Computer Physics Communications 266 (2021) 107396

Contents lists available at ScienceDirect

Computer Physics Communications

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

MFC: An open-source high-order multi-component, multi-phase, and multi-scale compressible flow solver

Spencer H. Bryngelson [a,*], Kevin Schmidmayer [a], Vedran Coralic [a], Jomela C. Meng [a], Kazuki Maeda [b], Tim Colonius [a]

[a] Division of Engineering and Applied Science, California Institute of Technology, Pasadena, CA 91125, USA
[b] Center for Turbulence Research, Stanford University, Stanford, CA 94305, USA

ARTICLE INFO

Article history:
Received 25 July 2019
Received in revised form 9 April 2020
Accepted 29 April 2020
Available online 23 May 2020

Keywords:
Computational fluid dynamics
Multi-phase flow
Diffuse-interface method
Compressible flow
Ensemble averaging
Bubble dynamics

ABSTRACT

MFC is an open-source tool for solving multi-component, multi-phase, and bubbly compressible flows. It is capable of efficiently solving a wide range of flows, including droplet atomization, shock–bubble interaction, and bubble dynamics. We present the 5- and 6-equation thermodynamically-consistent diffuse-interface models we use to handle such flows, which are coupled to high-order interface-capturing methods, HLL-type Riemann solvers, and TVD time-integration schemes that are capable of simulating unsteady flows with strong shocks. The numerical methods are implemented in a flexible, modular framework that is amenable to future development. The methods we employ are validated via comparisons to experimental results for shock–bubble, shock–droplet, and shock–water–cylinder interaction problems and verified to be free of spurious oscillations for material-interface advection and gas–liquid Riemann problems. For smooth solutions, such as the advection of an isentropic vortex, the methods are verified to be high-order accurate. Illustrative examples involving shock–bubble–vessel-wall and acoustic–bubble–net interactions are used to demonstrate the full capabilities of MFC.

Program summary

Program title: MFC (Multi-component Flow Code)
CPC Library link to program files: http://dx.doi.org/10.17632/8y55zscjd3.1
Developer's repository link: https://mfc-caltech.github.io
Licensing provisions: GNU General Public License 3
Programming language: Fortran 90 and Python
Nature of problem: Computer simulation of multi-component flows requires careful physical model selection and sophisticated treatment of spatial and temporal derivatives to keep solutions both thermodynamically consistent and free of spurious oscillations. Further, such methods should be high-order accurate for smooth solutions to reduce computational cost and promote sharper interfaces for discontinuous ones. These problems are particularly challenging for flows with material interfaces, which are important in numerous applications.
Solution method: The present software incorporates multiple physical models and numerical schemes for treatment of compressible multi-phase and multi-component flows. Additional physical effects and sub-grid models are included, such as an ensemble-averaged bubbly flow model. The architecture was designed to ensure that further development is straightforward.

1. Introduction

The multi-component flow code (MFC) is an open source high-order solver for multi-phase and multi-component flows. Such flows are central to a wide range of engineering problems. For example, cavitating flow phenomena are of critical importance to the development of artificial heart valves and pumps [1], minimizing injury due to blast trauma [2,3], and improving shock- and burst-wave lithotripsy treatments [4–6]. Bubble cavitation is also pervasive in flows around hydrofoils, submarines, and high-velocity projectiles [7–9], during underwater explosions [10,11], and within pipe systems and hydraulic machinery [12,13]. Unfortunately, cavitation in these settings is usually detrimental, causing noise and material deterioration [12,13]. Other cases of interest include the breakup of liquid droplets and jets [14–16], erosion of aircraft surfaces during supersonic flight [17,18],

* Corresponding author.
E-mail addresses: spencer@caltech.edu (S.H. Bryngelson), colonius@caltech.edu (T. Colonius).

https://doi.org/10.1016/j.cpc.2020.107396
0010-4655/

shock-wave attenuation of nuclear blasts [19], and needle-free injection for drug delivery [20,21].

Robust simulation of multi-component compressible flow phenomena requires a numerical method that maintains discrete conservation, suppresses oscillations near discontinuities, and preserves numerical stability. For such simulations to be computationally efficient, the method used should also be high-order accurate away from discontinuities. Schemes that can potentially achieve these requirements can be classified as either interface-tracking or interface-capturing [22]. Examples of interface-tracking methods include free-Lagrange [23,24], front-tracking [25,26], and level-set/ghost fluid schemes [27–33]; these methods differ from interface-capturing as they treat material interfaces as sharp features in the flow. This allows interfacial fluids to have differing equations of state and ensures that interfacial physics are straightforward to implement. Unfortunately, they do not naturally enforce discrete or total conservation, though there have been some recent attempts to partially include these properties [34,35]. Interface-capturing methods instead treat interfaces as discontinuities in material properties via advected volume fractions. Such methods are generally more efficient than interface-tracking schemes [36] and can achieve discrete conservation by simply solving the governing equations in conservative form. However, they also smear material interfaces via numerical diffusion [37–39]. In these smeared regions, we must ensure that the mixture properties are treated in a thermodynamically- and numerically-consistent way, such that spurious oscillations (and other physical inconsistencies) are avoided in the presence of high density contrasts between materials. Fortunately, such methods have been developed and prove to be a robust treatment of interfacial dynamics [37,40,41]. Still, the lack of a coherent material interface means that interfacial physics are more challenging to implement than interface-tracking methods, though conservative treatments do exist for both interfacial heat transfer [42] and capillary effects [16,38,43].

Here, we choose to use an interface-capturing scheme because computational efficiency and discrete conservation are of principal importance to many problems of interest. The interface-capturing schemes we use follow from the so-called 5- and 6-equation models, which are known to be sufficient for representing a wide range of flow phenomenologies [42,44–46]. They are complemented by a discretely-conservative numerical method that solves the conservative form of the compressible flow equations. To maintain a non-oscillatory behavior near material interfaces, the material advection equations are formulated in a quasi-conservative form [47]. The equations of motion are then closed by a thermodynamically consistent set of mixture rules [44]. The governing equations are solved with a shock-capturing finite-volume method. Specifically, we adopt a WENO spatial reconstruction that can achieve high-order accuracy while maintaining non-oscillatory behavior near material interfaces [41, 48–50]. This scheme is then coupled with an HLL-type approximate Riemann solver [51,52] and a total-variation-diminishing (TVD) time stepper [53].

MFC is, of course, not the only viable option for simulation of compressible multi-phase flows. For example, ECOGEN [54] offers a pressure-disequilibrium-based interface-capturing scheme that is well suited for the same problems reached by MFC. However, ECOGEN is built upon an intrinsically low-order MUSCL scheme that can inhibit both efficient simulation and physically fidelity when compared to the WENO schemes we use here, especially when augmented with our high-order cell-average approximations. This is demonstrated in Section 5.4 for an isentropic vortex problem and in Section 5.3 and Schmidmayer et al. [55] for cavitating gas bubbles. In pursuit of this, MFC was also constructed with a phase-averaged flow model that represent unresolved multi-phase dynamics at the sub-grid level. More mature CFD solvers such as OpenFOAM are also available. OpenFOAM natively supports finite-volume methods for multi-component flows [56], though higher-order methods and interface-capturing models are only available via links to external forked projects. MFC instead offers an integrated approach that avoids any conflicts from such libraries. Finally, the parallel I/O file systems we employ ensures that the MFC architecture can scale up to the largest modern HPC systems. Note that MFC suffers from some of the same limitations as many compressible flow solvers. For example, time step restrictions can be prohibitive for low-Mach-number flows, such as those over ship screw propellers. Additionally, satisfaction of the closure assumptions of the bubbly-flow model must also be checked. Indeed, cases involving sheet cavitation will likely violate this closure.

Herein, we describe version 1.0 of MFC. In Section 2 we present an overview of what is included in the MFC package, including its organization, features, and logistics. The physical models we use are presented in Section 3, including the associated mixture rules, governing equations, equations of state, and our implementation of an ensemble-averaged bubbly flow model. The numerical methods used to solve the associated equations are presented in Section 4. A series of test cases simulated using MFC are discussed in Section 5; these verify and validate our method. These are complemented by a set of illustrative example cases that further demonstrate MFC's capabilities in Section 6 and parallel benchmarking in Section 7, which analyzes the performance of MFC on large scale computing clusters. Section 8 concludes our presentation of MFC.

2. Overview and features

2.1. Package, installation, and testing

MFC is available at `https://mfc-caltech.github.io`. Its source code is written in Fortran 90 with MPI bindings for parallel communication. Python scripts are used to generate input files. Installation of MFC requires the FFTW package for cylindrical coordinate treatment [57], and, optionally, Silo and its dependencies for post-treatment of data files [58]. We only provide the FFTW package with MFC to keep the package size relatively small.

The MFC package includes several directories; their organization and descriptions are shown in Table 1. New users should consult the `CONFIGURE` and `INSTALL` files for instructions on how to compile MFC. In brief, the user must ensure that Python and an MPI Fortran compiler are loaded and that the FFTW package can be located by `Makefile`; this can be done by pointing `Makefile.user` to the correct location or by installing the included FFTW distribution in the `installers` directory. Once the software has been built, the `test` target of `Makefile` should be called; it runs multiple tests (which are located in the tests directory) to ensure that MFC is operating as intended.

2.2. Features

MFC uses a fully parallel environment via message passing interfaces (MPI), the performance of which is the subject of Section 7. Computationally, it includes structured Cartesian and cylindrical grids with non-uniform mesh stretching available; characteristic-based Thompson, periodic, and free-slip boundary conditions have also been implemented. The 5- and 6-equation flow models can be used with a flexible number of components, as discussed in Section 3. Ensemble-averaged dilute bubbly flow modeling is also available, including options for Gilmore and Keller–Miksis single-bubble models (see Section 3.3). The nu-

Table 1
Descriptions of the files and directories included in MFC. [component] is one of pre_process, simulation, or post_process.

Name	Description
example_cases/	Example and demonstration cases
doc/	Documentation
installers/	Package installers: Includes FFTW
lib/	Libraries
src/	Source code
_____ master_scripts/	Python modules and dictionaries, optional source files
_____ pre_process_code/	Generates initial conditions and grids
_____ simulation_code/	Flow solver
_____ post_process_code/	Processes simulation data
tests/	Test cases to ensure software is operating as intended
AUTHORS	List of contributors and their contact information
CONFIGURE	Package configuration guide
COPYRIGHT	Copyright notice
INSTALL	Installation guide
LICENSE	The GNU public license file
Makefile.in	Makefile input; generally does not need to be modified
Makefile.user	User inputs for compilation; requires attention from user
Makefile	Targets: all (default), [component], test, clean
RELEASE	Release notes

merical methods are discussed in Section 4; they include 1-, 3-, and 5th order accurate WENO reconstructions on optionally the primitive, conservative, or characteristic variables. Within each finite volume, high-order evaluation is available for the cell-averaged variables via Gaussian quadrature for multi-dimensional problems. The shock-capturing schemes are paired with either HLL, HLLC, or exact Riemann solvers. For time-stepping, 1–5th order accurate Runge–Kutta methods are available. These features will, of course, evolve and expand with time.

Of great practical importance are the user interfaces we utilize. MFC features Python input scripts, which operate via dictionaries to automatically write input files that are read via Fortran namelists. Additionally, the file system and data formats were selected to enable large-scale parallel simulations. Specifically, we use the Luster file system to generate and read restart files; it can support gigabyte-per-second-scale IO operations and petabyte-scale storage requirements [59], which ensures that the MFC can utilize the full capabilities of modern HPC systems. We also utilize HDF5 Silo databases, which keeps the file structure compact and enables parallel visualization. We have confirmed that MFC works as expected on various high-performance computing platforms, including modern SGI- and Dell-based supercomputers.

2.3. Software structure

2.3.1. Pre-processing

The pre-processor generates initial conditions and spatial grids from the physical patches specified in the Python input file and exports them as binary files to be read by the simulator. Specifically, this involves allocating and writing either a Cartesian or cylindrical mesh, with the option of mesh stretching, according to the input parameters. The specified physical variables for each patch are transformed into their conservative form and written in a manner consistent with the mesh. The pre-processor is comprised of individual Fortran modules that read input values and export mesh and initial condition files, assign then distribute global variables via MPI, perform variable transformations, generate grids, parse and assign patch types, and check that specified input variables are physically consistent and that specified options do not contradict each other.

2.3.2. Simulation

The simulator, given the initial-condition files generated by the pre-processor, solves the corresponding governing flow equations with the specified boundary conditions using our interface-capturing numerical method. Simulations are conducted for the number of time steps indicated. The simulator exports run-time information, restarts files that can be used to either restart the simulation or post-process the associated data, and, optionally, human-readable output data. The structure of the simulator follows that of pre-processor, with individual Fortran modules conducting each software component; this includes reading and exporting data and grid files, performing Fourier transforms, assigning and distributing global variables via MPI, performing variable transformations, computing time and spatial derivatives using WENO and the Riemann solver specified, computing boundary values, including ensemble-averaged bubbly flow physics, and checking that the input variables are valid.

2.3.3. Post-processing

The post-processor reads simulation data and exports HDF5/Silo databases that include variables and derived variables, as specified in the input file. Since the simulator can export human-readable data, post-processing is not essential for the usage of MFC, but is a useful tool, especially for large or parallel data structures. Specifically, the post-process component of the MFC reads the restart files exported by the simulator at distinct time intervals and computes the necessary derived quantities. The HDF5 database is then generated and exported, and can be readily viewed using, for example, VisIt [60] or Paraview [61]. Again, individual Fortran modules perform the associated tasks, including reading data, parameter conversion, assigning and distributing global MPI variables, computing Fourier transforms, exporting HDF5 Silo databases, and checking that the input parameters are consistent.

2.4. Description of input/output files

We next describe the contents of a case-specific directory and its logistics. The specific file structure is shown in Table 2. The Python script `input.py` is used to generate the input files (∗.inp) for the source codes and execute an MFC component (one of pre_process, simulation, or post_process) either in the active window or as a submitted batch script. This Python file contains the input parameters available for the MFC. MFC, depending upon the component used and options selected, will generate several files and directories. If enabled, the `run_time.inf` file will be generated by the simulator and includes details about the current

Table 2
Input/output files in the case-specific directory.

	Name	Description
Input	input.py	Input parameters
	_____ pre_process.inp	Pre-process input parameters, auto-generated
	_____ simulation.inp	Simulation input parameters, auto-generated
	_____ post_process.inp	Post-process input parameters, auto-generated
Output	run_time.inf	Run-time information including current simulation time and CFL
	D/	Formatted simulation output files
	p_all/	Binary simulation restart files (depending upon options used)
	restart_data/	Luster restart files (depending upon options used)
	silo_HDF5/	Silo post-process files (depending upon options used)
	binary/	Binary post-process files (depending upon options used)

time step, simulation time, and stability criterion. Directories that contain binary restart data and output files for visualization or further post-treatment are generated, again depending upon the specified options, by the simulator and post-processor.

3. Physical model and governing equations

The mechanical-equilibrium compressible multi-component flow models we use can be written as

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{q}) + \boldsymbol{h}(\boldsymbol{q}) \nabla \cdot \boldsymbol{u} = \boldsymbol{s}(\boldsymbol{q}), \tag{1}$$

where $\boldsymbol{q}$ is the state vector, $\boldsymbol{F}$ is the flux tensor, $\boldsymbol{u}$ is the velocity field, and $\boldsymbol{h}$ and $\boldsymbol{s}$ are non-conservative quantities we describe subsequently.

3.1. 5-equation model

We first introduce our implementation of the thermodynamically consistent mechanical-equilibrium model of Kapila et al. [45]. Our multi-component implementation can be used for N_k components, though we present a two-component ($N_k = 2$) configuration here for demonstration purposes. It consists of five partial differential equations as

$$\boldsymbol{q} = \begin{bmatrix} \alpha_1 \\ \alpha_1\rho_1 \\ \alpha_2\rho_2 \\ \rho\boldsymbol{u} \\ \rho E \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \alpha_1\boldsymbol{u} \\ \alpha_1\rho_1\boldsymbol{u} \\ \alpha_2\rho_2\boldsymbol{u} \\ \rho\boldsymbol{u}\boldsymbol{u} + p\boldsymbol{I} - \boldsymbol{T} \\ (\rho E + p)\boldsymbol{u} - \boldsymbol{T}\cdot\boldsymbol{u} \end{bmatrix}, \quad \boldsymbol{h} = \begin{bmatrix} -\alpha_1 - K \\ 0 \\ 0 \\ \boldsymbol{0} \\ 0 \end{bmatrix}, \quad \boldsymbol{s} = \boldsymbol{0}, \tag{2}$$

where ρ, $\boldsymbol{u}$, and p are the mixture density, velocity, and pressure, respectively, α_k is the volume fraction of component k, and $\boldsymbol{T}$ is the viscous stress tensor (with no bulk stresses)

$$\boldsymbol{T} = 2\eta\left(\boldsymbol{D} - \frac{1}{3}(\nabla\cdot\boldsymbol{u})\boldsymbol{I}\right), \tag{3}$$

where η is the mixture shear viscosity and

$$\boldsymbol{D} = \frac{1}{2}\left(\nabla\boldsymbol{u} + (\nabla\boldsymbol{u})^{\top}\right) \tag{4}$$

is the strain rate tensor. The mixture total and internal energies are $E = e + \|\boldsymbol{u}\|^2/2$ and

$$e = \sum_{k=1}^{N_k} Y_k e_k(\rho_k, p), \tag{5}$$

respectively, where $Y_k = \alpha_k\rho_k/\rho$ are the mass fractions of each component. We close (5) using the stiffened-gas equation of state, which is chosen for its ability to faithfully model both liquids and gases [62]; for component k it is

$$p_k = (\gamma_k - 1)\rho_k e_k - \gamma_k\pi_{\infty,k}, \tag{6}$$

where γ is the specific heat ratio and π_∞ is the liquid stiffness (gases have $\pi_\infty = 0$) [63]. For liquids, these are usually interpreted as fitted parameters from shockwave-Hugoniot data [64,65]. The speed of sound of each component is then

$$c_k = \sqrt{\frac{\gamma_k(p_k + \pi_{\infty,k})}{\rho_k}}. \tag{7}$$

The K term in $\boldsymbol{h}$ of (2) represents expansion and compression in mixture regions. For an $N_k = 2$ configuration it is

$$K = \frac{\rho_2 c_2^2 - \rho_1 c_1^2}{\frac{\rho_2 c_2^2}{\alpha_2} + \frac{\rho_1 c_1^2}{\alpha_1}} \tag{8}$$

and the mixture speed of sound c follows as the so-called Wood speed of sound [66,67]

$$\frac{1}{\rho c^2} = \sum_{k=1}^{N_k} \frac{\alpha_k}{\rho_k c_k^2}. \tag{9}$$

Ultimately, the equations are closed by the usual set of mixture rules

$$1 = \sum_{k=1}^{N_k}\alpha_k, \quad \rho = \sum_{k=1}^{N_k}\alpha_k\rho_k, \quad \rho e = \sum_{k=1}^{N_k}\alpha_k\rho_k e_k, \quad \text{and} \quad \eta = \sum_{k=1}^{N_k}\alpha_k\eta_k. \tag{10}$$

We note that the models of Allaire et al. [44] and Massoni et al. [42] do not include the K term in (2) and thus do not strictly obey the second-law of thermodynamics, nor reproduce the correct mixture speed of sound (9). While MFC also supports these models, accurately representing the sound speed is known to be important for some problems, such as the cavitation of gas bubbles [55]. However, it is also known that the K term can result in numerical instabilities for problems with strong compression or expansion in mixture regions due to its non-conservative nature [55]. Thus, the decision of what 5-equation model to use (if any) is problem dependent and left to the user.

3.2. 6-equation model

While the 5-equation model described in Section 3.1 is efficient and represents the correct physics, the $K\nabla\cdot\boldsymbol{u}$ term that makes the model thermodynamically consistent can sometimes introduce numerical instabilities [46,55]. In such cases, a pressure-disequilibrium is preferable [55]. We also support the

6-equation pressure-disequilibrium model of Saurel et al. [46], which for a two-component configuration is expressed as

$$\boldsymbol{q} = \begin{bmatrix} \alpha_1 \\ \alpha_1\rho_1 \\ \alpha_2\rho_2 \\ \rho\boldsymbol{u} \\ \alpha_1\rho_1 e_1 \\ \alpha_2\rho_2 e_2 \end{bmatrix}, \quad \boldsymbol{F} = \begin{bmatrix} \alpha_1\boldsymbol{u} \\ \alpha_1\rho_1\boldsymbol{u} \\ \alpha_2\rho_2\boldsymbol{u} \\ \rho\boldsymbol{u}\boldsymbol{u} + p\boldsymbol{I} - \boldsymbol{T} \\ \alpha_1\rho_1 e_1\boldsymbol{u} \\ \alpha_2\rho_2 e_2\boldsymbol{u} \end{bmatrix}, \tag{11}$$

$$\boldsymbol{h} = \begin{bmatrix} -\alpha_1 \\ 0 \\ 0 \\ \boldsymbol{0} \\ \alpha_1 p_1 \\ \alpha_2 p_2 \end{bmatrix}, \quad \boldsymbol{s} = \begin{bmatrix} \mu\delta p \\ 0 \\ 0 \\ \boldsymbol{0} \\ -\mu p_I \delta p - \alpha_1 \boldsymbol{T}_1 : \nabla\boldsymbol{u} \\ \mu p_I \delta p - \alpha_2 \boldsymbol{T}_2 : \nabla\boldsymbol{u} \end{bmatrix},$$

where $\boldsymbol{T}_k$ are the component-specific viscous stress tensors and the other terms of $\boldsymbol{s}$ represent the relaxation of pressures between components with coefficient μ. The interfacial pressure is

$$p_I = \frac{z_2 p_1 + z_1 p_2}{z_1 + z_2}, \tag{12}$$

where $z_k = \rho_k c_k$ is the acoustic impedance of component k and

$$\delta p = p_1 - p_2, \tag{13}$$

is the pressure difference. Since $p_1 \neq p_2$, the total energy equation of the mixture is replaced by the internal-energy equation for each component. The mixture speed of sound is defined according to

$$c^2 = \sum_{k=1}^{2} Y_k c_k^2, \tag{14}$$

though after applying the numerical infinite pressure-relaxation procedure detailed in Section 4.6 the effective mixture speed of sound matches (9).

3.3. Bubbly flow model

3.3.1. Implementation

MFC includes support for the ensemble-phase-averaged bubbly flow model of Zhang and Prosperetti [68], and our implementation of it matches that of Bryngelson et al. [69]. The bubble population has void fraction α_b, which is assumed to be small, and the carrier components have mixture pressure p_I. The equilibrium radii of the bubble population are represented discretely as $\boldsymbol{R}_o$, which are N_{bin} bins of an assumed log-normal PDF with standard deviation σ_p [70,71]. The instantaneous bubble radii are a function of these equilibrium states as $\boldsymbol{R}(\boldsymbol{R}_o) = \{R_1, R_2, \ldots, R_{N_{\text{bin}}}\}$. The total mixture pressure is modified as

$$p = (1-\alpha_b)p_I + \alpha_b \left(\frac{\overline{\boldsymbol{R}^3 \boldsymbol{p}_{bw}}}{\overline{\boldsymbol{R}^3}} - \rho \frac{\overline{\boldsymbol{R}^3 \dot{\boldsymbol{R}}^2}}{\overline{\boldsymbol{R}^3}} \right), \tag{15}$$

where $\dot{\boldsymbol{R}}$ are the bubble radial velocities and $\boldsymbol{p}_{bw}$ are the bubble wall pressures. Overbars $\bar{\cdot}$ denote the usual moments with respect to the log-normal PDF. The bubble void fraction is advected as

$$\frac{\partial \alpha_b}{\partial t} + \boldsymbol{u}\cdot\nabla\alpha_b = 3\alpha_b \frac{\overline{\boldsymbol{R}^2\dot{\boldsymbol{R}}}}{\overline{\boldsymbol{R}^3}}, \tag{16}$$

and the bubble dynamic variables are evolved as

$$\frac{\partial n\boldsymbol{\phi}}{\partial t} + \nabla\cdot(n\boldsymbol{\phi}\boldsymbol{u}) = n\dot{\boldsymbol{\phi}}, \tag{17}$$

where $\boldsymbol{\phi} \equiv \left\{\boldsymbol{R}, \dot{\boldsymbol{R}}, \boldsymbol{p}_b, \boldsymbol{m}_v\right\}$ (see Section 3.3.2) and n is the conserved bubble number density per unit volume

$$n = \frac{3}{4\pi}\frac{\alpha_b}{\overline{\boldsymbol{R}^3}}. \tag{18}$$

3.3.2. Single-bubble dynamics

A partial differential equation following (17) is evolved for each bin representing equilibrium radius R_o. These equations assume that each bubble evolves, without interaction with its neighbors, in an otherwise uniform flow whose properties are dictated by the local mixture-averaged flow quantities [72]. We also assume that the bubbles remain spherical, maintain a uniform internal pressure, and do not break-up, or coalesce. Our model includes the thermal effects, viscous and acoustic damping, and phase change. The bubble radial accelerations $\ddot{R}$ are computed by the Keller–Miksis equation [73]:

$$R\ddot{R}\left(1-\frac{\dot{R}}{c_b}\right) + \frac{3}{2}\dot{R}^2\left(1-\frac{\dot{R}}{3c_b}\right) = \frac{p_{bw}-p_l}{\rho}\left(1+\frac{\dot{R}}{c_b}\right) + \frac{R\dot{p}_{bw}}{\rho c_b}, \tag{19}$$

where c_b is the usual speed of sound associated with the bubble and

$$p_{bw} = p_b - \frac{4\mu\dot{R}}{R} - \frac{2\sigma}{R} \tag{20}$$

is the bubble wall pressure, for which p_b is the internal bubble pressure, σ is the surface tension coefficient, and μ is the liquid viscosity. The evolution of p_b is evaluated using the model of Ando [72]:

$$\dot{p}_b = \frac{3\gamma_b}{R}\left(\Re_v T_{bw}\dot{m}_v - \dot{R}p_b + \frac{\gamma_b - 1}{\gamma_b}\lambda_{bw}\left.\frac{\partial T}{\partial r}\right|_{r=w}\right), \tag{21}$$

where T is the temperature, λ is the thermal conductivity, $\Re_v$ is the gas constant and γ_b is the specific heat ratio of the gas. Mass transfer of the bubble contents follows the reduced model of Preston et al. [74] as

$$\dot{m}_v = \frac{\mathcal{D}\rho_{bw}}{1-\chi_{vw}}\left.\frac{\partial \chi_v}{\partial r}\right|_w. \tag{22}$$

4. Solution method

Our numerical scheme generally follows that of Coralic and Colonius [41]. The spatial discretization of (1) in three-dimensional Cartesian coordinates is

$$\frac{\partial \boldsymbol{q}}{\partial t} + \frac{\boldsymbol{F}^x(\boldsymbol{q})}{\partial x} + \frac{\boldsymbol{F}^y(\boldsymbol{q})}{\partial y} + \frac{\boldsymbol{F}^z(\boldsymbol{q})}{\partial z} = \boldsymbol{s}(\boldsymbol{q}) - \boldsymbol{h}(\boldsymbol{q})\nabla\cdot\boldsymbol{u}, \tag{23}$$

where $\boldsymbol{F}^{x_i}$ are the $i \in (x,y,z)$-direction flux vectors and the treatment of $\nabla\cdot\boldsymbol{u}$ is discussed later.

4.1. Treatment of spatial derivatives

We use a finite volume method to treat the spatial derivatives of (23). The finite volumes are

$$I_{i,j,k} = [x_{i-1/2}, x_{i+1/2}] \times [y_{j-1/2}, y_{j+1/2}] \times [z_{k-1/2}, z_{k+1/2}]. \tag{24}$$

We spatially integrate (23) within each cell-centered finite volume as

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{q}_{i,j,k}}{\mathrm{d}t} =& \frac{1}{\Delta x_i}[\boldsymbol{F}^x_{i-1/2,j,k} - \boldsymbol{F}^x_{i+1/2,j,k}] + \\ & \frac{1}{\Delta y_j}[\boldsymbol{F}^y_{i,j-1/2,k} - \boldsymbol{F}^y_{i,j+1/2,k}] + \\ & \frac{1}{\Delta z_k}[\boldsymbol{F}^z_{i,j,k-1/2} - \boldsymbol{F}^z_{i,j,k+1/2}] + \boldsymbol{s}(\boldsymbol{q}_{i,j,k}) - \boldsymbol{h}(\boldsymbol{q}_{i,j,k})(\nabla\cdot\boldsymbol{u})_{i,j,k}, \end{aligned} \tag{25}$$

where

$$\boldsymbol{q}_{i,j,k} = \frac{1}{V_{i,j,k}}\iiint_{I_{i,j,k}} \boldsymbol{q}(x,y,z,t)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z, \tag{26}$$

$$\boldsymbol{s}_{i,j,k} = \frac{1}{V_{i,j,k}} \iiint_{I_{i,j,k}} \boldsymbol{s}(x,y,z,t)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z, \tag{27}$$

$$\boldsymbol{F}_{i+1/2,j,k} = \frac{1}{\Delta y_j \Delta z_k} \iint_{A_{i+1/2,j,k}} \boldsymbol{F}(x,y,z,t)\,\mathrm{d}y\mathrm{d}z, \tag{28}$$

are cell-volume and face averages, for which $\Delta x_i = x_{i+1/2} - x_{i-1/2}$ are the mesh spacings (Δy and Δz have the same form), and $V_{i,j,k}$ and $A_{i+1/2,j,k}$ are the cell volumes and face areas.

MFC can approximate this equation using high-order quadratures as opposed to simple cell-centered averages. This approach computes the flux surface integrals and source terms using a two-point, fourth-order, Gaussian quadrature rule, e.g.

$$\boldsymbol{F}_{i+1/2,j,k} = \frac{1}{4}\sum_{m=1}^{2}\sum_{l=1}^{2} \boldsymbol{F}(\boldsymbol{q}(x_{i+1/2}, y_{j_l}, z_{k_m})), \tag{29}$$

where l and m are the Gaussian quadrature point indices and

$$y_{j_l} = y_j + (2l+1)\frac{\Delta y_j}{2\sqrt{3}} \quad \text{and} \quad z_{k_m} = z_k + (2m+1)\frac{\Delta z_k}{2\sqrt{3}}. \tag{30}$$

Note that while this approach enables higher-order accuracy, it does entail greater computational expense. The divergence terms are treated using a midpoint rule

$$\begin{aligned}(\nabla\cdot\boldsymbol{u})_{i,j,k} = &\\ &\frac{1}{\Delta x_i}(u_{i+1/2,j,k} - u_{i-1/2,j,k}) + \frac{1}{\Delta y_j}(v_{i,j+1/2,k} - v_{i,j-1/2,k})\\ &+ \frac{1}{\Delta z_k}(w_{i,j,k+1/2} - w_{i,j,k-1/2}),\end{aligned} \tag{31}$$

where $\boldsymbol{u} = \{u, v, w\}$ are the cell-averaged velocity components computed analogous to (29).

To avoid spurious oscillations at material interfaces, we ultimately evaluate the fluxes by reconstructing the primitive or characteristic variables at the cell faces via a 5th-order-accurate WENO scheme [41] (though reconstruction of the conservative variables and lower-order WENO schemes are also supported). This allows us to apply a Riemann solver, so

$$\boldsymbol{F}_{i+1/2,j,k} = \frac{1}{4}\sum_{m=1}^{2}\sum_{l=1}^{2} \widehat{\boldsymbol{F}}(\boldsymbol{q}^L_{i+1/2,j,k}, \boldsymbol{q}^R_{i+1/2,j,k}), \tag{32}$$

where $\widehat{\boldsymbol{F}}$ is the numerical flux function of the Riemann solver and superscripts L and R indicate left- and right-sided WENO reconstructions. Reconstruction in the y and z directions has the same form, though the high-order cell-averaging of (29) is required to achieve higher than second-order accuracy (see Section 5.4). We use the HLLC approximate Riemann solver to compute the fluxes [52], though other Riemann solvers are also supported.

4.2. Limiters for improved numerical stability

4.2.1. Volume fraction limiting

Following our mixture rules (10), the volume fractions are physically required to sum to unity. However, the accumulation of numerical errors can preclude this if the first $N_k - 1$ volume fractions exceed unity, and thus one of the volume fractions must be negative [75]. This is of course unphysical and leads to other numerical issues, such as complex speeds of sound. To treat this, we impose the volume fraction mixture rule by limiting each volume fraction as $0 \leq \alpha_k \leq 1$ for all components k, then rescaling them as

$$\alpha_k = \frac{\alpha_k}{\sum_{k=1}^{N_k}\alpha_k} \quad \text{for } k = 1,\ldots,N_k. \tag{33}$$

We note that we only use this limiting when computing the mixture properties, and do not alter them otherwise as to avoid polluting the mass conservation properties of the method.

4.2.2. Flux limiting

The WENO schemes we utilize are, in general, not TVD. This can be problematic when WENO cannot form a smooth stencil to reconstruct, and can lead to numerical instabilities even when the usual CFL criteria are met. MFC supports advective flux limiting to treat this issue, which improves stability, though it also increases numerical dissipation and thus smearing of material interfaces. However, we use the gradient of the local volume fraction χ to minimize this effect, which localizes the limiter to non-smooth regions of the flow. In one dimension of our dimensional-splitting procedure this yields

$$\chi_i = \begin{cases} \frac{\alpha_i - \alpha_{i-1}}{\alpha_{i+1} - \alpha_i} & \text{if } u^* \geq 0,\\ \frac{\alpha_{i+2} - \alpha_{i+1}}{\alpha_{i+1} - \alpha_i} & \text{if } u^* < 0,\end{cases} \tag{34}$$

where i is the spatial index and u^* is the local velocity computed by the Riemann solver. Ultimately, the modified flux is a combination of a low- and high-order accurate flux approximation. The low-order flux is chosen to be equivalent to a first-order WENO reconstruction and the high-order flux is the Riemann flux from the WENO reconstruction. Specifically, MFC supports the minmod, MC, ospre, superbee, Sweby, van Albada, or van Leer flux limiters, each of which is a function of the volume fraction gradient χ. Further details of our numerical implementation are located in Meng [75].

4.3. Cylindrical coordinate considerations

MFC also supports the use of cylindrical coordinates. They are formulated in similar fashion to (23), though with the addition of an additional set of source terms on the right-hand-side associated with the $1/r$ cylindrical terms of the divergence operator. Our implementation was detailed by Meng [75] and does not use an alternative treatment of the WENO weights in the azimuthal direction, such as would be required to ensure high-order accuracy away from discontinuities. While such methods for this exist, they are generally complex and suffer from numerical stability issues [76–78]. The cylindrical coordinate treatment also requires velocity gradients, rather than just velocities, at cell boundaries. For this, we use a second-order-accurate finite-difference and averaging procedure to obtain the necessary velocity gradients at these points. Finally, the coordinate singularity at $r \to 0$ is treated using the method of Mohseni and Colonius [79], which performs differentiation in the radial direction via a redefinition of the radial coordinate; in our implementation, we place the singularity at the finite-volume cell boundary (rather than the center). We note that this implementation requires an even number of grid cells in the azimuthal coordinate direction. A consequence of the cylindrical coordinate system is that the grid cells near the radial center are much smaller than those far away, which restricts the global CFL criterion. Following Mohseni and Colonius [79], we address this issue by using a spectral filter, which filters the high-frequency components of the solution near the centerline and relaxes the CFL criterion. Our implementation was verified by Meng [75] to be second-order accurate away from discontinuities via a propagating spherical pressure pulse, while the construction of the viscous stress tensor was verified using the method of manufactured solutions.

4.4. One-way acoustic wave generation

The source term $\boldsymbol{s}$ of (1) and (23) can be augmented with additional terms $\boldsymbol{s}^s(t)$ for the generation of one-way acoustic waves, for examples to model an ultrasound transducer immersed in the flow. Following Maeda and Colonius [80], these take the form

$$\boldsymbol{s}^s(t) = \int_\Gamma \Omega_\Gamma(\boldsymbol{\xi}, t)\delta(\boldsymbol{X}(\boldsymbol{\xi}, t) - \boldsymbol{x})\mathrm{d}\boldsymbol{\xi} \tag{35}$$

where Ω_Γ is (possibly time dependent) forcing, Γ is the surface the forcing acts upon, δ is the Dirac delta function, and $\boldsymbol{X}$ maps the $\boldsymbol{\xi}$ coordinate to physical space $\boldsymbol{x}$. In our numerical framework this is represented at cell $I_{i,j,k}$ as

$$\boldsymbol{s}^s_{i,j,k}(t) = \sum_{m=1}^{M} \Omega_\Gamma(\xi_m, t)\delta_h(|\boldsymbol{X}(\xi_m, t) - \boldsymbol{x}_{i,j,k}|)\Delta\xi_m, \tag{36}$$

where δ_h is the discrete delta function operator, $\Delta\xi_k$ are the sizes of the discrete patch or line the forcing is applied to, and M are the number of such patches. For example, in two dimensions the one-dimensional line operator follows as

$$\delta_h(h) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{h^2}{\sigma^2}}, \tag{37}$$

where $\sigma = 3\Delta$ and Δ is the largest mesh spacing. For a single-component, two-dimensional problem (note that there is no volume-fraction advection equation in this case) the forcing is expressed as

$$\Omega_\Gamma(t) = f(t)[1/c_o, \cos(\theta), \sin(\theta), c_o^2/(\gamma - 1)], \tag{38}$$

where $f(t)$ is the time-dependent pulse amplitude, c_o is the speed of sound, and θ is the forcing direction as measured from the first-coordinate axis.

4.5. Time integration

Once the spatial derivatives have been approximated, (23) becomes a semi-discrete system of ordinary differential equations in time. We treat the temporal derivative using a Runge–Kutta time-marching scheme for the state variables. To achieve high-order accuracy and avoid spurious oscillations, we use the third-order-accurate total variation diminishing scheme of Gottlieb and Shu [53]:

$$\begin{aligned}
\boldsymbol{q}^{(1)}_{i,j,k} &= \boldsymbol{q}^n_{i,j,k} + \Delta t\boldsymbol{L}(\boldsymbol{q}^n_{i,j,k}),\\
\boldsymbol{q}^{(2)}_{i,j,k} &= \frac{3}{4}\boldsymbol{q}^n_{i,j,k} + \frac{1}{4}\boldsymbol{q}^{(1)}_{i,j,k} + \frac{1}{4}\Delta t\boldsymbol{L}(\boldsymbol{q}^{(1)}_{i,j,k}),\\
\boldsymbol{q}^{n+1}_{i,j,k} &= \frac{1}{3}\boldsymbol{q}^n_{i,j,k} + \frac{2}{3}\boldsymbol{q}^{(2)}_{i,j,k} + \frac{2}{3}\Delta t\boldsymbol{L}(\boldsymbol{q}^{(2)}_{i,j,k}),
\end{aligned} \tag{39}$$

where (1) and (2) are intermediate time-step stages, $\boldsymbol{L}$ represents the right hand side of (25), and n is the time-step index. MFC also supports Runge–Kutta schemes of orders 1–5. The time step size Δt is limited by the usual CFL criterion, which is proportional to the mixture sound speed for the 5/6-equation models or the sound speed associated with the hyperbolic portion of the ensemble-averaged flow equations as defined by Ando [72] when it is used.

4.6. Pressure-relaxation procedure

The pressure-disequilibrium model (11) requires a pressure-relaxation procedure to converge to an equilibrium pressure. We use the infinite-relaxation procedure of Saurel et al. [46]. At each time step, it solves the non-relaxed hyperbolic equations ($\mu \to 0$) using first-order-accurate explicit time step integration and a re-initialization procedure that ensures total energy conservation at the discrete level. After this, the disequilibrium pressures are relaxed as $\mu \to +\infty$. This procedure is performed at each Runge–Kutta stage, and so there is a unique pressure at the end of each stage and the 5- and 6-equation models reconstruct the same variables. As a result, simulations of the pressure-disequilibrium model are only modestly more expensive than the 5-equation models (about 5% for spherical bubble collapses [55]).

5. Simulation verification and validation

We next present several test cases that validate and verify MFC's capabilities. These include one-, two-, and three-dimensional cases that span a wide variety of flow problems.

5.1. Shock–bubble interaction

We first consider a Mach 1.22 shock wave impinging on a 5 cm diameter spherical helium bubble in air. This problem was investigated via experimental methods by Haas and Sturtevant [81] and has been previously used as a validation case for multi-component flow simulations [82–84]. Our simulations are performed using an axisymmetric configuration; further simulation specifications can be found in Coralic and Colonius [41].

Visualizations of the shock impinging the bubble and subsequent breakup and vortex ring production are shown in Fig. 1. We see that the simulation results qualitatively match those of the experiment. Importantly, no spurious oscillations can be seen in the numerical schlieren images, despite their sensitivity to small density differences. We quantitatively compare our results to the experiment by considering the velocity of key flow features: Haas and Sturtevant [81] measured the velocity of the incident, reflected, and transmitted shocks, as well as the up and down-stream interfaces and jets. Our simulation results are within 10% of the experiments for all cases and are generally within about 5% of the experimental means. Our results are also consistent with those computed independently via the level set [86] and diffuse-interface methods [87], including the Kelvin–Helmholtz instability that develops along the bubble interface (see Fig. 1(b)–(e)).

5.2. Shock–droplet/cylinder interaction

We next consider air shocks interacting with liquid media in two and three dimensions. These problems are more computationally challenging, primarily due to the larger density ratio. The first case we analyze consists of a Mach 1.47 shock impinging a 4.8 mm diameter liquid water cylinder. The simulation parameterization can be found in Meng and Colonius [15].

Fig. 2 shows a visualization of experimental and our numerical results as the shock passes over the liquid cylinder. At early times it is difficult to assess the cylinder's deformation, so we instead compare the primary and secondary waves that are generated. In Fig. 2(a) we see that the primary wave system, including the incident and reflected shock, has the same locations for both experimental and numerical results. The secondary wave system is generated when the Mach stems on both sides of the cylinder converge to the rear stagnation point; in Fig. 2(b) we see that these also match closely.

We also consider the breakup of a spherical water droplet due to a helium shock (Mach 0.59 observed in the post-shock flow), following the experimental conditions of Theofanous et al. [89]. A full exposition of the simulation conditions can be found in Meng and Colonius [14]. Fig. 3 shows their experimental image and volume fraction isosurfaces and sliced isocontours from our simulations. We use a small $\alpha_l = 0.01$ value for the isosurface of Fig. 3(b) for comparison purposes since images from experiments are often obscured by the fine mist generated. While it is challenging to obtain accurate timing data from the experiments, a qualitative agreement between experiment and simulation is still observed for the shear-induced entrainment of the droplet.

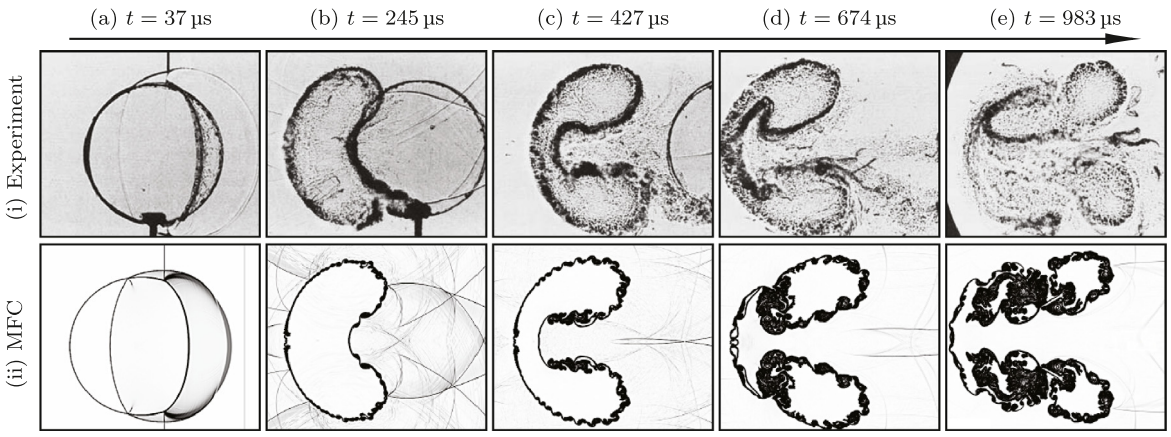


Fig. 1. Comparison between (i) experimental shadowgraphs of Haas and Sturtevant [81] and (ii) numerical schlieren visualizations [85] using MFC at select times (a)–(e) as labeled. Experimental images are ©Cambridge University Press 1987.

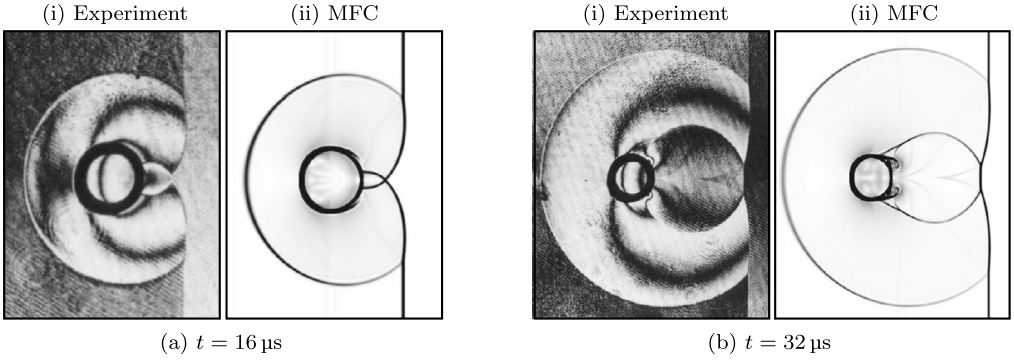


Fig. 2. Comparison between (i) holographic interferograms [88] and (ii) numerical schlieren visualizations using MFC at select times (a) and (b) as labeled. *Source:* The experimental images are reprinted from Igra and Takayama [88].

5.3. Spherical bubble dynamics

Numerical simulation of cavitating spherical gas bubbles is challenging because mixture-region compressibility must be properly treated, discrete conservation must be enforced, and sphericity should be maintained in the presence of large density and pressure ratios. We consider a collapsing and rebounding air bubble in water at 10 times higher pressure as a test of the capabilities of MFC. Specific simulation specifications were presented in Schmidmayer et al. [55]; further, we include a guided description of the simulation setup for this problem in the `example_cases/3D_sphbubcollapse` directory of the MFC package (see Table 1).

Fig. 4(a) shows the evolution of the bubble radius; it reaches a minimum near the nominal Rayleigh collapse time t_c [90], then rebounds, as expected. The radius R of our simulations is computed from the gas volume by assuming the shape is nearly spherical; this closely matches the solution expected following the Keller–Miksis equation [73]. We assess and compare the quality of our simulation with ECOGEN [54] via computation of the bubble sphericity during the collapse-rebound process. Fig. 4(b) shows this sphericity ψ of the bubble, which is defined following Wadell [91] and a value of 1 indicates a spherical bubble. We see that the WENO scheme used by MFC can better maintain sphericity than a MUSCL scheme of ECOGEN formulated for the same diffuse-interface model [55], and is thus preferable for this problem.

5.4. Isentropic and Taylor–Green vortices

We next consider two- and three-dimension vortex problems as a means to verify our solutions of the flow equations. The two-dimension problem we consider is the evolution of a steady, inviscid, isentropic, ideal-gas vortex. This problem has been used previously to assess the convergence properties of high-order WENO schemes for smooth solutions to the Euler equations

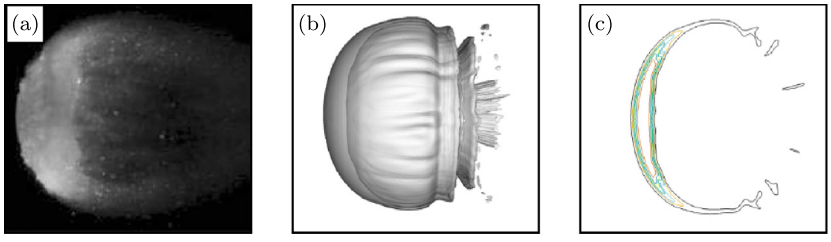


Fig. 3. Comparison of (a) experimental [89] and (b) numerical water droplets isosurface $\alpha_l = 0.01$. (c) shows isocontours of α_l ranging from 0.01 to 0.99.

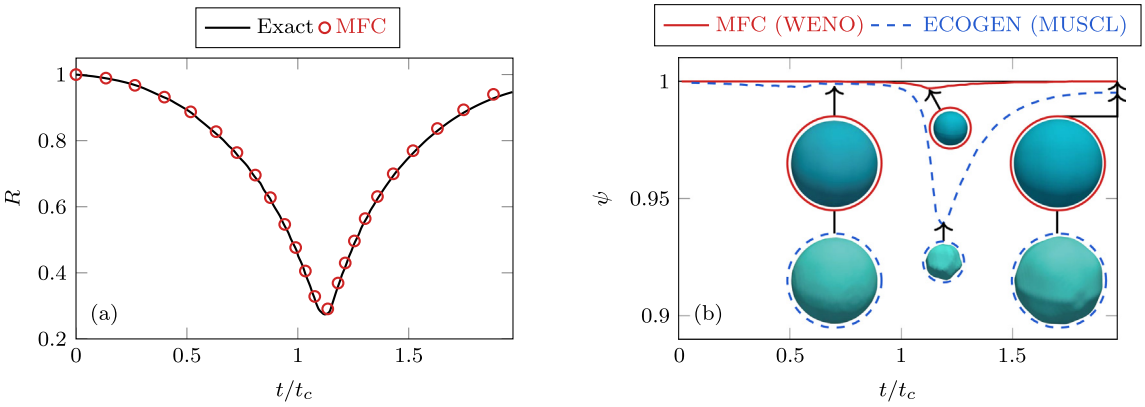


Fig. 4. Evolution of (a) the dimensionless bubble radius R and (b) its sphericity ψ. In (b) the nominal bubble shapes, represented by $\alpha_l = 0.5$ isosurfaces, are shown at select times for MFC (WENO) and ECOGEN (MUSCL).

[49,92]. We use it here to verify that we obtain high-order accuracy away from shocks and material interfaces; details regarding the numerical parameters and exact problem formulation can be found in Coralic and Colonius [41].

Fig. 5(a) shows the density error for both low- and high-order finite volume cell-averaging (following Section 4). Since the solution should be steady, the error is computed as the deviation from the initial condition after 1 dimensionless time unit as a function of the grid size, for which N is the number of finite volumes in one spatial direction. The convergence is 2nd- and 5th-order accurate for 2nd- and 4th-order-accurate cell-averaging schemes, respectively. Thus, we conclude that for multi-dimensional problems the 4th-order-accurate cell averaging we employ is required to achieve 5th-order accuracy associated with the WENO reconstructions.

We use the three-dimensional Taylor–Green vortex problem of Brachet et al. [93] to study the production of small length scales, including vortex stretching and dissipation. The simulation parameters again follow from Coralic and Colonius [41]. Fig. 5(b) shows the dimensionless dissipation rate of the kinetic energy ε in dimensionless time t, as computed over the entire computational domain. We see that the vortex stretching grows until $t \approx 5$, after which the effects of viscous dissipation begin to dominate. MFC results closely match those of the direct solution computed by Brachet et al. [93] using spectral methods.

5.5. Further verification

MFC has been verified using several other problems, including several one-dimensional test cases. For example, of great importance are the development of spurious oscillations at material interfaces, which can significantly pollute simulation quality. To determine if such oscillations appear when using our method, Coralic and Colonius [41] considered the advection of an isolated air–water interface at constant velocity in a periodic domain. It was shown that when conservative variables are reconstructed, the interface is corrupted by spurious oscillations and the pressures can even become negative. Whereas when primitive variables are reconstructed, their character is oscillation free for both velocity and pressure down to round off error.

It is also challenging to predict the correct position and speed of waves that emit from shock–interface interactions. While the shock–bubble interaction problem considered in Section 5.1 showed that our method can approximate these quantities when comparing to experiments, it is helpful to consider a similar problem in one-dimension where exact solutions are available. Following Liu et al. [28], Coralic and Colonius [41] used the methods employed by MFC to analyze a Mach 8.96 helium shock wave impinging an air interface. It was shown that the numerical results quantitatively match the associated exact solution, correctly identifying the position and speed of all waves in the problem while avoiding any spurious oscillations. Of similar character is the gas–liquid shock tube problem of Cocchi et al. [94], which has been used as a model for underwater explosions. For this, Coralic and Colonius [41] also showed that the numerical solution matches the exact one and correctly identifies the position and speed of all waves.

Finally, the ensemble-averaged bubbly flow model introduced in Section 3.3 was verified by simulating a weak acoustic pulse impinging a dilute bubble screen and comparing to the linearized bubble dynamic results of Commander and Prosperetti [95]. We saw that the measured phase speed and acoustic attenuation, computed via the method of Ando [72] and Bryngelson et al. [69], match the expected results. Further, Bryngelson et al. [69] showed that this method quantitatively matches the volume-averaged formulation of the same problem [96,97].

6. Illustrative examples

6.1. Shock–bubble dynamics in a vessel phantom

We demonstrate the capabilities of MFC by first considering the shock-induced collapse of a gas bubble inside a deformable vessel. This is closely related to the vascular injury that can occur during shock-wave lithotripsy treatments [41,98]. This problem is particularly challenging because it involves large density-, pressure-, and viscosity-ratios as well as components with significantly different equation of state parameters. Specifically, we consider a 20 μm diameter air bubble centered in a 26 μm diameter cylindrical vessel filled with water and surrounded by 10% gelatin (see Fig. 6(a)). The problem is initialized via a 40 MPa shock wave impinging the side of the vessel. Details on the specific numerical setup can be found in Coralic [99].

We visualize the bubble dynamics and subsequent impingement and deformation of the vessel in Fig. 6. We see that by $t =$ 39 ns the bubble shape is asymmetric and the vessel contracts due to the incoming shock, after $t = 46$ ns the bubble surface has gained an inflection point and by $t = 66$ ns it impinges the vessel surface and becomes mushroom-shaped. Importantly, all surfaces remain smooth and free of spurious oscillations, despite the large density- and viscosity-ratios.

6.2. Vocalizing humpback whales

We next demonstrate the utility, flexibility, and robustness of the ensemble-averaged bubbly flow model described in Section 3.3. For this, we model the humpback whale bubble-net

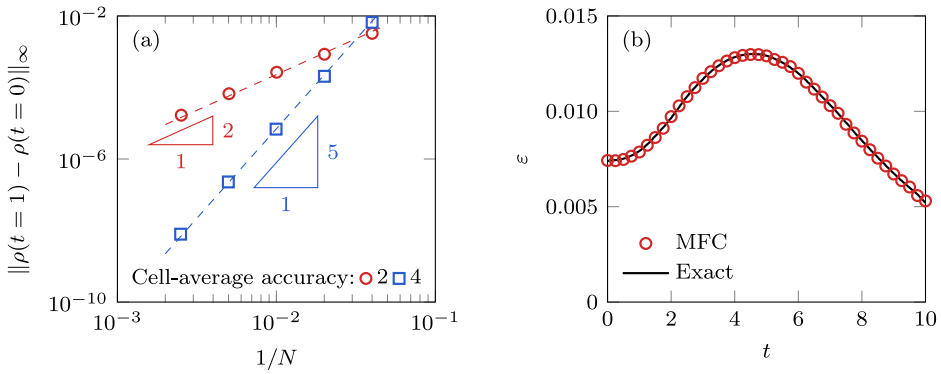
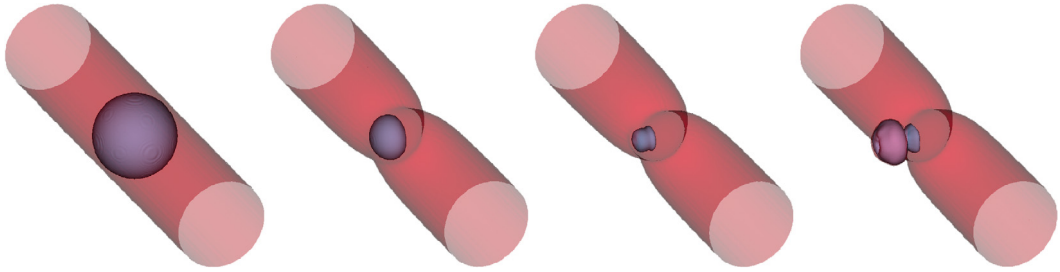


Fig. 5. (a) L_∞ density error associated with an isentropic steady vortex for the cell-averaged scheme accuracy labeled. (b) Non-dimensional kinetic energy dissipation rate ε associated with the three-dimensional Taylor–Green vortex problem; the MFC solution is compared to the direct spectral solution of Brachet et al. [93].

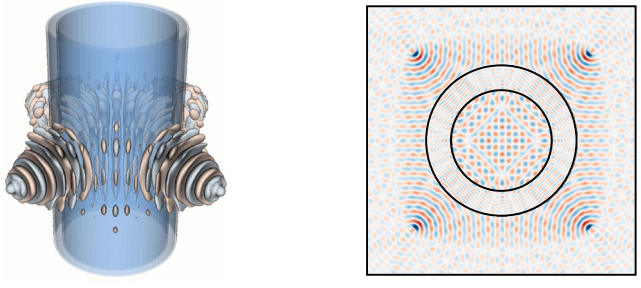


(a) $t = 0$ ns (b) $t = 39$ ns (c) $t = 46$ ns (d) $t = 58$ ns

Fig. 6. Temporal snapshots (a)–(d) that show the bubble collapse and vessel wall deformation. The shock impinges the vessel from right to left. The bubble and vessel walls are illustrated via their 0.5 volume-fraction isosurfaces.

feeding process [100], as described in Bryngelson and Colonius [101]. Specifically, multiple humpback whales vocalize towards an annular bubbly region called a bubble net, which is modeled using the acoustic source terms of Section 4.4 and bubbly regions described using the phase-averaged model.

We show the acoustics associated with the periodic excitation of the bubble net in Fig. 7 for both two- and three-dimensional configurations. In both cases, we see that the impedance associated with the relatively dilute bubble net (void fraction 10^{-4}) effectively shields the core region from the vocalizations. Indeed, it is anticipated that the whales use their nets as a tool for corralling their prey into this relatively quiet, compact region. We also see that the curved material interfaces remain smooth and free of spurious oscillations.



(a) (b)

Fig. 7. Visualizations of four model humpback whales vocalizing towards an annular bubble net in (a) three- and (b) two-dimensions. The acoustics are shown via isocontours of pressure and the annular bubble net is shaded.

7. Parallel performance benchmarks

It is important to ensure that our parallel implementation can utilize modern, large computer resources. To do this, we benchmark MFC's parallel architecture via the usual scaling and speedup tests. These tests were carried out using 288 compute nodes, each containing two 2.6 GHz six-core AMD Opteron processors.

Fig. 8 shows parallel performance benchmarks of MFC. The strong scaling test of Fig. 8(a) measures how the solution time varies for a fixed problem size as the number of computing cores varies, the weak scaling test (Fig. 8(b)) measures how well the computational load is balanced across the available cores by measuring solution time while fixing the grid size distributed to each core and varying the number of cores (thus changing the overall problem size), and the speedup test (Fig. 8(c)) measures how the solution time increases with respect to serial computation as the number of cores varies. Thus, speedup is defined by the ratio of time cost of a parallel simulation with a certain of cores to a serial computation. The strong scaling and speedup tests are carried out on a 500^3 grid, while a constant load of 50^3 cells per core is maintained during the weak scaling test. For all tests, MFC performs very near the ideal threshold until the number of cores is 4096, at which point the results deviate slightly from ideal.

8. Conclusions

We presented MFC, an open-source tool capable of simulating multi-component, multi-phase, and multi-scale flows. It uses state-of-the-art diffuse-interface models, coupled with high-order interface-capturing and Riemann solvers, to represent multi-material dynamics. MFC includes a variety of flow models

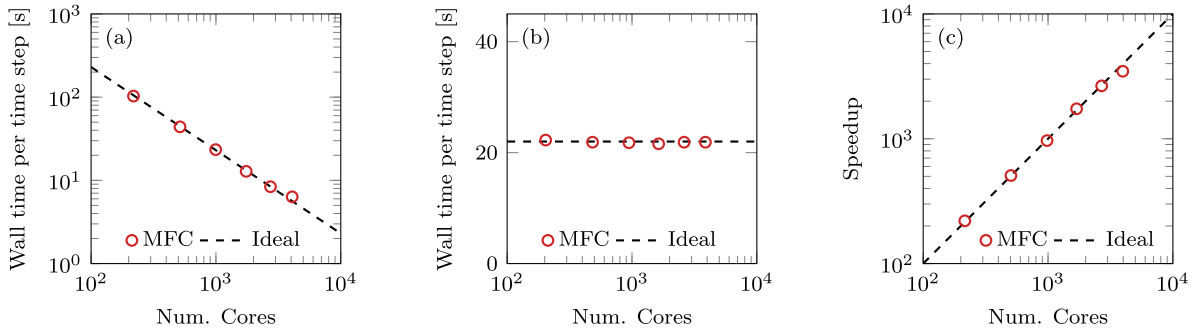


Fig. 8. MFC performance benchmarks: (a) strong scaling, (b) weak scaling, and (c) speedup tests.

and numerical methods, including spatial and temporal orders of accuracy, that are useful when considering the computational requirements of challenging open problems. It also includes options for additional physics and modeling techniques, including a sub-grid ensemble-averaged bubbly flow model.

We also described the requirements to build MFC and its design. This included external open-source software libraries that are readily available online. MFC was divided into three main components that initialize and simulate the flow, then process the exported simulation data. Each of these components is modular, and thus can be readily modified by new developers. They are coupled together via an intuitive input Python script that automatically generates the required Fortran input files and executes the software component. The exported simulation files can be readily analyzed or treated via parallel post-processing.

Finally, we presented a comprehensive set of validations, verifications, and illustrative examples. Validation was performed via comparisons to expected bubble dynamics and shock–bubble, shock–droplet, shock–water–cylinder experiments, while verification was obtained via numerical experiments involving isentropic and Taylor–Green vortices, as well as advected- and interface-interaction problems. The capabilities and fidelity of MFC were also illustrated using challenging studies of shock–bubble–viscous-vessel-wall interaction in application to shock-wave lithotripsy and acoustic-bubble-net dynamics in application to feeding humpback whales. A set of performance benchmarks also showed that MFC was able to perform near the ideal threshold of parallel computational efficiency.

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.

Acknowledgments

The authors are grateful for the suggestions of Dr. Benedikt Dorschner when making MFC open source. This work was supported in part by multiple past grants from the US National Institutes of Health (NIH), the US Office of Naval Research (ONR), and the US National Science Foundation (NSF), as well as current NIH Grant No. 2P01-DK043881 and ONR Grant Nos. N0014-17-1-2676 and N0014-18-1-2625. The computations presented here utilized the Extreme Science and Engineering Discovery Environment, which is supported under NSF , USA grant number CTS120005. K.M. acknowledges support from the Funai Foundation for Information Technology, USA via the Overseas Scholarship.

References

[1] C.E. Brennen, Interface Focus 5 (5) (2015).
[2] K. Laksari, S. Assari, B. Seibold, K. Sadeghipour, K. Darvish, Biomech. Model. Mechanobiol. 14 (3) (2015) 459–472.
[3] W.G. Proud, T.-T.N. Nguyen, C. Bo, B.J. Butler, R.L. Boddy, A. Williams, S. Masouros, K. Brown, Metall. Mater. Trans. A 46 (10) (2015) 4559–4566.
[4] A.J. Coleman, J.E. Saunders, L. Crum, M. Dyson, Ultrasound Med. Biol. 13 (2) (1987) 69–76.
[5] Y.A. Pishchalnikov, O.A. Sapozhnikov, M.R. Bailey, J.C. Williams, R.O. Cleveland, T. Colonius, L.A. Crum, A.P. Evan, J.A. McAteer, J. Endourol. 17 (7) (2003) 435–446.
[6] T. Ikeda, S. Yoshizawa, T. Masakata, J.S. Allen, S. Takagi, N. Ohta, T. Kitamura, Y. Matsumoto, Ultrasound Med. Biol. 32 (9) (2006) 1383–1397.
[7] R. Saurel, F. Petitpas, R. Abgrall, J. Fluid Mech. 607 (2008) 313–350.
[8] F. Petitpas, J. Massoni, R. Saurel, E. Lapebie, L. Munier, Int. J. Multiph. Flow 35 (8) (2009) 747–759.
[9] M. Pelanti, K.-M. Shyue, J. Comput. Phys. 259 (331–357) (2014).
[10] P.C. Etter, Underwater Acoustic Modeling and Simulation, CRC Press, 2013.
[11] V. Kedrinskii, Acta Astron. 3 (7–8) (1976) 623–632.
[12] V. Streeter, J. Hydraul. Eng. 109 (11) (1983) 1407–1423.
[13] M. Weyler, V. Streeter, P. Larsen, J. Basic Eng. 93 (1) (1971) 1–7.
[14] J.C. Meng, T. Colonius, J. Fluid Mech. 835 (2018) 1108–1135.
[15] J.C. Meng, T. Colonius, Shock Waves 25 (4) (2015) 399–414.
[16] K. Schmidmayer, F. Petitpas, E. Daniel, N. Favrie, S.L. Gavrilyuk, J. Comput. Phys. 334 (2017) 468–497.
[17] O.G. Engel, J. Res. Natl. Bur. Stand. 60 (3) (1958) 245–280.
[18] D.D. Joseph, J. Belanger, G. Beavers, Int. J. Multiph. Flow 25 (6) (1999) 1263–1303.
[19] A. Chauvin, E. Daniel, A. Chinnayya, J. Massoni, G. Jourdan, Shock Waves 26 (4) (2015) 403–415.
[20] Y. Tagawa, N. Oudalov, A.E. Ghalbzouri, C. Sun, D. Lohse, Lab Chip 13 (7) (2013) 1357–1363.
[21] J.-C. Veilleux, Pressure and Stress Transients in Autoinjector Devices (Ph.D. thesis), California Institute of Technology, 2019.
[22] D. Fuster, Flow Turbul. Combust. (2018) 1–40.
[23] G.J. Ball, B.P. Howell, T.G. Leighton, M. Schofield, Shock Waves 10 (2000) 265–276.
[24] C.K. Turangan, G.J. Ball, A.R. Jamaluddin, T.G. Leighton, Proc. R. Soc. A 473 (2205) (2017) 20170315.
[25] J. Glimm, X.L. Li, Y. Liu, Z.L. Xu, N. Zhao, SIAM J. Numer. Anal. 41 (2003) 1926–1947.
[26] J.P. Cocchi, R. Saurel, J. Comput. Phys. 137 (1997) 265–298.
[27] R. Abgrall, S. Karni, J. Comput. Phys. 169 (2001) 594–623.
[28] T.G. Liu, B.C. Khoo, K.S. Yeo, J. Comput. Phys. 190 (2003) 651–681.
[29] W. Liu, L. Yuan, C. Shu, Commun. Comput. Phys. 10 (2011) 785–806.
[30] S. Pan, L. Han, X. Hu, N.A. Adams, J. Comput. Phys. 371 (2018) 870–895.
[31] X.Y. Hu, B.C. Khoo, N.A. Adams, F.L. Huang, J. Comput. Phys. 219 (2) (2006) 553–578.
[32] L.H. Han, X.Y. Hu, N.A. Adams, J. Comput. Phys. 262 (2014) 131–152.
[33] C.-H. Chang, X. Deng, T.G. Theofanous, J. Comput. Phys. 242 (2013) 946–990.
[34] F. Denner, C.-N. Xiao, B.G.M. van Wachem, J. Comput. Phys. 367 (2018) 192–234.
[35] D. Fuster, S. Popinet, J. Comput. Phys. 374 (2018) 752–768.
[36] S. Mirjalili, C.B. Ivey, A. Mani, Comparison between the diffuse interface and volume of fluid methods for simulating two-phase flows, 2018, arXiv preprint arXiv:1803.07245.
[37] E. Johnsen, F. Ham, J. Comput. Phys. 231 (2012) 5705–5717.
[38] G. Perigaud, R. Saurel, J. Comput. Phys. 209 (2005) 139–178.
[39] K.M. Shyue, J. Comput. Phys. 456 (1999) 43–88.
[40] E. Johnsen, T. Colonius, J. Comput. Phys. 219 (2006) 715–732.
[41] V. Coralic, T. Colonius, J. Comput. Phys. 219 (2) (2014) 715–732.

[42] J. Massoni, R. Saurel, B. Nkonga, R. Abgrall, Int. J. Heat Mass Transfer 45 (2002) 1287–1307.
[43] J.C. Meng, Numerical Simulations of Droplet Aerobreakup (Ph.D. thesis), California Institute of Technology, 2016.
[44] G. Allaire, S. Clerc, S. Kokh, J. Comput. Phys. 181 (2002) 577–616.
[45] A. Kapila, R. Menikoff, J. Bdzil, S. Son, D. Stewart, Phys. Fluids 13 (2001) 3002–3024.
[46] R. Saurel, F. Petitpas, R.A. Berry, J. Comput. Phys. 228 (5) (2009) 1678–1712.
[47] R. Abgrall, J. Comput. Phys. 125 (1996) 150–160.
[48] G.S. Jiang, C.W. Schu, J. Comput. Phys. 126 (1996) 202–228.
[49] D.S. Balsara, C.W. Shu, J. Comput. Phys. 160 (2000) 405–452.
[50] A.K. Henrick, T.D. Aslam, J.M. Powers, J. Comput. Phys. 207 (2005) 542–567.
[51] E. Toro, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction, Springer, Dordrecht, New York, 2009.
[52] E. Toro, M. Spruce, W. Speares, Shock Waves 4 (1) (1994) 25–34.
[53] S. Gottlieb, C.-W. Shu, Math. Comp. 67 (221) (1998) 73–85.
[54] K. Schmidmayer, F. Petitpas, S. Le Martelot, E. Daniel, Comput. Phys. Comm. 251 (2020) 107093.
[55] K. Schmidmayer, S.H. Bryngelson, T. Colonius, J. Comput. Phys. 402 (2020) 109080.
[56] H.G. Weller, G. Tabor, H. Jasak, C. Fureby, Comput. Phys. 12 (6) (1998) 620–631.
[57] M. Frigo, S.G. Johnson, Proc. IEEE Intl. Conf. Acoustics Speech and Signal Processing, Vol. 3, IEEE, 1998, pp. 1381–1384.
[58] M. Miller, Silo–a Mesh and Field I/O Library and Scientific Database, Lawrence Livermore National Laboratory, 2009, URL https://wci.llnl.gov/simulation/computer-codes/silo.
[59] N. Halbwachs, P. Caspi, P. Raymond, D. Pilaud, Proc. IEEE 79 (9) (1991) 1305–1320.
[60] H. Childs, E. Brugger, B. Whitlock, J. Meredith, S. Ahern, D. Pugmire, K. Biagas, M. Miller, C. Harrison, G.H. Weber, H. Krishnan, T. Fogal, A. Sanderson, C. Garth, E.W. Bethel, D. Camp, O. Rübel, M. Durant, J.M. Favre, P. Navrátil, High Performance Visualization–ENabling Extreme-Scale Scientific Insight, 2012, pp. 357–372.
[61] J. Ahrens, B. Geveci, C. Law, Vis. Handb. 717 (2005).
[62] R. Menikoff, B.J. Plohr, Rev. Modern Phys. 61 (1) (1989) 75–130.
[63] O. Le Métayer, J. Massoni, R. Saurel, Int. J. Therm. Sci. 43 (2004) 265–276.
[64] S.P. Marsh, LASL Shock hugoniot data, los alamos series on dynamic material properties, University of California Press, Berkeley, 1980.
[65] A.B. Gojani, K. Ohtani, K. Takayama, S.H.R. Hosseini, Shock Waves 26 (2009) 63–68.
[66] A.B. Wood, A Textbook of Sound, G. Bell and Sons LTD, London, 1930.
[67] G.B. Wallis, One-Dimensional Two-Phase Flow, McGraw-Hill, 1969.
[68] D.Z. Zhang, A. Prosperetti, Phys. Fluids 6 (2956) (1994).
[69] S.H. Bryngelson, K. Schmidmayer, T. Colonius, Int. J. Multiph. Flow. 115 (2019) 137–143.
[70] T. Colonius, R. Hagmeijer, K. Ando, C.E. Brennen, Phys. Fluids 20 (040902) (2008).
[71] S.H. Bryngelson, A. Charalampopoulos, T.P. Sapsis, T. Colonius, Int. J. Multiph. Flow. 127 (2020) 103262.
[72] K. Ando, Effects of Polydispersity in Bubbly Flows (Ph.D. thesis), California Institute of Technology, 2010.
[73] J.B. Keller, M. Miksis, J. Acoust. Soc. Am. 68 (628) (1980).
[74] A. Preston, T. Colonius, C.E. Brennen, Phys. Fluids 19 (123302) (2007).
[75] J.C. Meng, Numerical Simulations of Droplet Aerobreakup (Ph.D. thesis), California Institute of Technology, 2016.
[76] S. Li, WENO Schemes for Cylindrical and Spherical Geometry, Technical Report Los Alamos Report LA-UR-03-8922, Los Alamos National Laboratory, Los Alamos, NM, 2003.
[77] A. Mignone, J. Comput. Phys. 270 (2014) 784–814.
[78] S. Wang, E. Johnsen, 21st AIAA Computational Fluid Dynamics Conference, AIAA 2013-2430, San Diego, CA, 2013.
[79] K. Mohseni, T. Colonius, J. Comput. Phys. 157 (2000) 787–795.
[80] K. Maeda, T. Colonius, Wave Motion 75 (2017) 36–49.
[81] J.F. Haas, B. Sturtevant, J. Fluid Mech. 181 (1987) 41–76.
[82] H. Terashima, G. Tryggvason, J. Comput. Phys. 228 (2009) 4012–4037.
[83] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, J. Comput. Phys. 152 (1999) 457–492.
[84] X.Y. Hu, B.C. Khoo, J. Comput. Phys. 198 (35–64) (2004).
[85] J.J. Quirk, S. Karni, J. Fluid Mech. 318 (1996) 129–163.
[86] B. Hejazialhosseini, D. Rossinelli, M. Bergdorf, P. Koumoutsakos, J. Comput. Phys. 229 (2010) 8364–8383.
[87] K.K. So, X.Y. Hu, N.A. Adams, J. Comput. Phys. 231 (11) (2012) 4304–4323.
[88] D. Igra, K. Takayama, Shock Waves 11 (2001) 219–228.
[89] T.G. Theofanous, V.V. Mitkin, C.L. Ng, C.-H. Chang, X. Deng, S. Sushchikh, Phys. Fluids 24 (2012) 022104.
[90] C.E. Brennen, Cavitation and Bubble Dynamics, Oxford University Press, 1995.
[91] H. Wadell, J. Geol. 43 (3) (1935) 250–280.
[92] V.A. Titarev, E.F. Toro, J. Comput. Phys. 201 (2004) 238–260.
[93] M.E. Brachet, D.I. Meiron, S.A. Orszag, B.G. Nickel, R.H. Morf, U. Frisch, J. Fluid Mech. 130 (1983) 411–452.
[94] J.P. Cocchi, R. Saurel, J.C. Loraud, Shock Waves 5 (1996) 347–357.
[95] K.W. Commander, A. Prosperetti, J. Acoust. Soc. Am. 85 (732) (1989).
[96] D. Fuster, T. Colonius, J. Fluid Mech. 688 (2011) 352–389.
[97] K. Maeda, T. Colonius, J. Comput. Phys. 371 (2018) 994–1017.
[98] V. Coralic, T. Colonius, Eur. J. Mech.–B/Fluids 40 (2013) 64–74.
[99] V. Coralic, Simulation of Shock-Induced Bubble Collapse with Application to Vascular Injury in Shockwave Lithotripsy (Ph.D. thesis), California Institute of Technology, 2015.
[100] J.H.W. Hain, G.R. Carter, S.D. Kraus, C.A. Mayo, H.E. Winn, Fishery Bull. 8 (2) (1982) 259–268.
[101] S.H. Bryngelson, T. Colonius, J. Acoust. Soc. Am. 147 (2020) 1126–1135.