Made available through Montana State University's ScholarWorks

Efficient extraction of atomization processes from high-fidelity simulations

Brendan Christensen, Mark Owkes

© This manuscript version is made available under the CC-BY-NC-ND 4.0 license https://creativecommons.org/licenses/by-nc-nd/4.0/

Efficient Extraction of Atomization Processes from High-Fidelity Simulations

Brendan Christensen*, Mark Owkes

Department of Mechanical and Industrial Engineering, Montana State University, Bozeman, MT, 59717-3800, USA

Abstract

Understanding the process of primary and secondary atomization in liquid jets is crucial in describing spray distribution and droplet geometry for industrial applications and is essential in the development of physics-based low-fidelity atomization models that can quickly predict these sprays. Significant advances in numerical modelling and computational resources allows research groups to conduct detailed numerical simulations and accurately predict the physics of atomization. These simulations can produce hundreds of terabytes of data. The substantial size of these data sets limits researchers' ability to analyze them. Consequently, the process of a coherent liquid core breaking into droplets has not been analyzed in simulation results even though a complete description of the jet dynamics exists. The present work applies a droplet physics extraction technique to high-fidelity simulations to track breakup events and data associated with the local flow. The data on the atomization process are stored in a Neo4j graphical database providing an easily accessible format. Results provide a robust, quantitative description of the process of atomization and the details on the local flow field will be useful in the development of low-fidelity atomization models.

1. Introduction

Atomizing sprays have a wide range of applications in industrial and environmental fields (e.g. fuel injection [1], agricultural sprays [2], pharmaceutical sprays [3], and air-sea interaction [4]). Consequently, the body of research on this topic is vast and spans multiple centuries [5, 6]. Researchers have made significant progress in understanding many aspects of the atomization process, however, the mechanisms that drive instabilities in coherent liquid structures and ultimately cause breakup are still widely unresolved. This is, in part, due to limitations in experimental data collection from atomizing flows. Visual analysis of atomizing jets is difficult because of the scale and speed at which these systems develop. Furthermore, the development of droplets creates an opaque cloud, which blocks view of the liquid core and severely limits the ability to study primary instability and breakup mechanics. Recent advances in ballistic [7, 8, 9] and x-ray [10, 11, 12] imaging of atomizing flows have attempted to remedy visualization issues. These methods provide snapshots of the atomization process and lack important temporal information. Because of challenges with current experimental methodology, numerical simulations, and particularly high-fidelity simulations that resolve the relevant time and length scales, have been developed and provide an alternative method to study the physics of atomization.

Advances in numerical methods and computational efficiency within the past decade have greatly expanded the capabilities of numerical simulations. Researchers are now able to simulate multiphase flows with exceptional accuracy. Numerically simulating multiple phases is not a trivial process, however. These systems involve a wide range of topological scales in fluid and turbulent structures in addition to jumps in physical qualities at the phase interfaces. Significant effort has been and is still invested in improving numerical simulation methodology to more accurately and efficiently simulate atomizing systems. Accordingly, most high-fidelity simulation studies on atomization to date are focused on the development of numerical methods and testing against experimental results rather than using the simulations to produce novel atomization data. But, with the pending maturity of the field, several research groups have been able to gather some insightful statistics from high-fidelity atomization simulations.

*Corresponding Author

Email address: brendan.christensen@montana.edu (Brendan Christensen)

February 17, 2023

Following Gorokhovski and Herrmann's review of atomization simulations [13], the field developed rapidly. Among the first high-fidelity simulations of atomizing liquid were Shinjo and Umemura [14] and Desjardins and Pitsch [15] who simulated round and planar liquid jets, respectively. These works provided a foundation for future simulation studies by identifying numerous breakup mechanisms and providing qualitative descriptions of previously unclassified processes. Following these pioneering studies, various other works were published which utilized high-fidelity simulations to expand upon existing descriptions and identify novel breakup mechanisms. Several of these groups incorporated vorticity measurements into their post-processing to strengthen descriptions of breakup by integrating this additional dimension to their analysis [16, 17, 18, 19, 20, 21, 22]. Additionally, many researchers focus their efforts on comparing linear-stability theory [23, 24] and high-fidelity simulations [16, 25, 26, 27, 28]. These studies process visualizations of the atomizing jet to analyze and track wavelengths as they become more unstable, ultimately leading to breakup. Several recent studies have obtained turbulence statistics from throughout the simulation using constrained spatial and/or temporal domains to limit data-processing requirements [29, 30, 31].

The numerical studies mentioned above have greatly improved understanding of the atomization process. They developed robust qualitative descriptions of breakup events and utilized bulk quantities to elucidate some of the underlying physics of liquid breakup. However, despite the recent advances in atomization simulations, there remain significant gaps in our understanding of the process. This is primarily due to a lack of available methodology for obtaining many relevant statistics from high-fidelity simulations. In particular, current research methods have not been able to extract local, temporally continuous, quantitative data on liquid breakup. The data include information on droplet shape and size characteristics, local flow field data in both the liquid and gas phases, and information on how these values evolve temporally and spatially throughout a spray system. An appreciable challenge obtaining this data is derived from the massive size of resultant data sets; these can be hundreds of terabytes or even larger. Parsing such data sets for relevant information is not practical and is often impossible.

While numerical simulations are becoming more efficient, it remains computationally expensive and requires the use of high-performance computing. Reduced-order atomization models aim to provide a viable alternative to high-fidelity simulations at a fraction of the computational cost. Some prominent examples of atomization models include the Taylor analogy breakup (TAB) model [32], the Pilch-Erdman (PE) model [33], the Kelvin-Helmholtz-Rayleigh-Taylor (KH-RT) model [23, 34, 35], the Eulerian-Lagrangian spray and atomization (ELSA) model [36], the bag type breakup (BTB) model [37], the multi-mode breakup (MMB) model [38], and the modified Taylor analogy breakup (MTAB) model [39]. These atomization models try to predict how and when a liquid structure breaks apart without fully resolving the physics. As mentioned above, however, there is little information on the mechanisms that control breakup events and even less information on the process of atomization. Model developers would greatly benefit from local data sampled from breakup events throughout the simulation.

The present work aims to address the above obstacles through the improvement of a framework, first introduced in Rubel and Owkes [40], which extracts relevant data from breakup and coalescence events in atomization simulations. Through the use of this tool, many researchers would gain access to extensive and easily accessible data sets from atomization simulations. Using this data, they could build on previous work, introduce high-quality statistical analyses, and significantly advance the field. Additionally, the structure geometry and flow field can be assessed to quantify the details of the breakup conditions. This new information will advance our understanding of the atomization process and provide statistics with which to inform and improve reduced-order atomization models.

2. Methods

The droplet physics extraction tool, originally proposed in Rubel and Owkes [40], is improved and applied in this work. The tool is implemented into high-fidelity atomization simulations to gather data from discrete events within these simulations coinciding to liquid breakup and coalescence. This section will outline the methodology for this process in detail.

2.1. Computational Platform

The proposed extraction tool can be applied to any high-fidelity Navier-Stokes solver, given the solver incorporates two identification numbers used to determine breakup and coalescence events, which will be defined in the

following sections. This work employs the NGA computational platform [41, 42, 43, 44]. NGA solves the two-phase formulations of the Navier-Stokes mass and momentum conservation equations, defined as

$$\frac{\partial \rho_\phi}{\partial t} + \nabla \cdot (\rho_\phi \boldsymbol{u}_\phi) = 0 \tag{1}$$

and

$$\frac{\partial \rho_\phi \boldsymbol{u}_\phi}{\partial t} + \nabla \cdot (\rho_\phi \boldsymbol{u}_\phi \otimes \boldsymbol{u}_\phi) = -\nabla p_\phi + \nabla \cdot (\mu_\phi [\nabla \boldsymbol{u}_\phi + \nabla \boldsymbol{u}_\phi^T]) + \rho_\phi \boldsymbol{g} \tag{2}$$

respectively. In these equations, ρ_ϕ is density, $\boldsymbol{u}_\phi$ is the velocity vector, p_ϕ is the pressure, μ_ϕ is the dynamic viscosity, t is time, $\boldsymbol{g}$ is the gravitational acceleration vector and ϕ is the phase indicator with either $\phi = g$ (gas) or $\phi = l$ (liquid). These equations are solved on a staggered Cartesian mesh, in which scalar values, such as pressure, are stored at cell centers and velocities at the cell faces. Time is discretized with an iterative Crank-Nicolson formulation and a semi-implicit correction is applied on each sub-iteration [45]. Away from the phase interface, mass, momentum, and any other scalars are transported with conservative, high-order finite difference operators [41]. Near the interface, an un-split semi-Lagrangian geometric volume-of-fluid (VOF) method is utilized to ensure conservative transport of mass and momentum [43, 44]. The interface is reconstructed using a piece-wise linear interface reconstruction (PLIC) [46] with interface normal vectors computed using the efficient least-squares VOF interface reconstruction algorithm (ELVIRA) [47]. The pressure Poisson equation is solved utilizing the ghost fluid method [48] and a black box solver [49]. Interface curvature is computed with the adjustable curvature evaluation scale (ACES) method [50]. NGA is fully parallelized with message passing interface (MPI) and scales well to tens-of-thousands of cores [15].

2.2. Breakup and Coalescence Event Identification

The coalescence and split identification processes operate through the implementation of two identification numbers, which are integers unique to every independent liquid structure within a simulation. These values are referred to as the structure identification number $\mathcal{S}$ and the liquid identification number $\mathcal{L}$. Every liquid structure in the simulation domain is tagged with each of these values. The process in which $\mathcal{S}$ and $\mathcal{L}$ are assigned, transported, and reassigned is the basis for identifying liquid splitting and merging events within a simulation.

2.2.1. Identification Numbers

Liquid identification numbers ($\mathcal{L}$) are transported with the liquid and provide a time history of where liquid moves. Therefore, the $\mathcal{L}$ values are not reassigned at every timestep. They are persistent through time and are only reassigned after a split event creates a new liquid structure or a coalescence event destroys a liquid structure. $\mathcal{L}$ values are transported with the liquid they occupy, so $\mathcal{L}$ transport must be tied into the simulation's liquid transport scheme. In our implementation, $\mathcal{L}$ fluxes are calculated in the same manner as the semi-Lagrangian liquid fluxes in NGA's geometric VOF scheme [43, 44]. A list ($\mathcal{L}_{\text{Flux}}$) of all the $\mathcal{L}$ that flux into a cell during a timestep is constructed. This is done by first constructing lists at each cell face with signed $\mathcal{L}$ values indicating the direction. After lists associated with each cell face are computed, they are compiled, to create a list of all $\mathcal{L}$'s within a cell. These lists are looped through in a later section of the code to identify coalescence. Fig. 1 displays a visual representation of this process. See Appendix A for the $\mathcal{L}$ transport and coalescence identification algorithm.

Structure identification numbers ($\mathcal{S}$) are assigned at every timestep following liquid transport. The values of $\mathcal{S}$ are not consequential, provided that every structure is assigned a unique value. Independent structures are identified and tagged using a band-growth algorithm first described by Herrmann [51]. In the present application, the algorithm operates as follows. To begin, $\mathcal{S}$ is assigned on the entire domain to be zero. Then, the domain is looped through to find the first untagged liquid node, meaning the liquid volume fraction is greater than zero and $\mathcal{S} = 0$. This point is then tagged with an $\mathcal{S}$ and the cells around that node are looped through until the entire continuous structure is identified and assigned the $\mathcal{S}$ value. A logical statement determines whether an adjacent cell will be assigned the same $\mathcal{S}$ value as the present cell. The statement requires that: 1) there must be liquid in that cell, 2) it must be untagged ($\mathcal{S} = 0$), and 3) the $\mathcal{L}$ of the adjacent cell must equal that of the current cell. The third statement ensures that liquid structures do not re-merge immediately following breakup as structures advance and naturally inhabit adjacent cells. See Section 2.2.4 for a more detailed description of these criteria. Herrmann's work also described the parallelization of this process, which is utilized in the extraction tool. This requires special consideration be given to communicating $\mathcal{S}$ between processors. Many other methods exist to assign the structure identification number. This type of tagging problem is known as Connected Component Labeling, e.g. [52, 53].

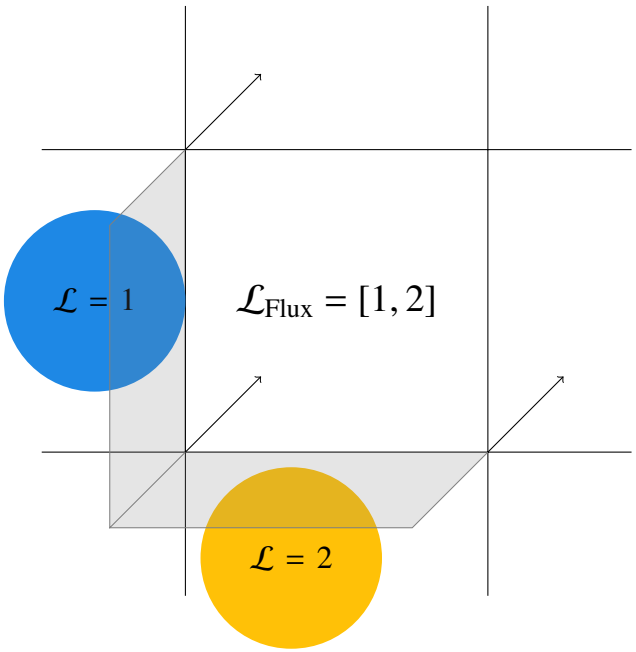

Figure 1: An example of two droplets fluxing into the same cell. Gray regions are semi-lagrangian flux regions [43], which carry $\mathcal{L}$ into the center cell due to the velocity shown with vectors. The $\mathcal{L}$'s of the two droplets are stored in the list $\mathcal{L}_{\text{Flux}}$.

2.2.2. *Breakup Identification*

Liquid structure splitting or breakup is identified when two different structures (two unique $\mathcal{S}$'s) have the same $\mathcal{L}$. This $\mathcal{L}$ is persistent through time, indicating that these two droplets were part of the same structure at the previous timestep. The logical statement, described in Section 2.2.1, loops over adjacent cells to connect liquid structures. Two structures become independent of each other when they are not in adjacent cells i.e., there is a full cell separating them. This is when the $\mathcal{S}$ assignment portion of the code assigns two structures with unique $\mathcal{S}$'s. The two structures have the same $\mathcal{L}$, indicating that a breakup occurred. When a breakup is identified, data are extracted from the event, and a new $\mathcal{L}$ is assigned to the smaller structure(s) created by the breakup event. See $t = 0$ and $t = 1$ in Fig 2 for a visual representation of this process.

2.2.3. *Coalescence Identification*

Coalescence of liquid structures is identified using the $\mathcal{L}_{flux}$ lists mentioned in Section 2.2.1. More than one unique $\mathcal{L}$ in a cell indicates that two structures are merging together. When coalescence is identified, data are extracted from the event, and the new merged structure takes the smaller $\mathcal{L}$ of the two parent structures.

2.2.4. *Addressing Fictitious Events*

An issue from the previous work, described in Rubel and Owkes [40] was the occurrence of numerous fictitious merge and split events. These occurred when the tool identified a structure that broke into two structures and then the two structures coalesced. The algorithm identified this process and repeated many times, when in fact the liquid structures never re-merged following breakup. The error stemmed from the original $\mathcal{S}$ assignment algorithm. Droplets in adjacent cells were automatically treated as the same structure, which resulted in recently split liquid structures being incorporated back into their parent droplet when they were in neighboring cells. To avoid an accumulation of non-physical results, the authors opted to run the split identification portion of the tool only every 10 timesteps. The present work changed the parameters that define a coalescence event to prevent droplets from merging together immediately after breakup. This allows the tool to run every timestep, bolstering it's accuracy and usefulness. See Fig. 3 for a visualization of this issue and the result following the updates in this work.

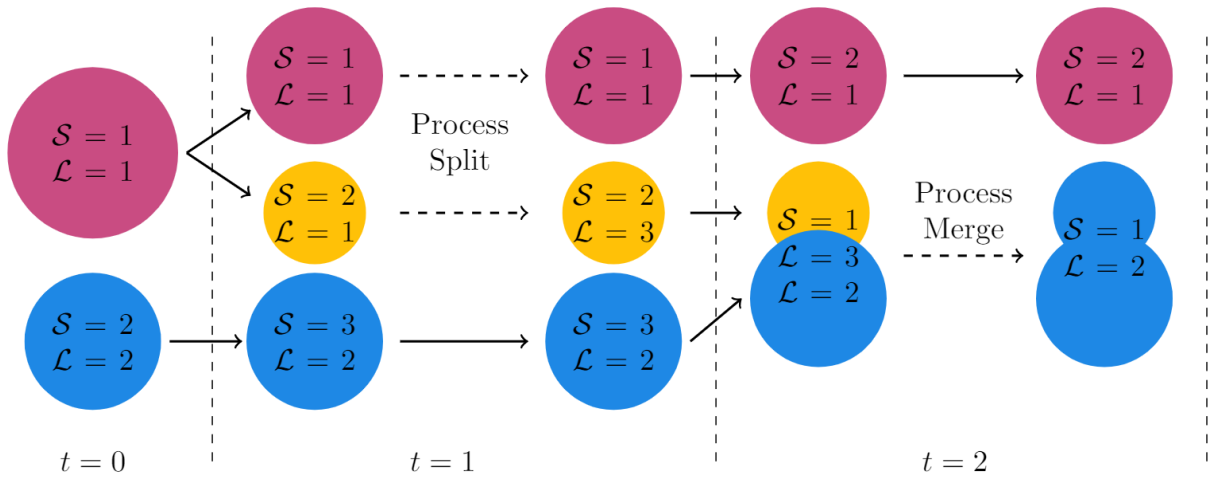

Figure 2: An example of two droplets undergoing breakup and coalescence. Two droplets start on the left, the top droplet breaks into two droplets. The two droplets are identified and the split event is processed. The bottom two droplets coalesce, the merge is identified and the merge event is processed. Note that the $\mathcal{S}$ can vary between timesteps, but the $\mathcal{L}$ is persistent until a split or merge causes the value to change.

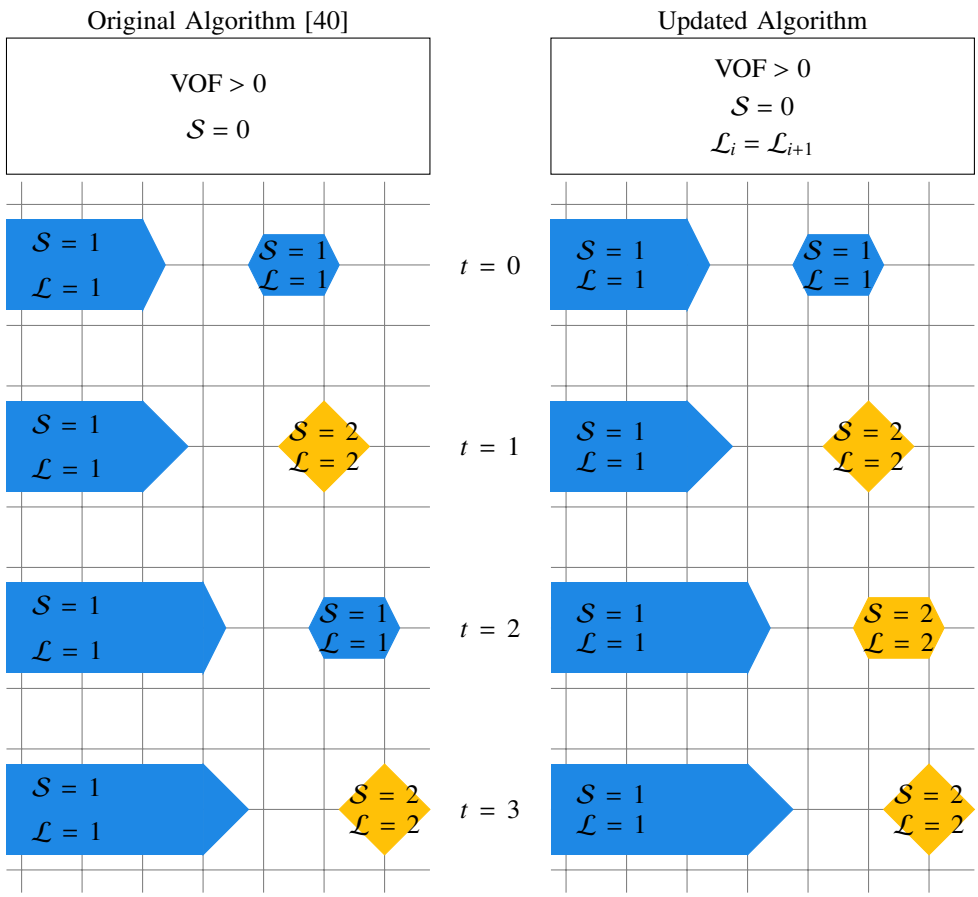

Figure 3: The diagram on the left shows the issue with fictitious merging and splitting events in the original algorithm [40]. The diagram to the right shows how the split and merge identifier works in the present work. Notice that at $t = 2$ in the updated code the structures do not merge back together. This is because the criterion for coalescence was changed to require the structures to exist within the same cell, not adjacent cells.

As stated above, in Section 2.2.3, coalescence now occurs when two structures flux into the same cell. This methodology requires that special care be given to the $\mathcal{S}$ assignment algorithm to ensure that separate structures in adjacent cells are not combined. To address this, the $\mathcal{L}$ of droplets was incorporated into the band-growth algorithm. Fig. 3 illustrates this updated $\mathcal{S}$ assignment process. At $t = 0$ a droplet is breaking up so that there is liquid in adjacent cells. Since a split has not been identified yet, the $\mathcal{L}$ of both cells is the same, resulting in a continuous structure. Then, at $t = 1$ the structure on the right moves, leaving a full cell between it and the other structure. The cells adjacent to the blue structure are void of liquid, so the $\mathcal{S}$ value is only assigned to that structure. Since the droplet on the right becomes independent, it is assigned a new $\mathcal{S}$, a split is processed and it is also assigned a new $\mathcal{L}$. Finally, at $t = 2$, previously, the two droplets would have merged back together as they occupy neighboring cells. But the structures remain separate with the new algorithm, because the $\mathcal{L}$ of the adjacent droplet does not equal the $\mathcal{L}$ of the blue droplet, they remain separate structures and will not coalesce unless they enter the same cell.

2.3. Data Extraction

Following identification of breakup or coalescence in the simulation, data are extracted from that location and time. This is intended to provide access to the local conditions of these events. The original work extracted data from the timestep immediately following breakup. We expanded the temporal sampling range, allowing the tool to extract data from the timestep immediately preceding breakup in addition to the timestep following breakup. This improvement is intended to provide insight into the conditions which lead to breakup and to the types of structures which the conditions produce. To highlight some potential uses of the tool, the present work extracted the locations of the events, the gas and liquid velocities, droplet volume, and droplet shape information.

For the purposes of this study, liquid and gas velocities are calculated as volume averaged velocities,

$$\boldsymbol{U}_{\text{liquid}} = \frac{\sum_{i=1}^{N_s} \boldsymbol{u}_i \mathcal{V}_{\text{cell},i}\, \alpha_i}{\sum_{i=1}^{N_s} \mathcal{V}_{\text{cell},i}\, \alpha_i} \tag{3}$$

$$\boldsymbol{U}_{\text{gas}} = \frac{\sum_{i=1}^{N_s} \boldsymbol{u}_i \mathcal{V}_{\text{cell},i}\, (1-\alpha_i)}{\sum_{i=1}^{N_s} \mathcal{V}_{\text{cell},i}\, (1-\alpha_i)} \tag{4}$$

respectively. These values are calculated by considering all cells that contain a liquid structure and all cells adjacent to those (N_s total cells). In the expressions, $\boldsymbol{u}_i$ is the velocity vector, $\mathcal{V}_{\text{cell},i}$ is the cell volume, and α_i is the liquid volume fraction in the ith cell. This method for calculating the velocities is preliminary and does not fully capture the local flow field dynamics. Future work focused on extracting topological data from the flow field will work to better quantify these values.

2.4. Graphical Database

The primary goal of the tool is to make data from atomization simulations more accessible. Special consideration was given to ensuring that extracted data was stored in an efficient and easily queriable format. We opted to store the atomization data in a Neo4j graphical database. Graph databases are commonly used in the corporate sector for companies to create connections between users and products through paths, which can reveal patterns in group dynamics and buying trends. The same principles can be applied to atomization simulations through the construction of paths which connect droplets through breakup and coalescence events. Storing atomization data in a graph database format allows researchers to analyze the evolution of liquid as it moves from the liquid core to small droplets in an atomizing flow. In this work, the Neo4j database is used to store the atomization data. Data from the simulation in this work is initially written to a CSV file. Then, rather than simply storing the data as a disconnected list of events, the CSV file is uploaded to Neo4j. The graph format presents an easy way to connect related events and develop a droplet ancestry. Additionally, storing data in graph databases provide a unique way to visualize atomization data and a novel method for studying the atomization process.

Fig. 4 displays how data are stored in a Neo4j database. This image represents some of the data from the simulation described in Section 3. The nodes are droplets produced from breakup events. The breakup events are represented by the lines connecting the nodes. The node colors represent the number of times they have broken up. The liquid core is in the center of the image and the red droplets represent the sixth breakup events. This image is made up of all the droplets which broke up 6 times during the first 2.3 μs of the simulation, and all the intermediate droplets between

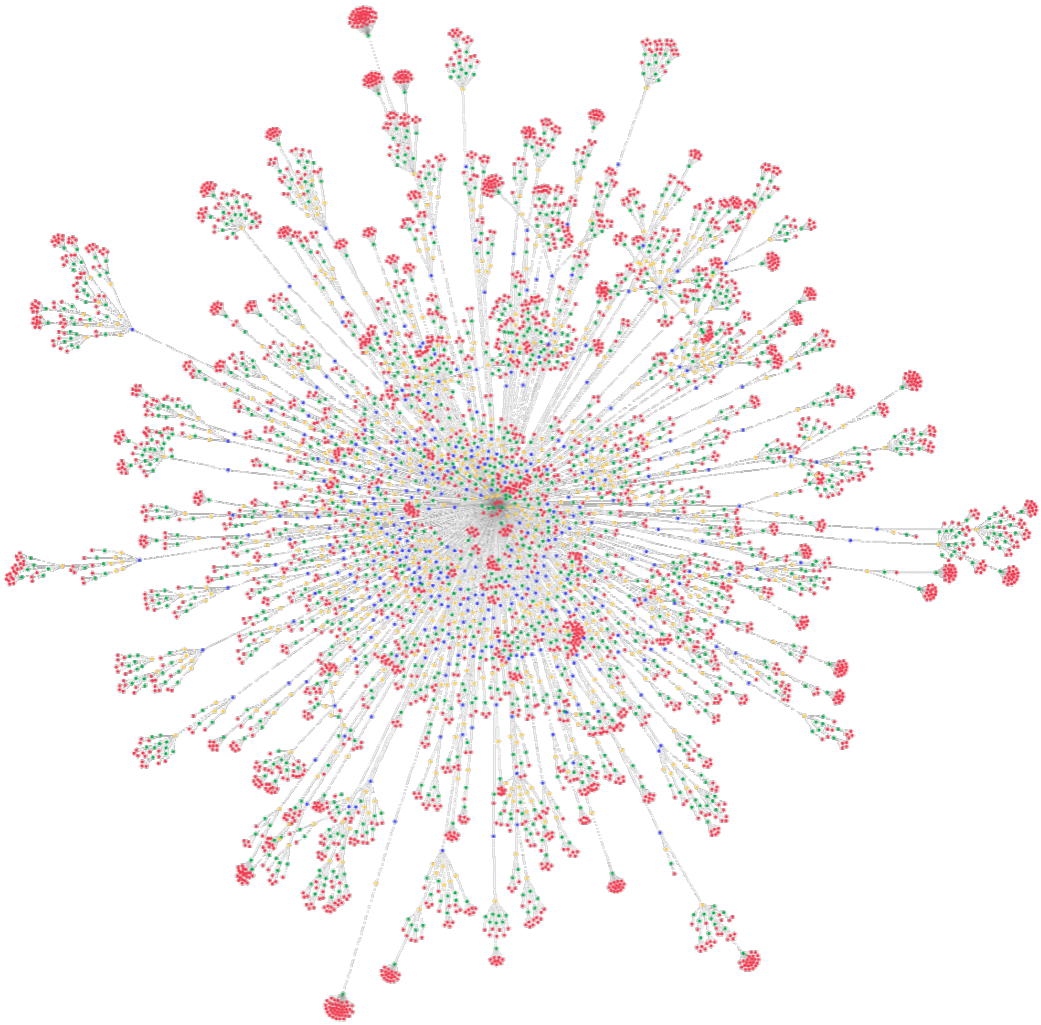

Figure 4: Example of data points in Neo4j graphical database. Red nodes represent some droplets which broke up 4 times throughout the simulation. Other colors represent intermediate breakups (blue = primary, yellow = secondary, green = tertiary). Lines connect related droplets, i.e., droplets which split from each other. The liquid core is present at the center of the image. Each node contains relevant statistics from the breakup event.

the coherent liquid core and these sixth breakup droplets. Within each node is stored the extracted data from the point and time in the simulation in which that droplet broke up and became independent. So, this database is saturated with data, but organized in a logical and accessible format.

The graph database can be efficiently edited, reorganized, and parsed with the Cypher Query Language. It is syntactically similar to SQL, but designed to specifically query nodes and relationships and the paths formed with these components. As mentioned above, data is imported to Neo4j via a CSV file, organized in such a way that each row is an independent liquid breakup or coalescence event. These rows are imported into Neo4j as droplet nodes and breakup and coalescence relationships are created. See Appendix B for a detailed description of the process to import data into Neo4j. Following data import, Cypher can be used to further organize the data and/or query the data to analyze the atomization process. Below are some examples of the capabilities of Cypher in the present application.

```
// Rename the liquid core
MATCH(n:droplet)      // Find nodes with the "droplet" label
WHERE n.Event='None' // Node with Event = none is the liquid core
CALL apoc.refactor.rename.label("droplet","core",n)  // Rename node
YIELD committedOperations
RETURN committedOperations

// Identify and rename primary droplets
MATCH (n:droplet)
WHERE n.OldLID = 1    // Find droplets which broke off of the liquid core
WITH collect(n) as p // Compile a list of nodes matching criteria
CALL apoc.refactor.rename.label("droplet","primary",p)
YIELD committedOperations
RETURN committedOperations

// Rename secondary,tertiary, etc. (repeat until no "droplet" nodes remain)
MATCH (n:droplet),(d:primary)
WHERE n.OldLID = d.NewLID // Find "droplet" nodes that split from "primary"
WITH collect(n) as p
CALL apoc.refactor.rename.label("droplet","secondary",p)
YIELD committedOperations
RETURN committedOperations
```

This portion of code first identifies the core by finding the only node which has an event not equal to either breakup or coalescence. Then, it renames this droplet "core". Note: the "Awesome Procedures on Cypher" (APOC) library must be enabled to access the renaming features. Next, primary droplets are identified by finding all the droplet nodes which previously had a $\mathcal{L}$ (LID) equal to one (the liquid core has $\mathcal{L} = 1$). These droplets are renamed "primary". After primary droplets are identified, secondary droplets can be identified using a similar process. The algorithm loops through all droplets and looks for nodes which previously had $\mathcal{L}$ equal to the $\mathcal{L}$ of a primary droplet. This process can then be repeated for each subsequent breakup event until no more "droplet" nodes remain. This is very useful in order to analyze the evolution of droplets as they break up further. This will be analyzed in more detail in the Results section.

```
// Collect all breakup paths of droplets which break up six times
MATCH (n:sixth), (c:core), p = shortestPath((c)-[:Split*]->(n))
RETURN [d in nodes(p)| d.Volume]
```

The "shortestPath" function is used to extract the paths between specific nodes. In the above example, the shortest path from the liquid core to droplets that broke up six times is queried. Then, the volume of each droplet in that path is returned. This feature provides a simple method to extract useful relational data from the breakup process.

In addition to the Neo4j graphical database system and the Cypher Query Language, we utilized Python for analysis of results. Both programs have benefits and drawbacks, which led to the utilization of both. Neo4j allowed us to build paths very easily between droplets which break up from each other or coalesce together. From this

information, we can easily output a CSV which displays how statistics evolve with breakup events or coalescence events. Following the output of these CSV files, it is simple to analyze the data using Python's data science libraries. Additionally, multiple options exist to query or create a graph databases through Python. These include the official Neo4j driver for Python and the py2neo Python library. These prove to be very powerful tools, because they allow users to combine python loops, if statements, and data science libraries with the unique organizational system of Neo4j. The py2neo library was used in the present work to build the time-series plots in section 3.6.2. See appendix **??** for an example python script querying a Neo4j database.

3. Application of Extraction Tool on Diesel Jet

3.1. Simulation Setup

To test the utility of the updated tool we ran a simulation inspired by a diesel injector. The simulation is the same as that described in Rubel and Owkes [40] and consists of a turbulent liquid injection into a quiescent air. The turbulence was computed from a preliminary turbulent pipe-flow simulation and the velocity field was stored and used as the inlet boundary condition for the liquid jet. Table 1 provides the parameters of the simulated jet. The simulation was run on 160 processors on the Hyalite High-Performance Computing Cluster at Montana State University. The dimensions of the computational mesh are $N_x \times N_y \times N_z = 1024 \times 128 \times 128$. Note that the resolution of this simulation is not fine enough to accurately capture small scale interface features or the smallest scales of turbulence. However, this simulation is sufficient to demonstrate the efficacy of the proposed tool. Future work will focus on applying the tool to a higher resolution simulation to more accurately extract information on physical phenomena.

Number	Definition	Value
Bulk Reynolds number	$\rho_l U_{\text{jet}} D_{\text{jet}} / \mu_l$	25,000
Bulk Weber number	$\rho_l U_{\text{jet}}^2 D_{\text{jet}} / \sigma_l$	10,000
Density ratio	ρ_l / ρ_g	40
Viscosity ratio	μ_l / μ_g	1.67
Domain length	L_x / D_{jet}	60
Domain widths	$L_{y,z} / D_{\text{jet}}$	7.5
Cells across diameter	$D_{\text{jet}} / \Delta x$	17.06
CFL number	$\lvert u \rvert_{\max} \Delta t / \Delta x$	0.4

Table 1: Non-dimensional parameters used in the diesel jet simulation.

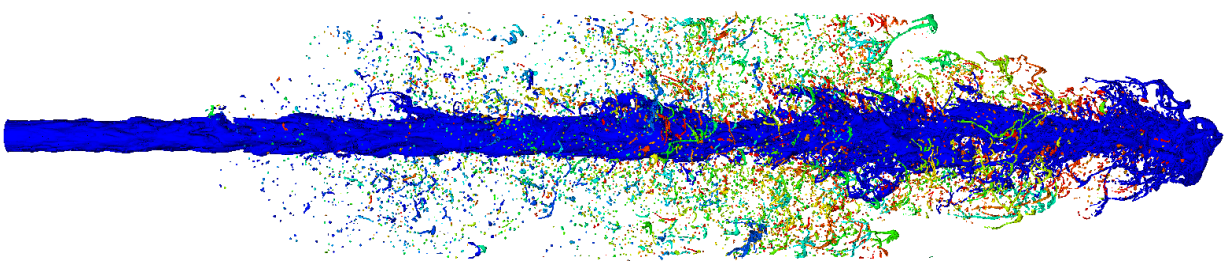

Figure 5: Rendering of the liquid jet run in simulation. $\mathcal{L}$ values at final timestep represented by colors. Rendered using VisIt [54].

3.2. Addressing Fictitious Events Confirmation

Section 2.2.4 discusses an update in the coalescence and breakup identification algorithm, which is intended to prevent the occurrence of fictitious events. In the original work [40], the fictitious events coincided with every breakup within the simulation, which forced the authors to institute a delay in the breakup identification portion of the code. This meant they only identified breakup events every 10 timesteps. In the present work, breakup identification is

performed on every timestep. It is reasonable in complex flow systems that droplets may breakup and then re-merge with their parent droplet, so the algorithm is not intended to prevent this phenomena from occurring altogether.

To confirm that the updated method is producing reasonable results, we utilized Neo4j to identify all the droplets which split and re-merged with their parent droplet. Then, the difference in time between the coalescence event and the initial breakup event was calculated. 17.93% of all droplets re-merged with their parent droplet, with the majority of those being primary droplets coalescing with the liquid core. This is a well documented mechanism [14] and expected in a round jet injected into quiescent gas. Analyzing only secondary droplets, we found that 9.31% of secondary droplets re-merged with their parent droplets. The fictitious events were documented to occur within one or two timesteps of breakup, but in the present simulation, the average time between coalescence and breakup is 61.7 μs which translates to about 300 timesteps. Given these statistics, we conclude that the updates made to the tool in the present work remedied the fictitious breakup and coalescence issue addressed in the original work and the identified events in the present work are consistent with physical events in the spray.

3.3. Secondary Atomization Analysis

Common descriptions of breakup regimes within an atomizing system involve discussion of primary atomization, i.e. droplets splitting from the liquid core, and secondary atomization, i.e., all further breakup. As seen in Fig. 6, many droplets are created after multiple breakup events. Further analysis shows that 53.6% of the total number of droplets formed in the test diesel jet broke up three or more times, with 17.0% percent breaking up at least five times. At later times in the simulation, this percentage becomes more pronounced. The yellow curve with triangle markers in Fig. 6 is the distribution of droplets at the final timestep. 67.7% of drops present at this timestep broke up 3 or more times. Thus, a majority of the breakup in the simulation occurred after secondary breakup. This indicates that the final spray field is heavily dependent on mechanisms within these later atomization regimes, which are all generally grouped in secondary atomization.

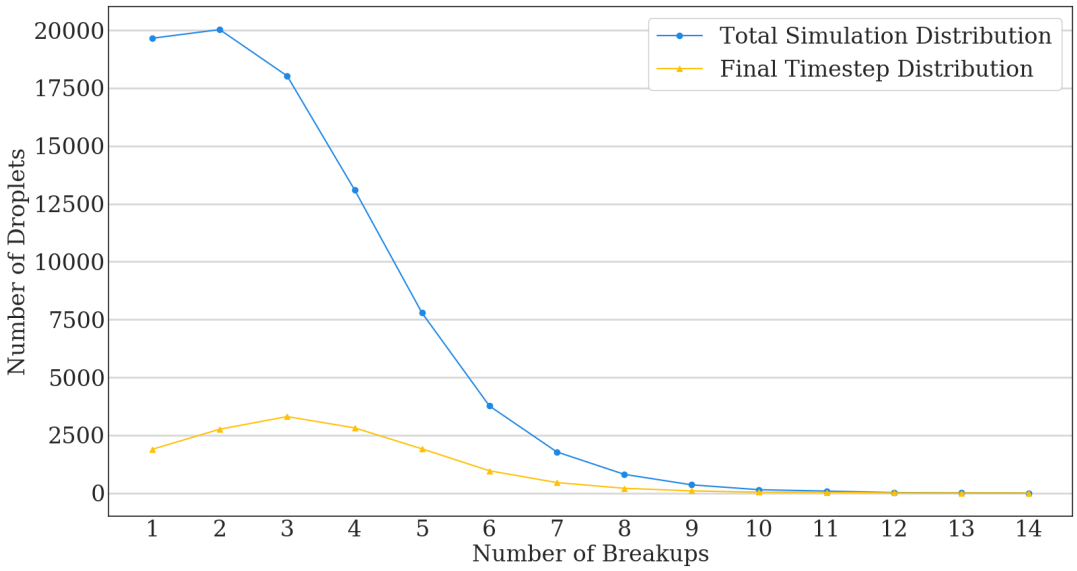

Figure 6: The total number of droplets associated with each breakup stage from all breakup events within the diesel jet simulation.

3.4. Coalescence Analysis

A trend extracted from the test case simulation is the prevalence of coalescence events. This matches findings in Prakash et al. [55], who identified a significant number of coalescence events within their simulation. The present

simulation produced 76,608 breakup events and 57,908 coalescence events. Table 2 displays the average number of coalescence events for a given number of breakup events. These data were calculated using a path finding algorithm in Neo4j. The algorithm finds the shortest path of breakup events from droplets at the given breakup stage back to the liquid core and counts the number of coalescence events along that path. The average number of coalescence events shown in the table is the average amongst all droplets that have broken up the associated number of times. The data in Table 2 indicates that the proportion of coalescence events increases as the jet evolves and more breakup occurs. This makes intuitive sense, because as the jet evolves, a high droplet density cloud develops, which leads to more droplet collisions.

Breakup Stage	Average Number of Coalescence Events	Ratio of Coalescence to Breakup
Secondary	1.077	0.5385
Third	1.899	0.6330
Fourth	2.898	0.7245
Fifth	3.974	0.7948
Sixth	5.241	0.8735
Seventh	6.823	0.9747
Eighth	8.276	1.035

Table 2: The average number of coalescence events between primary breakup and the breakup stage is listed in the second column. The third column displays the ratio of coalescence events to breakup events along breakup paths.

3.5. Local Flow Field Statistics

Another use for this tool is to better understand how the local flow field is affecting the atomization process. This is information that is exceedingly difficult, if not impossible, to obtain through experimental methods. Moreover, gathering a sufficient sample of statistics on these local flow mechanisms has remained nonviable for those conducting numerical simulations. This work extracted preliminary data on the local flow field surrounding breakup events and calculated the resultant local Weber number, defined as $\mathrm{We}_{\mathrm{local}} = \rho_g U_s^2 L/\sigma$, where ρ_g is the gas velocity, U_s is the slip velocity (defined as $U_s = |U_{\mathrm{liquid}} - U_{\mathrm{gas}}|$), L is the characteristic length (in this case, it is the equivalent spherical diameter of the droplet prior to break up), and σ is the surface tension coefficient. A probability density function (PDF) of the logarithm of $\mathrm{We}_{\mathrm{local}}$ number is developed from every breakup within the simulation (Fig. 7). The values are centered roughly about $\mathrm{We}_{\mathrm{local}} = 0.37$. These are very small values of the Weber number, which likely indicates that aerodynamic forces are not major factors in breakup in this system. This is logical, because the simulation is liquid injected into quiescent air.

3.6. Atomization Evolution

The evolution of local droplet characteristics and global jet development throughout the atomization process are important attributes, which can help elucidate the underlying physics of atomization. Understanding how droplets change from first breakup off of the liquid core to a final droplet in dilute spray could provide useful information to atomization model developers seeking to not only describe the final spray formation, but also the intermediate spray development. The tool provided in this work enables researchers to extract droplet statistics from throughout a simulation and analyze the evolution of these droplets and the system as a whole.

3.6.1. Droplet Shape and Size Evolution

Fig. 8 displays a probability density function of the equivalent spherical diameter of droplets as a function of the number of breakup events. A reasonable trend toward smaller and more uniformly sized droplets is seen, which provides confidence in the values extracted by the tool. Fig. 9 displays the change in diameter of droplets between breakup stages. Notice that not all values are negative. This further confirms the prevalence of coalescence events within these systems. Positive values in the figure indicate that a significant portion of droplets undergo coalescence and increase in size between breakup events. Additionally, notice the tendency of droplets to change size less as more breakup events occur, with the PDFs becoming more narrow and centered around 0 μm.

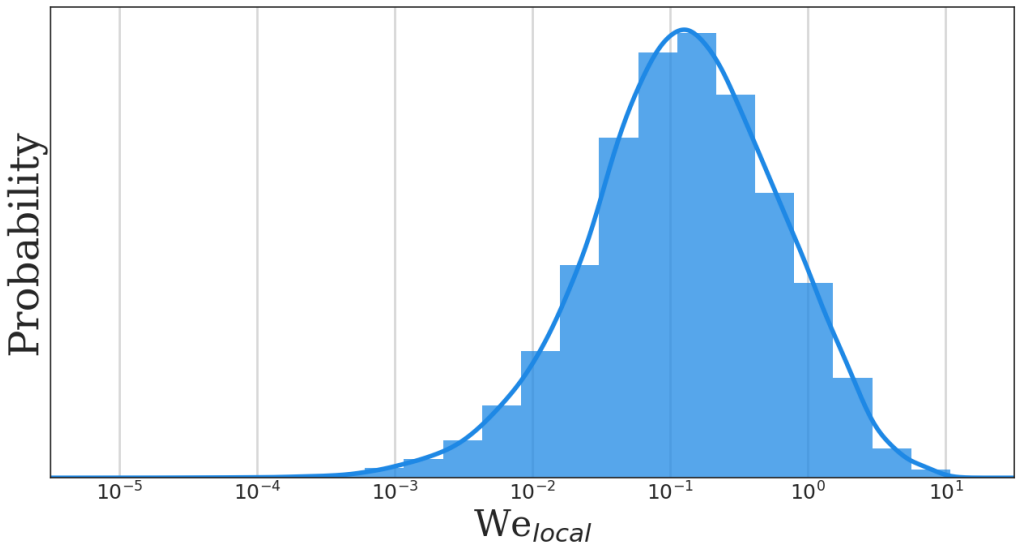

Figure 7: A probability density function of the local Weber Number associated with every breakup in the simulation.

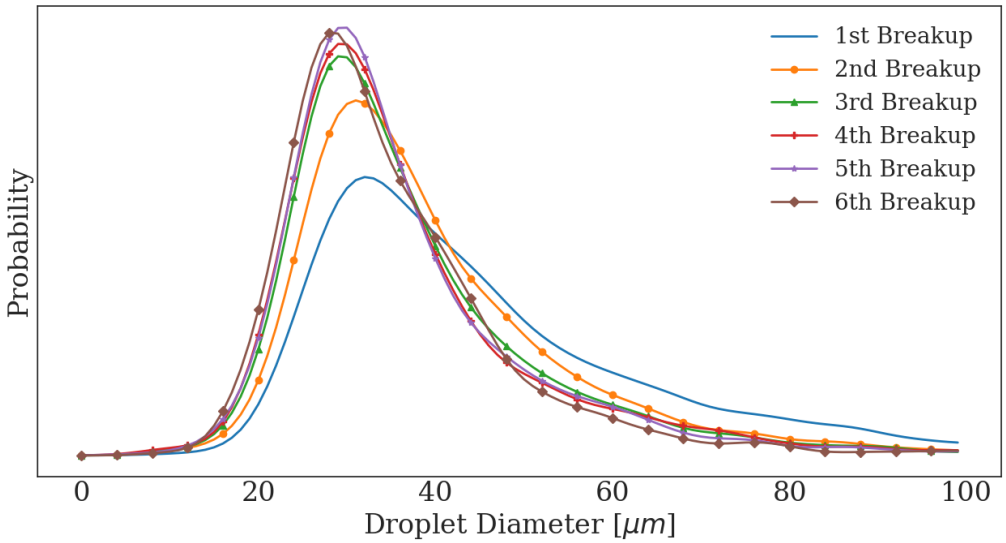

Figure 8: A probability density function of droplet diameter as a function of number of breakups

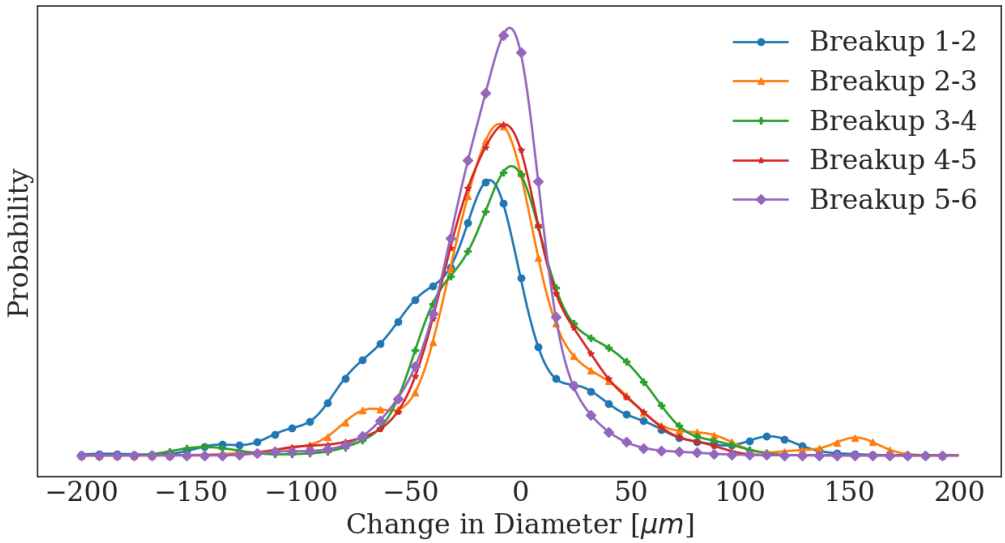

Figure 9: A probability density function displaying the change in diameter between breakup events.

3.6.2. Time Evolution

Because the extraction tool samples from every timestep in which breakup occurs. Thus, once the jet develops, breakup occurs on every timestep, providing high resolution time-series statistics describing evolution of the jet. Fig. 10 shows an example of some of the time dependent statistics that were extracted in the present work. These plots describe the rate of droplet development in the simulation. It is clear from this plot that secondary atomization quickly dominates droplet production in this system. Future work will focus on improving the extracted quantities and further explore how atomizing systems develop through time.

4. Conclusions

An extraction tool for numerical simulations of atomization was developed and tested. Improvements upon the methodology originally proposed by Rubel and Owkes [40] were made. Improvements include 1) an updated algorithm to stop fictitious identification of merge and split events, 2) enabling a wider temporal sampling range, and 3) the extraction of more atomization statistics.

A diesel-type jet was simulated and droplet statistics were extracted. The tool displayed utility for extracting data relevant to atomization model developers as well as providing previously inaccessible information on the underlying mechanisms of atomization. The data extracted are preliminary, but show promise for the tool's utility in future high-resolution studies.

Future work will focus on the implementation of the tool into high-resolution atomization simulations to extract useful data to help better quantify atomization mechanisms on global and local scales. Special effort will be given to aiding reduced-order model developers to develop relevant statistics for creating more accurate models. Additionally, we will focus on the improvement of sampling methods, particularly the local flow field to attempt to extract not only velocity magnitudes, but also topological flow data to better understand the small-scale turbulence that effects liquid breakup.

5. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1749779.

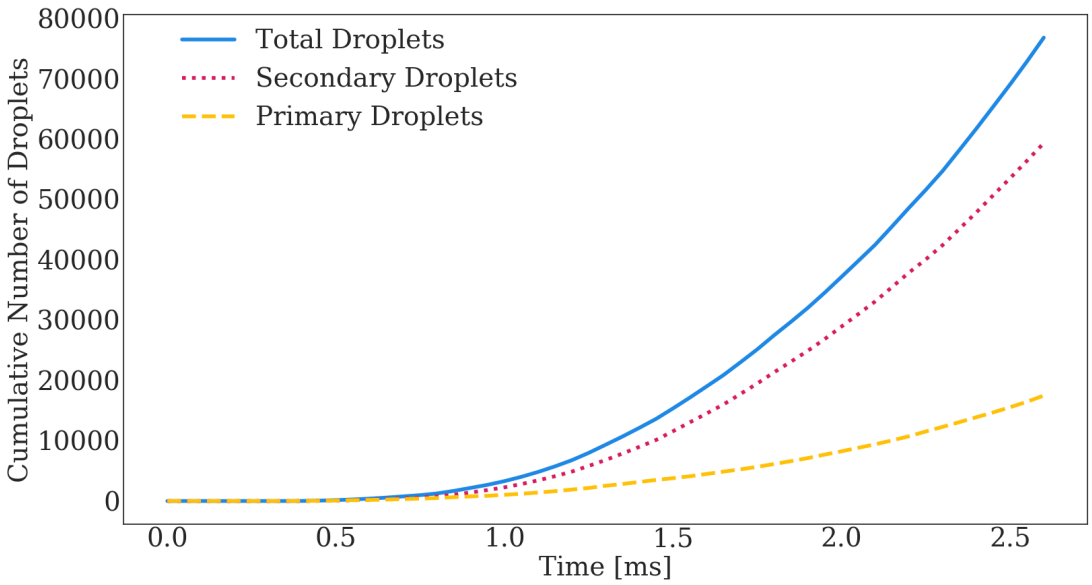

Figure 10: The cumulative number of primary, secondary, and total droplets. Note: secondary droplets in this image represent the classical definition of secondary droplets (i.e. all breakup following primary).

Computational efforts were performed on the Hyalite High Performance Computing System, operated and supported by University Information Technology Research Cyberinfrastructure at Montana State University.

References

[1] F Zhao, M.-C Lai, and D L Harrington. Automotive spark-ignited direct-injection gasoline engines. *Progress in Energy and Combustion Science*, 25:437–562, 1999.

[2] S. Edward Law. Agricultural electrostatic spray application a review of significant research and development during the 20th century. *Journal of Electrostatics*, 51:25–42, 2001.

[3] Yuh-Fun Maa, Phuong-Anh Nguyen, Theresa Sweeney, Steven Shire, and Chung Hsu. Protein inhalation powders: Spray drying vs spray freeze drying. *Pharmaceutical Research*, 16, 7 1999.

[4] E Fuentes, H Coe, D Green, G De Leeuw, and G Mcfiggans. Laboratory-generated primary marine aerosol via bubble-bursting and atomization. *Atmospheric Measuring Techniques*, 3:141–162, 2010.

[5] Felix Savart. Mémoire sur la constitution des veines liquides lancées par des orifices circulaires en mince paroi. *Annales de chimie et de physique*, 53:337–386, 1833.

[6] Lord Rayleigh. On the instability of jets. *Proceedings of the London Mathematical Society*, 1878.

[7] Mark A Linne, Megan Paciaroni, James R Gord, and Terrence R Meyer. Ballistic imaging of the liquid core for a steady jet in crossflow. *Applied Optics*, 2005.

[8] David Sedarsky, Edouard Berrocal, and Mark Linne. Numerical analysis of ballistic imaging for revealing liquid breakup in dense sprays. *Atomization and Sprays*, 20, 2010.

[9] Pierre R. Slangen, Pierre Lauret, Frederic Heymes, Laurent Aprin, and Nicolas Lecysyn. High-speed imaging optical techniques for shock-wave and droplets atomization analysis. *Optical Engineering*, 55:121706, 6 2016.

[10] Y. J. Wang, Kyoung Su Im, K. Fezzaa, W. K. Lee, Jin Wang, P. Micheli, and C. Laub. Quantitative x-ray phase-contrast imaging of air-assisted water sprays with high weber numbers. *Applied Physics Letters*, 89, 2006.

[11] Theodore J. Heindel. X-ray imaging techniques to quantify spray characteristics in the near field. *Atomization and Sprays*, 28, 2018.

[12] Zhilong Li, Wenbo Zhao, and Zhijun Wu. Understanding transient internal flow processes in high-pressure nozzles using synchrotron radiation x-ray phase contrast imaging technology. *Atomization and Sprays*, 31, 2021.

[13] Mikhael Gorokhovski and Marcus Herrmann. Modeling primary atomization. *Annual Review of Fluid Mechanics*, 40:343–366, 2008.

[14] J. Shinjo and A. Umemura. Simulation of liquid jet primary breakup: Dynamics of ligament and droplet formation. *International Journal of Multiphase Flow*, 36:513–532, 7 2010.

[15] O Desjardins and H Pitsch. Detailed numerical investigation of turbulent atomization of liquid jets. *Atomization and Sprays*, 20:311–336, 2010.

[16] J. Shinjo and A. Umemura. Surface instability and primary atomization characteristics of straight liquid jet sprays. *International Journal of Multiphase Flow*, 37:1294–1304, 12 2011.
[17] D. Jarrahbashi and W. A. Sirignano. Invited article: Vorticity dynamics for transient high-pressure liquid injection. *Physics of Fluids*, 26, 9 2014.
[18] Tomoaki Watanabe, Yasuhiko Sakai, Kouji Nagata, Yasumasa Ito, and Toshiyuki Hayase. Vortex stretching and compression near the turbulent/non-turbulent interface in a planar jet. *Journal of Fluid Mechanics*, 758:754–785, 11 2014.
[19] D. Jarrahbashi, W. A. Sirignano, P. P. Popov, and F. Hussain. Early spray development at high gas density: Hole, ligament and bridge formations. *Journal of Fluid Mechanics*, 792:186–231, 3 2016.
[20] Xiaoyi Li and Marios C. Soteriou. High fidelity simulation and analysis of liquid jet atomization in a gaseous crossflow at intermediate weber numbers. *Physics of Fluids*, 28, 8 2016.
[21] A. Zandian, W. A. Sirignano, and F. Hussain. Planar liquid jet: Early deformation and atomization cascades. *Physics of Fluids*, 29, 6 2017.
[22] A. Zandian, W. A. Sirignano, and F. Hussain. Understanding liquid-jet atomization cascades via vortex dynamics. *Journal of Fluid Mechanics*, 843:293–354, 5 2018.
[23] Rolf D. Reitz and R. Diwakar. Structure of high-pressure fuel sprays. *SAE Technical Papers*, 2 1987.
[24] Jennifer C. Beale and Rolf D. Reitz. Modeling spray atomization with the kelvin-helmholtz/rayleigh-taylor hybrid model. *Atomization and Sprays*, 9, 1999.
[25] D. Fuster, J. P. Matas, S. Marty, S. Popinet, J. Hoepffner, A. Cartellier, and S. Zaleski. Instability regimes in the primary breakup region of planar coflowing sheets. *Journal of Fluid Mechanics*, 736:150–176, 12 2013.
[26] Suraj S. Deshpande, Soumil R. Gurjar, and Mario F. Trujillo. A computational study of an atomizing liquid sheet. *Physics of Fluids*, 27:082108, 8 2015.
[27] Arpit Agarwal and Mario F. Trujillo. A closer look at linear stability theory in modeling spray atomization. *International Journal of Multiphase Flow*, 109:1–13, 12 2018.
[28] Will Krolick and Mark Owkes. Primary atomization instability extraction using dynamic mode decomposition. *Atomization and Sprays*, 28:1061–1079, 2018.
[29] Josef Hasslberger, Sebastian Ketterl, Markus Klein, and Nilanjan Chakraborty. Flow topologies in primary atomization of liquid jets: A direct numerical simulation analysis. *Journal of Fluid Mechanics*, 859:819–838, 1 2019.
[30] Antonio J. Torregrosa, Raúl Payri, F. Javier Salvador, and Marco Crialesi-Esposito. Study of turbulence in atomizing liquid jets. *International Journal of Multiphase Flow*, 129, 8 2020.
[31] F. J. Salvador, S. Ruiz, Marco Crialesi-Esposito, and Ignacio Blanquer. Analysis on the effects of turbulent inflow conditions on spray primary atomization in the near-field by direct numerical simulation. *International Journal of Multiphase Flow*, 102:49–63, 5 2018.
[32] Peter J. O'Rourke and Anthony A. Amsden. The tab method for numerical calculation of spray droplet breakup. *SAE Technical Papers*, 11 1987.
[33] M Pilch and C A Erdman. Use of breakup time data and velocity history data to predict the maximum size of stable fragments for acceleration-induced breakup of a liquid drop. *International Journal of Multiphase Flow*, 13:741–757, 1987.
[34] Spray Teclllwlvgy and Rolf D Reitz. Modeling atomization processes in high-pressure vaporizing sprays. *Science*, 3, 1987.
[35] Mark A. Patterson and Rolf D. Reitz. Modeling the effects of fuel spray characteristics on diesel engine combustion and emission. *SAE Technical Papers*, 1998.
[36] Yue Wang, Won Geun Lee, Rolf D. Reitz, and Ramachandra Diwakar. Numerical simulation of diesel sprays using an eulerian-lagrangian spray and atomization (elsa) model coupled with nozzle flow. *SAE Technical Papers*, 4 2011.
[37] C. Wang, S. Chang, H. Wu, and J. Xu. Modeling of drop breakup in the bag breakup regime. *Applied Physics Letters*, 104, 4 2014.
[38] C Wang, S Chang, H Wu, L Ding, and J M Thompson. Theoretical modeling of spray drop deformation and breakup in the multimode breakup regime. *Atomization and Sprays*, 25:857–869, 2015.
[39] C. Sula, H. Grosshans, and M. V. Papalexandris. Assessment of droplet breakup models for spray flow simulations. *Flow, Turbulence and Combustion*, 105:889–914, 9 2020.
[40] Clark Rubel and Mark Owkes. Extraction of droplet genealogies from high-fidelity atomization simulations. *Atomization and Sprays*, 29:709–739, 2019.
[41] Olivier Desjardins, Guillaume Blanquart, Guillaume Balarac, and Heinz Pitsch. High order conservative finite difference scheme for variable density low mach number turbulent flows. *Journal of Computational Physics*, 227:7125–7159, 7 2008.
[42] Olivier Desjardins, Vincent Moureau, and Heinz Pitsch. An accurate conservative level set/ghost fluid method for simulating turbulent atomization. *Journal of Computational Physics*, 227:8395–8416, 9 2008.
[43] Mark Owkes and Olivier Desjardins. A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (vof) method. *Journal of Computational Physics*, 270:587–612, 8 2014.
[44] Mark Owkes and Olivier Desjardins. A mass and momentum conserving unsplit semi-lagrangian framework for simulating multiphase flows. *Journal of Computational Physics*, 332:21–46, 3 2017.
[45] Haecheon Choi and Parviz Moin. Effects of the computational time step on numerical solutions of turbulent flow. *Journal of Computational Physics*, 113, 7 1994.
[46] David L. Youngs. *Numerical Methods in Fluid Dynamics*, chapter Time-Dependent Multi-material Flow with Large Fluid Distortion, pages 273–285. Academic Press, 1 1982.
[47] James Edward Pilliod and Elbridge Gerry Puckett. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. *Journal of Computational Physics*, 199:465–502, 9 2004.
[48] Ronald P. Fedkiw, Tariq Aslam, Barry Merriman, and Stanley Osher. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). *Journal of Computational Physics*, 152, 1999.
[49] J. E. Dendy. Black box multigrid. *Journal of Computational Physics*, 48, 1982.
[50] Mark Owkes, Eric Cauble, Jacob Senecal, and Robert A. Currie. Importance of curvature evaluation scale for predictive simulations of dynamic gas–liquid interfaces. *Journal of Computational Physics*, 365:37–55, 7 2018.

[51] M. Herrmann. A parallel eulerian interface tracking/lagrangian point particle multi-scale coupling procedure. *Journal of Computational Physics*, 229:745–759, 2 2010.

[52] Kelli Hendrickson, Gabriel D. Weymouth, and Dick K.P. Yue. Informed component label algorithm for robust identification of connected components with volume-of-fluid method. *Computers and Fluids*, 197, 1 2020.

[53] J. Hoshen and R. Kopelman. Percolation and cluster distribution. i. cluster multiple labeling technique and critical concentration algorithm. *Physical Review B*, 14, 1976.

[54] Hank Childs, Eric Brugger, Brad Whitlock, Jeremy Meredith, Sean Ahern, David Pugmire, Kathleen Biagas, Mark Miller, Cyrus Harrison, Gunther H. Weber, Hari Krishnan, Thomas Fogal, Allen Sanderson, Christoph Garth, E. Wes Bethel, David Camp, Oliver Rübel, Marc Durant, Jean M. Favre, and Paul Navrátil. Visit: An end-user tool for visualizing and analyzing very large data. In *High Performance Visualization–Enabling Extreme-Scale Scientific Insight*, pages 357–372. October 2012.

[55] Surya Prakash, Suhas S Jain, Jeffery A Lovett, B N Raghunandan, R V Ravikrishna, and Gaurav Tomar. Detailed numerical simulations of atomization of a liquid jet in a swirling gas crossflow. *Atomization and Sprays*, pages 1–28, 2019.

Appendix A. Liquid Identification and Coalescence Code

The following is a pseudocode describing how the liquid identification number is transported and coalescence is identified.

Algorithm 1 $\mathcal{L}$-transport and Coalescence Pseudocode

```
for i = 1 → N_cell do                                          ▷ Loop over cells in domain
    ▷ Form list of L for this cell
    L_count = 0
    if cell has interface or VoF > 0.5 then                    ▷ Search through cells with liquid
        if L > 0 then
            L_count ← L_count + 1                              ▷ Keep track of the number of unique L values within the cell
            L_cell(L_count) ← L                                ▷ Keep track of L value in the cell
        end if
    end if
    for f = 1 → N_faces do                                     ▷ Loop over all faces of a cell
        for n = 1 → L_max do                                   ▷ Loop through all L that can flux through a face
            if L_flux,n ≠ 0 then                               ▷ Check to see if there is an n^th L flux into the cell
                if L_flux,n ≠ L_cell(1 : L_cell) then          ▷ Check to see if L fluxes matches any of the current L in the cell
                    L_count ← L_count + 1                      ▷ Update L counter in the cell
                    L_cell(L_count) ← L_flux                   ▷ Keep track of all L values in the cell
                end if
            end if
        end for
    end for
    ▷ Check for coalescence or merge event
    if L_count ≥ 2 then                                        ▷ More than one unique L for this cell
        for n = 2 → L_count do
            nMerge = nMerge+1                                  ▷ Keep track of the number of coalescence events
            L1(nMerge) = min(L_cell(1), L_cell(n))             ▷ Make a list of the L's which need to be merged
            L2(nMerge) = max(L_cell(1), L_cell(n))
        end for
    end if
end for
▷ Compute new L after coalescence event
for n = 1 → nMerge do                                          ▷ Loop over the all identified coalescence events
    if L1(n) = L1(n + 1) then                                  ▷ Combine identical events
        if L2(n) = L2(n + 1) then
            L_new ← min(L1, L2)                                ▷ Assign the smaller L to the new merged structure
            L_old ← max(L1, L2)
        end if
    end if
end for
▷ Update L in domain
where (L = L_old): L ← L_new                                   ▷ Update L's throughout domain
```

Appendix B. Neo4j Data Input

Data from our simulation are exported in the form of a CSV file. Each row within the CSV represents a breakup event within the simulation and each column contains statistics we extracted from the event. Column headers are listed in the following Cypher code, following the "csvline" variable (i.e. csvline.OldLID and csvline.NewLID are the old and new $\mathcal{L}$'s associated with each droplet). We import the CSV into Neo4j to create nodes and relationships. Each node corresponds to a row within the CSV and the relationships are created using the identification numbers.

```
// Import data from CSV and create split/merge nodes
LOAD CSV WITH HEADERS FROM "link_to_csv" AS csvline CREATE (n:droplet {id:
    TOINTEGER(csvline.NewLID), Event: csvline.Merge_Split, OldLID: TOINTEGER(
    csvline.OldLID), OldSID: TOINTEGER(csvline.OldSID), NewLID: TOINTEGER(
    csvline.NewLID), NewSID: TOINTEGER(csvline.NewSID), Volume: TOFLOAT(
    csvline.Vol), Event_Time: TOFLOAT(csvline.Time), X: TOFLOAT(csvline.X), Y
    : TOFLOAT(csvline.Y), Z: TOFLOAT(csvline.Z), U: TOFLOAT(csvline.U), V:
     TOFLOAT(csvline.V), W: TOFLOAT(csvline.W), U_gas: TOFLOAT(csvline.U_gas
    ), V_gas: TOFLOAT(csvline.V_gas), W_gas: TOFLOAT(csvline.W_gas), L1_old:
      TOFLOAT(csvline.L1_old), L2_old: TOFLOAT(csvline.L2_old), L3_old:
    TOFLOAT(csvline.L3_old), L1_new: TOFLOAT(csvline.L1_new), L2_new:
    TOFLOAT(csvline.L2_new), L3_new: TOFLOAT(csvline.L3_new), Vol_old:
    TOFLOAT(csvline.Vol_old)})

// Create merge relations between droplets
MATCH (n:droplet{Event:"Split"}),(d:droplet{Event:"Split"}),(m:droplet{Event:
    "Merge"})
WHERE n.NewLID = m.NewLID AND d.NewLID = m.OldLID
CREATE (d)-[:Merge]->(n)

// Create merge relations between droplets and core
MATCH (n:droplet{Event:"None"}),(d:droplet{Event:"Split"}),(m:droplet{Event:"
    Merge"})
WHERE n.NewLID = m.NewLID AND d.NewLID = m.OldLID
CREATE (d)-[:Merge]->(n)

// Deletes merge nodes
MATCH (d:droplet)
WHERE d.Event = 'Merge'
DELETE d;

// Creates split relations
MATCH (n:droplet),(d:droplet)
WHERE n.Event = "Split" and n.OldLID = d.NewLID
CREATE (d)-[:Split]->(n);
```

Appendix C. Python Neo4j Driver Script

The below script uses the py2neo library to query a Neo4j graph database from Python. This script was used to analyze the data presented in Fig. 10. First, communication with a graph is established. The graph must be running in Neo4j when the script is executed. Then, a time series is created and result arrays are initialized. Following this, the Cypher queries are executed. To use a Python variable within the Cypher query, a $ must be placed immediately before the variable, then the variable is defined in the second argument in the ".run" function. The ".evaluate()"

function ensures that only values from the query are returned, otherwise results appear as a dictionary. Results from the query are then appended to the results lists to be used for further analysis.

```
import numpy as np
from py2neo import Graph
# Using Python Neo4j API "py2neo" to query a graph database
gs = Graph("<Graph URI>",password="<Graph Password>") # Graph must be running
    on the Neo4j platform for this to work. First input is graph URI and
    second is a required password

# Set up time array
dt = 5e-5
time = []
x = np.arange(0,2.65e-3,dt)

# Convert time class from numpy.float64 to python native float
for t in x:
    time.append(t.item())

# Initialize result arrays
nd = [] # total number of droplets
n1 = [] # number of primary droplets
n2 = [] # total number of secondary droplets

# Run Cypher Queries

# Total Droplets
for t in time:
    g = gs.run("MATCH(n:droplet),(c:core) WHERE n.Event_Time <= $t RETURN
        count(n)",t=t).evaluate() # evaluate() returns only values from the
        query
    nd.append(g)
# Primary Droplets
for t in time:
    g = gs.run("MATCH(n:primary),(c:core) WHERE n.Event_Time <= $t RETURN
        count(n)",t=t).evaluate()
    n1.append(g)
# All secondary
for t in time:
    g = gs.run("MATCH(n:droplet),(c:core) WHERE n.Event_Time <= $t AND NOT n:
        primary RETURN count(n)",t=t).evaluate()
    n2.append(g)
```