Computers and Fluids 254 (2023) 105808

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

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

ARTICLE INFO

Keywords:
Atomization
Data extraction
NGA
High-fidelity simulation
Reduced-order models

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 modeling and computational resources allow 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 as they occur and extract 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 the view of the liquid core and severely limits the ability to study primary instability and breakup mechanics. Recent advances in ballistic [7–9] and X-ray [10–12] 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.

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 that 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

* Corresponding author.
E-mail address: brendan.christensen@montana.edu (B. Christensen).

https://doi.org/10.1016/j.compfluid.2023.105808
Received 5 April 2022; Received in revised form 9 January 2023; Accepted 30 January 2023
Available online 3 February 2023

descriptions of breakup by integrating this additional dimension to their analysis [16–22]. Additionally, many researchers focus their efforts on comparing linear-stability theory [23,24] and high-fidelity simulations [16,25–28]. These studies process renderings 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 throughout the simulation using constrained spatial and/or temporal domains to limit data-processing requirements [29–31].

The numerical studies mentioned above have greatly improved our 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 in 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], the Eulerian–Lagrangian spray and atomization (ELSA) model [35], the bag type breakup (BTB) model [36], the multi-mode breakup (MMB) model [37], and the modified Taylor analogy breakup (MTAB) model [38]. 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 [39], that extracts relevant data from breakup and coalescence events in atomization simulations. The tool operates concurrently with the simulation, extracting information from every breakup or coalescence event. This eliminates the need for extensive post-processing of massive simulation datasets. Improvements to the tool include a major revision of the breakup and coalescence identification algorithm, implementation of a greater temporal sampling range, and the addition of several new data types to be extracted. The values extracted are not entirely novel, however, the ability to extract these values from around every breakup event in an atomizing system is novel. Previous studies rely on time-consuming post-processing methods and/or must constrain the data they extract either temporally or spatially, which limits the statistical relevance of the dataset. The present work provides a method to gather significant quantities of these data without constraining the location or time within the simulation. Through the use of this tool, many researchers will gain access to extensive and easily accessible data sets from atomization simulations.

2. Methods

The droplet physics extraction tool, originally proposed in Rubel and Owkes [39], 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 with 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 [40–43]. 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 [44]. Away from the phase interface, mass, momentum, and any other scalars are transported with conservative, high-order finite difference operators [40]. Near the interface, a geometric volume-of-fluid (VOF) method is utilized to ensure conservative transport of mass and momentum [42,43]. The VOF scheme utilizes semi-Lagrangian transport to predict the volume of liquid that fluxes through each cell face during a timestep. This is accomplished through the calculation of geometric flux regions from the velocity field. In Fig. 1, the geometric flux regions are displayed as the shaded regions. These are signed volume regions that calculate fluxes into or out of cells in a given timestep. They are constructed from velocities at the cell vertices. This formulation of geometric transport is un-split, and the use of cell vertex velocities prevents flux region overlap and an accumulation of errors. The interface is reconstructed using a piece-wise linear interface reconstruction (PLIC) [45] with interface normal vectors computed using the efficient least-squares VOF interface reconstruction algorithm (ELVIRA) [46]. The pressure Poisson equation is solved utilizing the ghost fluid method [47] and a black box solver [48]. Interface curvature is computed with the adjustable curvature evaluation scale (ACES) method [49]. NGA is fully parallelized with a message passing interface (MPI) protocol and scales well to tens of thousands of cores [15].

2.2. Breakup and coalescence event identification

The coalescence and breakup 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 the liquid moves. $\mathcal{L}$'s are assigned to every region where the volume fraction of liquid is greater than zero. Because these values are intended to provide a history of liquid movement, $\mathcal{L}$'s 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. Details of how $\mathcal{L}$ is transported is described below in Section 2.2.2 as coalescence identification is closely tied with $\mathcal{L}$ transport.

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 [50]. In the present application, the main steps of the algorithm are:

1. $\mathcal{S}$ is assigned on the entire domain to be zero
2. Loop over the domain to find a cell with liquid and where $\mathcal{S} = 0$
3. Incorporate adjacent cells into the structure using a key logical condition, which requires:
   (a) There is liquid in the cell
   (b) $\mathcal{S} = 0$ (cell has not already been tagged)
   (c) $\mathcal{L}$ of the cell equals that of the current cell (avoids merging neighboring droplets)
4. Repeat process until there are no more liquid cells with $\mathcal{S} = 0$

Here additional details of the algorithm to compute the $\mathcal{S}$'s is provided. 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 $\mathcal{S} = 0$. When an untagged liquid node is found, all adjacent cells are compared, using a key logical statement to create continuous liquid structures. This statement has been modified in the present work and drastically improves the accuracy of breakup and coalescence identification. The logical statement requires that: (a) there is liquid in that cell, (b) it is untagged ($\mathcal{S} = 0$), and (c) the $\mathcal{L}$ of the adjacent cell must equal that of the current cell. The third requirement is the primary modification and ensures that structures, which were not together at the previous timestep (meaning different $\mathcal{L}$ values), do not coalesce in adjacent cells. Additionally, the algorithm does not prematurely identify breakup. In order for structures to break up, there must be a full cell between them. This process prevents structures from breaking up and re-merging immediately following breakup as structures advance and naturally inhabit adjacent cells. Herrmann's work also described the parallelization of this process, which is utilized in the extraction tool. This requires that 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. [51,52].

2.2.2. Coalescence identification

Coalescence of liquid structures is identified by locating liquid structures with more than one unique associated $\mathcal{L}$. The un-split, semi-Lagrangian VOF scheme, described in Section 2.1, predicts the volume of liquid that fluxes through a cell face. So, when liquid is fluxed through a cell face, the $\mathcal{L}$ that inhabits that liquid will also be fluxed. $\mathcal{L}$ fluxes are assigned a sign (+) or (-) to track the direction they move through the face using the same sign convention as the VOF transport. $\mathcal{L}$ transport is similar to the transport of liquid, however, when multiple $\mathcal{L}$'s move into the same cell, they must be treated separately instead of being summed. An $\mathcal{L}_{\text{Flux}}$ list is created for each cell, which tracks all the unique values of $\mathcal{L}$ that inhabit or move into the cell within that timestep. $\mathcal{L}_{\text{Flux}}$ lists store all the $\mathcal{L}$'s that flux into a given cell during a timestep. These lists are looped through and any containing multiple unique $\mathcal{L}$'s indicate coalescence. This represents a significant change in the algorithm. Previously, structures that were in adjacent cells could be identified as coalesced, which led to repeated non-physical breakup and coalescence identifications (see Section 2.2.4). Fig. 1 displays a visual representation of this process. See Appendix A for the $\mathcal{L}$ transport and coalescence identification algorithm. 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.

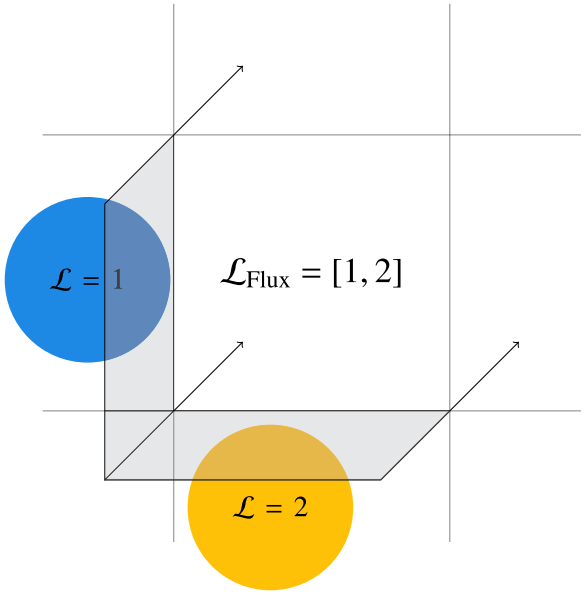

Fig. 1. An example of two droplets fluxing into the same cell. Shaded regions are semi-lagrangian flux regions [42], which transport liquid and $\mathcal{L}$'s into the center cell due to the velocity shown with vectors. The $\mathcal{L}$'s of the two droplets are stored in the list $\mathcal{L}_{\text{Flux}}$.

2.2.3. Breakup identification

Liquid structure splitting or breakup is identified when two different structures (two unique $\mathcal{S}$'s) have the same $\mathcal{L}$. 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. If the two structures have the same $\mathcal{L}$, this indicates that a breakup occurred. Since $\mathcal{L}$'s are persistent through time, the same value in neighboring cells indicates that these two droplets were part of the same structure at the previous timestep. 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.

2.2.4. Addressing fictitious events

An issue from the previous work, described in Rubel and Owkes [39] 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 authors of the original algorithm opted to run the breakup identification portion of the code every 10 timesteps to avoid an accumulation of non-physical results. This severely limited the potential usefulness of the tool.

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 coalescing into their parent droplet when they were in neighboring cells. The present work introduces two major updates to address this issue.

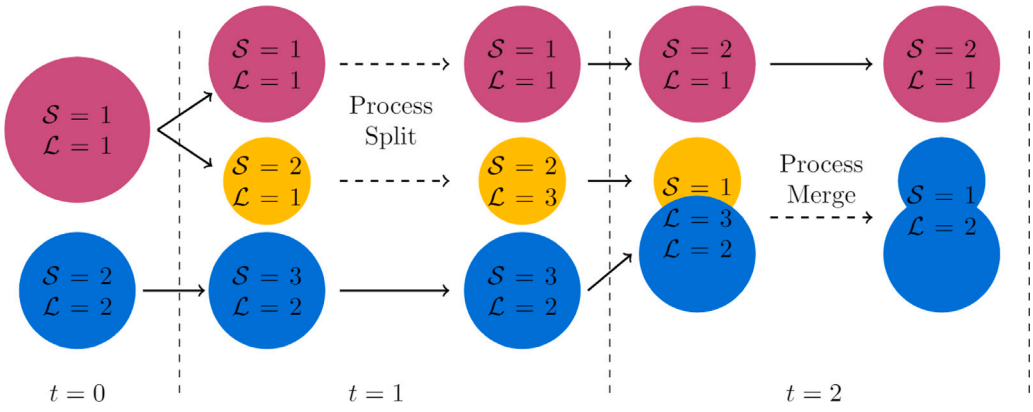

Fig. 2. An example of two droplets undergoing breakup and coalescence. Two droplets start on the left, and the top droplet breaks into two droplets. The two droplets are identified and the split event is processed. The bottom two droplets coalesce, the merge is identified and the merge event is processed. 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.

First, the algorithm which identifies continuous liquid structures was improved by introducing a logical statement. The statement requires that for neighboring cells to be labeled as existing within the same structure, the cells must have the same $\mathcal{L}$. Second, the algorithm which identifies coalescence was rewritten to create $\mathcal{L}_{Flux}$ lists for each cell. This was implemented to only identify coalescence when multiple $\mathcal{L}$'s occupy the same cell, indicating two liquids, which were separate at the previous timestep have merged together. Together these changes require droplets to be separated by at least one grid cell for breakup to be identified, contrarily coalescence requires droplets to occupy the same grid cell. This difference in length scales for breakup and coalescence ensures that droplets do not breakup and then fictitiously coalesce. For more information on these updates, see Section 2.2.1. The updated algorithms drastically boost the accuracy of the tool and allow it to be run at every timestep without erroneous event identifications. This provides a high-resolution sampling of data as the jet breaks up further and breakup and coalescence happen more frequently.

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

The updated algorithm used to identify breakup and coalescence is run every timestep and thus identifies these events when they occur throughout the entire simulation. When an event is identified, important statistics including the location, volume, local flow field, and shape characteristics are computed for each liquid structure and associated with that event. These statistics are computed for each structure present in the simulation at a timestep and stored in structure arrays. The structure arrays are linked lists of the statistics and identification numbers that are used to tie the statistics to the breakup and coalescence events.

The original work identified breakup events and extracted data every 10 timesteps causing the extracted data to be temporally removed from the actual event. The updated algorithm runs every timestep, thus identifying breakup when it occurs, furthermore, the temporal sampling range was extended by extracting data from the timestep immediately preceding breakup in addition to the timestep following breakup. We accomplish this by saving the structure arrays from the previous timestep, which contain the statistics and $\mathcal{L}$'s of the structures at that time. Then, when breakup occurs, the data extraction algorithm searches the old structure arrays for the structure which has an $\mathcal{L}$ equal to the old $\mathcal{L}$ of the newly split structures. This improvement provides insight into the conditions which lead to breakup and the resultant structures from the breakup.

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. We calculate structure volume simply as

$$\mathcal{V}_{\text{struct}} = \sum_{i=1}^{N_s} \mathcal{V}_{\text{cell},i}\ \alpha_i \tag{3}$$

where N_s is the total number of cells encompassing the structure, $\mathcal{V}_{\text{cell},i}$ is the volume of the ith cell, and α_i is the volume fraction of liquid within the ith cell. For the purposes of this study, liquid and gas velocities are calculated as volume-averaged velocities,

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

$$\boldsymbol{U}_{\text{gas}} = \frac{\sum_{i=1}^{N_s} \boldsymbol{u}_i \mathcal{V}_{\text{cell},i}\ (1-\alpha_i)}{\sum_{i=1}^{N_s} \mathcal{V}_{\text{cell},i}\ (1-\alpha_i)} \tag{5}$$

respectively. These values are calculated by considering the calculated velocity field of all cells that contain a liquid structure and all cells adjacent to those. Where $\boldsymbol{u}_i$ is the velocity vector in the 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 that connect droplets through breakup and coalescence events. Storing atomization data in a graph

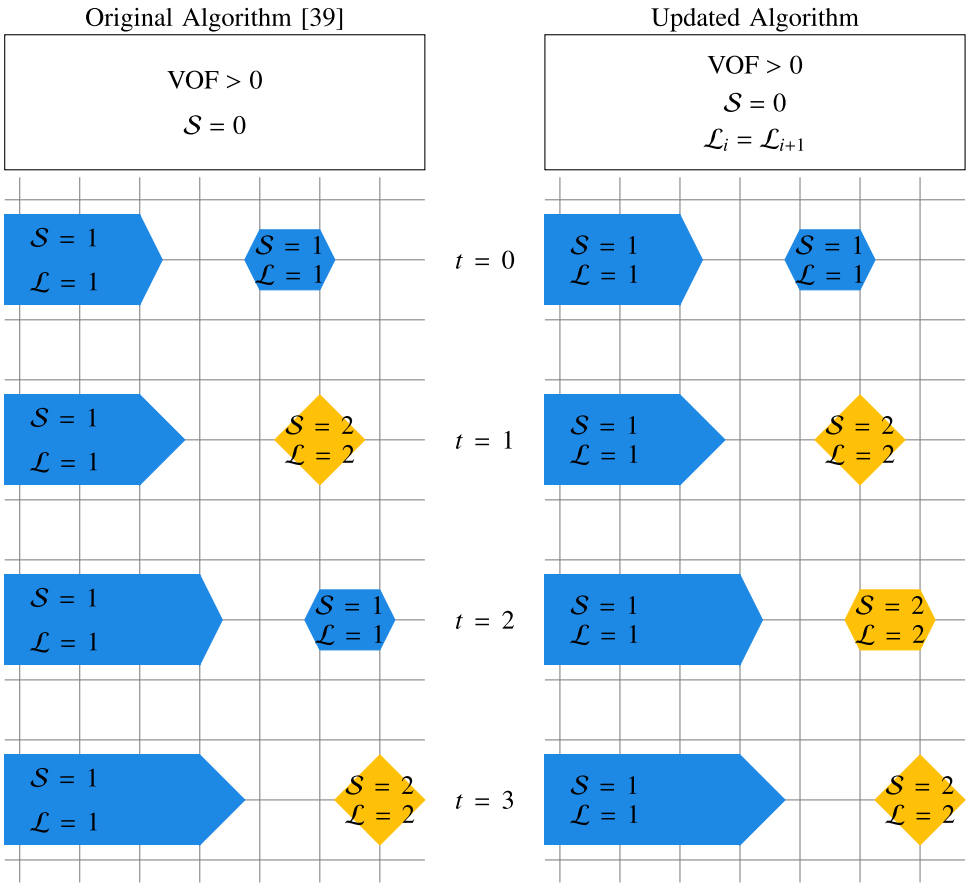

Fig. 3. The diagram on the left shows the issue with fictitious merging and splitting events in the original algorithm [39]. The diagram to the right shows how the split and merge identifier works in the present work. Notice that at $t = 2$ in the original algorithm, the two structures occupy neighboring cells and fictitiously merge back together. The updated code avoids the fictitious merge and 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 and the breakup identification uses $\mathcal{L}$ in the logical statement for identifying structures.

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 fourth breakup event. This image is made up of all the droplets which broke up 4 times during the first 2.3 μs of the simulation, and all the intermediate droplets between the coherent liquid core and these fourth 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
```

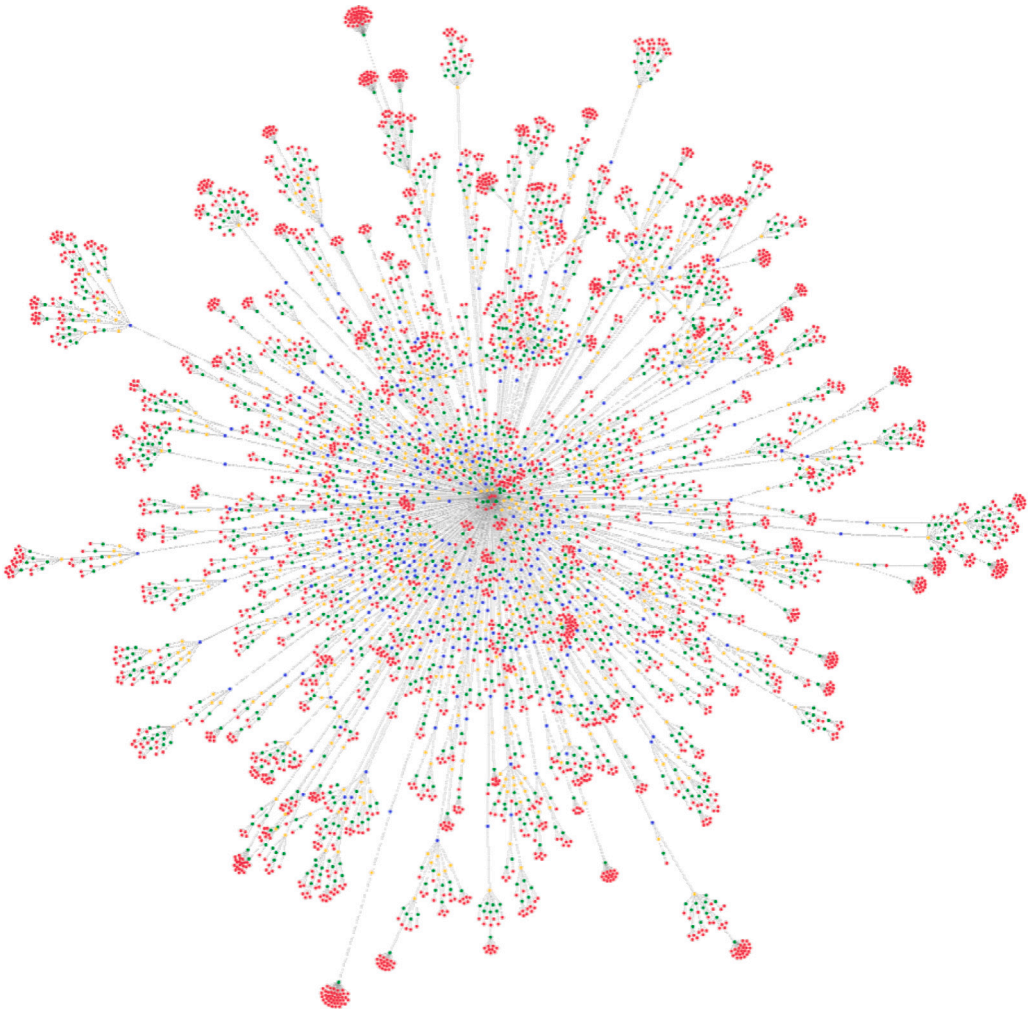

Fig. 4. Example of data points in Neo4j graphical database. Colors represent the number of breakups a structure has undergone (blue = primary, yellow = secondary, green = tertiary, red = quaternary). Lines connect related droplets, i.e., droplets which split from each other. The liquid core is present at the center of the image. Each node contains relevant statistics from the breakup event.

```
// 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 that 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 breakup from each other or coalesce together. From this information, we can easily output a CSV that 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 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

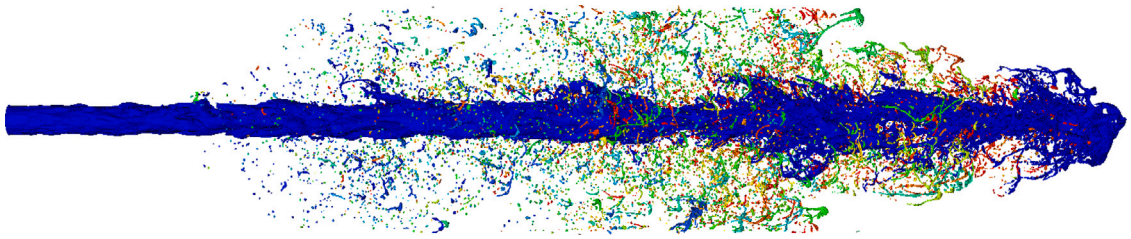

Fig. 5. Rendering of the liquid jet run in simulation. $\mathcal{L}$ values at final timestep represented by colors with $\mathcal{L}$ ranging from 1 to 76,608 for the number of breakups that occurred. Rendered using VisIt [53].

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

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

the time-series plots in Section 3.6.2. See Appendix C 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 [39] and consists of a turbulent liquid injection into acquiescent 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 $\mathcal{N}_x \times \mathcal{N}_y \times \mathcal{N}_z = 1024 \times 128 \times 128$. 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 (see Fig. 5).

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 [39], 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 these 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 is 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 [39], 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. Below, the number of breakup events that a droplet has undergone will be referred to as its breakup stage (e.g. a droplet which broke up three times is in the third breakup stage).

3.4. Coalescence analysis

Coalescence is a vital process in the study of atomization. Coalescing droplets will result in larger droplets, and in turn, lead to more breakup. It is well understood that coalescence occurs in atomization, however, without access to efficient extraction methods, researchers have not been able to fully quantify or analyze how it affects a full atomizing system. A recent study by Prakash et al. [54] was able to identify coalescence event locations on a limited time domain within their jet in cross-flow simulation. Our tool provides the ability to analyze the prevalence of coalescence events throughout entire atomizing systems. Furthermore, from the paths created by Neo4j, we can track the relationship between breakup and coalescence from the evolution of every liquid structure within a system. This provides statistically relevant insights into a little-understood process. 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

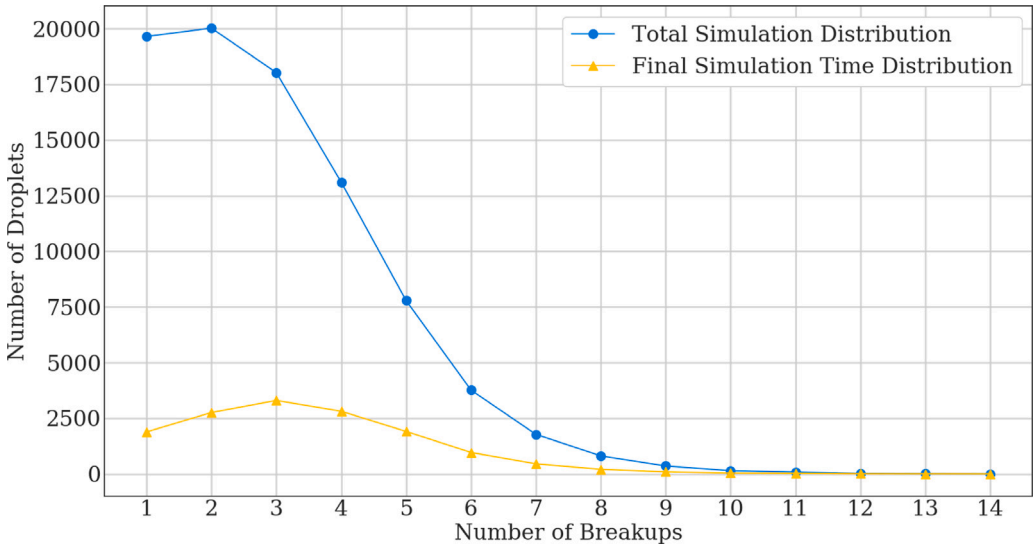

Fig. 6. Plot displaying the distribution of breakup stages throughout the entire simulation (blue, circle nodes), e.g. about 7,600 droplets underwent a 5th breakup throughout this system. At the end of the simulation (yellow, triangle nodes), there were about 2,300 droplets present that had undergone 5 breakups.

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.

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

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.

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 breakup), and σ is the surface tension coefficient. U_{liquid} and U_{gas} are calculated from Eqs. (4) and Eq. (5). 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 of liquid injected into quiescent air. This represents the first method to gather the local Weber number from every breakup event within a complex atomizing system. This type of data can be used to analyze how the effects of various atomization configurations and bulk parameters propagate through these systems and affect the physics on the local level.

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 the 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 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. 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 this system. 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. The coarse resolution of the present study affects the accuracy of these values related to a physical system. However, they display the ability to analyze large quantities of statistics relevant to the evolution of liquid shapes and sizes throughout an atomization simulation. The extraction of shape characteristics from breakup events can provide information on the intermediate processes of atomization. Current methods allow researchers to analyze final droplet size and shape distributions, however, there is little information on the mechanisms that lead to those droplets and the rate of change of droplet sizes. The extraction tool provides access to this type of information.

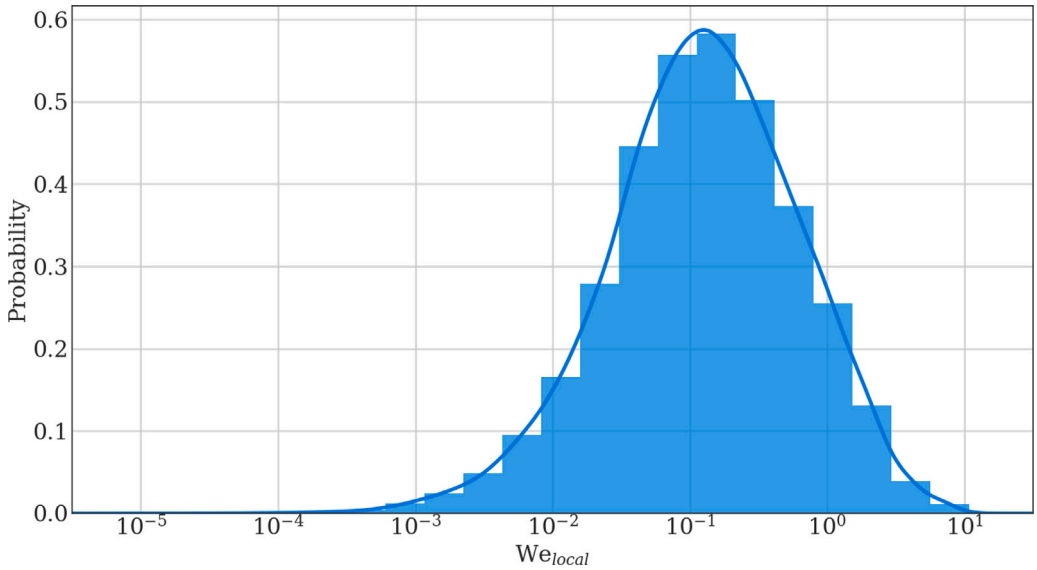

Fig. 7. A probability density function of the local Weber Number associated with every breakup in the simulation.

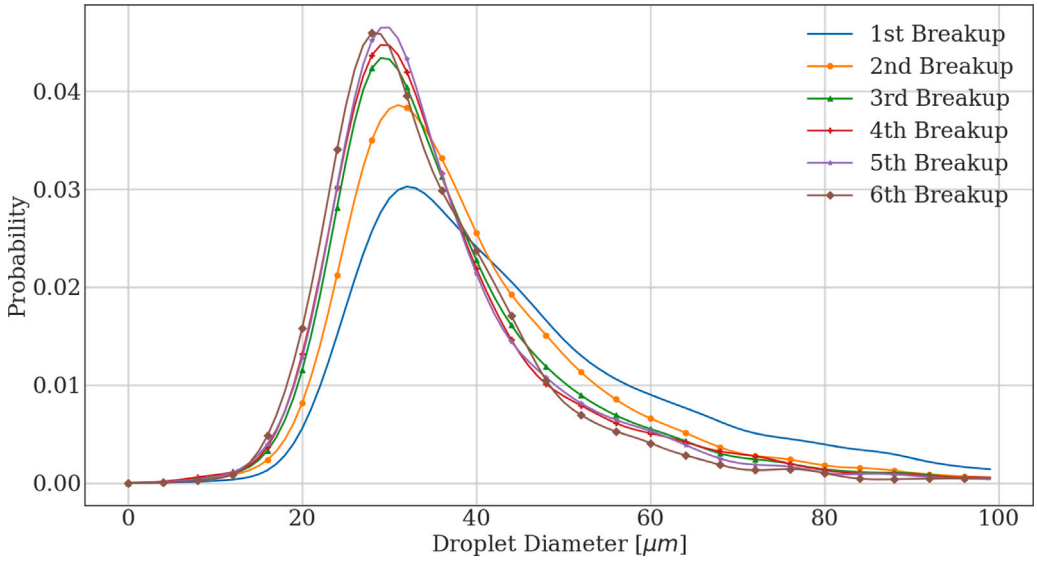

Fig. 8. A probability density function of droplet diameter as a function of the number of breakups the droplet has undergone.

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 the 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. Understanding when breakups are occurring throughout these systems is vital to better understanding atomization as well as in the development of reduced-order models, which need information describing when liquid breaks up.

3.7. Computational cost

The extraction tool is presented as an addition to atomization simulations rather than a separate post-processing method. The purpose is to (1) provide access to information that would otherwise be impossible to obtain via traditional post-processing, and (2) extract the information efficiently. Thus, the added computational cost of incorporating the tool into an already computationally intensive simulation is an important consideration. The tool assigns $\mathcal{S}$, transports $\mathcal{L}$, identifies breakup and coalescence, and outputs data. We conducted a timing study using 175 timesteps at the end of the Diesel jet simulation. Since the tool becomes more expensive as more liquid structures are created and as more breakup and coalescence occur, this analysis represents the most expensive iterations of the tool in the present simulation. Table 3 displays the average percent of each timestep used by the four main processes of the tool. $\mathcal{L}$ transport is incorporated into the VOF scheme and the existing flux calculation. The cost of the $\mathcal{L}$ flux is negligible and only requires 0.01% of the total timestep time. Multiple $\mathcal{L}$ values can exist within a single flux, which does require a minor amount of memory. The identification of liquid structures and assignment of $\mathcal{S}$ values was not trivial and required 2.27% of each timestep. This is due to the cost of the band-growth algorithm that identifies each unique structure in the domain. At this stage in the simulation, it is identifying ~15,000 structures per timestep. Other, more efficient, connected component labeling algorithms could be used in place of this to further reduce the computational cost. The breakup and coalescence

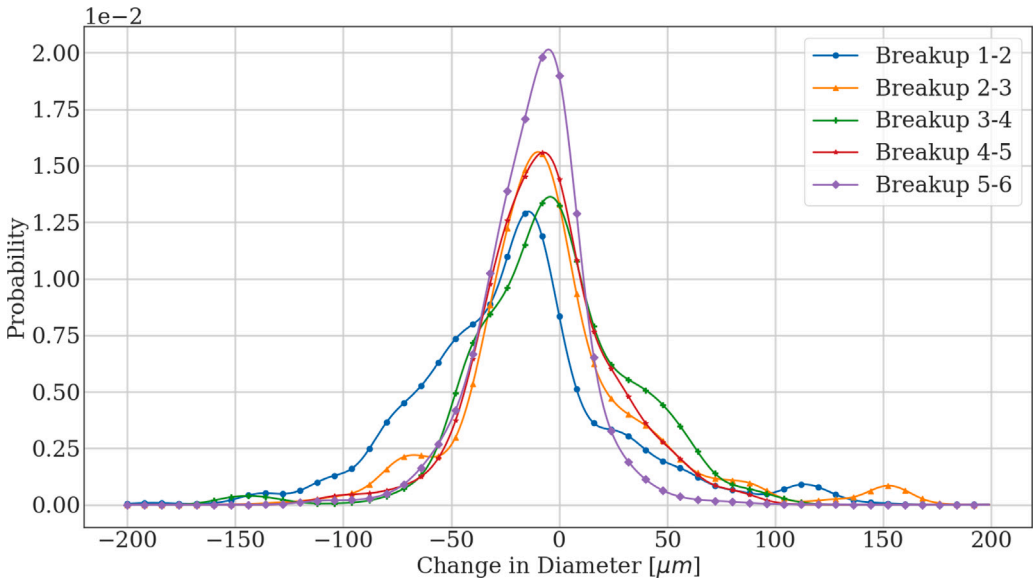

Fig. 9. A probability density function displaying the change in diameter between breakup events. Positive values indicate that coalescence occurred between the breakup events indicated in the legend.

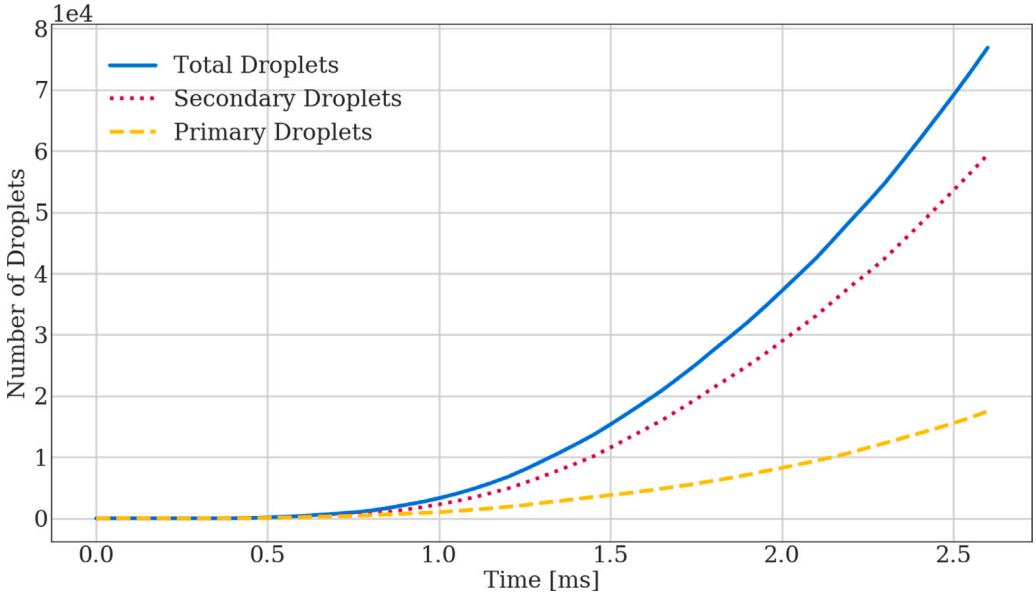

Fig. 10. The cumulative number of droplets as a function of time is shown with the solid blue line. This line is broken out into primary and secondary droplets shown as dashed yellow and dotted red lines, respectively. Note: secondary droplets in this image represent the classical definition of secondary droplets (i.e. all breakup following primary).

event identifications involve simple list comparisons and only cost 0.28% of the timestep. Data extraction is not costly, only using 0.001% of each timestep. This is reasonable because it simply involves writing lines to a CSV file. The total cost for the entire tool was 2.56% of the total timestep in this simulation. This cost can be reduced by improving the $\mathcal{S}$ assignment algorithm.

Table 3
The cost of each process within the tool. Displayed as the average percent of time per timestep. Sampled from 175 timesteps at the end of the simulation.

Process	Percent of timestep [%]
$\mathcal{S}$ Assignment	2.27
$\mathcal{L}$ Transport	0.01
Event Identification	0.28
Data Extraction	0.001

4. Conclusions

An extraction tool for numerical simulations of atomization was developed and tested. Improvements to the methodology originally proposed by Rubel and Owkes [39] 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. Improvements were made through an extensive rewrite of algorithms within the tool. Two major

changes were made. The first was the incorporation of a key logical statement in the portion of the code that identifies independent liquid structures. This prevents non-physical coalescence when liquid structures exist in adjacent computational cells. Additionally, the criteria for identifying coalescence were updated by improving the transport and tracking scheme for identification numbers ($\mathcal{L}$), which move with the liquid. Both of these changes together remedied the identification of fictitious breakup and coalescence events, present in the previous work. In addition, the updated tool now samples data from both the timestep preceding events as well as the timestep following them. This provides insights into the conditions which led to breakup as well as the resultant structures from breakup or coalescence.

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. New analyses of the local flow field, the breakup evolution of liquid structures, and coalescence dynamics were introduced. 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 affects liquid breakup.

CRediT authorship contribution statement

Brendan Christensen: Methodology, Software, Data curation, Writing – original draft, Visualization. **Mark Owkes:** Conceptualization, Methodology, Writing – review & editing, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This material is based upon work supported by the National Science Foundation, United States under Grant No. 1749779. Computational efforts were performed on the Hyalite High Performance Computing System, operated and supported by University Information Technology Research Cyberinfrastructure at Montana State University.

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 $i = 1 \rightarrow N_{\text{cell}}$ **do** ▷ Loop over cells in domain
  ▷ **Form list of $\mathcal{L}$ for this cell**
  $\mathcal{L}_{\text{count}} = 0$
  if cell has interface or VoF > 0.5 **then** ▷ Search through cells with liquid
    if $\mathcal{L} > 0$ **then**
      $\mathcal{L}_{\text{count}} \leftarrow \mathcal{L}_{\text{count}} + 1$ ▷ Keep track of the number of unique $\mathcal{L}$ values within the cell
      $\mathcal{L}_{\text{cell}}(\mathcal{L}_{\text{count}}) \leftarrow \mathcal{L}$ ▷ Keep track of $\mathcal{L}$ value in the cell
    end if
  end if
  for $f = 1 \rightarrow N_{\text{faces}}$ **do** ▷ Loop over all faces of a cell
    for $n = 1 \rightarrow \mathcal{L}_{\max}$ **do** ▷ Loop through all $\mathcal{L}$ that can flux through a face
      if $\mathcal{L}_{\text{flux},n} \neq 0$ **then** ▷ Check to see if there is an n^{th} $\mathcal{L}$ flux into the cell
        if $\mathcal{L}_{\text{flux},n} \neq \mathcal{L}_{\text{cell}}(1 : \mathcal{L}_{\text{cell}})$ **then** ▷ Check to see if $\mathcal{L}$ fluxes matches any of the current $\mathcal{L}$ in the cell
          $\mathcal{L}_{\text{count}} \leftarrow \mathcal{L}_{\text{count}} + 1$ ▷ Update $\mathcal{L}$ counter in the cell
          $\mathcal{L}_{\text{cell}}(\mathcal{L}_{\text{count}}) \leftarrow \mathcal{L}_{\text{flux}}$ ▷ Keep track of all $\mathcal{L}$ values in the cell
        end if
      end if
    end for
  end for
  ▷ **Check for coalescence or merge event**
  if $\mathcal{L}_{\text{count}} \geq 2$ **then** ▷ More than one unique $\mathcal{L}$ for this cell
    for $n = 2 \rightarrow \mathcal{L}_{\text{count}}$ **do**
      nMerge = nMerge+1 ▷ Keep track of the number of coalescence events
      $\mathcal{L}1(\text{nMerge}) = \min(\mathcal{L}_{\text{cell}}(1), \mathcal{L}_{\text{cell}}(n))$ ▷ Make a list of the $\mathcal{L}$'s which need to be merged
      $\mathcal{L}2(\text{nMerge}) = \max(\mathcal{L}_{\text{cell}}(1), \mathcal{L}_{\text{cell}}(n))$
    end for
  end if
end for
▷ **Compute new $\mathcal{L}$ after coalescence event**
for $n = 1 \rightarrow$ nMerge **do** ▷ Loop over the all identified coalescence events
  if $\mathcal{L}1(n) = \mathcal{L}1(n+1)$ **then** ▷ Combine identical events
    if $\mathcal{L}2(n) = \mathcal{L}2(n+1)$ **then**
      $\mathcal{L}_{new} \leftarrow min(\mathcal{L}1, \mathcal{L}2)$ ▷ Assign the smaller $\mathcal{L}$ to the new merged structure
      $\mathcal{L}_{old} \leftarrow max(\mathcal{L}1, \mathcal{L}2)$
    end if
  end if
end for
▷ **Update $\mathcal{L}$ in domain**
where $(\mathcal{L} = \mathcal{L}_{old})$: $\mathcal{L} \leftarrow \mathcal{L}_{new}$ ▷ Update $\mathcal{L}$'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)
```

References

[1] Zhao F, Lai M-C, Harrington DL. Automotive spark-ignited direct-injection gasoline engines. Prog Energy Combust Sci 1999;25:437–562.

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

[3] Maa Yuh-Fun, Nguyen Phuong-Anh, Sweeney Theresa, Shire Steven, Hsu Chung. Protein inhalation powders: spray drying vs spray freeze drying. Pharm Res 1999;16.

[4] Fuentes E, Coe H, Green D, De Leeuw G, Mcfiggans G. Laboratory-generated primary marine aerosol via bubble-bursting and atomization. Atmos Meas Tech 2010;3:141–62.

[5] Savart Felix. Mémoire sur la constitution des veines liquides lancées par des orifices circulaires en mince paroi. Ann Chim de Phys 1833;53:337–86.

[6] Rayleigh Lord. On the instability of jets. Proc Lond Math Soc 1878.

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

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

[9] Slangen Pierre R, Lauret Pierre, Heymes Frederic, Aprin Laurent, Lecysyn Nicolas. High-speed imaging optical techniques for shockwave and droplets atomization analysis. Opt Eng 2016;55:121706.

[10] Wang YJ, Im Kyoung Su, Fezzaa K, Lee WK, Wang Jin, Micheli P, Laub C. Quantitative x-ray phase-contrast imaging of air-assisted water sprays with high Weber numbers. Appl Phys Lett 2006;89.

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

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

[13] Gorokhovski Mikhael, Herrmann Marcus. Modeling primary atomization. Annu Rev Fluid Mech 2008;40:343–66.

[14] Shinjo J, Umemura A. Simulation of liquid jet primary breakup: Dynamics of ligament and droplet formation. Int J Multiph Flow 2010;36:513–32.
[15] Desjardins O, Pitsch H. Detailed numerical investigation of turbulent atomization of liquid jets. Atomization Sprays 2010;20:311–36.
[16] Shinjo J, Umemura A. Surface instability and primary atomization characteristics of straight liquid jet sprays. Int J Multiph Flow 2011;37:1294–304.
[17] Jarrahbashi D, Sirignano WA. Invited Article: Vorticity dynamics for transient high-pressure liquid injection. Phys Fluids 2014;26.
[18] Watanabe Tomoaki, Sakai Yasuhiko, Nagata Kouji, Ito Yasumasa, Hayase Toshiyuki. Vortex stretching and compression near the turbulent/non-turbulent interface in a planar jet. J Fluid Mech 2014;758:754–85.
[19] Jarrahbashi D, Sirignano WA, Popov PP, Hussain F. Early spray development at high gas density: Hole, ligament and bridge formations. J Fluid Mech 2016;792:186–231.
[20] Li Xiaoyi, Soteriou Marios C. High fidelity simulation and analysis of liquid jet atomization in a gaseous crossflow at intermediate weber numbers. Phys Fluids 2016;28.
[21] Zandian A, Sirignano WA, Hussain F. Planar liquid jet: Early deformation and atomization cascades. Phys Fluids 2017;29.
[22] Zandian A, Sirignano WA, Hussain F. Understanding liquid-jet atomization cascades via vortex dynamics. J Fluid Mech 2018;843:293–354.
[23] Reitz Rolf D, Diwakar R. Structure of high-pressure fuel sprays. SAE Tech Pap 1987.
[24] Beale Jennifer C, Reitz Rolf D. Modeling spray atomization with the Kelvin-Helmholtz/Rayleigh-Taylor hybrid model. Atomization Sprays 1999;9.
[25] Fuster D, Matas JP, Marty S, Popinet S, Hoepffner J, Cartellier A, et al. Instability regimes in the primary breakup region of planar coflowing sheets. J Fluid Mech 2013;736:150–76.
[26] Deshpande Suraj S, Gurjar Soumil R, Trujillo Mario F. A computational study of an atomizing liquid sheet. Phys Fluids 2015;27:082108.
[27] Agarwal Arpit, Trujillo Mario F. A closer look at linear stability theory in modeling spray atomization. Int J Multiph Flow 2018;109:1–13.
[28] Krolick Will, Owkes Mark. Primary atomization instability extraction using dynamic mode decomposition. Atomization Sprays 2018;28:1061–79.
[29] Hasslberger Josef, Ketterl Sebastian, Klein Markus, Chakraborty Nilanjan. Flow topologies in primary atomization of liquid jets: A direct numerical simulation analysis. J Fluid Mech 2019;859:819–38.
[30] Torregrosa Antonio J, Payri Raúl, Salvador F Javier, Crialesi-Esposito Marco. Study of turbulence in atomizing liquid jets. Int J Multiph Flow 2020;129.
[31] Salvador FJ, Ruiz S, Crialesi-Esposito Marco, Blanquer Ignacio. Analysis on the effects of turbulent inflow conditions on spray primary atomization in the near-field by direct numerical simulation. Int J Multiph Flow 2018;102:49–63.
[32] O'Rourke Peter J, Amsden Anthony A. The tab method for numerical calculation of spray droplet breakup. SAE Tech Pap 1987.
[33] Pilch M, Erdman CA. 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. Int J Multiph Flow 1987;13:741–57.
[34] Patterson Mark A, Reitz Rolf D. Modeling the effects of fuel spray characteristics on diesel engine combustion and emission. SAE Tech Pap 1998.
[35] Wang Yue, Lee Won Geun, Reitz Rolf D, Diwakar Ramachandra. Numerical simulation of diesel sprays using an eulerian-lagrangian spray and atomization (ELSA) model coupled with nozzle flow. SAE Tech Pap 2011.
[36] Wang C, Chang S, Wu H, Xu J. Modeling of drop breakup in the bag breakup regime. Appl Phys Lett 2014;104.
[37] Wang C, Chang S, Wu H, Ding L, Thompson JM. Theoretical modeling of spray drop deformation and breakup in the multimode breakup regime. Atomization Sprays 2015;25:857–69.
[38] Sula C, Grosshans H, Papalexandris MV. Assessment of droplet breakup models for spray flow simulations. Flow Turbul Combust 2020;105:889–914.
[39] Rubel Clark, Owkes Mark. Extraction of droplet genealogies from high-fidelity atomization simulations. Atomization Sprays 2019;29:709–39.
[40] Desjardins Olivier, Blanquart Guillaume, Balarac Guillaume, Pitsch Heinz. High order conservative finite difference scheme for variable density low Mach number turbulent flows. J Comput Phys 2008;227:7125–59.
[41] Desjardins Olivier, Moureau Vincent, Pitsch Heinz. An accurate conservative level set/ghost fluid method for simulating turbulent atomization. J Comput Phys 2008;227:8395–416.
[42] Owkes Mark, Desjardins Olivier. A computational framework for conservative, three-dimensional, unsplit, geometric transport with application to the volume-of-fluid (VOF) method. J Comput Phys 2014;270:587–612.
[43] Owkes Mark, Desjardins Olivier. A mass and momentum conserving unsplit semi-Lagrangian framework for simulating multiphase flows. J Comput Phys 2017;332:21–46.
[44] Choi Haecheon, Moin Parviz. Effects of the computational time step on numerical solutions of turbulent flow. J Comput Phys 1994;113.
[45] Youngs David L. In: Morton KW, Baines MJ, editors. Numerical methods in fluid dynamics. Academic Press; 1982, p. 273–85, chapter Time-Dependent Multi-material Flow with Large Fluid Distortion.
[46] Pilliod James Edward, Puckett Elbridge Gerry. Second-order accurate volume-of-fluid algorithms for tracking material interfaces. J Comput Phys 2004;199:465–502.
[47] Fedkiw Ronald P, Aslam Tariq, Merriman Barry, Osher Stanley. A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). J Comput Phys 1999;152.
[48] Dendy JE. Black box multigrid. J Comput Phys 1982;48.
[49] Owkes Mark, Cauble Eric, Senecal Jacob, Currie Robert A. Importance of curvature evaluation scale for predictive simulations of dynamic gas–liquid interfaces. J Comput Phys 2018;365:37–55.
[50] Herrmann M. A parallel Eulerian interface tracking/Lagrangian point particle multi-scale coupling procedure. J Comput Phys 2010;229:745–59.
[51] Hendrickson Kelli, Weymouth Gabriel D, Yue Dick KP. Informed component label algorithm for robust identification of connected components with volume-of-fluid method. Comput & Fluids 2020;197.
[52] Hoshen J, Kopelman R. Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm. Phys Rev B 1976;14.
[53] Childs Hank, Brugger Eric, Whitlock Brad, Meredith Jeremy, Ahern Sean, Pugmire David, et al. VisIt: an end-user tool for visualizing and analyzing very large data. In: High performance visualization–enabling extreme-scale scientific insight. 2012, p. 357–72.
[54] Prakash Surya, Jain Suhas S, Lovett Jeffery A, Raghunandan B N, Ravikrishna R V, Tomar Gaurav. Detailed numerical simulations of atomization of a liquid jet in a swirling gas crossflow. Atomization Sprays 2019;1–28.