multiUQ: A software package for uncertainty quantification of multiphase flows

Brian Turnquist

Mark Owkes

Montana State University

Bozeman, MT

May 3, 2022

multiUQ is a novel tool that simulates gas-liquid multiphase flows and quantifies uncertainty in results due to variability about fluid properties and initial/boundary conditions. The benefit over a typical deterministic solver is that inexact information, such as variability in fluid properties or flow rates, can be included to determine the affect on simulation solutions. It is common to deploy *non-intrusive* methods which utilize many solutions from a deterministic solver to generate a distribution of possible results. Contrarily, multiUQ uses an *intrusive* uncertainty quantification method wherein variables of interest are functions of space, time, and additional uncertainty dimensions. The intrusive solver is run once, giving a distribution of solutions as an output, as well as desired statistics. We use polynomial chaos to create the stochastic variables, which represent a distribution of values at each grid point. The stochastic variables are substituted into the incompressible Navier-Stokes equations, which govern the stochastic fluid dynamics. A stochastic

level set is used to capture the distribution of interfaces that are present in an uncertain multiphase flow. multiUQ is written in Fortran and uses a message passing interface (MPI) for parallel operation. Given the many applications of multiphase flows, including open flows, hydraulics, fuel injection systems, and atomizing jets, there is a massive potential benefit to calculating uncertainty information about these flows in a cost-effective manner.

Keywords: stochastic, polynomial chaos, gas-liquid flows, variable density Navier-Stokes, intrusive uncertainty quantification

1 Introduction

Given the growth in capacity and efficiency of computational resources over the past several decades, physics based models of fluid dynamics have become commonplace. Worldwide, many groups are creating their own in house software, developing new and more robust techniques to accurately solve the Navier-Stokes equations for a variety of applications including gas-liquid multiphase flows (e.g. [1, 4, 7, 19]). With improved computational resources and software, high-fidelity simulations help us better understand variable-density multiphase flows. This has aided in a myriad of engineering improvements, including direct injection systems for combustion engines [2], rotary bell cup atomization for paint application [18], and fire sprinkler spray efficiency [13].

While methods for multiphase flow simulations have been steadily improving, a method gap has existed in uncertainty quantification (UQ). Until recently, little effort has been placed on UQ of fluid dynamics in general, with even less on UQ of multiphase flows. In an attempt to fill this gap and also to encourage further development in this realm, multiUQ has been created as an open source application. multiUQ is a tool which allows for predicting fluid systems with uncertainty about each fluid parameter (i.e. density, viscosity, surface tension coefficient), initial and boundary conditions for the velocity field, and the location of the interface between phases. Imposing uncertainty about any of these variables creates a stochastic system, where a range of outcomes are possible and quantifiable. From this stochastic solution set statistics can be extracted, as well as extreme or average solutions.

For a computationally efficient approach to UQ of multiphase flows, several methods may be considered. Broadly speaking, UQ can be divided into non-intrusive or intrusive categories. Non-intrusive schemes include Monte-Carlo [12], collocation approaches [20], or non-intrusive polynomial chaos [14]. For these methods, a deterministic solver is run many times with a distribution of inputs, with statistics computed from the many saved solutions.

Alternatively, intrusive methods incorporate stochastic variables that are a function of uncertainty. These variables are typically represented with polynomial chaos (PC) [24] or Karhunen-Loeve [9, 10] expansions that capture variability in some uncertainty dimension(s), $\boldsymbol{\zeta}$. All together each variable is a function of time (t), space $\boldsymbol{x}$, and uncertainty dimension(s) $\boldsymbol{\zeta}$. These stochastic variables are substituted into a set of governing equations, which requires major changes to a code, and often development of a new solver from the ground up. However, once this software has been developed, uncertainty can be added as a model input, providing the affect of input uncertainty on model outputs from a single simulation at potentially less computational cost than non-intrusive methods. An example application is shown in Fig. 1, where there is uncertainty about the magnitude of an incoming jet (i.e. stochastic boundary condition) resulting in uncertainty about the solution of the jet at a given time (i.e. stochastic output).

multiUQ is the first intrusive, stochastic multiphase flow solver, and is capable of computing uncertainty about systems with high density ratios in three dimensions. As a tool, it is designed to aid in understanding the affect of uncertainty about initial and/or boundary conditions and fluid parameters on simulation solutions. Being able to quantify uncertainty caused by input variations may allow for improved engineering of a modeled system.

This article seeks to provide a general overview of how multiUQ is programmed and how it can be used. As shown in Fig. 2, multiUQ takes stochastic inputs, runs them through a procedure which discretely steps forward in time, and outputs stochastic results that can be visualized. To get a more detailed idea of how it works we will first provide the mathematical basis of the software, then describe the software mesh, discretization, and where to find the code and compile it. We then cover simulation capabilities and test cases that are included, how to run the code, and finally discuss the performance and scalability with some final closing remarks.

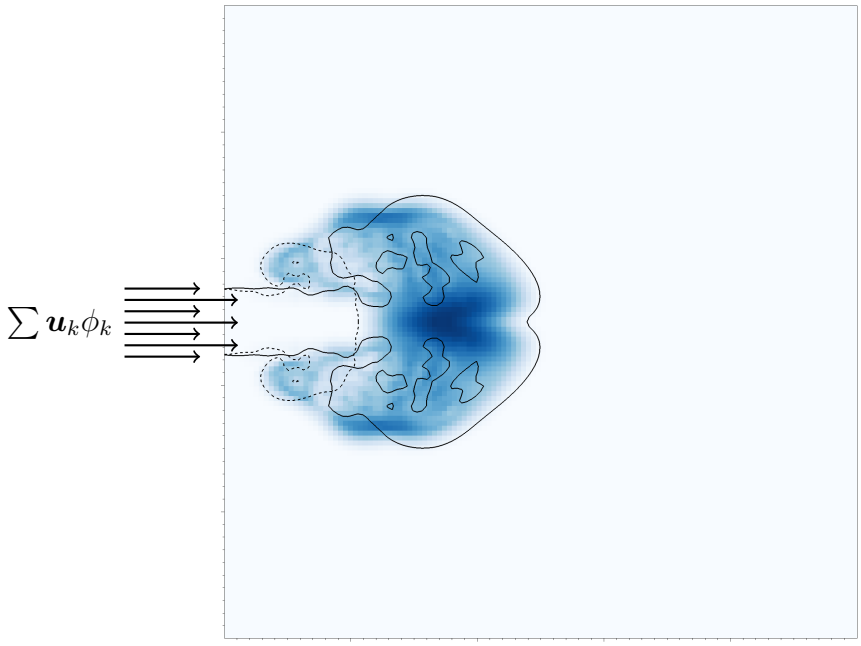


Fig. 1: Example simulation from multiUQ, an intrusive stochastic multiphase flow solver. A single simulation has been performed of a jet with uncertainty about the incoming velocity magnitude. The dotted line indicates the interface boundary of one solution at a low velocity, while the solid line indicates the solution of another at a high velocity. The color map indicates the variance about the interface location.

2 Mathematical development

This section provides a summary of the stochastic governing equations and representation of stochastic variables. Additional details are available in [23].

2.1 Governing equations

multiUQ uses the one-fluid approach to solve the incompressible Navier-Stokes equations, where fluid motion is governed by

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\eta \nabla P + \eta \nabla \cdot \left[\mu \left(\nabla \boldsymbol{u} + \nabla^{\mathrm{T}} \boldsymbol{u}\right)\right] + \eta \boldsymbol{f}_{\sigma} \delta_s \tag{1}$$

for velocity $\boldsymbol{u}$, specific volume $\eta = 1/\rho$ (for density ρ), pressure P, and surface tension force $\boldsymbol{f}_{\sigma}$, which is non-zero only at the interface as denoted by Dirac delta δ_s. Additionally, in an incompressible framework, mass conservation leads to the continuity equation

$$\nabla \cdot \boldsymbol{u} = 0. \tag{2}$$

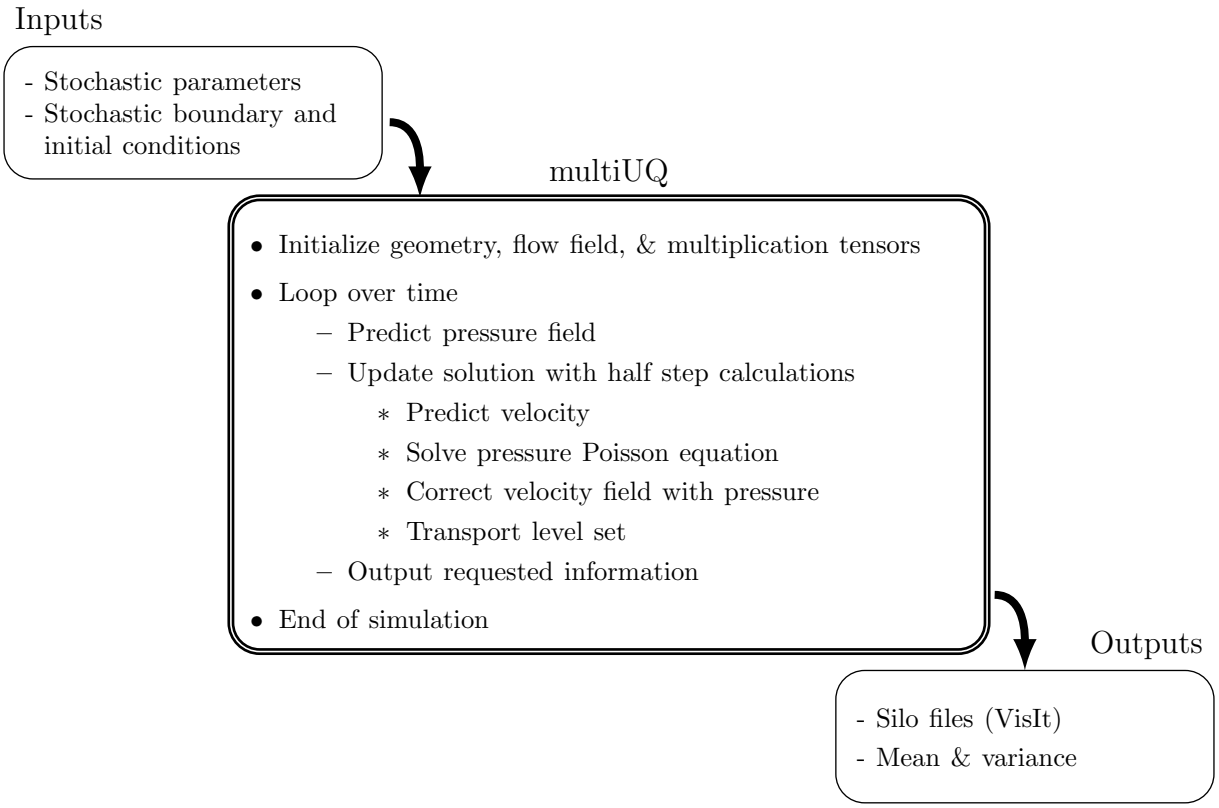


Fig. 2: Visual flow of multiUQ software.

These are the equations of motion for a deterministic, incompressible system of fluids. To allow for uncertainty to exist about these equations stochastic variables are used. For instance, some variable ψ is described as

$$\psi(\boldsymbol{x}, t, \boldsymbol{\zeta}) = \sum_{k=0}^{N} \psi_k(\boldsymbol{x}, t)\phi_k(\boldsymbol{\zeta}) = \psi_k\phi_k, \tag{3}$$

which is a polynomial chaos expansion of ψ at each spatial location $\boldsymbol{x}$ and time t. The polynomial chaos expansions project ψ onto a series of $N+1$ polynomial basis functions ϕ_k that are a function of uncertainty dimensions $\boldsymbol{\zeta}$. As shown and moving forward, we will utilize Einstein notation, emphasizing the summation over repeated indices.

Allowing for uncertainty about all variables in equation 1 (except time), we substitute stochastic velocity $\boldsymbol{u}_k\phi_k$, specific volume $\eta_k\phi_k$, pressure $P_k\phi_k$, viscosity $\mu_k\phi_k$ and surface tension force $\boldsymbol{f}_{\sigma:k}\phi_k$, to arrive at a stochastic form of the Navier Stokes equations

$$\begin{aligned}\frac{\partial \boldsymbol{u}_k\phi_k}{\partial t} + \boldsymbol{u}_k\phi_k \cdot \nabla \boldsymbol{u}_l\phi_l = \\ -\eta_k\phi_k\nabla P_l\phi_l + \eta_k\phi_k\nabla\cdot\left[\mu_l\phi_l\left(\nabla\boldsymbol{u}_m\phi_m + \nabla^{\mathrm{T}}\boldsymbol{u}_m\phi_m\right)\right] + \eta_k\phi_k\boldsymbol{f}_{\sigma:l}\phi_l\delta_s.\end{aligned} \tag{4}$$

Because we wish to find the basis weights $\boldsymbol{u}_k$, we now utilize the property of orthogonality. This property depends on which basis functions are used. In multiUQ, we are currently utilizing Legendre polynomials, which are orthogonal on the interval $\boldsymbol{\zeta} \in [-1, 1]$, i.e.,

$$\int_{-1}^{1} \phi_k \phi_b \delta\zeta = \begin{cases} \langle \phi_k \phi_b \rangle & k = b \\ 0 & k \neq b \end{cases}. \tag{5}$$

To utilize orthogonality from equation 5, we first multiply equation 4 by a test function ϕ_b, giving us

$$\begin{aligned} &\frac{\partial \boldsymbol{u}_k \phi_k}{\partial t} \phi_b + \boldsymbol{u}_k \phi_k \cdot \nabla \boldsymbol{u}_l \phi_l \phi_b = \\ &\qquad -\eta_k \phi_k \nabla P_l \phi_l \phi_b + \eta_k \phi_k \nabla \cdot \left[\mu_l \phi_l \left(\nabla \boldsymbol{u}_m \phi_m + \nabla^{\mathrm{T}} \boldsymbol{u}_m \phi_m \right) \right] \phi_b + \eta_k \phi_k \boldsymbol{f}_{\sigma:l} \phi_l \delta_s \phi_b. \end{aligned} \tag{6}$$

We then integrate over the interval of orthogonality

$$\begin{aligned} &\int_{-1}^{1} \frac{\partial \boldsymbol{u}_k \phi_k}{\partial t} \phi_b + \boldsymbol{u}_k \phi_k \cdot \nabla \boldsymbol{u}_l \phi_l \phi_b \; d\boldsymbol{\zeta} = \\ &\qquad \int_{-1}^{1} -\eta_k \phi_k \nabla P_l \phi_l \phi_b + \eta_k \phi_k \nabla \cdot \left[\mu_l \phi_l \left(\nabla \boldsymbol{u}_m \phi_m + \nabla^{\mathrm{T}} \boldsymbol{u}_m \phi_m \right) \right] \phi_b + \eta_k \phi_k \boldsymbol{f}_{\sigma:l} \phi_l \delta_s \phi_b \; d\boldsymbol{\zeta}. \end{aligned} \tag{7}$$

Leveraging the orthogonality property, $\int_{-1}^{1} \phi_b \phi_b d\zeta$ can be factored out of the time derivative and divided through all terms. This leads to the stochastic Navier-Stokes equations in a form that can be solved to find the stochastic velocity weights $\boldsymbol{u}_b$

$$\frac{\partial \boldsymbol{u}_b}{\partial t} + \boldsymbol{u}_k \cdot \nabla \boldsymbol{u}_l C_{klb} = -\eta_k \nabla P_l C_{klb} + \eta_k \nabla \cdot \left[\mu_l \left(\nabla \boldsymbol{u}_m + \nabla^{\mathrm{T}} \boldsymbol{u}_m \right) \right] C_{klmb} + \eta_k \boldsymbol{f}_{\sigma:l} \delta_s C_{klb} \tag{8}$$

for $b = 0, \ldots, N$, where

$$C_{klb} = \frac{\int_{-1}^{1} \phi_k \phi_l \phi_b d\zeta}{\int_{-1}^{1} \phi_b \phi_b d\zeta} = \frac{\langle \phi_k \phi_l \phi_b \rangle}{\langle \phi_b \phi_b \rangle} \tag{9}$$

and

$$C_{klmb} = \frac{\langle \phi_k \phi_l \phi_m \phi_b \rangle}{\langle \phi_b \phi_b \rangle} \tag{10}$$

are the multiplication tensors that have been factored out to simplify the equations. Addition-

ally, the stochastic continuity equations are

$$\nabla \cdot \boldsymbol{u}_b = 0 \tag{11}$$

for $b = 0, \ldots, N$, which closes the incompressible system. Next, we define the location of the interface between phases.

2.2 Interface capturing

To create a solver capable of both incompressible single- and multi-phase flows, we need to define a method for determining the location of the interface. While there exist both interface tracking and interface capturing schemes, an interface capturing scheme such as the level set approach [17] combines most easily with the stochastic PC design due to the continuous definition of the level set function that can be projected onto basis functions. More specifically, to capture the interface between phases a conservative level set approach is utilized, as outlined in [23] and following the work of [15, 5]. Transport of the interface is accomplished by

$$\frac{\partial \psi}{\partial t} + \nabla \cdot (\psi \boldsymbol{u}) = 0, \tag{12}$$

for level set ψ and time t. The level set profile is set to a hyperbolic tangent, which is initialized with a signed-distance function $g(\boldsymbol{x}, t)$ by

$$\psi(\boldsymbol{x}, t) = \left[1 + \exp\left(\frac{-2g(\boldsymbol{x}, t)}{\varepsilon_1 + \varepsilon_2}\right)\right]^{-1} \tag{13}$$

where ε_1 and ε_2 control the sharpness of the interface. These two values control the amount of diffusion for two distinct diffusion terms present in the accompanying reinitialization procedure discussed below. We set the values to $\varepsilon_1 = 9\text{max(dx,dy,dz)}/8$ and $\varepsilon_2 = \text{max(dx,dy,dz)}/8$, which provides seven grid cells to represent the hyperbolic tangent profile. These values may be reduced or increased to sharpen or diffuse the interface profile.

Expanding the level set transport equation to include uncertain variables is done by substituting $\psi_k \phi_k$ and $\boldsymbol{u}_k \phi_k$ into equation 12. We then multiply through by a test function ϕ_b and

integrate over $\boldsymbol{\zeta} \in [-1, 1]$ to get

$$\frac{\partial \psi_b}{\partial t} + \nabla \cdot (\psi_k \boldsymbol{u}_l) \, C_{klb} = 0. \tag{14}$$

Similarly, equation 13 is generalized to include a stochastic interface position through a stochastic signed-distance function $g_k \phi_k$. Substituting in the stochastic signed-distance function, multiplying by a test function ϕ_b, integrating over $\boldsymbol{\zeta} \in [-1, 1]$, and solving for ψ_b leads to

$$\psi_b = \frac{1}{\langle \phi_b \phi_b \rangle} \int_{-1}^{1} \left[1 + \exp\left(\frac{-2 g_k \phi_k}{\varepsilon_1 + \varepsilon_2} \right) \right]^{-1} \phi_b d\boldsymbol{\zeta}. \tag{15}$$

Unfortunately, this equation has no analytic solution, and thus requires the use of numerical integration. However the equation only sets the initial condition and the integration is performed once at the beginning of the simulation. multiUQ has both a Gaussian quadrature and a Romberg integration with Richardson extrapolation for the initialization of such profiles. While Romberg integration converges to machine precision, given the potentially high cost, it can be useful to compute an estimate with a Gaussian quadrature.

When transporting the conservative level set there is no guarantee the hyperbolic interface profile will be maintained. Due to this issue, a reinitialization procedure must be performed. As presented in [23], reinitialization is accomplished by

$$\frac{\partial \psi}{\partial \tau} + \nabla \cdot [\psi \, (1 - \psi) \, \boldsymbol{r}] = \nabla \cdot [\varepsilon_1 \, (\nabla \psi \cdot \boldsymbol{r}) \, \boldsymbol{r}] + \nabla \cdot (\varepsilon_2 \nabla \psi) \tag{16}$$

where τ is pseudo-time and $\boldsymbol{r}$ is a continuous (non-unit) interface normal vector calculated by

$$\boldsymbol{r} = \frac{\nabla \psi}{|\nabla \psi|_{\max}}. \tag{17}$$

As shown in [23], $|\nabla \psi|_{\max} = 1/4(\varepsilon_1 + \varepsilon_2)$. The use of continuous normal vectors $\boldsymbol{r}$ allows for easy implementation in the PC framework, where the smoothly varying $\boldsymbol{r}$ field is projected onto orthogonal basis functions. This is different from other methodologies, such as [5, 15, 16], where unit normal vectors (i.e. $\boldsymbol{n} = \nabla \psi / |\nabla \psi|$) are used.

To calculate the basis weights ψ_b and perform a stochastic reinitialization, we substitute stochastic variables into equation 16, multiply by a test function ϕ_b and integrate over $\boldsymbol{\zeta} \in$

$[-1, 1]$, finding

$$\frac{\partial \psi_b}{\partial \tau} + \nabla \cdot (\psi_k \boldsymbol{r}_l C_{klb} - \psi_k \psi_l \boldsymbol{r}_m C_{klmb}) = \nabla \cdot \left[\varepsilon_1 \left(\nabla \psi_k \cdot \boldsymbol{r}_l\right) \boldsymbol{r}_m\right] C_{klmb} + \nabla \cdot (\varepsilon_2 \nabla \psi_b) \,. \tag{18}$$

With a satisfactory interface capturing method and method of reinitialization, we utilize the level set to calculate fluid properties $\mu(\boldsymbol{x}, t)$ and $\eta(\boldsymbol{x}, t)$. In a deterministic setting, this is accomplished with

$$\mu = \mu_1 \psi + (1 - \psi)\mu_2 = \mu_2 + (\mu_1 - \mu_2)\psi \tag{19}$$

$$\eta = \eta_1 \psi + (1 - \psi)\eta_2 = \eta_2 + (\eta_1 - \eta_2)\psi. \tag{20}$$

Allowing for uncertainty about both fluid phases (i.e. $\mu_1(\boldsymbol{\zeta}) = \mu_{1:k}\phi_k$ and $\mu_2(\boldsymbol{\zeta}) = \mu_{2:k}\phi_k$), we calculate stochastic quantities via

$$\mu_b = \mu_{2:b} + \left(\mu_{1:k}\psi_l - \mu_{2:k}\psi_l\right) C_{klb} \tag{21}$$

$$\eta_b = \eta_{2:b} + \left(\eta_{1:k}\psi_l - \eta_{2:k}\psi_l\right) C_{klb}. \tag{22}$$

2.3 Surface tension force

Having described the mathematics of the stochastic level set implementation, we can now calculate the surface tension force that appears in the Navier-Stokes equations, i.e.,

$$\boldsymbol{f}_\sigma = \sigma \kappa \boldsymbol{n} \tag{23}$$

for surface tension coefficient σ, curvature κ, and unit normal $\boldsymbol{n}$ about the interface. A commonly used approach is the continuum surface force method, first described by Brackbill *et al.* [3], where with the conservative level set

$$\boldsymbol{f}_\sigma \delta_s \approx \sigma \kappa \nabla \psi. \tag{24}$$

As we are using a hyperbolic tangent profile to describe the jump in fluid properties that occurs at the interface, we also use this profile to smooth the surface tension force. The hyperbolic

tangent operates as a smoothed Heaviside function (i.e. $H(\boldsymbol{x}) \approx \psi(\boldsymbol{x})$). As outlined by [21], we can then approximate the unit normal at the interface with $\boldsymbol{n}\delta_s = \nabla H \approx \nabla\psi$. The surface tension force is thus smoothed over the few cells that surround the $\psi = 0.5$ isosurface, where the interface is defined. The number of cells defining this width depends on the previously defined ε_1 and ε_2, or inversely $|\nabla\psi|_{\max}$.

Curvature κ is calculated with

$$\kappa = -\nabla \cdot \boldsymbol{n} \tag{25}$$

for unit normal $\boldsymbol{n} = \nabla\psi/|\nabla\psi|$ about the interface. In a deterministic model, we then have

$$\boldsymbol{f}_\sigma = -\sigma\left(\nabla \cdot \boldsymbol{n}\right)\nabla\psi. \tag{26}$$

Defining the surface tension in a stochastic PC regime, we then substitute stochastic variables, multiply by a test function ϕ_b, and integrate over $\boldsymbol{\zeta} \in [-1, 1]$ to find

$$\boldsymbol{f}_{\sigma:b} = \sigma_k \kappa_l \nabla\psi_m C_{klmb}. \tag{27}$$

To compute a stochastic curvature we could compute a stochastic normal vector and then a stochastic curvature. However, projecting a field of unit normal vectors onto smooth basis functions is problematic as the unit normal vector field contains directional discontinuities. Alternatively, we directly compute a stochastic curvature by evaluating the projection onto basis functions with a Gaussian quadrature. At each quadrature point, along $\boldsymbol{\zeta}$ dimension(s), unit normal vectors are computed (but not projected onto basis functions) and used to evaluate the stochastic curvature.

The stochastic curvature weights can be computed with

$$\kappa_b = \frac{1}{\langle\phi_b\phi_b\rangle}\sum_{n=1}^{N_q} -w_n \nabla \cdot \frac{\nabla\psi_k(\zeta_n)\phi_k(\zeta_n)}{|\nabla\psi_l(\zeta_n)\phi_l(\zeta_n)|} \tag{28}$$

for N_q quadrature points with weights w_n and abscissas ζ_n for $n = 1, \ldots, N_q$.

This completes the description of the governing equations used within multiUQ to solve for the velocity and pressure fields, define the interface location and surface tension force, and compute fluid properties. The next section will describe how these equations are solved on a

computational mesh.

3 Software description

multiUQ is written in modern Fortran, and can be compiled by GNU or Intel compilers. Parallelization of the code for computation on multiple processors is accomplished by use of the Message Passing Interface (MPI) library. A Linux or Unix terminal is currently required for compiling and running and the code.

3.1 Computational mesh and parallelization

multiUQ uses a three-dimensional rectangular computational domain with a structured Cartesian mesh. Scalar values S, such as pressure P, level set ψ, density ρ, and viscosity μ are held at the cell center. Subscripts on $S^n_{i,j,k}$ denote discrete spatial indexing in the x, y, and z directions, respectively, while superscripts denote time discretization. Vector components of velocity $\boldsymbol{u}$, surface tension $\boldsymbol{f}_\sigma$, and continuous normal vector $\boldsymbol{r}$ are held at the cell walls, as shown in Fig. 3 for two dimensions with $\boldsymbol{u} = [u, v]$. When vector or scalar values are needed at half step locations (for either space or time), linear interpolation is used.

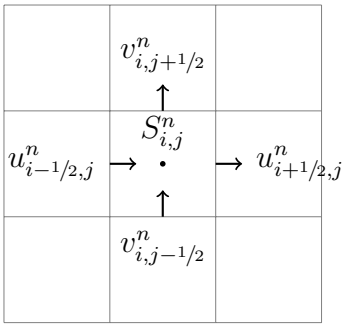


Fig. 3: Schematic of staggered grid used for the stochastic multiphase solver.

Parallelization is achieved by decomposing the computational domain along spatial $\boldsymbol{x}$ directions. Users may specify the processor division in the x, y, and z directions, depending on the extent of the domain in each. The stochastic dimension(s) $\boldsymbol{\zeta}$ are not decomposed in the current implementation. Inter-processor communication is achieved by defining ghost cells at the edge of each processor. The ghost cells provide memory to store information communicated using MPI from the neighboring processor. At the edge of the spatial domain, these ghost cells are

used to discretize the boundary conditions.

3.2 Numerical implementation

Discretization of the governing equations (equations. 8, 11, 14, and 18) on the Cartesian mesh is done with the finite difference method. For the purpose of simplicity in this work, a deterministic notation is used. In practice, a derivative is found for each basis weight b.

Time marching is accomplished through an iterative semi-implicit Crank-Nicolson approach coupled with a pressure correction method which breaks up Navier-Stokes into two steps. For each Crank-Nicolson iteration, we first calculate a partially discretized predicted velocity field with

$$\begin{aligned}\frac{\boldsymbol{u}^* - \boldsymbol{u}^n}{\Delta t} &= -\boldsymbol{u}^{n+1/2} \cdot \nabla \boldsymbol{u}^{n+1/2} \\ &\quad + \eta^{n+1/2} \nabla \cdot \left[\mu^{n+1/2} \left(\nabla \boldsymbol{u}^{n+1/2} + \nabla^{\mathrm{T}} \boldsymbol{u}^{n+1/2}\right)\right] + \eta^{n+1/2} \boldsymbol{f}_\sigma^{n+1/2}\end{aligned} \tag{29}$$

using a midpoint velocity $\boldsymbol{u}^{n+1/2} = \frac{1}{2}(\boldsymbol{u}^n + \boldsymbol{u}^{n+1})$ (for the first iteration we set $\boldsymbol{u}^{n+1} = \boldsymbol{u}^n$). Calculation of derivatives for the gradient of velocity is accomplished through an upwinding scheme for model stability, where

$$\nabla u_{i-1/2,j,k} = \frac{u_{i-1/2,j,k} - u_{i-3/2,j,k}}{\Delta x} \quad \text{for} \quad u_{i-1/2,j,k} > 0 \tag{30}$$

and

$$\nabla u_{i-1/2,j,k} = \frac{u_{i+1/2,j,k} - u_{i-1/2,j,k}}{\Delta x} \quad \text{for} \quad u_{i-1/2,j,k} < 0. \tag{31}$$

Components for the y and z domains are calculated similarly.

The predicted field $\boldsymbol{u}^*$ is then corrected to a divergence free condition by including pressure. To avoid creating a system of coupled Poisson equations in a stochastic implementation, we use a fast, decomposed pressure correction method as presented by [6] and [22], which breaks up the pressure term into explicit and implicit parts. The pressure correction equation is then

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^*}{\Delta t} = -\eta_0 \nabla P^{n+1} - \left(\eta^{n+1} - \eta_0\right) \nabla \hat{P} \tag{32}$$

for constant specific volume η_0 (where $\eta_0 = 1/\min(\rho_1, \rho_2)$) and estimated pressure field $\hat{P}$. The estimation of $\hat{P}$ is accomplished by a semi-Langrangian projection for the initial iteration, but

is updated several times at each time step to reduce numerical error. This approach improves computational efficiency for increasing orders of uncertainty. We then calculate P^{n+1} by taking the divergence of equation 32 leading to

$$\nabla^2 P^{n+1} = \rho_0 \frac{\nabla \cdot \boldsymbol{u}^*}{\Delta t} - \rho_0 \nabla \cdot \left(\eta^{n+1} - \eta_0\right) \nabla \hat{P}. \tag{33}$$

The divergence, $\nabla \cdot \boldsymbol{u}^*$, is discretized with second order accurate finite difference operators, e.g. the x-component of the divergence is computed with

$$\left(\frac{du^*}{dx}\right)_{i,j,k} = \frac{u^*_{i+1/2,j,k} - u^*_{i-1/2,j,k}}{\Delta x}. \tag{34}$$

Similar calculations are performed for the other components, forcing a divergence free condition at each cell for calculation of the pressure $P^{n+1}_{i,j,k}$ at the cell center. Evaluation of the $\nabla \hat{P}$ term for equation 33 is found by first evaluating the gradient at the cell walls with

$$\left(\frac{dP}{dx}\right)_{i-1/2,j,k} = \frac{P_{i,j,k} - P_{i-1,j,k}}{\Delta x}. \tag{35}$$

We then multiply by specific volume η^{n+1} (found at cell walls by linear interpolation) and η_0 to find

$$\begin{aligned} &\rho_0 \left(\nabla \cdot \left(\eta^{n+1} - \eta_0\right) \nabla \hat{P}\right)_{i,j,k} = \\ &\rho_0 \frac{\left(\eta^{n+1}_{i+1/2,j,k} - \eta_0\right) \left(d\hat{P}/dx\right)_{i+1/2,j,k} - \left(\eta^{n+1}_{i-1/2,j,k} - \eta_0\right) \left(d\hat{P}/dx\right)_{i-1/2,j,k}}{\Delta x} \end{aligned} \tag{36}$$

in the x domain (applied similarly to y and z domains). Several methods for solving the pressure Poisson equation are available within multiUQ. However, utilizing the PFMG parallel semi-coarsening multigrid solver within the HYPRE package from Lawrence Livermore National Lab has been found to be efficient and accurate. Having calculated both P^{n+1} and $\hat{P}$, we finally apply equation 32 to the predicted velocity $\boldsymbol{u}^*$, utilizing the discretization in equation 35 to calculate the gradient for either P^{n+1} or $\hat{P}$ at cell walls.

Transport of the level set is performed in sync with Navier-Stokes. As mentioned previously, transport of the interface is governed by equation 12. Discretization is accomplished by utilizing a finite volume operator across the cell. For example, transport of the level set is discretized in

the x direction as

$$\nabla \cdot (\psi u) = \frac{\psi_{i+1/2,j,k} u_{i+1/2,j,k} - \psi_{i-1/2,j,k}, u_{i-1/2,j,k}}{\Delta x} \tag{37}$$

resulting in a calculation of ψ at the cell center. Similar gradients are calculated for the y, and z dimensions. Additionally, upwinding is necessary for accurate transport of the level set. We utilize a high order upwind central (HUOC-5) scheme, as described in [5], to interpolate values of ψ to the cell walls.

Stepping forward in time for the level set transport is done in the same manner as the velocity predictions. We utilize a half step location $\psi^{n+1/2} = \frac{1}{2}(\psi^n + \psi^{n+1})$ for calculation of the next time, estimating with ψ^n for the first iteration. Calculating the gradients at the half-step, we then update the level set with

$$\psi^{n+1} = \psi^n + \Delta t \left[\nabla \cdot \left(\psi^{n+1/2} \boldsymbol{u}^{n+1/2}\right)\right]. \tag{38}$$

With these discretized equations, the first iteration of the Crank-Nicholson scheme is conducted. For the next iteration the mid-time velocity and mid-time level set are computed and used to calculate mid-time specific volume $\eta^{n+1/2}$ and viscosity $\mu^{n+1/2}$. At least one iteration of the Crank-Nicholson scheme is performed to get a 2^{nd} order accurate estimate of the solution at t^{n+1}. However, this process can occur for some user defined number of maximum iterations, or potentially to some convergence tolerance.

As presented in [22], we can iterate over the Crank-Nicolson scheme more than once to improve our calculation of the estimated pressure field $\hat{P}$ and thus converge to a standard pressure correction method. Not only does this reduce the error imposed by the decomposed pressure correction approach, it also improves simulation stability and allows for more accurate simulations of high density ratio systems. Our findings in [22] show that without iteration or very small time steps, there can be significant error between the decomposed and standard pressure correction methods due to poor estimates of $\hat{P}$. Several possible combinations are discussed in [22].

Program Title	multiUQ
Code Availability	https://bitbucket.org/markowkes/multiuq
Licensing Provisions	GPLv3
Programming Language	Fortran 95/2003
Parallelization	OpenMPI >3.xx
Assumptions	Incompressible, low Mach, DNS
Interface Scheme	Conservative level set, continuum surface force (CSF)
Dependencies	Szip, HDF5, Silo, HYPRE, FFTW, VisIt

Fig. 4: General information about the code.

4 Software functionality

The goal of this section is to minimally describe the necessary components needed to get multiUQ up and running. The source code of multiUQ is available for download through Bitbucket, the details of which are presented in Fig. 4. The website contains all the required source (*.f90) files as well as an example makefile for compilation and inputs file for executing a simulation.

After obtaining the source code, the next step is to compile the code. Additional libraries are necessary are needed for solving the pressure Poisson equation and for writing output files for post-simulation visualization. We next discuss the needed libraries, then how to compile the code, and finally provide details for setting up a simulation and executing the program.

4.1 Required libraries

Several libraries are required to compile and run multiUQ. Minimum software versions include Szip 2.1 and HDF5 1.10 provided by The HDF Group, Silo 4.10.2 and HYPRE 2.11.2 provided by Lawrence Livermore National Lab (LLNL), FFTW 3.3.8 developed at Massachusetts Institute of Technology, and OpenMPI 3 or above.

Calculation of the pressure field by solution of the pressure Poisson equation is accomplished with either the HYPRE package or one of the built-in solvers. Several different linear solvers are available within the HYPRE package, though not all have been implemented for use by multiUQ. Presently the SMG, PFMG, BICGSTAB, and GMRES methods are available within multiUQ. Of these, the PFMG method offers the most computationally efficient and stable solution for the various test cases present in the code. Currently in development is the use of

a fast Fourier transform, using FFTW for quick solution of the pressure field. However, this method is not currently active, but is required for code compilation.

Compilation of the Silo library depends on Szip and HDF5 which together provide the method for storing the output simulation data efficiently. Silo files are output by multiUQ for visualization of results, which is done with the VisIt package from LLNL after running a simulation with multiUQ. Additionally, a wiki is available on Bitbucket with details on compiling the code as well as additional libraries.

4.2 Simulation setup

Several inputs are required to run a successful simulation. Included with the source code is an inputs file, which is a text file containing all needed information. Among the variables most commonly used are mesh size, processor allocation (in space), simulation (initial velocity field and multiphase geometry), boundary conditions, CFL number for time step size determination, application of uncertainty and the number of basis functions used, tolerance levels for velocity field divergence and pressure fields, as well as the desired pressure solver.

A number of predefined cases have been implemented in the code including Poiseuille and Couette flows, a deformation case, lid driven cavity flow, oscillating droplet, standing wave, atomizing jet, and a jet in crossflow. When one of these cases is selected as the velocity simulation and geometry profile, the boundary conditions and level set geometries are created with the predefined conditions (found in simulation.f90, geometry.f90, and boundary.f90). It is still necessary to choose fluid parameters and a pressure solver, as well as the precision of the divergence free condition and frequency of data output. Additionally, the output files required for visualization of the system depending on quantities of interest must be chosen. These outputs include velocity, pressure, density, and variance about these quantities, among others.

Having set the desired inputs, multiUQ is executed through the command line terminal in either Linux or Unix environments using

```
>> mpiexec -np 4 ./multiUQ inputs
```

for the number of proccesors requested (-np). Once the code is running, some basic simulation information will output to the command line terminal, while most will be saved to the folder where the run command was executed.

4.3 Outputs

Aside from the data printed to the command line terminal, outputs are stored in the form of a Silo file, which can be read by LLNL's VisIt program for visualization. A variety of arrays are output by default, including average values of velocity, density, viscosity, pressure, and level set. It is also possible to output variance and probability about these variables at the request of the user.

Statistical variables, including variance and probability, are useful for determining areas of a system which have the most uncertainty. Those regions of uncertainty contain a wide array of potential solutions, which are stored within the stochastic variables. Given each stochastic variable is a function of uncertainty, any number of particular solutions may be output. We presently output 3 separate solutions by evaluating a stochastic variable $S_k\phi_k$ at $\zeta = [-1, 0, 1]$ prior to output (in visit.f90). We also output the average for each variable, which is stored in the 0^{th} basis weight, i.e. S_0. Outputting additional particular solutions would require modification to the code in visit.f90. In the future, it may be useful to track and output extreme solutions.

5 Simulation capabilities

Several test cases have been built into the multiUQ package which allow for testing the accuracy of the code. These tests exist to determine the accuracy of each component of the multiphase system, including the incompressible velocity field and the level set transport and conservation, as well as the fully coupled system. Given reasonable results in each of these tests, further simulations can be performed by modification of the geometry and velocity initialization and boundary conditions. Presently each of these tests must be run manually after installation.

5.1 Running a test

To set up a simulation, several values must be included in the inputs file. For all tests, it is necessary to choose the dimensions of the domain, the number of grid cells in each dimension, as well as the number of processors on which to run for each dimension. All necessary inputs are included in the example inputs file downloaded with the multiUQ source code.

To choose a specific test, there are five main inputs to consider. The first is the velocity

simulation, which sets up the initial velocity field, and also controls the boundary conditions that are called and enforced throughout the simulation. If fluid is being injected into the domain, the incoming velocity magnitude must be set. Additionally, it may be necessary to turn on periodic boundary conditions in any of the three dimensions.

The second is the geometry input. There is a toggle switch for multiphase simulations, which if turned off results in a flat (or null) geometry profile. When turned on a geometry must be selected, with potentially other measures to place it in the domain. For example, if a sphere is selected, the center of the sphere and its radius will need to be read in the input file.

A third set of inputs is that of the fluid parameters. multiUQ accepts interfacial (inside the level set) and extrafacial (outside the level set) fluid density and viscosity, as well as the surface tension coefficient.

The fourth set of inputs are related to setting up a stochastic simulation. A toggle exists to turn UQ on or off, which when off will automatically set the basis order to 0. If on, the user can select the order of the model (number of basis functions to use) and the number of quadrature points for numerical integration.

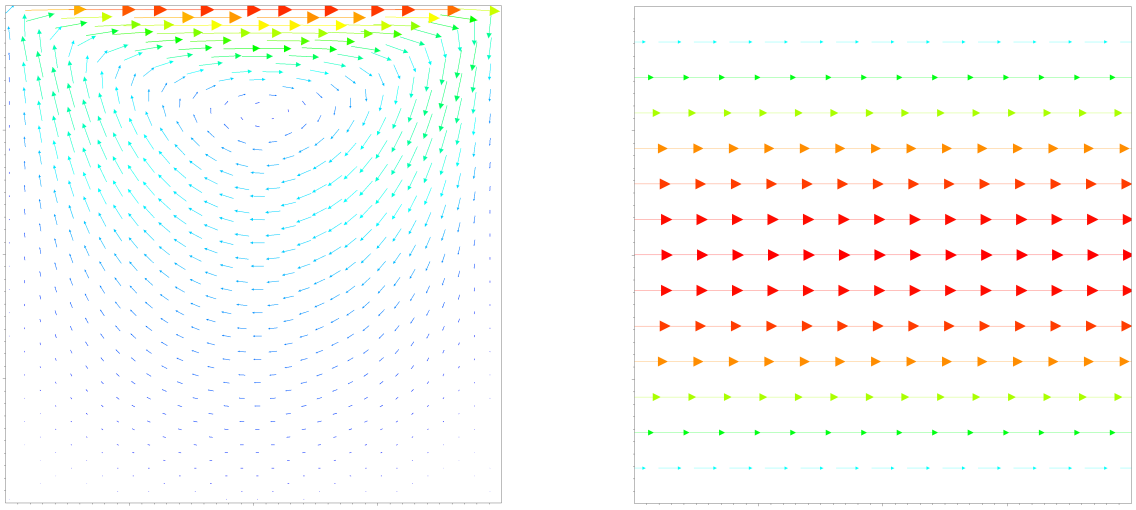
It is also useful to consider the fifth set of inputs, which controls the pressure correction method used. Either the Gauss-Seidel or fast, decomposed pressure correction method can be chosen. For the fast method, it is possible to use a built in biconjugate gradient solver, or one of the several HYPRE solvers built in to multiUQ.

The remainder of the inputs in the example file can be thought of as default settings. These can be utilized for more control over the testing process once a user is more comfortable operating multiUQ.

multiUQ is run through the command line terminal in a Unix or Linux system with mpiexec. It will output basic information to the terminal, while several other outputs will be saved to the folder in which the command is run. A folder titled 'Visit' will contain all the output files. Opening a version of VisIt from LLNL, you can open a new file, navigate to the 'Visit' folder, and open the file called multiUQ.visit. This will provide access to all the variables output by a simulation, such as velocity, level set, or curvature.

5.2 Navier-Stokes tests

Basic tests of single phase flows are necessary to test the velocity and pressure solver. These can be run once the code has been compiled to be sure the program is running properly and without error. Several test cases are available within multiUQ, including Couette, Poiseuille, and lid-driven cavity flows, examples of which are shown in Fig. 5.



(a) Cavity flow (b) Poiseuille flow

Fig. 5: Single phase testing for comparison to analytic results.

These tests can be run by choosing a test of interest, i.e. 'Couette', in the velocity simulation subject field of the inputs file. Additionally, there is a choice to either use or not use multiphase, which will turn off the level set transport and set all fluid parameters to that of fluid 1. In running these tests we look for convergence to analytic results, a divergence free velocity field, and boundary conditions that are consistent with the flow scenario.

5.3 Level set tests

Several level set tests are built into multiUQ for verification of the numerical framework through comparison to analytic results. A simple test case is that of a channel flow, where imposing a uniform velocity field results in transport of the level set a certain distance for a given time. The analytic solution for this test is a displacement of the initial condition by $\boldsymbol{u}t$. Additionally, we can test the ability of multiUQ to transport an uncertain interface, such as that of an uncertain uniform velocity field as shown in Fig. 6. Here, we see that uncertainty in the magnitude of the flow velocity causes uncertainty in the displacement of the initial condition.

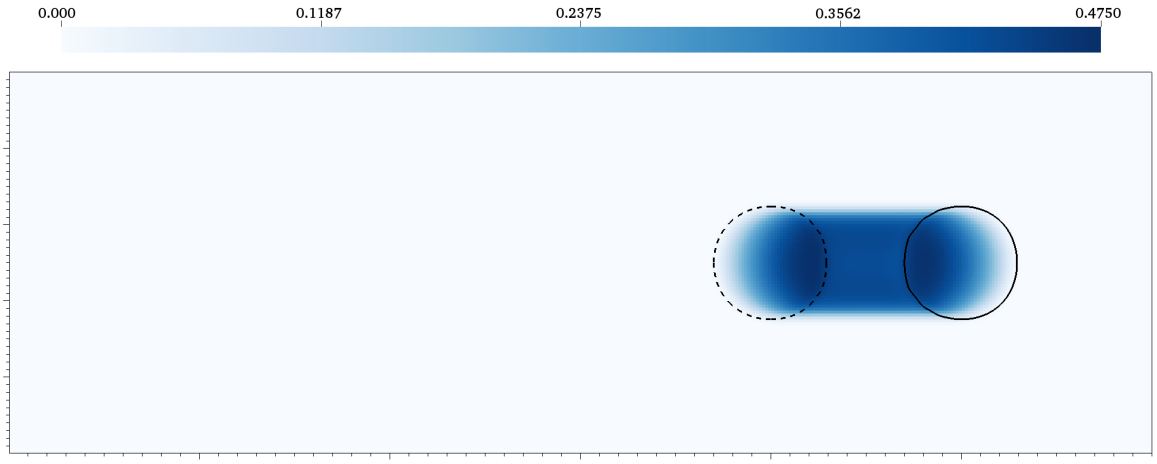
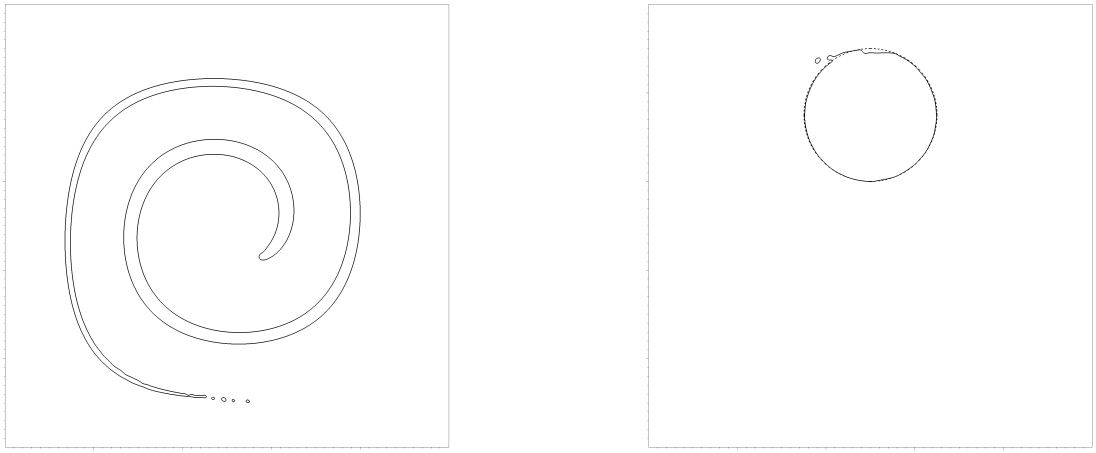


Fig. 6: Position of the stochastic circle with $\zeta = -1$ (dashed) and $\zeta = +1$ (solid). Level set variance is represented by the color bar, while interface locations correspond to analytic solutions.

A common benchmark for testing the quality of interface transport and reinitialization is that of the deformation test, as shown in Fig. 7. In this scenario a 2D level set is placed in a rotating velocity field. The level set is stretched to a point, then rotated back to its original position as shown in [23]. In a perfect scenario, the final solution would match the initial condition. Inspection of the level set at its final location provides a means to assess the accuracy of the level set transport and the ability of the reinitialization procedure to maintain the level set. Furthermore, this test allows the mass conservation properties to be studied.



(a) $t = 4$ (b) $t = 8$

Fig. 7: Illustration of the $\psi = 0.5$ interface for the deformation test case at maximum stretching left and final condition right for a 512×512 mesh. At the final time the interface is compared to the initial condition, displayed as a dotted line.

Also available is the case of Zalesak's disk, which tests the transport of the level set in 2D ([23]). Beginning with a circle, a slot is removed from the bottom, creating two sharp inside

and two sharp outside corners. The disk is then rotated in a rigid-body vortex velocity field for one full cycle. At the conclusion of the test, deformation of the disk is inspected. Given the movement is rigid, a perfect test would result in the final state matching the initial state.

5.4 Multiphase flows tests

multiUQ contains a few pre-built multiphase simulations which can be used to test transport of the level set in a calculated velocity field, accuracy of the surface tension force, and the precision of the numerical methods employed. A classic test case is that of the oscillating droplet, which has an analytic solution to the period of oscillation, as discussed by [11]. This case, which is typically performed in two dimensions, directly tests the ability of the surface tension force to drive flow, given no imposition of velocity from the boundaries. A global kinetic energy is calculated at each time step, and the results are output for comparison to periodic expectation. An example is shown in Fig. 8, where an uncertain droplet is displayed for a given time, as well as the kinetic energy output for two specific solutions of the case.

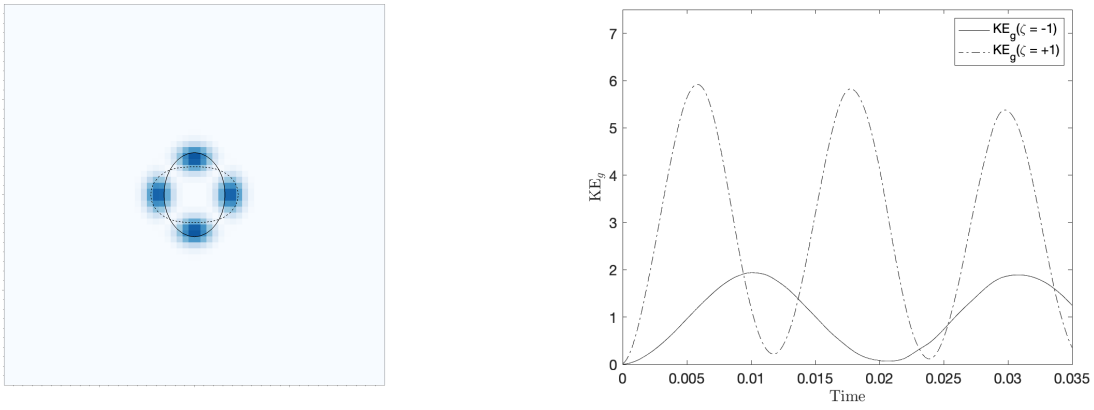
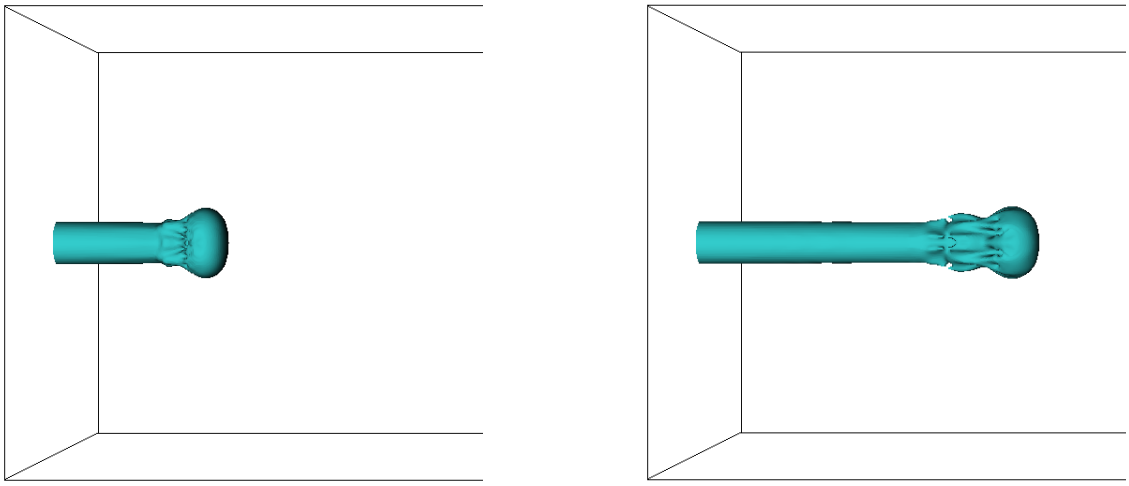


Fig. 8: An uncertain oscillating droplet is presented, with uncertainty imposed about the surface tension coefficient. At left is an image of the solution at a given time, where the solid line indicates the solution to the droplet with the smallest surface tension coefficient, and the dotted line indicates the solution to the droplet with the largest surface tension coefficient. The color map indicates variance of the level set. At right we see the kinetic energy of each potential solution over time.

Another test is that of the damped surface wave. This test, again run in 2D, gauges the interplay between surface tension and viscous damping of the fluid. At initialization, a flat surface is disturbed slightly. With no gravity imposed, the surface tension term pushes the interface towards a flat surface. Two main analytic solutions are discussed by [8], which are built into the solver for comparison to numeric results.

The comparisons between numerical and analytic solutions are used to test the implementation and build confidence the solver is working properly. With this knowledge, we can utilize the solver for other research problems. One focus of the multiUQ project was for application to the field of atomizing jets. For example, Fig. 9 displays a deterministic simulation of an atomizing jet at 2 separate time steps. This jet was run in a domain of 2 cm^2 for a jet with diameter 0.2 cm with an incoming jet velocity of 1000 cm/s for an air and water system, which results in an incoming Reynold's number of Re = 2000.



(a) Time = 0.001 s (b) Time = 0.002 s

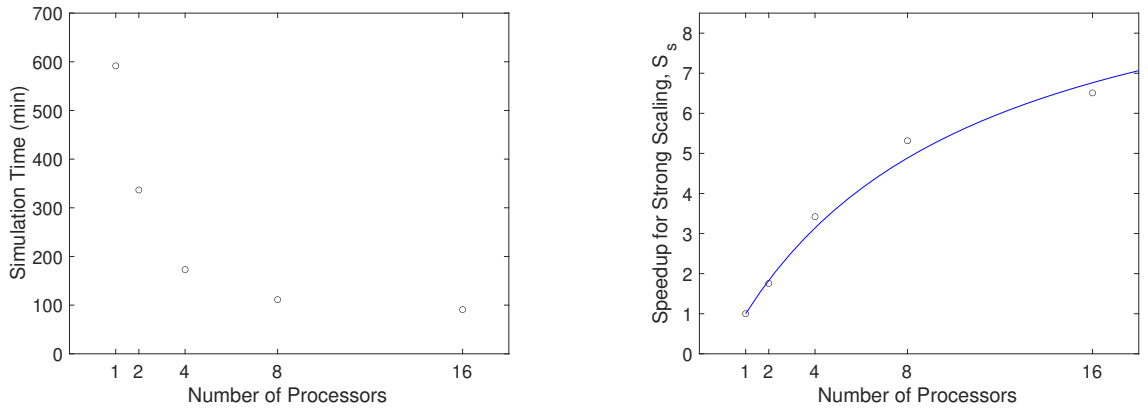
Fig. 9: Deterministic simulation (256^3) of an atomizing jet at two progressive time steps.

5.5 Performance and scaling

multiUQ is written to be operated in parallel and capable of high-fidelity multiphase flow simulations with uncertainty about several variables. Given the expense of running deterministic models on large clusters, it is also necessary for UQ modeling to be scalable. Amdahl's law for strong scaling states

$$S_s = \frac{1}{s + p/N_p}, \tag{39}$$

where S_s is speedup for strong scaling, s is the proportion of the time spent on serial operations, $p = 1 - s$ is the proportion of time spent on parallel operations, and N_p is the number of processors. We expect to see a speedup of simulations based on the proportion of serial to parallelized calculations in the code. For deterministic simulations we show good strong scaling in Fig. 10, with $s = 0.091$ found by way of a non-linear least squares fit.



(a) Simulation timing

(b) Speedup from 1 processor

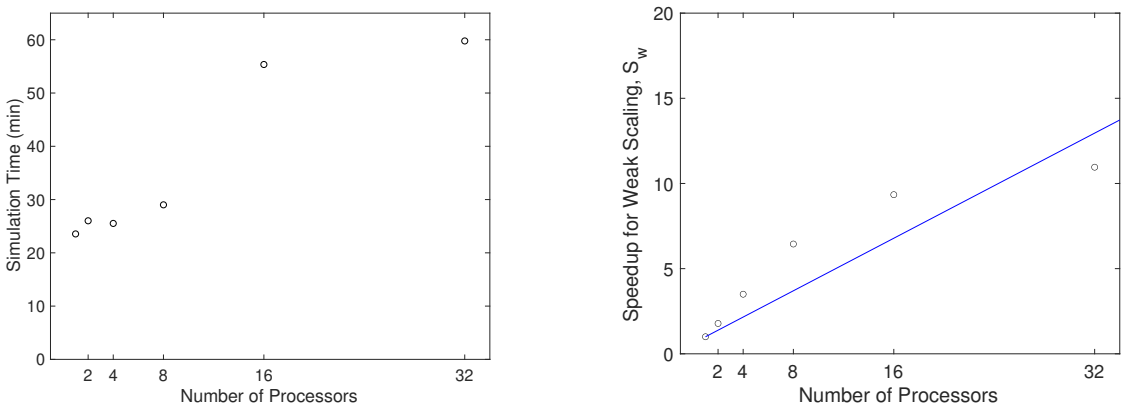
Fig. 10: Simulation scaling of a coarse (100^3 mesh) running over a range of processors on a single node. At right, the line drawn illustrates strong scaling with Amdahl's law, with $s = 0.091$.

A different approach to scaling is that of weak scaling, which suggests that as the problem size grows, so to should the number of processors used to perform the simulation. Gustafson's law states

$$S_w = s + pN_p, \tag{40}$$

where S_w is the speedup for weak scaling, while other variables are as that described for equation 39. Fig. 11 displays the results of timing as the mesh is refined and the processor count increases, beginning with a 64^3 mesh on a single processor. Using least squares regression, we find that for weak scaling we have a value of $s = 0.603$, which is significantly different from the results of our strong scaling tests. This scaling value is not ideal, and suggests further code development is necessary to improve the scalability of multiUQ. However, to run fine mesh simulations, it is still useful to run in parallel. Operation works well with 64^3 mesh per core for a deterministic system. When adding basis functions (up to order $N = 25$), it can be helpful to reduce this mesh by half to decrease simulation time.

Additionally, given the parallelization scheme is set up to arrange processors along the spatial domain, we see that the uncertainty domain is not processor dependent. Because of this, we may also need to increase the number of processors in use for simulations requiring more basis functions, i.e. a weak scaling problem.



(a) Simulation timing

(b) Speedup from 1 processor

Fig. 11: Simulation efficiency and scaling for a range of mesh sizes, increasing the processor count as mesh is refined. At right, the line drawn illustrates weak scaling with Gustafson's law, with $s = 0.603$.

6 Conclusion

We have presented an overview of the methodology and implementation of multiUQ. At present, this code is available for download as discussed. multiUQ is the first of its kind, and represents a niche sector of uncertainty quantification in multiphase flow dynamics. Certainly, given it is a development level software, modifications are ongoing and necessary to improve the quality of results and further the knowledge of UQ in multiphase systems.

In future work, perhaps the most pressing issues are related to the interface capturing scheme and weak scaling. Ideally a more conservative and robust method, such as a stochastic volume of fluid method, could be employed. Additionally, modifications to the code for improvements in run-time efficiency and scalability are needed, and would aid those wanting to use the software for research of more detailed and complex systems.

Additionally, the 3D implementation is a fairly recent development. More research is necessary to validate the results of 3D simulations and improve time stability. One possibility is the implementation of a higher order upwinding scheme for treatment of the convective term in the momentum equation, which would improve the order of the model and reduce numerical dissipation.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant Nos. 1511325 and 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.

References

[1] Thomas Bellotti and Maxime Theillard. "A coupled level-set and reference map method for interface representation with applications to two-phase flows simulation". In: *Journal of Computational Physics* 392 (Sept. 2019), pp. 266–290. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.05.003. URL: http://www.sciencedirect.com/science/article/pii/S0021999119303286.

[2] Jesus Benajes et al. "Optimization of the combustion system of a medium duty direct injection diesel engine by combining CFD modeling with experimental validation". In: *Energy Conversion and Management* 110 (Feb. 2016), pp. 212–229. ISSN: 0196-8904. DOI: 10.1016/j.enconman.2015.12.010. URL: http://www.sciencedirect.com/science/article/pii/S0196890415010985.

[3] J.U Brackbill, D.B Kothe, and C Zemach. "A continuum method for modeling surface tension". In: *Journal of Computational Physics* 100.2 (June 1992), pp. 335–354. ISSN: 0021-9991. DOI: 10.1016/0021-9991(92)90240-Y. URL: http://www.sciencedirect.com/science/article/pii/002199919290240Y.

[4] Robert Chiodi and Olivier Desjardins. "A reformulation of the conservative level set reinitialization equation for accurate and robust simulation of complex multiphase flows". In: *Journal of Computational Physics* 343 (Aug. 2017), pp. 186–200. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2017.04.053. URL: http://www.sciencedirect.com/science/article/pii/S0021999117303327.

[5] Olivier Desjardins, Vincent Moureau, and Heinz Pitsch. "An accurate conservative level set/ghost fluid method for simulating turbulent atomization". In: *Journal of Computational Physics* 227.18 (Sept. 2008), pp. 8395–8416. ISSN: 0021-9991. DOI: 10.1016/j.

jcp.2008.05.027. URL: http://www.sciencedirect.com/science/article/pii/S0021999108003112.

[6] Michael S. Dodd and Antonino Ferrante. "A fast pressure-correction method for incompressible two-fluid flows". In: *Journal of Computational Physics* 273 (Sept. 2014), pp. 416–434. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2014.05.024. URL: http://www.sciencedirect.com/science/article/pii/S0021999114003702.

[7] Thomas Frachon and Sara Zahedi. "A cut finite element method for incompressible two-phase Navier–Stokes flows". In: *Journal of Computational Physics* 384 (May 2019), pp. 77–98. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.01.028. URL: http://www.sciencedirect.com/science/article/pii/S0021999119300798.

[8] M. Herrmann. "A balanced force refined level set grid method for two-phase flows on unstructured flow solver grids". In: *Journal of Computational Physics* 227.4 (Feb. 2008), pp. 2674–2706. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2007.11.002. URL: http://www.sciencedirect.com/science/article/pii/S0021999107004998.

[9] Kari Karhunen. *Über lineare Methoden in der Wahrscheinlichkeitsrechnung*. Vol. 37. Sana, 1947.

[10] Michel Loeve. "Probability theory: foundations, random sequences". In: (1955).

[11] F. R. S. Lord Rayleigh. "VI. On the capillary phenomena of jets". In: *Proceedings of the Royal Society of London* 29.196-199 (Jan. 1879), pp. 71–97. DOI: 10.1098/rspl.1879.0015. URL: http://rspl.royalsocietypublishing.org/content/29/196-199/71.short.

[12] Nicholas Metropolis and Stanislaw Ulam. "The monte carlo method". In: *Journal of the American Statistical Association* 44.247 (1949), pp. 335–341.

[13] T.M. Myers and A.W. Marshall. "A description of the initial fire sprinkler spray". In: *Fire Safety Journal* 84 (Aug. 2016), pp. 1–7. ISSN: 0379-7112. DOI: 10.1016/j.firesaf.2016.05.004. URL: http://www.sciencedirect.com/science/article/pii/S0379711216300716.

[14] Habib N. Najm. "Uncertainty Quantification and Polynomial Chaos Techniques in Computational Fluid Dynamics". In: *Annual Review of Fluid Mechanics* 41.1 (Dec. 2008), pp. 35–52. ISSN: 0066-4189. DOI: 10.1146/annurev.fluid.010908.165248. URL: https://doi.org/10.1146/annurev.fluid.010908.165248 (visited on 04/04/2018).

[15] Elin Olsson and Gunilla Kreiss. "A conservative level set method for two phase flow". In: *Journal of Computational Physics* 210.1 (Nov. 2005), pp. 225–246. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2005.04.007. URL: http://www.sciencedirect.com/science/article/pii/S0021999105002184.

[16] Elin Olsson, Gunilla Kreiss, and Sara Zahedi. "A conservative level set method for two phase flow II". In: *Journal of Computational Physics* 225.1 (July 2007), pp. 785–807. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2006.12.027. URL: http://www.sciencedirect.com/science/article/pii/S0021999107000046.

[17] Stanley Osher and James A Sethian. "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations". In: *Journal of Computational Physics* 79.1 (1988), pp. 12–49. ISSN: 0021-9991.

[18] Mohammad-Reza Pendar and José Carlos Páscoa. "Numerical modeling of electrostatic spray painting transfer processes in rotary bell cup for automotive painting". In: *International Journal of Heat and Fluid Flow* 80 (Dec. 2019), p. 108499. ISSN: 0142-727X. DOI: 10.1016/j.ijheatfluidflow.2019.108499. URL: http://www.sciencedirect.com/science/article/pii/S0142727X19305995.

[19] M. Schick, V. Heuveline, and O. P. Le Maître. "A Newton-Galerkin Method for Fluid Flow Exhibiting Uncertain Periodic Dynamics." In: *SIAM Review* 58.1 (Jan. 2016), pp. 119–140. ISSN: 00361445. URL: http://search.ebscohost.com.proxybz.lib.montana.edu/login.aspx?direct=true&db=a9h&AN=112816633&login.asp&site=ehost-live.

[20] Menner A. Tatang et al. "An efficient method for parametric uncertainty analysis of numerical geophysical models". In: *Journal of Geophysical Research: Atmospheres* 102.D18 (Sept. 1997), pp. 21925–21932. ISSN: 2156-2202. DOI: 10.1029/97JD01654. URL: http://dx.doi.org/10.1029/97JD01654.

[21] G. Tryggvason, R. Scardovelli, and S. Zaleski. *Direct Numerical Simulations of Gas–Liquid Multiphase Flows*. Cambridge University Press, 2011. ISBN: 978-1-139-49670-4. URL: https://books.google.com/books?id=nY5bjSYq-AEC.

[22] Brian Turnquist and Mark Owkes. "A fast, decomposed pressure correction method for an intrusive stochastic multiphase flow solver". In: *Computers & Fluids* Under Review ().

[23] Brian Turnquist and Mark Owkes. "multiUQ: An intrusive uncertainty quantification tool for gas-liquid multiphase flows". In: *Journal of Computational Physics* 399 (Dec. 2019), p. 108951. ISSN: 0021-9991. DOI: 10.1016/j.jcp.2019.108951. URL: http://www.sciencedirect.com/science/article/pii/S0021999119306564.

[24] Norbert Wiener. "The Homogeneous Chaos". In: *American Journal of Mathematics* 60.4 (1938), pp. 897–936. ISSN: 00029327, 10806377. DOI: 10.2307/2371268. URL: http://www.jstor.org/stable/2371268.