boldsymbol{u}_i)^T\right) \tag{2.56}$$

where $\boldsymbol{\tau}_i$ is the shear stress tensor originating from viscous forces on particle i. With operator splitting, we can obtain an intermediate velocitiy field where only the non-pressure forces are applied:

$$\boldsymbol{u}_i^* = \boldsymbol{u}_i(t) + \frac{\Delta t}{\rho_i}\nabla \cdot \boldsymbol{\tau}_i^* = \boldsymbol{u}_i(t) + m\Delta t \sum_j \left(\frac{\boldsymbol{\tau}_i^*}{\rho_i^2} + \frac{\boldsymbol{\tau}_j^*}{\rho_j^2}\right)\nabla W_{ij} \tag{2.57}$$

$$\boldsymbol{\tau}_i^* = \mu_i \sum_j V_j \left(\left(\boldsymbol{u}_j^* - \boldsymbol{u}_i^*\right) \otimes \nabla W_{ij} + \left(\left(\boldsymbol{u}_j^* - \boldsymbol{u}_i^*\right) \otimes \nabla W_{ij}\right)^T\right) \tag{2.58}$$

Substituting $\boldsymbol{\tau}_i^*$ into equation 2.57:

$$\tag{2.59}$$

$$\begin{aligned}\boldsymbol{u}_i^* = \boldsymbol{u}_i(t) + m\Delta t \sum_j \Bigg(&\frac{\mu_i \sum_j V_j \left(\left(\boldsymbol{u}_j^* - \boldsymbol{u}_i^*\right) \otimes \nabla W_{ij} + \left(\left(\boldsymbol{u}_j^* - \boldsymbol{u}_i^*\right) \otimes \nabla W_{ij}\right)^T\right)}{\rho_i^2} \\ &+ \frac{\mu_j \sum_k V_k \left(\left(\boldsymbol{u}_k^* - \boldsymbol{u}_j^*\right) \otimes \nabla W_{jk} + \left(\left(\boldsymbol{u}_k^* - \boldsymbol{u}_j^*\right) \otimes \nabla W_{jk}\right)^T\right)}{\rho_j^2}\Bigg)\nabla W_{ij}\end{aligned} \tag{2.60}$$

Note that the system involves the "neighbors of the neighbors". The above system of equations is positive definite, and can be solved for $\boldsymbol{u}^*$ using a matrix solver. Refer to the original paper of Takahashi et al. ([113]) for deatils, since the author provides a more concise form that is easier to program with.

2.2.2.2 Weiler et al. (2018)

Recently, Weiler et al. introduced a physically consistent implicit viscosity solver ([122]). The fact that many previous implicit viscosity solvers meant for computer graphics applications were not physically consistent gives this approach a great advantage for solving engineering problems. The viscous forces in SPH is represented as

$$\nabla^2 \boldsymbol{v}_i = 2(d+2) \sum_j \frac{m_j}{\rho_j} \frac{\boldsymbol{v}_{ij} \cdot \boldsymbol{x}_{ij}}{||\boldsymbol{x}_{ij}||^2 + 0.01h^2} \nabla W_{ij} \tag{2.61}$$

where d is the number of spatial dimensions. This referes to Monaghan's viscosity operator found in ([87]). The velocity of the intermediate timestep can be written as

$$\boldsymbol{v}^* = \boldsymbol{v}(t) + \Delta t \frac{\mu}{\rho} \nabla^2 \boldsymbol{v}^* \tag{2.62}$$

The above system can be re-written as a system of linear equations:

$$(\boldsymbol{I} - \Delta t \boldsymbol{A}) \boldsymbol{v}^* = \boldsymbol{v}(t) \tag{2.63}$$

with block matricies $\boldsymbol{A}_{ij}$ that correspond to particle i and its neighbors j:

$$\boldsymbol{A}_{ij} = -2(d+2) \frac{\mu \bar{m}_{ij}}{\rho_i \rho_j} \frac{\nabla W_{ij} \otimes \boldsymbol{x}_{ij}}{||\boldsymbol{x}_{ij}||^2 + 0.01h^2}, \ \boldsymbol{A}_{ii} = -\sum_j \boldsymbol{A}_{ij} \tag{2.64}$$

The above system is a positive definite linear system, and can be solved with a method such as conjugate gradient.

2.3 Heat Transfer

2.3.1 Heat Equation

Writing out Fourier's Law for the diffusion flux $\boldsymbol{q}$:

$$\boldsymbol{q} = -\mathbb{K}(\nabla \theta) \tag{2.65}$$

where $\mathbb{K}$ represents the thermal conductivity coefficient tensor and θ represents the temperature. The rate of change of the temperature then can be written as

$$\frac{D\theta}{Dt} = \nabla \cdot (-\boldsymbol{q}) = \nabla \cdot (\mathbb{K}(\nabla \theta)) \tag{2.66}$$

Assuming the thermal conductivity of the material is isotropic ($\boldsymbol{\mathit{I\!K}} = k\boldsymbol{\mathit{I\!I}}$),

$$\frac{D\theta}{Dt} = k\nabla^2\theta \tag{2.67}$$

This is known as the Heat Equation.

2.3.2 SPH Discretization

In [32], the following operator was suggested for the discretization of $\nabla\left(\boldsymbol{\mathit{I\!K}}\left(\nabla\theta\right)\right)$ that can handle discrepancies between particles that have larges differences regarding their thermal conductivity k_a and k_b:

$$\left\langle k\nabla^2\theta\right\rangle = \sum_{b\in\mathcal{N}_a}\frac{4}{\rho_a}\frac{k_a k_b}{k_a+k_b}\left(\theta_a-\theta_b\right)\frac{\boldsymbol{r}_{ab}\cdot\nabla W_{ab}}{r_{ab}^2+\epsilon}v_b \tag{2.68}$$

where ϵ is a small number to avoid singularities, and k_a and k_b are respectively the thermal conduction coefficient of the two particles. It is well known that the above operator generates large errors when the uniformity of the particle distribution becomes poor. This especially applies for regions near a free-surface where the support domain of the kernel functions are no longer filled with neighboring particles.

2.3.3 Ghost Particle Boundary for the Heat Equation

2.3.3.1 Method

When using Eq. 2.68 to solve the diffusion equation, special treatment is required for the particles that are located near the free-surface in order to recover consistency when computing the Laplacian. Several methods based on correction matrices were proposed (for example, [44, 107]) as a remedy. These methods are known to provide excellent convergence properties. However, in our practical experience, the correction matrices that must be inverted at each particle is very easy to become singular throughout the simulation. For example, the correction scheme introduced in [44] requires at least 1 neighboring particle in each octant (defined by placing the indexed particle at the origin), for a particle to have a valid correction matrix.

A simple but effective approach is to employ ghost particles. This approach was successfully applied to heat conduction problems (for example, [63]) and is relatively easy to implement. In order to reduce computational costs tied to placing the ghost particles, we only place them near "rear" particles that have a particle density lower than a pre-defined value (Algorithm 2.5). The function PLACEGHOSTPARTICLES($Particle$) takes in the position of the particle, and fills empty pre-determined gridpoints that is within the support region of the kernel with ghost particles. Afterwards. ghost particles that penetrate the physical region are trimmed according to a rule based on particle density ($\sum_{b\in\mathcal{N}_a} v_b W_{ab}$). For our application, $\eta_{place} = 0.85$ and $\eta_{trim} = 1.3$ yielded satisfactory results.

2.3.3.2 Boundary Conditions

For the case of heat transfer, Dirichlet boundary conditions are obtained by fixing the temperature of the corresponding ghost particles. This is a common practice for many SPH based heat trasnfer simulations ([32, 21, 5, 63]). Neumann conditions are rather tricky to implement with SPH. In our implementation, the temperatures of the ghost particles are assigned to be the same temperature as the nearest non-ghost particle at the beginning of the timestep. This is conceptually similar to how finite difference methods when implementing insulating conditions ([72]).

Algorithm 2.5 Place / Trim Ghost Particles

```
$ghostParticles \leftarrow \varnothing$
for Each $Particle$ in $SolidParticles$ do
    if $particle\ density < \eta_{place}$ then
        placeGhostParticles($Particle$)
        Push Ghost Particle to $ghostParticles$
    end if
end for
for Each $ghostParticle$ in $ghostParticles$ do
    if $ghostParticle\ density < \eta_{trim}$ then
        Remove $ghostParticle$ from $ghostParticles$
    end if
end for
```

2.4 Thermal Stress occurring from Hot Droplet Impacting Surface

2.4.1 Introduction

Additive manufacturing (AM) methods are becoming common choice for fabricating cutting edge devices. AM broadly covers various methods that involve droplets being deposited onto a surface, such as DBM ([30]), ink-jet printing ([86], [35]), LIFT (Laser-Induced Forward Transfer, [40], [123], [2]) and Direct Ink Writing (DIW, [**?**]). In some cases, the droplets may be laden with extremely fine micro or nano materials tailored to each specific application, where good examples are given in [27] and [26]. Despite the considerable amount of recent advancement in additive manufacturing technologies, development of numerical frameworks that may be applied to such methods have been mostly stagnant.

When the targeted substrate is expected to be fragile, the integrity of the substrate during the deposition process is critical for the engineer. During the process, the induced stress within the substrate is highly dependent on both heat transfer and momentum transfer between the droplet and the substrate.

In most cases, observations can only be made after a resource consuming experimental process ([67], [22], [105], [56]). We propose a new numerical framework, based directly on the continuum equations involved in such processes. The proposed numerical framework involves a thermo-mechanical solution solved via SPH (Smoothed Particle Hydrodynamics) and a one-way coupled FEM setup which is used to compute the thermal/mechanical stress induced in the impacted solid. Advantages of such an approach includes:

- · Computationally affordable simulation of the free-surface fluid object (droplet).
- · Rapidly deforming free-surfaces of the fluid object is trvially represented.
- · Obviates the difficulties due to time step differences within the solid domain and the fluid domain.

During the process, the geometry of the fluid droplet undergoes rapid deformation. A mesh-based method (such as the Finite Element Method) will be very expensive for such problems, due to the re-meshing process that will be required after each time step. Thus, a mesh-free method such as SPH can be appropriate alternative approach. The main intention of this paper is to provide an effective numerical framework that can assist the engineer designing a droplet based manufacturing process.

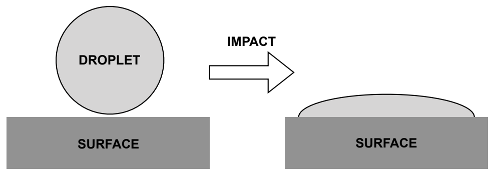


Figure 2.2: Droplet impacting a surface.

This paper is organized as follows. *First*, we state the governing equations of the problem. *Second*, a detailed explanation regarding the proposed discretization of the governing equations, for a Newtonian-type ink, is given. *Third*, a numerical convergence study is performed to justify the usage of the framework. *Fourth*, a practical numerical example is provided to illustrate the method, along with a brief comparison against a previously developed predictive model regarding heat penetration. Finally, we mention some limitations of the proposed approach, which will be investigated in future work.

Remark: *In some applications, the composition of the ink may dictate the viscous behaviour the ink. The droplet must be modeled as a Non-Newtonian fluid, in this case. While the topic regarding solving Non-Newtonian flows with SPH exceeds the scope of this paper, a brief note regarding the solution process for such prolems will be given later in this paper on.*

2.4.2 Governing Equations

2.4.2.1 Navier-Stokes Equation for Newtonian Fluids

The incompressible Navier-Stokes equations in Lagrangian formalism are the field equations:

$$\frac{D\mathbf{r}}{Dt} = \mathbf{u}, \tag{2.69}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.70}$$

$$\rho\frac{D\mathbf{u}}{Dt} = \nabla \cdot \boldsymbol{\sigma}_f + \rho\mathbf{g}. \tag{2.71}$$

where for a Newtonian fluid

$$\boldsymbol{\sigma}_f = -p\mathbf{1} + \mu\nabla^2\mathbf{u}, \tag{2.72}$$

where $\mathbf{r}$ and $\mathbf{u}$ are the position vector and the velocity vector respectively, ρ is the density of the fluid, $\mathbf{g}$ is the body force, $\boldsymbol{\sigma}_f$ is the stress tensor of the fluid, μ is the dynamic viscosity of the fluid, and p represents the pressure. Note that $\frac{D(\cdot)}{Dt}$ represents the material derivative.

Smoothed Particle Hydrodynamics (SPH), is a mesh-free numerical method that can used to discretize various continuum mechanics problems. Originally intended for astrophysical problems, SPH found many practical usage cases in analyzing fluid flows involving free-surfaces ([33], [11], [96]). The set of equations given by SPH is usually closed with a weakly compressible scheme in which the pressure term is directly correlated to the density variable through an equation of state, which is used throughout this paper :

$$p = \frac{1}{\gamma}\rho_0 c_0^2\left(\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right) \tag{2.73}$$

where ρ_0 represents the reference density, c_0 represents the speed of sound, and γ is a constant, usually chosen to be 7. This type of equation of state is typically known as a variant of Tait's equation of state, and is widely used in the SPH community.

For the problem in interest, equation (2.71) should incorporate the surface tension, where its effect may be significant in microscale droplets:

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \boldsymbol{\sigma}_f + \rho \mathbf{g} + \underbrace{\mathbf{f}^s}_{\text{Surface Tension}} . \tag{2.74}$$

There are two popular methods for implementing surface tension. In the context of SPH, one method originally proposed by [97] takes the form :

$$\rho \frac{D\mathbf{u}}{Dt} = \nabla \cdot \boldsymbol{\sigma}_f + \rho \mathbf{g} + \underbrace{\delta_s \kappa \varsigma \mathbf{n}}_{\text{Surface Tension}} , \tag{2.75}$$

where ς is the surface tension coefficient of the fluid-air interface, κ is the curvature of the free-surface and $\mathbf{n}$ is the surface normal defined at $\mathbf{x}_j$. Also, δ_s is a function such that

$$\delta_s(\mathbf{x}) = \begin{cases} 1 & \mathbf{x} \in \Omega_{FS}, \\ 0 & \mathbf{x} \in \Omega_I, \end{cases} \tag{2.76}$$

where Ω_{FS} represents the subset of the fluid domain located on the free-surface, and Ω_I represents the subset of the fluid domain located in the interior, so that $\mathbf{x} \in \Omega_{FS} \cup \Omega_I = \Omega$ and $\Omega_{FS} \cap \Omega_I = \varnothing$. This is often called the CSF surface tension model introduced in [98].

The method has several drawbacks. To accurately evaluate the surface curvature, it is advantageous that kernel truncation does not occur near the free surface. To alleviate errors due to such truncation, it is thus required to model the surrounding medium (such as air) with a separate set of SPH particles. The method also is reported to return surface curvatures with large errors ([92]), when particles are sparsely distributed (such as small detached droplets).

Another method to model surface tension is the IIF (Inter-particle Interaction Force) model, which directly adds forces between the SPH particles that represent attraction/repulsion forces due to molecular interactions ([114]). The forces only act on the particles near the free surface, since the forces cancel each other out in the interior of the fluid. Since the force is not derived from a continuum model, the force must be calibrated to bridge the scale between the SPH and the molecular particles. When calibrated, the model accurately reproduces surface tension effects, without the computation of free-surface curvature, which is prone to generate numerical errors. In [114], the authors employ the following interaction model:

$$\mathbf{f}^s_{ij} = \begin{cases} s_{ij} \cos\left(\frac{1.5\pi}{3h} r_{ij}\right) \mathbf{e}_{ji}, & r_{ij} \le h \\ 0, & r_{ij} > h. \end{cases} \tag{2.77}$$

where s_{ij} is a parameter used to calibrate the surface tension model. In this paper, we propose a modified form of this interaction model that weighs the interaction force with the mass m_i, in order to account for the difference in mass, with s as a calibration parameter and that exactly conserves linear momentum:

$$\mathbf{f}^s_{ij} = \begin{cases} s\frac{2m_i}{m_i+m_j}\cos\left(\frac{1.5\pi}{3h}r_{ij}\right)\mathbf{e}_{ji}, & r_{ij} \leq h \\ 0, & r_{ij} > h. \end{cases} \tag{2.78}$$

2.4.2.2 Thermo-Elasticity

For the purpose of this work, we assume that the deformation of the solid due to change in temperature and momentum transfer from the fluid droplet is small. We also consider the timescale of the momentum transfer process to be significantly longer than the timescale of the shockwaves traveling throughout the solid (which is directly correlated to the speed of sound in the solid). Thus, a linear-quasi-static formulation may be justified:

$$\nabla \cdot \boldsymbol{\sigma}_s = -\,\mathbf{f}_{ext}, \tag{2.79}$$

$$\boldsymbol{\sigma}_s = \lambda\left(\nabla \cdot \mathbf{v} - \alpha\Delta T\left(3 + 2\frac{G}{\lambda}\right)\right)\mathbf{1} + 2G\boldsymbol{\varepsilon}(\mathbf{v}) \tag{2.80}$$

$$\boldsymbol{\varepsilon} = \frac{1}{2}\left(\nabla\mathbf{v} + (\nabla\mathbf{v})^T\right) \tag{2.81}$$

where $\boldsymbol{\sigma}_s$ is the Cauchy stress of the solid, $\mathbf{f}_{ext}$ is the external forces exerted by the impacting fluid, $\boldsymbol{\varepsilon}$ is the resulting strain, $\mathbf{u}$ are the displacements, λ is the first Lamé Constant, G is the shear modulus, α is the thermal expansion coefficient and ΔT is the increase in temperature. We employ the Finite Element Method as described later in the paper.

2.4.3 Discretization

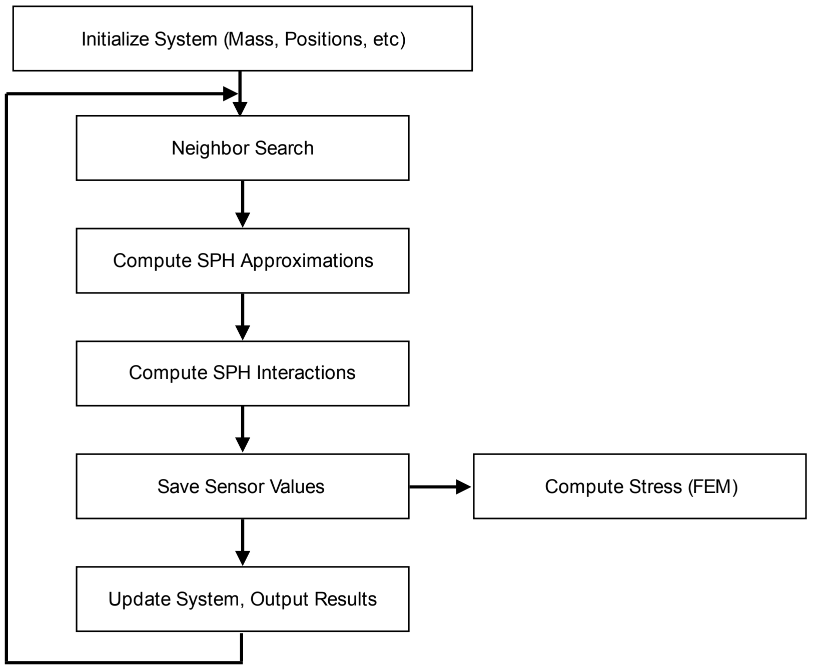

Initialize System (Mass, Positions, etc)
Neighbor Search
Compute SPH Approximations
Compute SPH Interactions
Save Sensor Values
Compute Stress (FEM)
Update System, Output Results

Figure 2.3: Algorithmic procedures of the code.

2.4.3.1 Navier-Stokes

We discretize the Navier-Stokes equation using SPH:

$$\frac{D\mathbf{u}}{Dt} = -\left\langle \frac{1}{\rho}\nabla p \right\rangle + \left\langle \frac{\mu}{\rho}\nabla^2 \mathbf{u} \right\rangle + \mathbf{g} \tag{2.82}$$

$$\frac{D\rho}{Dt} = \langle \rho \nabla \cdot \mathbf{u} \rangle \tag{2.83}$$

$$p = f_{EOS}(\rho) \tag{2.84}$$

Where $< \cdot >$ represents the discretized version of the expression using SPH, and f_{EOS} represents the equation of state. The pressure gradient contribution in (2.82) can be written in the symmetric form specifically found in [96]:

$$-\left\langle \frac{1}{\rho}\nabla p \right\rangle_i = -\sum_{j\in\mathcal{N}_i} \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W_{ij} m_j \tag{2.85}$$

Throughout the paper, the index i represents the ith SPH particle, while $j \in \mathcal{N}_i$ represents the set of neighbors of the ith particle. W_{ij} is the kernel function centered around particle i, $\nabla_i W_{ij}$ represents the gradient of the kernel, p_i is the pressure of particle i, p_j is the pressure of particle j, ρ_i is the density of particle i, ρ_j is the density of particle j, and m_i and m_j represent the mass of particle i and j, respectively. The Quintic Wendland kernel was employed due to its superior properties when used with SPH, which was rigorously verified in [79]. As in [108], the viscosity contribution is discretized as

$$\left\langle \frac{\mu}{\rho}\nabla^2 \mathbf{u} \right\rangle_i = \sum_{j\in\mathcal{N}_i} \frac{4m_j(\mu_i+\mu_j)\mathbf{r}_{ij}\cdot\nabla_i W_{ij}}{(\rho_i+\rho_j)^2(r_{ij}^2+\epsilon)}\mathbf{u}_{ij} \tag{2.86}$$

The continuity equation is discretized with the method found in [13], also known as δ-SPH :

$$\langle \rho\nabla\cdot\mathbf{u}\rangle_i = -\rho_i \sum_{j\in\mathcal{N}_i} (\mathbf{u}_j - \mathbf{u}_i)\cdot\nabla_i W_{ij} V_j + \delta h c_0 \mathcal{D}_i \tag{2.87}$$

where $\mathcal{D}_i$ is the "diffusion" term, defined as

$$\mathcal{D}_i = 2\sum_{j\in\mathcal{N}_i} \psi_{ij} \frac{\mathbf{r}_{ji}\cdot\nabla W_{ij}}{r_{ij}^2} V_j, \tag{2.88}$$

where δ is a tunable constant (usually chosen to be 0.1), h is the SPH smoothing length, c_0 is the speed of sound, and ψ_{ij} is defined as :

$$\psi_{ij} = (\rho_j - \rho_i) - \frac{1}{2}\left(\langle\nabla_1\rho_i\rangle + \langle\nabla_1\rho_j\rangle\right)\cdot\mathbf{r}_{ji}, \tag{2.89}$$

where $\langle\nabla_1(\,\cdot\,)\rangle$ represents the renormalized gradient operator. $\langle\nabla_1\rho_i\rangle$ can then be written as :

$$\langle\nabla_1\rho_i\rangle = \mathbf{B}_i \sum_{j\in\mathcal{N}(i)} (\rho_j - \rho_i)\nabla_i W_{ij} V_j \tag{2.90}$$

Here, $\mathbf{B}_i$ is the *first derivative renormalization matrix* suggested by [99], which is defined as :

$$\mathbf{B}_i = \left[-\sum_{j\in\mathcal{N}(i)} \mathbf{r}_{ij}\otimes\nabla W_{ij} V_j \right]^{-1} \tag{2.91}$$

Physically, this term adds "artificial diffusivity" to the continuity equation. Although exact conservation of mass no longer holds, the added diffusivity greatly increases stability throughout the system.

2.4.3.2 Heat Equation

We now represent the balance of energy for an *incompressible* fluid, also in Lagrangian Formalism :

$$\rho \frac{De}{Dt} = \Phi + \nabla \cdot (k \nabla T) \tag{2.92}$$

Here, e is the specific energy, k is the thermal conductivity of the medium and Φ is the dissipation function. This accounts for the dissipation that occurs in the fluid due to viscous effects, and for an *incompressible* fluid, where bulk viscosity is assumed to be negligible,

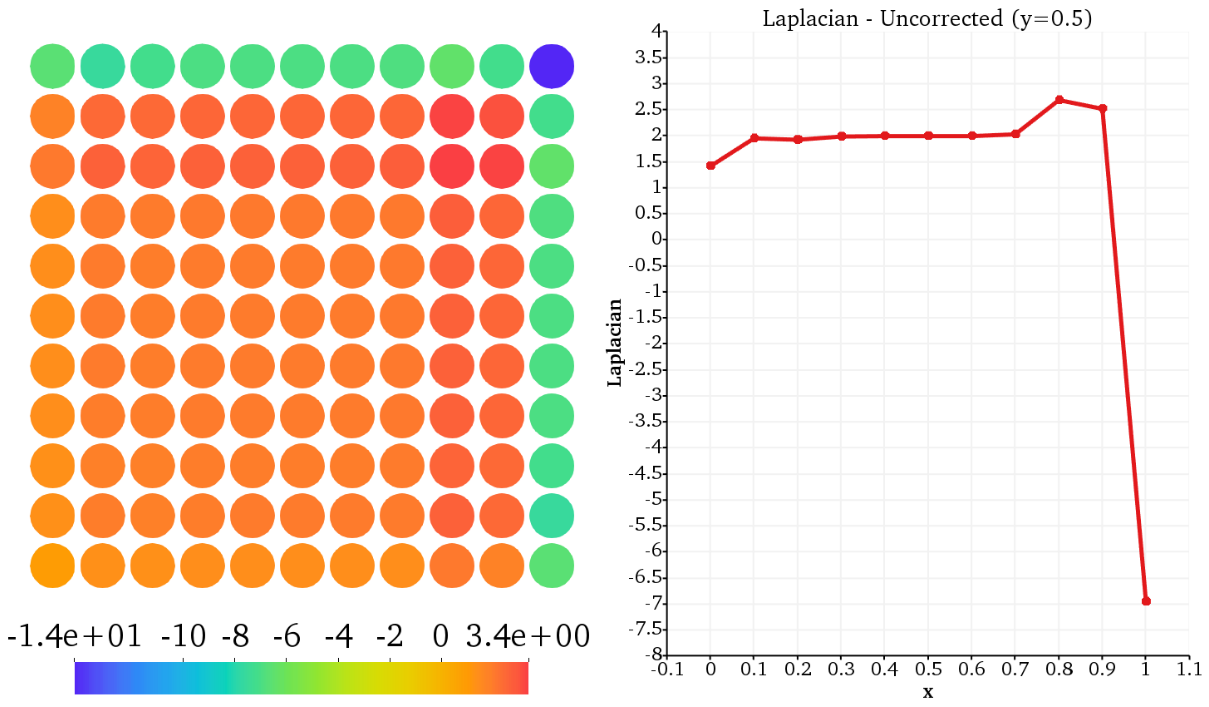
$$\Phi = 2\mu \mathbf{D} : \mathbf{D}, \tag{2.93}$$

$$\mathbf{D} = \frac{1}{2}\left(\nabla \mathbf{u} + (\nabla \mathbf{u})^T\right) \tag{2.94}$$

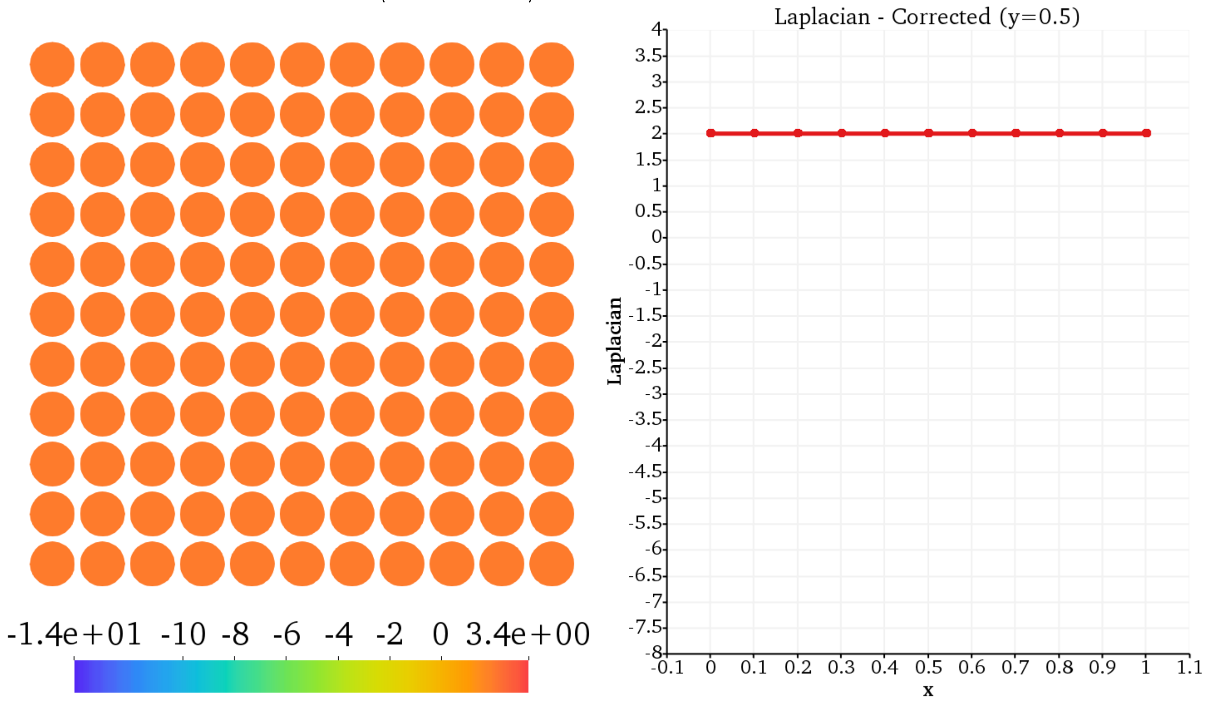
The significance of the dissipation due to the deforming fluid for our problem is low, and is excluded in our formulation. Note that this assumption may not hold in other applications (such as lubricants in bearings). Also, since the droplet is assumed to be incompressible, temperature increase due to adiabatic compression of the fluid is ignored. Equation (2.92) now simplifies to :

$$\rho c \frac{DT}{Dt} = \nabla \cdot (k \nabla T) \,. \tag{2.95}$$

Here, c represents the specific heat capacity of the medium. The energy equations must solved for both the fluid and the solid, in order to model the heat transfer between the fluid droplet and the impacted surface. The temperature of the system is solved according to the discretized version of the heat equation:



$\nabla^2 \left\langle \frac{1}{2}(x^2+y^2) \right\rangle$ (Uncorrected laplacian)



$\nabla_1^2 \left\langle \frac{1}{2}(x^2+y^2) \right\rangle$ (Corrected laplacian (Our method))

Figure 2.4: Left : Particle Configuration, Right : Laplacian @ $y = 0.5$, $h = 3\Delta x$

$$\rho c \frac{DT}{Dt} = \langle \nabla \cdot (k \nabla T) \rangle \tag{2.96}$$

In [32], the following discretization was suggested:

$$c_i \frac{DT_i}{Dt} = \sum_{j \in \mathcal{N}_{FS}(i)} \frac{4m_j}{\rho_i \rho_j} \frac{k_i k_j}{k_i + k_j} T_{ij} \frac{\mathbf{r}_{ij} \cdot \nabla_i W_{ij}}{r_{ij}^2 + \epsilon} \tag{2.97}$$

Figure 2.4 shows a brief comparison of the two Laplacian operators on a 11 × 11 cartesian particle distribution representing $x = [0, 1], y = [0, 1], f(x, y) = \frac{1}{2}(x^2 + y^2)$. The analytical Laplacian of $f(x, y)$ should be 2, in this case. It is a well known that when using conventional SPH operators (such as the one above), the accuracy of the Laplacian severely deteriorates and fails to recover the correct value near the free surface (Figure 2.4), as mentioned in the original paper ([31]). Such effects are usually ignored with justification ([110]) when discretizing the viscosity in (2.82), but accurately computing the heat transfer inside our defined geometry is an important aspect which should be addressed.

Recent advancement in SPH discretization techniques ([119], [44]) suggest the following renormalization tensor for second-derivatives, which recovers first-order accuracy of the Laplacian for an *arbitrary* distribution of SPH particles :

$$\begin{aligned} &\left\langle \nabla_1^2 f_i \right\rangle = \\ &2\mathbf{L}_i : \left[\left(\sum_{j \in \mathcal{N}_{FS}(i)} \mathbf{e}_{ij} \otimes \nabla_i W_{ij} \right) \left(\frac{f_i - f_j}{r_{ij}} - \mathbf{e}_{ij} \cdot \nabla_1 f_i \right) V_j \right] \end{aligned} \tag{2.98}$$

Using the above correction, it is no longer required that the particles must retain the predefined stencil (dx) to reach convergence. Here, $\mathbf{L}_i$ is the correction tensor for the second derivative, and is the solution to the following set of equations :

$$-\delta^{mn} = \sum_{j \in \mathcal{N}_{FS}(i)} (A^i_{kmn} e^k_{ij} + r^m_{ij} e^n_{ij})(L^{op}_i + e^o_{ij} (\nabla_i W_{ij})^p V_j); \tag{2.99}$$

$$A^{kmn}_i = G^{kq}_i \sum_{j \in \mathcal{N}_{FS}(i)} (r^m_{ij} r^n_{ij} (\nabla_i W_{ij})^q V_j) \tag{2.100}$$

With m, n, o, p, q being the indices used for the Einstein notation. In 3D, the above system is a 6 × 6 matrix where the components of the solution vector correspond to the entries of $\mathbf{L}_i$ (which is in result, a function of $\mathbf{A}_i = A^{kmn}_i \mathbf{e}_k \otimes \mathbf{e}_m \otimes \mathbf{e}_n$ and $\boldsymbol{\delta}_i = \delta^{mn}_i \mathbf{e}_m \otimes \mathbf{e}_n$).

Using this correction, we thus modify equation (2.97) to :

$$c_i \frac{DT_i}{Dt} = \sum_{j \in \mathcal{N}_{FS}(i)} \left[\frac{4}{\rho_i} \left(\frac{k_i k_j}{k_i + k_j} \right) (\mathbf{L}_i : \mathbf{e}_{ij} \otimes \nabla_i W_{ij}) \left(\frac{T_i - T_j}{r_{ij}} - \mathbf{e}_{ij} \cdot \nabla_i T_i \right) V_j \right] \tag{2.101}$$

here, k_i or k_j is defined to be the thermal conductivity of the fluid or solid, depending on which domain the SPH particles is defined to be within.

Although the above correction seems promising for many situations, the method carries a drawback. For a correction to exist, a particle requires at least one neighboring particle in each quadrant with respect to itself. Thus, in 3D, at least 8 neighbors are required. A simple workaround for particles where the correction is not defined, is to resort back to (2.97).

2.4.3.3 Finite Element Method

Previous effort regarding modeling solids via SPH were made in [9], [49], [120]. The main weakness of such formulations were on stability and consistency. Since the solid domain of our problem is assumed to involve a stiff material (with stiffness being on the order of $100GPa$s), the fluid domain and the solid domain would require timesteps with large discrepancies, and renders modeling the whole domain with SPH difficult.

The Finite Element Method (FEM) is a well-established method for discretizing PDEs, and is adequate for modeling solid mechanics problems; thus is used to model the solid substrate in our problem. Each timestep, after the temperature field and the velocity / position field is updated, a linear-quasi-static thermo-elasticity problem is solved within the solid domain. Writing the weak-form of the proposed problem :

$$\int_{\Omega_s} \boldsymbol{\sigma}_s(\mathbf{v}) : \boldsymbol{\varepsilon}(\bar{\mathbf{v}})\, d\Omega = \int_{\Omega_s} \mathbf{f} \cdot \bar{\mathbf{v}}\, d\Omega + \int_{\Gamma_s} \boldsymbol{\sigma}_s \mathbf{n} \cdot \bar{\mathbf{v}} d\Gamma, \tag{2.102}$$

$$\boldsymbol{\sigma}_s(\mathbf{v}) = \lambda \left(\nabla \cdot \mathbf{v} - \alpha \Delta T \left(3 + 2\frac{G}{\lambda} \right) \right) \mathbf{1} + 2G\boldsymbol{\varepsilon}(\mathbf{v}) \tag{2.103}$$

$$\boldsymbol{\varepsilon}(\mathbf{v}) = \frac{1}{2} \left(\nabla \mathbf{v} + (\nabla \mathbf{v})^T \right), \tag{2.104}$$

Where $\mathbf{v}$ is the solution (displacement field) of the problem, $\bar{\mathbf{v}}$ is an arbitrary test function defined over the mesh, and ΔT (temperature change), $\boldsymbol{\sigma}_s \mathbf{n}|_{\Gamma_s}$ (traction boundary conditions) are values defined over Ω_s and Γ_s respectively, and are precalculated from the SPH scheme. For simplicity, the formulation is chosen to be quasi-static. This is under the assumption that the impact speed is comparably smaller than the speed of the shockwaves travelling through the solid.

The weak form above is then discretized via an isoparametric mapping with linear shape functions, and is represented as a set of linear equations involving the displacement of the nodes :

$$\mathbf{K}\mathbf{v} = \mathbf{f} \tag{2.105}$$

where $\mathbf{v}$ is the displacement field, $\mathbf{K}$ is the stiffness matrix and $\mathbf{f}$ is computed with the inputs (temperature, force) subjected to the solid mesh. The displacement field is then post-processed to obtain the desired stress distributions. Detailed solution procedures are beyond the scope of this paper, and we refer to well-known literature such as [131] and [14] for details.

2.4.4 Numerical Algorithm

The above equations are solved explicitly for each field, similar to the formulations found in [133]. The main numerical procedures for each time-step follow as (also depicted in Figure 2.3) :
[(i)]

1. *Neighbor Searching.*

2. *Solve the Momentum equation for the fluid with SPH.*

3. *Solve the Heat equation for the fluid / solid with SPH.*

4. *Solve the mechanical response of the solid with FEM.*

2.4.4.1 SPH-FEM 1-way Coupling

As seen in 2.3, the information obtained from the SPH solution is fed into the FEM solver, in a 1-way coupled fashion. This is obtained by placing a grid of SPH particles on top of the finite element nodes (Figure 2.5) where relevant. The FEM solver takes the inputs (pressure field / viscous forces / temperature) and then solves the thermoelasticity problem stated earlier.

2.4.4.2 Neighbor Searching

Many particle methods, including SPH, rely on *Neighbor Searching* algorithms before computing the interactions between the particles. Here, we employ the neighbor search algorithm found in [69], which is a variant of the *Compact Hashing* algorithm developed in [60].

2.4.4.3 Particle Interaction

In order to compute the correction matrices and the diffusion term used in δ-SPH, several passes (the normalized density gradients, such as $\langle \nabla_1 \rho_i \rangle$ and the normalization matrices $\mathbf{L}_i$ and $\mathbf{B}_i$ need to be computed first) are conducted over the particles. The linear algebra library [50] was used to solve the 6×6 matrices involved in the second-derivative renormalization matrices ($\mathbf{L}_i$).

2.4.4.4 SPH-FEM Coupling and Thermoelasticity

A finite element mesh is defined over the solid domain, where the nodes coincide with the SPH particles. "Sensor Particles" that are located near the surface of the solid directly feed the observed temperature and forces to the finite element nodes. This input can be regarded as the corresponding nodal loading. The open-source FEM library by [4] was used to define and solve a thermoelasticity problem.

2.4.4.5 Initial Placement

To remove the *aliased* edges of a simple-rejection-sampled Cartesian placement, the initial particle representation of the fluid droplet was created using a 3D Poisson sampling algorithm by [55].

The SPH particles representing the solid domain was placed using a simple Cartesian placement. Before the first timestep, the mass of each SPH particle is assigned using the number density :

$$m_i = \frac{\rho_0}{\sum_{j \in \mathcal{N}(i)} W_{ij}} \tag{2.106}$$

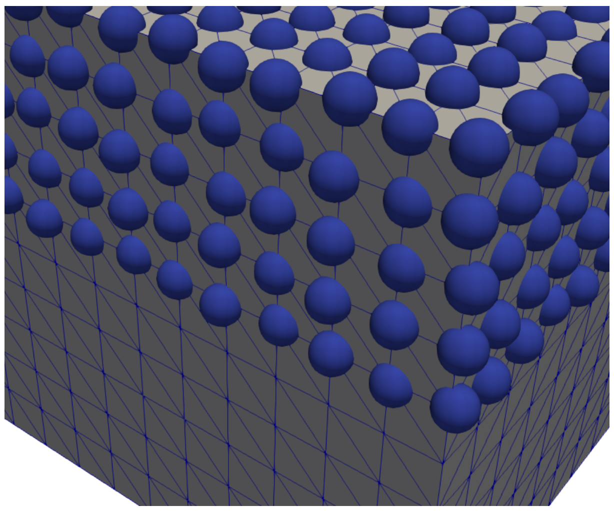


Figure 2.5: SPH particles placed on top of Finite Element nodes

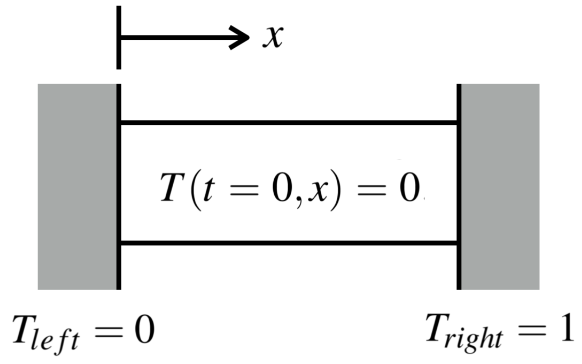
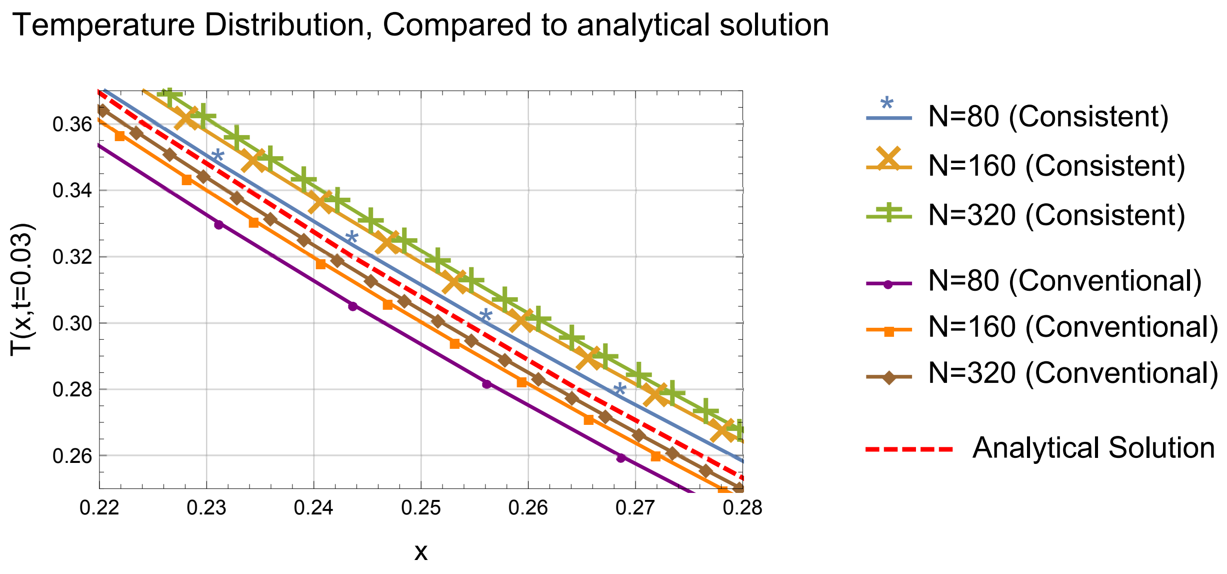
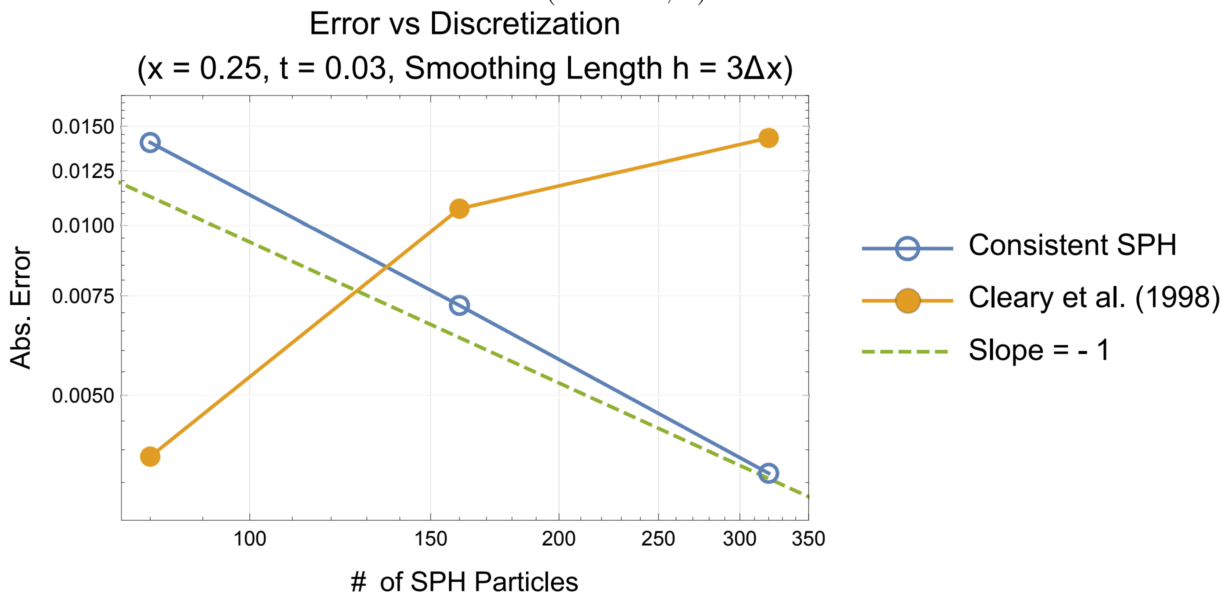


Figure 2.6: 1D Transient Heat Transfer Problem Schematic



$T(t = 0.03, x)$



Error Analysis

Figure 2.7: 1D Transient Heat Transfer Problem @ $t = 0.03$

2.4.5 Convergence Study

2.4.5.1 Heat Equation with Consistent Operators

To highlight the effectiveness of the consistent operators, we compare the results to the inconsistent one by solving a (dimensionless) 1D transient heat equation. A uniform rod of length $L = 1$ and initial temperature distribution $T(x, t = 0) = 0,\ x \in [0, 1]$ is placed between two walls with constant temperature $T_{left} = 0, T_{right} = 1$ (Fig. 2.6). This is an interesting problem, since at $(x,t) = (0,0)$, the solution is discontinuous.

The analytical time-dependent solution for this problem is given by:

$$T_{analyitcal}(x,t) = 1 - Erf(\frac{x}{2\sqrt{t}}) \tag{2.107}$$

We compare the solutions obtained by using the formulation given in [32] and [44] by solving the above transient 1D heat transfer prolem until $t = 0.03$, (Fig. 2.7), with 3 different discretization scales (80 nodes, 160 nodes, 320 nodes) and with smoothing radius $3\Delta x$. We observe convergence from the corrected operators, in contrast to the conventional operators. This is a well known issue when solving the Heat Equation with SPH, since the conventional operators assume sufficient amount of smoothness of the solution.

2.4.5.2 SPH-FEM Momentum Coupling

We now justify the approach regarding heat-transfer and mechanical coupling between SPH/FEM.

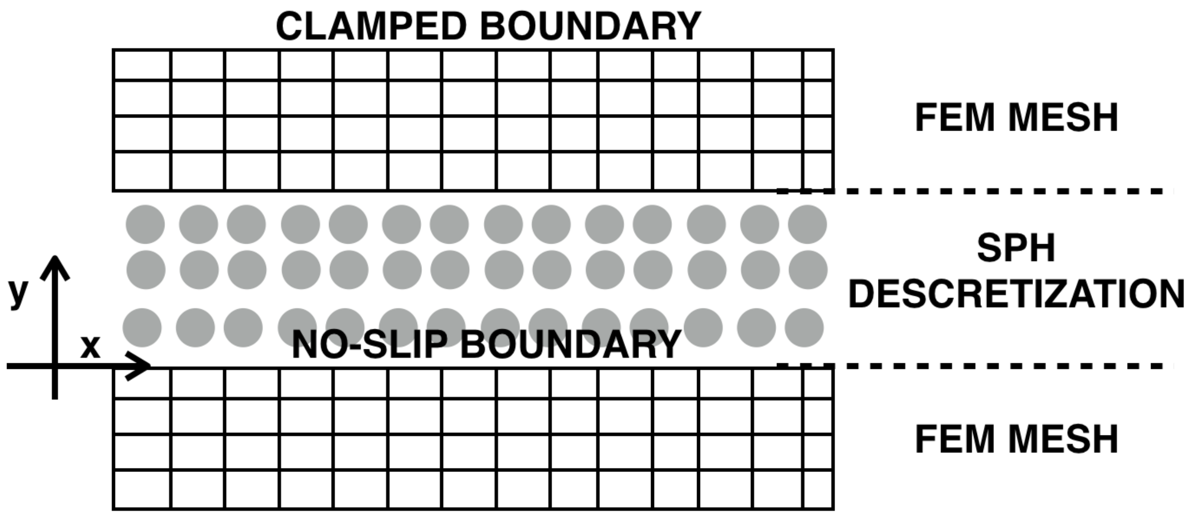


Figure 2.8: Poiseuille flow shear transfer test

We verify the 1-way momentum coupling scheme described earlier, by considering a steady-state 2D Poiseuille flow. Finite elements were used to represent the solid boundaries, while SPH particles represented the Newtonian fluid (Figure). Viscosity of the fluid will cause shear on the walls, where the analytical steady-state velocity profile is given by

$$v_x(y) = (H^2/2\mu)(-\frac{dP}{dx})(\frac{y}{H}(1 - \frac{y}{H})) \tag{2.108}$$

Where H is the distance between the two plates. The analytical shear stress at the wall is given by

$$\tau_{xy} = \mu \frac{\partial v_x}{\partial y} \tag{2.109}$$

For the test, we use $H = 1, \mu = 0.001, \frac{\partial p}{\partial x} = 10$, with periodic boundary conditions on the inlet / outlet. The thickness of the wall modeled with finite elements was chosen to be $T = 0.25$. All the y-direction DOFs on the finite element mesh was enforced to be 0, in order to rule out any volumetric deformations. Then, the average stress throughout the finite element domain was computed by

$$\sigma_{ave} = \int_V \sigma(x, y) dV \tag{2.110}$$

and was compared (Figure 2.9) against the analytical value ($\tau_{xy} = 0.02$ from Eq. 2.108). The results indicate that the coupling technique is suitable to model consistent momentum transfer between the solid and the fluid.

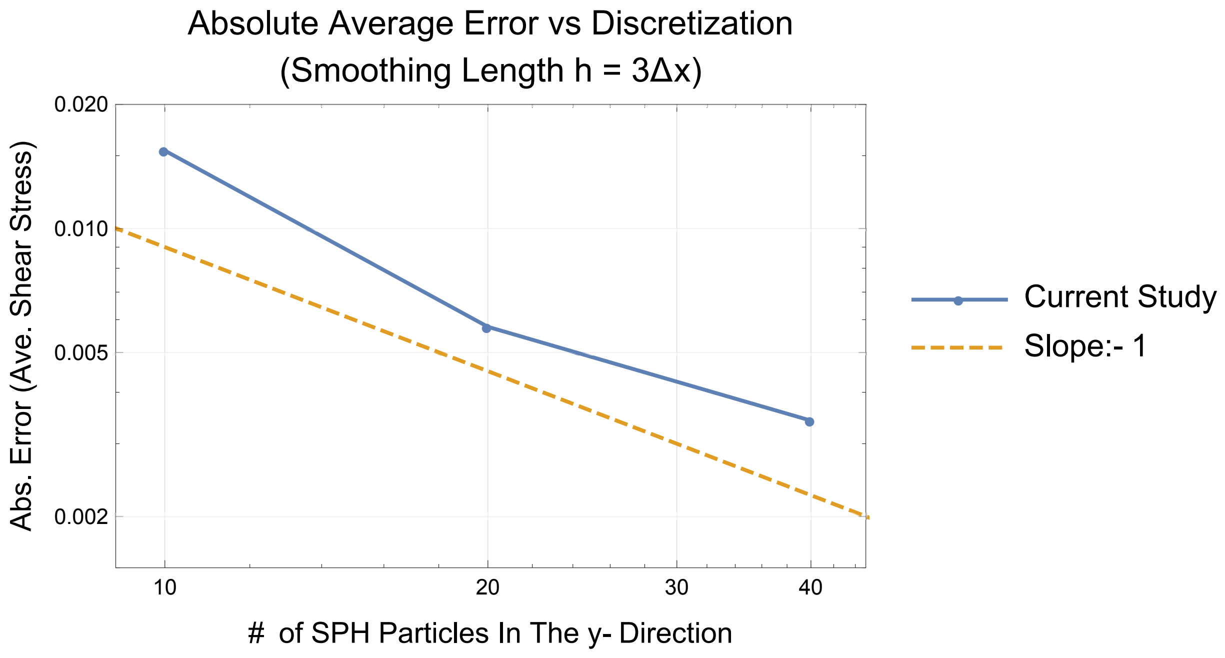


Figure 2.9: Average Shear Stress Error

2.4.6 Numerical Example

2.4.6.1 Thermal Droplet Impact Case

A numerical example involving a droplet with high-temperature impacting a low-temperature surface was performed. Approximately 2×10^5 particles were used to represent the droplet and the solid domain. The solid substrate was clamped down with

Dirichlet boundary conditions on its 4 sides, and was assumed to be thermally insulated from the surrounding air.

$t/t_c = 0.25$
$t/t_c = 0.75$
$t/t_c = 1.25$
$t/t_c = 1.75$

Figure 2.10: Evolution of a fluid droplet impacting the substrate.

$t/t_c = 1.75$

Figure 2.11: Temperature distribution on the substrate surface at $t/t_c = 1.75$.

2.4.6.2 Fluid Properties

- $\rho_0 = 1000$ $[kg/m^3]$ (Rest density)
- $T_{0,f} = 100$ $[C]$ (Initial temperature)
- $c_f = 4000$ $[J/K\,kg]$ (Specific heat)
- $k_f = 0.6$ [W/m K] (Thermal Conductivity)
- $\mu_f = 0.0003$ $[Pa\,s]$ (Dynamic viscosity of fluid)
- $c_0 = 500$ $[m/s]$ (Sound of speed)
- $D = 0.1$ $[mm]$ (Diameter of droplet)
- $v_0 = 50$ $[m/s]$ (Impact velocity)
- $s = 1\,10^6 [N]$ (Surface Tension Coefficient)
- $t_c = 2.0 \cdot 10^{-6}$ $[s]$ (Characteristic Time)

Note : The below parameters represent the properties of water, in general.

2.4.6.3 Solid Properties

- $L = 2mm \times 2mm \times 0.26mm$
- $\rho_s = 3950$ $[kg/m^3]$ (Specific mass)
- $T_{0,s} = 20$ $[C]$ (Initial temperature)
- $\mu_s = 0.0003$ $[Pa\,s]$ (Dynamic viscosity of solid/fluid interface)
- $c_s = 900$ $[J/kg\,K]$ (Specific heat)

- $k_s = 10\ [W/m\,K]$ (Thermal Conductivity)
- $E = 300\ [GPa]$
- $\nu = 0.21$ (Poisson's Ratio)
- $\alpha = 8.1 \cdot 10^{-6}$ (Thermal expansion coefficient)

Note : The below parameters broadly represent the properties of typical Alumina.

2.4.6.4 General Parameters

- $\Delta x = 2.5 \cdot 10^{-6}\ [m]$ (Particle stencil)
- $h = 3\Delta x\ [m]$ (Smoothing length)
- $\gamma = 7$ (Tait EOS parameter)
- $dt = 2.5 \cdot 10^{-10}\ [s]$ (Timestep)

2.4.6.5 Analysis

Here, we have demonstrated an useful example regarding the usage of the computational framework we have introduced in this paper. In detail, figure 2.10 shows the temperature of the droplet along with the evolution of the droplet geometry. Figure 2.11 shows the temperature distribution of the substrate at $t/t_c = 1.75$.

Initially, the heat transfer is confined within the small contact patch of the droplet, where the temperature arises locally. The timescale of the impact limits the increase of temperature of the solid at deeper depths, as seen in figure 2.13. We observe that for the timescale of interest, the heat transfer occuring between the fluid and the solid dominates the temperature increase in the solid.

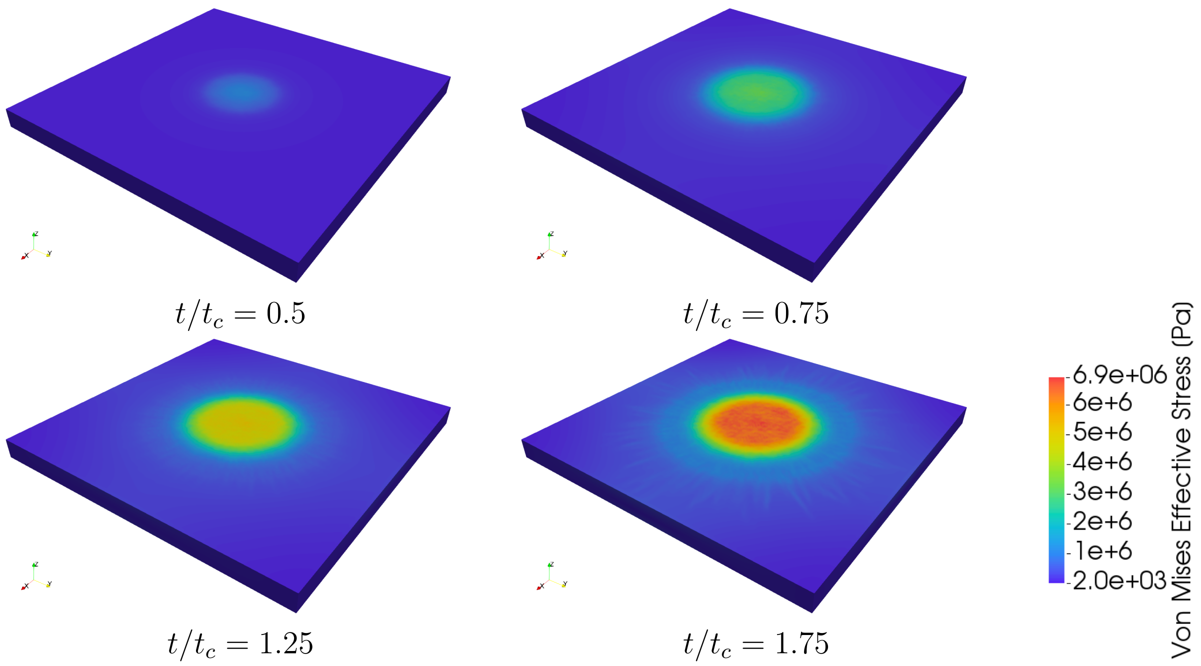
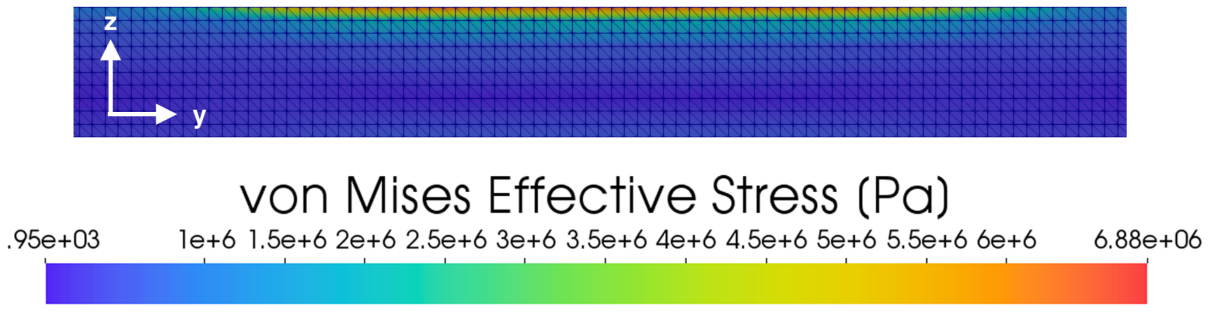


Figure 2.12: Stress (Von-Mises) distribution on the substrate surface.

The stress within the substrate increases with time (figure 2.12), as the droplet breaks spreads over the surface. The stress is largest near the surface of the substrate, as seen in figure 2.13. This is due to the large heat and momentum transfer between the droplet and the surface at the contact patch (2.12). The contact patch between the droplet and the substrate expands radially.



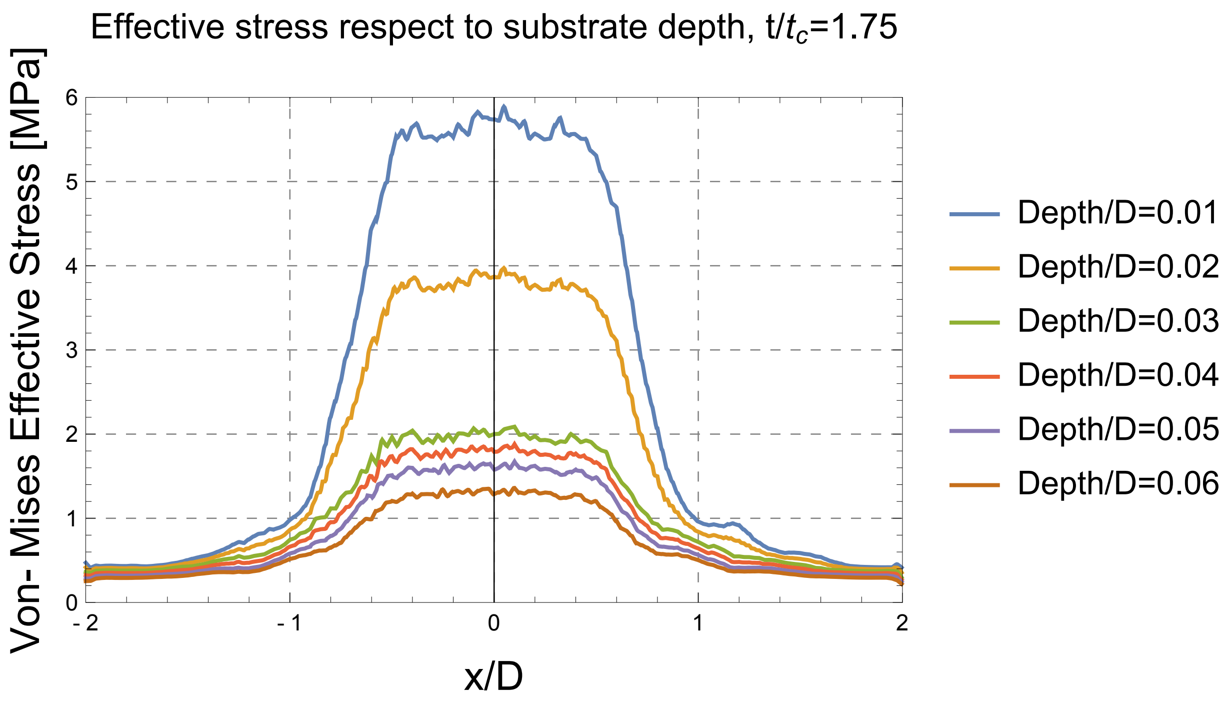


Figure 2.13: Stress on the substrate with respect to depth ($t/t_c = 1.75, x = 0$).

We focus on time $t/t_c = 1.75$, where the droplet is fully deposited onto the substrate (as shown in figure 2.10). Figure 2.11 shows the surface temperature distribution, which indicates that the majority of the heat transfer is made through the initial contact patch with diameter of roughly 0.6 times the initial droplet diameter ($0.06mm$). The effective (Von-Mises) stress experienced by the substrate along the cross-section is depicted in figure 2.13, which shows the concentrated stress near the surface (2.15).

For the homogeneous material (Alumina) in consideration for the current example, we can predict that no type of failure will occur, since the stress levels shown in 2.13 are well below the known failure criterion. On the other hand, for tailor-made substrates obtained by mixing specific functional microparticles into a binding matrix, this type of concentrated stresses confined to the surface will usually result in a delamination-type defect ([**?**]). This is a well known micromechanical pheonomena, and is due to the stress concentrations that occur between the interface of the functional particle and the matrix. The stress concentration factors may become extremely high in certain situations, as seen in ([**?**]). Thus, for such applications, it is critical to first find the stress concentration factor by experiments or computational methods via RVE (Representative Volume Element) methods before making a conclusion regarding the design of the deposition process.

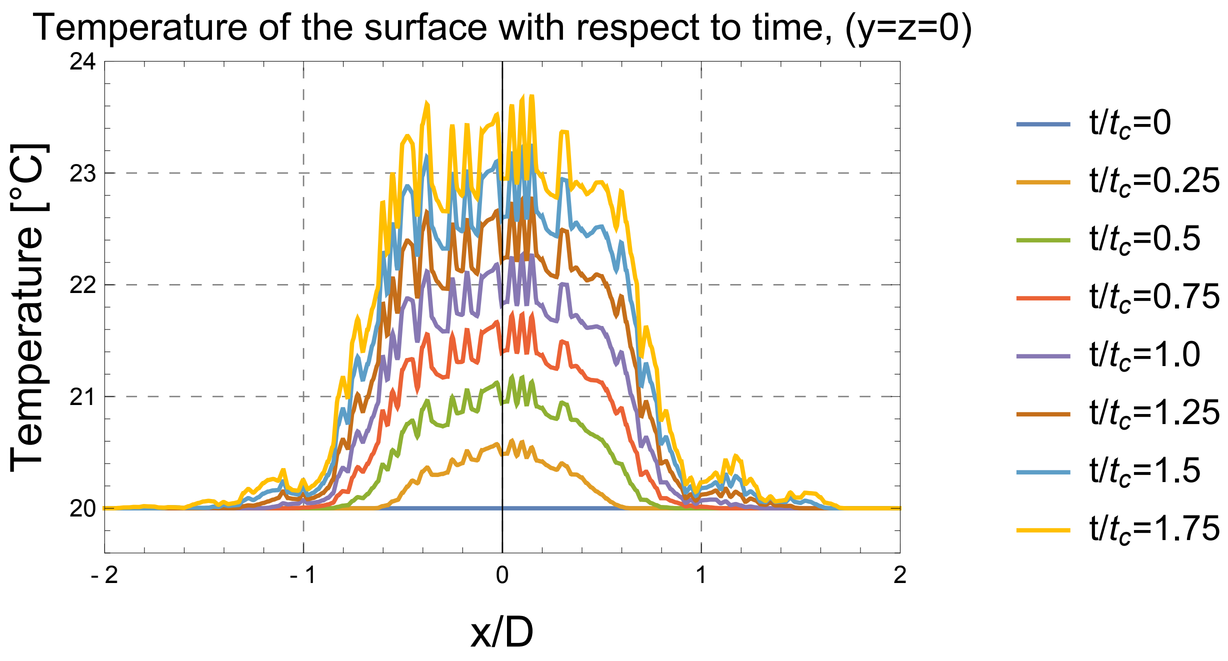
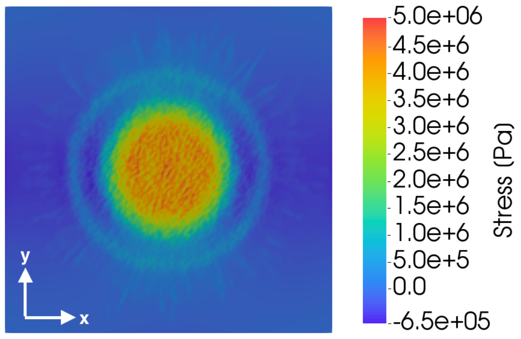
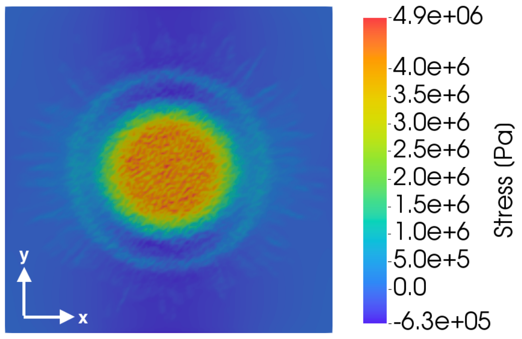


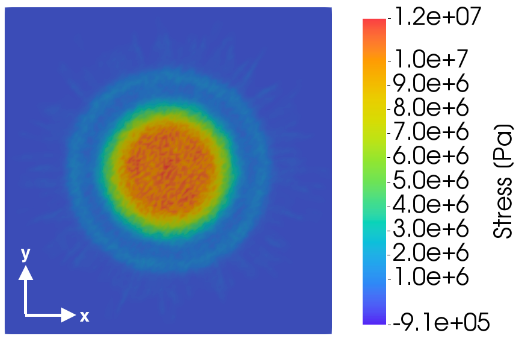
Figure 2.14: Temperature evolution on substrate surface, with respect to time.



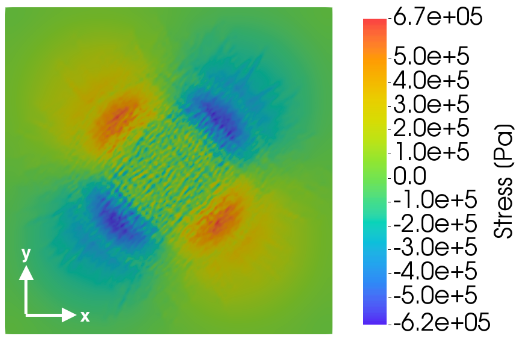
σ_{xx}



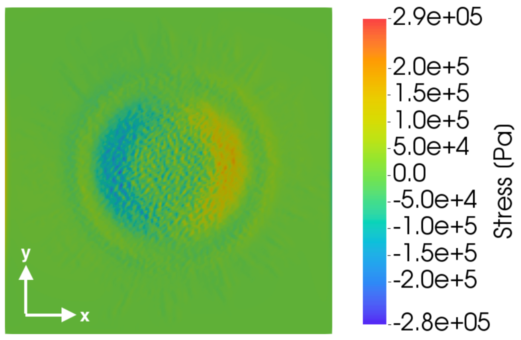
σ_{yy}



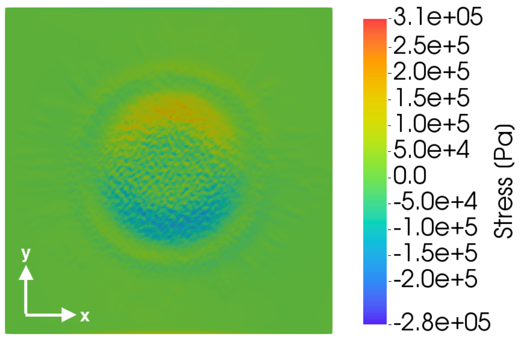
σ_{zz}



σ_{xy}



σ_{xz}



σ_{yz}

Figure 2.15: Stress Components , $t/t_c = 1.75$

2.4.7 Discussion

2.4.7.1 Summary

An enhanced formulation of SPH (Smoothed Particle Hydrodynamics) was coupled with a finite element solver to deliver a practical tool that can quickly simulate the thermo-mechanical stresses occurring on the substrate of such processes. The modified corrected SPH scheme robustly handles various starting configurations, including free surfaces, while providing improved accuracy over its conventional counterpart. The coupling between SPH and FEM was achieved via staggering of SPH particles on top of finite element nodes, where the physics observed by the SPH particles were directly enforced on the corresponding finite element node. To summarize, the resulting framework allows:

- · Fast and accurate modeling of Newtonian fluid droplets.
- · Thermal and mechanical coupling between fluids with free-surfaces and solids.
- · Removal of timestep limitations imposed due to different CFL conditions for the solid and fluid.

As noted earlier, the above aspects are all very useful features to have when conducting numerical experiments on droplet deposition manufacturing techniques. Mesh-based approaches require frequent re-meshing, and may become computationally extremely expensive.

Overall, a rapid simulation framework that can provide useful insight regarding the induced mechanical stresses for droplet based additive manufacturing was developed. With an example demonstrating a droplet impacting a solid surface, we have shown that the framework is capable of providing useful insights that may be sometimes hard to obtain by purely experimental methods. Justification of the framework was also given by performing convergence analysis on the thermal / mechanical coupling methods. Since the numerical results are derived from a direct numerical simulation of the proposed continuum problem, the various material/mechanical parameters used in the simulation can be altered freely to suit the users needs.

2.4.7.2 Limitations

Although the method assumes the fluid model to be homogeneous, this is not necessarily the case for many emerging additive manufacturing methods. In some applications, specifically tailored micro/nano materials are mixed with a base fluid ([27],[26],[86],[116],[45]). Often, these are referred to as “functionalized” inks.

In many cases, mixing functional powders into the solvent forces the ink to have shear-thinning properties, along with a yielding stress. The mechanical behavior of such inks can usually be modeled with Non-Newtonian fluid models, one being the popular Herschel-Bulkley fluid model. For this model, the shear stress of a fluid can be written as

$$\tau = \tau_y + K\dot{\gamma}^n \tag{2.111}$$

where τ_y is the yield stress, $\dot{\gamma}$ is the effective shear rate, and K, n are parameters to be fitted. Many previous work on DIW (Direct Ink Writing) fabrication methods have successfully employed ([**?**], [**?**]) the Herschel Bulkley model to explain the mechanical behaviour of these inks. In numerical implementations, the "Regularized Herschel Bulkley" model is frequently used, where a shear-rate dependent effective viscosity is used for the viscosity term in the Navier-Stokes equations:

$$\tau = 2\mu_{eff}(\dot{\gamma}), \quad \mu_{eff}(\dot{\gamma}) = \begin{cases} k\dot{\gamma_0}^{n+1} + \tau_0\dot{\gamma_0}^{-1} & \dot{\gamma} \leq \dot{\gamma}_0 \\ k\dot{\gamma_0}^{n-1} + \tau_0\dot{\gamma_0}^{n-1} & \dot{\gamma} \geq \dot{\gamma}_0 \end{cases} \tag{2.112}$$

For unyielded regions within the fluid, the high effective viscosity will make the fluid behave similar to a rigid body. For yielded regions, the viscosity will be several magnitueds smaller compared to the unyielded regions. Therefore, with an explicit time-stepping approach with SPH, the problem becomes very difficult to solve, and an implicit time stepping scheme must be implemented to alleviate the difficulties. To our best knowledge, no such schemes have been devised specifically for Non-Newtonian fluids with SPH. This topic is currently under investigation by the authors.

Ackowledgements Thanks to Prof. Antonio Souto Iglesias and Dr. Henning Wessels for providing many fruitful discussions. This material is based upon work supported by the National Science Foundation under award number 1547112.

2.5 Finite Deformation Elasticity: A Total Lagrangian Approach

2.5.1 Introduction

Smoothed Particle Hydrodynamics originates from the early work by Monaghan ([94, 88, 89][94, 88, 89]) that was proposed to solve astrophysical problems. Recently, its framework has been widely adopted for solving complex free-surface fluid flows. Since the particle set does not require a pre-defined mesh, it is very well suited for such problems. The effort to increase convergence, stability and accuracy in the field of fluid mechanics resulted in several pioneering methods that may be classified as "Weakly Compressible" SPH (WCSPH), such as δ-SPH ([12, 71]) or Riemann SPH ([71, 62, 25, 51]). It was proven that such state-of-the-art SPH solvers can be as accurate as Finite Element and Finite Difference Method based fluid solvers. However, compared to the massive success it has experienced for free-surface fluid problems, SPH based solid mechanics solvers remain relatively unpopular due to several reasons. At a first glance, solving solid mechanics problems with SPH seems straight-forward, since one may think that all that is needed is to compute the Cauchy Stress ($\boldsymbol{\sigma}$) occurring on each particle, and then adjust the acceleration of the particle accordingly by computing the divergence of $\boldsymbol{\sigma}$. In many cases, the Cauchy-Stress is integrated in time via an objective stress rate rule (for example, [76, 74, 10]). However, it should be noted that:

- Field approximations are inconsistent near the free surface due to kernel truncation.
- Unlike fluids, inconsistent approximations of the stress tensor of the divergence of the stress tensor can cause immediate instability.
- The popular Jaumann Rate is “inaccurate” for stretch-dominated deformations.

To obviate these issues, rather than employing the updated Lagrangian / Eulerian formulation, a total Lagrangian formulation can be employed to describe the deformation of the body. This approach allows us to obviate the usage of stress rates or incremental deformation gradients, and allow us to directly compute the Second Piola-Kirchhoff stress. For example, in [19], a total Lagrangian form of the deformation gradient was used in conjunction with an explicit time stepping scheme to avoid the so-called “Tensile Instabilities” that arise from standard SPH formulations. To the author’s best knowledge, this seems to be the earliest, extensive research regarding the topic. [103] formalized this new approach towards SPH, where it was successfully applied to numerous elasto-plasticity problems. Later, [121] also summarized the total-Lagrangian approach, with emphasis on various kernel correction methods. An incremental deformation gradient was computed each timestep, which was used to track the overall deformation gradient. Recently, [53] successfully applied the method for solving Fluid-Structure interaction problems.

In this section, our goal is to contribute towards the development of this relatively new approach. Specifically, it will include aspects that to our best knowledge, has not been discussed before:

- Formulation of the multiplicative deconstruction of the deformation gradient with SPH.
- Finite-Deformation thermoelastic constitutive modeling.
- Ghost particle treatment algorithm for thermal boundary conditions.
- Monolithic, implicit time-stepping scheme for SPH-based thermo-elasticity.

2.5.2 SPH Discretization in Current Configuration

The angled bracket notation was employed to represent SPH approximations. For the interpolation of the physical property and its spatial gradient at position $\boldsymbol{r}_a$, we have

$$\psi\left(\boldsymbol{r}_a\right) \approx \langle\psi\rangle_a = \sum_{b\in\mathcal{N}_a} \psi_b v_b W\left(\boldsymbol{r}_{ab}, h\right) \tag{2.113}$$

$$\nabla\psi\left(\boldsymbol{r}_a\right) \approx \langle\nabla\psi\rangle_a = \sum_{b\in\mathcal{N}_a} \psi_b v_b \nabla W_{ab} \tag{2.114}$$

where ψ_a, ψ_b are the physical values stored at particle a, b and W_{ab} represents the kernel centered around $\boldsymbol{r}_a$ and its evaluation at $\boldsymbol{r}_b$, and ∇W_{ab} represents it's gradient evaluated at $\boldsymbol{r}_b$. It is well known that the consistency of the gradient approximation given by eq. 2.114 deteriorates near free-surface boundaries. For the purpose of pursuing solid mechanics, even a slight amount of error in the deformation gradient near the boundaries result in immediate instability. The correction tensor which was first suggested in the pioneering work by Johnson and Beissel([66]) is used as a remedy for the issue:

$$\nabla\psi\left(\boldsymbol{r}_a\right) \approx \langle\nabla\psi\rangle_a = \left(\sum_{b\in\mathcal{N}_a} \psi_b v_b \nabla W_{ab}\right) \cdot \boldsymbol{B}_a \tag{2.115}$$

with

$$\boldsymbol{B}_a = \left(\sum_{b\in\mathcal{N}_a} v_b \left(\boldsymbol{r}_b - \boldsymbol{r}_a\right) \otimes \nabla W_{ab}\right)^{-1} \tag{2.116}$$

For the Laplace operator ($\nabla^2\psi$), we use a ghost particle approach which is explained later in this section. We also define the correction tensor in the reference configuration:

$$\boldsymbol{B}_{a,o} = \left(\sum_{b\in\mathcal{N}_{a,o}} v_{b,o} \left(\boldsymbol{r}_{b,o} - \boldsymbol{r}_{a,o}\right) \otimes \nabla W_{ab,o}\right)^{-1} \tag{2.117}$$

Where the subscript o is used to describe the variable in its reference state.

2.5.3 Total Lagrangian Formalism for Finite Deformation Thermoelasticity

2.5.3.1 Constitutive Relations

In thermoelasticity, it is common to introduce a fictional intermediate state where it is assumed that only deformations due to thermal effects have taken place. This is often called the multiplicative decomposition of the deformation gradient. The total deformation gradient is then represented as

$$\boldsymbol{F} = \boldsymbol{F}_e \cdot \boldsymbol{F}_\theta \tag{2.118}$$

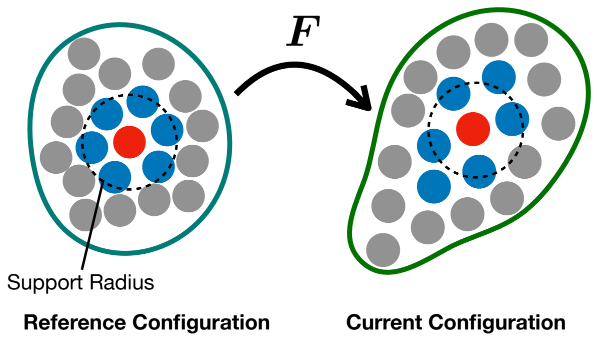


Figure 2.16: Initial Neighbors ($\mathcal{N}_0$) are stored within memory and used throughout the simulation. Note that the initial neighbors may leave the kernel support radius in the current configuration.

where $\boldsymbol{F}_e$ and $\boldsymbol{F}_\theta$ inlcudes only the deformations due to elasticity and thermal expansion effects, respectively. Assuming the material in interest possesses an isotropic swelling coefficient matrix, the thermal deformation gradient can be readily computed:

$$\boldsymbol{F}_\theta(\theta) = v(\theta)\,\mathbf{1} = (1+\alpha(\theta))\,\mathbf{1} \tag{2.119}$$

where $\alpha(\theta)$ is a temperature dependent thermal expansion parameter. Then, the elastic portion of the deformation gradient is computed with

$$\boldsymbol{F}_e = \boldsymbol{F}\cdot\boldsymbol{F}_\theta^{-1} \tag{2.120}$$

In order to define the corresponding stress, we first define the Helmholtz free-energy per unit mass as a function dependent on the temperature (θ) and deformation $\left(\boldsymbol{E} = \frac{1}{2}\left(\boldsymbol{F}^T\cdot\boldsymbol{F} - \mathbf{1}\right)\right)$

$$\psi = \hat{\psi}(\theta, \boldsymbol{E}) = \psi_e(\boldsymbol{E}_e, \theta) + \psi_\theta(\theta) \tag{2.121}$$

Note that we have split the function into two parts, where ψ_e is an isotropic function of the elastic strain $\boldsymbol{E}_e$ and temperature θ, and ψ_θ is a function that should be obtained and adjusted via experimental data. One possible option for ψ_e is a function derived from the well-known iso-thermal finite elasticity free energy function. The constitutive relations for the Second Piola-Kirchhoff stress may then be established:

$$\boldsymbol{S} = \frac{\rho_o}{v^2}\frac{\partial\psi_e}{\partial\boldsymbol{E}_e} = v\boldsymbol{S}_e \tag{2.122}$$

where the relationship regarding the density is defined by $\rho_o = v^3\rho_\theta$. Assuming we choose ψ_e to be a quadratic function of the elastic strain,

$$\rho_e\psi_e = \frac{1}{2}\lambda(\theta)\left(tr\boldsymbol{E}_e\right)^2 + \mu(\theta)\,\boldsymbol{E}_e:\boldsymbol{E}_e \tag{2.123}$$

the corresponding Second Piola-Kirchhoff stress is derived for the concentration-dependent fourth order stiffness tensor $\boldsymbol{\mathcal{C}}_e = \hat{\boldsymbol{\mathcal{C}}}_e\left(\theta\right)$:

$$\boldsymbol{S}_e = \boldsymbol{\mathcal{C}}_e : \boldsymbol{E}_e,\ \boldsymbol{\mathcal{C}}_e = \hat{\boldsymbol{\mathcal{C}}}_e\left(\theta\right) = \lambda\left(\theta\right)\mathbf{1}\otimes\mathbf{1} + 2\mu\left(\theta\right)\mathbb{1} \tag{2.124}$$

Where $\mathbf{1}\otimes\mathbf{1}$ is the fourth order unit tensor and $\mathbb{1}$ is the tensor given by $\mathbb{1}_{ijkl} = \frac{1}{2}\left(\delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk}\right)$. Further assuming that the Lamé parameters can be considered constant with respect to the concentration θ, we have for the overall Second Piola Kirchhoff stress:

$$\boldsymbol{S} = \lambda_o\left(tr\boldsymbol{E}\right)\mathbf{1} + 2\mu_o\boldsymbol{E} - 3\alpha\left(\theta - \theta_o\right)\kappa_o\mathbf{1} \tag{2.125}$$

2.5.4 Total Lagrangian SPH discretization for solid mechanics

Algorithm 2.6 Compute the deformation gradient

```
function ComputeDeformationGradient($Particle_a$)
    for $Particle_b$ in $\mathcal{N}_{a,o}$ do                ▷ Loop over initial neighbors
        $\boldsymbol{F}_a += (\boldsymbol{x}_b - \boldsymbol{x}_a)v_{b,o} \otimes \nabla W_{ab,o}$
    end for
    $\boldsymbol{F}_a := \boldsymbol{F}_a \cdot \boldsymbol{B}_{a,o}$                ▷ Apply Renormalization return $\boldsymbol{F}_a$
end function
```

Algorithm 2.7 Compute accelerations

```
function ComputeAccelerations($Particle_a$)
    for $Particle_b$ in $\mathcal{N}_{a,o}$ do                ▷ Loop over initial neighbors
        $\boldsymbol{a}_a += \frac{1}{\rho_o}\left(\boldsymbol{F}_a \cdot \boldsymbol{S}_a \cdot \boldsymbol{B}_{a,o} \cdot \nabla W_{ab,o} - \boldsymbol{F}_b \cdot \boldsymbol{S}_b \cdot \boldsymbol{B}_{b,o} \cdot \nabla W_{ba,o}\right)$
    end for
end function
```

The balance of linear momentum gives us the acceleration of each SPH particle in the reference configuration:

$$\nabla_0 \cdot \boldsymbol{P} + \rho_0\boldsymbol{b} = \rho_0\boldsymbol{a} \tag{2.126}$$

Which requires us the computation of the first Piola-Kirchhoff stress $\boldsymbol{P} = \boldsymbol{F}\cdot\boldsymbol{S}$, which is also a function of the deformation gradient (eq. 2.125).

At the beginning of the simulation, initial neighbor maps are constructed in order to take the gradient of a physical field in the current configuration. Each particle "remembers" its original coordinates throughout the simulation. For example, the gradient of a physical quantity ψ with respect to its original configuration is computed as

$$\nabla_o \psi\left(\boldsymbol{r}_a\right) \approx \left\langle \nabla_o \psi \right\rangle_a = \left(\sum_{b \in \mathcal{N}_{a,o}} \psi_b v_{b,o} \nabla W_{ab,o} \right) \cdot \boldsymbol{B}_{a,o} \tag{2.127}$$

Where $\mathcal{N}_{a,o}$ represents the neighbors of particle a at the reference configuration, $v_{b,o}$ represents the original volume of particle b, $\nabla W_{ab,o}$ is the kernel gradient defined between a and b at the reference configuration and $\boldsymbol{B}_{a,o}$ is the renormalization tensor for particle a defined at the same configuration. Following this formalism, the deformation gradient of particle a may be directly computed using its definition:

$$\boldsymbol{F}_a = \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{X}}|_{\boldsymbol{x}=\boldsymbol{r}_a} = \left(\sum_{b \in \mathcal{N}_{a,o}} \left(\boldsymbol{x}_b - \boldsymbol{x}_a\right) v_{b,o} \otimes \nabla W_{ab,o} \right) \cdot \boldsymbol{B}_{a,o} \tag{2.128}$$

The divergence of $\boldsymbol{P}$ then may be obtained:

$$\nabla_o \cdot \boldsymbol{P} = \rho_o \left(\sum_{b \in \mathcal{N}_{a,o}} \frac{v_{a,o} v_{b,o}}{m_a} \left(\boldsymbol{F}_a \cdot \boldsymbol{S}_a \cdot \boldsymbol{B}_{a,o} \cdot \nabla W_{ab,o} - \boldsymbol{F}_b \cdot \boldsymbol{S}_b \cdot \boldsymbol{B}_{b,o} \cdot \nabla W_{ba,o} \right) \right) \tag{2.129}$$

Remark: There are multiple formulations that may be employed to compute the divergence in the reference configuration; the above popular formulation was first introduced in [47].

2.5.4.1 The choice of divergence operators

There exists several possible divergence operators we can choose from when computing the divergence $\nabla \cdot \boldsymbol{\sigma}$ or $\nabla_0 \cdot \boldsymbol{P}$. To introduce a few,

1. $$\left\langle \nabla \cdot \boldsymbol{\sigma} \right\rangle_a = \rho_a \sum_{b \in \mathcal{N}_a} m_a \left(\frac{\boldsymbol{\sigma}_a}{\rho_a^2} + \frac{\boldsymbol{\sigma}_b}{\rho_b^2} \right) \cdot \nabla \tilde{W}_{ab}$$

2. $$\frac{1}{\rho_a} \left\langle \nabla \cdot \boldsymbol{\sigma} \right\rangle_a = \frac{1}{m_a} \sum_{b \in \mathcal{N}_a} v_a v_b \left(\boldsymbol{\sigma}_a + \boldsymbol{\sigma}_b \right) \cdot \nabla \tilde{W}_{ab}$$

3. $$\left\langle \nabla \cdot \boldsymbol{\sigma} \right\rangle_a = \frac{2}{m_a} \sum_{b \in \mathcal{N}_a} v_a v_b \boldsymbol{\sigma}_a \cdot \nabla \tilde{W}_{ab}$$

And for computing the divergence in the reference configuration :

1. $$\boldsymbol{a}_a = \frac{1}{\rho_a} \left\langle \nabla_X \cdot \boldsymbol{P} \right\rangle_a = \frac{1}{m_a} \sum_{b \in \mathcal{N}_a^0} V_a V_b \left(\boldsymbol{P}_a + \boldsymbol{P}_b \right) \cdot \nabla_X W_{ab}$$

Here, the notation $\nabla\tilde{W}_{ab}$ represents the normalized kernel gradient $\nabla\tilde{W}_{ab} = \boldsymbol{B}_a \cdot \nabla W_{ab}$. The same applies to $\nabla_0 \cdot \boldsymbol{P}$ only that $\nabla\tilde{W}_{ab}$ is replaced with $\nabla_0\tilde{W}_{ab}$ and $\sum\limits_{b\in\mathcal{N}_a}$ is replaced with $\sum\limits_{b\in\mathcal{N}_a^0}$, and v_a is replaced with v_a^0. It is usally preferred to use symmetrized operators such as 1 and 2, since this guarantees momentum conservation (the force exerted to particle a from particle b equals the force exerted to particle b by particle a). Formulation 3 assumes localized stresses, and results in non-physical deforamtions (although frequently employed in the computer graphics community due to it's stability).

2.5.5 Implicit Timestepping

Algorithm 2.8 Fixed Point Iteration

```
function FixedPointIteration
    $\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^0 = \begin{bmatrix} \boldsymbol{x}^{t} \\ \boldsymbol{v}^{t} \end{bmatrix}$    ▷ Initialize Guess
    while Norm($\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^{k+1} - \begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^{k}$) $> TOL$ do
        for $Particle_a$ in $SolidParticles$ do    ▷ Compute $\boldsymbol{\Psi}^{k+1}$
            ComputeAccelerations($Particle_a$)
            ComputeDiffusion($Particle_a$)
        end for
        for $Particle_a$ in $SolidParticles$ do    ▷ Update Guess
            $\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^{k+1} = \boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^{k}\right)$
        end for
    end while
end function
```

Implementation of the implicit Trapezoidal method based on a fixed-point iteration scheme for a set of nonlinear system of equations is proposed to solve for the next time step (Algorithm 2.9). Although the convergence rate for fixed-point iterations are known to be slow, it provides a relatively easy way of solving for the next timestep. Note that for particle systems, obtaining the Hessian (for Newton-Raphson iterations) is extremely computationally expensive.

For simplicity, we first consider a system where the acceleration of each particle only depends on $\boldsymbol{x}$ and $\boldsymbol{v}$. For the displacement field and the acceleration field we have:

$$\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{t} \\ \boldsymbol{v}^{t} \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{v}^{t+1} \\ \boldsymbol{a}^{t+1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}^{t} \\ \boldsymbol{a}^{t} \end{bmatrix} \right) \tag{2.130}$$

Where $\begin{bmatrix} \boldsymbol{x}^t \\ \boldsymbol{v}^t \end{bmatrix}$ represents the vector of displacements and velocities of the whole particle system at time t. Plugging in $[\boldsymbol{v}^{t+1}] = [\boldsymbol{v}^t] + \frac{\Delta t}{2}\left([\boldsymbol{a}^{t+1} + \boldsymbol{a}^t]\right)$, we have for $[\boldsymbol{x}^{t+1}]$:

$$\left[\boldsymbol{x}^{t+1}\right] = \left[\boldsymbol{x}^t\right] + \frac{\Delta t}{2} \cdot \left(\left[\boldsymbol{v}^t\right] + \frac{\Delta t}{2} \cdot \left(\left[\boldsymbol{a}^{t+1}\right] + \left[\boldsymbol{a}^t\right] \right) + \left[\boldsymbol{v}^t\right] \right) = \left[\boldsymbol{x}^t\right] + \Delta t \cdot \left[\boldsymbol{v}^t\right] + \frac{(\Delta t)^2}{4} \left(\left[\boldsymbol{a}^{t+1}\right] + \left[\boldsymbol{a}^t\right] \right) \tag{2.131}$$

we have

$$\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^t \\ \boldsymbol{v}^t \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{v}^t + \frac{\Delta t}{2} \cdot (\boldsymbol{a}^{t+1} + \boldsymbol{a}^t) \\ \boldsymbol{a}^{t+1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}^t \\ \boldsymbol{a}^t \end{bmatrix} \right) \tag{2.132}$$

According to our constitutive model, the acceleration vector $\boldsymbol{a}^{t+1}$ is a nonlinear function of the velocity and the position vectors at time $t+1$:

$$\left[\boldsymbol{a}^{t+1}\right] = \boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} \right) \tag{2.133}$$

Writing everything out again,

$$\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^t \\ \boldsymbol{v}^t \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{v}^t + \frac{\Delta t}{2} \cdot \left(\boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} \right) + \boldsymbol{a}^t \right) \\ \boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} \right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}^t \\ \boldsymbol{a}^t \end{bmatrix} \right) \tag{2.134}$$

Notice that the only unknowns are $\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}$. Thus, we have a system of nonlinear equations. Defining the function

$$\boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} \right) \overset{def}{=} \begin{bmatrix} \boldsymbol{x}^t \\ \boldsymbol{v}^t \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{v}^t + \frac{\Delta t}{2} \cdot \left(\boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} \right) + \boldsymbol{a}^t \right) \\ \boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} \right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}^t \\ \boldsymbol{a}^t \end{bmatrix} \right) \tag{2.135}$$

we have the form suitable for a fixed-point iteration solution process :

$$\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} = \boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix} \right) \tag{2.136}$$

Thus, we can iterate with a fixed-point iteration method to solve for $\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}$:

$$\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^{k+1} = \boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^{k}\right) \tag{2.137}$$

Where k is the iteration counter. We simply use $\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \end{bmatrix}^{0} = \begin{bmatrix} \boldsymbol{x}^{t} \\ \boldsymbol{v}^{t} \end{bmatrix}$ as the initial "guess".

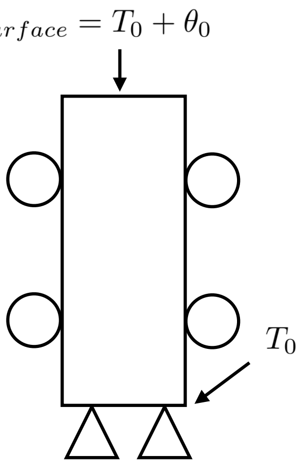
To include heat transfer, the effects of the temperature needs to be considered. Thus, the temperature vector unknown vector ($\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \\ \boldsymbol{\theta}^{t+1} \end{bmatrix}$). Similarly, we perform fixed-point iteration on the system $\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \\ \boldsymbol{\theta}^{t+1} \end{bmatrix}^{k+1} = \tilde{\boldsymbol{\Psi}}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \\ \boldsymbol{\theta}^{t+1} \end{bmatrix}^{k}\right)$ with

$$\tilde{\boldsymbol{\Psi}}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \\ \boldsymbol{\theta}^{t+1} \end{bmatrix}\right) \overset{def}{=} \begin{bmatrix} \boldsymbol{x}^{t} \\ \boldsymbol{v}^{t} \\ \boldsymbol{\theta}^{t} \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{v}^{t} + \frac{\Delta t}{2} \cdot \left(\tilde{\boldsymbol{\xi}}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \\ \boldsymbol{\theta}^{t+1} \end{bmatrix}\right) + \boldsymbol{a}^{t}\right) \\ \tilde{\boldsymbol{\xi}}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \\ \boldsymbol{\theta}^{t+1} \end{bmatrix}\right) \\ \bar{\boldsymbol{\xi}}\left(\begin{bmatrix} \boldsymbol{x}^{t+1} \\ \boldsymbol{v}^{t+1} \\ \boldsymbol{\theta}^{t+1} \end{bmatrix}\right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}^{t} \\ \boldsymbol{a}^{t} \\ \dot{\boldsymbol{\theta}}^{t} \end{bmatrix}\right) \tag{2.138}$$

where $\tilde{\boldsymbol{\xi}}\left(\begin{bmatrix} \boldsymbol{x}^{t} \\ \boldsymbol{v}^{t} \\ \boldsymbol{\theta}^{t} \end{bmatrix}\right) = \boldsymbol{a}^{t}$ and $\bar{\boldsymbol{\xi}}\left(\begin{bmatrix} \boldsymbol{x}^{t} \\ \boldsymbol{v}^{t} \\ \boldsymbol{\theta}^{t} \end{bmatrix}\right) = \dot{\boldsymbol{\theta}}^{t}$.

2.5.6 Numerical Expamples

Several numerical simulations were conducted to demonstrate the proposed numerical method. All simulations were done with the following parameters:



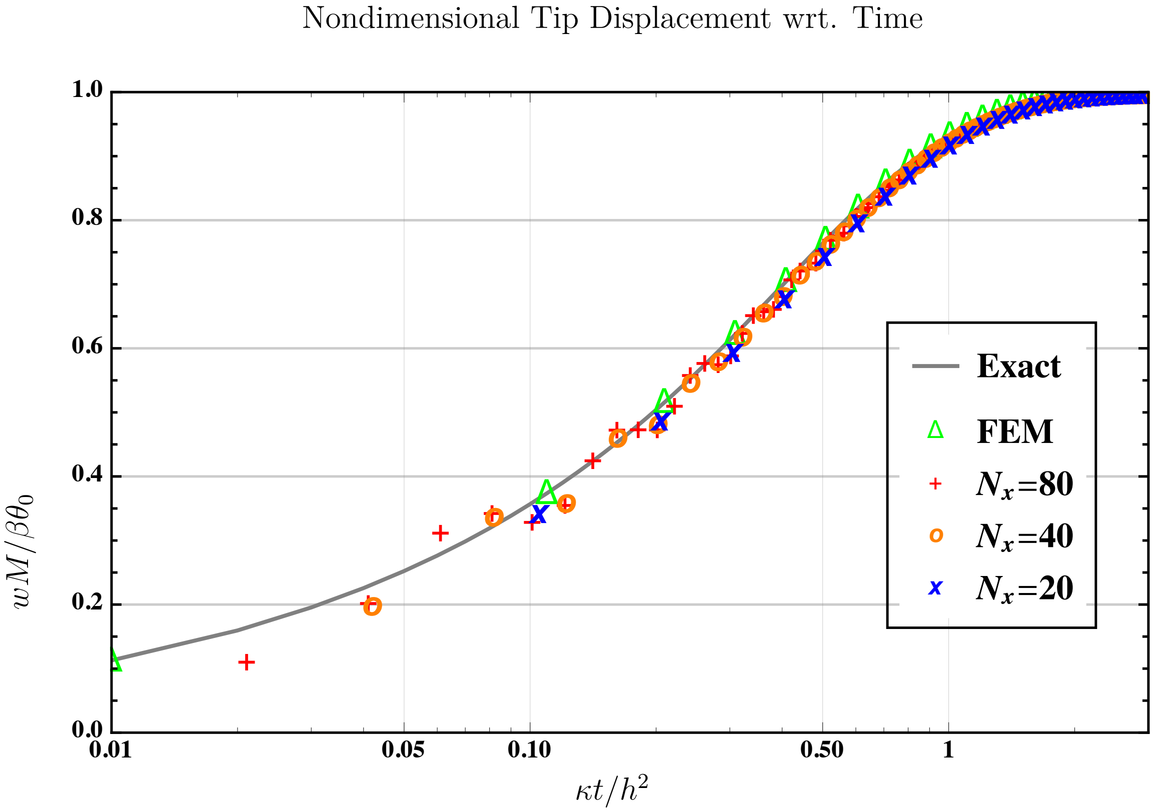


Figure 2.17: Transient Danilovskaya Problem $\lambda = 5 \times 10^5, \mu = 2.5 \times 10^5$. w represents the displacement of the tip.

2.5.7 Transient Heat Transfer on an Expanding Rod

We first consider the where a sudden temperature field is applied to the tip of a rod where its other faces are insulated, often known as the Danilovskaya problem [39]. An analytical solution in exists in the Laplace transformation space, and is often used to benchmark numerical thermoelastic schemes ([24, 115, 23]). A thermoelastic rod is constrained at its base, while the sides are only allowed to move in the y-direction. Initially, the temperature is T_0 throughout. A temperature of $T_{surface} = T_0 + \theta_0$ is prescribed as a boundary condition on the top part of the structure, allowing the block to expand accordingly. The sides and the base is thermally insulated. Assuming the displacements are moderate, it is possible to obtain an analytical solution for the y-displacement field of the tip and the temperature field within the rod via the inverse-Laplace transformation of the following:

$$\bar{\theta} = \left(\frac{\theta_0}{s}\right) \frac{cosh(\eta x)}{cosh\,(\eta h)},\ \bar{u} = \left(\frac{\theta_0}{\eta s}\right)\left(\frac{\beta}{M}\right) \frac{sinh\,(\eta z)}{cosh\,(\eta h)}$$

where

$$\eta = \sqrt{\frac{s}{\kappa}},\ M = \frac{E(1-\nu)}{(1+\nu)\,(1-2\nu)},\ \beta = E\alpha/\,(1-2\nu)\,.$$

Here, $\kappa = \frac{k}{\rho c}$ is the thermal diffusivity and s is the parameter for the Laplace transformation. Figure 2.17 shows the numerical solution obtained via our method. Although some oscillations during the start-up phase are noticed due to the instantaneous temperature application of θ_0, the overall solution is in good agreement with the analytical solution and the FEM solution.

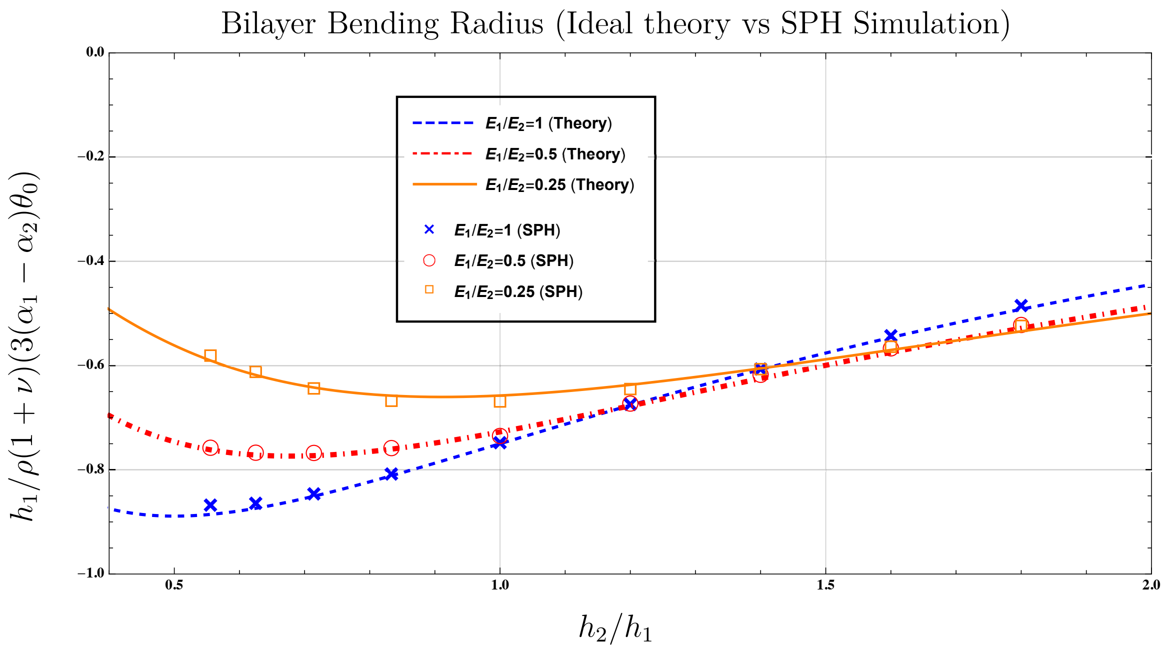


Figure 2.18: Bilayer Bending Radius (Ideal theory vs SPH Simulation)

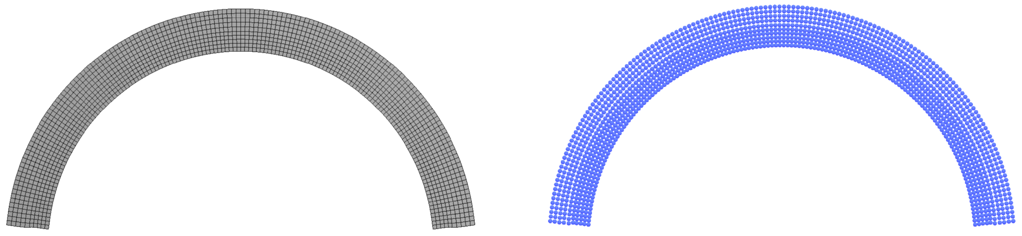


Figure 2.19: Comparison of FEM and SPH for the Bilayer Bending problem. Shown: Final deformed state for data point corresponding to $E_1/E_2 = 0.5$, $h_1/h_2 = 1$ in Figure 2.18.

2.5.8 Expansion Induced Curvature in a Free Standing Bimaterial

One application that exploits thermal expansion is thermally actuated switches. In [129], an analytical approach was used to derive the idealized curvature of a uniformly-heated bi-layered structure. The setup was replicated within our framework using two types of SPH particles, each with varying mechanical / thermal properties. The resulting curvature, assuming the difference in curvature between the upper/lower strip is small, is:

$$\frac{h_1}{\rho} = \frac{2\frac{E_2}{E_1} \cdot \frac{h_2}{h_1}\left(1 + \frac{h_2}{h_1}\right)\left(3\left(\alpha_1 - \alpha_2\right)\theta_0\right)}{1 + \left(\frac{E_2}{E_1} \cdot \frac{h_2^2}{h_1^2}\right)^2 + 2 \cdot \frac{h_2}{h_1} \cdot \frac{E_2}{E_1}\left(2 + 3\frac{h_2}{h_1} + \frac{h_2^2}{h_1^2}\right)} \tag{2.139}$$

Which is also equivalent to the relation stated in [117] and [84]. Where h_1 and h_2 represent the height of the strips. Multiple simulation cases were considered by varying h_1/h_2 and E_1/E_2, and was compared to the above analytical curvature in Figure 2.19. Overall, they were in good agreement.

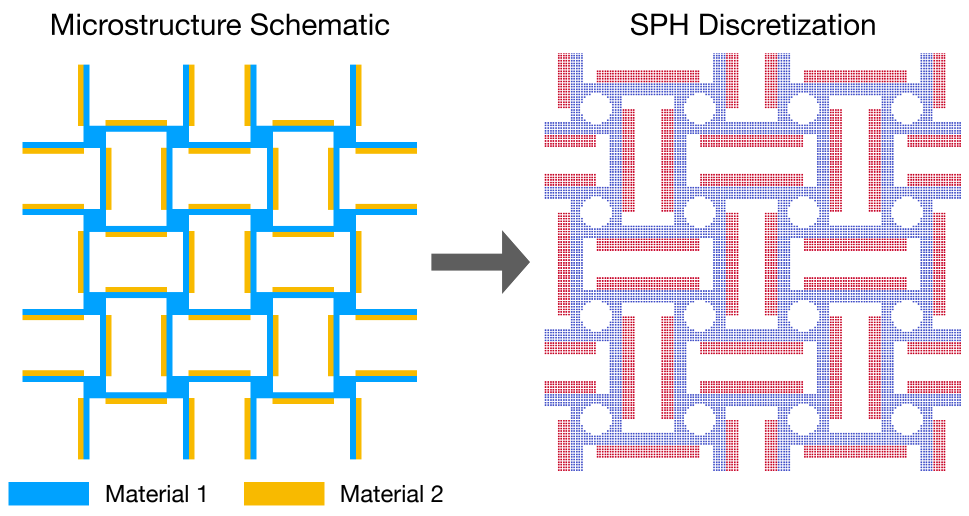


Figure 2.20: A 2D version of the metamaterial structure proposed in [124]. Material 1 is a more compliant material with a smaller thermal expansion coefficient, while material 2 is a stiffer material with a larger thermal expansion coefficient.

	Material 1	Material 2
Thermal Diffusivity	$2.0 \times 10^{-5} m^2/s$	$2.0 \times 10^{-5} m^2/s$
First Lamé Modulus	$1.0 \times 10^5 Pa$	$1.0 \times 10^6 Pa$
Shear Modulus	$0.5 \times 10^5 Pa$	$0.5 \times 10^6 Pa$
Coefficient of Thermal Expansion	2.0×10^{-5}	2.0×10^{-4}

Table 2.1: Physical Parameters

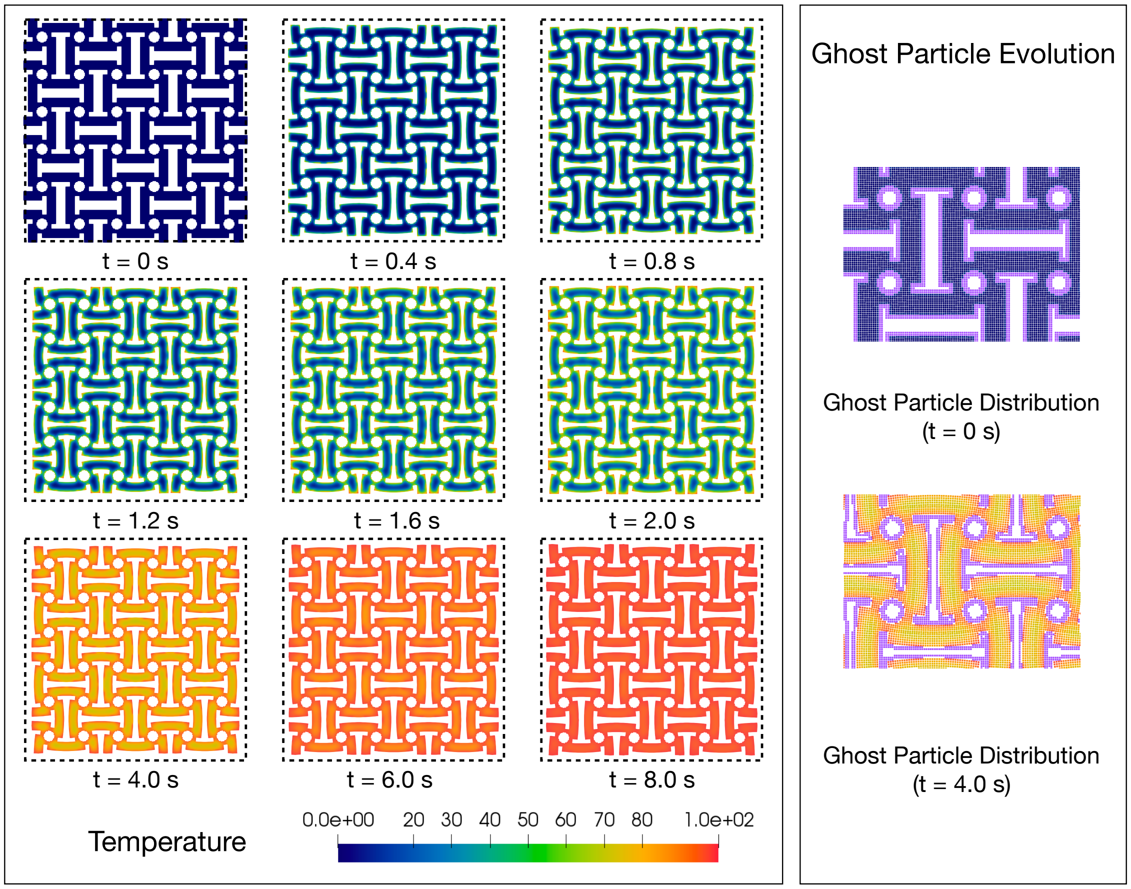


Figure 2.21: Numerical example of the 2D microstructure proposed in [124]. Dotted line corresponds to the original volume of the microstructure. The structure is suddenly exposed to an elevated background temperature of $\Delta\theta_{ext} = 100\ K$. At $\sim 8s$, the structure nearly reaches thermal equilibrium, exhibiting a negative overall thermal expansion coefficient. The right 2 figures show the ghost particles that were placed by our algorithm (purple markers).

2.5.9 Analysis of a Metamaterial Structure with Negative Thermal Expansion

It is possible to engineer a metamaterial that will possess an overall negative thermal expansion coefficient by carefully designing a specific microstructure utilizing multiple materials. Recently in [124], a hypotherical microstructure that may possess such property was proposed and demonstrated via simple large-scale structural experiments. In [102] the same proposed microstructure (Figure 2.20) was fabricated via laser lithography. The expected "negative thermal expansion" effects were then verified experimentally.

As a practical numerical demonstration, a similar 2D microstructure was devised within our SPH framework, and the microstructure was subjected to instantaneous temperature conditions on the boundary (free surface). Table 2.1 lists the physical parameters that was assumed for our demonstration. A $0.6\ m \times 0.6\ m$ 2D structure with a repeating microstructure (shown in Figure 2.20) was suddenly exposed to an elevated ambient temperature of $\Delta\theta = 100K$, and pure conduction is assumed at the free surface. The evolution of the structure is plotted in Figure 2.21. As the material near the boundary expands, the individual scaffolds connecting the circular nodes start to bend; minimizing voids in the initial structure. For the structure used in the demonstration, the final average thermal expansion coefficient (sometimes called the secant coefficient of thermal expansion) was computed to be $\alpha = 2.054 \times 10^{-6}$.

2.5.10 Discussion

A comprehensive solution process based on SPH for one-way coupled thermoelasticity problems was proposed. In order to obviate instabilities and inconsistencies, a total Lagrangian approach was employed as the basis of the SPH implementation. An implicit time-stepping scheme was also proposed to complement the proposed method. By solving well-known benchmark problems with the proposed solver, we showed that the results obtained via SPH were in good agreement with its analytical counterpart and FEM. We also included a practical numerical example for a multi-material structure exhibiting negative thermal expansion. Several drawbacks still exist due to the limitation of implementing thermal / mechanical boundary conditions in SPH. For example, the adaptive ghost particle placing algorithm that are required to enforce thermal boundary conditions requires many dummy particles that do not really participate in the simulation. Difficulties in enforcing Neumann and Dirichlet conditions for the solid also exist. A more robust solution process involving Newton iterations would be extremely desirable, although obtaining the Hessian (or even an approximate) of the particle system may be very expensive. We hope to revisit this topic as future work.

2.6 Non-Newtonian Fluids

2.6.1 Introduction

Originally developed for astrophysical simulations, SPH has become a popular numerical method of choice for simulating free-surface Newtonian flow problems ([32, 75, 34]). Naturally, the topic regarding the simulation of non-Newtonian flows with this method have been in question for some time, although successful attempts were limited due to the explicit nature of SPH. Such efforts can be found in previous work such as in [57]. Although satisfactory results were obtained for some cases, the effective viscosity values used within the simulation had little variance. For example, for a Herschel-Bulkley type fluid, unyielded regions exhibit viscosity values that are multiple magnitudes larger compared to yielded regions. In our experience, it becomes very difficult to resolve a SPH system with such viscosity variance exclusively with explicit methods, especially with SPH. For this section, we will employ Chorin's projection method to split the viscous forces and the pressure forces. We then introduce an iterative process to solve for the viscous forces implicitly for the next timestep. A brief convergence study for a simple 2D flow field of a Herschel-Bulkley fluid was given at the end.

2.6.2 Problem Formulation

We first discretize the Navier-Stokes equation via SPH formalism. For each SPH particle, we have

$$\frac{D\boldsymbol{u}_i}{Dt} = -\left\langle \frac{1}{\rho}\nabla p \right\rangle_i + \left\langle \frac{\mu}{\rho}\nabla^2 \boldsymbol{u} \right\rangle_i + \boldsymbol{b}_i \tag{2.140}$$

$$\frac{D\rho_i}{Dt} = \langle \rho \nabla \cdot \boldsymbol{u} \rangle_i \tag{2.141}$$

$$p_i = f_{EOS}(\rho_i) \tag{2.142}$$

The physical quantity included within the angles brackets $< \cdot >$ represents the physical value discretized via SPH, and f_{EOS} represents the equation of state. Tait's equation of state ([96]) was used for this paper. Note that taking such approach results in a weakly-compressible SPH model (WCSPH). The pressure gradient discretization (2.140) is usually chosen to be ([96]):

$$-\left\langle \frac{1}{\rho}\nabla p \right\rangle_i = -\sum_{j \in \mathcal{N}_i} \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W_{ij} m_j \tag{2.143}$$

The index i represents the ith SPH particle, while $j \in \mathcal{N}_i$ represents the set of neighbors of the ith particle. W_{ij} is the kernel function centered around particle i, $\nabla_i W_{ij}$ represents the gradient of the kernel, p_i is the pressure of particle i, p_j is the pressure of particle j,

ρ_i is the density of particle i, ρ_j is the density of particle j, and m_i and m_j represent the mass of particle i and j, respectively.

We may split the pressure and the viscous contributions via a projection method ([29]):

$$\boldsymbol{u}_i^* = \boldsymbol{u}_i^n + \left(\frac{\Delta t}{\rho_i}\right)(\boldsymbol{a}_p)_i \tag{2.144}$$

$$\boldsymbol{u}_i^{n+1} = \boldsymbol{u}_i^* + \left(\frac{\Delta t}{\rho_i^*}\right)(\boldsymbol{a}_v)_j \tag{2.145}$$

where $\boldsymbol{u}_i^n$ and $\boldsymbol{u}_i^{n+1}$ respectively represents the velocity of particle i at the nth and $n+1$th timestep, and $\boldsymbol{n}_i^*$ represents the “intermediate velocity”. The acceleration of each particle due to pressure forces can be computed using (2.143). We now look into the acceleration contribution for the viscous step $(\boldsymbol{a}_{a,v})$. Influenced by the viscous operator utilized in[108], we explicitly expand out the viscous operator. For an explicit version, we will have:

$$\boldsymbol{u}_i^{n+1} = \boldsymbol{u}_i^n + (\Delta t)\left(\sum_{j\in\mathcal{N}_i} \frac{4m_j(\bar{\mu}_{eff})_{ij}(\boldsymbol{x}_i^* - \boldsymbol{x}_j^*)\cdot\nabla W_{ij}}{(\rho_i^* + \rho_j^*)(\boldsymbol{x}_i^* - \boldsymbol{x}_j^*)\cdot(\boldsymbol{x}_i^* - \boldsymbol{x}_j^*)}(\boldsymbol{u}_i^* - \boldsymbol{u}_j^*)\right) \tag{2.146}$$

For the average effective viscosity $\bar{\mu}_{eff,ab}$, we pick

$$(\bar{\mu}_{eff})_{ij} = \frac{(\bar{\mu}_{eff})_i + (\bar{\mu}_{eff})_j}{2} \tag{2.147}$$

Note that there exists different methods for “mixing” the viscosities between particles ([58, 32]).

Later on, we will re-formulate the above via an implicit timestepping scheme.

The effective viscosity of each particle is computed according to the non-Newtonian fluid model before the viscous step. For a regularized Herschel-Bulkley fluid model (Tanner / milthrope 1983):

$$\mu_{eff} = \begin{cases} k\dot{\gamma_o}^{n+1} + \tau_o\dot{\gamma_o}^{-1} & \dot{\gamma} < \dot{\gamma}_o \\ k\dot{\gamma}^{n-1} + \tau_o\dot{\gamma}^{-1} & \dot{\gamma} \geq \dot{\gamma}_o \end{cases} \tag{2.148}$$

With the viscous shear $(\boldsymbol{\tau}_i)$ computed using the above effective viscosity being

$$\boldsymbol{\tau} = 2(\mu_{eff})_i(\dot{\gamma})\boldsymbol{D}_i, \quad \boldsymbol{D}_i = \frac{1}{2}(\nabla\boldsymbol{u}_i^* + \nabla\boldsymbol{u}_i^{*T}) \tag{2.149}$$

For continuity, the method found in [13] was employed (widely known within the SPH community as δ-SPH) :

$$\langle\rho\nabla\cdot\boldsymbol{u}\rangle_i = -\rho_i\sum_{j\in\mathcal{N}_i}(\boldsymbol{u}_j - \boldsymbol{u}_i)\cdot\nabla_i W_{ij}V_j + \delta h c_0\mathcal{D}_i \tag{2.150}$$

where $\mathcal{D}_i$ is the "diffusion" term, defined as

$$\mathcal{D}_i = 2 \sum_{j \in \mathcal{N}_i} \psi_{ij} \frac{\boldsymbol{r}_{ji} \cdot \nabla W_{ij}}{r_{ij}^2} V_j, \tag{2.151}$$

where δ is a tunable constant (usually chosen to be 0.1), h is the SPH smoothing length, c_0 is the speed of sound, and ψ_{ij} is defined as :

$$\psi_{ij} = (\rho_j - \rho_i) - \frac{1}{2} \left(\langle \nabla_1 \rho_i \rangle + \langle \nabla_1 \rho_j \rangle \right) \cdot \boldsymbol{r}_{ji}, \tag{2.152}$$

where $\langle \nabla_1 (\cdot) \rangle$ represents the renormalized gradient operator. $\langle \nabla_1 \rho_i \rangle$ can then be written as :

$$\langle \nabla_1 \rho_i \rangle = \boldsymbol{B}_i \sum_{j \in \mathcal{N}(i)} (\rho_j - \rho_i) \nabla_i W_{ij} V_j \tag{2.153}$$

Here, $\boldsymbol{B}_i$ is the *first derivative renormalization matrix* suggested by [99], which is defined as :

$$\boldsymbol{B}_i = \left[- \sum_{j \in \mathcal{N}(i)} \boldsymbol{r}_{ij} \otimes \nabla W_{ij} V_j \right]^{-1} \tag{2.154}$$

Physically, this term adds "artificial diffusivity" to the continuity equation. Although exact conservation of mass no longer holds, the added diffusivity onto the density greatly increases stability throughout the system. As a result, δ-SPH was proven to be an accurate, versatile approach for many SPH applications as an alternative to a PPE (pressure Poisson Equation) solver for estimating the pressure field. ([13, 12, 80])

2.6.3 Viscous Forces: Semi-Implicit Time Stepping

Non-Newtonian flows usually consist of yielded and non-yielded regions within the fluid. Many practical fluid models usually generate effective viscosities that mare magnitudes lower for yielded regions, allowing them to "flow". Obtaining stability and convergence for such systems has been a challenge for SPH due to this property of non-Newtonian fluid models. In order to address this issue (2.9), we propose an implicit time-stepping scheme for the viscous step. A fixed-point iteration scheme is used to solve for the whole particle system, similar to ([133, 132]).

Consider a system where the acceleration of each particle only depends on $\boldsymbol{x}$ and $\boldsymbol{v}$. Using an implicit trapezoidal rule, for the displacement field and the acceleration field we have:

$$\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}^{*} \\ \boldsymbol{u}^{*} \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{u}^{t+1} \\ \boldsymbol{a}_v^{t+1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{u}^{*} \\ \boldsymbol{a}_v^{*} \end{bmatrix} \right) \tag{2.155}$$

Where $\begin{bmatrix} \boldsymbol{r}^* \\ \boldsymbol{u}^* \end{bmatrix}$ represents the vector of displacements and velocities of the whole particle system at time t. Plugging in $[\boldsymbol{u}^{t+1}] = [\boldsymbol{u}^*] + \frac{\Delta t}{2}\left([\boldsymbol{a}_v^{t+1} + \boldsymbol{a}_v^*]\right)$, we have for $[\boldsymbol{x}^{t+1}]$:

$$\left[\boldsymbol{r}^{t+1}\right] = [\boldsymbol{r}^*] + \frac{\Delta t}{2} \cdot \left([\boldsymbol{u}^*] + \frac{\Delta t}{2} \cdot \left(\left[\boldsymbol{a}_v^{t+1}\right] + [\boldsymbol{a}_v^*] \right) + [\boldsymbol{u}^*] \right) \tag{2.156}$$

$$= [\boldsymbol{r}^*] + \Delta t \cdot [\boldsymbol{u}^*] + \frac{(\Delta t)^2}{4} \left(\left[\boldsymbol{a}_v^{t+1}\right] + [\boldsymbol{a}_v^*] \right) \tag{2.157}$$

Thus, we have for $\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}$:

$$\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}^* \\ \boldsymbol{u}^* \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{u}^* + \frac{\Delta t}{2} \cdot (\boldsymbol{a}_v^{t+1} + \boldsymbol{a}_v^*) \\ \boldsymbol{a}_v^{t+1} \end{bmatrix} + \begin{bmatrix} \boldsymbol{u}^* \\ \boldsymbol{a}_v^* \end{bmatrix} \right) \tag{2.158}$$

Notice that the acceleration vector $\boldsymbol{a}_v^{t+1}$ is a nonlinear function of the velocity and the position vectors at time t+1; thus $[\boldsymbol{a}_v^{t+1}] = \boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}\right)$, where

$$(\boldsymbol{a}_v^{t+1})_i = \left(\sum_{j \in \mathcal{N}_i} \frac{4 m_j (\bar{\mu}_{eff})_{ij} (\boldsymbol{r}_i^{t+1} - \boldsymbol{r}_j^{t+1}) \cdot \nabla W_{ij}}{(\rho_i^{t+1} + \rho_j^{t+1})(\boldsymbol{r}_i^{t+1} - \boldsymbol{r}_j^{t+1}) \cdot (\boldsymbol{r}_i^{t+1} - \boldsymbol{r}_j^*)} (\boldsymbol{u}_i^{t+1} - \boldsymbol{u}_j^{t+1}) \right) \tag{2.159}$$

Writing everything out again,

$$\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix} = \boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix} \right) \overset{def}{=} \begin{bmatrix} \boldsymbol{r}^* \\ \boldsymbol{u}^* \end{bmatrix} + \frac{\Delta t}{2} \cdot \left(\begin{bmatrix} \boldsymbol{u}^* + \frac{\Delta t}{2} \cdot \left(\boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix} \right) + \boldsymbol{a}_v^t \right) \\ \boldsymbol{\xi}\left(\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix} \right) \end{bmatrix} + \begin{bmatrix} \boldsymbol{u}^* \\ \boldsymbol{a}_v^* \end{bmatrix} \right) \tag{2.160}$$

Thus, we have a system of nonlinear equations with the unknown vector $\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}$. We now have the form suitable for a fixed-point iteration solution process :

$$\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^{k+1} = \boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^{k} \right) \tag{2.161}$$

Where k is the iteration counter. We simply use $\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^0 = \begin{bmatrix} \boldsymbol{r}^* \\ \boldsymbol{u}^* \end{bmatrix}$ as the initial "guess".

Algorithm 2.9 Fixed Point Iteration

```
function FixedPointIteration
    $\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^0 = \begin{bmatrix} \boldsymbol{r}^{t} \\ \boldsymbol{u}^{t} \end{bmatrix}$                ▷ Initialize Guess
    while Norm($\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^{k+1} - \begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^{k}$) $> TOL$ do
        for $Particle_a$ in $SolidParticles$ do                ▷ Compute $\boldsymbol{\Psi}^{k+1}$
            ComputeAccelerations($Particle_a$)
        end for
        for $Particle_a$ in $SolidParticles$ do                ▷ Update Guess
            $\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^{k+1} = \boldsymbol{\Psi}\left(\begin{bmatrix} \boldsymbol{r}^{t+1} \\ \boldsymbol{u}^{t+1} \end{bmatrix}^{k}\right)$
        end for
    end while
end function
```

2.6.4 Boundary Conditions

A ghost-particle approach was used for our no-slip boundary implementation. Such approach is fairly straight-forward, although for Newtonian fluids one must be cautious when assigning velocities to the ghost particles. In previous work regarding Newtonian fluids such as ([93, 17]), the ghost particles were assigned mirrored velocities in order to improve convergence near the boundaries when implementing no-slip conditions. Unfortunately, for non-Newtonian fluids, this approach would be inappropriate since the rate of deformation tensor that is required to compute the shear tensor ($\boldsymbol{\tau} = 2(\mu_{eff})_i\dot{\gamma}\boldsymbol{D}_i, \quad \boldsymbol{D}_i = \frac{1}{2}(\nabla\boldsymbol{u}_i^* + \nabla\boldsymbol{u}_i^{*T})$) would incorrectly indicate that the fluid is "yielding" near the boundary. Thus, the velocities are required to be zero for the ghost particles.

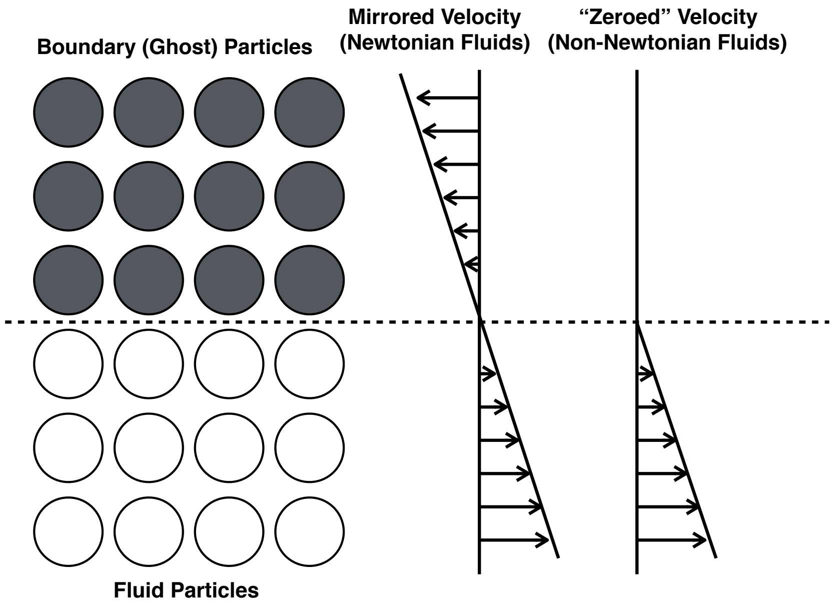

Boundary (Ghost) Particles
Mirrored Velocity (Newtonian Fluids)
"Zeroed" Velocity (Non-Newtonian Fluids)
Fluid Particles

Figure 2.22: No-slip boundary conditions for Newtonian fluids and Non-Newtonian fluids.

2.6.5 Numerical Example: Herschel-Bulkley 2D Poiseuille Flow

A simple steady-state numerical example is presented to show convergence properties of our method, as shown in figure 2.23. The left/right boundaries were treated with a periodic boundary condition, where the leaving particles were feeded back into the domain. The quintic Wendland kernel was exclusively used as our kernel function, due to its many known benefits ([78]). The kernel support radius was chosen to be 3 times the particle diameter ($3 \times \Delta x$).

To compare our results, the analytical solution of the above problem ([46]) was used:

$$u_x\left(y\right) = \begin{cases} \left(\frac{n}{n+1}\right)\left[\left(\frac{\left(\frac{\partial p}{\partial x}\right)}{k}\right)^{\frac{1}{n}}\left(\left(\frac{H}{2} - \frac{\tau_0}{\left(\frac{\partial p}{\partial x}\right)}\right)^{\frac{n+1}{n}} - \left|\left(y - \frac{H}{2}\right) - \frac{\tau_0}{\left(\frac{\partial p}{\partial x}\right)}\right|^{\frac{n+1}{n}}\right)\right] & \left|y - \frac{H}{2}\right| \geq \frac{\tau_0}{\left(\frac{\partial p}{\partial x}\right)} \\ \left(\frac{n}{n+1}\right)\left[\left(\frac{\left(\frac{\partial p}{\partial x}\right)}{k}\right)^{\frac{1}{n}}\left(\frac{H}{2} - \frac{\tau_0}{\left(\frac{\partial p}{\partial x}\right)}\right)^{\frac{n+1}{n}}\right] & \left|y - \frac{H}{2}\right| < \frac{\tau_0}{\left(\frac{\partial p}{\partial x}\right)} \end{cases} \tag{2.162}$$

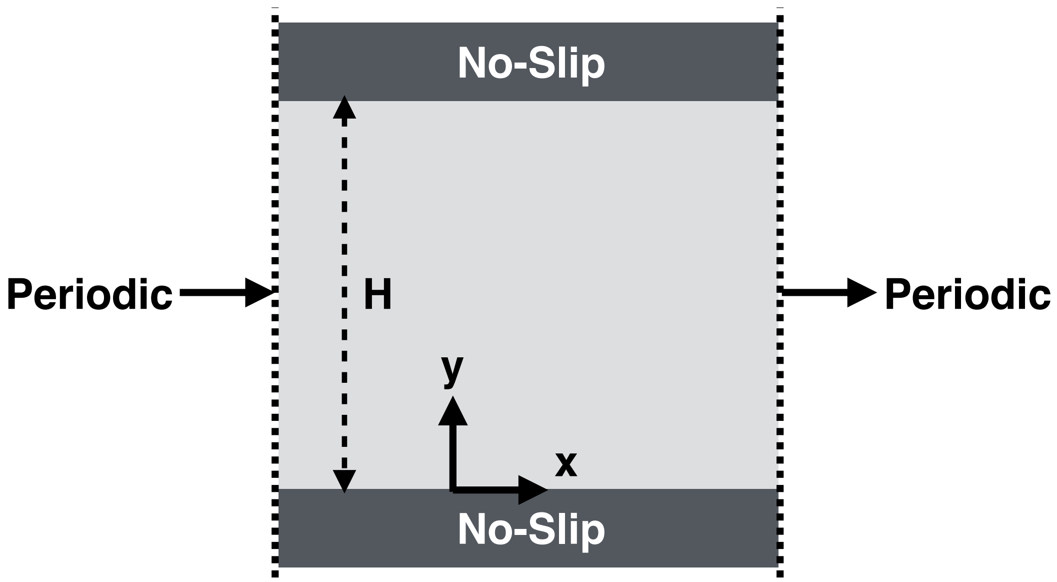


Figure 2.23: Physical parameters: $\frac{\partial p}{\partial x} = 10\ [Pa/m],\ \rho = 1\ [kg/m^3],\ H = 0.5\ [m], \boldsymbol{u}(y = 0) = 0,\ \boldsymbol{u}(y = H) = 0,\ k = 0.5\ [Pa \cdot s],\ \tau_o = 0.5\ [Pa],\ n = 0.5,\ \gamma_o = 0.001.$

2.6.6 Numerical Results

The results of the numerical solution is shown in figures 2.24, 2.26. 4 simulations with different resolutions (11, 21, 41, 81 particles in the y-direction, respectively).

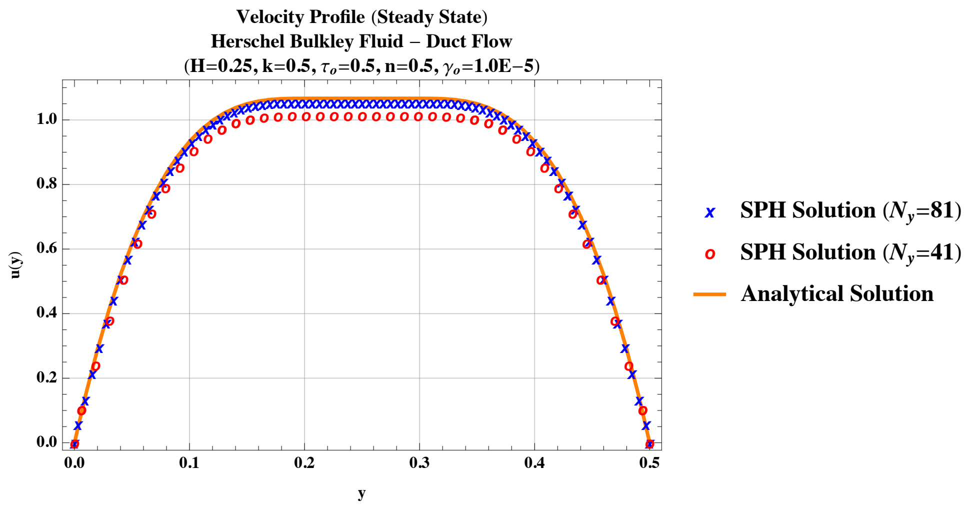


Figure 2.24: Steady state velocity profile compared to analytical solution.

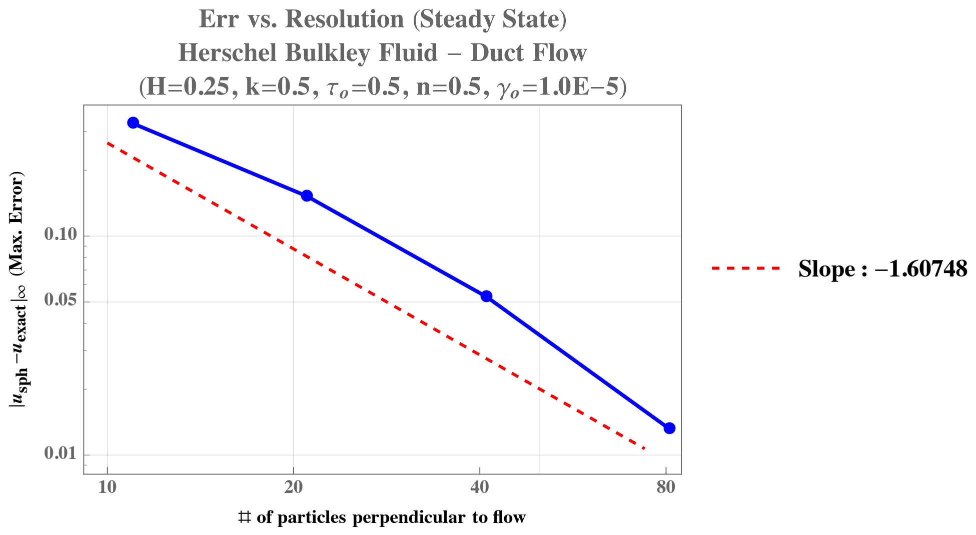


Figure 2.25: The method showed a convergence order of 1.6 in space.

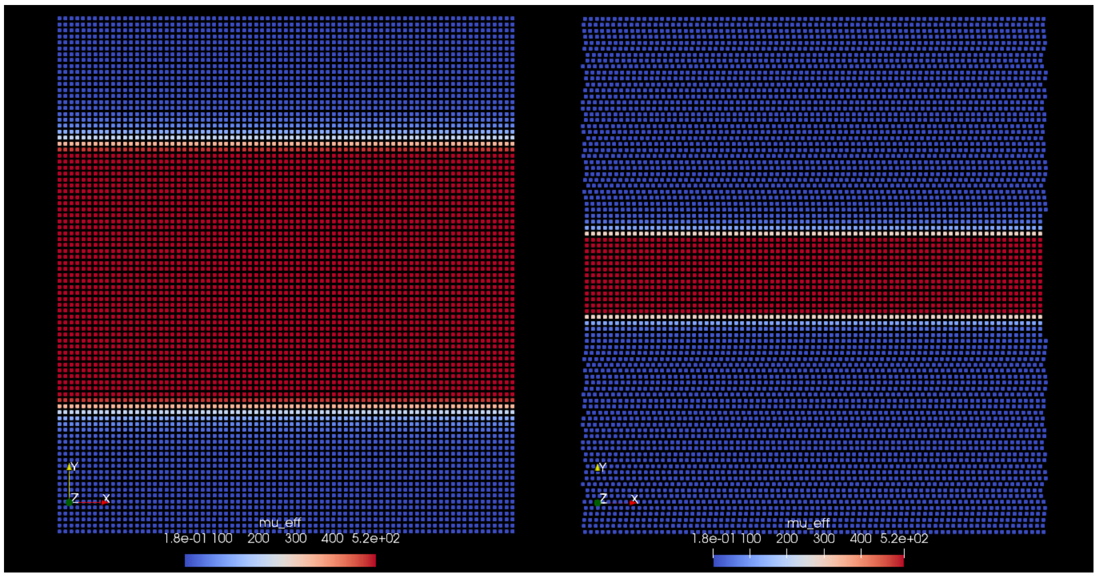


Figure 2.26: Red region is the “unyielded” fluid where the effective viscosity is significantly larger. At startup (left), the region is significantly larger compared to steady state (right).

2.6.7 Discussion

Figure 2.24 compares the velocity profile of the SPH solution to the analytical solution. Figure 2.25 shows the convergence rate of our proposed method. The results show that our semi-implicit approach is capable of resolving large viscosity contrasts between the two different regions. Nevertheless, compared to the simulation with the highest resolution ($N_x = 81$), the lower resolution cases seemed to show rather large errors. Since the shear rate $\dot{\gamma} = \sqrt{2\boldsymbol{D} : \boldsymbol{D}}$ is dependent on the rate of deformation tensor, which is computed at each timestep with the SPH operators, it is difficult to maintain "sharp" transitions for the rate of deformation tensor. This transition region can also be observed in figure 2.26). This indicates that the boundaries between the yielded region and the non-yielded regions are being excessively smoothed (which SPH operators are known to do), especially for lower resolution simulations.

Chapter 3

Computational Optimization of Nozzle Extrusion Paths in Ink Writing Manufacturing Procedures

3.1 Introduction

Direct ink writing (DIW) is an additive manufacturing method where an extrusion deposition process of a fluid-like material (often described as "gels" and "pastes") is used. The depositions are usually tailored, functional inks, often used to manufacture devices such as sensors, micro-fluidic networks, tissue-engineering scaffolds, etc. Desired patterns are generated using a controlled nozzle that continuously, or discontinuously (eg, Drop-On-Demand type methods) ejects such inks. Typical dimensions of the nozzle diameter can reach from several hundred microns to even sub-micro scales. In many applications, these inks are a colloidal suspensions, obtained by a combination of

- Solvents (water is widely used)
- Functional particles (Alumina, Titania, Mullite, β-TCP, etc)
- Dispersants
- Rheology Modifiers

The weight percentage of the functionalizing particles (powders) can reach up to 50%, where higher concentration gives benefits in terms of printing dynamics / shrinking. As the extrusion extrudes out from the nozzle, the ink usually undergoes solidification. Some types of solidification methods include:

- Solidification due to drying. ex, Colloidal Suspensions
- Gelation or chemical crosslinking ([48])
- Crystalization ex, freezing of water

The behaviour of the extrusion can sometimes be non-trivial, due to the non-Newtonian behaviour of the extrusion. In order to obtain accurate desired depositions, it is sometimes required to "fine-tune" the extrusion toolpath to match the physical behaviour of the extrusion. In many cases, a naive toolpath identical to the desired shape will return undesirable depositions. There are several factors that may influence the optimal toolpath. Some include:

- Path that the tool follows
- Distance from the nozzle tip to the target substrate
- Nozzle angle
- Extrusion speed
- Translation speed of the nozzle

- Material properties of the "uncured" material
- Temperature
- etc.

The goal of this research is to investigate towards finding the optimal toolpath for a given set of physical writing parameters. The basic key ideas of framework can be stated as below:

1. Optimization is performed for a given desired deposition path via a "swarm intelligence" optimization method.
2. A set of parameters (cite) are chosen in a way such that it represents a mildly modified path from the desired one. For each "timestep" of the "swarm member", the parameters are either randomly generated or obtained by moving each member to a more desirable location within the parameter space.
3. The "fitness" of each path is then evaluated by a single physical simulation. A parameter set is considered to be more "fit", if the deviation of the actual deposition from the desired path is smaller.
4. The physical simulation is achieved by using a numerical solver based on smoothed particle hydrodynamics (SPH).

Ideally, there are infinite ways one can characterize a toolpath. In reality, due to the limitation of computational power and practicality, it is convenient to chose to only optimize a certain parameter. Thus, for the work in this paper, we only attempt to optimize the path that the tool actually follows. The procedures on how a toolpath is characterized is discussed in detail later on.

In order to create a fast, efficient physical simulation step, smoothed particle hydrodynamics (SPH) was used to model the material being deposited onto the target substrate. To model the highly-viscous and shape-retaining characteristics of the material, an incompressible, a fully implicit formulation was used ([122]). Although directly using non-Newtonian fluid models (which are known to match the flow of the materials discussed earlier) would be favorable due to the abundance of physical measurement data, we chose not to use non-Newtonian models for our optimization study due to the following reasons:

1. Even with fully implicit formulations, we encountered difficulties for accurately computing the viscous forces with a reasonable timestep and discretization stencil.
2. Many physical characteristics of the inks can also be obtained by using a Newtonian fluid model with large viscosities, combined with large surface tension.

	Stress Tensor
Newtonian Fluid	$\boldsymbol{\sigma} = p\mathbf{1} + 2\mu\boldsymbol{D}$
Power Law	$\boldsymbol{\sigma} = p\mathbf{1} + 2k\dot{\gamma}^{n-1}\boldsymbol{D}$
Bingham	$\boldsymbol{\sigma} = p\mathbf{1} + \boldsymbol{\tau}_o + 2\mu\boldsymbol{D}$
Carreau-Yasuda	$\boldsymbol{\sigma} = p\mathbf{1} + \boldsymbol{\tau}_o + 2k\dot{\gamma}^{n}\boldsymbol{D}$
Herschel Bulkley	$\boldsymbol{\sigma} = p\mathbf{1} + \boldsymbol{\tau}_o + 2\left(k\dot{\gamma}^{n-1} + \tau_o\dot{\gamma}^{-1}\right)\boldsymbol{D}$

Table 3.1: Non-Newtonian Fluid Models

The fact that there are many experimentalists who also prefer to adopt our second argument (cite) backs our assumption regarding our choice of fluid model.

Finally, as a proof of concept, several test cases were performed, including a 90-Degree bend, a 30-Degree bend. The tooltip is assumed to have a constant tangential velocity throughout the whole deposition. In a manufacturing standpoint, the benefit would realize as a reduced turnaround time. We compare the naive path of choice (a path identical to the desired deposition shape) to the optimized path. The optimization algorithm produced a non-intuitive toolpath that generated more “fit” results than a naive path.

3.2 Modeling of Inks

3.2.1 Mechanics of Inks

In Direct Ink Writing, obtaining the desired pattern is a very sensitive procedure, and sometimes can be extremely difficult to control. In many cases, iterative fine tuning is required to obtain a set of printing parameters that will produce extrusions that will suit the the desired application. In order to devise a computer based simulation, it is required to employ a suitable fluid model. Although there is no concrete agreement on what model that should be employed for inks, non-Newtonian fluid models are widely accepted due to its ability to model “shear-thinning”. This “shear-thining” behavior allows the fluid to “set” after deposition, so that the post-extrusion shape is maintained throughout the un-yielded region. The stress-strain rate response of such fluid models are shown in table 3.1.

One popular model that can be employed is the Herschel-Bulkley fluid model ([73, 54, 48, 85]), which accounts for the shear-thinning properties of inks that are used in DIW applications.

It is also possible to employ Newtonian-based fluid models for ink modeling. This type of approach towards modeling the ink was also used in previous work such as ([100, 101]). A very high viscosity combined with high surface tension allows the fluid to “set” at a shape after deposition. Regarding which approach (Newtonian vs Non-Newtonian) is more appropriate is debatable. One must also acknowledge that any of the two approaches require careful calibrations to match the actual fluid, and is highly dependent on the type

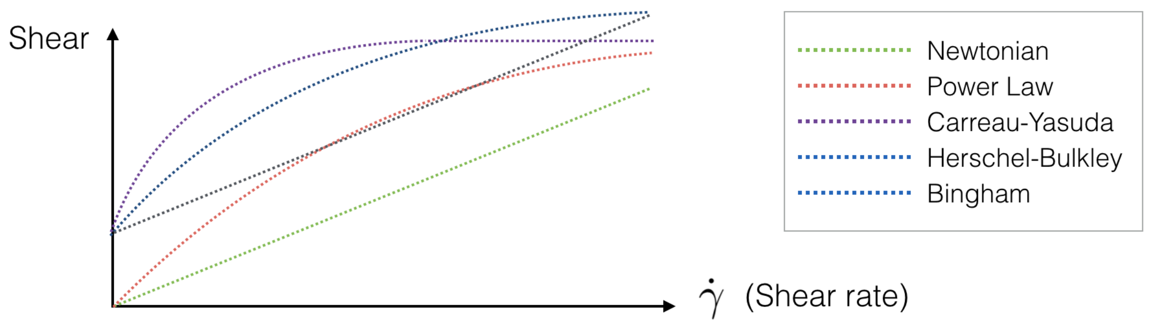


Figure 3.1: Non-Newtonian Fluid Models

of ink as well; thus we assume that any of the two approaches are acceptable as long as the model is adequately calibrated towards one's specific application.

In this paper, we employ a Newtonian fluid model combined with high surface tension to model the ink. The governing equations of the model follows as:

$$\rho \frac{D\boldsymbol{u}}{Dt} = -\nabla p + \mu \nabla^2 \boldsymbol{u} \ \ \text{(Navier-Stokes)} \tag{3.1}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \ \ \text{(Continuity)} \tag{3.2}$$

Where $\boldsymbol{u}$ is the velocity of the fluid, ρ is the density of the fluid, p is the pressure field and μ is the viscosity. We would also like to mention that the continuity equation in incompressible SPH implementations is automatically satisfied, which will be discussed later on in the text. In order to represent surface tension, we must further introduce $\boldsymbol{f}^{st}$ which represents the surface tension. One widely employed surface tension model is the CSF surface tension method. In the work by Brackbill et al ([20]), a continuum form of the surface tension was derived:

$$\boldsymbol{f}^{st} = \sigma \kappa (\boldsymbol{r}) \frac{\nabla c (\boldsymbol{r})}{[c (\boldsymbol{r})]} \tag{3.3}$$

Where σ is the surface tension coefficient, $\kappa(\boldsymbol{r})$ is the curvature of the free surface defined at position $\boldsymbol{r}$, and $c(\boldsymbol{r})$ is the "color function" defined at position $\boldsymbol{r}$:

$$c(\boldsymbol{r}) = \begin{cases} c_1 & \text{(Fluid 1)} \\ c_2 & \text{(Fluid 2)} \\ \frac{c_1+c_2}{2} & \text{(At the interface)} \end{cases} \tag{3.4}$$

The squared bracket notation $[c(\boldsymbol{r})]$ represents the "jump operator", which gives the change in value of the color field at the interface. Thus, for this particular surface tension method, the Navier-Stokes equation including surface tension becomes

$$\rho \frac{D\boldsymbol{u}}{Dt} = -\nabla p + \mu \nabla^2 \boldsymbol{u} + \sigma \kappa \left(\boldsymbol{r}\right) \frac{\nabla c\left(\boldsymbol{r}\right)}{\left[c\left(\boldsymbol{r}\right)\right]} \text{ (Navier-Stokes)} \tag{3.5}$$

This approach does not require calibration, since the physical parameter (σ) is already available for many fluids. Nevertheless, despite its popular usage, the above method is often problematic for our discretization method ([97]). Later on, we will introduce a different surface tension method that can be well-integrated with our numerical framework of choice.

3.2.2 Numerical Method - Smoothed Particle Hydrodynamics (SPH)

3.2.2.1 Brief Introduction

SPH (Smoothed Particle Hydrodynamics) was first introduced by JJ. Monaghan to solve astrophysics problems ([94, 89, 88, 95]). In SPH, a kernel function that centers around a SPH "particle" is defined as $W\left(\boldsymbol{r}, h\right)$, where $\boldsymbol{r}$ is the position vector to the center of the kernel (the position of the SPH particle), and h is the smoothing length. SPH is an ideal method of choice for our problem, since the free-surfaces of the extrusions are automatically resolved without any type of re-meshing. Thus, the method is used exclusively for our deposition framework.

Without derivation, we state the so-called "SPH operators' for a scalar field $\psi\left(\boldsymbol{r}\right)$ and the vector field $\boldsymbol{\phi}\left(\boldsymbol{r}\right)$:

$$\left\langle \psi \right\rangle_a = \sum_{b \in \mathcal{N}_a} \psi_b v_b W\left(\boldsymbol{r}_{ab}, h\right) \text{ (Field value)} \tag{3.6}$$

$$\left\langle \nabla \psi \right\rangle_a = \sum_{b \in \mathcal{N}_a} \left(\psi_b - \psi_a\right) v_b \nabla W_{ab} \text{ (Gradient)} \tag{3.7}$$

$$\left\langle \nabla \cdot \boldsymbol{\phi} \right\rangle_a = \sum_{b \in \mathcal{N}_a} v_b \left(\boldsymbol{\phi}_b - \boldsymbol{\phi}_a\right) \cdot \nabla W_{ab} \text{ (Divergence)} \tag{3.8}$$

$$\left\langle \nabla^2 \psi \right\rangle_a = \sum_{b \in \mathcal{N}_a} 2 v_b \frac{\psi_b - \psi_a}{r_{ab}} \boldsymbol{e}_{ab} \nabla W_{ab} \text{ (Laplacian)} \tag{3.9}$$

Here, $\mathcal{N}_a$ represents all the neighboring particles (particles that fall within the kernel support domain centered around particle a, v_a is the physical volume represented by particle a, $W_{ab} = W\left(\boldsymbol{r}_{ab}, h\right)$, and $\boldsymbol{r}_{ab} = \boldsymbol{r}_a - \boldsymbol{r}_b$. The kernel function is usually chosen to be 3 times the particle radius for efficiency. Although there exists many possible choices for the kernel function, for our work we employ the quintic Wendland kernel due to its known benefits ([78]).

The introduced operators were first used in pioneering earlier research such as [89, 21], and proved itself to be practical and accurate throughout the years . Recently, advanced higher order operators are becoming more and more popular, although such operators were not employed in this work (refer to [44]). Also, for convenience, we write the angled brackets $\langle\cdot\rangle$ to represent the value obtained from the SPH operators.

3.2.2.2 Discretization of the Navier-Stokes Equation

Using the operators introduced above, the Navier-Stokes equation without surface tension becomes

$$\left(\frac{D\boldsymbol{u}}{Dt}\right)_a = -\left\langle \frac{1}{\rho}\nabla p\right\rangle_a + \left\langle \frac{\mu}{\rho}\nabla^2\boldsymbol{u}\right\rangle_a + \mathbf{g} \tag{3.10}$$

$$\left(\frac{D\rho}{Dt}\right)_a = \langle \rho\nabla\cdot\boldsymbol{u}\rangle_a \tag{3.11}$$

Many surface tension discretization methods exist for SPH ([3, 1, 91, 97, 114]). While some are based on the CSF surface tension model described earlier, others require parameters that must be calibrated. For our work, we will follow the surface tension model described in the work by Akinci et al ([3]). The model combines molecular-like forces between the SPH particles, similar to ([114]), along with surface area minimizing forces. According to the original paper, the “molecular” cohesion force is defined as

$$\boldsymbol{F}^{molecular}_{a\leftarrow b} = -\gamma_1 m_a m_b C\left(|\boldsymbol{r}_a - \boldsymbol{r}_b|\right)\left(\frac{\boldsymbol{r}_a - \boldsymbol{r}_b}{|\boldsymbol{r}_a - \boldsymbol{r}_b|}\right) \tag{3.12}$$

where C is a cubic spline function defined as

$$C(r) = \frac{32}{\pi h^9}\begin{cases} (h-r)^3 r^3 & \frac{h}{2} < r \leq h \\ 2(h-r)^3 r^3 - \frac{h^6}{64} & 0 \leq r < \frac{h}{2} \\ 0 & \text{Otherwise} \end{cases} \tag{3.13}$$

The curvature derived forces are additionally computed with

$$\boldsymbol{F}^{curvature}_{a\leftarrow b} = -\gamma_2 m_a\left(\boldsymbol{n}_a - \boldsymbol{n}_b\right) \tag{3.14}$$

with

$$\boldsymbol{n}_a = h\sum_{b\in\mathcal{N}_a}\frac{m_b}{\rho_b}\nabla W_{ab} \tag{3.15}$$

which is a vector normal to the surface, and γ_1 and γ_2 are physical constants with appropriate units that require calibration. For an interior fluid particle, the summation of the kernel gradients will cancel each other out, leaving a near zero vector. The combined forces that will contribute towards the surface tension is then

$$\boldsymbol{F}^{st}_{a\leftarrow b} = \boldsymbol{F}^{molecular}_{a\leftarrow b} + \boldsymbol{F}^{curvature}_{a\leftarrow b} \tag{3.16}$$

Adding the surface tension forces onto the discretized Navier-Stokes equation (equation 3.10),

$$\left(\frac{D\boldsymbol{u}}{Dt}\right)_a = -\left\langle\frac{1}{\rho}\nabla p\right\rangle_a + \left\langle\frac{\mu}{\rho}\nabla^2\boldsymbol{u}\right\rangle_a + \mathbf{g} + \boldsymbol{f}_a^{st} \tag{3.17}$$

where the following kernel summations for each term (following [87, 21]):

$$\left\langle\frac{1}{\rho}\nabla p\right\rangle_a = \sum_{b\in\mathcal{N}_a} m_b\left(\frac{P_a(t)}{\rho_a^2}+\frac{P_b(t)}{\rho_b^2}\right)\nabla W_{ab}(t) \tag{3.18}$$

$$\left\langle\frac{\mu}{\rho}\nabla^2\boldsymbol{u}\right\rangle_a = \sum_{b\in\mathcal{N}_a}\frac{8\mu m_b\boldsymbol{r}_{ab}\cdot\nabla W_{ab}}{(\rho_a+\rho_b)^2(\boldsymbol{r}_{ab}^2+\epsilon)}\boldsymbol{u}_{ab} \tag{3.19}$$

$$\boldsymbol{f}_a^{st} = -\sum_{b\in\mathcal{N}_a}\left(\gamma_1 m_b C\left(|\boldsymbol{r}_a-\boldsymbol{r}_b|\right)\left(\frac{\boldsymbol{r}_a-\boldsymbol{r}_b}{|\boldsymbol{r}_a-\boldsymbol{r}_b|}\right)+\gamma_2\left(\boldsymbol{n}_a-\boldsymbol{n}_b\right)\right) \tag{3.20}$$

Note that we assumed that the density of the fluid is constant. This is possible due to the incompressible pressure solver that we will be employing.

3.2.2.3 Solution of the Pressure Poisson Equation (Incompressible SPH)

In weakly compressible SPH implementations (such as δ-SPH, [13]), the continuity equation is used to update the density of each SPH particle:

$$\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\boldsymbol{u})=0 \quad\rightarrow\frac{D\rho}{Dt}=\langle\rho\nabla\cdot\mathbf{u}\rangle \tag{3.21}$$

and discretized as

$$\langle\rho\nabla\cdot\boldsymbol{u}\rangle_a = -\rho_a\sum_{b\in\mathcal{N}_a}(\boldsymbol{u}_b-\boldsymbol{u}_a)\cdot\nabla_a W_{ab}v_b+\delta h c_0\mathcal{D}_a \tag{3.22}$$

where $\mathcal{D}_a$ is the “diffusion” term, defined as

$$\mathcal{D}_a = 2\sum_{b\in\mathcal{N}_a}\psi_{ab}\frac{\boldsymbol{r}_{ba}\cdot\nabla W_{ab}}{\boldsymbol{r}_{ab}^2}V_j, \tag{3.23}$$

where δ is a tunable constant (usually chosen to be 0.1), h is the SPH smoothing length, c_0 is the speed of sound, and ψ_{ab} is defined as :

$$\psi_{ab} = (\rho_b-\rho_a)-\frac{1}{2}\left(\langle\nabla_1\rho_a\rangle+\langle\nabla_1\rho_b\rangle\right)\cdot\boldsymbol{r}_{ba}, \tag{3.24}$$

Then, the equations regarding the motion of the fluid are closed of by an equation of state. An example is Tait’s equation of state:

$$P = \frac{1}{\gamma}\rho_0 c_0^2 \left(\left(\frac{\rho}{\rho_0} \right)^\gamma - 1 \right) \tag{3.25}$$

where c_0 is the speed of sound, ρ_0 is the rest density and γ is a parameter. This approach effectively eliminates the need to solve the pressure Poisson equation (PPE), and is easy to implement. On the otherhand, exact incompressibility no longer holds, and the simulation may require relatively small timesteps for it to be stable. Since our optimization framework demands many simulation runs with different parameters, being able to quickly finish a simulation is a critical factor for us. Thus, an incompressible SPH solver was employed for our simulations. For our application, we follow the work by [16], which is based on operator splitting ([28, 29]).

Algorithm 3.1 DFSPH Timestep (Using notation from Bender's original paper)

```
1. Compute non-pressure forces $\boldsymbol{a}_a^v$
2. Compute $\boldsymbol{v}_a^*(t) = \boldsymbol{v}_a + \Delta t \boldsymbol{a}_a^v$
3. CorrectDensityError()
4. Update Positions $\boldsymbol{x}_a(t + \Delta t) = \boldsymbol{x}_a(t) + \Delta t \boldsymbol{v}_a^*$
4. UpdateDensities $\rho_a$
5. Compute $\alpha_a$
6. CorrectDivergenceError()
7. UpdateVelocities $\boldsymbol{v}_a(t + \Delta t) = \boldsymbol{v}_a^*$
```

First, the viscous forces are applied and the system takes a half-step towards the intermediate state $(\boldsymbol{r}^*, \boldsymbol{u}^*, \rho^*)$. Second, the step *CorrectDensityError* and *CorrectDivergenceError* attempts to update $\boldsymbol{u}^*$ so that the updated velocity field represents a divergence-free vector field.

The acceleration that will be applied due to pressure forces is computed with:

$$\boldsymbol{a}_a^p(t) = -m_a \nabla P_a(t) \tag{3.26}$$

Then, instead of the classic pressure gradient operator (which Ihmsen also adopted earlier) it is assumed that the pressure gradient ∇P_a can be represented as

$$\nabla P_a = \kappa_a^v \nabla \rho_a = \kappa_a^v \sum_{b \in \mathcal{N}_a} m_b \nabla W_{ab} \tag{3.27}$$

where κ_a^v represents the stiffness parameter which we need to determine. After updating the velocity field to $\boldsymbol{v}^*$ (Step 2), the divergence of the velocity field is no longer zero. This results in a $\frac{D\rho_a}{Dt}$ that is non-zero:

$$\frac{D\rho_a}{Dt} = -\rho_a \nabla \cdot \boldsymbol{u}_a^* = \sum_{b \in \mathcal{N}_a} m_b \left(\boldsymbol{u}_a^* - \boldsymbol{u}_b^* \right) \nabla W_{ab} \tag{3.28}$$

If we could add-on the pressure force induced velocity changes to $\boldsymbol{u}^*$ so that $\frac{D\rho_a}{Dt}$ is small, that would be exactly what we are looking for. Let us call the total pressure force "density" acting on particle a to be $\boldsymbol{f}_a^p$ and the pressure "density" acting on a coming from particle b to be $\boldsymbol{f}_{a\leftarrow b}^p$,. Then the velocity change of particle a, and the velocity change of b counting only the contribition coming from particle a can be written as

$$\delta \boldsymbol{v}_a = \Delta t \frac{\boldsymbol{f}_a^p}{\rho_a},\ \delta \boldsymbol{v}_{b\leftarrow a} = \Delta t \frac{\boldsymbol{f}_{b\leftarrow a}^p}{\rho_a} \tag{3.29}$$

The term "Pressure force density" is a term that was used throughout the original paper, defined as $\boldsymbol{f}_a^p = -\nabla P_a$ with the unit $\left[\frac{N}{m^3}\right]$. The resulting density change resulting from this new velocity field should yield a density change that should cancel out $\frac{D\rho_i}{Dt}$:

$$\sum_{b\in\mathcal{N}_a} m_b \left(\delta \boldsymbol{v}_a - \delta \boldsymbol{v}_b\right) \nabla W_{ab} = \Delta t \sum_{b\in\mathcal{N}_a} m_v \left(\frac{\boldsymbol{f}_{b\leftarrow a}^p}{\rho_a} - \frac{\boldsymbol{f}_a^p}{\rho_a}\right) \nabla W_{ab} = -\frac{D\rho_a}{Dt} \tag{3.30}$$

substituting $\boldsymbol{f}_a^p = \nabla P_a = \kappa_a^v \nabla \rho_a = \kappa_a^v \sum_{b\in\mathcal{N}_a} m_b \nabla W_{ab}$,

$$\Delta t \sum_{b\in\mathcal{N}_a} m_b \left(\frac{\kappa_a^v m_b \nabla W_{ab}}{\rho_a} - \frac{\kappa_a^v \sum_{b\in\mathcal{N}_a} m_b \nabla W_{ab}}{\rho_a}\right) \nabla W_{ab} = -\frac{D\rho_a}{Dt} \tag{3.31}$$

Expanding out,

$$\frac{D\rho_a}{Dt} = \kappa_a^v \frac{\Delta t}{\rho_a} \left(\left|\sum_b m_b \nabla W_{ab}\right|^2 + \sum_b \left|m_b \nabla W_{ab}\right|^2\right) \tag{3.32}$$

We then can solve the system above wrt. κ:

$$\kappa_a^v = \frac{1}{\Delta t} \frac{D\rho_a}{Dt} \cdot \frac{\rho_a}{\alpha_a} \tag{3.33}$$

where $\alpha_a = \left|\sum_{b\in\mathcal{N}_a} m_b \nabla W_{ab}\right|^2 + \sum_{b\in\mathcal{N}_a} \left|m_b \nabla W_{abb}\right|^2$. Bender and Koschier observed that using Jacobi iterations, along with the clamping technique introduced by Ihmsen worked well for solving the above system of equations:

Enforcing Zero Density Change On top of the zero divergence enforcement, Bender et al., also included a second step for the pressure solver. It is possible to consider this as a separate solver that can be applied on its own. In Bender et al., this solver was applied once more independantly after the zero divergence solver to deal with the residual errors. Ignoring the zero divergence solver step, after The non-pressure forces, the intermediate density can be written as

$$\rho_a^* = \rho_a + \Delta t \frac{D\rho_a}{Dt} = \rho_a + \Delta t \cdot \sum_{b\in\mathcal{N}_a} m_b \left(\boldsymbol{v}_a^* - \boldsymbol{v}_b^*\right) \nabla W_{ab} \tag{3.34}$$

Algorithm 3.2 DFSPH - CorrectDivergenceError

```
while $\left(\left(\frac{D\rho}{Dt}\right)_{ave} > TOL_{div}\right)$ do
    for $Particle_a$ in all particles do
        Compute $\frac{D\rho_i}{Dt} = -\rho_a \nabla \cdot \boldsymbol{v}_a^*$
    end for
    for $Particle_a$ in all particles do
        $\kappa_a^v = \frac{1}{\Delta t}\frac{D\rho_a}{Dt} \cdot \frac{\rho_a}{\alpha_a}$, $\kappa_b^v = \frac{1}{\Delta t}\frac{D\rho_a}{Dt} \cdot \frac{\rho_b}{\alpha_b}$
        $\boldsymbol{v}^* \leftarrow \boldsymbol{v}^* - \Delta t \sum_{bb} m_b \left(\frac{\kappa_a^v}{\rho_a} + \frac{\kappa_b^v}{\rho_b}\right) \nabla W_{ab}$
    end for
end while
```

the pressure forces that will enforce zero density change can written as

$$\rho_a^* - \rho_0 = \Delta t^2 \sum_{b \in \mathcal{N}_a} m_b \left(\frac{\boldsymbol{f}_a^p}{\rho_a} - \frac{\boldsymbol{f}_{b \leftarrow a}^p}{\rho_a} \right) \nabla W_{ab} \tag{3.35}$$

defining (note that this is the same as the zero divergence solver, only with $\frac{D\rho_a}{Dt} = \sum_{b \in \mathcal{N}_a} m_b \left(\boldsymbol{v}_a^* - \boldsymbol{v}_b^*\right) \nabla W_{ab}$ replaced with $\frac{\rho_a^* - \rho_0}{\Delta t}$, we have the following stiffness value:

$$\kappa_a = \frac{1}{\Delta t^2}\left(\rho_a^* - \rho_0\right), \tag{3.36}$$

where we could use for Jacobi iterations.

Algorithm 3.3 DFSPH - CorrectDensityError

```
while $(\rho_{avg} - \rho_0)_{ave} > TOL_{dense}$ do
    for $Particle_i$ in all particles do
        Compute $\rho_a^*$
    end for
    for $Particle_a$ in all particles do
        $\kappa_a = \frac{\rho_a^* - \rho_0}{\Delta t^2}\alpha_a$, $\kappa_b = \frac{\rho_b^* - \rho_0}{\Delta t^2}\alpha_b$
        $\boldsymbol{v}^* \leftarrow \boldsymbol{v}^* - \Delta t \sum_b m_b \left(\frac{\kappa_a}{\rho_a} + \frac{\kappa_b}{\rho_b}\right) \nabla W_{ab}$
    end for
end while
```

3.2.2.4 Implicit Viscosity Solver

The viscosities of the fluids involved in DIW are usually extremely large compared to typical fluids due to their composition. The required timestep restriction almost renders explicit stepping methods impossible. The only plausible way is to use an implicit method that will provide reasonably large timestpes. Recently, Weiler et al. introduced a physically consistent implicit viscosity solver ([16, 122]). The fact that many previous implicit viscosity solvers meant for computer graphics applications were not physically consistent gives this approach a great advantage for solving engineering problems.

As stated before, the viscous forces in SPH is represented as

$$\left\langle \frac{\mu}{\rho} \nabla^2 \boldsymbol{v} \right\rangle_a = \sum_{b \in \mathcal{N}_a} \frac{8 \mu m_b \boldsymbol{r}_{ab} \cdot \nabla W_{ab}}{(\rho_a + \rho_b)^2 (\boldsymbol{r}_{ab}^2 + \epsilon)} \boldsymbol{u}_{ab} \tag{3.37}$$

Note that this is a slightly different operator compated to the one user in the original paper by Weiler et al. The velocity of the intermediate timestep can be written as

$$\boldsymbol{v}^* = \boldsymbol{v}(t) + \Delta t \frac{\mu}{\rho} \nabla^2 \boldsymbol{v}^* \tag{3.38}$$

The above system can be re-written as a system of linear equations:

$$(\boldsymbol{I} - \Delta t \boldsymbol{A}) \boldsymbol{v}^* = \boldsymbol{v}(t) \tag{3.39}$$

with block matricies $\boldsymbol{A}_{ab}$ (with a and b representing the rows and columns, respectively) that correspond to particle a and its neighbors

$$\boldsymbol{A}_{ab} = -\frac{8 \mu m_b}{(\rho_a + \rho_b)^2} \frac{\nabla W_{ab} \otimes \boldsymbol{r}_{ab}}{||\boldsymbol{r}_{ab}||^2 + 0.01 h^2}, \;\; \boldsymbol{A}_{aa} = -\sum_b \boldsymbol{A}_{ab} \tag{3.40}$$

The above system is a positive definite linear system, and can be solved with a method such as conjugate gradient.

3.3 Computational Optimization of Extrusions in DIW

3.3.1 Overview

Due to the unique mechanical properties of inks usually used in DIW processes, the final deposition geometry may not always closely follow the desired one. Some factors that can cause such issues are (but not limited to): Height from nozzle to substrate, acceleration of nozzle, extrusion rate, nozzle diameter, ink viscosity / surface tension. For example, when a nozzle changes its direction rapidly, the trailing extrusion of ink may "whip" and set at a undesired location. Experimental approaches on controlling such phenomena have been investigated recently in research articles such as [64, 65, 130, 59]. Our computational framework allows the practitioner to avoid such time-consuming procedures via a computational optimization method. Our goal will be to provide an algorithm that will "learn" the optimal printing parameters. A particle swarm optimization (PSO) algorithm was employed for the optimization procedure, while a single SPH simulation is instatiated for the evaluation of each particle (figure 3.3). Although optimization of

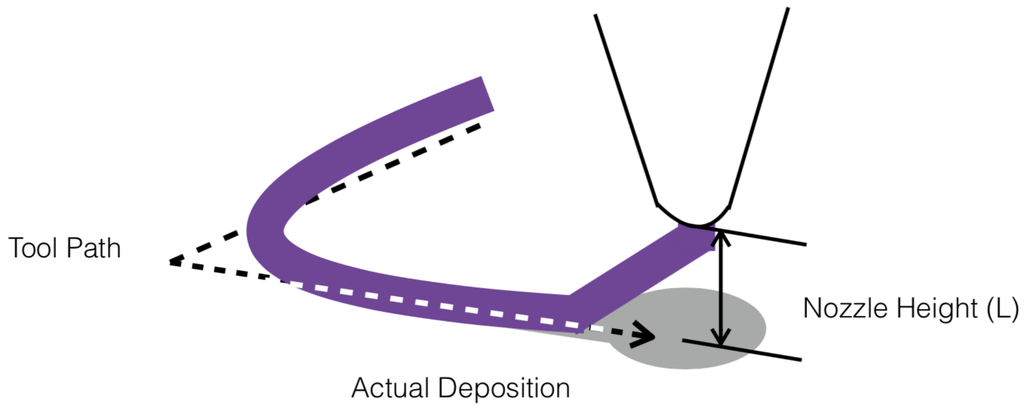


Figure 3.2: Ink being "dragged" by the nozzle . The ink may inhibit very-high viscosity values and large surface tension effects, or non-Newtonian effects ([64, 65, 130, 59]). In effect, this becomes an obstacle for obtaining a desired deposition, which can involve many experimental iterations.

a print can include a virtually endless amount of parameters (variable extrusion rate, variable nozzle motion speed, etc.), for our investigation we will only be considering optimizing the path for the nozzle to follow with all the other parameters fixed. We later show that this type of optimization technique alone can lead to significant improvements towards the quality of the deposition.

3.3.2 Waypoint generation

In order to optimize the nozzle path, we introduce "waypoints" that the nozzle must pass. Then, an interpolation scheme is used to connect the "waypoints", which is used as the actual path for the nozzle. The waypoints are then randomly generated so that they slightly deviate from the original path. The process follows:

1. The desired deposition path is converted into a parametric form:

$$\boldsymbol{p}_{desired}\left(s\right)=\begin{bmatrix} x\left(s\right) \\ y(s) \end{bmatrix},\ 0\leq s\leq 1 \tag{3.41}$$

2. N random values are then generated to produce a set of random points ($\boldsymbol{w}_i,\ 1\leq i\leq N$), that lie on the desired toolpath:

$$\left[\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_N\right]=\left[\boldsymbol{p}_{desired}\left(s_1\right),\boldsymbol{p}_{desired}\left(s_2\right),\cdots,\boldsymbol{p}_{desired}\left(s_N\right)\right] \tag{3.42}$$

3. For each point, the normal vector is calculated with

$$\boldsymbol{n}_i=\begin{bmatrix} \frac{dy}{ds}\big|_{s=s_i} \\ -\frac{dx}{ds}\big|_{s=s_i} \end{bmatrix},\ 1\leq i\leq N \tag{3.43}$$

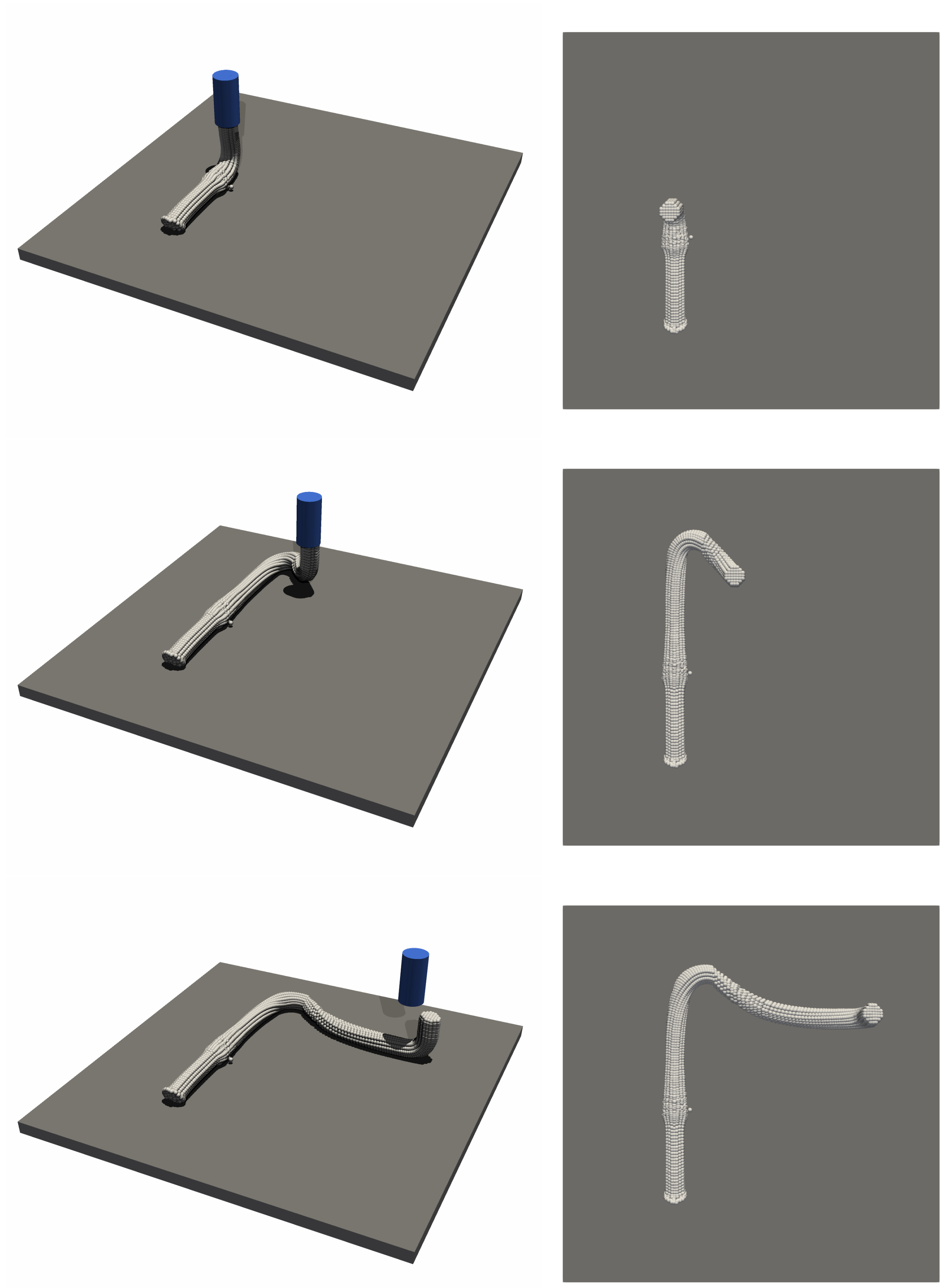


Figure 3.3: Computational Simulation of Ink Deposition with SPH. Blue object represents the nozzle, whilte the white particles are the SPH particles representing the ink.

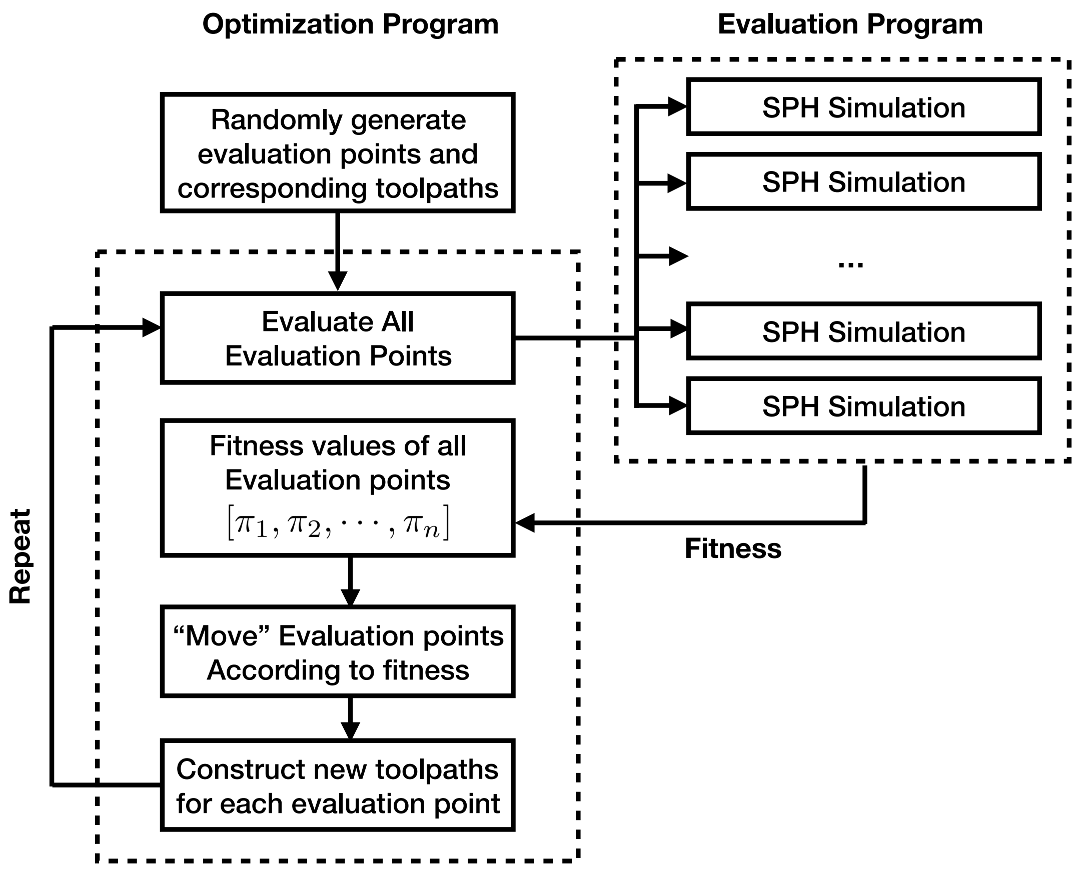

Optimization Program
Evaluation Program
Randomly generate evaluation points and corresponding toolpaths
Evaluate All Evaluation Points
Fitness values of all Evaluation points
$[\pi_1, \pi_2, \cdots, \pi_n]$
"Move" Evaluation points According to fitness
Construct new toolpaths for each evaluation point
Repeat
SPH Simulation
SPH Simulation
...
SPH Simulation
SPH Simulation
Fitness

Figure 3.4: Structure of our optimization framework, including the PSO optimizer and SPH based fitness value solver. The multiple evaluations corresponding to each evaluation point can be done simultaneously by invoking multiple solvers.

4. Each point is then offsetted by a random distance, where the distance is bounded by $-d \leq \delta_i \leq d$:

$$[\tilde{\boldsymbol{w}}_1, \tilde{\boldsymbol{w}}_2, \cdots, \tilde{\boldsymbol{w}}_N] = [\boldsymbol{w}_1 + \delta_1 \boldsymbol{n}_1, \boldsymbol{w}_2 + \delta_2 \boldsymbol{n}_2, \cdots, \boldsymbol{w}_N + \delta_N \boldsymbol{n}_N] \tag{3.44}$$

5. Using the offsetted points $[\tilde{\boldsymbol{w}}_1, \tilde{\boldsymbol{w}}_2, \cdots, \tilde{\boldsymbol{w}}_N]$, we may defined a new interpolated curve for the nozzle to follow:

$$\boldsymbol{p}_{nozzle}(s) = \begin{bmatrix} \tilde{x}(t) \\ \tilde{y}(t) \end{bmatrix}, \ 0 \leq t \leq 1 \tag{3.45}$$

The above procedure allows us to define a toolpath that is represented by a set of parameters:

$$\boldsymbol{\lambda} = \begin{bmatrix} s_1 & s_2 & \cdots & s_N \\ \delta_1 & \delta_2 & \cdots & \delta_N \end{bmatrix}^T \tag{3.46}$$

. Thus, for a randomized toolpath with N waypoints, there will be $2N$ parameters. Although it would be possible to characterize the toolpath in a different manner, we concluded that our characterization method reduces the parameters to be optimized, allowing practical, rapid optimizations.

After the generation of the waypoints, the PCHIP (Piecewise Cubic Hermite Interpolaying Polynomial) interapolation method was used to interpolate the variables between the waypoints. This specific interpolation scheme was preferrable, since the method does not overshoot within the interpolated region.

3.3.3 Particle Swarm Optimization (PSO)

We have employed the Particle Swarm Optimization (PSO, [68, 41, 109]) method to achieve our desired goals. In the PSO algorithm, a "swarm" is deployed throughout the parameter space, followed by evaluating the fitness of each "swarm member" (or "particle"). Then, each swarm member is relocated according to the algorithm shown in algorithm 3.4. The idea is that by adjusting the velocity of each swarm particle so that the particle will move towards a point where the difference between its current position and its personal best / global best tends to be minimized, the particle will converge onto a optimal point. Since the method does not require any type of gradient computation, implementation is straight forward. We also introduced randomized points during each timestep (inspired by genetic algorithms), replacing the lowest-performing swarm members. In our experience, PSO produced more stable results for our problem than only using Generic Algorithms (GA). For our application, the parameters that define the waypoints (as described in section 3.3.2) contructs the optimization space.

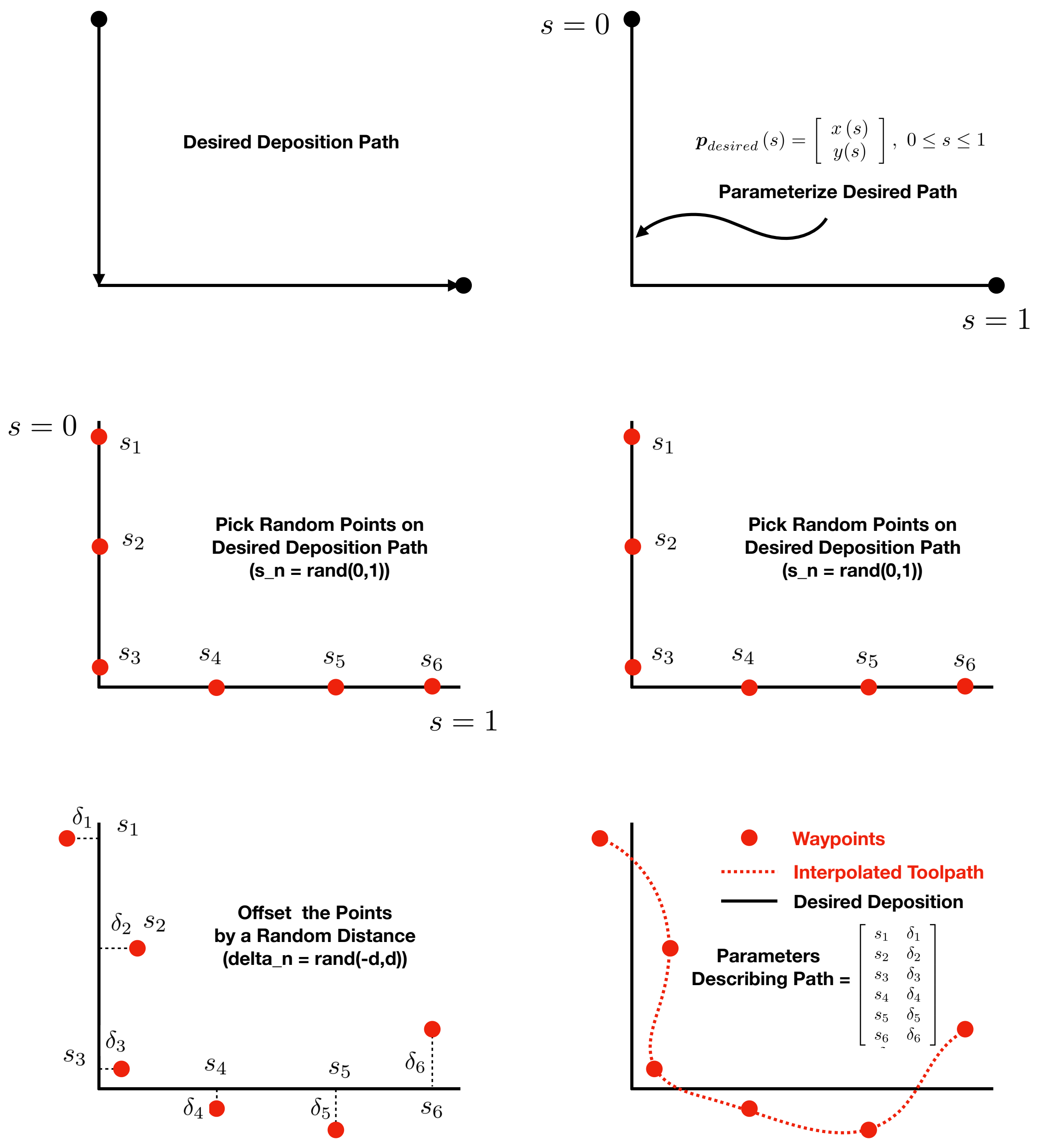

Desired Deposition Path
$s = 0$
$p_{desired}(s) = \begin{bmatrix} x(s) \\ y(s) \end{bmatrix}, \ 0 \le s \le 1$
Parameterize Desired Path
$s = 1$
s_1 s_2 s_3 s_4 s_5 s_6
Pick Random Points on Desired Deposition Path (s_n = rand(0,1))
δ_1 δ_2 δ_3 δ_4 δ_5 δ_6
Offset the Points by a Random Distance (delta_n = rand(-d,d))
Waypoints
Interpolated Toolpath
Desired Deposition
Parameters Describing Path = $\begin{bmatrix} s_1 & \delta_1 \\ s_2 & \delta_2 \\ s_3 & \delta_3 \\ s_4 & \delta_4 \\ s_5 & \delta_5 \\ s_6 & \delta_6 \end{bmatrix}$

Figure 3.5: Toolpath Generation Procedure (From Top Left to Bottom Right)

Algorithm 3.4 PSO Optimization

```
for $member_i$ in all Swarm Members do
    Initialize member position: $\boldsymbol{\lambda}_i = [\lambda_1^i, \lambda_2^i, \cdots, \lambda_k^i]$
    Initialize member velocity: $\boldsymbol{v}_i = [\dot{\lambda}_1^i, \dot{\lambda}_2^i, \cdots, \dot{\lambda}_k^i]$
end for
while $min(f(member_1), \cdots, f(member_N)) > TOL$ do
    for $member_i$ in all Swarm Members do
        Update member's best position: If $(f(\hat{\boldsymbol{\lambda}}_i) > f(\boldsymbol{\lambda}_i))$, $\hat{\boldsymbol{\lambda}}_i \leftarrow \boldsymbol{\lambda}_i$
        Update global best position: If $(f(\boldsymbol{\lambda}_{gb}) > f(\boldsymbol{\lambda}_i))$, $\boldsymbol{\lambda}_{gb} \leftarrow \boldsymbol{\lambda}_i$
    end for
    for $member_i$ in all Swarm Members do
        $\boldsymbol{v}_i \leftarrow \boldsymbol{v}_i + C_1 \cdot rand \cdot (\hat{\boldsymbol{\lambda}}_i - \boldsymbol{\lambda}_i) + C_2 \cdot rand \cdot (\boldsymbol{\lambda}_{gb} - \boldsymbol{\lambda}_i)$
        $\boldsymbol{\lambda}_i \leftarrow \boldsymbol{\lambda}_i + \boldsymbol{v}_i$
    end for
    for $member_i$ in $M$ poorest performing Swarm Members do
        Initialize member position: $\boldsymbol{\lambda}_i = [\lambda_1^i, \lambda_2^i, \cdots, \lambda_k^i]$
        Initialize member velocity: $\boldsymbol{v}_i = [\dot{\lambda}_1^i, \dot{\lambda}_2^i, \cdots, \dot{\lambda}_k^i]$
    end for
end while
```

3.4 Optimization Examples

The performance of the suggested computational optimization approach is shown by applying the method to several paths, all which are the most frequently occurring patterns in real DIW processes. The ink property was assumed to follow the constants listed in table 3.2 for our optimization attempts. In order to evaluate the fitness of the extruded topology, the following simple metric was used throughout our examples:

$$f(\boldsymbol{\lambda}) = \sum_{a \in \text{Particle Set}} \left(\frac{dist(\boldsymbol{r}_a)}{h} \right)^3 \tag{3.47}$$

where the function $dist(\boldsymbol{r}_a)$ represents the minimum distance from the SPH particle located at position $\boldsymbol{r}_a$ projected onto the substrate plane to the desired, ideal deposition path, and h is the smoothing length of the SPH discretization. The popular SPH Library SPlisHSPlasH ([15]) was modified for our fluid model, and to evaluate/output a fitness value for a given input file.

3 different deposition paths that are frequently utilized in additive manufacturing was considered. The parameters shown in Table 3.2 was assumed for all of the simulations.

Physical Parameters	Value
Fluid density (ρ)	$800\ kg/m^3$
Fluid viscosity (μ)	$0.1\ Pa \cdot s$
Fluid surface tension (σ)	$0.01\ N/m$
Nozzle Width (D)	$0.7\ mm$

Table 3.2: Fluid Properties

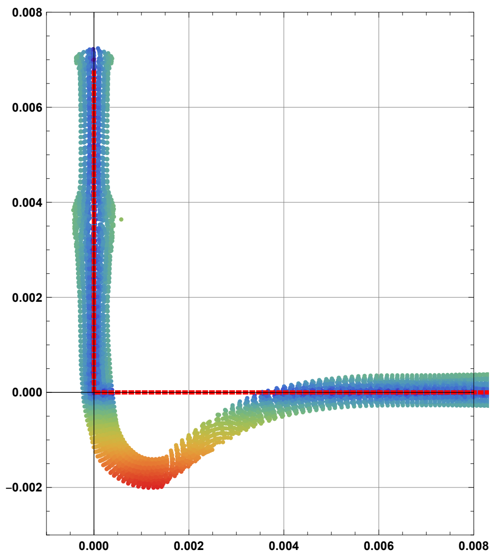


Figure 3.6: Naive Deposition (Dotted red: desired deposition path), Fitness=$2,021,340.0$

3.4.1 90 Degree Bend

3.4.1.1 Non-Optimized Extrusion Case

We first consider the case for the “naive” deposition, where the nozzle toolpath follows the same path as the deposition path (figure 3.6). Near the origin, the extruder abruptly changes its direction. We can observe that the extrusion material yet to be deposited onto the substrate carries on to move in the original direction due to its momentum. This creates an undesirable final deposition, where the maximum deviation was around 2.86 times the nozzle diameter. The fitness value of the naive case was $2,021,340.0$.

3.4.1.2 Optimized Extrusion Case

4 randomized waypoints, along with a fixed waypoint (table 3.3) was used for the optimization process. The evolution of the best fitness value for each iteration is shown in figure 3.7. Notice the fluctuations in the fitness value, since the PSO algorithm does not necessarily produce a monotonically-decreasing result. This is mainly due to the “inertia” of each swarm member. For each iteration, 100 swarm members were deployed throughout

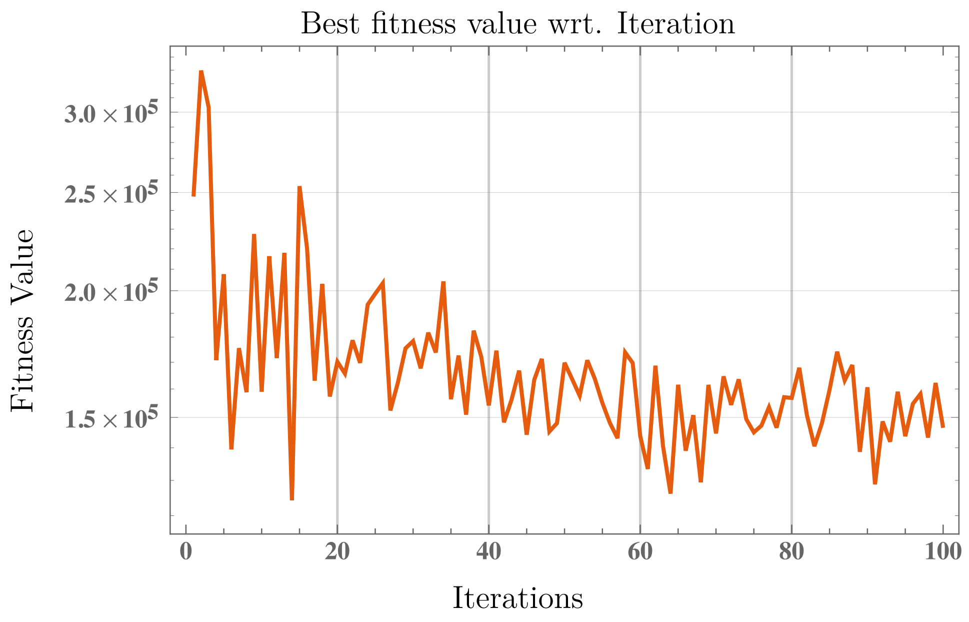


Figure 3.7: Evolution of Fitness Value, 90 Degree Bend

Optimization Parameters	Bounds (interval)
s_1, s_2, s_3	$(0.5, 0.9)$
$\delta_1, \delta_2, \delta_3$	$(0, 0.0015)$ $[m]$
Fixed $s_4 = 0$	N/A
δ_4	$(0, 0.0005)$ $[m]$
Fixed Waypoint @ $s = 0, \delta = 0$	N/A

Table 3.3: Bounds for Randomization (90 Degree Bend)

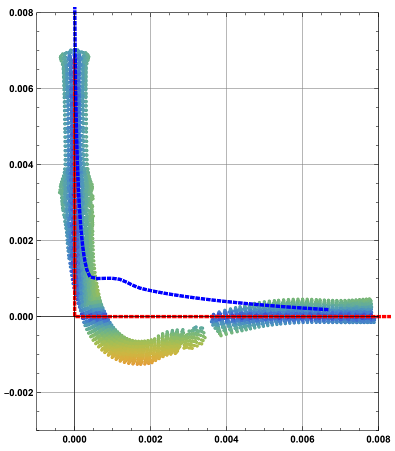
100 iterations (10,000 total simulations). The best performaing swarm member was found during iteration 63 and 90, as shown in 3.8. Although the optimized toolpath was not able to completely resolve the "intertial overshooting" near the bend, the deviation was greatly reduced compared to the naive deposition case.

For this specific case, 4 randomized waypoints were generated along the original toolpath, according to the characterization method described in 3.3.2. The bounds for the randomization step are shown in table 3.3.

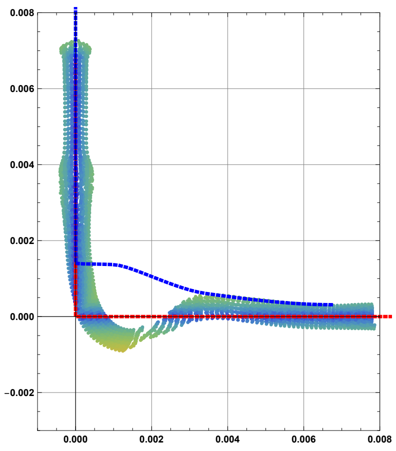
3.4.2 180 Degree Bend (Small Radius)

3.4.2.1 Non-Optimized Extrusion Case

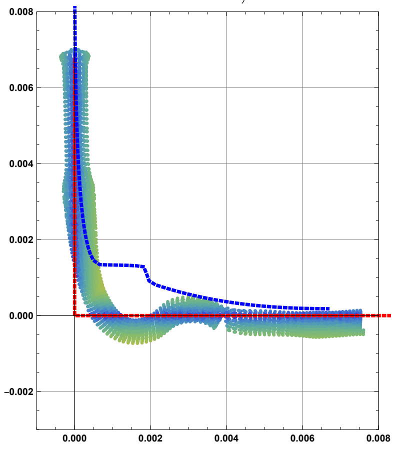
For the 180 degree bend case, the tooltip undergoes even more rapid acceleration. This causes more material to jet outwards from the desired path, as shown in figure 3.9.



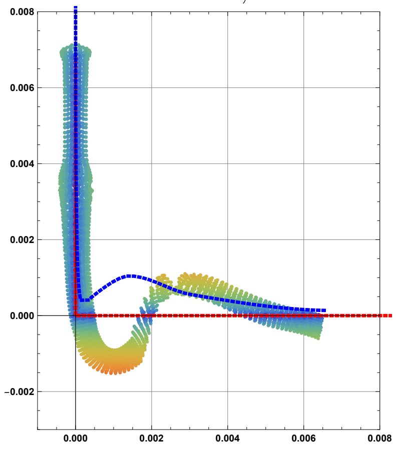
Best member of Iteration 1,
Fitness=$329,744.0$



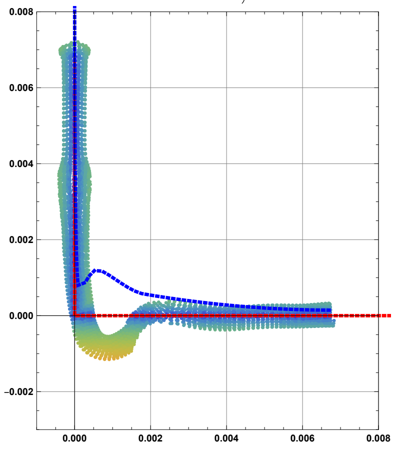
Best member of Iteration 5,
Fitness=$139,473.0$



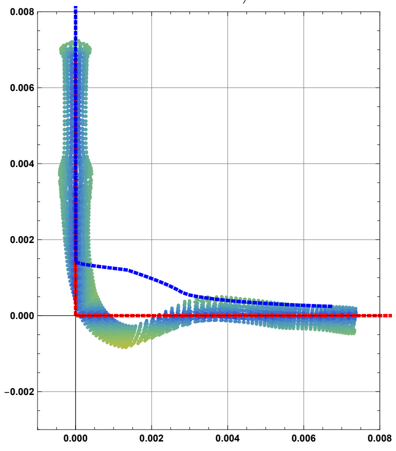
Best of Iteration 20,
Fitness=$165,691.0$



Best of Iteration 40,
Fitness=$174,549.0$



Best of Iteration 63,
Fitness=$126,164.0$



Best of Iteration 90,
Fitness=$128,844.0$

Figure 3.8: Naive Deposition (Dotted red: desired deposition path, Dotted blue: nozzle path)

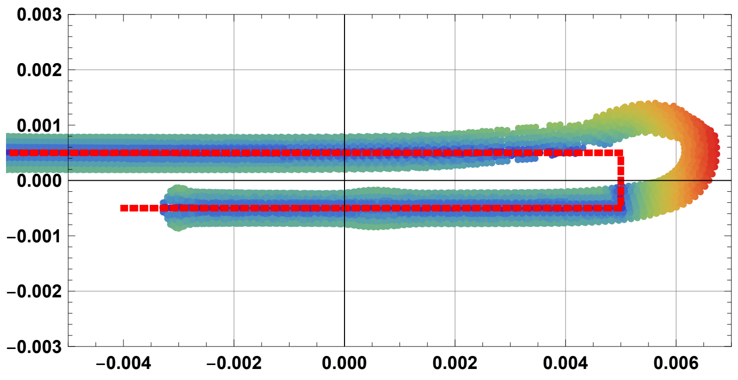


Figure 3.9: Naive Deposition (Dotted red: desired deposition path), Fitness=$631,201.0$

Optimization Parameters	Bounds (interval)
s_1	$(0.1, 0.324)$
Fixed $\delta_1 = 0$	N/A
s_2, s_3	$(0.324, 0.771)$
δ_2, δ_3	$(-0.002, 0.002)$ $[m]$
s_4	$(0.771, 0.9)$
Fixed $\delta_4 = 0$	N/A
Fixed Waypoint @ $s = 0, \delta = 0$	N/A
Fixed Waypoint @ $s = 1, \delta = 0$	N/A

Table 3.4: Randomization Parameters(Small Radius 180 Degree Bend)

For this case, the maximum deviation was around 2.48 times the nozzle diameter D, and the fitness value was $631,201.0$.

3.4.2.2 Optimized Extrusion Case

4 randomized waypoints, along with 2 fixed waypoints located on each end of the desired path (table 3.4) were used for the optimization process. The evolution of the best fitness value for each iteration is shown in figure 3.10. For each iteration, 200 swarm members were deployed throughout 100 iterations (20,000 total simulations). The best performing swarm member was found during iteration 83, as shown in 3.11. Although the deposition closely matches the desired one, one can observe a discontinuous section in the actual deposition. This is mainly due to the fact that no penalization regarding such effects were enforced in our algorithm. For example, using a different fitness formulation where the average distance between the SPH particles is penalized, will try to avoid any such discontinuous depositions.

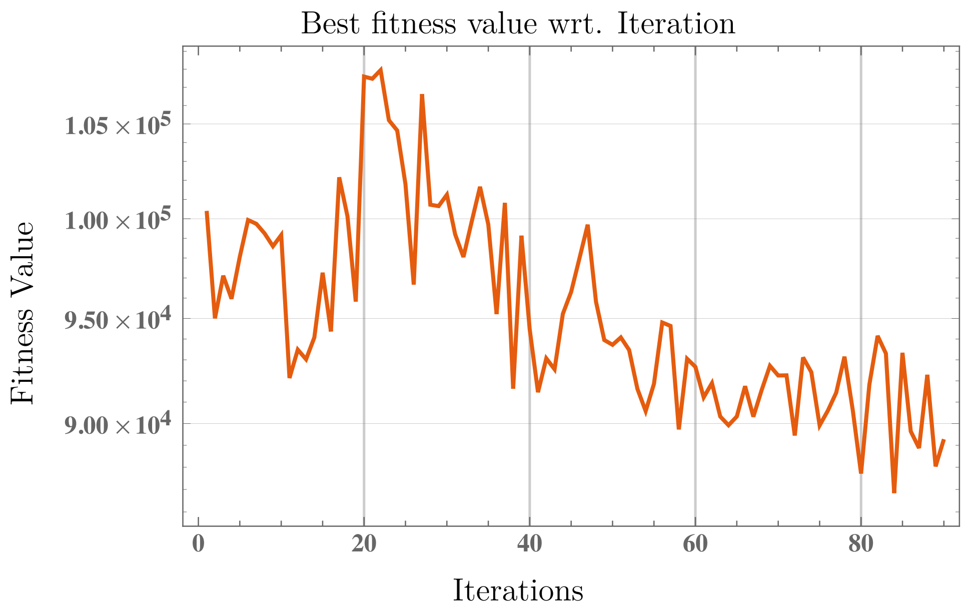


Figure 3.10: Evolution of Fitness Value, 180 Degree Small Radius Bend

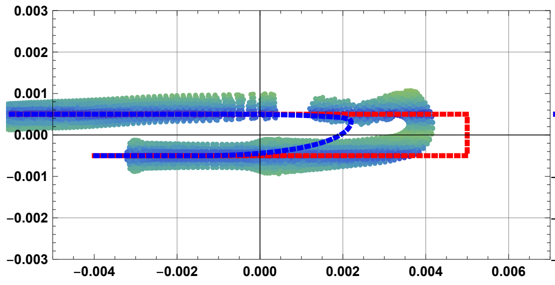
3.4.3 180 Degree Bend (Large Radius)

3.4.3.1 Non-Optimized Extrusion Case

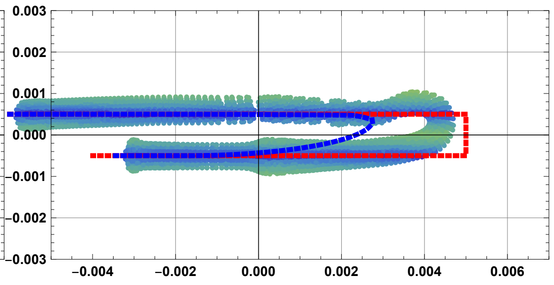
The centrifugal forces acting on the stream of fluid forces the deposition to land outside the desired toolpath, although the effect is not as magnified seen in the small radius 180 degree bend. The maximum deviation from the desired toolpath for this case was 1.28 times the nozzle diameter, with a fitness value of $870,687.0$.

3.4.3.2 Optimized Extrusion Case

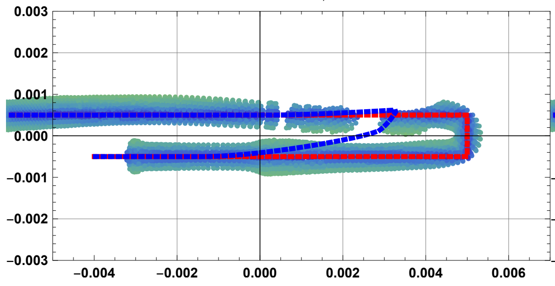
5 randomized waypoints, along with 2 fixed waypoints located on each end of the desired path (table 3.5) were used for the optimization process. The evolution of the best fitness value for each iteration is shown in figure 3.13. For each iteration, 200 swarm members were deployed throughout 100 iterations (20,000 total simulations). The best performing swarm member was found during iteration 83, as shown in 3.14. We observe that the actual deposition matches the desired one closely, although some improvements regarding maintaining a constant radius can be further considered. This can be done by increasing the number of randomized points, or introducing additional optimization parameters as mentioned earlier in the paper.



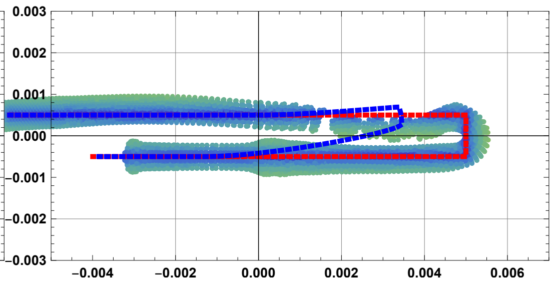
Best member of Iteration 1, Fitness$=100,407.0$



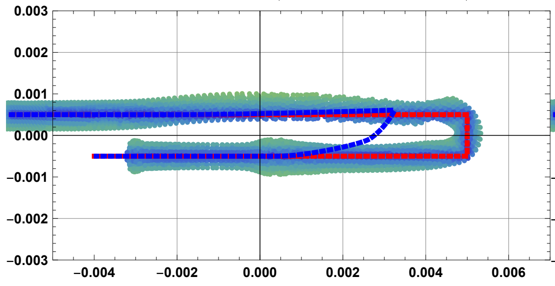
Best member of Iteration 5, Fitness$=99935.0$



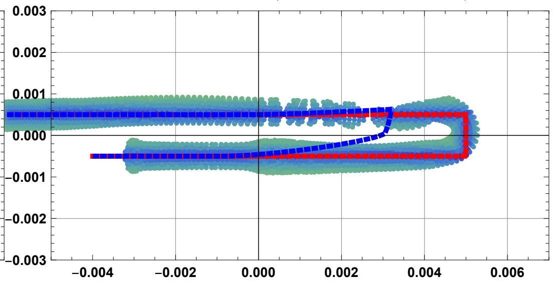
Best of Iteration 10, Fitness$=921,134.0$



Best of Iteration 20, Fitness$=107,471.0$



Best of Iteration 40, Fitness$=91,463.0$



Best of Iteration 83, Fitness$=86,837.0$

Figure 3.11: Optimized Deposition (Dotted red: desired deposition path, Dotted blue: nozzle path)

Optimization Parameters	Bounds (interval)
s_1, s_2, s_3, s_4, s_5	$(0.2, 0.8)$
$\delta_1, \delta_2, \delta_3, \delta_4, \delta_5$	$(-0.002, 0)$ $[m]$
Fixed Waypoint @ $s = 0, \delta = 0$	N/A
Fixed Waypoint @ $s = 1, \delta = 0$	N/A

Table 3.5: Bounds for Randomization (Large Radius 180 Degree Bend)

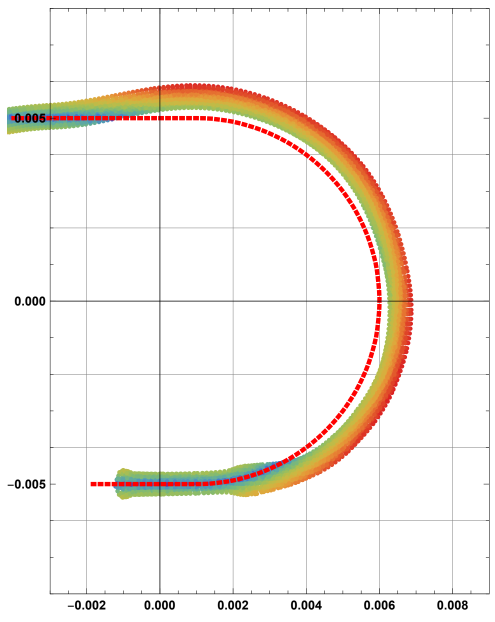


Figure 3.12: Naive Deposition (Dotted red: desired deposition path), Fitness=$870,687.0$

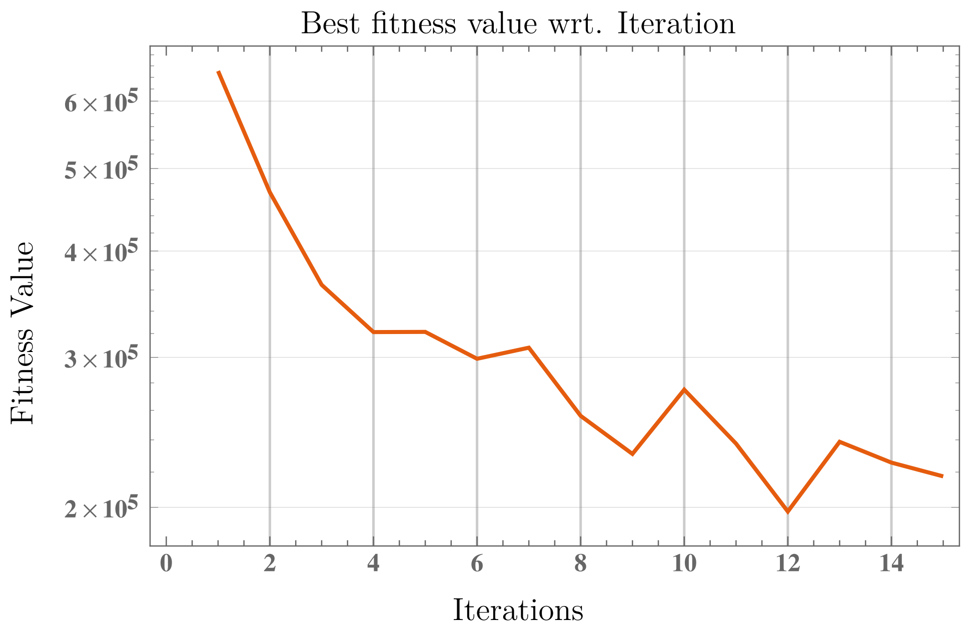
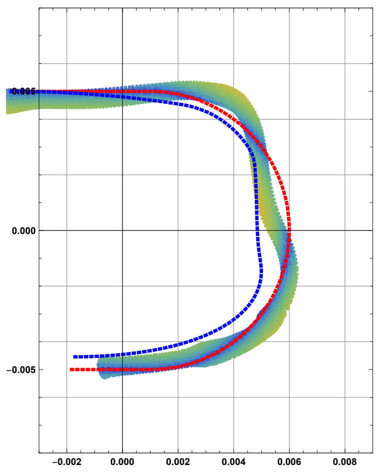
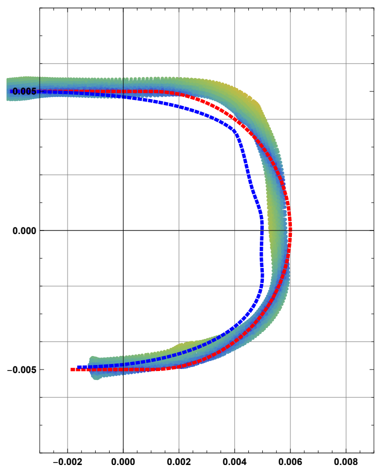


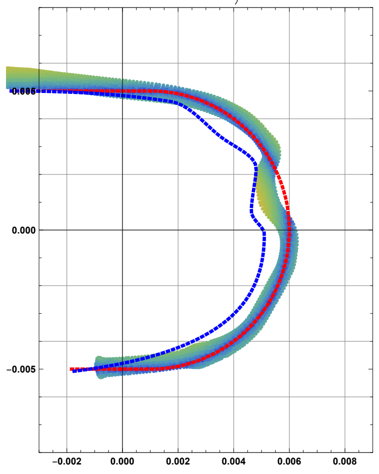
Figure 3.13: Evolution of Fitness Value, 180 Degree Large Radius Bend



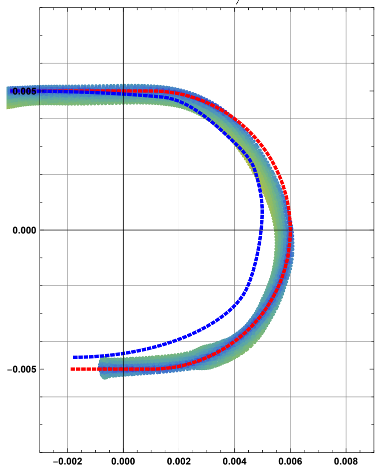
Best member of Iteration 1,
Fitness=468, 858.0



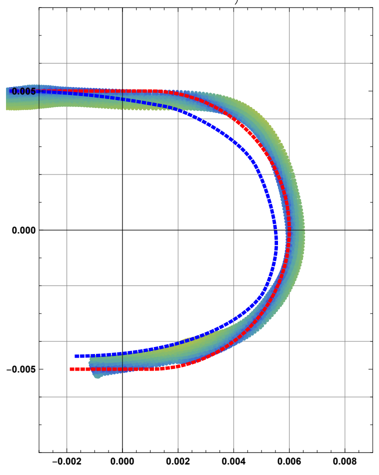
Best member of Iteration 2,
Fitness=365, 053.0



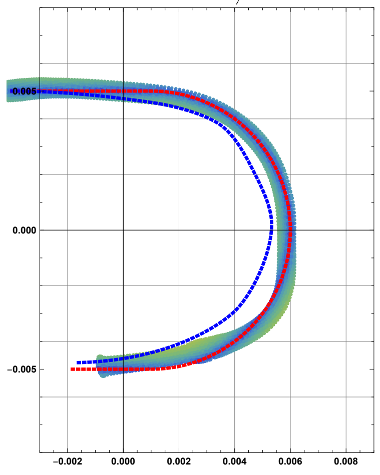
Best of Iteration 5,
Fitness=298, 880.0



Best of Iteration 7,
Fitness=256, 119.0



Best of Iteration 10,
Fitness=237, 600.0



Best of Iteration 11,
Fitness=197, 738.0

Figure 3.14: Optimized Deposition (Dotted red: desired deposition path, Dotted blue: nozzle path)

3.5 Conclusion

3.5.1 Discussion

We have proposed an extrusion optimization framework based on physical computational simulations that can account for the non-intuitive behavior of inks used in DIW. Based on several test cases, we were able to obtain final extrusion morphology that are much closer to the intended shape, compared to a “naive” extrusion path. This can be observed by noticing the fitness value decreasing approximately 3 orders of magnitude. Note that the optimal toolpath was characterized only with spatial waypoints for our test cases. We believe our computation based optimization method is capable of addressing many difficulties that arise regarding parameter tuning during experimental set-ups ([48, 73, 125]).

3.5.2 Limitations

3.5.2.1 Ink Model

For shear-thinning non-Newtonian fluids, there exists a sharp transition of the effective viscosity values between the yielded region and the unyielded region. The difference in viscosity is at least more than 4~5 orders of magnitude higher in the unyielded region, and requires an implicit viscosity step. Although previous work exist for simulating such fluids with SPH ([108, 57, 127]), in our implementation experience, this is extremely difficult to efficiently simulate with SPH, due to the large number of particles that are required to accurately represent the transition. Thus, a Newtonian fluid model with large viscosity and surface tension was employed instead. Although it is still possible to calibrate one’s ink model to our specified fluid model, this may come to an inconvenience, since ink models that are calibrated using non-Newtonian fluids seem to be relatively easier to find. Thus, the usage of other numerical methods such as MPM (Material Point Method) or FEM (Finite Element Method) seems reasonable, and is currently under investigation by the authors.

3.5.2.2 Parameter Space

It is expected that even more optimized printing parameters may be found by introducing additional complexity / parameters to the characterization of the toolpath. For example, some additional parameters that may be added onto the waypoints may include extrusion rates and the tooltip speed, while employing an approach similar to what was implemented in our example between the waypoints for such additional parameters. Tuning the properties of the ink is also a practice regularly performed by DIW researchers ([64, 112, 101, 125]). This can also be accounted for by allowing the optimization algorithm to perform a search on various fluid parameters such as viscosity, density and shear-thinning properties.

Bibliography

[1] S. Adami, X. Y. Hu, and N. A. Adams. A new surface tension formulation for multi-phase SPH using a reproducing divergence approximation. *Journal of Computational Physics*, March 2010.

[2] FJ Adrian, J Bohandy, BF Kim, An N Jette, and P Thompson. A study of the mechanism of metal deposition by the laser-induced forward transfer process. *Journal of Vacuum Science & Technology B: Microelectronics Processing and Phenomena*, 5(5):1490–1494, 1987.

[3] Nadir Akinci, Gizem Akinci, and Matthias Teschner. Versatile surface tension and adhesion for SPH fluids. *ACM Transactions on Graphics (TOG)*, 32(6):182, 2013.

[4] Martin S. Alnæs, Jan Blechta, Johan Hake, et al. The FEniCS Project Version 1.5. *Archive of Numerical Software*, 3(100), 2015.

[5] Alshaer, Rogers, and Li. Smoothed Particle Hydrodynamics (SPH) modelling of transient heat transfer in pulsed laser ablation of Al and associated free-surface problems. *Computational Materials Science*, 127:161–179, 2017. cited By 0.

[6] Andrea Amicarelli, Jean-Christophe Marongiu, Francis Leboeuf, Julien Leduc, and Joëlle Caro. SPH truncation error in estimating a 3D function. *Computers & Fluids*, 44(1):279–296, 2011.

[7] L Anand and NM Ames. On modeling the micro-indentation response of an amorphous polymer. *International journal of plasticity*, 22(6):1123–1170, 2006.

[8] Lallit Anand, Nicoli M Ames, Vikas Srivastava, and Shawn A Chester. A thermo-mechanically coupled theory for large deformations of amorphous polymers. Part I: Formulation. *International Journal of Plasticity*, 25(8):1474–1494, 2009.

[9] Carla Antoci, Mario Gallati, and Stefano Sibilla. Numerical simulation of fluid–structure interaction by SPH. *Computers & Structures*, 85(11-14):879–890, 2007.

[10] Carla Antoci, Mario Gallati, and Stefano Sibilla. Numerical simulation of fluid-structure interaction by SPH. *Computers & Structures*, 85(11-14):879–890, 2007.

[11] M. Antuono, A. Colagrossi, S. Marrone, and C. Lugni. Propagation of gravity waves through an SPH scheme with numerical diffusive terms. *Computer Physics Communications*, 182(4):866–877, 2011.

[12] M. Antuono, A. Colagrossi, S. Marrone, and D. Molteni. Free-surface flows solved by means of SPH schemes with numerical diffusive terms. *Computer Physics Communications*, 181(3):532–549, 2010.

[13] Matteo Antuono, Andrea Colagrossi, Salvatore Marrone, and Diego Molteni. Free-surface flows solved by means of SPH schemes with numerical diffusive terms. *Computer Physics Communications*, 181(3):532–549, 2010.

[14] Klaus-Jürgen Bathe. *Finite element procedures*. Klaus-Jurgen Bathe, 2006.

[15] Jan Bender. SPlisHSPlasH Library. *https://github.com/InteractiveComputerGraphics/SPlisHSPlasI* 2017.

[16] Jan Bender and Dan Koschier. Divergence-free SPH for incompressible and viscous fluids. *IEEE Transactions on Visualization and Computer Graphics*, 23(3):1193–1206, 2017.

[17] F. Bierbrauer, P.C. Bollada, and T.N. Phillips. A consistent reflected image particle approach to the treatment of boundary conditions in smoothed particle hydrodynamics. *Computer Methods in Applied Mechanics and Engineering*, 198(41-44):3400–3410, 2009.

[18] J. Bonet and S. Kulasegaram. Correction and stabilization of smooth particle hydrodynamics methods with applications in metal forming simulations. *Int. J. Numer. Methods Engineering*, 47(6):1189–1214, Feb. 2000.

[19] J. Bonet and S. Kulasegaram. Remarks on tension instability of Eulerian and Lagrangian corrected smooth particle hydrpdynamics (CSPH) methods. *Int. J. Numer. Methods Engineering*, 52:1203–2120, 2001.

[20] Jeremiah U Brackbill, Douglas B Kothe, and Charles Zemach. A continuum method for modeling surface tension. *Journal of computational physics*, 100(2):335–354, 1992.

[21] L. Brookshaw. A method of calculating radiative heat diffusion in particle simulations. *Proc. Astron. Soc. Aust.*, 6(2):207–210, 1985.

[22] Alexander V Bulgakov, Nathan Goodfriend, Oleg Nerushev, et al. Laser-induced transfer of nanoparticles for gas-phase analysis. *JOSA B*, 31(11):C15–C21, 2014.

[23] Agostino Antonio Cannarozzi and Francesco Ubertini. A mixed variational method for linear coupled thermoelastic analysis. *International Journal of Solids and Structures*, 38(4):717–739, 2001.

[24] John Phillip Carter and John R Booker. Finite element analysis of coupled thermoelasticity. *Computers & Structures*, 31(1):73–80, 1989.

[25] J.L. Cercos-Pita, R.A. Dalrymple, and A. Herault. Diffusive terms for the conservation of mass equation in {SPH}. *Applied Mathematical Modelling*, 40(19):8722 – 8736, 2016.

[26] Sun Choi, Arash Jamshidi, Tae Joon Seok, et al. Fast, high-throughput creation of size-tunable micro/nanoparticle clusters via evaporative self-assembly in picoliter-scale droplets of particle suspension. *Langmuir*, 28(6):3102–3111, 2012.

[27] Sun Choi, Stefano Stassi, Albert P Pisano, and Tarek I Zohdi. Coffee-ring effect-based three dimensional patterning of micro/nanoparticle assembly with a single droplet. *Langmuir*, 26(14):11690–11698, 2010.

[28] A. Chorin. Numerical study of slightly viscous flow. *Journal of Fluid Mechanics*, 57(04):785–796, 1973.

[29] Alexandre J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, 22(104):745–762, October 1968.

[30] Jung-Hoon Chun, Christian H Passow, and Nam P Suh. Droplet-based manufacturing. *CIRP ANNALS-manufacturing technology*, 42(1):235–238, 1993.

[31] P. Cleary and J.J. Monaghan. Conduction modelling using smoothed particle hydrodynamics. *J. Comp. Phys.*, 148:227–264, 1999.

[32] Paul W Cleary. Modelling confined multi-material heat and mass flows using SPH. *Applied Mathematical Modelling*, 22(12):981–993, 1998.

[33] A. Colagrossi, M. Antuono, A. Souto-Iglesias, D. Le Touzé, and P. Izaguirre-Alza. Theoretical analysis of SPH in simulating free-surface viscous flows. In *5th ERCOFTAC SPHERIC workshop on SPH applications*, 2010.

[34] A. Colagrossi and M. Landrini. Numerical Simulation of Interfacial Flows by Smoothed Particle Hydrodynamics. *J. Comp. Phys.*, 191:448–475, 2003.

[35] Liying Cui, Yingfeng Li, Jingxia Wang, et al. Fabrication of large-area patterned photonic crystals by ink-jet printing. *Journal of Materials Chemistry*, 19(31):5499–5502, 2009.

[36] S.J. Cummins. Projection methods and SPH. Technical report, Monash University, 1997.
1 COPIA. Report de Monash sobre proyeccion...

[37] S.J. Cummins. *Applications of Projection Techniques for incompressible flows.* PhD thesis, Monash University, 2000.

[38] S.J. Cummins and M. Rudman. An SPH projection method. *J. Comp. Phys.*, 152(2):584–607, July 1999.

[39] VI Danilovskaya. On a dynamical problem of thermoelasticity. *Prikl. Mat. Mekh*, 16(3):341–344, 1952.

[40] Philippe Delaporte and Anne-Patricia Alloncle. Laser-induced forward transfer: a high resolution additive manufacturing technology. *Optics & Laser Technology*, 78:33–41, 2016.

[41] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. Ieee, 1995.

[42] Marco Ellero, Mar Serrano, and Pep Español. Incompressible smoothed particle hydrodynamics. *Journal of Computational Physics*, 226(2):1731–1752, October 2007.

[43] R Fatehi and MT Manzari. A remedy for numerical oscillations in weakly compressible smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 67(9):1100–1114, 2011.

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

[45] Natalja E Fedorovich, Wouter Schuurman, Hans M Wijnberg, et al. Biofabrication of osteochondral tissue equivalents by printing topologically defined, cell-laden hydrogel scaffolds. *Tissue Engineering Part C: Methods*, 18(1):33–44, 2011.

[46] LL Ferrás, JM Nóbrega, and FT Pinho. Analytical solutions for Newtonian and inelastic non-Newtonian flows with wall slip. *Journal of Non-Newtonian Fluid Mechanics*, 175:76–88, 2012.

[47] Georg C Ganzenmüller. An hourglass control algorithm for Lagrangian smooth particle hydrodynamics. *Computer Methods in Applied Mechanics and Engineering*, 286:87–106, 2015.

[48] Gregory M Gratson, Mingjie Xu, and Jennifer A Lewis. Microperiodic structures: Direct writing of three-dimensional webs. *Nature*, 428(6981):386, 2004.

[49] JP Gray, JJ Monaghan, and RP Swift. SPH elastic dynamics. *Computer methods in applied mechanics and engineering*, 190(49):6641–6662, 2001.

[50] Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

[51] P. M. Guilcher, G. Ducrozet, B. Alessandrini, and P. Ferrant. Water wave propagation using SPH models. In *2nd SPHERIC workshop*, May 2007.

[52] Morton E Gurtin. *An introduction to continuum mechanics*, volume 158. Academic press, 1982.

[53] Luhui Han and Xiangyu Hu. SPH modeling of fluid-structure interaction. *Journal of Hydrodynamics*, 30(1):62–69, 2018.

[54] Jennifer N Hanson Shepherd, Sara T Parker, Robert F Shepherd, et al. 3D microperiodic hydrogel scaffolds for robust neuronal cultures. *Advanced functional materials*, 21(1):47–54, 2011.

[55] Tommy Hinks. Poisson Disc Samping. https://github.com/thinks/poisson-disk-sampling, 2015.

[56] Kenzo Hiraoka, Riou Takaishi, Daiki Asakawa, Yuji Sakai, and Yoshitoki Iijima. Surface characterization of polymethylmetacrylate bombarded by charged water droplets. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, 27(4):748–753, 2009.

[57] SM Hosseini, MT Manzari, and SK Hannani. A fully explicit three-step SPH algorithm for simulation of non-Newtonian fluid flow. *International Journal of Numerical Methods for Heat & Fluid Flow*, 17(7):715–735, 2007.

[58] X.Y. Hu and N.A. Adams. A multi-phase SPH method for macroscopic and mesoscopic flows. *J. Comp. Phys.*, 213 (2):844–861, 2006.

[59] YongAn Huang, Yongqing Duan, Yajiang Ding, et al. Versatile, kinetically controlled, high precision electrohydrodynamic writing of micro/nanofibers. *Scientific reports*, 4:5949, 2014.

[60] Markus Ihmsen, Nadir Akinci, Markus Becker, and Matthias Teschner. A Parallel SPH Implementation on Multi-Core CPUs. In *Computer Graphics Forum*, volume 30, pages 99–112. Wiley Online Library, 2011.

[61] Markus Ihmsen, Jens Cornelis, Barbara Solenthaler, Christopher Horvath, and Matthias Teschner. Implicit incompressible SPH. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):426–435, 2014.

[62] S. Inutsuka. Reformulation of Smoothed Particle Hydrodynamics with Riemann Solver. *J. Comp. Phys.*, 179:238–2667, 2002.

[63] JH Jeong, MS Jhon, JS Halow, and J Van Osdol. Smoothed particle hydrodynamics: Applications to heat conduction. *Computer Physics Communications*, 153(1):71–84, 2003.

[64] Yifei Jin, Wenxuan Chai, and Yong Huang. Printability study of hydrogel solution extrusion in nanoclay yield-stress bath during printing-then-gelation biofabrication. *Materials Science and Engineering: C*, 80:313–325, 2017.

[65] Yifei Jin, Danyang Zhao, and Yong Huang. Study of extrudability and standoff distance effect during nanoclay-enabled direct printing. *Bio-Design and Manufacturing*, 1(2):123–134, 2018.

[66] Gordon R Johnson and Stephen R Beissel. Normalized smoothing functions for SPH impact computations. *International Journal for Numerical Methods in Engineering*, 39(16):2725–2741, 1996.

[67] Nicholas T Kattamis, Priscilla E Purnick, Ron Weiss, and Craig B Arnold. Thick film laser induced forward transfer for deposition of thermally and mechanically sensitive materials. *Applied Physics Letters*, 91(17):171120, 2007.

[68] James Kennedy. Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766, 2010.

[69] Dan Koschier. CompactNSearch. https://github.com/thinks/poisson-disk-sampling, 2017.

[70] J Leduc, F Leboeuf, M Lance, E Parkinson, and JC Marongiu. Improvement of multiphase model using preconditioned Riemann solvers. In *5th International SPHERIC workshop*, 2010.

[71] J Leduc, JC Marongiu, F Leboeuf, M Lance, and E Parkinson. Multiphase SPH: a new model based on acoustic Riemann solver. In *4th International SPHERIC workshop*, pages 8–13, 2009.

[72] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. Siam, 2007.

[73] Jennifer A Lewis. Direct ink writing of 3D functional materials. *Advanced Functional Materials*, 16(17):2193–2204, 2006.

[74] Larry D Libersky, Albert G Petschek, Theodore C Carney, Jim R Hipp, and Firooz A Allahdadi. High strain Lagrangian hydrodynamics: a three-dimensional SPH code for dynamic material response. *Journal of computational physics*, 109(1):67–75, 1993.

[75] MB Liu and GR Liu. Smoothed particle hydrodynamics (SPH): an overview and recent developments. *Archives of computational methods in engineering*, 17(1):25–76, 2010.

[76] S Ma, X Zhang, and XM Qiu. Comparison study of MPM and SPH in modeling hypervelocity impact problems. *International Journal of Impact Engineering*, 36(2):272–282, 2009.

[77] F. Macià, A. Colagrossi, M. Antuono, and A. Souto-Iglesias. Benefits of using a Wendland kernel for free-surface flows. In *6th ERCOFTAC SPHERIC workshop on SPH applications*, 2011.

[78] F. Macià, A. Colagrossi, M. Antuono, and A. Souto-Iglesias. Benefits of using a Wendland kernel for free-surface flows. In *6th ERCOFTAC SPHERIC workshop on SPH applications*, pages 30–37. Hamburg University of Technology, 2011.

[79] Fabricio Macia Lang, Antonio Souto Iglesias, Matteo Antuono, and A Colagrossi. Benefits of using a Wendland kernel for free-surface flows. 2011.

[80] S. Marrone, M. Antuono, A. Colagrossi, et al. Delta-SPH model for simulating violent impact flows. *Computer Methods in Applied Mechanics and Engineering*, 200(13-16):1526–1542, 2011.

[81] S. Marrone, A. Colagrossi, M. Antuono, G. Colicchio, and G. Graziani. An accurate SPH modeling of viscous flows around bodies at low and moderate Reynolds numbers. *Journal of Computational Physics*, 245:456–475, 2013.

[82] S. Marrone, A. Colagrossi, M. Antuono, C. Lugni, and M.P. Tulin. A 2D+t SPH model to study the breaking wave pattern generated by fast ships. *Journal of Fluids and Structures*, 27(8):1199–1215, 2011.

[83] S. Marrone, A. Colagrossi, D. Le Touzé, and G. Graziani. Fast free-surface detection and level-set function definition in SPH solvers. *Journal of Computational Physics*, 229(10):3652–3663, 2010.

[84] Frank A McClintock and Ali S Argon. Mechanical behavior of materials. *BOOKS*, 1966.

[85] Sarah Michna, Willie Wu, and Jennifer A Lewis. Concentrated hydroxyapatite inks for direct-write assembly of 3-D periodic scaffolds. *Biomaterials*, 26(28):5632–5639, 2005.

[86] Hiromi Minemawari, Toshikazu Yamada, Hiroyuki Matsui, et al. Inkjet printing of single-crystal films. *Nature*, 475(7356):364, 2011.

[87] J. J. Monaghan. Smoothed particle hydrodynamics. *Reports on Progress in Physics*, 68:1703–1759, 2005.

[88] J.J. Monaghan. Particle methods for hydrodynamics. *Computer-physics-reports*, 3(2):71–124, 1985.

[89] J.J. Monaghan. An introduction to SPH. *Computer Physics Communications*, pages 89–96, 1988.

[90] J.J. Monaghan. Simulating Free Surface Flows with SPH. *J. Comp. Phys.*, 110(2):39–406, 1994.

[91] J.J. Monaghan. SPH Technical NOTEs : Improved Modelling of boundaries. Technical report, University of Monash, 1995.

[92] JJ Monaghan. Smoothed particle hydrodynamics and its diverse applications. *Annual Review of Fluid Mechanics*, 44:323–346, 2012.

[93] J.J. Monaghan and J.B. Kajtar. SPH particle boundary forces for arbitrary boundaries. *Computer Physics Communications*, 180(10):1811–1820, 2009.

[94] J.J. Monaghan and J.C. Lattanzio. A refined particle method for astrophysical problems. *Astron. Astrophys.*, 149:135, 1985.

[95] J.J. Monaghan and S.R. Varnas. Artificial viscosity e.t.c. *MNRAS*, 231:515, 1988.

[96] Joe J Monaghan. Simulating free surface flows with SPH. *Journal of computational physics*, 110(2):399–406, 1994.

[97] Joseph P Morris. Simulating surface tension with smoothed particle hydrodynamics. *International journal for numerical methods in fluids*, 33(3):333–353, 2000.

[98] J.P. Morris. Simulating surface tension with smoothed particle hydrodynamics. *International Journal for Numerical Methods in Fluids*, 33(3):333–353, 2000.

[99] Guillaume Oger, Mathieu Doring, Bertrand Alessandrini, and Pierre Ferrant. An improved SPH method: Towards higher order convergence. *Journal of Computational Physics*, 225(2):1472–1492, 2007.

[100] Bong Kyun Park, Dongjo Kim, Sunho Jeong, Jooho Moon, and Jang Sub Kim. Direct writing of copper conductive patterns by ink-jet printing. *Thin solid films*, 515(19):7706–7711, 2007.

[101] Jungho Park, Jooho Moon, Hyunjung Shin, Dake Wang, and Minseo Park. Direct-write fabrication of colloidal photonic crystal microarrays by ink-jet printing. *Journal of colloid and interface science*, 298(2):713–719, 2006.

[102] Jingyuan Qu, Muamer Kadic, Andreas Naber, and Martin Wegener. Microstructured two-component 3D metamaterials with negative thermal-expansion coefficient from positive constituents. *Scientific reports*, 7:40643, 2017.

[103] T Rabczuk, T Belytschko, and SP Xiao. Stable particle methods based on Lagrangian kernels. *Computer methods in applied mechanics and engineering*, 193(12-14):1035–1063, 2004.

[104] P.W. Randles and L.D. Libersky. Smoothed Particle Hydrodynamics: some recent improvements and applications. *Computer methods in applied mechanics and engineering*, 39:375–408, 1996.

[105] M Sanz, M Walczak, M Oujja, et al. Femtosecond laser deposition of TiO2 by laser induced forward transfer. *Thin Solid Films*, 518(19):5525–5529, 2010.

[106] Robert R Schaller. Moore's law: past, present and future. *IEEE spectrum*, 34(6):52–59, 1997.

[107] Hans F Schwaiger. An implicit corrected SPH formulation for thermal diffusion with linear free surface boundary conditions. *International Journal for Numerical Methods in Engineering*, 75(6):647–671, 2008.

[108] Songdong Shao and Edmond YM Lo. Incompressible SPH method for simulating Newtonian and non-Newtonian flows with a free surface. *Advances in water resources*, 26(7):787–800, 2003.

[109] Yuhui Shi et al. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 81–86. IEEE, 2001.

[110] Antonio Souto-Iglesias, Fabricio Macià, Leo M González, and Jose L Cercos-Pita. On the consistency of MPS. *Computer Physics Communications*, 184(3):732–745, 2013.

[111] Anthony James Merrill Spencer. *Continuum mechanics*. Courier Corporation, 2004.

[112] Robert W Style and Stephen SL Peppin. Crust formation in drying colloidal suspensions. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, page rspa20100039. The Royal Society, 2010.

[113] Tetsuya Takahashi, Yoshinori Dobashi, Issei Fujishiro, Tomoyuki Nishita, and Ming C Lin. Implicit formulation for SPH-based viscous fluids. In *Computer Graphics Forum*, volume 34, pages 493–502. Wiley Online Library, 2015.

[114] Alexandre Tartakovsky and Paul Meakin. Modeling of surface tension and contact angles with smoothed particle hydrodynamics. *Physical Review E*, 72(2):026301, 2005.

[115] TE Tay. Finite element analysis of thermoelastic coupling in composites. *Computers & structures*, 43(1):107–112, 1992.

[116] Halil L Tekinalp, Vlastimil Kunc, Gregorio M Velez-Garcia, et al. Highly oriented carbon fiber–polymer composites via additive manufacturing. *Composites Science and Technology*, 105:144–150, 2014.

[117] Stephen Timoshenko. Analysis of bi-metal thermostats. *Josa*, 11(3):233–255, 1925.

[118] Nathaniel Trask, Martin Maxey, Kyungjoo Kim, et al. A scalable consistent second-order SPH solver for unsteady low Reynolds number flows. *Computer Methods in Applied Mechanics and Engineering*, 289:155–178, 2015.

[119] Nathaniel Trask, Martin Maxey, Kyungjoo Kim, et al. A scalable consistent second-order SPH solver for unsteady low Reynolds number flows. *Computer Methods in Applied Mechanics and Engineering*, 289:155–178, 2015.

[120] Y Vidal, J Bonet, and Antonio Huerta. Stabilized updated Lagrangian corrected SPH for explicit dynamic problems. *International journal for numerical methods in engineering*, 69(13):2687–2710, 2007.

[121] Rade Vignjevic, Juan R Reveles, and James Campbell. SPH in a total Lagrangian formalism. *CMC-Tech Science Press-*, 4(3):181, 2006.

[122] Marcel Weiler, Dan Koschier, Magnus Brand, and Jan Bender. A physically consistent implicit viscosity solver for SPH fluids. In *Computer Graphics Forum*, volume 37, pages 145–155. Wiley Online Library, 2018.

[123] David A Willis and Vicentiu Grosu. Microdroplet deposition by laser-induced forward transfer. *Applied Physics Letters*, 86(24):244103, 2005.

[124] Lingling Wu, Bo Li, and Ji Zhou. Isotropic negative thermal expansion metamaterials. *ACS applied materials & interfaces*, 8(27):17721–17727, 2016.

[125] Baojun Xie, Robert L Parkhill, William L Warren, and James E Smay. Direct writing of three-dimensional polymer scaffolds using colloidal gels. *Advanced Functional Materials*, 16(13):1685–1693, 2006.

[126] Rui Xu, Peter Stansby, and Dominique Laurence. Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach. *Journal of Computational Physics*, 228(18):6703–6725, 2009.

[127] Xiaoyang Xu, Jie Ouyang, Binxin Yang, and Zhijun Liu. SPH simulations of three-dimensional non-Newtonian free surface flows. *Computer Methods in Applied Mechanics and Engineering*, 256:101–116, 2013.

[128] Xiaoyang Xu and Peng Yu. A technique to remove the tensile instability in weakly compressible SPH. *Computational Mechanics*, pages 1–28, 2018.

[129] Fuqian Yang and JCM Li. Diffusion-induced beam bending in hydrogen sensors. *Journal of Applied Physics*, 93(11):9304–9309, 2003.

[130] Hyunwoo Yuk and Xuanhe Zhao. A new 3D printing strategy by harnessing deformation, instability, and fracture of viscoelastic inks. *Advanced Materials*, 30(6):1704028, 2018.

[131] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Olgierd Cecil Zienkiewicz, and Robert Lee Taylor. *The finite element method*, volume 3. McGraw-hill London, 1977.

[132] Tarek Zohdi. Particle collision and adhesion under the influence of near-fields. *Journal of Mechanics of Materials and Structures*, 2(6):1011–1018, 2007.

[133] TI Zohdi. On the dynamics of charged electromagnetic particulate jets. *Archives of Computational Methods in Engineering*, 17(2):109–135, 2010.