Building Fully Adaptive Stochastic Models for Multiphase Blowdown Simulations

by

Isaac M. Asher

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Aerospace Engineering)
in the University of Michigan
2015

Doctoral Committee:

Associate Professor Krzysztof J. Fidkowski, Chair
Associate Professor Hany Abdel-Khalik, Purdue University
Associate Professor Annalisa Manera
Associate Professor Joaquim R.R.A. Martins

Some heroes are a bird, proudly soaring above a vast emptiness, alone
I am one small man contemplating the dirt supporting me
gazing at the metropolises of rock and trees and ice
stretching to the horizon.

ACKNOWLEDGEMENTS

There were countless individuals without whom this thesis could not have been. I will name those that I can remember here.

First, to my beautiful, smart, amazing girlfriend Mariah who kept me steady throughout this time, the keel in my journey. You keep me focused on the real priorities in life.

Second, to my advisor whose weekly meetings centered and recharged me through months and months of failed convergence.

The Fidkowski research group has helped with numerous suggestions and feedback. My first few years included fruitful collaboration with the nuclear engineering department, including graduate students Tim Drzewiecki, Tim Grunloh, Aaron Wysocki, Artem Yankov, research scientist Victor Petrov, and Professors Downar and Manera. Some of the work in Chapter 2 was done by these individuals. I would also like to thank the CASL project for funding us.

I would also like to thank my collaborators at other institutions, including Simon Lo at Star-CD, Dillon Shaver, Steven Antal, and Professor Podowski at RPI, Corey Bryant at UT Austin, Tim Wildey at Sandia, and Troy Butler at Colorado State.

Finally, to my parents Samuel and Shelli and my sisters Yvonne and Miriam, unfailingly supportive and proud of just about everything I do. You are a great family and even more importantly great friends.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

LIST OF SYMBOLS

$u, \mathbf{u}$	State variable in physical and or stochastic space (scalar and vector)
u_{true}	Exact value of state variable
$\overline{u}, \overline{\mathbf{u}}$	Coefficient of the stochastic approximation of u
$R, \mathbf{R}$	Residual equation or error estimate (scalar and vector)
$F, \mathbf{F}$	Flux
$\mathbf{S}$	Source term
$\mathbf{q}$	Gradient of the state variable
$\mathbf{g}$	Time component of physical solution
K, k	Output of a single physical simulation, $K = \int k(u) d\Omega_{\vec{z}}$
μ	A parameter
$\boldsymbol{\psi}$	Adjoint or dual variable in physical and or stochastic space
J, j	Output of an uncertainty quantification study (scalar), $J = \int j(u) d\Omega_{\vec{x}}$
$\vec{x}$	Point in stochastic (parameter) space
n_d	Number of stochastic dimensions (number of parameters)
$\vec{d}$	Direction in stochastic space
Γ	Active linear subspace of the stochastic space
$\Omega_{\vec{x}}$	Stochastic domain
p	Polynomial order
n_{term}	Number of terms in the stochastic approximation
n_{samp}	Number of samples taken in the stochastic space
$\vec{e}_i$	Vector of all zeros except the i^{th} component is 1
ϕ^p	Univariate Legendre Polynomial of order p
$\mathbf{A}^{LS}$	Least-squares fitting matrix for the stochastic approximation
s	Coordinate along a line
$\Delta J(\vec{d})$	Error metric associated with the direction $\vec{d}$
Ω_{swarm}	Domain for particle swarm optimization (PSO)
$\vec{\xi}, \vec{\eta}$	Position and velocity of particles in PSO
$\text{Unif}(a, b)$	Random number in $[a, b]$ drawn from a uniform distribution
$\chi, C_{\text{cognitive}}, C_{\text{social}}$	Parameters of PSO
$\delta \bar{J}(\vec{d}, \vec{x})$	Modified error metric, computed at a single sample
G	Matrix whose columns are various values of $\nabla u(\vec{x})$
USV^{-1}	Singular value decomposition of G; s_i is the i^{th} singular value
$E[\cdot]$	Expected value

$var(\cdot)$	Variance
$p_x(x)$	Probability density function of x
$\Omega_{\vec{z}}$	Domain for physical simulations
$\vec{z}$	Point in physical space (or space-time)
z, t	Spatial and temporal coordinates
$v, \mathbf{v}, \mathbf{w}$	Test or trial basis function in physical space
$\boldsymbol{\nu}$	Test or trial basis function in time
ρ_m, ρ_l, ρ_g	Fluid density (mixture average, liquid, gas)
$v_m, v_l, v_g, \mathbf{v}$	Fluid velocity – mixture average, liquid, gas (vector)
α	Void fraction, volume fraction of gas in a mixture
V_{dj}, C_0	Drift velocity ($= v_g - v_l$) and distribution parameter
$\dot{m}_g$	Gas generation rate in a mixture (kg/m^3-s)
p_m	Pressure of mixture
μ_m, μ_l, μ_g	Viscosity (mixture, liquid, gas)
g_z	Gravitational acceleration in the z-direction
f_m	Darcy friction factor
r_{pipe}, P, A	Pipe radius, perimeter, and area
COV_m,C_{vm}	Covariance parameters
h_m, h_l, h_g	Enthalpy (mixture, liquid, gas)
k_m, k_l, k_g	Thermal conductivity (mixture, liquid, gas)
c_{pm}, c_{pl}, c_{pg}	Specific heat capacity at constant pressure (mixture, liquid, gas)
q_w	Wall heat flux, W/m^2
h_{gas}	Assumed constant gas enthalpy
γ, K_l	Ratio of specific heats of gas and bulk modulus of liquid
d_b	Bubble diameter
$\text{Nu}, \text{Re}, \text{Pr}$	Nusselt, Reynolds, and Prandtl numbers
HTC, FC	Heat transfer coefficient, flashing coefficient
A_i	Interficial area density
$\mathcal{Z}(a_1, a_2, \alpha)$	Interficial area modification function with parameters a_1, a_2
$h_{l,sat}, h_{g,sat}$	Saturation enthalpy (liquid, gas)
$N_{e,z}, N_{e,t}$	Number of spatial and temporal elements
$\mathcal{P}^p$	Space of p^{th} order polynomials
$\hat{\mathbf{F}}, \mathbf{u}^R, \mathbf{u}^L$	Interface flux, state on the right and left of the interface
$V\Lambda V^{-1}$	Eigenvalue decomposition of the flux Jacobian ($d\mathbf{F}/d\mathbf{u}$)
$H(\cdot)$	Heavyside function
C_{11}, C_{12}, C_{22}	Stabilization coefficients for DG method
$\mathbf{u}_{ghost}, \mathbf{u}_{interior}$	For boundary conditions, the assumed exterior state and the current guess of the interior state
$\dot{m}_{spec}, h_{spec}, \alpha_{spec}, p_{spec}$	Specified boundary values
n_g	Number of integration points in a finite element
P_{phys}	Penalization to enforce physically correct solution
μ_{pen}	Penalization scale factor
$c_i(\mathbf{u})$	Constraint to enforce physically correct solution
$\mathbf{M}$	Spatial mass matrix

$_{h,H}$	Fine and coarse discretizations
$\mathcal{I}_H^h$	Injection operator from coarse to fine spaces
$\gamma_{num}, \gamma_{err}$	Fraction of elements and fraction of error to refine
ϵ_k	Error indicator for element k
τ_{remove}	Tolerance on singular values for removing a direction
$n_{\text{samp,min}}, n_{\text{samp,step}}$	Number of samples to consider for error metric (minimum and step size)
$n_{\text{pso,pop}}, n_{\text{pso,maxiter}},$	Parameters for particle swarm optimization (number of particles,
$n_{\text{pso,samp}}$	iterations, and samples for computing the error metric)

ABSTRACT

Building Fully Adaptive Stochastic Models for Multiphase Blowdown Simulations

by

Isaac Asher

Chair: Krzysztof J. Fidkowski

A new method for uncertainty quantification (UQ) that combines adaptivity in the physical (deterministic) space and the stochastic space is presented. The sampling-based method adaptively refines the physical discretizations of the simulations, along with adaptively building a stochastic model and adding samples. UQ studies can be very expensive due to complex physics requiring large physical solutions and due to a large number of parameters which results in a very large stochastic space to explore and model. The new UQ method can result in lower errors and lower cost by balancing different sources of error. By adaptively refining the physical and stochastic models, an overall prescribed error level can be reached without overly excessive and costly accuracy in either space.

The UQ method takes advantage of an active linear subspace to reduce the dimensionality of the stochastic space while retaining relevant interaction terms and anisotropy. Driven by low-cost error estimates, a particle-swarm optimization method explores the stochastic space and drives adaptation that results in an efficient stochastic approximation. The UQ method is compared to to two modern methods for three test functions in a 100-dimensional space. The current method is shown to result in up to three orders of magnitude lower error and up to two orders of magnitude fewer samples. Next, a simulation is developed using the discontinuous Galerkin method which is well-suited to adaptivity. A transient

multiphase flashing flow model is used to simulate the Edwards-O'Brien blowdown problem which is relevant for loss of coolant accidents in nuclear reactors. Details are included for adjoint-consistent treatment of gradient dependent sources, non-linear equations of state, and boundary conditions (including choking). The adjoint equations are successfully solved and used to drive a space-time anisotropic adaptation based on a complex output of interest. This results in an efficient phsyical discretization. Finally, the UQ method is used to assess modeling and discretization errors in a modified multiphase flow simulation. Based on an overall stochastic output of interest, the UQ method simultaneously drives adaptation of the stochastic and deterministic discretizations in order to balance the two sources of error. That is, terms are added to the stochastic model, samples are added, and the physical grid of each individual simulation is refined simultaneously. Error estimates based on semi-refined discretizations retain anisotropic accuracy, and a common grid is used to compare solutions from samples. The method for combined adaptivity performs well on the test problem, reducing the stochastic dimensionality from 20 to two and reducing deterministic errors on select samples. For about the same computational time, the method results in an order of magnitude less error and an order of magnitude fewer degrees of freedom compared to three other methods.

CHAPTER 1

Introduction

1.1 Motivation

Simulating physical systems offers many potential advantages over experimental methods for design, optimization, and safety analysis. While allowing for cheaper data collection and faster turnaround time, the transition from the real-world to the computational world introduces errors due to inadequate modeling, imperfect numerics, and incomplete knowledge of inputs and operating conditions. In addition, some problems are too large to handle on even the biggest computers to date. While computing power does increase over time, so does the complexity of problems tackled, and hence it is important to develop more efficient simulations that give higher accuracy and run faster. One common problem is that we simulate too much. That is, without knowing how much fidelity is needed in a simulation, we add as many degrees of freedom as possible according to some a-priori insight, with the constraint that the simulation finishes within a given time frame. In some problems, fewer degrees of freedom could be arranged to provide much more accuracy if they are concentrated in important regions of the solution domain. Adaptive methods that automatically and appropriately distribute fidelity in a simulation have enjoyed much success in recent years.

A second problem is that we cannot trust the numerical accuracy of simulations. While most simulations are convergent, meaning that the numerical solution approaches the true solution of the model as the fidelity is increased, very few simulations provide error bars

for each solution. The error in a solution due to insufficient fidelity is called discretization error. In addition, many simulations rely on approximate models that introduce unknown errors. The error in a solution due to imperfect models is called modeling error. In order to trust a simulation result, one must quantify these two sources of error – discretization error and modeling error. Perturbation methods, which analyze error propagation by linearizing the problem, have been used with good results for quantifying discretization errors. This is because many (most) simulations rely on a relatively simple, linear decompositions of the solution, for example as a linear combination of basis functions. Applied to modeling error, this is called sensitivity analysis. However, modeling errors need not obey any simple form. Indeed, if modeling errors were linear, they would be easy to correct as one could simply include another linear term in the model! Rather, modeling errors are generally complex and non-linear, so sensitivity analysis is often inadequate. More accurate quantification of modeling errors requires a fuller exploration of the behavior of the models.

One way to explore how models behave is to select a number of parameters in those models and analyze how the solution changes with the parameters. Often, this analysis is performed to statistically quantify variations due to uncertain inputs, so the parameters are treated as random variables and the solution is a function of the random variables. In any case, exploring how the solution changes with modifications to a large number of parameters is a hard problem. Each parameter becomes a dimension and we must explore a function of a high-dimensional space (sometimes called the stochastic space). As the number of parameters increases, the size of this space grows exponentially, rendering most analysis methods intractable. This is called the curse of dimensionality.

Many methods have been developed to quantify modeling errors by exploring (and modeling) the stochastic space. Methods that perform the best for a large number of parameters automatically and adaptively tailor their exploration and modeling to the simulation at hand. For example, if the response of the simulation to the parameters only varies over a subset of the parameters or over some region in the stochastic space, great savings can be achieved;

otherwise intractable problems can become tractable.

In this work, we develop a new method that extends ideas of perturbation analysis and adaptivity to build efficient models in the stochastic and physical spaces. This is done by estimating and then balancing stochastic errors and discretization errors. By separating the two sources of errors, the stochastic model and the physical grid(s) are simultaneously adapted, each to the level required in order to balance out the sources of error. Having balanced errors ensures that the approach is computationally efficient. It does not make sense to have an extremely fine grid and just one or two samples in stochastic space – even though the discretization error is small, the stochastic error would be very large and the results almost useless. The same applies to a very coarse grid and many samples in stochastic space – while the stochastic space may be fully explored, each sample has so much error that little information is actually gained. Some work has been done to quantify both sources of error in an uncertainty quantification study, and to reduce one or the other (e.g. add samples until the stochastic error is about the same as the discretization error). However, to the author's knowledge, there has yet to be a fully adaptive UQ study that dynamically balances both sources of error by adapting the stochastic and physical approximations. This is the goal of the current work.

To this end, the current method combines previous methods for error estimation and adaptation in both spaces. Most methods developed for one domain are not compatible with those of a different domain, so many of the contributions of this work are related to developing consistent, compatible error estimates. The output-based (or goal-oriented) framework is used because all errors are measured by their effect on a single, scalar "output of interest." This output is assumed to be defined at the outset of the UQ study and serves as a common basis for measuring errors[1]. The resulting error estimates also have the advantage of being accurate but relatively inexpensive to evaluate.

In order to facilitate the adaptive process, we use a stochastic model and a physical dis-

[1] Alternatively, one can view the adaptive process as generating an approximation whose goal is to accurately predict the output.

cretization that allow for localized error estimates and localized adaptation. The stochastic model is based on directions in stochastic space, and can be adapted by increasing fidelity along certain directions (including adding samples along the directions). Since the number of possible directions is very large, selecting just a few with the most error for adaption can result in an efficient stochastic model. The physical discretization is discontinuous across elements, which facilitates localizing errors to individual elements. Then, only those elements with the most error can be refined, resulting (after a number of iterations) in an efficient grid. Errors are balanced by allocating new degrees of freedom for adaptation in each space according to the fraction of error due to that space (found from separate error estimates for each space).

The method for stochastic modeling can be used with any parameters of the simulation, not just those in approximate models. For example, using parameters that account for variability in manufacturing or in operational conditions yields a better quantification of performance of a device. Design parameters can be included to explore a design space. The approach used here is quite general and applies equally to many situations once one defines the goal of the analysis.

1.2 Background

Simulations of complex phenomena often require approximate models derived from experimentation or simplifications of physics. Most models have parameters that control their behavior and have nominal values derived from experiments, analysis, or expert judgment. Some simulations have few if any parameters; for example, direct numerical simulation of fluids uses only governing equations derived from first principles. When approximate models are used to decrease complexity, though, errors are introduced into the results of the simulation. Quantifying these errors is important for making engineering decisions based on simulations. In order to quantify these errors, one must essentially re-run the simulation for every possible combination of parameters. Another way to look at this is that the simulation

is a function of the parameters; the parameters become dimensions in a stochastic space. The uncertainty quantification (UQ) study can be thought of as a way to model the simulation output as a function of the parameters, i.e. as a high-dimensional function approximation.

Many UQ methods can be described in terms of high-dimensional function approximation. For example, Monte Carlo methods generate a random set of samples and use them to approximate the function or its moments. Polynomial Chaos methods model the function as tailored polynomials along the dimensions (or some subset of the directions for an adaptive method) [119, 72]. Support Vector Regression models the function as a set of S-functions (which transition from a low value to a high one, for example $\mathrm{erf}(x)$) along the support vectors [106]. The biggest challenge in UQ studies is that as the dimensionality of the stochastic space increases (as the number of parameters one desires to consider increases), the number of samples required and hence the computational time usually increases exponentially. This is called the curse of dimensionality.

On the other hand, simulations usually also require a discretization of physical space into computational elements (or cells). As the number of elements increases, so does the accuracy and the cost. Any particular discretization and element size results in discretization error (which would be zero if infinitely many elements were used). An efficient way to build a discretization would be to have small elements in some parts of the domain (where the solution is complex) and larger elements elsewhere (where the solution is simple). This can be done by starting with a coarse grid (all large elements), computing localized error estimates, and using them to selectively refine the grid.

The curse of dimensionality requires an approximate stochastic analysis (a full analysis would require very many samples), resulting in stochastic error. The overall UQ study, thus, has stochastic error (from the approximate stochastic model) and discretization error (from the chosen grid). The goal of the current work is to balance these sources of error and adaptively refine both the grid and the stochastic model. This can result in a UQ study with higher accuracy that takes less time to execute compared with non-adaptive methods.

The following two subsections describe the particular methods chosen for the stochastic models and the physical discretization. The methods are put in context with other state-of-the-art methods, and we give reasons why they were chosen for inclusion in the combined adaptivity approach. The reasons are, generally, availability of accurate, localizeable error estimates, ability to generate a compact but accurate model (when possible), and ease of incremental adaptation. Next, we introduce the output-based or *a-posteriori* error estimation framework. Finally, we give some background regarding the choice of simulation to which we will apply the fully adaptive UQ study, namely a multiphase flow simulation.

1.2.1 Stochastic Modeling

Almost all modern UQ methods are predicated on finding some way to get around the curse of dimensionality. This is also important in the current framework in order to have a stochastic approximation that does not require a huge number of samples. Most UQ methods are designed to overcome the curse by assuming that the underlying function has some particular structure which, if it does, allows the computation to proceed much more quickly. For example, Polynomial Chaos methods work very well if the underlying function can be well-approximated by just a few of the selected polynomials [119]. As higher order polynomials are required, the expense rises. One attractive class of methods is those that use an active linear subspace. These methods assume that the function only varies within a linear subspace of the full high-dimensional space. By assuming that the function does not vary in the inactive subspace, the expense of the computation can be diminished considerably. Examples of methods with an active linear subspace include Support Vector Regression [106], where the support vectors define the active subspace, and Orthogonal Matching Pursuit [88, 87], where the atoms with nonzero coefficients define the subspace. Other related methods are ridgelet regression [120, 15, 121] and subspace pursuit [30, 113] The contribution of the current method is that it adaptively restricts modeling in two ways: by identifying an active subspace and by restricting interactions within that space.

The method developed in this work can also be viewed as a Reduced Order Model (ROM) [101] with a reduction in the size of the input space and the state space. We follow the methodology of adaptive model building to increase the accuracy of the ROM (while also increasing the number of samples) until a desired tolerance is reached. This is different than strategies such as that in [100, 10] where the full, high-order ROM is constructed all at once. The adaptive methodology may result in a less optimal ROM, but has the advantages of giving a partial answer with a restricted number of samples and providing a natural way to update the model as new samples are added. The particular form of the ROM reduces the input space to an active linear subspace. The state is also reduced to vary in a polynomial way within the active linear subspace. This becomes a surrogate model [101] for the high-fidelity simulations. Given enough terms (and samples), the surrogate model can grow to become a full polynomial representation of the high-fidelity simulation over the parameters. This ensures that the method will eventually produce a very accurate ROM, assuming that the high-fidelity simulation, while non-linear, has a convergent Taylor series representation in the stochastic space.

A second difference between the current method and that in [100, 10] is the way in which the active linear subspace is detected. Their method uses a set of local gradient calculations to approximate the global gradient information. In this work, the same approach is used except that we do not require gradients with respect to parameters to be calculated directly. Instead, output-based (or other) error estimates are used to gather the same information. The error estimates do require gradients to be calculated, but not with respect to parameters[2]. This eliminates the need for either analytic differentiation (with respect to the parameters) or approximate finite difference calculations that require many function evaluations. Of course, if the analytic derivatives are available, both methods could use them and

[2]The adjoint requires gradients of the residuals of the governing equations with respect to the solution, and the output with respect to the solution. However, we do not require gradients with respect to the parameters. The major expense for the adjoint method is in calculating the gradients of the residuals with respect to the solution, but this is also needed for implicit time stepping of the solution. Gradients with respect to parameters may be difficult to implement when there are many parameters. Many UQ methods, though, could be extended to take advantage of this information and be more efficient.

incur approximately the same expense.

In order to identify directions in parameter space in which there is no variation, the current method uses error estimates that approximate the gradient. This method of identifying directions with no variation only models the function with linear terms, so it is not good at identifying anisotropy within the active subspace. Projection pursuit methods are better at finding a single direction in which there is large variation, that is, in identifying anisotropy within the active subspace. Thus, directions of high variability are found by searching *within* the active subspace using a projection pursuit approach. The resulting directions in which variability is detected are added to the surrogate model. Thus, the current method combines gradient sampling and projection pursuit to quickly neglect the inactive directions and focus on anisotropic behavior within the active subspace.

The statistical model thus developed is essentially a surrogate model. Here we focus on adaptively building a good surrogate model, without necessarily utilizing any statistics (although many successful UQ methods work directly with statistical quantities). The surrogate model approach is advantageous because it yields accurate error estimates for general outputs. The adjoint-weighed residual formulation requires a functional form representation of the state and adjoint. Having a functional form allows for accurate error estimates for any output of interest, whereas statistical methods usually focus on accurately computing just a few metrics of the output probability distribution (e.g. the mean and variance). In building the surrogate model model, it is important *not* to take a greedy approach and strictly increase the size of the active linear subspace. It is known [79, 32, 88] that errors due to the projection into the active linear subspace require the algorithm to add and remove directions from the active linear subspace[3].

We develop a method in this work which adaptively seeks and models the active linear subspace. The new method is based on a Matching Pursuit framework (see [53, 41] for early

[3]That is, given that the underlying function really does only vary in an active subspace, the subspace can only be discovered if the algorithm is allowed to add and remove directions from its guess of the active linear subspace.

developments and [37] for a modern output-based method), similar to the active subspace methods [100, 10, 28]. The function is modeled as a sum of terms, each term accounting for variation in a single direction. Most projection pursuit methods search for a direction in which the current model results in stochastic errors. That is, they search for large residuals in stochastic space. The contribution of the current method is that we utilize physical and stochastic error estimates to search for the active subspace and to optimize the model. Some methods model the responses or the error in stochastic space, requiring them to be in some functional form. In this work, modeling is restricted to the state (and adjoint), so fewer assumptions on the functional form of the responses and other quantities are needed. The model is built in an adaptive fashion so that the model is updated incrementally as more samples are taken.

Sampling methods work with fully converged simulations at single points in parameter space; with an existing simulation code, one simply fixes the parameters and runs the code. Non-sampling based methods modify (and usually expand) the governing equations to solve for behavior in the physical and parameter spaces at the same time. For example, stochastic finite element methods [46] create a mesh in both the stochastic and physical domains and solve in both. We focus on a sampling-based method so that the current method can be readily applied to existing simulations. In addition, adaptivity is easy for sampling-based methods as one can simply add more samples to improve accuracy. Since we are using error estimates to drive the stochastic adaptivity, it is assumed that there is some available way of computing an error estimate given a physical solution (parameters and sample values). For example, this can be an adjoint-weighted-residual error estimate for a finite-element code with some defined output of interest [114]. The method performs even better with an error estimate that is fast to compute.

Overall, the stochastic modeling method developed in this work has the ability to get around the curse of dimensionality by being compact (requiring few samples). The portion of the stochastic domain that is modeled is found using error estimates, and in this way

the model is adaptively built. Finally, sampling-based model is easily updated when new samples or terms to the model are added. Thus, the active subspace method is a good choice for inclusion in the fully adaptive UQ framework.

1.2.2 Multiphase Flow Discretization

Multiphase flow problems have been discretized with a variety of methods include finite difference, finite volume, and finite element methods. The former two are low order methods are advantageous because they execute quickly and are easy to couple with other methods for simulating other parts of a domain. Higher order methods, for example finite element methods, sometimes require more computational time for a given problem. However, they are advantageous because they offer the promise of lower error with fewer degrees of freedom. This often must be achieved through both fast solution techniques and automatic grid refinement. In the end, any discretization method can lead to a fast and accurate solution if care is taken. In this work, a discontinuous Galerkin finite element discretization is used because the author has experience with it and it is simple to use with adjoint-based error estimation and adaptation.

The discontinuous Galerkin finite element method [25] (DG FEM) begins by breaking up the domain into disjoint regions called elements. The solution to the governing equations is approximated as a polynomial (of variable order p) inside of each element (of size Δz). The solution is not forced to be continuous across elements, so there may be a "jump" at the element interfaces (hence *discontinuous* Galerkin). Residuals, $\mathbf{R}$, are defined as weighted integrals of the governing equations. This results in a non-linear system of equations $\mathbf{R}(\mathbf{u}) = \mathbf{0}$ (since the method is Galerkin, the test and basis functions are the same, yielding a unique solution). An iterative method such as Newton-Raphson is used to solve the non-linear system. The solution generally converges at a rate of $O(\Delta z^{p+1})$ for viscous problems. Thus, the advantage of a high-order discretization is faster mesh convergence. The discontinuous nature of DG FEM, along with its interpretation as a variational method, lends itself to

adaptive grid refinement.

Overall, the physical discretization has localizeable error estimates (due to the discontinuous nature of the approximation) that are known to be well-suited for grid adaptation. The grid also allows for simple adaptation by splitting elements. With a coarse enough starting grid and many iterations, the resulting adapted grids can be highly specialized and efficient for a given problem and highly accurate. Thus, the discontinuous Galerkin method is a good choice for inclusion in the fully adaptive UQ framework.

1.2.3 A-Posteriori Error Estimation

The fully adaptive approach to uncertainty quantification in this work is driven by *a-posteriori* error estimates. That is, once a solution has been found, a post-processing step is performed to determine how much numerical error is present in the solution. This error is then localized to particular components of the solution approximation (e.g. mesh cells or elements) which are modified (adapted) to give increased fidelity. Such error estimates have been developed for a variety of problems and contexts [38, 4]. In some cases, simple error estimates using residuals of the governing equations or interpolation error estimates are sufficient for driving adaptation. In other cases, a more detailed analysis may be necessary.

One type of error estimate is developed by a perturbation analysis. Suppose we run a simulation with the value of some parameter μ and get the result $\mathbf{u}$, a $[N_u \times 1]$ vector of field (or state) variables. The solution $\mathbf{u}$ satisfies the N_u governing equations $\mathbf{R}(\mathbf{u}, \mu) = \mathbf{0}$. Denote by $K(\mathbf{u})$ a quantity or output that we are interested in computing to high accuracy. We are interested in how different a solution with a modified parameter would be (denote by $\tilde{\mu} = \mu + \delta\mu$ the modified parameter). That is, we would like to estimate the parameter sensitivity $\delta K = K(\mathbf{u}(\mu + \delta\mu)) - K(\mathbf{u}(\mu))$.

Begin by linearizing the governing equations and the output definition:

$$\text{Gov. Eqns.:} \quad \mathbf{R}(\mathbf{u},\mu) = 0 \quad \rightarrow \quad \underbrace{\frac{\partial \mathbf{R}}{\partial \mathbf{u}}}_{N_u \times N_u} \underbrace{\delta \mathbf{u}}_{N_u \times 1} + \delta \mathbf{R} = \mathbf{0} \tag{1.1}$$

$$\text{Output:} \quad K = K(\mathbf{u}) \quad \rightarrow \quad \underbrace{\frac{\partial K}{\partial \mathbf{u}}}_{1 \times N_u} \underbrace{\delta \mathbf{u}}_{N_u \times 1} = \delta K \tag{1.2}$$

The linearized equations are matrix equations. Now, if one examines what happens after perturbing the parameter $\mu \rightarrow \tilde{\mu}$, there will be a perturbation in the residuals, $\delta\mathbf{R} = \mathbf{R}(\mathbf{u},\tilde{\mu}) - \mathbf{R}(\mathbf{u},\mu) = \mathbf{R}(\mathbf{u},\tilde{\mu})$ (since $\mathbf{R}(\mathbf{u},\mu) = \mathbf{0}$). The solution to the perturbed equations is $\tilde{\mathbf{u}} \equiv \mathbf{u} + \delta\mathbf{u}$, and the perturbation in the quantity of interest is δK. The equations above relate these perturbations to one another. Now we can formally compute the relationship of δK to $\delta\mathbf{R}$. First, rewrite Eqn. 1.1 as

$$\delta \mathbf{u} = -\frac{\partial \mathbf{R}}{\partial \mathbf{u}}^{-1} \mathbf{R}(\mathbf{u},\tilde{\mu}) \tag{1.3}$$

Then substitute into Eqn. 1.2

$$\delta K = \frac{\partial K}{\partial \mathbf{u}} \delta \mathbf{u} = -\underbrace{\frac{\partial K}{\partial \mathbf{u}}}_{1 \times N_u} \underbrace{\frac{\partial \mathbf{R}}{\partial \mathbf{u}}^{-1}}_{N_u \times N_u} \underbrace{\mathbf{R}(\mathbf{u},\tilde{\mu})}_{N_u \times 1} \tag{1.4}$$

This is the general sensitivity relationship. It can be used to calculate regular or "forward" sensitivities by testing various parameters and computing δK for each one. This can be computationally expensive for many parameters and few (e.g. one) quantities of interest, due to the inversion of the residual Jacobian matrix $\partial\mathbf{R}/\partial\mathbf{u}$.

Alternatively, one can pre-compute the first two parts to form a notional "$\partial K/\partial\mathbf{R}$" vector, which can then be applied to a large number of parameters. This can reduce the computational expense because the Jacobian need only be inverted once per quantity of

interest. The vector is called the adjoint solution, or simply the adjoint, $\boldsymbol{\psi}$:

$$\boldsymbol{\psi}^T \equiv -\frac{\partial K}{\partial \mathbf{u}} \frac{\partial \mathbf{R}}{\partial \mathbf{u}}^{-1} \quad \rightarrow \quad \delta K = \boldsymbol{\psi}^T \mathbf{R}(\mathbf{u}; \tilde{\mu}) \tag{1.5}$$

Rewriting the above definition as a matrix equation makes it clear how to solve for the adjoint vector:

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}}^T \boldsymbol{\psi} = -\frac{\partial K}{\partial \mathbf{u}}^T \tag{1.6}$$

Solving for the adjoint requires a regular matrix-vector solve, but with the transpose of the Jacobian. The Jacobian is commonly both available and inverted when solving the original equations with an implicit method. The adjoint solve then does not require much extra code to be written. For explicit methods, or Jacobian-free implicit methods, automatic differentiation or the complex step method [82] can be used to compute the required derivatives. In fact, adjoint solutions are becoming available in commercial simulation software [91]. Normally, every time we solve for $\mathbf{u}$, we also perform one more matrix inversion to solve for $\boldsymbol{\psi}$. Once the adjoint is computed, we arrive at the final method for calculating the perturbation in the quantity of interest

$$\delta K = \boldsymbol{\psi}^T \mathbf{R}(\mathbf{u}, \mu + \delta\mu) \tag{1.7}$$

The perturbation method applies equally well when the "parameter" is in fact the computational grid and its perturbed value is a high-fidelity grid. In this case, δK gives an estimate of the difference between the output on the original and high-fidelity grids, that is an estimate of the discretization error. Practically speaking, once the adjoint is computed, one must simply generate a finer grid and compute the high-fidelity residual of the known (lower-fidelity) solution.

The adjoint approach just developed is based on a discrete system of equations but can be extended to the underlying continuous governing equations. This results in the continuous

adjoint equations which are PDEs in the same form as the governing equations (though they are always linear). In fact, if the governing equations and the discretization satisfy certain properties, then the discrete adjoint equation 1.6 is actually a consistent and convergent discretization of the continuous adjoint equations [49, 6, 90]. In Section 3.4.2, we give a sketch of the continuous adjoint derivation and use it to motivate an extension to the stochastic space.

One advantage of the adjoint approach for error estimation is that it has a good theoretical basis, at least for variational discretizations, and does not require heuristics. Another advantage is that it can identify error propagation, where errors in the solution in one part of the domain affect an output that may a function of states in a different part of the domain. This can be especially important for simulations of transport phenomena.

A disadvantage of the adjoint approach is that it requires extra computational expense and memory usage (especially for time-dependent problems). Sometimes, though, this can be outweighed by the savings resulting from efficient computational grids generated through adaptation. A second disadvantage is that the approach only takes into account first order sensitivity information from the governing equations. For highly nonlinear equations and solutions on very coarse grids, this can result in inaccurate error estimates and poor adaptive choices. One solution is to use second order adjoints, which requires more analysis, coding, and computation time. Still, in many instances the first order information is enough to drive adaptation (which tolerates inaccurate error estimates to some extent). In this work, the error estimates resulted in reasonable adaptation choices with good reduction in errors over many steps of the process.

In this work, the adjoint approach requires defining one or more (scalar) *statistical* outputs of interest. That is, one must define the goal of the entire UQ study at the outset. At each adaptive step, all of the approximations are refined in an attempt to minimize the error in the stochastic output. This is essentially an extension of the adjoint techniques to the stochastic domain [35]. The stochastic output serves as a consistent measure of error

across the stochastic and physical domains. Separate (but comparable) error estimates for each domain are used to decide whether to focus adaptation in the stochastic domain or the physical domain.

1.2.4 Simulating Multiphase Flow

In order to demonstrate the fully adaptive UQ method, an application should include many approximate models and parameters (whose effects will be quantified). This usually results from simplified models of complex physical phenomena. The large range of physical scales and complexity involved in multiphase flow, together with the need for simulations of large systems like nuclear reactors, has driven much research into developing simple models that are cheap and approximate and often have many parameters. One group of these simplified models attempts to model flow through a pipe as a one-dimensional problem. While some models simplify detailed, three-dimensional equations of motion, others rely on correlations from large experimental data sets. This work focuses on a one-dimensional "drift–flux" formulation, which is based on space and time averaging of the three-dimensional flow field and includes empirical correlations for some parameters. Specific to two-phase drift-flux models is the reduction of two momentum equations to one by specifying a correlation for the relative velocity between the phases.

Ishii & Hibiki derived a four-equation drift-flux model and presented a number of correlations for the resulting parameters for various void regimes for cylindrical, annular, rod bundle, and pool boiling scenarios [56, 57]. The void fraction and inter-phase drift are related in this formulation, so correlations for drift velocity must also take into account boiling regimes. A variety of models for drift velocity and void fraction have been proposed since then. Eight models are compared in [23], and more mechanistic-based models are derived in [51]. Various models are studied for the case of flashing flow in [80]. Thirteen models are compared against a wide range of experimental data in [26]. In general, complex models were found to be necessary for reasonable prediction accuracy. In this work, a more mechanistic

approach is taken in which the drift velocity and void fraction correlations are decoupled, as is done in some of the proposed models. The drift velocity is related solely to the flow parameters, and the void fraction is related to the boiling parameters.

The numerical characteristics of the drift-flux formulation have also been investigated. The loss of hyperbolicity in the governing equations is often a source of numerical instability. In [105], stability was shown to be enhanced by special choices of some parameters, though not proven in general. A special form for the drift velocity was derived in [47] such that the first first order terms would always remain hyberbolic, ensuring stability for many standard numerical schemes. An approximate Riemann solver for the drift-flux equations was proposed in [40], and it conformed to two of the three conditions for a "Roe" type solver. The third condition of hyperbolicity, though, was not proven in all cases. Others have used more general Riemann solver formulations, which often involve more approximations, e.g. [17, 16]. In this work, a linearized exact Riemann solver is used to upwind the inviscid fluxes. This requires solving a 4-by-4 eigenvalue problem at each interface, but the expense of this step is quite minor, considering that the analytic flux-Jacobian is used. In addition, errors due to the linearization are generally made small when adaptation and higher-order approximations are used.

The large range of spatial (and temporal) scales inherent in multiphase problems often demands methods that are accurate at many of these scales. All of the methods discussed above are confined to the largest physical scales in order to make them computationally inexpensive. Certainly much research has been done in more detailed (smaller scale) simulations, generally of smaller problem sizes due to limits on computational resources. The possibility of attacking a large problem size with targeted small-scale simulation is attractive, and would lead to improved predictability. The goal of adaptive and multiscale methods is precisely this, and it is achieved by carefully combining methods for various spatial scales. In this work the goal is to show the possible advantages of such methods on a model problem. Using a basic adaptive method, the physics models are kept constant but the discretization

is adapted to the solution. In particular, the adaptation is driven by a goal-oriented (or adjoint) method, where a single "output of interest" is computed to a given accuracy. A review of these methods is presented in [38], which highlights that full mesh convergence of outputs is often not achieved on standard meshes used in industry. Indeed, this author has found this to be the case in multiphase flow models as well, see [34] and Chapter 2. An adjoint-based adaptation procedure is developed in Section 4.5 and applied to the multiphase flow problem.

In the interest of simplifying the calculations, some correlations are used in ranges in which they are invalid. We will use some correlations for the drift velocity and the interphase mass transfer that assume bubbly liquid even when the void fraction is large and the flow is in the churn or droplet regimes. The goal here is to demonstrate the potential savings in computational time from adaptive methods. More complex and accurate simulations (e.g. realistic 1D, full 3D including turbulence, etc) can often reap even more benefits from adaptivity than are demonstrated here. This work is not concerned with developing a drift-flux model that fits well with experimental data. Rather, we are looking to use a flow model that is complex enough to capture interesting physics and include many parameters, yet simple enough to have relatively quick execution. The one-dimensional drift-flux model of multiphase flow fulfills these requirements.

In conclusion, a fully adaptive method for uncertainty quantification will use an active subspace stochastic model and a discontinuous Galerkin discretization of one-dimensional multiphase flow equations. Output-based error estimates from both models (and spaces) will be separated to drive adaptivity in the models. Both models allow for targeted adaptation which can lead to tailored, low-error approximations. As the adaptive iterations proceed, stochastic errors will be balanced with discretization errors, resulting in a UQ study that is relatively accurate yet inexpensive to execute.

1.3 Contributions

The contributions of the adaptive method for uncertainty quantification developed in this work are

- coupled adaptivity in the stochastic and physical spaces to control and balance stochastic and discretization errors simultaneously, potentially leading to large cost savings for UQ studies of complex simulations
- a UQ method that exploits functions with a small active subspace, a low level of interaction within that subspace, and high anisotropy within that subspace
- utilization of both full solutions and low-cost error estimates to explore and model the parameter space; in particular, adjoint-based error estimates are used to detect the active subspace and explore it
- an output-based UQ method that only models the state and adjoint over the parameter space, but does not assume any functional form of the physical output or the error estimate
- a new stochastic error metric that is based on directions in the stochastic space, is cheap to compute, and can be used for adaptation in the stochastic space
- an transient, multiphase, higher-order Discontinuous Galerkin simulation with adjoint solutions, a-posteriori error estimates, and adjoint-based adaptation
- demonstration of the cost-savings of the fully adaptive UQ method for a relatively simple multiphase flow simulation

1.4 Thesis Overview

This work is organized as follows. In Chapter 2, standard methods for uncertainty quantification in multiphase flow are used to investigate subcooled boiling models in the widely

available Star-CD and Nphase simulation packages. The results motivate the need for a more adaptive and accurate method. In Chapter 3, a new method for adaptive uncertainty quantification is developed based on active subspaces. The new method is compared against other modern UQ methods. A transient, one-dimensional multiphase flow model is solved via the discontinuous Galerkin method in Chapter 4. The solution method is extended to solve for the adjoint and automatically adapt the grid. The Edwards' blowdown problem is solved with this method. In Chapter 5, the adaptive uncertainty quantification method is combined with the adaptive multiphase simulation. The result is an uncertainty quantification method that is fully adaptive in both deterministic and stochastic spaces. The method is applied to assess model and experimental uncertainties in the blowdown simulation. Finally, Chapter 6 discusses conclusions and directions for future work.

CHAPTER 2

UQ for Multiphase Flow Problems

2.1 Motivation

The objective of this chapter is to quantify sensitivities of computational thermalhydraulics outputs to parameters in boiling, multiphase flow, and turbulence models. The motivation for this work comes from the observation that most CFD boiling and multiphase models rely on correlations with empirically-determined parameters. These parameters offer flexibility when matching theory to experimental data, which are generally limited and available for only a handful of geometries and conditions. The parameters then become a liability when simulating novel designs or conditions, as errors due to mistuning are generally not quantified.

Knowledge of sensitivities of CFD results to these tunable parameters can aid uncertainty quantification (UQ) studies by effectively reducing the dimension of the parameter space. That is, parameters that do not strongly affect outputs may not need to be considered in the UQ studies. In addition, combined with estimates of parameter variability, sensitivity information can guide model improvement by identifying key parameters and associated models to which outputs are most sensitive.

In this chapter, we examine two widely available multiphase simulation codes and their ability to model an established benchmark problem that has experimental data (the DEBORA experiments [44]). The two codes are Star-CD, a commercial package developed by CD-

adapco [19], and Nphase, a research code developed at Rensselaer Polytechnic Institute [74]. UQ studies are performed for the various multiphase model parameters. Thousands of runs of the simulations are required to complete the UQ studies, though the useful results have relatively little quantitative data. This motivates the use of more precise and adaptive UQ methods; we develop such a method in Chapter 3. In addition, difficulties with numerical errors in the simulations are encountered. Numerical errors can go unquantified in Star-CD and Nphase, potentially polluting the UQ study. This motivates the error estimation and adaptive techniques used for the multiphase flow simulation in Chapter 4.

2.2 Star-CD Study

2.2.1 Approach

Numerous models exist for simulating multiphase flow, boiling, and turbulence. A comprehensive treatment of all possible formulations is beyond the scope of this work. Instead, we choose to focus on a subset of models that are relatively standard and representative of those used in thermalhydraulics applications. Specifically, we use the models implemented in the commercial software package, STAR-CD from CD-adapco.

STAR-CD employs an Eulerian multiphase model that is representative of treatments in other commercial codes. The model does rely on empirical correlations, and these are included in the sensitivity study. Details on this model are given in the following section.

The boiling model is somewhat more contentious among codes, as many different theories and correlations exist. To the author's knowledge, however, the boiling model implemented in STAR-CD has been validated for certain benchmark test conditions. This model also relies on several empirical parameters, and the importance of these is investigated in the sensitivity study. In addition, sensitivity to the form of the boiling model is assessed by a parallel sensitivity study using a simpler model based on a single correlation.

Finally, the turbulence model implemented in STAR-CD is a high-Reynolds number $k-\epsilon$

model with a multiphase correction. Numerous closure parameters enter into this model, but for generality and computational tractability, we consider only high-level effects due to varying the computed turbulent eddy viscosity in both the fluid and the dispersed gas phase.

2.3 Star-CD Physical Models and Parameter Ranges

The physical models below are described as implemented in the STAR-CD program from CD-adapco. Additional information can be found in the STAR-CD methodology document [18].

2.3.1 Governing Equations

The equations governing mass, momentum, and energy transport in multiphase flow are

$$\frac{\partial(\alpha_k\rho_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k\mathbf{v}_k) = \sum_{j=1}^{N}(\dot{m}_{jk} - \dot{m}_{kj}), \tag{2.1}$$

$$\frac{\partial(\alpha_k\rho_k\mathbf{v}_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k\mathbf{v}_k\mathbf{v}_k) = -\alpha_k\nabla p + \alpha_k\rho_k\mathbf{g} + \nabla\cdot[\alpha_k(\boldsymbol{\tau}_k + \boldsymbol{\tau}_k^t)] + \mathbf{M}_k, \tag{2.2}$$

$$\frac{\partial(\alpha_k\rho_k h_k)}{\partial t} + \nabla\cdot(\alpha_k\rho_k\mathbf{v}_k h_k) = Q_k + \nabla\cdot\left[\alpha_k\left(\lambda_k\nabla T_k + \frac{\mu_t}{\sigma_h}\nabla h_k\right)\right], \tag{2.3}$$

where the phase-to-phase momentum and heat transfer sources are

$$\mathbf{M}_k = \mathbf{F}_{Dk} + \mathbf{F}_{TDk} + \mathbf{F}_{Lk} + \mathbf{F}_{VMk} + \sum_{j=1}^{N}(\dot{m}_{jk}\mathbf{v}_j - \dot{m}_{kj}\mathbf{v}_k), \tag{2.4}$$

$$Q_k = \alpha_k\frac{Dp_k}{Dt} + \alpha_k(\boldsymbol{\tau}_k + \boldsymbol{\tau}_k^t) : \nabla\mathbf{v}_k + \sum_{i\neq k} Q_{ki} + \sum_{(ik)} Q_k^{(ik)} + \sum_{i\neq k}\left(\dot{m}_{ki}h_k^{(ik)} - \dot{m}_{ik}h_k\right) \tag{2.5}$$

Quantities entering into these equations are defined as follows:

α_k	mass fraction of phase k
ρ_k	density of phase k
$\mathbf{v}_k$	velocity vector of phase k
$\dot{m}_{jk}$	rate of mass transfer from phase j to phase k

N	number of phases
p	pressure
$\mathbf{g}$	acceleration due to gravity
$\boldsymbol{\tau}_k$	laminar/turbulent viscous stress tensors for phase k
h_k	specific total enthalpy of phase k
λ_k	thermal conductivity of phase k
T_k	temperature of phase k
μ_t	turbulent viscosity
σ_h	turbulent thermal diffusion Prandtl number
Q_{ki}	heat transfer rate from phase k to phase i
$Q_k^{(ik)}$	heat transfer rate from phase k to the interface between i and k.

We restrict our attention to two-phase flows resulting from boiling, in which the two phases are liquid (l) and vapor (g). Wall boiling is accounted for by two processes. First, heat transfer from the wall to the liquid is added as a heat source. Second, the amount of gas generated at the wall due to boiling is added to $\dot{m}_{lg}$. Details of the wall boiling model are given in Section 2.3.5.

2.3.2 Phase-to-Phase Heat and Mass Transfer Parameters

When bubbles exist, heat and mass transfer between the liquid and vapor phases are linked by the following equations, summarized in Figure 2.1.

$$\text{Heat transfer from liquid to bubble interface } = \dot{q}_l = \text{HTC}_l A_i (T_l - T_{sat}) \tag{2.6}$$

$$\text{Heat transfer from vapor to bubble interface } = \dot{q}_g = \text{HTC}_g A_i (T_g - T_{\text{sat}}) \tag{2.7}$$

$$\text{Liquid-to-vapor mass transfer rate} = \dot{m}_{lg} = \dot{m} = \left(\dot{q}_l + \dot{q}_g\right) / \Delta h_{gl} \tag{2.8}$$

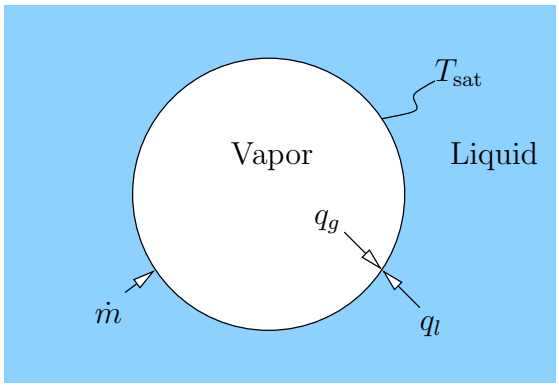


Figure 2.1: Depiction of phase-to-phase heat and mass transfer

There is no coefficient to adjust the mass transfer rate directly in Eqn. 2.8, e.g. for the purpose of a sensitivity study. Instead, the quantities that are available for adjustment are the relationships for the interfacial area concentration and the individual heat fluxes.

2.3.2.1 Interfacial Area (Bubble Size)

The interfacial area concentration is given by the Eqn. 2.9, where the representative bubble diameter is obtained in a manner similar to Kurul and Podowski [63].

$$A_i = \frac{6\max(\alpha_g, 1-\alpha_g)}{d_b}. \tag{2.9}$$

The expression for the bubble diameter d_b, is given by Eqn. 2.10, with baseline values of $d_{b,0} = 0.15mm, d_{b,1} = 2mm, \Delta T_0 = 13.5K$, and $\Delta T_1 = -5K$.

$$d_b = \frac{d_{b,1}(\Delta T_{\text{sub}} - \Delta T_0) + d_{b,0}(\Delta T_1 - \Delta T_{\text{sub}})}{\Delta T_1 - \Delta T_0} \tag{2.10}$$

For the purpose of the sensitivity study, a physical basis to bound the interfacial area concentration has not been obtained. Additionally, Eqn. 2.9 is not an empirical correlation, but is an exact expression for a domain consisting of bubbles of the same size. We therefore work with Eqn. 2.10, and without additional information on the calibrated baseline coefficients, we adjust these by $\pm 30\%$.

2.3.2.2 Interfacial Heat Transfer Coefficients

The liquid-to-interface and vapor-to-interface heat transfer rates are given in Eqns. 2.6 and 2.7, respectively. The heat transfer coefficient HTC_l is obtained from a Nusselt-number correlation due to Ranz and Marshall [97], which states

$$\text{Nu}_l = 2 + K_1 \text{Re}_d^{0.5} \text{Pr}_l^{0.3}, \qquad K_1 = 0.6(\text{baseline}). \tag{2.11}$$

A lower bound of 2 on this Nusselt number is physical (corresponding to pure conduction), however it is unclear how to place an upper bound – large Nu_l are physically attainable in certain flow conditions. Without such case-dependent in formation, for the sensitivity study, the coefficient 0.6 in the Ranz-Marshall model in Eqn 2.11 was adjusted by ±30%. A plot of the Ranz-Marshall correlation against data for evaporating water drops is shown in Figure 2.2. This plot shows that adjusting the coefficient in the Ranz-Marshall correlation by ±30% from 0.6 encompasses all of the data.

For the vapor-to-interface heat transfer coefficient, HTC_g, a fixed Nusselt number of 26 is used in the base model. For the sensitivity study the range of 2 to 30 was chosen in order to encompass the limit of pure conduction.

2.3.2.3 Summary

A summary of the heat and mass transfer parameters chosen for the sensitivity study, as well as their ranges, is given in Table 2.2.

Table 2.2: Summary of phase-to-phase heat and mass transfer parameters

Parameter	Adjustment	Explanation
d_b	±30%	None
K_1 in Ranz-Marshall	$K_1 = 0.6 \pm 30\%$	encompasses experimental data
Nu_g	$2 \leq \text{Nu}_g \leq 30$	Conduction sets lower bound, no basis for upper bound

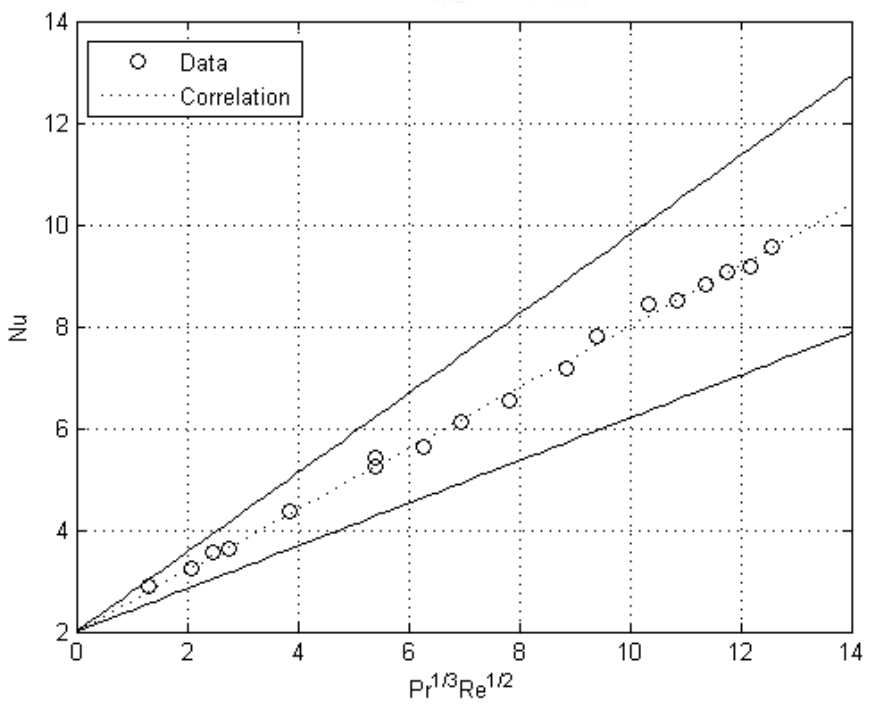


Figure 2.2: Ranz-Marshall correlation plotted against data for evaporating water drops. Error bands at ±30%.

2.3.3 Phase-to-Phase Momentum Transfer Parameters

The phase-to phase momentum transfer forces are given in Eqn. 2.4. These consist of (with subscript k dropped),

$$\begin{aligned} \mathbf{F}_D &: \text{drag force} \\ \mathbf{F}_{TD} &: \text{turbulent dispersion force} \\ \mathbf{F}_L &: \text{lift force} \\ \mathbf{F}_{VM} &: \text{virtual mass force} \end{aligned}$$

2.3.3.1 Drag Force

The bubble drag force is given by

$$\mathbf{F}_D = \frac{3}{4}\frac{\alpha_g \rho_l C_D}{d_b}|\mathbf{v}_r|,$$

where $\mathbf{v}_r = \mathbf{v}_l - \mathbf{v}_g$ is the relative velocity.

The bubble drag coefficient, C_D, is obtained using the correlation of Wang [116], given by Eqn 2.12 and the coefficients provided in Table 2.3. Note that Re_d is the Reynolds number based on the bubble diameter. Without additional details on the basis for adjustment of this correlation, for the sensitivity study, the drag coefficient is varied in a range of ±30% from the baseline value.

$$C_D = \exp\left[a + b\ln \mathrm{Re}_d + c(\ln \mathrm{Re}_d)^2\right] \tag{2.12}$$

Table 2.3: Parameters entering into the drag coefficient correlation due to Wang [116].

Re_d	a	b	c
$\mathrm{Re}_d < 1$	ln(24)	-1	0
$1 < \mathrm{Re}_d < 450$	2.699	-0.3358	-0.07136
$450 < \mathrm{Re}_d < 4000$	-51.772	13.167	-0.8236
$\mathrm{Re}_d > 4000$	ln(8/3)	0	0

2.3.3.2 Turbulent Dispersion Force

The turbulent dispersion force is given by a correlation derived by Burns [14],

$$\mathbf{F}_{TD} = -\frac{3}{4}\frac{\alpha_g \rho_l C_D}{d_b}|\mathbf{v}_r|\frac{\nu_l^t}{\sigma_\alpha}\left[\frac{\nabla\alpha_l}{\alpha_l} - \frac{\nabla\alpha_g}{\alpha_g}\right], \tag{2.13}$$

where σ_α is an empirical turbulent Prandtl number with baseline value of 0.9. In our sensitivity study we chose to adjust the turbulent dispersion force by varying σ_α by $\pm 30\%$.

2.3.3.3 Lift Force

The lift force is given by Eqn. 2.14

$$\mathbf{F}_L = C_L \alpha_g \rho_l \mathbf{v}_r \times (\nabla \times \mathbf{v}_r), \tag{2.14}$$

where the lift coefficient is obtained from a correlation due to Tomiyama. Figure 2.3 plots the result of this correlation for one choice of relative speed, $|\mathbf{v}_r| = 25$ cm/s, a conservatively high value. As shown, the lift coefficient is bound between -0.3 and 0.3 for bubble diameters up to 2mm.

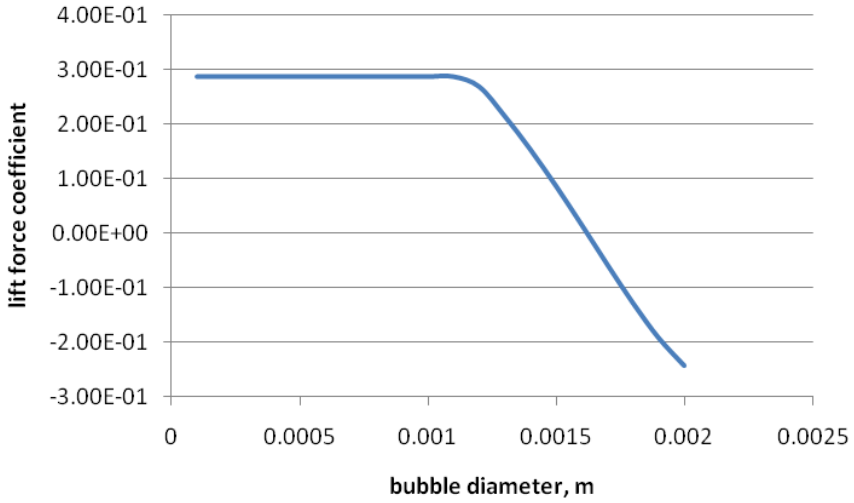


Figure 2.3: Tomiyama correlation for $|\mathbf{v}_r| = 25$ cm/s.

2.3.3.4 Virtual Mass Force

The virtual mass force is given by Eqn. 2.15

$$\mathbf{F}_{VM} = C_{VM}\alpha_g\rho_l\left[\frac{D\mathbf{v}_l}{Dt} - \frac{D\mathbf{v}_g}{Dt}\right]. \tag{2.15}$$

The coefficient of the virtual mass force, C_{VM}, is a function of the void fraction, and a plot of its variation is shown in Figure 2.4. Since the void fraction in the DEBORA test case

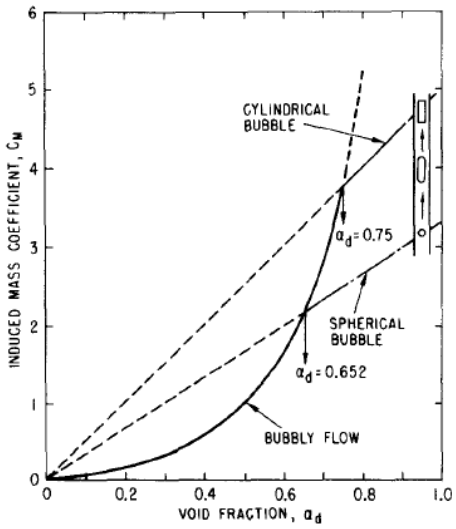


Figure 2.4: Virtual mass force coefficient as a function of void fraction. Figure taken from Ishii and Mishima [55].

does not exceed 50%, the virtual mass coefficient can by adjusted from 0 to 1.

2.3.3.5 Summary

A summary of the momentum transfer parameters chosen for the sensitivity study, as well as their ranges, is given in Table 2.4.

2.3.4 Multiphase Turbulence Model

In the present work, turbulence is modeled using the high-Reynolds number $k-\epsilon$ model. The equations solved govern k and ϵ in the continuous phase, and turbulent viscosity in the

Table 2.4: Summary of phase-to-phase momentum transfer parameters

Parameter	Adjustment	Explanation
C_d	$\pm 30\%$	None
σ_α	$\pm 30\%$	None
C_L	$-0.3 \leq C_L \leq 0.3$	Tomiyama correlation
C_{VM}	$0 \leq C_{VM} \leq 1.0$	Ishii and Mishima

dispersed (gas) phase is obtained through a correlation. Following the description in [18], the governing equations in the continuous (c) phase are

$$\frac{\partial(\alpha_c \rho_c k_c)}{\partial t} + \nabla \cdot (\alpha_c \rho_c \mathbf{v}_c k_c) = \nabla \cdot \left(\frac{\alpha_c(\mu_c + \mu_c^t)}{\sigma_k} \nabla k_c \right) + \alpha_c (G - \rho_c \epsilon_c) + S_{k2} + \sum_{i \neq c} \left(\dot{m}_{ci} k_c^{(ic)} - \dot{m}_{ic} k_c \right), \quad (2.16)$$

$$\frac{\partial(\alpha_c \rho_c \epsilon_c)}{\partial t} + \nabla \cdot (\alpha_c \rho_c \mathbf{v}_c \epsilon_c) = \nabla \cdot \left(\frac{\alpha_c(\mu_c + \mu_c^t)}{\sigma_\epsilon} \nabla \epsilon_c \right) + \alpha_c \frac{\epsilon_c}{k_c} (C_1 G - C_2 \rho_c \epsilon_c) + S_{\epsilon 2} + \sum_{i \neq c} \left(\dot{m}_{ci} \epsilon_c^{(ic)} - \dot{m}_{ic} \epsilon_c \right), \quad (2.17)$$

where the definitions of the new quantities are as follows:

k_c	continuous phase turbulent kinetic energy
μ_c	continuous phase molecular viscosity
σ_k	turbulent Prandtl number for the k_c equation
ϵ_c	dissipation rate of k_c
σ_ϵ	turbulent Prandtl number for the ϵ_c equation
C_1, C_2	empirical constants
G	$\mu_c(\nabla \mathbf{v}_c + (\nabla \mathbf{v}_c)^T) : \nabla \mathbf{v}_c$
$S_{k2}, S_{\epsilon 2}$	phase interaction terms

Turbulent stress in each phase k (not to be confused with the turbulent kinetic energy used in the above equations) is modeled using the eddy-viscosity approach,

$$\boldsymbol{\tau}^t = \mu_k^t \left(\nabla \mathbf{v}_k + (\nabla \mathbf{v}_k)^T - \frac{2}{3} \nabla \cdot \mathbf{v}_k \boldsymbol{\delta} \right) - \frac{2}{3} \rho_k k_k \boldsymbol{\delta}, \tag{2.18}$$

where the turbulent eddy viscosity in the continuous phase is given by

$$\mu_k^t = C_\mu \rho_k \frac{k_k^2}{\epsilon_k}. \tag{2.19}$$

The coefficient C_μ is treated as a parameter for the sensitivity study. The eddy viscosity for the dispersed (d) gas phase is also required, and it is obtained according to

$$\mu_d^t = \frac{\rho_d}{\rho_c} C_t^2 \mu_c^t, \tag{2.20}$$

where the response function C_t that relates the viscosities is treated as a parameter for the sensitivity study. The baseline value of C_t is a constant of 1. Finally, the phase interaction source terms used above are expressed as

$$S_{k2} = -A_i \frac{\nu_c^t}{\alpha_c \alpha_d \sigma_\alpha} \mathbf{v}_r \cdot \nabla \alpha_d + 2A_i (C_t - 1) k_c, \tag{2.21}$$

$$S_{\epsilon 2} = 2A_i (C_t - 1) \epsilon_c. \tag{2.22}$$

2.3.5 Wall-to-Flow Heat Transfer Models and Parameter Ranges

The wall-to-flow heat trasnfer model implemented in STAR-CD by default employs a heat-partitioning formulation. The parameters governing this heat transfer are included in the present sensitivity study, and hence the model basics are reproduced below. In addition, we consider an alternate wall-to-flow heat transfer model not based on heat partitioning,

Chen's correlation, and we include this model in the sensitivity study.

2.3.5.1 Heat Partitioning

In a heat-partitioning formulation, the heat transfer from the wall to the fluid in subcooled boiling consists of three parts: single phase convective heat transfer, evaporative heat transfer, and the quenching heat transfer,

$$\dot{q}_w = \dot{q}_c + \dot{q}_e + \dot{q}_q \tag{2.23}$$

The evaporative heat flux is obtained from

$$\dot{q}_e = \frac{\pi d_b^3}{6} p \rho_g \Delta h_{gl} f n'', \tag{2.24}$$

where n'' is the nucleation site density and f is the bubble departure frequency, given by

$$n'' = (m\Delta T_{\text{sup}})^p, \tag{2.25}$$

$$f = \sqrt{\frac{4}{3}\frac{g(\rho_l - \rho_g)}{d_{b,w}\rho_l}}. \tag{2.26}$$

The wall is divided into two regions: A_e is the area fraction where evaporative heat transfer occurs and $A_c = 1 - A_e$ is the area fraction where single phase convective heat transfer occurs. A_e is obtained from

$$A_e = F_A \frac{\pi}{4} d_{b,w}^2 n'', \quad F_A = 2. \tag{2.27}$$

The single-phase convective heat flux and quenching heat flux are obtained using

$$\dot{q}_c = \text{HTC}_c(1 - A_c)(T_{\text{wall}} - T_l), \tag{2.28}$$

$$\dot{q}_q = \text{HTC}_q A_e (T_{\text{wall}} - T_l), \tag{2.29}$$

where the single-phase heat transfer coefficient is obtained using the $k-\epsilon$ turbulence model with wall functions. The quenching heat transfer coefficient is obtained using the model of Del Valle and Kenning,

$$\mathrm{HTC}_q = 2f\sqrt{t_w \rho_l C_{p,l} \lambda_l / \pi}, \quad t_w = 0.8/f. \tag{2.30}$$

A bubble departure size is needed to determine the evaporative heat flux and evaporative area fraction. The expression of Tolubinsky and Kostanczuk [109], obtained for water at a liquid velocity of 0.2 m/s, is used,

$$d_{b,w} = d_{b,w,0} \exp(-\Delta T_{\mathrm{sub}}/\Delta T_0). \tag{2.31}$$

In this expression, the coefficient $d_{b,w,0}$ is set to $0.6mm$ which is modified from the original value of $1.4mm$. A value for ΔT_0 of $45K$ is consistent with the original reference.

To reduce the number of parameters for the sensitivity study, this heat-flux partitioning model was analyzed at a high-level by introducing multiplicative factors, $\pm 30\%$, in front of the individual heat fluxes. Hence all of the detailed low-level parameters in the sub-models for the heat fluxes were not uncovered.

2.3.5.2 Chen's Correlation

In Chen's model [21], we assume that the total surface heat flux is made up of a nucleate boiling contribution and a single-phase forced-convection contribution,

$$\dot{q}_w = \mathrm{HTC}_{nb}(T_{\mathrm{wall}} - T_{\mathrm{sat}}) + \mathrm{HTC}_{1\phi}(T_{\mathrm{wall}} - T_{\mathrm{fluid}}). \tag{2.32}$$

In the present implementation in STAR-CD, $\mathrm{HTC}_{1\phi}$ is obtained via wall functions from the turbulence model and HTC_{nb} is obtained using

$$\mathrm{HTC}_{nb} = C_{\mathrm{chen}} \left[\frac{k_f^{0.79} c_{pf}^{0.45} \rho_f^{0.49}}{\sigma^{0.5} \mu_f^{0.29} i_{fg}^{0.24}} \right] \Delta T_{\mathrm{sat}}^{0.24} \Delta p_{\mathrm{sat}}^{0.75} S. \tag{2.33}$$

In the above equation, k_f, c_{pf} are respectively the thermal conductivity and heat capacity of the fluid phase. The suppression factor, S, is defined as the ratio of the mean superheat (ΔT_e) to the wall superheat (ΔT_{sat}),

$$S = \left(\frac{\Delta T_e}{\Delta T_{sat}} \right)^{0.99}. \tag{2.34}$$

For pool boiling, $S = 1$. For convective boiling, the value of S is obtained from a correlation matching experimental data, as illustrated in Figure 2.5. The value of F in the figure is the

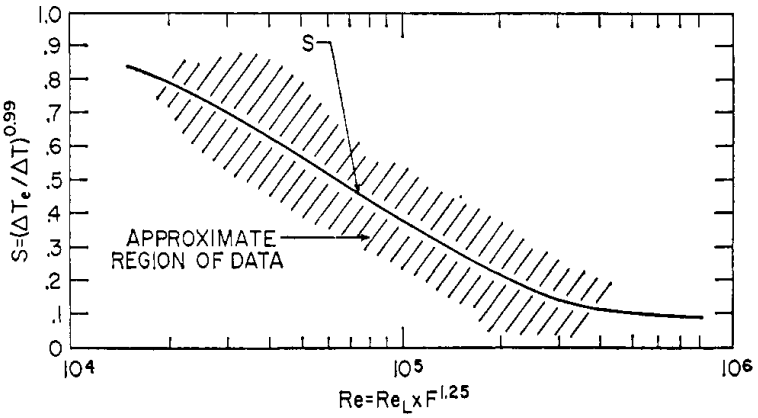


Figure 2.5: Correlation of the suppression factor to Reynolds number in Chen's model. This is Figure 7 in Chen's original paper [21].

defined as:

$$F = \left(\frac{\mathrm{Re}_{2\phi}}{\mathrm{Re}_f} \right)^{0.8}, \tag{2.35}$$

where the fraction in parentheses is the ratio of the two-phase Reynolds number to the single-phase liquid Reynolds number. For subcooled boiling the value for F is unity. Finally,

C_{chen} is a constant with baseline value of 0.00122.

2.3.6 Summary of all Parameters

A summary of all of the parameters considered, and their ranges is given in Table 2.5. We note that for most of the parameters, the imposed variation for the sensitivity study is ±30%. Without additional information, this is a reasonable range for most parameters. The only exception is the lift coefficient, which can assume both positive and negative values, and which starts out at a small baseline value. However, iterative convergence problems were encountered when the entire range from the Tomiyama correlation, −0.3 to 0.3, was used for imposing variation in the lift coefficient. Therefore for the present study we restricted the variation to 30% around the baseline hard-coded value in STAR-CD.

Table 2.5: Summary of all parameters considered for the sensitivity study.

Parameter	Baseline	Variation	Comments
d_b	Kurul+Podowski	±30%	
K_1	0.6	±30%	Encompasses experimental data
Nu_g	26	$[2, 30]$	Conduction sets lower bound
C_d	Wang correlation	±30%	
σ_α	0.9	±30%	
C_L	-0.03	±30%	Tomiyama correlation not used
C_{VM}	0	$[0, 1]$	Ishii and Mishima
C_μ	1.0	±30%	Scaling factor for turbulent eddy viscosity
C_t	1.0	±30%	Scaling factor for gas/liquid viscosity ratio
$C_{\eta''}$	1.0	±30%	Scaling factor for nucleation site density
C_{q_e}	1.0	±30%	Scaling factor for evaporative heat flux
C_{q_q}	1.0	±30%	Scaling factor for quenching heat flux
C_{chen}	.00122	±30%	Coefficient in Chen's boiling correlation

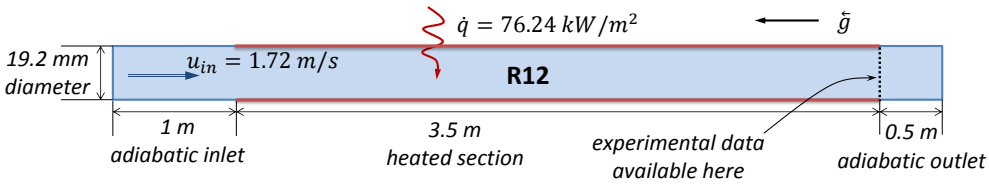


Figure 2.6: DEBORA problem setup.

2.4 The DEBORA Test Problem

2.4.1 Geometry and Flow Conditions

A prototypical test case, the DEBORA experiment [44], was chosen to assess the performance of CFD in modeling subcooled boiling. In the DEBORA experiment, the refrigerant $R12$ is used as the working fluid to simulate pressurized water reactor conditions under low pressure. Liquid $R12$ flows upward inside a vertical pipe having an internal diameter equal to $19.2mm$. The whole pipe can be divided axially into three parts: the adiabatic inlet section ($1m$ in length), the heated section ($3.5m$ in length), and the adiabatic outlet section ($0.5m$ in length). The system pressure is 1.459 MPa with inlet conditions specified at an inlet velocity of $1.72m/s$ and void fraction of 0.001. The wall heat flux is 76.24 kW/m^2.

Vapor bubbles are generated by nucleation onto the wall surface and they condense into the subcooled liquid when they are far from the wall. In this experiment, local measurements can be performed with a sensor displaced in the radial direction only [43]. At the end of the heated section, the radial profiles of the void fraction and bubble diameter have been measured by means of an optical probe, and liquid temperature has been measured by thermocouples.

2.4.2 Discretization

The STAR-CD model of the DEBORA experiment uses an axisymmetric mesh, a portion of which is shown in Figure 2.7. Multiple meshes were constructed with different axial and radial resolutions for the purpose of the convergence study discussed in Section 2.5.

Ultimately one mesh was chosen for the study of physical parameter sensitivities. Implementation of the wall-to-fluid heat transfer models is accomplished through the use Fortran subroutines that are incorporated into STAR-CD through user-defined "ufiles."

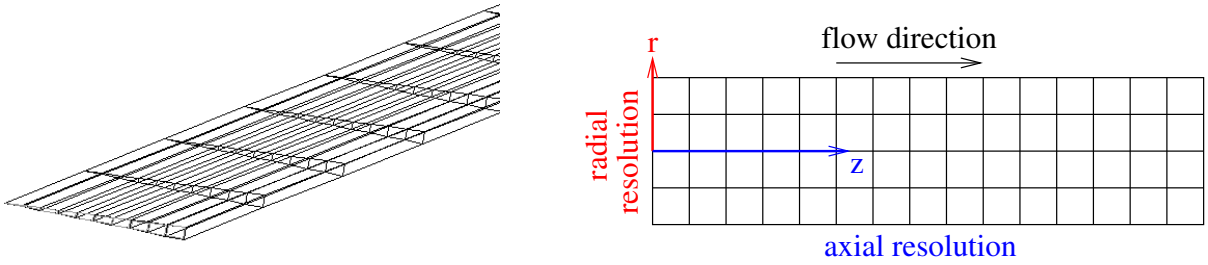


Figure 2.7: Portion of the mesh used in the STAR-CD model, and the definition axial and radial mesh resolutions.

2.4.3 Outputs

Four outputs of engineering relevance are considered for the sensitivity study. The first output is pressure drop over the channel (equal to the inlet pressure because of the prescribed outlet pressure $p_{\text{outlet}} = 0$),

$$\Delta p = \bar{p}(z = 5m) - \bar{p}(z = 0m) = -\bar{p}(z = 0m) = -\frac{1}{0.0192m} \int_{r=0m}^{r=0.0192m} p(r, z = 0m)\ dr. \quad (2.36)$$

The second output is the average wall temperature in the heated section

$$\bar{T}_w = \frac{1}{3.5} \int_{z=1m}^{z=4.5m} T_w(z)\ dz. \quad (2.37)$$

Third is the radially-averaged void fraction at the end of the heated section

$$\bar{\alpha}_g = \frac{1}{0.0192m} \int_{r=0m}^{r=0.0192m} \alpha_g(r, z = 4.5m)\ dr. \quad (2.38)$$

Finally, the fourth output is the centroid of the radial void fraction profile

$$\bar{r}_{\alpha_g} = \frac{1}{0.0192m\ \bar{\alpha}_g} \int_{r=0m}^{r=0.0192m} r\alpha_g(r, z = 4.5m)\ dr. \tag{2.39}$$

2.5 Star-CD Convergence Studies

Convergence studies were performed on the STAR-CD DEBORA model with the goals of:

- Achieving robust iterative convergence.
- Investigating whether asymptotic output convergence is attained.

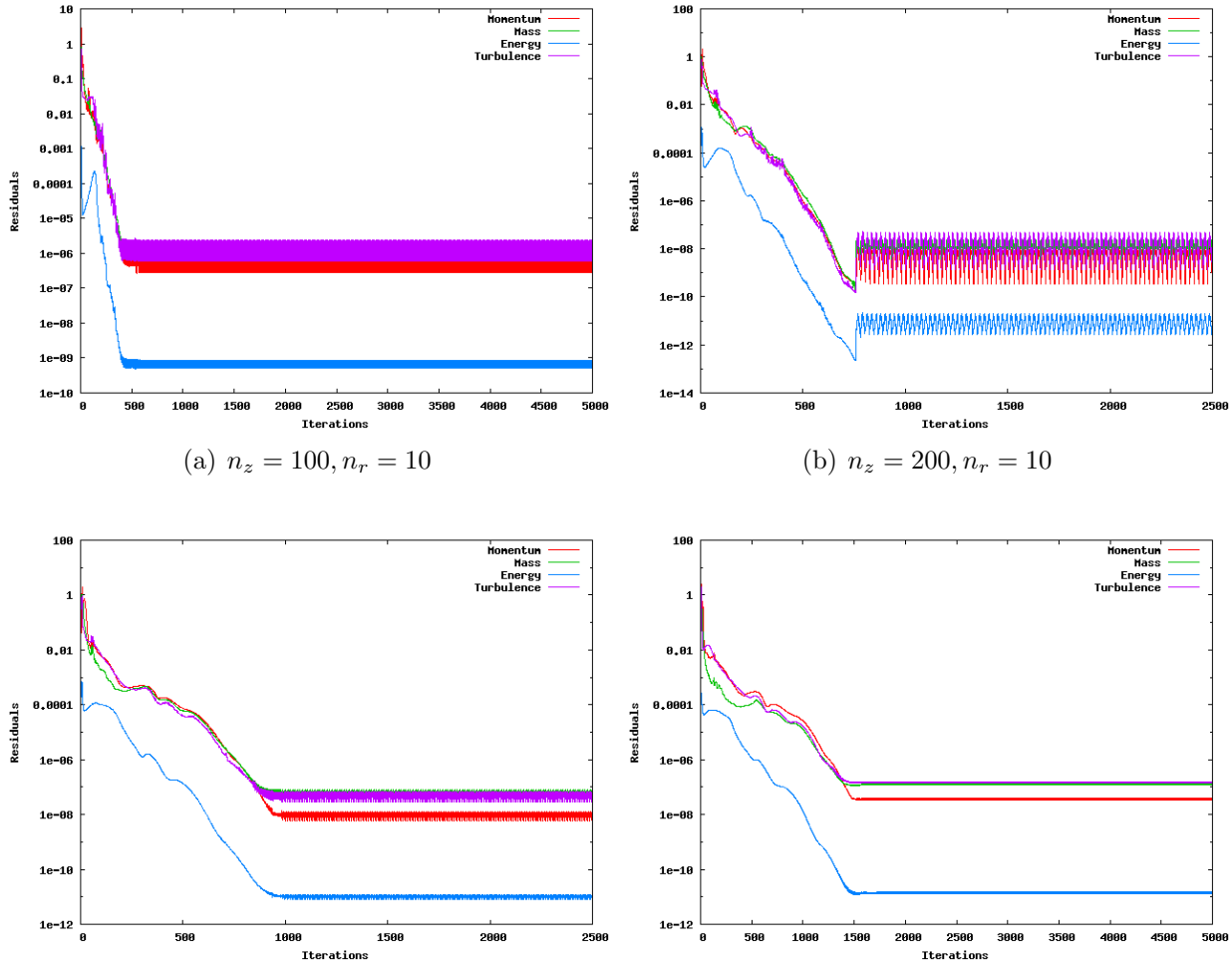
The following subsections present the results of these studies.

2.5.1 Mesh Anisotropy and Iterative Convergence

During initial variations of the mesh resolution, iterative convergence problems were encountered for some of the finer meshes. A study was performed to determine whether the axial or radial resolution, or both, were responsible for the iterative convergence problems. In this study, the radial mesh resolution was varied from 10 points to 30 points, and the axial resolution from 100 points to 800 points. The results of the study are shown in Figures 2.8, 2.9, and 2.10, in the form of residual norm histories versus iteration.

As shown in Figures 2.8, 2.9, and 2.10, convergence problems occur predominantly when the radial mesh resolution is fine relative to the axial mesh resolution. In particular, an axial resolution of 800 points shows no iterative convergence problems for any of the radial mesh resolutions tested.

Given the anisotropy of the domain (length to diameter ratio of 260), meshes without a large number of axial points will contain cells of moderate to high anisotropy. For example, in the 100×10 mesh, the cell aspect ratio is 26. In the 800×10 mesh it becomes 3.3. The data show that large cell anisotropy (greater than approximately 12) correlates to iterative



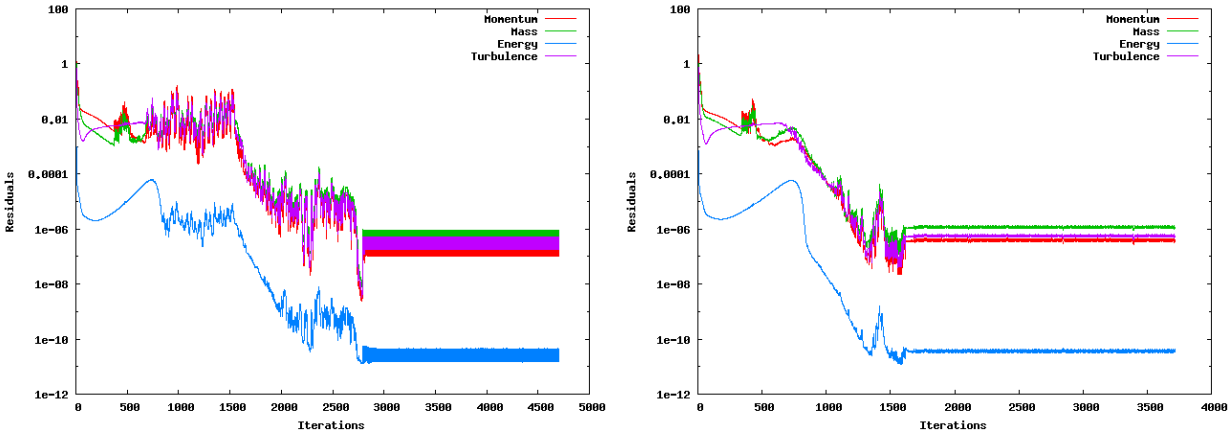
(a) $n_z = 100, n_r = 10$

(b) $n_z = 200, n_r = 10$

(c) $n_z = 400, n_r = 10$

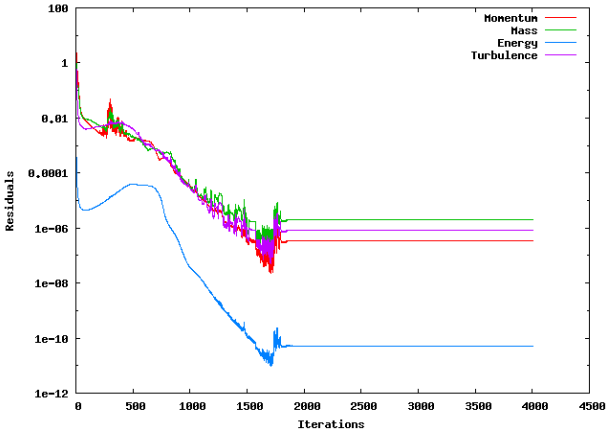
(d) $n_z = 800, n_r = 10$

Figure 2.8: Iterative convergence histories for low radial resolution, for various axial mesh sizes.

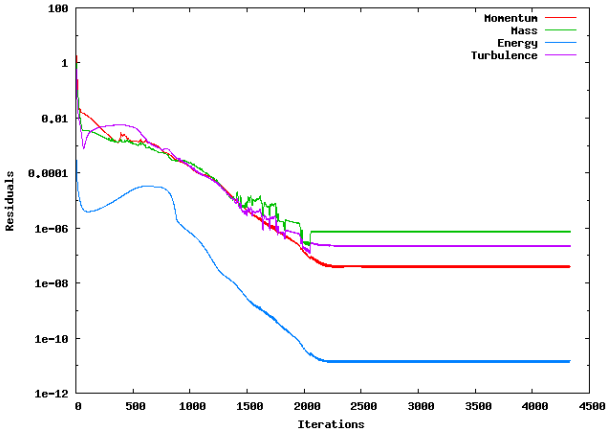


(a) $n_z = 100, n_r = 20$

(b) $n_z = 200, n_r = 20$

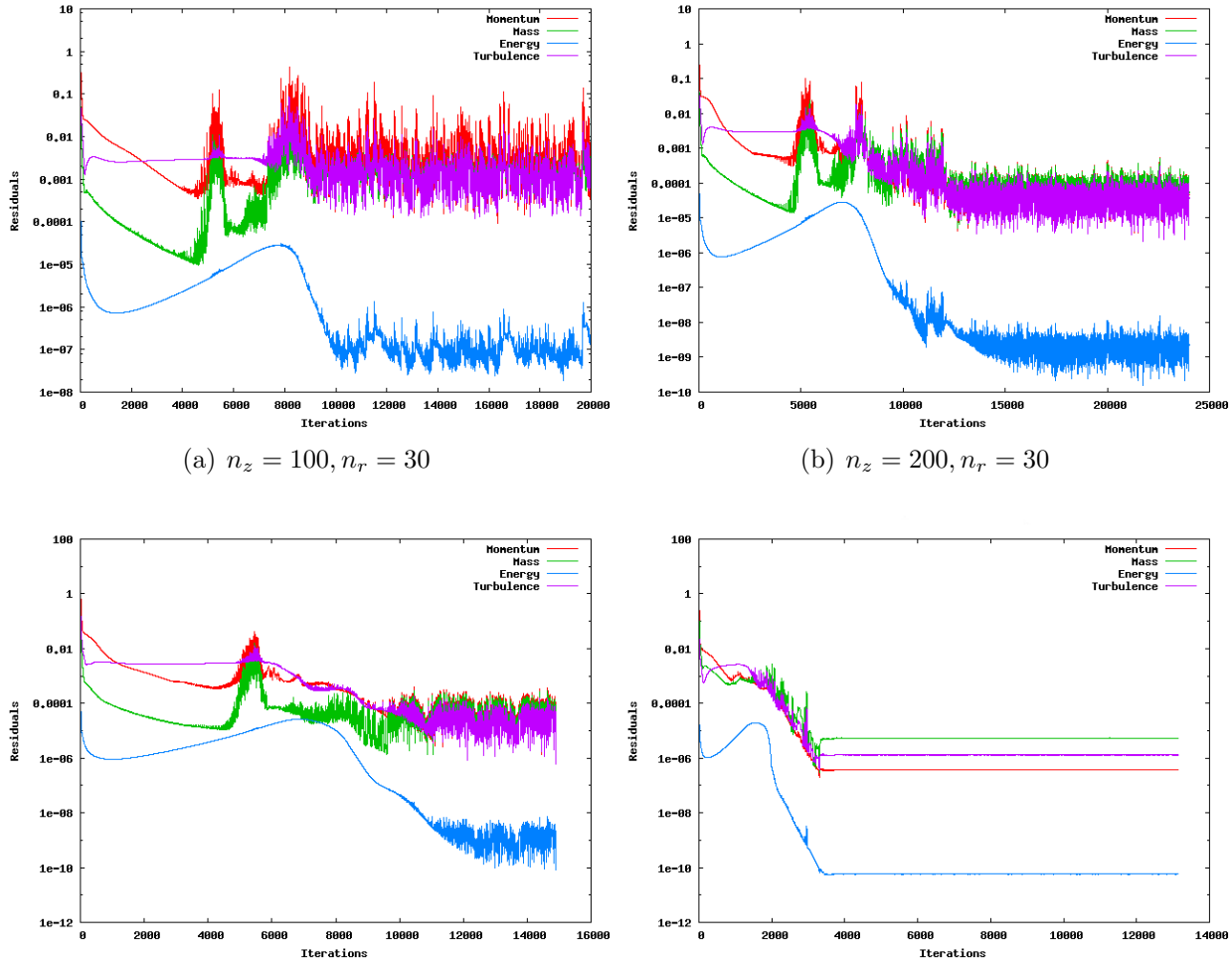


(c) $n_z = 400, n_r = 20$



(d) $n_z = 800, n_r = 20$

Figure 2.9: Iterative convergence histories for medium radial resolution, for various axial mesh sizes.



(a) $n_z = 100, n_r = 30$

(b) $n_z = 200, n_r = 30$

(c) $n_z = 400, n_r = 30$

(d) $n_z = 800, n_r = 30$

Figure 2.10: Iterative convergence histories for high radial resolution, for various axial mesh sizes.

convergence problems, and hence our recommendation is to work with meshes with sufficient axial resolution to make the anisotropy relatively low. We point out that under this definition, the 400×20 mesh is just on the border of sufficient axial resolution.

2.5.2 Asymptotic Convergence

For a convergent discretization, output errors are expected to decrease at a rate of at least first order with mesh refinement. A study was undertaken to determine whether such rates were observed for the outputs of interest in this study. Specifically, three outputs were considered: pressure drop, average wall temperature, and radially-averaged void fraction at the end of the heated section. We note that for some meshes iterative convergence was not attained, and these data were not included in the convergence study.

As the turbulent boundary layer treatment makes use of wall functions, which are not applicable beyond a certain near-wall resolution, radial refinement was performed with a fixed first node off the wall at $y^+ \approx 40 - 50$. Additional nodes were then spaced equally in the remaining radial distance. We note that an initial version of this study did not keep this distance fixed, and the refinement results did not show convergence, especially for the pressure drop.

If the outputs were convergent, and if we were in the asymptotic regime in terms of mesh resolution, we would expect the output values to asymptote/level-off with increasing radial/axial resolution. More precisely, a first order convergence rate would dictate a halving of the error with each doubling of resolution. Although without exact solutions the output error is not directly available, convergence can still be assessed by monitoring output values at three or more successive resolutions. Visually, this assessment is made easier by plotting outputs versus a logarithmic scale of resolution.

Figure 2.11 shows the results of axial and radial refinement studies for the three outputs, using the baseline Eulerian multiphase model in STAR-CD. We note that with the exception of average wall temperature versus radial resolution, none of the plots demonstrate

asymptotic convergence. That is, the changes in the output do not decrease, and sometimes increase, with additional mesh resolution.

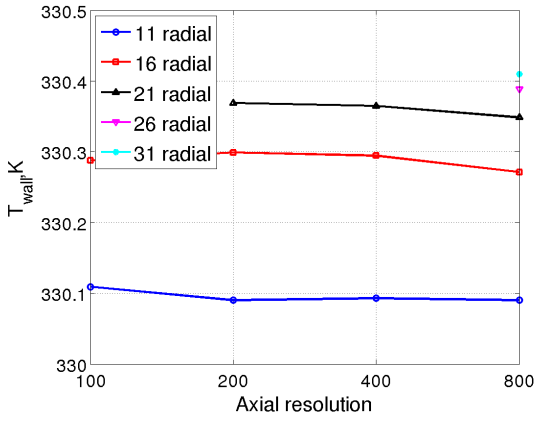
The author's first suspicion concerning the lack of asymptotic convergence fell on the multiphase model. To investigate whether this model is responsible for the lack of convergence, the same study was redone with the interfacial area concentration manually set to zero in the code. This effectively turned off phase-to-phase heat transfer. The resulting plots are shown in Figure 2.12. The plots showing outputs versus radial resolution appear qualitatively improved, inasmuch as changes in the output appear to decrease with additional resolution. We note that at increased axial resolution, the convergence rate slows down. Additional points at higher resolutions would help clearly determine the rate, but these were not attainable due to iterative convergence problems.

More troublesome in Figure 2.12 is the behavior of the outputs with increasing axial resolution. In this case, none of the outputs demonstrate convergence, and in most cases the changes in the outputs grow with additional axial mesh resolution.

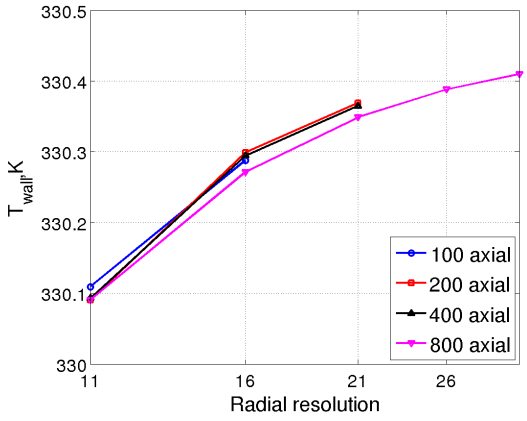
Based on the lack of observed convergence in these results, a single phase simulation was performed to assess whether the lack of convergence arises from the multiphase/boiling models, or from other factors such as turbulence modeling, mesh refinement strategy, etc. Figure 2.13 shows the results of this study. Note that void fraction is no longer present as an output. In addition, this case differs from the previous ones in that the wall flux was reduced 10% to suppress boiling.

The results in Figure 2.13 show much improved convergence of the pressure drop output with radial resolution. The average wall temperature also appears to be converging with radial resolution, albeit at a slower rate. While convergence with axial resolution is not demonstrated, we note that the changes in the outputs with axial resolution are small, and nonzero residuals from imperfect iterative convergence could be partially responsible.

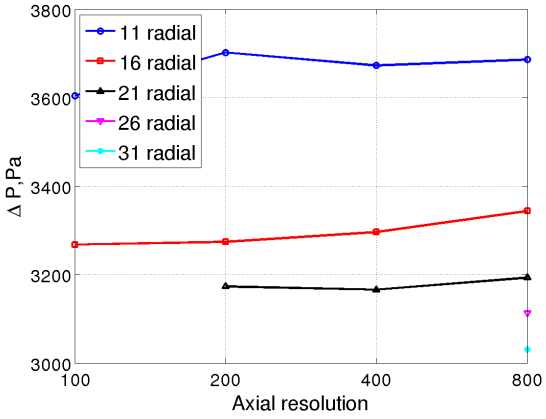
The above results show that asymptotic convergence with radial resolution is possible for single phase and for sub-cooled boiling with no interfacial area transport. With multiphase



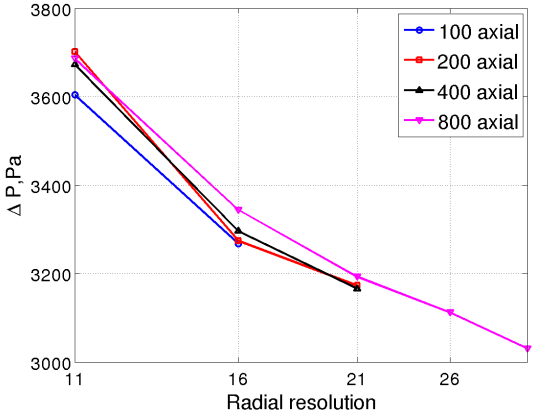
(a) Average wall temperature



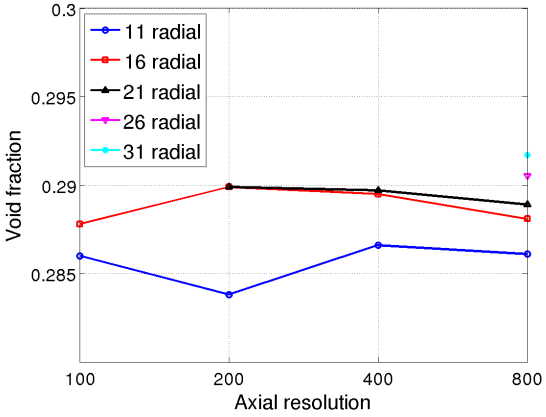
(b) Average wall temperature



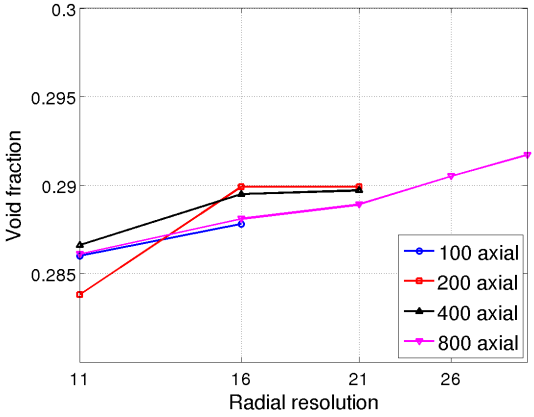
(c) Pressure drop



(d) Pressure drop

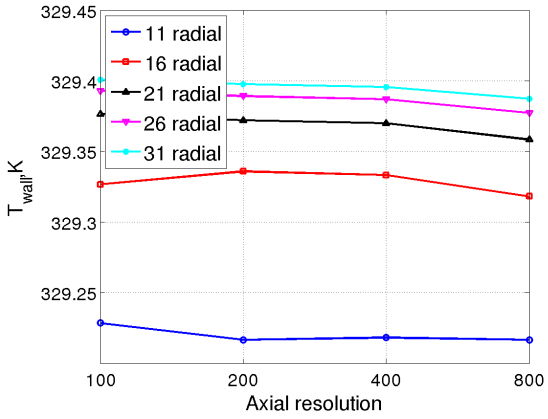


(e) Average void fraction

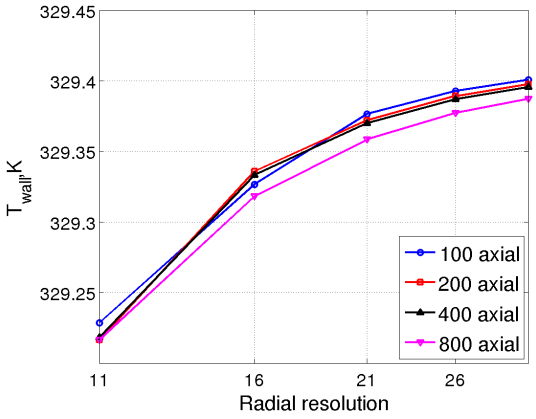


(f) Average void fraction

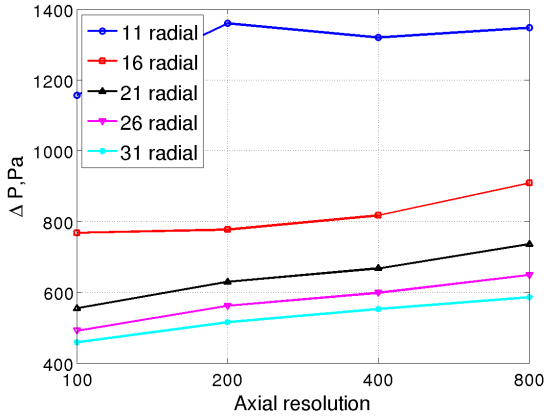
Figure 2.11: Mesh refinement convergence results for the base Eulerian multiphase model. Note, in the radial refinement, the first point off the wall remains fixed.



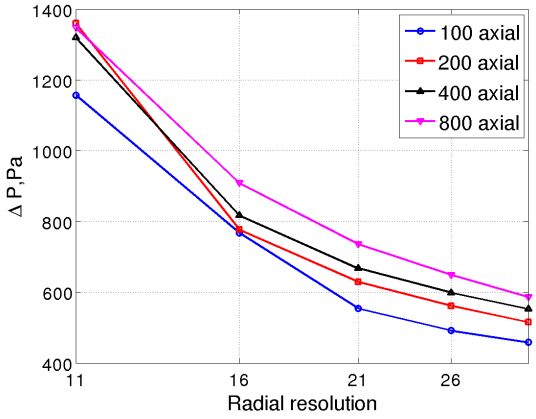
(a) Average wall temperature



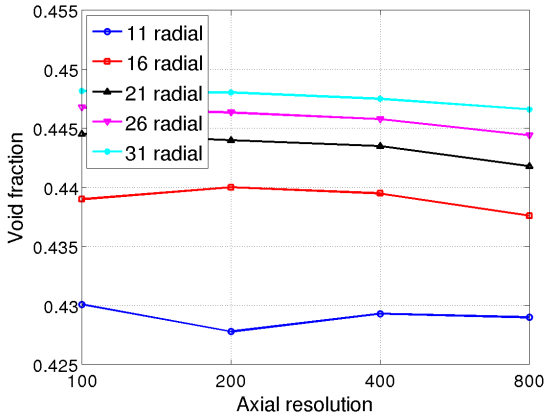
(b) Average wall temperature



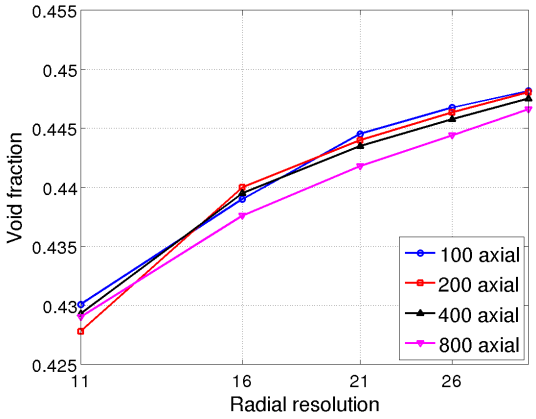
(c) Pressure drop



(d) Pressure drop

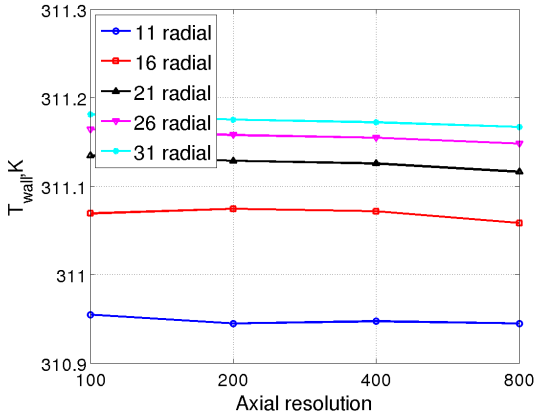


(e) Average void fraction

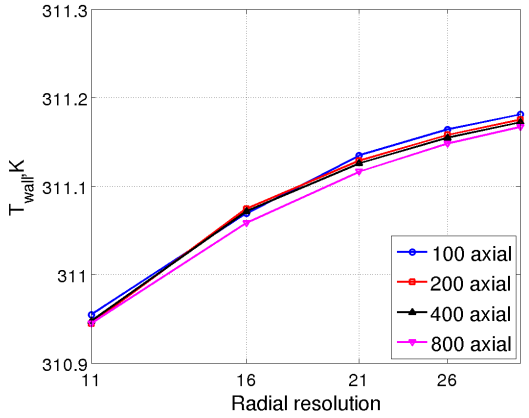


(f) Average void fraction

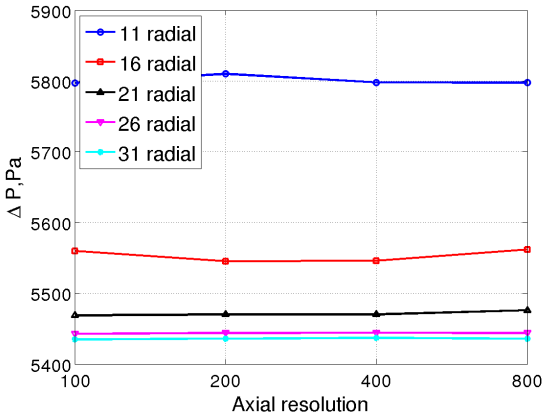
Figure 2.12: Mesh refinement convergence results for the Eulerian multiphase model with interfacial area manually set to zero. In the radial refinement, the first point off the wall remains fixed.



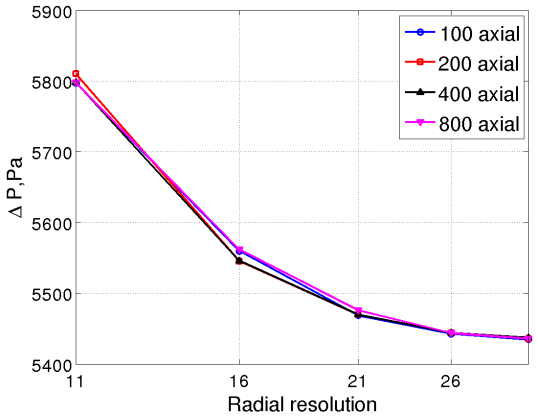
(a) Average wall temperature



(b) Average wall temperature



(c) Pressure drop



(d) Pressure drop

Figure 2.13: Mesh refinement convergence results for a single-phase simulation, with the first point off the wall at a fixed location.

heat transfer included, however, convergence in axial and radial mesh resolutions (independently) does not exist for the mesh sizes tested. Based on this result, we advise caution when using these models, as implemented in the software used, at comparable resolutions. It is plausible that increased mesh resolution might help (i.e. that the above results are not yet asymptotic). However, iterative convergence problems were encountered when attempting finer simulations, which also became significantly more demanding computationally. On more complex geometries, such finer resolutions would likely become prohibitive.

Finally, we make the remark that mesh anisotropy could possibly be a factor in asymptotic convergence as well. Refining only in the radial direction or only in the axial direction does not preserve cell anisotropy. If the numerical scheme introduces errors that are not bounded with increasing anisotropy, then lack of convergence in each individual refinement approach might not imply lack of convergence with uniform refinement. Uniform refinement studies with the baseline Eulerian multiphase model were performed, and the conclusion from those studies was that the outputs were not convergent, most egregiously for the pressure drop.

Given plausible explanations for lack of convergence, but finite resources to investigate all possibilities, we resolved to proceed with the sensitivity study on a baseline mesh of 400 radial by 21 axial resolution. This mesh is not the finest tested, yet changes in the outputs with additional refinement were not large in magnitude from an engineering standpoint.

2.6 Star-CD Sensitivity Studies

2.6.1 Methods

A Monte-Carlo approach was used to perform to the sensitivity study. Latin-Hypercube sampling, which attempts to ensure that the parameter space is fully explored, was used to determine the appropriate values of the parameters to simulate.

For both boiling models tested, all of the parameters were varied together in a monolithic sensitivity study. The complete list of parameters and associated ranges is given in

Section 2.3.6. We note that due to the high-level treatment of some of the models, such as heat-partitioning boiling, any magnification of uncertainties in sub-models was ignored. That is, a variation in heat transfer of ±30% may imply a variation of some empirical constant by 0.001%, or possibly 200%, depending on the structure of the sub-model.

For each parameter, a prior probability distribution was assumed. As noted in Section 2.3, fixed percentage variation was used in the absence of comprehensive a priori information about the ranges or distributions of the model parameters. In addition, each parameter was assumed to have a uniform probability density function within that range.

The Sandia DAKOTA sensitivity and uncertainty analysis software [3] was used to generate the Latin-Hypercube samples and execute STAR-CD with the modified parameters. STAR-CD only provides point-wise outputs, and these were stored in text files that were post-processed to yield the integrated outputs of interest. The integration was performed using a third-order method for the evenly-spaced points in the axial direction. In the radial direction, the mesh size varies, so the simpler midpoint method was used.

2.6.2 Parameter and Model Selection

The goal of the study was to find the most important uncertain parameters of approximate or empirical relations used in the STAR-CD two-phase flow model. Parameters were chosen to split up the two-phase flow model into a few major components. During initial prototyping, one parameter in each component was varied, and in subsequent studies the models and sub-models in the important components were varied. Each of these components may comprise a number of models, correlations, and empirical factors, but including all of those parameters in a sensitivity study would be prohibitively expensive. The dimensionality of the problem was reduced so that important components could be isolated for further study. The components involving uncertain parameters were as follows:

- gas-liquid heat and mass transfer
- gas-liquid momentum transfer (drag, lift, virtual mass, and turbulent dispersion forces)

- void fraction, interfacial area
- wall-to-flow heat transfer (heat partitioning model or Chen's correlation)
- turbulence

It was not possible to vary all of the components directly. For example the user cannot place a pre-multiplier on the drag force, $\mathbf{F}_D$, directly. However, the routine that calculates C_D can be altered by the user. Thus, we chose a set of parameters that were available for user-specification such that each component was linearly related to at least one parameter. The parameters actually varied and their ranges are described above in Section 2.3. They were

$$\mathrm{Nu}_g,\ K_1,\ C_D,\ C_L,\ C_{VM},\ \sigma_a,\ d_b,\ \mu_\ell^t,\ C_t,\ \{q_e,\ n'',\ q_q\} \text{ or } \{C_{\mathrm{chen}}\}$$

Figures 2.14 and 2.15 show how these parameters (boxed) are used in the various models. We could not avoid the fact that some parameters affected multiple components simultaneously. However, the effects of each component can be deduced from the effects of the parameters and the relationships described in Figures 2.14 and 2.15.

2.6.3 Results

For each parameter, results of the sensitivity study yielded a correlation of that parameter to the four outputs. This correlation is a measure of how strongly the output is dependent upon the parameter. A correlation coefficient of almost one (or minus one) means that the parameter is responsible for most of the variation in the output. It does not give information about how sensitive the physical output is to that parameter.

The correlation coefficients between the parameters should be zero if DAKOTA chose them independently. This is satisfied because all of the correlations were less than 0.1 in absolute value. In the heat flux partitioning model study there were 12 parameters and 836

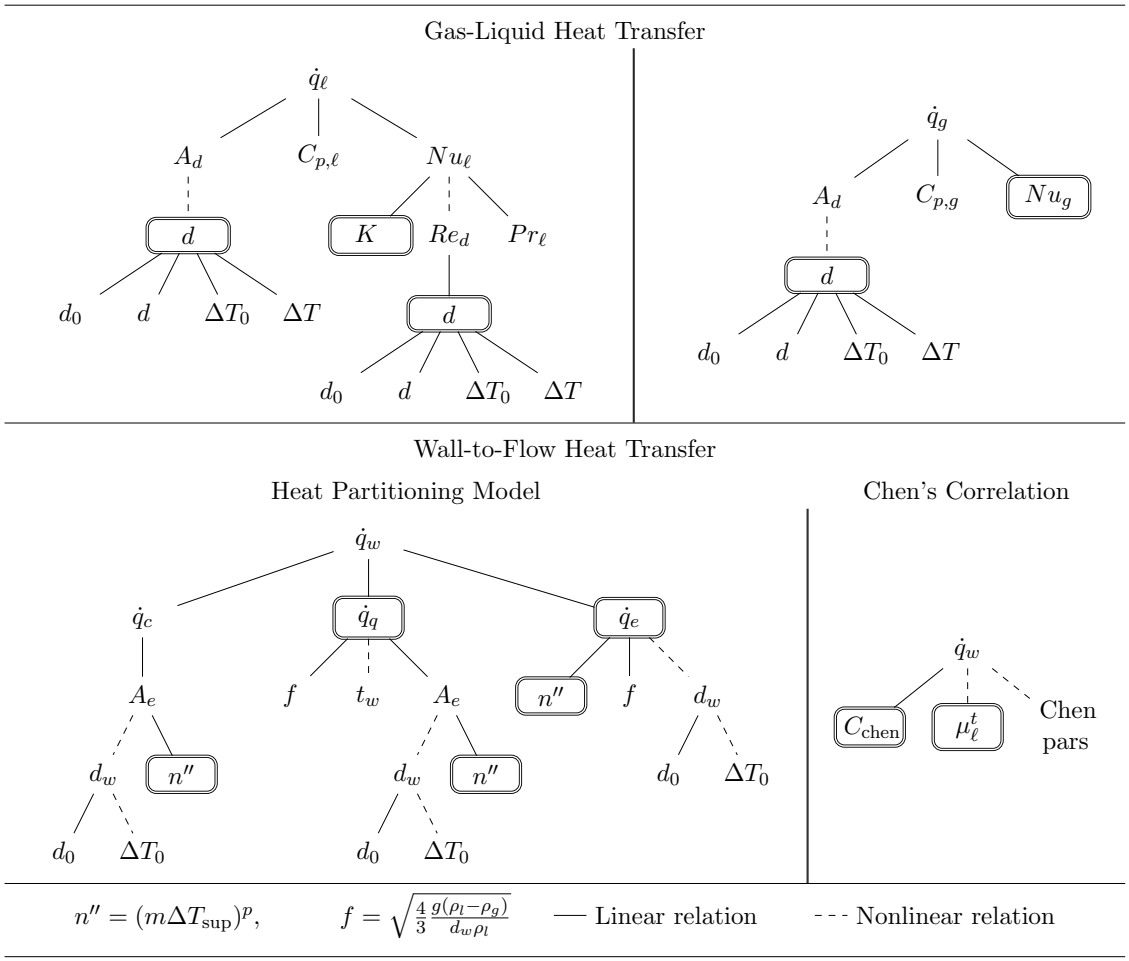


Figure 2.14: Heat transfer models and parameters in STAR-CD.

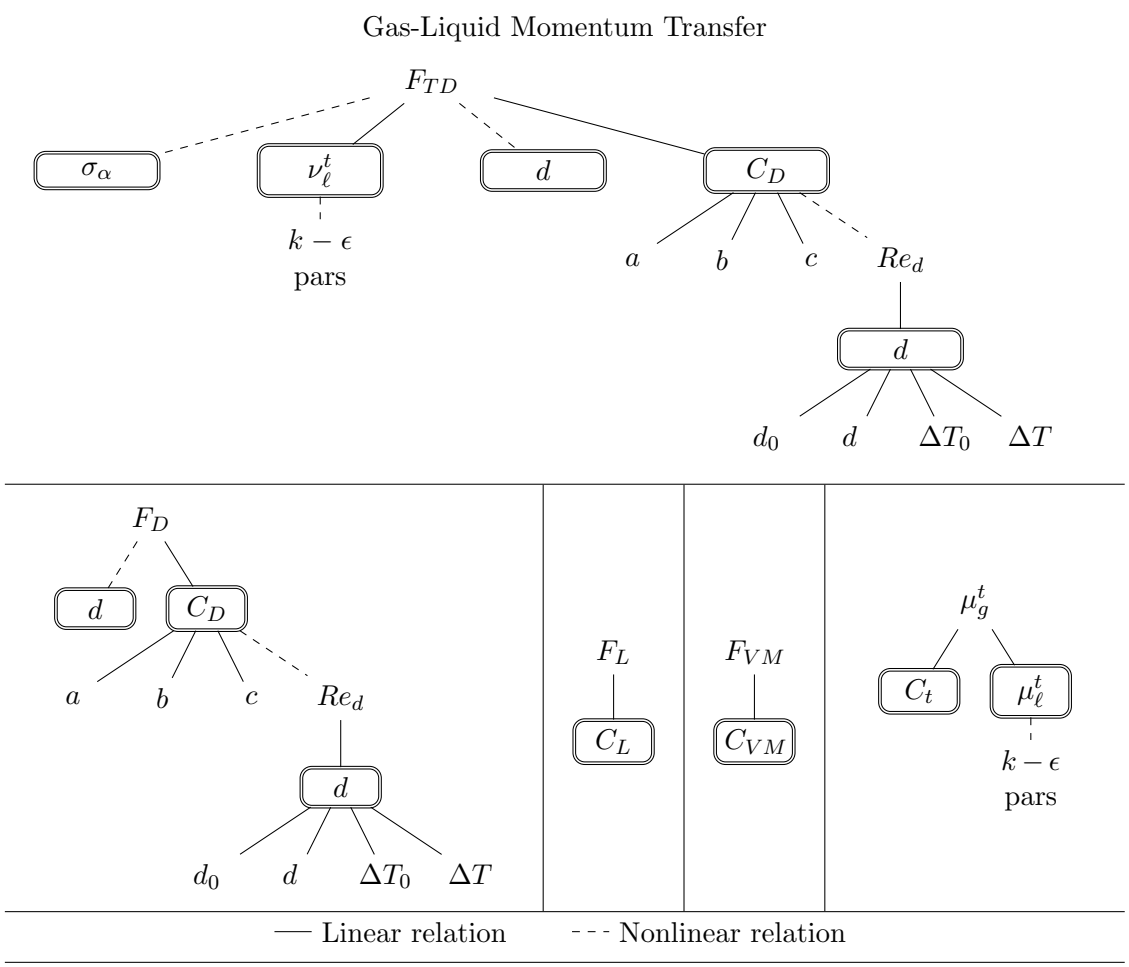


Figure 2.15: Momentum transfer models and parameters in STAR-CD.

total runs, of which 336 did not converge to the pre-set residual tolerance of 10^{-7}, but ran up to the maximum 2500 iterations. The study with Chen's correlation had 10 parameters and 600 total runs, of which 155 did not reach the residual tolerance. Figures 2.16 and 2.17 show the histograms of the residuals for both studies, which indicates that most runs were well converged, even if they did not reach the somewhat strict residual tolerance. Some of the data points did contain large residuals, and these were manually removed before post-processing. Figures 2.21 and 2.22 show that there are no major outliers in the outputs.

2.6.3.1 Heat partitioning model

Figure 2.18 shows, for each parameter, the correlation coefficients with respect to the four outputs. From this figure we can make a few remarks. First, the most important parameters are bubble diameter (d_b), liquid Nusselt number (K_1), turbulent viscosity coefficients (C_μ and C_t), nucleation site density (n''), and evaporative heat flux ($\dot{q}_e$). The liquid turbulent viscosity (C_μ) is mainly important for the pressure drop, and the virtual mass coefficient is moderately important. The wall-to-flow model in Figure 2.14 shows that n'' affects all of the heat fluxes equally (since it is linearly related to each of them). From the correlation coefficients, it is clear that n'' and $\dot{q}_e$ have the same effect, whereas $\dot{q}_q$ has no effect. Thus we conclude that n'' is in this case acting only through $\dot{q}_e$, and $\dot{q}_c$ has little effect.

Also, we note that the correlation coefficients with the void fraction ($\bar{\alpha}_g$) have the opposite sign as those for the pressure drop (Δp) and the centroid of the void fraction profile ($\bar{r}_{\alpha_g}$). Thus, in general higher void fractions implies lower Δp (higher inlet pressure) and more bubbles near the center of the pipe. Although the STAR-CD model cannot predict the void fraction peak moving away from the wall, it does successfully model the fact that more bubbles tend to be located in the middle of the pipe with higher temperatures and more boiling [61]. We can further see this by looking at the correlations between the outputs. The void fraction $\bar{\alpha}_g$ has a correlation coefficient of -0.77 with respect to Δp and -0.82 with respect to $\bar{r}_{\alpha_g}$. Note that this trend is not true of the gas turbulent viscosity parameter C_t.

When C_t increases, $\bar{r}_{\alpha_g}$ decreases as expected, but Δp increases. This may be due to the increased mixing in the gas phase that occurs with higher C_t, which would lead to more and larger bubbles further from the wall, which leads to a larger pressure drop.

The left plot of Figure 2.20 shows, for each output, the correlations of all of the parameters. First, the average wall temperature is almost exclusively affected by the evaporative heat flux (as mentioned above, we can infer that n'' acts only through $\dot{q}_e$).

Second, the gas Nusselt number Nu_g, the turbulent Prandtl number σ_α, and the lift coefficient C_L have little effect on any of the outputs. It is interesting to note that the liquid Nusselt number has a relatively strong effect, although the gas Nusselt number does not. This implies that there is little condensation going on, and heat is mostly being transferred from the liquid to the gas. Also, the relative unimportance of σ_α implies that bubble drag due to turbulent eddies is of little importance. The insignificance of C_L can be partially attributed to the fact that the values were quite small, but the other bubble forces (turbulent dispersion, drag, and virtual mass) had relatively small effects on the outputs.

Third, the bubble diameter (d_b) has a strong effect on all of the outputs except the average wall temperature. This observation has been made previously, most recently by Lo et al [76].

Overall, we can give the following summary

1. Bubble diameter is the most important parameter overall.
2. Evaporative heat flux is the only important parameter for average wall temperature, and is important for the other outputs as well.
3. Liquid-to-gas heat transfer and turbulence modeling are important.
4. Momentum transfer is overall not as important.

Figure 2.21 shows some scatter plots of the data. From these we can identify the physical sensitivities as well. The variations of the outputs are shown in Table 2.6. From these variations we see that Δp is not predicted well, and that $\bar{\alpha}_g$ is predicted to a coarse engineering

accuracy. The average wall temperature is predicted relatively well. Although $\bar{r}_{\alpha_g}$ seems to be predicted well here, we know that the void fraction profile is not accurate (see [61]). The apparent small sensitivity of $\bar{r}_{\alpha_g}$ implies that varying (or tuning) the parameters will probably not significantly improve the prediction of the void fraction profile. Again, a possible exception to this is the lift coefficient, which was not varied significantly in this study, but which could have a strong effect on the void fraction centroid.

2.6.3.2 Chen's correlation

The results for the Chen's correlation study are quite similar to those of the heat partitioning model study. In particular, Figure 2.19 shows all of the same patterns as Figure 2.18, with the Chen's correlation parameter C_{chen} having a large impact on the wall temperature $\bar{T}_w$. One exception is that C_{chen} had little impact on the other outputs, compared to the wall boiling parameters in the heat partitioning model.

Output	Approximate Range
Δp	2000-4000 Pa
$\bar{T}_w$	335-337 K
$\bar{\alpha}_g$	0.19-0.28
$\bar{r}_{\alpha_g}$	6-7.5 mm

Table 2.6: Overall variation of outputs for Star-CD using the heat partitioning model.

From Figure 2.20, we can see that C_{chen} is more strongly correlated to the wall temperature than $\dot{q}_e$ was, which is consistent with the fact that Chen's correlation is a simpler wall boiling model than heat partitioning. Also, it is again clear that C_{chen} had less of an impact on the other outputs than does $\dot{q}_e$. All other correlations are nearly the same between the two studies.

Figure 2.22 shows a few scatter plots of the data. The strong correlation between C_{chen} and $\bar{T}_w$ can be seen, along with the overall ranges of the outputs, which are similar to those in Table 2.6.

The heat partitioning model for wall boiling is complicated and therefore affects the

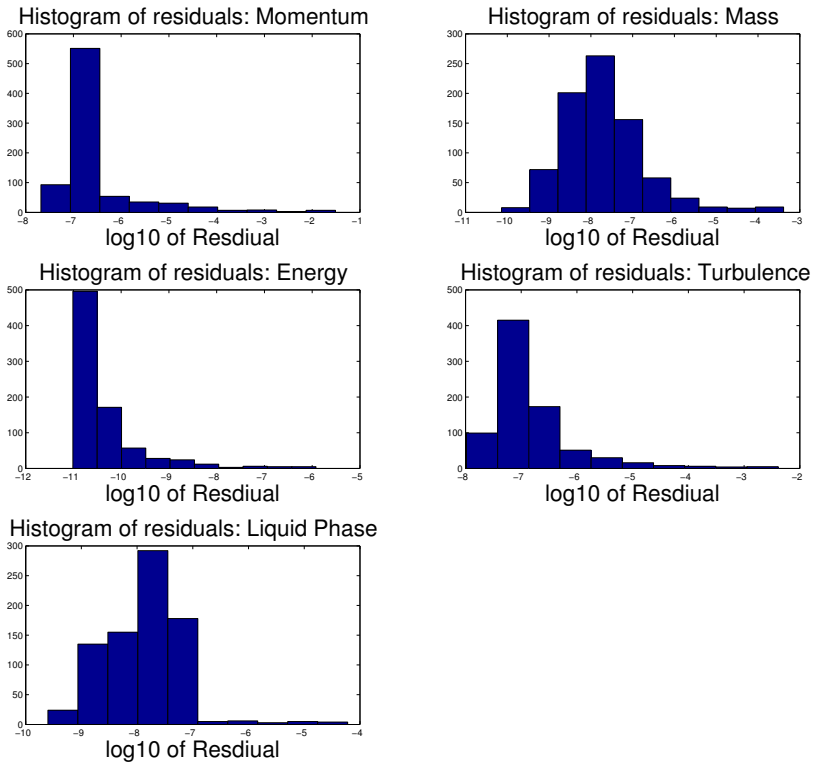


Figure 2.16: Histograms of residuals for heat partitioning model

overall solution and the outputs Δp, $\bar{\alpha}_g$ and $\bar{r}_{\alpha_g}$. The simpler Chen's correlation has less interaction with the rest of the solution. Therefore, it strongly affects the wall temperature, but does not affect the rest of the simulation.

As a final note, we have seen that small values of C_{VM} tend to cause problems with iterative convergence. This can be seen in the upper-middle plot of Figure 2.22, where the cases that did not reach the residual tolerance tend to be clustered toward low C_{VM}. Also, many of the cases that did not converge at all had very low values of C_{VM}. In addition, Figures 2.16 and 2.17 show that the momentum equation generally has the largest residual, probably due to the same problem.

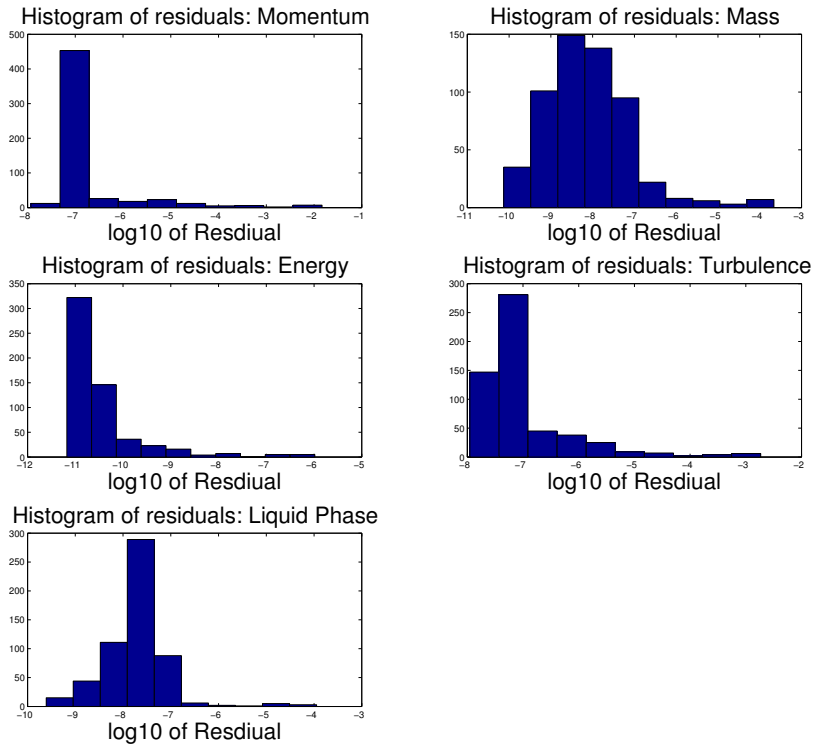


Figure 2.17: Histograms of residuals for Chen's correlation.

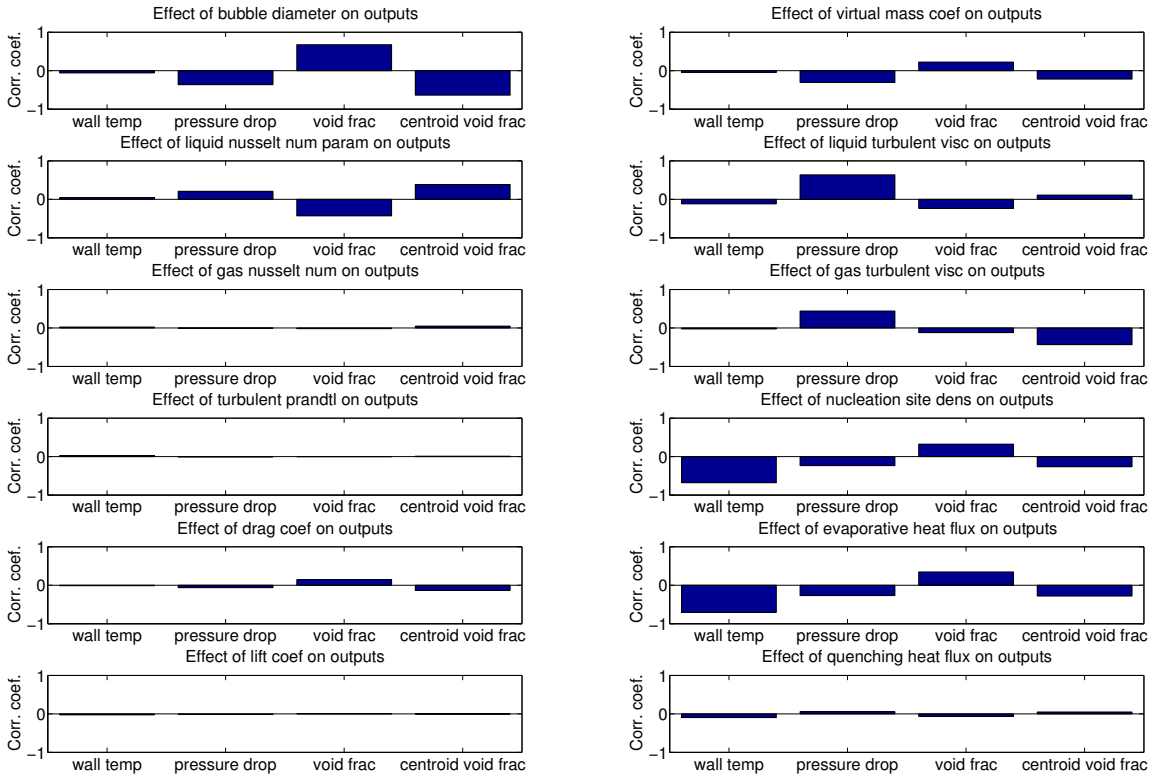


Figure 2.18: Correlation coefficient between parameters and outputs, heat partitioning model

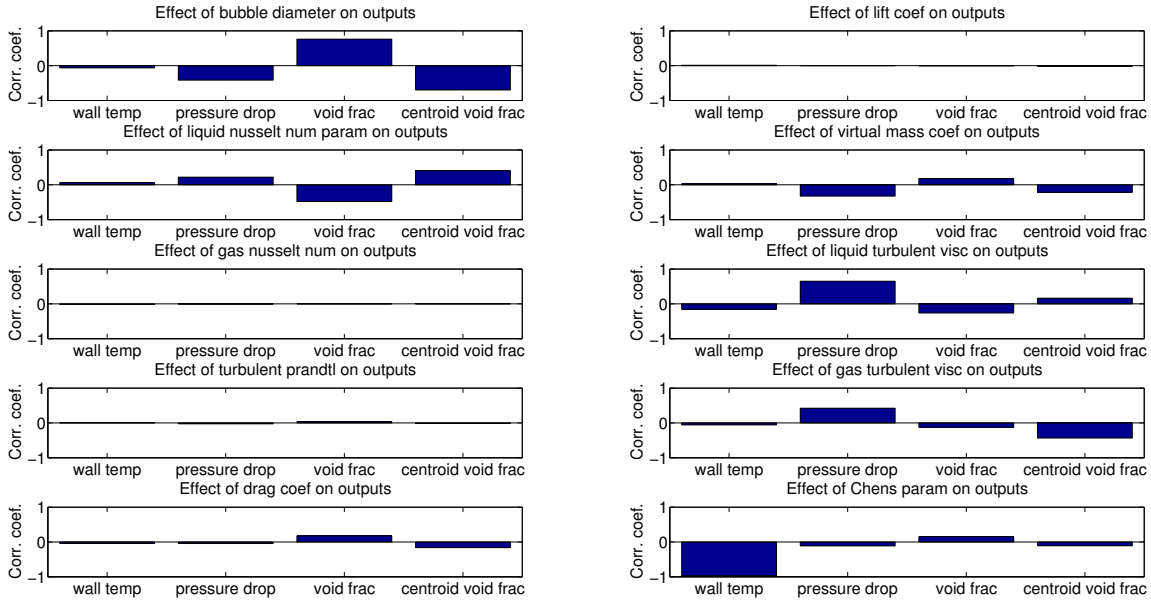


Figure 2.19: Correlation coefficient between parameters and outputs, Chen's correlation

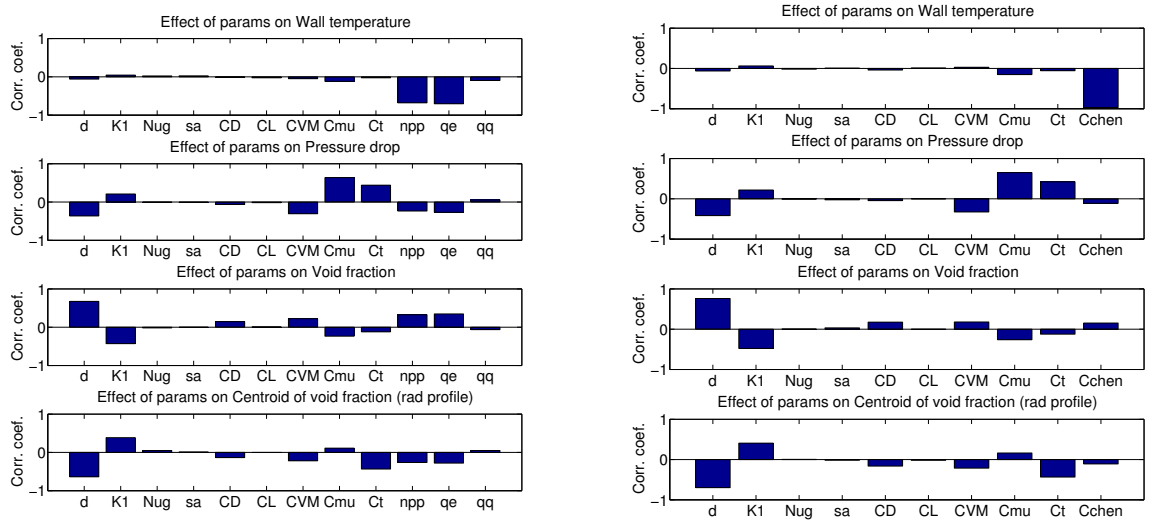


Figure 2.20: Correlation coefficient between outputs and parameters for heat partitioning model (left) and Chen's correlation (right)

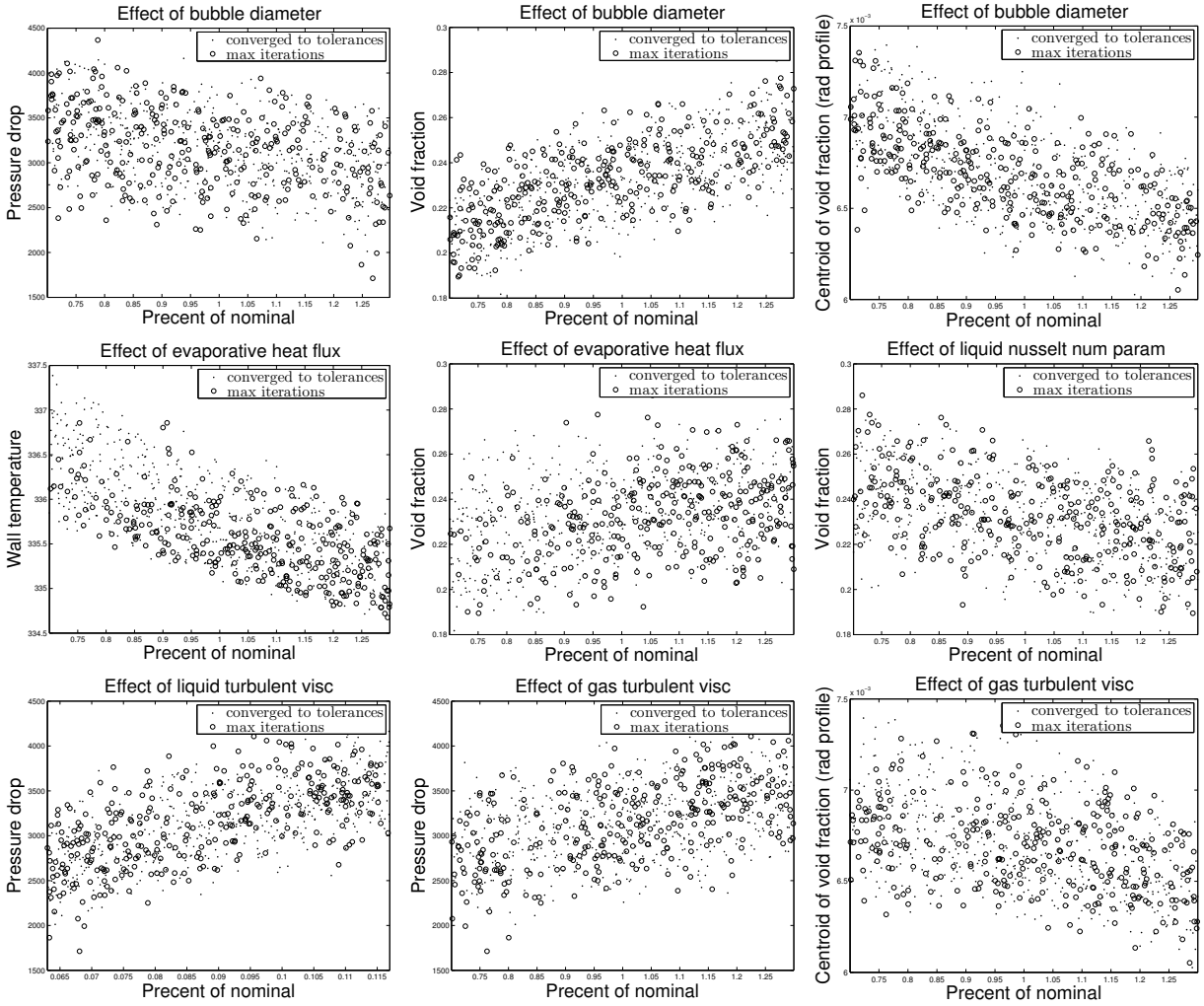


Figure 2.21: Selected scatter plots, parameter values versus outputs for heat partitioning model

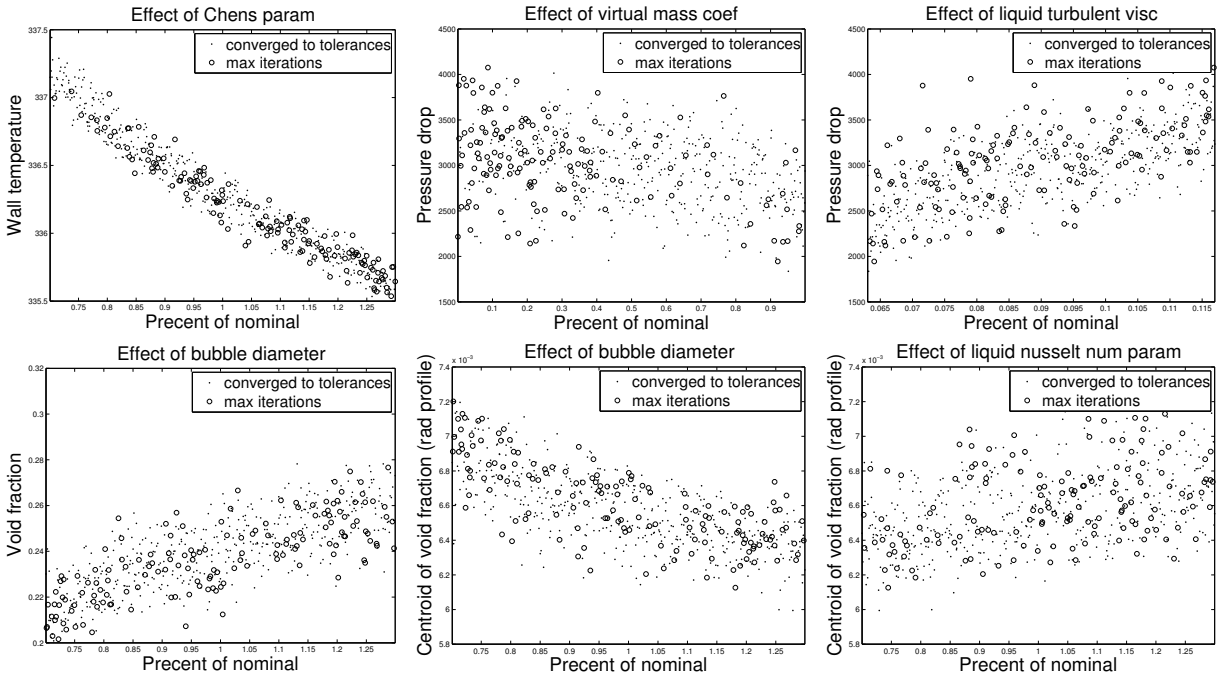


Figure 2.22: Selected scatter plots, parameter values versus outputs for Chen's correlation

2.6.4 Conclusions

This chapter presents the results of a mesh convergence and a sensitivity study of boiling, multiphase, and turbulence models in thermalhydraulics simulations. Most of the models considered, with the exception of a new boiling correlation, are the defaults implemented in the commercial software package chosen for this study, STAR-CD.

The target problem is a simulation of the DEBORA-10 experiment of $R12$ flowing through a vertical pipe with a heated test section. Outputs of interest consist of axial pressure drop, average wall temperature in the heated section, average void fraction at the end of the heated section, and the centroid of the radial distribution of the void fraction at the end of the heated test section. Sensitivity results for this problem demonstrate that bubble diameter is the most important overall parameter affecting the chosen outputs. Parameters governing the boiling model also show significant correlation with the outputs, with evaporative heat flux dominating the other terms in the heat-partitioning model. Turbulence terms are relatively important as well, but phase-to-phase momentum transfer terms do not show significant ef-

fect on the outputs in this problem; however, regarding the lift coefficient, no firm conclusion is possible as the baseline value is small in magnitude relative to the plausible range. Finally, variations in the boiling model based on Chen's correlation have a similar effect on the wall temperature output and a reduced effect on other outputs when compared to variations in the heat-partitioning model.

These conclusions are based on studies for one target problem. The extent to which the results generalize to other geometries and flow conditions requires further study. Nevertheless, we expect that conclusions concerning some of the fundamental underlying processes, specifically the importance of bubble diameter [76] and boiling models, to remain valid for other simulations involving sub-cooled boiling.

2.7 Nphase Study

The overall goal of this work is to perform a sensitivity study of the multiphase fluid dynamics models in Nphase [75]. First, we will describe the grid and model setup, which was complicated by several factors. Following this, we discuss convergence and solution verification. Further work was done to choose relevant parameters and their ranges for the sensitivity study. Finally, the sensitivity study was performed and the results interpreted.

Numerous models exist for simulating multiphase flow, boiling, and turbulence. A comprehensive treatment of all possible formulations is beyond the scope of this work. Instead, we choose to focus on a subset of models that are relatively standard and representative of those used in thermalhydraulics applications. In this section, we compare the software package Nphase-CMFD from Rensselaer Polytechnic Institute with the Star-CD software from CD-Adapco, discussed previously. The boiling model is somewhat more contentious among codes, and in the case of Nphase it is not yet implemented. We therefore "hard-wire" data from the heat partitioning distribution obtained using Star-CD when using Nphase for sensitivity studies.

2.7.1 Simulation Codes

Star-CD employs an Eulerian multiphase model that is representative of treatments in other commercial codes. Boiling is modeled by a variation of the Kurul-Podowski heat-partitioning model. Star-CD serves as the baseline reference code for this work, and it provides heat-partitioning data to Nphase,

Nphase-CMFD is a code developed at RPI. The finite volume, parallel code can handle two- and three-dimensional unstructured grids. Many built-in multiphase models are available, and user defined C-subroutines, with access to all data structures in the code, allow much flexibility. The code generally solves the same equations as Star-CD, see Section 2.3. There are some terms in the governing equations that are modeled slightly differently. A detailed comparison of the equation sets is given in Appendix B. Note, Nphase turbulent treatment includes both high and low Reynolds number models. The low Reynolds number model would presumably overcome some of the difficulties encountered with mesh refinement in Star-CD, allowing much finer resolution near the wall, but at the cost of more time-consuming simulations. In order to compare more directly with Star-CD, only the high-Reynolds number model was used in the results presented here.

Some features that are necessary for the DEBORA case have not yet been implemented. First, there is no wall boiling model. Thus, one must provide some input specification of how the heat flux is partitioned among the liquid and vapor phases. Since this depends on the problem being solved, we used the heat partitioning from the converged Star-CD solution.

Second, the bubble diameter is fixed for the entire domain. This could be overcome by defining multiple populations of bubbles, each with its own fixed size. However, that would require defining interactions between all of the bubble populations, which quickly becomes a very complex task. Instead, we simply specify a fixed bubble diameter based on the Star-CD solution.

Finally, there is no support for transferring solutions between meshes. This means that we must solve on our mesh of interest, starting from a programmable initial condition. A

coarser mesh cannot be used to initialize the solution. This makes it more difficult to reach a converged solution and requires finding an initial condition with a stable path to the solution.

While investigating Nphase, it became necessary to document the major modeling differences between Nphase, Star-CD, and Star-CCM+ for later comparisons. The models used by each code can be compared side by side, along with some information on other models developed by the multiphase community. This comparison also informs the choice of parameters and ranges in the sensitivity studies. The comparison can be found in Appendix B.

The discretizations available in Nphase are first and second order upwind and hybrid finite volume schemes. The mass and momentum equations are coupled and solved implicitly, while other equations are treated explicitly. The solution process is controlled by under-relaxation parameters and pseudo time stepping, along with some robustness enhancements (e.g. capped turbulence production).

2.8 Nphase Problem Setup

2.8.1 Wall Boiling Model

Since wall boiling is not implemented, the user must set the heat partitioning programmatically in a C-subroutine. That is, the user decides, for every cell, the fraction of the wall heat flux that causes liquid heating and the fraction that causes bubble generation and the remaining that causes gas heating. In this example, as in the Star-CD heat partitioning model, we set the gas heating to zero. That is, any heat that does not cause liquid heating will generate vapor. This can be a problem if the void fraction reaches 1.0, since then any excess heat that goes into gas generation is ignored and effectively lost. Thus, we must ensure that during the solution process, none of the cells near the heated wall have a void fraction of 1.0.

In order to have a realistic solution, we set the heat partitioning profile to the profile from the Star-CD DEBORA solution, which is shown in Figure 2.23. We took this approach

because our overall goal is to calculate the model sensitivities in both Star-CD and Nphase, and to compare the codes. The profile taken from Star-CD was run on a uniformly spaced 800x20 mesh with residuals converged as documented in previously. The profile, which ranges from 0 to about 70% of the heat going to gas generation, was fitted by hand with three quadratics. The resulting fit had an estimated L_2 error of 2.25 percentage points.

Gas generation in Nphase requires specifying a porous wall and injecting the gas with a small velocity. It precise velocity is not important, as long as it does not affect the overall momentum balance. The actual condition that is enforced is the imparted mass flux. The chosen velocity of 7.45×10^{-2} m/s was verified to not significantly affect the momentum balance (contributes an additional 0.016% to the overall momentum at the wall).

2.8.2 Bubble Diameter

In the Nphase solution, the bubble diameter was fixed at 7×10^{-4}m. This value was chosen by visually inspecting the Star-CD solution. The converged temperature distribution was used to calculate the bubble diameter that Star-CD would have used (i.e. the $d_b(T)$ correlation), which gave $d_b \in [1.5 \times 10^{-4}, 18.46 \times 10^{-4}]$m with an average of 6.35×10^{-4}m. Thus, the fixed value gives a relatively good estimate of the average bubble diameter. The error was considered small compared to the large range of d_b in Star-CD. The sensitivity study will determine if this range of d_b has a significant influence on the solution and the outputs of interest.

2.8.3 Computational Mesh

The computational mesh to be solved on was a 2D, axisymmetric, structured mesh. Solutions were found using both the low-Re and high-Re turbulence models, which require different meshes. For the low-Re model, the mesh must resolve the entire viscous sublayer near the wall. For the DEBORA test case, this results in a very small grid spacing near the wall of 1.5×10^{-6}m which gives a y^+ slightly below 1 for the converged solution.

For the high-Re model, a much coarser mesh is possible, since we require the mesh spacing near the wall to have a $y^+ \gtrsim 50$. A spacing of 4×10^{-4} results in $50 \lesssim y^+ \lesssim 80$ for the converged solution. The mesh had 30 cells in the radial direction and 400 axially, which was deemed sufficient resolution by visual inspection. The only quantity that varies rapidly with respect to mesh spacing is the radial velocity, where there seems to be a discontinuity at the end of the heated section. In this work, we did not have time to investigate why this happens or if resolving this feature significantly changes the solution. However, we have encountered numerous cases in which the radial velocity distribution is strongly affected by parameters that otherwise have little or no effect on the solution. Thus, we assume that resolving the (relatively small) radial velocity would not significantly change the solution. We should also note that the radial momentum equation almost always had the highest relative errors in converged solutions.

Although a solution on the low-Re mesh was converged, it was prohibitively expensive to do further tests or use it in the sensitivity study. Thus, all of the data and results in this report are based on the high-Re solution.

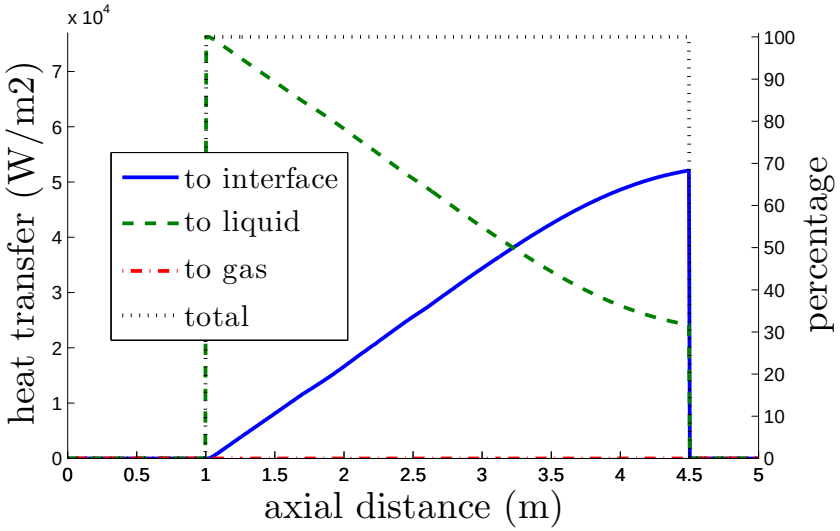


Figure 2.23: Heat partitioning profile from Star-CD DEBORA solution.

2.8.4 Additional Code

In order to reach a converged solution, an extra block of code was written to dynamically modify the false time step and relaxation factors. This was based on a simple strategy to accelerate the solution process if the solution is converging, and to severely slow down progress when the solution is degrading and becoming unstable. The only data available to identify how the solution is converging are the state updates (residuals of the governing equations are not computed). In addition, it would be advantageous to discard bad updates and re-calculate them with a smaller time step. This was possible only for some of the equations, since not all of the update values are available in the user-coded subroutines. In the end, the added capability contributed only modestly to the results.

The recommended method for assessing convergence in Nphase is checking the magnitude of the state updates, specifically the root-mean-squared update (not taking into account the mesh spacing). Since each variable is scaled differently (e.g. pressure is on the order of 10^6Pa, velocity around 1m/s), the updates must be compared to the magnitude of the state. To make this process easier, code was written to compute the root-mean-squared state, and convergence was assessed using the ratio RMS(update)/RMS(state). Note, the mesh spacing was not taken into account here. Also, rather than looking at each velocity component separately, since some velocities could be close to zero, we look for convergence of the magnitude of the velocity vector.

Another feature that was implemented was a ramping-up of the wall heat and mass fluxes during the initial phases of the solution, in order to automate the process of reaching a converged solution. As a result, it was necessary to add additional code to programmatically set the wall boundary condition and change it during iterations.

2.8.5 Getting to a Converged Solution

In order to arrive at a converged solution, we took a number of steps from simple flows to the final DEBORA case. At each stage, the solution from the previous stage was used

as the initial condition. Before moving to the next stage, the solution at a given stage was converged as much as possible. The stages were

1. Compute the approximate, fully developed turbulent pipe flow solution using known profiles from the literature. Velocity and pressure fields were computed.

2. Solve for the single-phase, unheated pipe flow. The goal is to solve for the turbulent quantities.

3. Solve for multi-phase, heated pipe flow with full heat flux but mass flux reduced to 10% of nominal.

4. Slowly increase to full mass flux.

If these steps are not followed (e.g. some were skipped), it is not possible to iteratively converge a solution.

2.9 Nphase Baseline Solution

The convergence of the DEBORA solution while increasing the mass flux is shown in Figure 2.24. The root-mean-square (RMS) of the state update, normalized by the RMS of the state, is plotted for the eight states. Each spike in the plot represents an increase in the mass flux and a restarting of the solution. The enthalpy ("h", the yellow line) converges quite quickly, since the energy equation is linear. The pressure and turbulent quantities ("p", "k", and "e") converge to the point where relative updates are $\approx 10^{-6}$. The void fraction and velocity magnitude ("a" and "u") do not converge as well, but relative updates are still less than 0.1%. Finally, the radial velocity component ("v", the green line) has constant oscillations around 1%. This may be due to the discontinuous nature of the radial velocity profile, as seen in Figure 2.25. Since the radial velocity component is not generally of much interest, and its magnitude is quite small compared to the axial velocity, we did not attempt to further address the issue.

The final solution is plotted in Figures 2.25 and 2.26. The calculation was done with an axisymmetric mesh, so only a cross section is shown. The top of the plot represents the wall of the cylinder, and the bottom is the centerline, and the flow is from left to right. The inlet effects, boundary layer, and radial velocity feature at the end of the heated section are apparent in Figure 2.25. Even though the velocity at the cell adjacent to the wall is far from zero, the wall functions in the High Re trubulence model enforce the correct boundary conditions. The heat transfer from the wall and the bubbles generated are apparent in Figure 2.26. The temperature exceeds the boiling point of 331.3K, so some bulk boiling occurs near the end of the heated section.

The void fraction is nearly constant close to the wall because the lift force is set to zero here. Further from the wall, the lift force causes the bubbles to migrate toward the center of the pipe. The baseline solution shown in Figure 2.26 uses a lift coefficient of $C_L = -0.03$ and the lift force is disabled within one bubble diameter from the wall ($\hat{y}_{\text{wall}} = 1$). The void fraction distribution is highly dependent on the chosen lift model and varies considerably between Nphase, Star-CD, and Star-CCM+. An experimentally determined void fraction profile is available at the end of the heated section, and Figure 2.27 shows this along with various computational models.

2.9.1 Checks and Verifications

The following checks were performed to ensure that the problem was set up correctly.

- Low- and High-Re models give similar solutions
- Verified energy conservation (modified due to gas injection)
- Negligible effect of added momentum due to gas injection
- Successful alternate implementation of mass and energy transfer, similar to Star-CD implementation

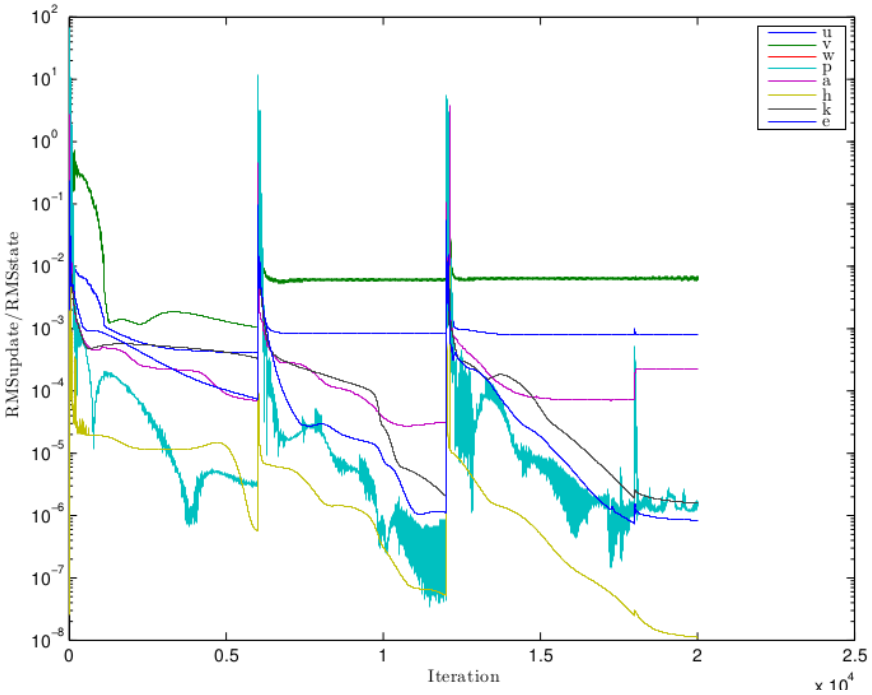


Figure 2.24: Convergence of Nphase baseline solution for DEBORA problem. Spikes are when mass flux was increased and solution process restarted.

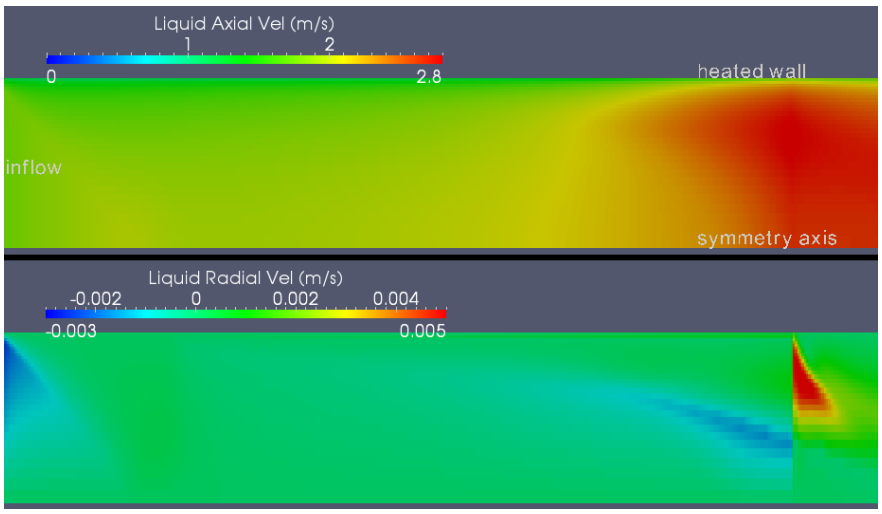


Figure 2.25: Nphase baseline solution for DEBORA problem. Top is axial velocity, bottom is radial velocity of the liquid phase.

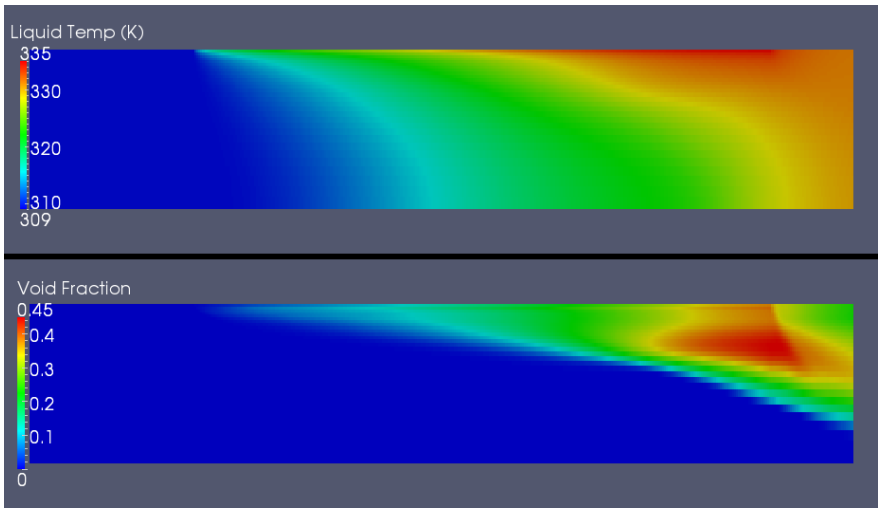


Figure 2.26: Nphase baseline solution for DEBORA problem. Top is temperature of the liquid, bottom is void fraction.

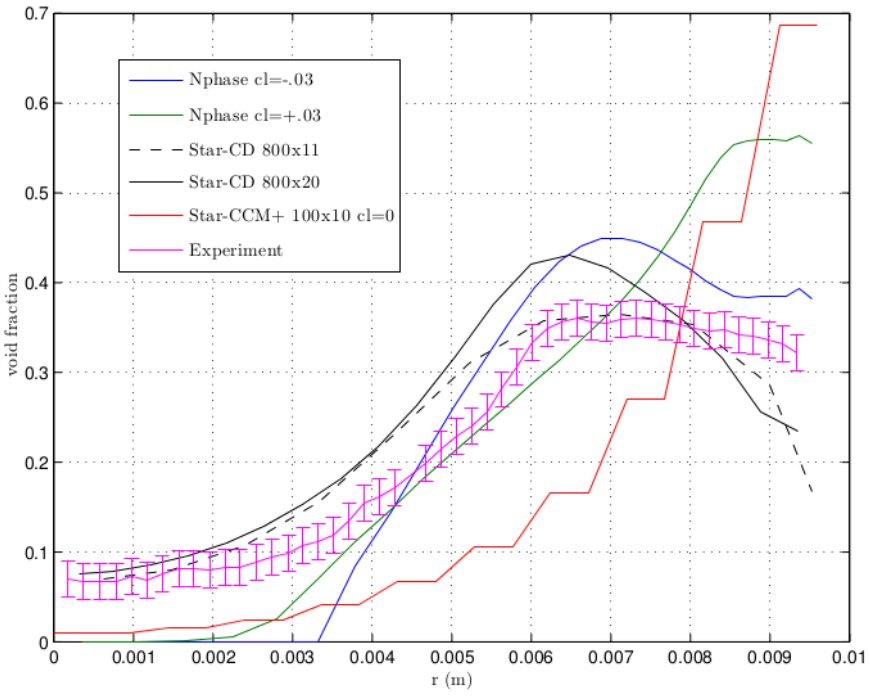


Figure 2.27: Comparison of void fraction profiles at the end of the heated section from various simulations and experimental data. Note, the "step-ladder" effect for Star-CCM+ is merely an artifact of the sampling method chosen; it is not a part of the solution.

- $C_L > 0$ hinders convergence,

  stabilized by setting $C_L(r > R - d_b) = 0$

- Radial velocity artifact at end of heated section is grid-independent

  and appears in Star-CD and Star-CCM+

2.9.2 Heat Partitioning Sensitivity Study

Since a sensitivity study was performed, it was necessary to see if parameter variations modify the wall heat partitioning profile that was taken from the Star-CD solution. Parameters with the strongest effect on outputs were C_L, d_b, and C_μ. A centered parameter study was performed to assess variability in the heat partitioning profile. Although the values of C_L in the Star-CD sensitivity study were conservative ($-0.03 \pm 30\%$), we now wish to explore a larger range of C_L. The literature survey suggests that $C_L \in [-0.3, 0.3]$, but Nphase does not converge well for $C_L \gtrsim -0.01$. Table 2.7 shows the parameters and ranges, and Figure 2.28 shows the resulting profiles. If we define q_l as the heat flux going into the liquid and q_g as the heat causing boiling, then the plot shows

$$\left(\frac{q_g}{q_l + q_g}\right) - \left(\frac{q_g}{q_l + q_g}\right)_{\text{baseline}} = \hat{q}_g - \hat{q}_{g,\text{baseline}}.$$

Thus, parameter variations cause at most a 3 percentage point change in the profile. This is the same order of magnitude as the interpolation error encountered in importing the profile into Nphase.

Table 2.7: Parameters for sensitivity study of heat flux partitioning profile.

Parameter	Range	Reasoning
C_L	$[-0.1, -0.01]$	required for convergence
d_b	$\pm 30\%$	used in Star-CD sensitivity study
C_μ	$\pm 30\%$	used in Star-CD sensitivity study

In order to see how much modified heat partitioning profiles affect the outputs of interest,

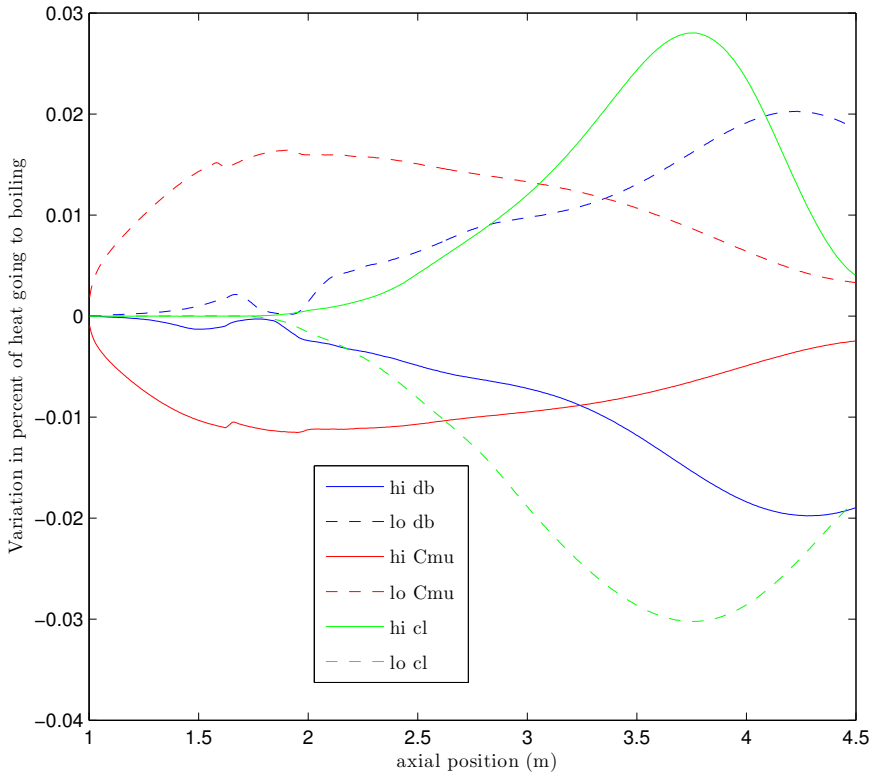


Figure 2.28: Variation of heat partitioning profile from Star-CD DEBORA solution.

Nphase was then run with the (four) heat partitioning profiles for the modified d_b and C_L. The parameters in Nphase remained at baseline, only the heat partitioning was changed. The resulting variation in the outputs is shown in Table 2.8. Previous work on the sensitivity of Star-CD showed that these variations in the outputs are small compared to variations in the outputs from directly altering parameters. Thus, it is sufficient to use the baseline heat partitioning profile for all Nphase runs, provided that output variations less than $\approx 2\%$ are deemed insignificant.

Table 2.8: Variation in outputs for different heat partitioning profiles.

	Δp	$\overline{T}_{\text{wall}}$	$\overline{\alpha}$	$\overline{r}_{\alpha}$
C_L	0.03%	0.07%	2%	0.4%
d_b	0.005%	0.05%	0.4%	0.15%

2.9.3 Preliminary Sensitivity Study

Using the baseline DEBORA solution as the initial condition, a preliminary sensitivity study was performed to verify that the relevant physics were being captured by Nphase. In addition, the study was done to test the link between Nphase and Dakota. Latin-Hypercube samples were generated for four simple parameters that would have strong and obvious effects on the solution. The parameters were acceleration due to gravity, overall magnitude of heat flux, magnitude of mass flux, and inlet velocity. Each parameter was varied by ±30%. The correlation coefficients between the parameters and outputs are shown in Figure 2.29.

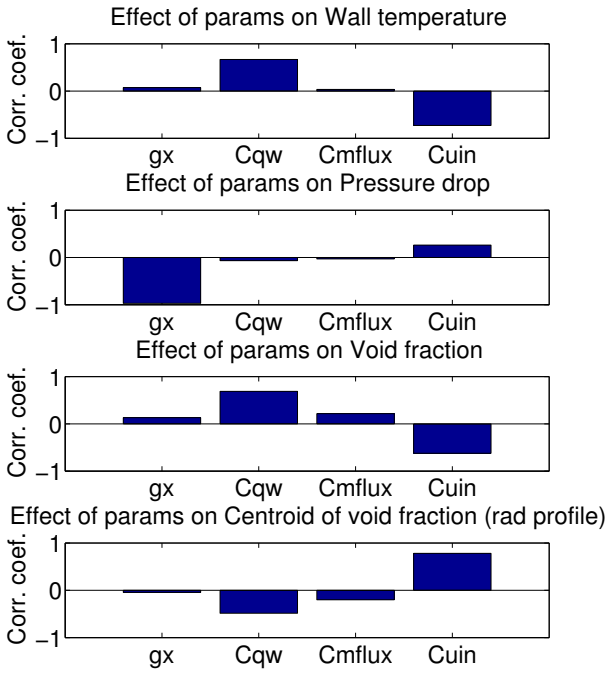


Figure 2.29: Correlation coefficients between parameters and outputs for the preliminary sensitivity study.

The sensitivity shows that Nphase correctly identifies gravity as having a strong effect on pressure drop and little effect on other parameters. The positive correlation of heat flux with wall temperature and void fraction shows that increasing heat flux will make the wall hotter and generate more gas, as expected. Also, it will move the centroid of the void fraction closer

to the center of the pipe (where $r = 0$). The mass flux is shown to have a similar but weaker effect, and it does not affect the wall temperature. Finally, a large inlet velocity reduces wall temperature due to convective cooling, thereby also reducing void fraction. Large velocities also enhance the lift force, keeping the bubbles near the pipe wall.

The solution behaved as expected for all of the parameters. Also, the correlation coefficients give reasonable insight into how the parameters affect the outputs. The next step was to conduct a full sensitivity study on all of the parameters of interest.

2.10 Nphase Sensitivity Study

The full sensitivity study was performed using the same baseline solution as before. The heat partitioning profile was fixed to the baseline profile from Star-CD. After a literature survey, ten high-level parameters and appropriate ranges were chosen as shown in Table 2.9. The study had 1474 useable data points, which is more than the 1024 required for a full 2^k design. Each run in the study had 4000 iterations, and less than 8% of the runs diverged. The rest converged to the point where RMS(update)/RMS(state) $< 1\%$ for all states (except the radial velocity, for which we had looser requirements as explained earlier).

Figure 2.30 shows the correlation coefficients for the full sensitivity study. The plots clearly show that the bubble diameter and the turbulent dispersion coefficient have overwhelmingly large effects on the outputs, compared to the other parameters in the study. Interestingly, the bubble diameter has little effect on the average wall temperature. Both of these models have relatively little experimental evidence, yet have significant impact on the outputs.

By contrast, the lift, drag, and virtual mass forces, turbulence model, and wall heat partitioning model have little effect on the outputs, although many of these models are also lacking in experimental evidence. The heat partitioning model does indeed have a small effect (accounting for only 1.3% variation in the wall temperature), so we can be confident that it was reasonable to use a single heat partitioning profile for the entire study. The liquid

Table 2.9: Parameters and ranges for full sensitivity study.

Parameter	Symbol	Nominal	Range	Reasoning
lift coefficient	C_L	-0.03	$[-0.1, -0.01]$	required for Nphase convergence, [111]
drag coefficient	C_D	Wang fit	$\pm 30\%$	approximate experimental variation (no data for Wang fit itself) [58]
virtual mass coefficient	C_{VM}	1.0	$[0.5, 1.5]$	nominal values in Star-CD (see also [33]) and Nphase, and part of range from [78]
turbulent dispersion coefficient	C_{TD}	$2/3$	$[0.3, 1.5]$	ecompasses much of range from [122] and calculated from Nphase solution using formula in [52].
bubble diameter	d_b	7×10^{-4}m	$[1.5, 20] \times 10^{-4}$m	range from Star-CD, Star-CCM+, Nphase, and [122].
lift force wall distance	$\hat{y}_{\text{wall}}$	1	$[1, 4]$	range for Star-CD, Nphase, and [52, 111, 5].
turbulent viscosity scaling	C_μ	0.09	$[0.07, 0.09]$	calculated for Nphase solution using formulas in [66, 67, 65, 98].
liquid to interface Nusselt number	Nu_l	Modified Ranz-Marshall	$\pm 30\%$	approximate range for many experimental results [84, 22, 54, 123, 118].
gas to interface Nusselt number	Nu_g	analytic (see [102])	$[0, 50]$	encompasses much of analytic form in [102], data from [89], and Star-CD.
heat flux partitioning	$\hat{q}_g$	from Star-CD solution	$\pm 5\%$	heat partitioning sensitivity study (see above).

Nusselt number seems to have a moderate effect on all of the outputs, but this model has enough evidence that time would be better spent improving other models.

Figure 2.31 shows some scatter plots for the various outputs. Table 2.10 shows the ranges of the various outputs, which are probably too large for many engineering applications. The results from this study suggest that more sophisticated models for bubble diameter, such as single or multi-equation interficial area transport models, are needed to accurately simulate two-phase flow. Also, more experimental work should be done to more carefully characterize the turbulent dispersion force.

2.11 Conclusions

The Star-CD and Nphase studies point to the large amount of uncertainty in multiphase flow models and the need for more accurate UQ methods. For even more complex models and larger simulations, each sample requires even more time and the simple UQ methods used here become prohibitively expensive. However, it is clear that the model has simple or even zero variation for some of the parameters. A more efficient UQ method would detect this behavior and exploit it to reduce the number of required samples. Even if the simple UQ methods are affordable, they do not result in much quantitative data – all one can say is that a few parameters have a strong effect on the outputs. It was found that the response is a non-linear function of some of the parameters. A better UQ method would also be able to capture this behavior to improve accuracy of the UQ study results. The UQ method developed in Chapter 3 addresses these challenges.

The UQ studies also show the numerical difficulties with simulating multiphase flows. Solutions of very similar equations can be quite different, and the equations are generally quite difficult to solve. Discretizations can suffer from too little or too much resolution, so the idea of grid convergence (and with it verification) becomes poorly defined. Numerical errors are rarely quantified and have an unknown effect on the UQ studies. For example, some parameters could have a strong effect on the numerical error, or the numerical errors

could be so large that the effects of parameters are masked by them. The error estimation and adaptive techniques developed in Chapter 4 address these challenges.

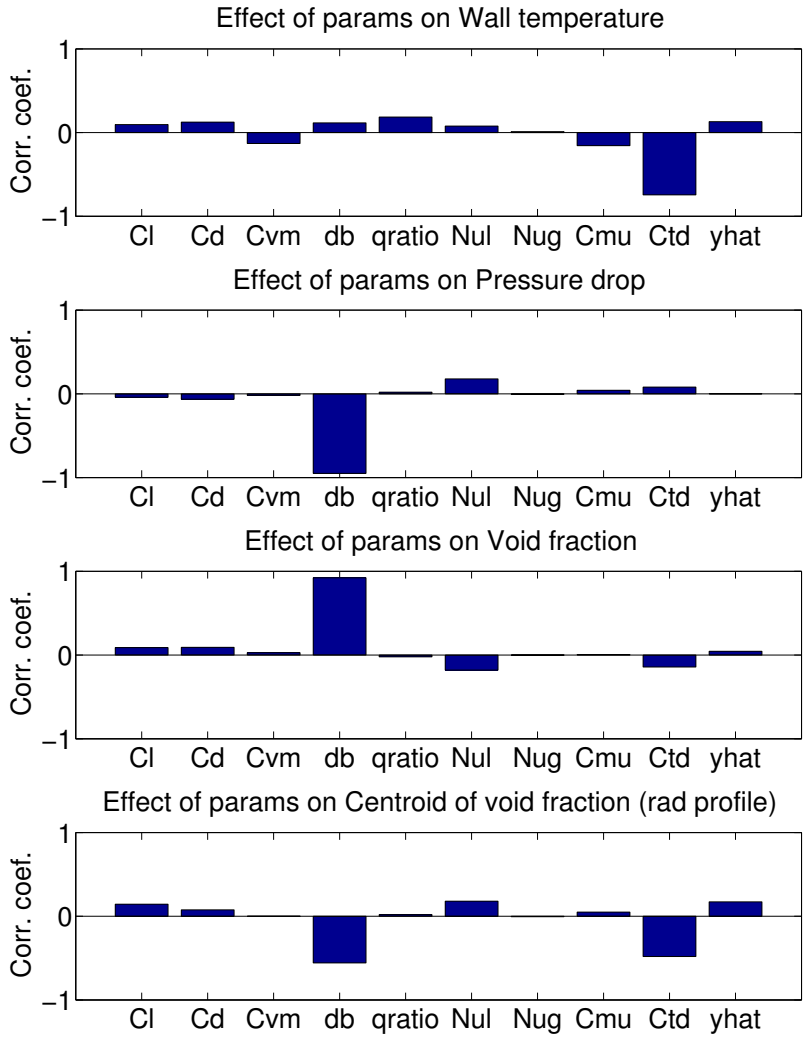


Figure 2.30: Correlation coefficients between parameters and outputs for the full sensitivity study.

Table 2.10: Overall variation of outputs for full Nphase sensitivity study.

Output	Approximate Range
Δp	57.5-61 KPa
$\overline{T}_{\text{wall}}$	326.5-330 K
$\overline{\alpha}$	0.2-0.45
$\overline{r}_{\alpha}$	6.2-8.2 mm

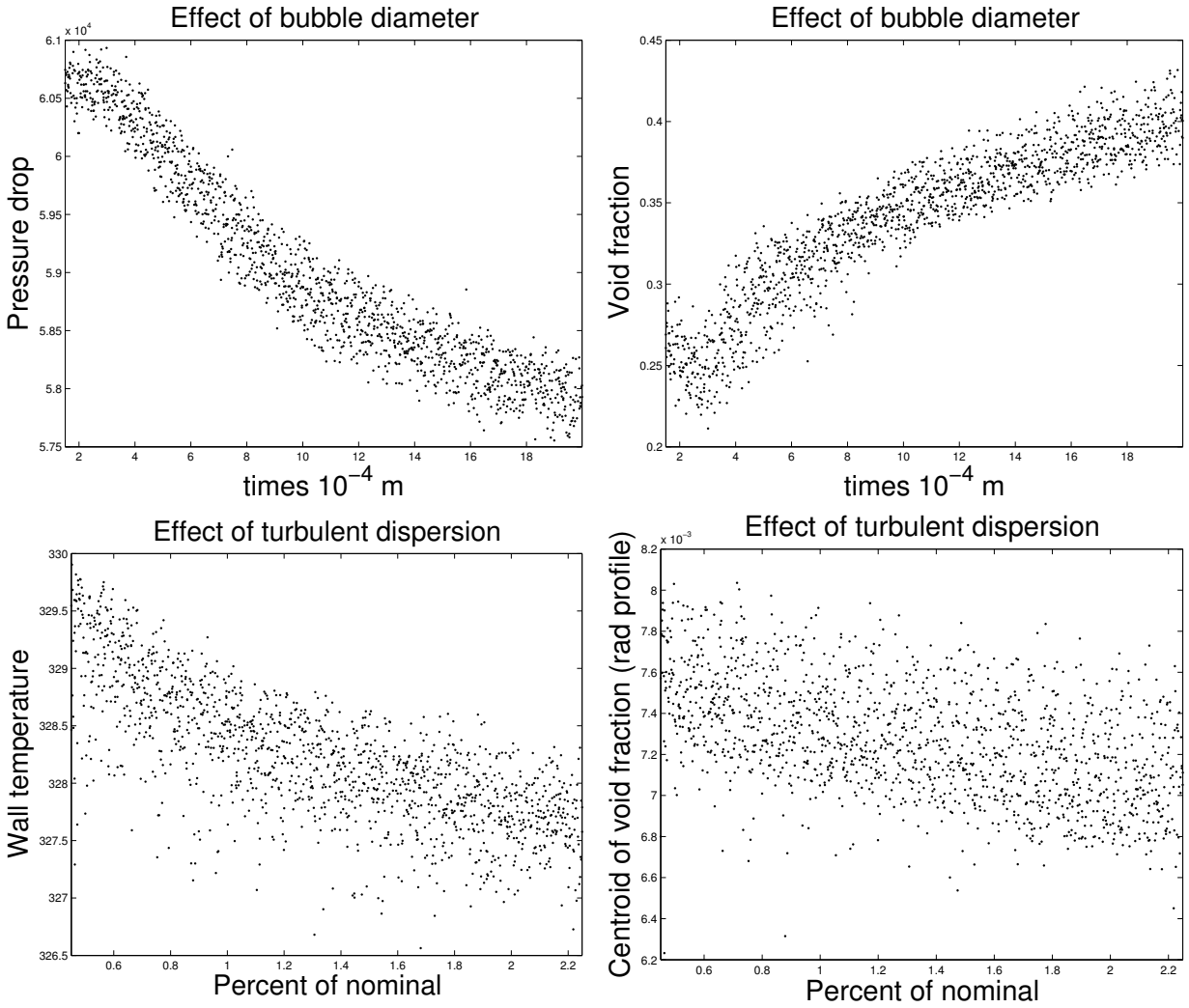


Figure 2.31: Example scatter plots for the full sensitivity study.

CHAPTER 3

A New Method for Uncertainty Quantification

3.1 Motivation

A new UQ method is developed which is more adaptive than previous methods. That is, under certain assumptions on the stochastic function, the current method can produce a more compact representation of the function under consideration, requiring fewer samples to be evaluated to fit the approximation. The assumptions enabling such a compact representation are that the function varies along only a few directions in the stochastic space and that the function has few interactions among those directions. This occurs in practice because the choice of parameters is often somewhat arbitrary. Parameters may be correlated to each other if they are not selected with great care. In this case, the function might vary along a (linear) combination of the parameters (i.e. a vector in parameter space). Methods have been developed for these types of problems which attempt to discover an active (linear) subspace of the parameter space in which the function varies [100, 10, 28]. Other methods assume that parameters are largely independent and reduce the number of interaction terms that are modeled. This work combines these two types of adaptive model building to create even more compact models for a wide class of functions. The result is more accurate uncertainty quantification with fewer samples.

3.2 Algorithm

We describe the prototypical algorithm for adaptive model building. This is the framework of the current method, while details of each step are discussed in the following sections. The goal is to compute a scalar, stochastic output $J = J(u)$ to a specified tolerance with as few samples as possible. The stochastic model for a function $u(\vec{x})$ is a sum of terms, each term associated with a direction $(\vec{d})$ in stochastic space. The linear subspace in which the model is active is $\Gamma = \text{span}\left(\vec{d}\right)$ and the stochastic domain is assumed to be the hypercube $[-1, 1]^{n_d}$.

ALGORITHM 3.1 Prototypical adaptive stochastic model building

1: Initialize $\vec{d_0} = \vec{0}$, $\Gamma = \text{span}\left(\{\vec{d}\}\right)$
2: Compute baseline sample $u_{\text{true}}(\vec{0})$
3: **while** error $<$ tolerance **do**
4: Fit the approximation for u to the samples
5: Compute error metrics
6: Find a new direction $\vec{d^*}$ and order p^* for which the current model is inaccurate
7: Add the term with $\vec{d^*}$ to the approximation for u, update $\Gamma \leftarrow \text{span}\left(\{\vec{d}, \vec{d^*}\}\right)$
8: Remove a direction if doing so reduces error
9: Generate new samples $u_{\text{true}}(\vec{x}_k)$
10: **end while**
11: Compute output $J = J(u)$

The model begins with a single sample and, if that is found to yield a sufficiently accurate model because the function is nearly constant, can terminate there. The adaptive algorithm continually updates the approximation u while samples are added. One advantage of this form is that the process can be terminated at any point and a full model (including an estimate of its accuracy) is available to the user. Another advantage is that the user can interrupt the algorithm at any point and modify $\vec{d}$. For example, expert opinion may point to one dimension as particularly important and as a result extra accuracy may be specified along it.

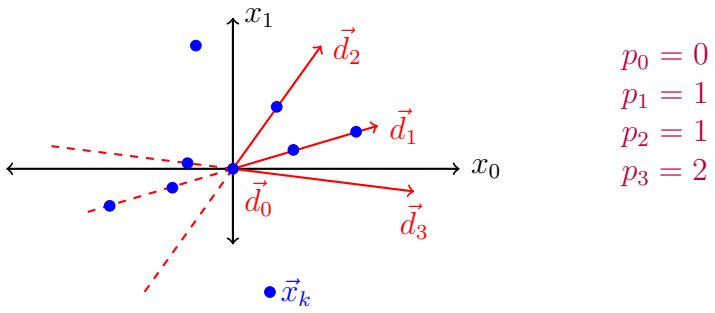


Figure 3.1: A diagram of the stochastic model $u(\vec{x})$. The model states that u varies only along a specified set of directions $\vec{d}_0, \vec{d}_1, \ldots$. Along a direction $\vec{d}_i$, the variation is a polynomial of order p_i.

Section 3.2.1 describes the active linear subspace approximation and how it is fit to the samples. Section 3.2.2 describes the error metrics used. Section 3.2.3 describes the way in which new directions are chosen in order to balance exploration and exploitation. Section 3.2.4 presents a method for removing directions from the approximation. Section 3.2.5 describes the simple way in which sample locations are chosen. Section 3.2.6 describes a method for computing the output of the UQ study, J.

3.2.1 Active Linear Subspace Approximation

For clarity, we will restrict attention to developing a surrogate model for a scalar high-dimensional function $u_{\text{true}}(\vec{x})$, where $\vec{x} \in \Omega_{\vec{x}}$ is a point in the n_d-dimensional space (i.e. a set of parameter values). The surrogate model, called $u(\vec{x})$, will be informed by samples $u_{\text{true}}(\vec{x}_k)$ that, for now, we assume are exact.

The active linear subspace approximation takes the form

$$u(\vec{x}) = \sum_{i=1}^{n_{\text{term}}} \overline{u}_i \left(\vec{d}_i^T \vec{x}\right)^{p_i} \tag{3.1}$$

Here, each term (indexed by i) has an associated direction $\vec{d}_i$ and polynomial order p_i. Each term accounts for variation along a single direction $\vec{d}_i$ and samples $(u_{\text{true}}(\vec{x}_k))$ are used to determine the $\overline{u}_i$ coefficients in the expansion. The advantage of this form of a surrogate

model is that it has the ability to capture any function with a Taylor series but, when the number of terms is truncated, can efficiently represent high-dimensional functions. Many high-dimensional functions that arise in practice are largely active in only some (relatively small) linear subspace. In this form, variability within that subspace can be captured by restricting $\vec{d}$ to lie within it. This restriction can significantly reduce the number of terms in the model, thereby similarly reducing the number of function samples required for an accurate fit. Further, some functions may have two or more linear subspaces that are active but orthogonal, meaning that there are no interactions between the directions (or subspaces). Compared to other active subspace models that capture full interactions within the active subspace, this form naturally allows for (but does not require) restricting interaction terms within the active subspace. Alternatively, functions with anisotropic behavior where a few directions (within the active subspace) have more complex behavior than the others can also be efficiently modeled by adding extra terms along the complex directions with higher p_i. By cutting out high-order terms and interactions, the model has the potential to use fewer terms and thus require fewer samples than other stochastic approximations. Or, from another point of view, this model can improve accuracy at a fixed number of samples. This, in turn, makes the computation of the output of the UQ study, J, to the required accuracy faster, which is the objective of the current method.

If the underlying function $u_{\text{true}}(\vec{x})$ has complex behavior in the entire stochastic space, then the model will require many terms and many samples. By including $\binom{n_d+p_i-1}{p_i}$ terms of order p_i (with non-colinear $\vec{d_i}$), the model will have as many coefficients $\overline{u}_i$ as there are terms in a general Taylor expansion. The model can then be fit to any function that has a Taylor series expansion. This full model would require very many samples due to the curse of dimensionality. For functions with a small active subspace and or anisotropic behavior, the number of terms (and thus samples required) can be substantially reduced.

The directions in the approximation 3.1, $\vec{d_i}$, need not be orthogonal, and in general there will be many more $\vec{d_i}$ than dimensions. To see this, it is helpful to rewrite 3.1 in Taylor-series

form. Consider for example a 2D approximation with the terms as shown in Table 3.1. In this case, the Taylor expansion becomes

$$u(\vec{x}) = \overline{u}_0 x_0^4 + \overline{u}_1 \left(4x_0^4 + 16x_0^3 x_1 + 24x_0^2 x_1^2 + 16x_0 x_1^3 + 4x_1^4\right) \tag{3.2}$$

The goal is to fit this approximation to the true fourth order terms

$$u_{\text{true}}(\vec{x}) = a_0 x_0^4 + a_1 x_0^3 x_1 + a_2 x_0^2 x_1^2 + a_3 x_0 x_1^3 + a_4 x_1^4 \tag{3.3}$$

Now u_{true} is approximated by 5 terms in the Taylor expansion in Equation 3.2, but we do not have independent coefficients for each term. Given that we take samples and perform some fitting to solve for the $\overline{u}_i$, the goal in choosing the directions $\vec{d}_i$ is to find those that yield a good approximation to u_{true}. When optimizing the $\vec{d}_i$ (finding better values than $[1, 0]$ and $[\sqrt{2}, \sqrt{2}]$), we can really only modify two values: the angle of $\vec{d}_0$ and the angle of $\vec{d}_1$. In principle the goal is to match the Taylor expansion in Equation 3.3, so there are in reality up to 5 independent values for the fourth order terms $(a_0, \ldots, a_4)$. Choosing two directions and the two values of $\overline{u}$ lets us fit only 4 of the a_i. Thus, another direction $\vec{d}_2$ must be added to the approximation in order to fit all the fourth order terms of an arbitrary function (i.e. all five a_i's). Note that $\vec{d}_2$ cannot be orthogonal to $\vec{d}_0$ and $\vec{d}_1$, since this is merely a 2D space.

Table 3.1: Example of terms in the approximation 3.1

i	$\vec{d}_i$	p_i
0	[1,0]	4
1	$[\sqrt{2}, \sqrt{2}]$	4

Thus, the $\vec{d}_i$ act as a way to collect terms of the full Taylor expansion together. When fitting the Taylor series terms, a relatively small number of parameters ($(n_d - 1)n_{term}$) is used to jointly vary a large number of Taylor terms ($\binom{p+n_d-1}{p}$ for all of the p^{th} order terms). This is one way to view the dimensionality reduction of the approximation.

The stochastic domain is assumed to be the hybercube $[-1, 1]^{n_d}$, so $\vec{x}, \vec{d} \in \mathbb{R}^{n_d}$. The

directions have two additional restrictions: they are normalized ($|\vec{d}| = 1$) and, since $\vec{d}$ is essentially the same as $-\vec{d}$ (only the sign of $\overline{u}$ will change), $\vec{d}^T\vec{e}_j \geq 0$. Here, $\vec{e}_j$ is the vector with 1 in the j^{th} component and zero elsewhere and j is chosen as the smallest index for which the j^{th} component of $\vec{d}$ is nonzero. The polynomial orders p_i must be non-negative integers. Note that if the $\overline{u}_i$ are fitted from the samples $u_{\text{true}}(\vec{x}_k)$ using a non-exact method, it may be true that $u(\vec{x}_k) \neq u_{\text{true}}(\vec{x}_k)$.

The approximation 3.1 yields a function u that varies only along the $\vec{d}_i$; there is no variation along any direction orthogonal to span $\left(\vec{d}_i\right)$. Thus, the active linear subspace is $\Gamma = \text{span}\left(\vec{d}_i\right)$. A good approximation will result if Γ coincides with the actual linear subspace in which $u_{\text{true}}(\vec{x})$ varies.

3.2.1.1 Basis

Each term in the approximation 3.1 is a monomial along $\vec{d}$. It is well known [11] that fitting a function to a series of monomials of increasing order becomes ill-conditioned. Better conditioning is achieved with a Legendre basis, so 3.1 becomes

$$u(\vec{x}) = \sum_{i=1}^{n_{term}} \overline{u}_i \; \phi_i^{p_i}\left(\vec{d}_i^T\vec{x}\right) \tag{3.4}$$

where $\phi^p(x)$ is the p^{th} order univariate Legendre polynomial. The Legendre polynomial can be generated from the recurrence relation [103]

$$\phi^0(x) = 1, \quad \phi^1(x) = x, \quad (p+1)\phi^{p+1}(x) = (2p+1)x\phi^p(x) - p\phi^{p-1}(x) \tag{3.5}$$

The canonical Legendre functions are orthogonal for $x \in [-1, 1]$. In the current approximation, the arguments to the basis functions are scaled so that they range from -1 to 1 as $\vec{x}$ spans the stochastic domain in the direction $\vec{d}$. For example, in 2D with $n_d = 2$ the polynomial $\phi_i^{p_i}$ associated with $\vec{d}_i = [\sqrt{2}, \sqrt{2}]$ is scaled by $\sqrt{2}$ so that its domain extends from $\vec{x} = [-1, -1]$ to $\vec{x} = [1, 1]$.

The order of the polynomials generally increases as terms are added to 3.1. As stated above, the problem of optimizing 3.1 is simplified by defining a fixed relation $p = p(\vec{d})$. Given a current approximation and a new direction $\vec{d}$, we simply choose the minimum possible order p_{min} such that the new term adds information to the approximation. One simple way to do this is to guess a low value of p and check if the approximation yields unique values of $\overline{u}$[1].

The assumption behind using the minimum possible p is that the Taylor series representation of u includes nonzero coefficients for every term. If this is true, then adding the next higher-order term will generally give a better approximation of u. There are, of course, functions for which this is not true. For example, even functions have no terms with order=$1, 3, 5, \ldots$. The problem can be partially ameliorated by testing, for a given $\vec{d}$, both p_{min} and $p_{min} + 1$. In order to test which order is better, we compare the direction-based error metrics discussed in Section 3.2.2.

3.2.1.2 Least Squares Fitting

A natural way to find $\overline{u}_i$ in 3.1 is to perform least-squares fitting over the sample points. This is done by inverting the least-squares matrix

$$\mathbf{A}^{LS}_{k,j} = (\vec{d}_j^T \vec{x}_k)^{p_j} \qquad \vec{\overline{u}} = \left(\mathbf{A}^{LS}\right)^{-1} \vec{u}_{\text{true}} \tag{3.6}$$

where $\vec{x}_k$ are the sample locations, $\vec{\overline{u}}$ is the vector of coefficients and $\vec{u}_{\text{true}}$ is the vector of sample values. The system is over-determined since there are generally many more samples than terms, $n_{\text{samp}} \gg n_{\text{term}}$ (so $\mathbf{A}^{-1}$ is a pseudo-inverse). This results in a non-exact fit, so $u(\vec{x}_k) \neq u_{\text{true}}(\vec{x}_k)$.

Another method of fitting based on the output error will be developed in Section 3.2.2.4

[1]This can be done by forming the least-squares matrix $A_{i,j} = \phi_i^{p_i}(\vec{d}_i^T \vec{x}_j)$ evaluated at the sample points $\vec{x}_j = \vec{x}_k$. Here we include the new direction $\vec{d}$ as one of the $\vec{d}_i$. Then we force A to be invertible so $p(\vec{d}) = \min_{p \geq 0, |A| \neq 0} p$. Forcing A to be invertible means that a new direction is either independent of previous ones, or its associated polynomial order is increased. When the active subspace is modeled to increasingly high order, the same direction may be included many times in the model, but with sequentially higher p_i.

after the error metric has been introduced.

3.2.2 Error Metric

To decide which direction to add to the approximation 3.1, we would like some measure of how adding that direction will improve the approximation. To that end, we develop a direction-based error metric. Further, by specifying a scalar output of the UQ study of interest, the error metric can be targeted to that output so that it is predicted with high confidence. The following details the development of a targeted, direction-based error metric, $\Delta J(\vec{d})$.

First, suppose the output of the UQ study J, is some functional of the approximation

$$J = \int_{\Omega_{\vec{x}}} j(u, \vec{x}) d\Omega_{\vec{x}} \tag{3.7}$$

This could be, for example the variance in u or some marginal statistical quantity. The function $j(u, \vec{x})$ defines the output; for example, if a point output is desired $J = u(\vec{x}_k)$, then $j = \delta(\vec{x} - \vec{x}_k)u$. As another example, if the average of u is desired, then $j = u$.

Next, a "cheap" error estimator is assumed. This could be, for example, a residual calculation in a finite element simulation or an interpolation error estimate. Both of these error estimates are much faster to compute than a full solution. Given some guess of the value at $\vec{x}$, $u(\vec{x})$, we assume an estimator of the form

$$R(u) \approx u - u_{\text{true}} \tag{3.8}$$

which is much faster to compute than finding u_{true} itself. The extra speed comes at the expense of accuracy. At this time, we do not have a bound on the required accuracy, but the output-based error estimates developed in Section 3.4.2 work well in practice.

The error in J due to the approximation u can be computed (approximately) using the

error estimate R

$$\Delta J = J - J_{\text{true}} = \int_{\Omega_{\vec{x}}} \frac{\partial j}{\partial u}(u - u_{\text{true}}) d\Omega_{\vec{x}} \approx \int_{\Omega_{\vec{x}}} \frac{\partial j}{\partial u} R(u) d\Omega_{\vec{x}} \tag{3.9}$$

If R is exact and the full integral over $\Omega_{\vec{x}}$ is performed, this will yield an exact value for ΔJ with the same expense as computing J_{true} itself.

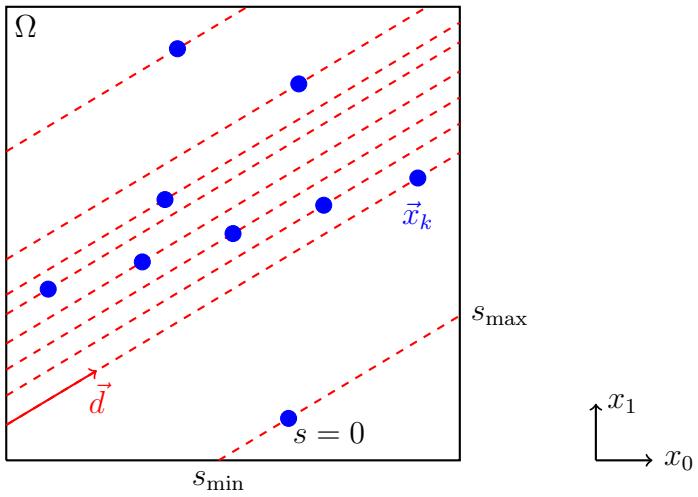


Figure 3.2: Diagram of the targeted, direction-based error metric. The stochastic space here is 2D. The error is integrated from sample points (blue dots) along the direction $\vec{d}$ (red dotted lines).

Next we make more concrete the idea of a direction-based metric. Consider a single sample point $\vec{x}_k$ where both u and u_{true} (and thus $R(u)$) are known[2]. Now traverse the stochastic space from the sample point along a direction $\vec{d}$. As one moves further from $\vec{x}$, the approximation u becomes less accurate and $R(u)$ will increase. Thus, integrating $R(u)$ along a line defined by $\vec{d}$ (which passes through a sample point $\vec{x}_k$) gives an indication of the error due specifically to the approximation *along* $\vec{d}$

$$\int_{s_{\min}}^{s_{\max}} \frac{\partial j}{\partial u} \left[R(u(\vec{x}_k + s\vec{d})) - R(u(\vec{x}_k)) \right] ds \tag{3.10}$$

[2]In general, the error at a sample point is non-zero, $R(u(\vec{x}_k)) \neq 0$.

Here, the limits of integration $s_{\min}$ and $s_{\max}$ constrain the integral to lie inside of $\Omega_{\vec{x}}$

$$\begin{aligned} s_{\min} &= s : s \leq 0, |\vec{x}_k + s\vec{d}|_\infty = 1 \\ s_{\max} &= s : s \geq 0, |\vec{x}_k + s\vec{d}|_\infty = 1 \end{aligned} \tag{3.11}$$

These are straightforward to compute analytically. The integral 3.10 can easily be performed with 1D Gaussian integration with a high enough order to capture the variation in u and R. The sample points x_k are where a full model is executed (potentially requiring considerable computational expense), but the integral along $\vec{d}$ only requires evaluating the surrogate model for u and the residual $R(u)$, which is often much less expensive. For convenience, denote

$$\Delta R(u(\vec{x}_k + s\vec{d})) = R(u(\vec{x}_k + s\vec{d})) - R(u(\vec{x}_k)) \tag{3.12}$$

Since the integral in Equation 3.10 is an estimate of the error along $\vec{d}$ for one sample point $\vec{x}_k$, a better estimate is achieved by averaging over $\vec{x}_k$

$$\frac{1}{n_{\text{samp}}} \sum_{k=1}^{n_{\text{samp}}} \int_{s_{\min,k}}^{s_{\max,k}} \frac{\partial j}{\partial u} \Delta R(u(\vec{x}_k + s\vec{d}))ds \tag{3.13}$$

Figure 3.2 shows this schematically. This set of line integrals can be re-cast as an approximation to a volume integral, similar to the decomposition of the Radon transform [15]. By multiplying the line integrals by the volume of the perpendicular hyperplane, the error metric becomes an approximation of the global integral in Equation 3.9. The volume of a

perpendicular hyperplane is approximately[3] 2^{n_d-1}.

$$\Delta J(\vec{d}) = 2^{n_d-1}\frac{1}{n_{\text{samp}}}\sum_{k=1}^{n_{\text{samp}}}\int_{s_{\min}}^{s_{\max}}\frac{\partial j}{\partial u}\Delta R(u(\vec{x}_k + s\vec{d}))ds \tag{3.14}$$

This is the targeted, direction-based error estimate. Because it is an approximation of a global integral, different values of $\Delta J(\vec{d})$ are directly comparable. If the underlying function u_{true} is constant along some direction $\vec{d'}$ (and the approximation u is also constant), then clearly $\Delta R = 0$ and $\Delta J(\vec{d'}) = 0$. Alternatively, if u is a perfect representation of u_{true}, then once again zero error is detected. In general, $\Delta J(\vec{d})$ yields the error in J due to errors in u along $\vec{d}$.

3.2.2.1 Modifications for Speed

The average over samples in Equation 3.14 may converge to a constant without considering every sample. We compute the value of $\Delta J(\vec{d})$ for an increasing number of samples until the percentage change in $\Delta J(\vec{d})$ is below a tolerance $\tau_{\Delta J}$ (here, 1%). The minimum number of samples to consider is $n_{\text{samp,min}} = 90$, and the change in ΔJ is checked after each additional $n_{\text{samp,step}} = 20$ samples. These values were chosen by trial and error and are implementation-dependent.

3.2.2.2 Post-Modification Error Metric

The error metric as discussed measures the error in a current approximation u along a new direction $\vec{d}$, not necessarily a part of u. Two other versions of the error metric are possible. First, the error due to a direction $\vec{d}$ can be calculated after adding $\vec{d}$ to the approximation or, second, after removing $\vec{d}$ from the approximation. These "post-facto" versions of $\Delta J(\vec{d})$

[3] Though the actual volume depends on the sample location and direction under consideration, it would be very difficult to compute the actual volume for any given scenario. Instead, we give equal weight to each sample. In a sense, we are using a Monte-Carlo method to sample the integral on the $n_d - 1$-dimensional subspace that is perpendicular to $\vec{d}$, with the sample locations being the original samples $\vec{x}_k$ projected onto this subspace. Note, if $\vec{d} = \vec{e}_i$ is a cardinal direction, then the volume is exact.

can be a more accurate representation of how the error in J will actually change if $\vec{d}$ is added to or removed from the approximation, but they are also more expensive to compute. In this work, the latter version is used to find directions that can be safely removed from the approximation. The modified error metric $\Delta\bar{J}(\vec{d})$ uses the fact that $u(\vec{x}+s\vec{d}) = u(\vec{x})$ if $\vec{d}$ is removed from the approximation:

$$\begin{aligned}
\Delta\bar{J}(\vec{d}) &= 2^{n_d-1}\frac{1}{n_{\text{samp}}}\sum_{k=1}^{n_{\text{samp}}}\int_{s_{\min}}^{s_{\max}}\frac{\partial j}{\partial u}\left[R(u(\vec{x}_k+s\vec{d})) - R(u(\vec{x}_k))\right]ds \\
&\approx 2^{n_d-1}\frac{1}{n_{\text{samp}}}\sum_{k=1}^{n_{\text{samp}}}\int_{s_{\min}}^{s_{\max}}\frac{\partial j}{\partial u}\left[\left(u(\vec{x}_k) - u_{\text{true}}(\vec{x}_k+s\vec{d})\right) - \left(u(\vec{x}_k) - u_{\text{true}}(\vec{x}_k)\right)\right]ds \\
&= 2^{n_d-1}\frac{1}{n_{\text{samp}}}\sum_{k=1}^{n_{\text{samp}}}\int_{s_{\min}}^{s_{\max}}\frac{\partial j}{\partial u}\left[u_{\text{true}}(\vec{x}_k+s\vec{d}) - u_{\text{true}}(\vec{x}_k)\right]ds
\end{aligned} \tag{3.15}$$

The modified error metric measures how much u_{true} deviates from a constant along $\vec{d}$. This can be used to detect the active and inactive subspaces and remove directions from the approximation (see Section 3.2.4).

3.2.2.3 Approximation of $\frac{\partial j}{\partial u}$

The error metric is targeted to the output J via the weighting $\frac{\partial j}{\partial u}$. This weight is a scalar function of the stochastic space. One way to think of this is as a stochastic "adjoint" where the governing equation is simply $u - u_{\text{true}} = 0$ and the output is J. It can be computed exactly at every interrogation point, but a potentially cheaper option is to use a surrogate model. In this work we use the same approximation as for u, namely Equation 3.1. The suitability of this surrogate model depends on the complexity of the definition of J. If J is simply the average, $J = \bar{u}$, then $\frac{\partial j}{\partial u} = 2^{-n_d}$ is a constant[4]. The surrogate model will be exact as long as it includes a term with order $p = 0$ (constant term). If J is the variance of u, $j(u) = (u-\bar{u})^2$ and $\frac{\partial j}{\partial u} = 2(u-\bar{u})2^{-n_d}$. In this case the surrogate model will do just as well

[4]Here 2^{-n_d} is the inverse of the volume of the stochastic domain $\Omega_{\vec{x}} = [-1,1]^{n_d}$

for $\frac{\partial j}{\partial u}$ as for u. The surrogate model is less accurate for higher order moments of u. In the extreme case, it is possible to utilize a different set of directions for the approximation of $\frac{\partial j}{\partial u}$ than for u; in this work we will not add this level of complexity.

3.2.2.4 Output Error Fitting

Another way to fit the approximation to the data is to use a targeted error-based approach. This approach is inspired by the error metric developed in 3.2.2. Two modifications are made to the standard least-squares method presented in Section 3.2.1.2. First, the points where the fit is tested (i.e. where $u \approx u_{\text{true}}$) are not just the sample locations $\vec{x}_k$, but also a few points along the directions $\vec{d}_i$ (starting at $\vec{x}_k$). The idea behind this is that a standard least-squares fit attempts to spread out errors evenly among the sample points. In the context of a direction-based approximation, though, we would like a good (i.e. low error) approximation along the directions $\vec{d}_i$ because this leads to a lower error metric and thus lower true error. Another way of looking at this is that including points along the directions $\vec{d}_i$ yields a kind of weighted least squares, where the weights penalize errors along $\vec{d}_i$ more than other directions. For simplicity we take four additional points $\vec{x}_k + s\vec{d}_i$ at $s = 0, \frac{1}{3}, \frac{2}{3}, 1$. This yields $5n_{\text{term}}n_{\text{samp}}$ total points where the fit is tested (called collectively $\vec{x}_m$).

Second, the standard error metric that is minimized in least-squares, $||\mathbf{A}\vec{\overline{u}} - \vec{u}_{\text{true}}||_2^2$, is switched to a targeted error metric[5]

$$\Delta J_{fit} = \sum_{m=1}^{5n_{\text{samp}}} \left[\frac{\partial j}{\partial u}\, R(u(\vec{x}_m))\right]^2 = \left\| \frac{\partial j}{\partial u}(\vec{u} - \vec{u}_{\text{true}}) \right\|_2^2 = \left\| \frac{\partial j}{\partial u}(\mathbf{A}\vec{\overline{u}} - \vec{u}_{\text{true}}) \right\|_2^2 \tag{3.16}$$

By utilizing the error estimate R, we avoid solving for u_{true} at the extra points along $\vec{d}_i$. In this form, we can simply solve a weighted least squares problem with the weights $\partial j/\partial u$. If we extend the method to physical simulations (where u becomes a state vector over the physical space), this can become a non-linear least-squares problem which requires iterations[6].

[5] In a slight abuse of notation, in this formula, $\vec{u}$ is u evaluated at $\vec{x}_m$, and similarly for $\vec{u}_{\text{true}}$

[6] See Section 3.4.2 for the extension to physical space. The least-squares objective function becomes

3.2.3 Optimizing the Approximation

There are two goals in optimizing the approximation in Equation 3.1. The first is to find the $\vec{d}$ and p that result in an accurate approximation. The second is to have a small $\dim(\Gamma)$ so that relatively few samples are required (only as many as are needed to explore the active linear subspace Γ). By succeeding at this goal, the curse of dimensionality is ameliorated (provided u has an active linear subspace) and the output J is calculated to high accuracy with as few samples as possible.

The first goal is difficult to approach in general because it is essentially an optimization problem with both integer (p) and real ($\vec{d}$) values. In many cases, though, it is sufficient to fix $p = p(\vec{d})$ such that when directions are added, the next highest polynomial order is used. By "next highest" we mean the lowest possible p that results in a well-conditioned system[7]. This simplification can fail if u has a Taylor representation where the terms of some order are all zero (e.g. if u is an even function so the terms of order $1, 3, 5, ...$ are all zero). Later a modification will be described that lets the method deal with the problem when only a single order is skipped (but not if, e.g., the terms of order 3 and 4 are all zero). By fixing $p = p(\vec{d})$, the optimization problem is over the $n_d - 1$ independent components of $\vec{d}$.

The second goal is approached by adaptively adding directions to the approximation based on error estimates. By starting with only $\vec{d}_0 = 0$ and carefully adding directions one at a time, $\dim(\Gamma)$ can be kept small. New directions that lie within Γ are added to increase the complexity of the model within the active subspace, potentially without requiring more samples. New directions that lie outside of Γ enlarge the active subspace and usually require more samples. As mentioned above, we must also occasionally *remove* directions from the approximation; this can reduce $\dim(\Gamma)$. An error metric will be derived which is used to

$\Delta J_{fit} = \left\| \frac{\partial j}{\partial K}(K(\mathbf{A}\vec{u}) - K_{\text{true}}) \right\|_2^2$. The problem will remain linear if the physical output $K(u)$ is a linear functional. If not, Newton iterations are performed to minimize ΔJ_{fit}. The starting guess for $\overline{u}$ is the least-squares solution (see 3.2.1.2).

[7] For example, if we already have a quadratic approximation along $\vec{d}$, then adding another term with the same $\vec{d}$ and $p = 1, 2$ would result in an underdetermined system; $p = 3$ is the smallest term that results in a fully determined system. When the approximation has many $\vec{d}$'s with various p, the lowest possible p is determined by trial and error.

decide which directions to add or remove.

The error metric is not inherently smooth due to how the line segments along directions vary in length. As the number of samples considered in the average in Equation 3.13 increases, though, it becomes smoother. Still, many local optima are present, so a global optimization method is needed to discover the true optimum. Examples of the value of $\Delta J(\vec{d})$ are plotted in Figure 3.3. The function is active in a three-dimensional subspace, so there are two angles that specify $\vec{d}$ within that space. The error metric is plotted over the values of the angles (each varies in $[0, \pi]$). Each plot has a different value of n_{samp} used in the calculation of $\Delta J(\vec{d})$. As the value of n_{samp} increases, the plots show that $\Delta J(\vec{d})$ approaches a relatively constant distribution over the angles. In particular, the optimal angles, around the values $(1, 2)$, is approximately constant. This motivates the use of the particle swarm optimization method, discussed next.

3.2.3.1 Particle Swarm Optimization of the Direction

The search for a new direction to add to the approximation is formulated as an optimization problem:

$$\vec{d}^* = \underset{\vec{d}\in D,\ |\vec{d}|=1,\ \vec{d}^T\vec{e}_j\geq 0}{\text{argmax}} \Delta J(\vec{d}) \tag{3.17}$$

That is, we search for the direction with maximum error. The search is split between searching within the active linear subspace ($D = \Gamma$) and outside of it ($D = \Gamma^c = \Omega_{\vec{x}} \backslash \Gamma$). Searching within Γ looks for directions that further refine the approximation and result in increasing its order. This is an "exploitation"-type of search. Searching in Γ^c looks to expand the active linear subspace, i.e. "exploration". By choosing the direction with maximum ΔJ at each adaptive step, the algorithm decides whether exploitation or exploration is more valuable for improving the accuracy of the stochastic output J. In the exploration step, the optimizer looks for the largest error in the un-modeled space Γ^c. In the exploitation step,

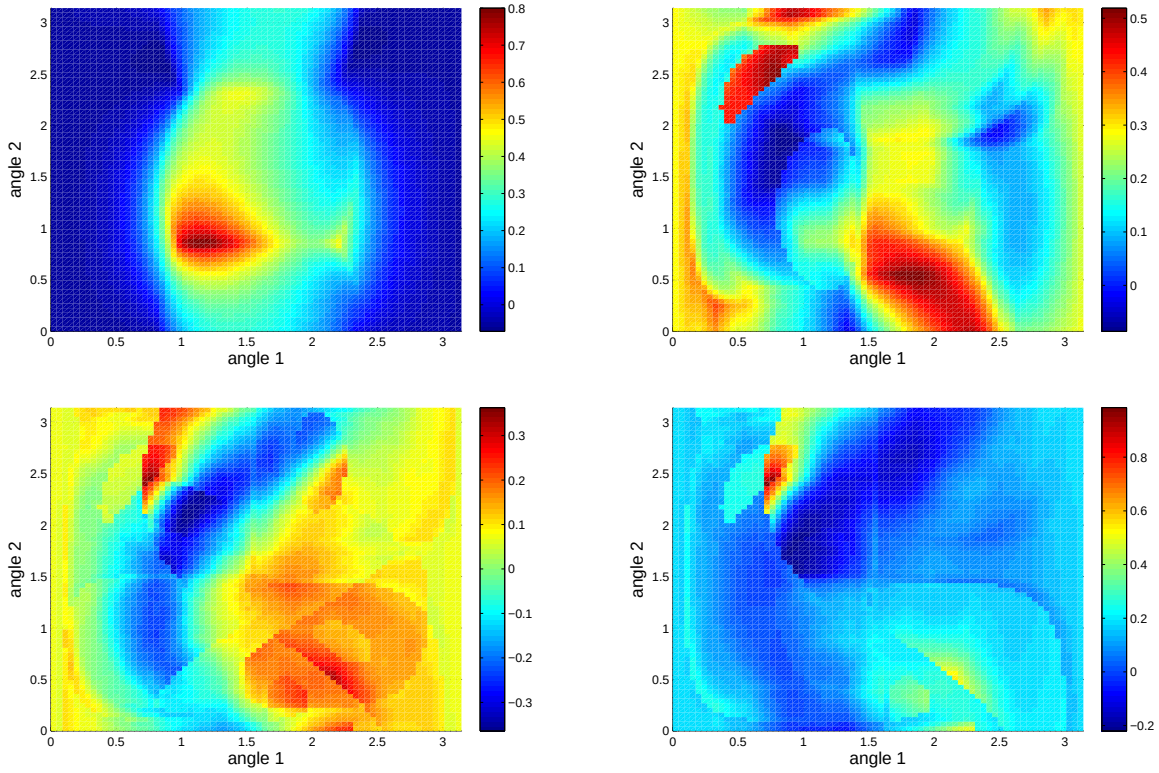


Figure 3.3: Plots of the error metric $\Delta J(\vec{d})$ as a function of two angles that specify the direction $\vec{d}$. Large values of $|\Delta J(\vec{d})|$ correspond to optimal directions. Each plot has a different value of n_{samp}. Starting from the upper left and going clockwise, $n_{\text{samp}} = 2, 5, 10, 20$.

the optimizer looks for the largest error in the modeled space Γ. Thus, the algorithm takes into account both types of error and either refines the model or enlarges the active subspace.

The function $\Delta J(\vec{d})$ has many local optima due to the numerous approximations in its definition. A particle-swarm optimization method is a good choice for optimizing $\Delta J(\vec{d})$ since it is robust to local optima [24, 94]. The optimization should only be performed over the variables that truly modify the direction of $\vec{d}$ (i.e. not the actual components of the vector $\vec{d}$, since these are restricted by $|\vec{d}| = 1$, $\vec{d} \in D$, and the fact that $\vec{d}$ is equivalent to $-\vec{d}$). To accomplish this, the vector $\vec{d}$ is transformed to a set of angles inside D and only those angle are optimized. There are $\dim D - 1$ angles and they all range in $[0, \pi]$[8]. The boundary conditions of the optimization are periodic since an angle of 0 is equivalent to an angle of π. This results in a compact domain for the swarm optimization: $\Omega_{swarm} = [0, \pi]^{\dim D - 1}$. The optimization problem can be rewritten as

$$\vec{d}^* = \underset{\vec{\xi}(\vec{d}) \in \Omega_{swarm}}{\operatorname{argmax}} \; \Delta J(\vec{d}) \tag{3.18}$$

The actual swarm mechanics follow a simple version of the algorithm, see [94]. The particles are initialized with random locations and velocities. In this case, we take a nominal time step of unity, so the maximum particle velocity need not exceed π at any point (otherwise the particle will just "wrap around"). The j^{th} particle's position ($\vec{\xi}_j$) and velocity ($\vec{\eta}_j$) is updated according to

$$\begin{cases} \vec{\eta}_j \leftarrow \chi \left(\vec{\eta}_j + \text{Unif}(0, C_{\text{cognitive}}) \left[\vec{\xi}_{j,best} - \vec{\xi}_j \right] + \text{Unif}(0, C_{\text{social}}) \left[\vec{\xi}_{best} - \vec{\xi}_j \right] \right) \\ \vec{\xi}_j \leftarrow \vec{\xi}_j + \vec{\eta}_j \end{cases} \tag{3.19}$$

Here, $\text{Unif}(a, b)$ is random number (one for each component of $\vec{\xi}_j$). The positions $\vec{\xi}_{j,best}$

[8] In general, a n-vector in spherical coordinates is represented by its length (here always 1) and $n - 1$ angles denoted $\vec{\xi}(\vec{d})$. There is one angle with the range $[0, 2\pi]$ (say $\vec{\xi}_0$) and the other $n - 2$ angles have the range $[0, \pi]$. Since $\vec{d}$ is equivalent to $-\vec{d}$, we can cut off half of the transformed domain. This is done by simply assuming that $\vec{\xi}_0$ is equivalent to $\vec{\xi}_0 + \pi$, effectively reducing the range of $\vec{\xi}_0$ to $[0, \pi]$. Thus, all $n - 1$ angles in the spherical coordinates range in $[0, \pi]$

and $\vec{\xi}_{best}$ are the locations where the maximum value of the objective function was seen by particle j and every particle, respectively. These terms balance local and global exploration with refinement of the current best estimate. The restriction coefficient χ prevents the velocities from growing too large. After each step, particle positions are reset to lie within Ω_{swarm} to enforce the boundary conditions. Nominal values are taken for the constants $C_{\text{cognitive}} = 2.05$, $C_{\text{social}} = 2.05$, and $\chi \approx 0.73$. The parameters $n_{\text{pso,pop}}$, $n_{\text{pso,maxiter}}$, and $n_{\text{pso,samp}}$ respectively denote the number of particles, number of iterations, and number of samples to consider in the error metric (Equation 3.14) respectively. These are heuristically set to 100, 80, and 5, respectively. Larger numbers may result in finding a better optimum, but also increase the expense of the optimization.

3.2.3.2 SQP Optimization

The particle swarm optimization method is good at finding the global optimum for $\Delta J(\vec{d})$ among many local optima. However, PSO is known to be slow to converge to the a local optimum with high accuracy [94]. To further refine the optimum, Sequential Quadratic Programming (SQP) is used. In this method, a single guess of the optimum solution is successively refined by approximating the function as a quadratic. The method is initialized with the best direction found by PSO and the optimization is again performed over the angles $\vec{\xi}$. As steps are taken toward the optimum, approximations to the gradient ($g_{\Delta J}$) and hessian ($H_{\Delta J}$) of ΔJ are created and updated. At each step i, the current quadratic model is

$$\Delta\tilde{J}(\vec{\xi}) = \Delta J(\vec{\xi}) + g_{\Delta J}^T\vec{\xi} + \frac{1}{2}\vec{\xi}^T H_{\Delta J}\vec{\xi} \tag{3.20}$$

and the step $\vec{\xi}_{i+1} = \vec{\xi}_i + \Delta\vec{\xi}_i$ brings the current point to the minimum of the quadratic

$$\Delta\vec{\xi}_i = -\left(H_{\Delta J}\right)^{-1} g_{\Delta J} \tag{3.21}$$

The hessian is updated as

$$H_{\Delta J,i+1} = H_{\Delta J,i} + \frac{q_i q_i^T}{q_i^T \Delta\vec{\xi}_i} - \frac{H_{\Delta J,i}\Delta\vec{\xi}_i \Delta\vec{\xi}_i^T H_{\Delta J,i}^T}{\Delta\vec{\xi}_i^T H_{\Delta J,i}\Delta\vec{\xi}_i} \tag{3.22}$$

with $q_i = g_{\Delta J,i+1} - g_{\Delta J,i}$. The gradient is computed at each step via finite differences. Note, this is the gradient of ΔJ with respect to the angles $\vec{\xi}$. It was mentioned in Chapter 1 that derivatives (of the function u) with respect to parameters are not needed by the current method. This is still true. It would be quite difficult to compute the gradient of $\Delta J(\vec{\xi})$ analytically (one would need to compute the derivative of the adjoint with respect to parameters). In SQP, we make a quadratic approximation to the function $\Delta J(\vec{\xi})$, including approximating the derivatives, but this does not require (and indeed would not be served by) analytic gradients of $u(\vec{x})$. Also note that we may make a linear or quadratic model of u, whose gradients are essentially estimated via least-squares fitting from the samples. Still, direct evaluation of $\nabla u(\vec{x})$ is not required. The implementation of the SQP algorithm is that built in to Matlab [83].

3.2.4 Removing a Direction

The first and simplest way to remove directions from the approximation 3.1 is to remove any $\vec{d}_i$ where the associated $\overline{u}_i \approx 0$. This is done every time the $\overline{u}$ are fitted to the approximation.

Another way to remove directions is due to Russi [100]. The function $u(\vec{x})$ is approximated as a linear function with the gradient varying over the stochastic space. The basic idea is that if a vector $\vec{d}^r$ is found that is orthogonal to *every* possible gradient vector of the solution, then it must be true that u never varies along $\vec{d}^r$. Mathematically, suppose $\overline{\Gamma} = \text{span}\left(\nabla u(\vec{x})\right)$ is the space that spans every possible gradient vector. Then, if $\text{null}\left(\overline{\Gamma}\right) \neq 0$, the nullspace of $\overline{\Gamma}$ contains vectors $(\vec{d}_r)$ that satisfy

$$\vec{d}_r^T \nabla u(\vec{x}) = 0 \quad \forall\vec{x} \tag{3.23}$$

That is, u has no variation along $\vec{d}_r$. Thus, the stochastic space can be partitioned into the active subspace $\Gamma = \text{span}\,(\nabla u)$ and its complement, the inactive subspace. Russi does this by forming a matrix whose columns are various evaluations of ∇u (via finite differences at the sample points). In this work, a matrix with the same column space is formed by evaluating the error estimate. Consider the following combination of error estimates, in the spirit of the post-modification error estimate in Section 3.2.2.2

$$\begin{aligned}
\delta\bar{J}(\vec{d},\vec{x}) &= \frac{-1}{2\Delta s}\left[R(u(\vec{x}+\Delta s\vec{d})) - R(u(\vec{x}-\Delta s\vec{d}))\right] \\
&\approx \frac{-1}{2\Delta s}\left[u(\vec{x}) - u_{\text{true}}(\vec{x}+\Delta s\vec{d}) - u(\vec{x}) + u_{\text{true}}(\vec{x}-\Delta s\vec{d})\right] \\
&= \frac{1}{2\Delta s}\left[u_{\text{true}}(\vec{x}+\Delta s\vec{d}) - u_{\text{true}}(\vec{x}-\Delta s\vec{d})\right] \\
&\approx (\nabla u_{\text{true}})^T\,\vec{d}
\end{aligned} \tag{3.24}$$

In the second line, we assume that the direction $\vec{d}$ has been removed from the approximation, so u is constant along $\vec{d}$ and $u(\vec{x}+s\vec{d}) = u(\vec{x})$. Thus, evaluating the output error with slight perturbations Δs results in an estimate of the derivative. This can be used to calculate gradients and is less expensive than the method of Russi, which requires n_d full solutions for each gradient[9].

Evaluating the $\delta\bar{J}$ along the directions $\vec{e}$ yields an approximation to the gradient[10]. As in

[9] If analytic derivatives are available, both codes could use them to speed this process. However, analytic differentiation with respect to arbitrary parameters can be difficult and is rarely available on commercial simulations.

[10] The error metric can include the weighting $\partial j/\partial u$, which will lead to detecting the union of the active subspaces of $\partial j/\partial u$ and u_{true}. Note, if J is a combination of averages and variances, the active subspace of $\partial j/\partial u$ is contained in the active subspace of u.

Russi's method, the gradient evaluations at various sample points are collected into a matrix

$$G = \begin{bmatrix} \delta\bar{J}(\vec{e}_0, \vec{x}_0) & \delta\bar{J}(\vec{e}_0, \vec{x}_1) & \cdots & \delta\bar{J}(\vec{e}_0, \vec{x}_{n_{\text{samp}}}) \\ \delta\bar{J}(\vec{e}_1, \vec{x}_0) & \delta\bar{J}(\vec{e}_1, \vec{x}_1) & \cdots & \delta\bar{J}(\vec{e}_1, \vec{x}_{n_{\text{samp}}}) \\ \vdots & \vdots & \ddots & \vdots \\ \delta\bar{J}(\vec{e}_{n_d}, \vec{x}_0) & \delta\bar{J}(\vec{e}_{n_d}, \vec{x}_1) & \cdots & \delta\bar{J}(\vec{e}_{n_d}, \vec{x}_{n_{\text{samp}}}) \end{bmatrix} \tag{3.25}$$

The active subspace is detected using the singular value decomposition of $G = USV^{-1}$. In order to form a full matrix G, we require $n_{\text{samp}} \geq n_d$ samples (or full model runs). In contrast, Russi's method requires $n_{\text{samp}} \geq {n_d}^2$ samples to get a full matrix G. If there is a clear drop-off in the singular values (from large to near zero) between the active subspace and the inactive space, then the number of columns of G required can be reduced to $\dim(()\,\Gamma)$ by using a sequential SVD approach [10]. In this case, the current method requires $n_{\text{samp}} \geq \dim(()\,\Gamma)$ and that of Russi requires $n_{\text{samp}} \geq n_d \dim(()\,\Gamma)$. Finally, if gradients with respect to the parameters are available without expense, both methods require only $n_{\text{samp}} \geq \dim(()\,\Gamma)$. A basis for the inactive subspace is found in the columns of U for which the associated singular values are small. In this work, a tolerance of $\tau_{\text{remove}} = 10^{-9}$ is sufficient to detect the truly inactive subspace. In particular, if the j^{th} direction is removed if $s_j / \max(s_i) < \tau_{\text{remove}}$, where s_j is the singular value associated with direction j[11]. Once a basis for the inactive subspace is found, the current directions in the approximation 3.1 are projected onto its complement, the active subspace[12].

Compared to the original method in Russi (using finite difference evaluations of the gradient), this method utilizes relatively cheap error estimates rather than full function evaluations. In the current method, full function evaluations are only needed at sample

[11] Note that the calculation could be done faster by building up the matrix G as samples of the gradient are calculated and terminated if the size of the singular values drops, as in Russi [100]. However, since the error estimates are relatively inexpensive to compute, there is not too much gained by this. In addition, as Russi mentions, particular choices of samples can in theory lead to premature termination.

[12] Note that the approximation within the active subspace need not be a full polynomial approximation. Also, if a direction lies completely in the *inactive* subspace, the term associated with that direction is completely removed from the approximation in Equation 3.1.

points and can be as few as $\dim(\Gamma)+1$. The current method will also target the subspace to the output J, which could reduce its size even further. Removing directions should be done periodically as the model is built. The error estimate $R(u)$ is not exact, so it is possible that incorrect directions are removed at any given step. Also, additional samples that are taken as the model is built may uncover active or inactive directions. It is possible that particular choices of the samples will cause a failure to detect the full active subspace. This is true in Russi's method and the current work. Thus, detection is only guaranteed in probability. See [48] for more details on randomized algorithms.

3.2.5 Generating Samples

After a new direction $\vec{d}$ is added to the approximation, more samples $u_{\text{true}}(\vec{x}_k)$ are usually required in order to capture more variation in u. The samples could be added based on the orientation of the active subspace [100], but in this work they are essentially added randomly with a constant number added for each additional term in Equation 3.1. In this work, the total number of samples is taken to be $n_{\text{samp}} = 2n_{\text{term}}$. In order to help spread out the distribution of points, they are randomly chosen from a Latin Hypercube design with 10 divisions per dimension [85].

3.2.6 Stochastic Domain Integration

Due to the form of the error metric (see Section 3.2.2), the current method requires mostly line integrals to be computed. Some integrals over the entire stochastic domain may be required for a few reasons. Note that these are integrals of the surrogate model u, so they do not require any more samples of the true function to be generated.

- Computing the stochastic output J. This can be done only once, when the adaptive process has converged to the prescribed tolerance. The integral can be computed with a sparse-grid approximation and can, to some extent, take advantage of the known active linear subspace.

In addition, an approximation to J in the spirit of Equation 3.14 can be computed faster than a full integral:

$$J(\vec{d}) = 2^{n_d - 1} \frac{1}{n_{\text{samp}}} \sum_{k=1}^{n_{\text{samp}}} \int_{s_{\min}}^{s_{\max}} j\left(u(\vec{x}_k + s\vec{d})\right) ds \tag{3.26}$$

The direction is arbitrary, and more accuracy can be obtained by averaging over multiple directions. Another option is to simply take latin-hypercube points $\vec{x}_k$ and use the sample average of $j(u(\vec{x}_k))$.

- Computing a certification for J, i.e. a high-resolution error estimate ΔJ (see 3.9). This is never required by the current method, but can be done occasionally to check that the method is performing as expected and after convergence to give a more accurate final error estimate.

- Computing the stochastic output linearization $\frac{\partial j}{\partial u}$. If the stochastic output is a known and simple function of u, then $\frac{\partial j}{\partial u}$ can be found analytically and the integral is avoided. This is the case if J is some combination of means, variances, or other moments of u. An example in which the integral is required is if the output is the probability that u is above some threshold, $J = P(u > u_{thresh})$.

If required, such an integral is computed with an adaptive sparse-grid approximation [60, 45]. The standard Smolyak sparse-grid integration is modified to allow for dimension-adaptivity (note, not direction-adaptivity). An interpolant of u is built with hierarchical polynomials and dimensions are flagged for extra refinement based on a hierarchical error indicator. The refinement choice also takes into account the cost of computing samples. A user-specified "degree of adaptivity" compromises between a conservative (non-adaptive) strategy and a greedy (fully-adaptive) one. The algorithm has shown good performance for a variety of problems, see [60].

The adaptive sparse-grid integration can be more expensive in terms of function evaluations than any of the other integrals required for Algorithm 3.1. In the worst case, the integration could require exponential time even though the algorithm terminates with only a few terms. This may occur if the inherent dimensionality of the problem is very small and not aligned with any of the dimensions $\vec{x}$. In this work, we avoid developing a special integration rule for the active linear subspace. Such an integration rule would likely help in this situation, but it is a complex endeavor due to the potentially irregular shape of the domain. Russi in [100] develops a method for this with mixed results. The integration rules must be created on the fly, require a significant amount of time to achieve reasonable accuracy and suffer from clustering of sample points.

3.2.6.1 Linear Functionals

The approximation in Equation 3.1 can be analytically integrated in the domain since it can be expanded into a polynomial form. For a given term, the multinomial theorem yields the expansion

$$(\vec{d}^T\vec{x})^p = \sum_{i_1+i_2+\cdots+i_{n_d}=p} \frac{n!}{i_1!i_2!\cdots i_{n_d}!} \prod_{1\leq m\leq n_d} (d_m x_m)^{i_m} \tag{3.27}$$

where x_m denotes the m^{th} component of $\vec{x}$. Each term in the above summation can be integrated analytically in the domain $[-1,1]^{n_d}$. This can become expensive to compute as the number of terms in the above sum is $\binom{p+n_d-1}{p}$.

Since the approximation in Equation 3.1 is a linear combination of terms, we can compute any linear or polynomial function of u analytically. Thus, if $j(u)$ is polynomial in u, we can compute $J(u)$ and $\partial j/\partial u$ analytically. Depending on p and n_d, this can be significantly faster and more accurate than the sparse-grid integration described above.

3.3 Results

As a first and simple example, consider modeling a function with a four-dimensional active subspace within a 100-dimensional stochastic space. To illustrate the contributions of the current method, we consider three such functions:

1. (u^1_{true}) The subspace is spanned by $\vec{e}_0, \vec{e}_1, \vec{e}_2, \vec{e}_3$. That is, the function does not vary along the input parameters x_4 through x_{99}. All possible interaction terms within the subspace are non-zero and the function is purely quadratic.

2. (u^2_{true}) The subspace is spanned by four arbitrary vectors $\vec{d}_0, \vec{d}_1, \vec{d}_2, \vec{d}_3$. None of the components of the vectors are zero (they are "diagonal" in the stochastic space). All possible interaction terms within the subspace are non-zero and the function is purely quadratic.

3. (u^3_{true}) The subspace is spanned by four arbitrary vectors $\vec{d}_0, \vec{d}_1, \vec{d}_2, \vec{d}_3$, but there are no interaction terms. The function is higher order along $\vec{d}_0$, but purely quadratic along the other vectors (the four vectors are orthogonal). In particular, the function is

$$\begin{aligned} u^3_{\text{true}}(\vec{x}) = 0.1818(\vec{d}_0^T\vec{x})^2 - 0.0812(\vec{d}_1^T\vec{x})^2 - 0.8993(\vec{d}_2^T\vec{x})^2 \\ - 0.5426(\vec{d}_3^T\vec{x})^2 + \sin(0.321 + 0.21455(\vec{d}_0^T\vec{x} + 0.3)^2) \end{aligned} \tag{3.28}$$

   We attempt to model the function up to fifth order for each method.

The actual numbers of terms in the Taylor expansions[13] are 10, 5050, and 91,962,520 for u^1_{true}, u^2_{true} and u^3_{true}, respectively. The first two functions are somewhat contrived but serve to highlight the advantage of using subspaces rather than adapting on the dimensions. When parameters are chosen ad-hoc, they can easily be correlated and result in a "diagonal" active subspace. The third function represents a more realistic case where some complex behavior is present but it is mostly a function of a single variable. For example, the direction $\vec{d}_0$ could

[13] The number of terms in a p^{th} order expansion of dimension n_d is $\binom{n_d+p-1}{p}$.

represent the amount of boiling occurring in a nuclear reactor simulation. The amount of boiling is affected by many parameters (hence $\vec{d}_0$ is diagonal), while other directions may have little or no impact on the output.

The performance of the current method is compared to the performance of an adaptive Polynomial Chaos (PC) method [12] and the method developed in Russi [100]. The output of interest, J, is set to the average value of u_{true}. The adaptive PC method tests a set of higher order terms for possible inclusion in the model followed by a selective removal of terms. In each case, the classic least-squares error (sometimes called R^2) is used to test terms. The process is repeated for various polynomial orders and levels of interaction. Since we are not interested in precise statistics, it is sufficient to use a monomial basis for the PC expansion. Russi's method searches for the active subspace using gradient evaluations. We assume that analytic gradients with respect to the parameters are not available, so finite differences are used. The sequential SVD approach of [10] used to reduce the number of gradients required to just five ($= \dim(()\,\Gamma) + 1$). Within the active subspace, Russi models the function with all interaction terms.

3.3.1 Function 1

For u^1_{true}, all methods find a (relatively) compact representation of the function and can represent it exactly with second order terms. Russi's method detects the active subspace by making gradient calculations (each requires $n_d + 1$ samples) until the subspace spanned by the gradients does not grow. Since we have a four-dimensional active subspace, this requires at least five gradients, for a total of $5 \times 101 = 505$ samples. We assume that whatever method is used for modeling within the active subspace, it does not require any more samples to be taken. Note that Russi uses a full-term model within the subspace, but in fact all of the linear terms are zero. Thus, Russi's method produces a model with 15 terms, while the current method produces a model with only 11.

The adaptive PC method tests a large number of second order terms and proceeds to

include them in the model and then discard most of them from the model. There are many terms to consider because each term in the Taylor expansion is treated separately. Since the function includes all interaction terms within the active subspace, each of these terms does need to be modeled separately, so the adaptive PC method produces a model of the same size as the current method. However, the adaptive PC method includes many terms (in the inactive subspace) that are later discarded. In order to model these terms in the intermediate step, many more samples are required.

3.3.2 Function 2

For u^2_{true}, the current method and Russi's method perform well again. Now that the active subspace is "diagonal," that is, the basis vectors of the subspace have all non-zero components, many more terms exist in the Taylor expansion. Thus, the adaptive PC method requires keeping many terms, thus requiring many more samples than the other methods. Still, the function is represented exactly with all methods.

3.3.3 Function 3

For u^3_{true}, there is a strong difference between all of the methods. The function is very anisotropic, with high-order variation along one direction, quadratic variation along three directions, and no variation in other directions. The current method exploits this anisotropy to build a high order model focused on the direction $\vec{d}_0$. Russi's method builds a full approximation in the subspace with many terms, so one would assume that the error would be small. However, the method uses samples that are not well spread out – they are the finite difference samples that are clustered. This leads to relatively large error, which could be diminished by taking additional samples that are more spread out.

The goal was to use the adaptive PC method with interaction order and polynomial order up to 5, since terms of that order are represented by the other two methods. However, this would be prohibitively expensive since it would require millions of samples and model

Table 3.2: Results for three test functions with four-dimensional active subspaces in a 100-dimensional stochastic space. Comparison of the current method with an adaptive Polynomial Chaos method and Russi's method. Error is the median of the error metric among all of the directions in the model. For the third function, ranges are given for $\Delta J(\vec{d})$ of the various $\vec{d}$ in the model. *Note, the adaptive PC method for u^3_{true} was only run with terms up to second order.

Method	Function	Number of samples	Number of terms	Error
Adaptive PC		5115	15	10^{-12}
Russi	u^1_{true}	505	15	10^{-12}
Current		101	11	10^{-12}
Adaptive PC		5115	5034	10^{-12}
Russi	u^2_{true}	505	15	10^{-12}
Current		101	11	10^{-12}
Adaptive PC*		5115	5041	0.0039 $[1.56 \times 10^{-4}, 0.0188]$
Russi	u^3_{true}	505	126	40.7 [13.96,599.04]
Current		101	40	0.015 [0.0034,0.1386]

evaluations. With 7 cores, even the third order terms (171,700 of them) took over 36 hours to compute; clearly a full 5^{th} order model with over 91 million terms would take far too long to build. Instead, we compare a quadratic polynomial chaos model. The model is surprisingly accurate given its low order, but still requires a very large number of samples.

The iterative convergence history of the current method is shown in Figure 3.4. For the first two functions, the error decreases slowly until the true active subspace is found and modeled up to second order. Once that occurs, the error is essentially zero. The error for the third function slowly decreases as the iterations progress, though the number of samples still remains at 101. The stochastic approximation becomes enriched with up to 5^{th} order terms. The current method results in intermediate approximations to the functions that have successively increasing complexity and decreasing errors. This can be useful because the accuracy can be closely tailored to the resources available. The other methods require many more terms and samples each time the order is increased and have fewer intermediate results.

Overall, the current method produced approximations with up to 3 orders of magnitude lower error and up to 2 orders of magnitude fewer samples. This is a significant reduction

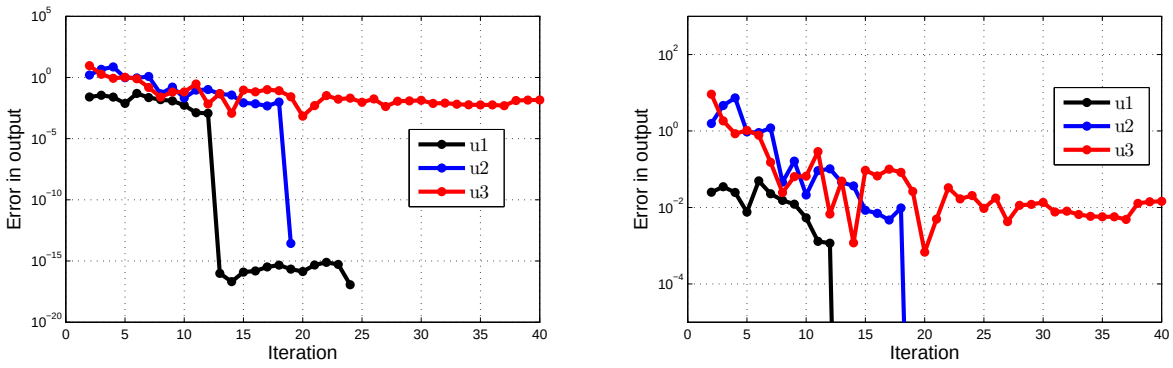


Figure 3.4: Error convergence of the current method applied to three test functions. The median of the error metric among all of the directions in the model is plotted. The number of samples remains fixed at 101 for all iterations. Right plot is a close-up of the left plot.

in computational time, especially when samples are expensive to generate (requiring a full model run).

For a typical run, the current method produces on the order of 20-50 terms in the model, thus requiring perhaps 40-100 samples (full model runs) for a decent least-squares fit. The current way that directions are removed requires $n_{\text{samp}} \geq n_d$, so the method requires at least as many samples as parameters[14]. Of course, more complex functions with high order behavior, larger active subspaces, and tighter error tolerances would result in more terms and require more samples.

3.4 Extensions

3.4.1 Input/prior PDFs

The UQ method models the functional link between input parameters $\vec{x}$ and the model output u. Extending this to deal with probability distributions on the input parameters is quite simple. Indeed, an input (or prior) PDF results in certain values of $\vec{x}$ being more

[14]We assume that the matrix G is full and do not use the sequential SVD approach of [10]. However, for the comparison in Table 3.2, we do assume that Russi's method uses the sequential SVD approach to reduce the number of samples.

"interesting" or relevant than others. Similarly, the output J is a stochastic quantity that determines which values of u are most relevant to the user. These two pre-specified PDFs are linked by the model $u(\vec{x})$. Separating out the PDFs from the more deterministic behavior $u(\vec{x})$ allows modeling (and therefore modeling errors) to be restricted to $u(\vec{x})$, so quite accurate statistics can be computed with the surrogate model.

Practically speaking, any input or prior PDFs are incorporated into the model by modifying the definition of the output J. For example, if the inputs $\vec{x}$ have a joint probability distribution $p_{\vec{x}}(\vec{x})$, and the desired output is the average value of u, then

$$J = E[u] = \int_{\Omega_{\vec{x}}} u p_{\vec{x}}(\vec{x}) \; d\Omega_{\vec{x}} \tag{3.29}$$

Comparing this with 3.7, we simply have $j(u, \vec{x}) = p_{\vec{x}}(\vec{x})u$ (for a uniform input distribution, $j(u, \vec{x}) = u$). As another example, take the output to be the variance of u, we have

$$J = E\left[(u - E[u])^2\right] = \int_{\Omega_{\vec{x}}} (u - E[u])^2 \; p_{\vec{x}}(\vec{x}) \; d\Omega_{\vec{x}} \tag{3.30}$$

so $j(u, \vec{x}) = p_{\vec{x}}(\vec{x}) \, (u - E[u])^2$.

For a more complicated example, suppose we partition $\vec{x}$ into noise variables $\vec{x}_n$ and design variables $\vec{x}_d$. This may represent uncertainty due to operating conditions (noise) and manufacturing (design). Take the output to be the expected variance of a design during operation in the noisy environment, $J = E[\text{var}\,(u|\vec{x}_d)]$. We have

$$J = E\left[\text{var}\,(u|\vec{x}_d)\right] = \int_{\Omega_{\vec{x}_d}} \left\{ \int_{\Omega_{\vec{x}_n}} (u - E[u|\vec{x}_d])^2 \; p_{\vec{x}_n|\vec{x}_d}(\vec{x}_n|\vec{x}_d) d\Omega_{\vec{x}_n} \right\} \; p_{\vec{x}_d}(\vec{x}_d) d\Omega_{\vec{x}_d} \tag{3.31}$$

It may be assumed that the noise is independent of the design so that $p_{\vec{x}_n|\vec{x}_d}(\vec{x}_n|\vec{x}_d) = p_{\vec{x}_n}(\vec{x}_n)$, which may simplify the calculation.

3.4.2 Physical Space

Extending the UQ method to analyze simulations with uncertain parameters is straightforward. The method becomes a Reduced Order Model where the state $\mathbf{u}$, which now is a field variable in the physical domain $\Omega_{\vec{z}}$, is modeled in the parameter space. We seek an approximation to the state $\mathbf{u}(\vec{x}, \vec{z}) \in \Omega_{\vec{x}} \times \Omega_{\vec{z}}$. A simple way to extend 3.1 is as follows

$$\mathbf{u}(\vec{x}, \vec{z}) = \sum_{i=1}^{n_{\text{term}}} \overline{\mathbf{u}}_i(\vec{z}) \ \left(\vec{d}_i^T \vec{x}\right)^{p_i} \tag{3.32}$$

As before, each term accounts for variation along a single direction $\vec{d}_i$ in parameter space. Samples are denoted $\mathbf{u}_{\text{true}}(\vec{x}_k, \vec{z})$ and are solutions to a physical model so they satisfy some residual equation $\mathbf{R}(\mathbf{u}_{\text{true}}(\vec{x}_k, \vec{z}), \vec{z}) = 0$. The samples are used to determine the coefficients $\overline{\mathbf{u}}_i(\vec{z})$ which are similar to physical solutions (they are field variables over $\Omega_{\vec{x}}$), except that they do not satisfy any physical equations. They are analogous to POD or PCA bases which may not be solutions on their own but, when combined properly, generate a good approximation to the solution behavior in parameter space.

We also assume a physical output of interest K, i.e. a relevant scalar quantity for a given simulation. This is in general a functional of the physical solution, and a scalar function of the parameters. $K = K(\mathbf{u}) = K(\vec{x}) : \Omega_{\vec{x}} \times \Omega_{\vec{z}} \to \Omega_{\vec{x}}$. For example, the physical output of a simulation of a nuclear reactor core might be the maximum temperature or the power output.

$$K = K(\mathbf{u}) = \int_{\Omega_{\vec{z}}} k(\mathbf{u}(\vec{x}, \vec{z})) \ d\Omega_{\vec{z}} = K(\vec{x}) \tag{3.33}$$

Note, the function $k(\mathbf{u})$ should not be confused with the index $(\cdot)_k$; the distinction will be made clear by context. The stochastic output J will be assumed to be a function of K only,

$J(\mathbf{u}) = J(K(\mathbf{u}))$. We can write $J : \Omega_{\vec{x}} \to \mathbb{R}$

$$J = \int_{\Omega_{\vec{x}}} j(K(\mathbf{u}))\ d\Omega_{\vec{x}} \tag{3.34}$$

While many physical error estimators are able to drive the adaptation, we focus on an adjoint-based error estimator. To this end, the physical adjoint is defined and extended to the parameter space. Such an extension has been investigated in [35], though there a meshing approach is used in the stochastic domain. The current method uses the same adjoint extension but with the active subspace stochastic approximation, which is expected to perform better for high-dimensional problems.

First define the physical weighed residual (e.g. a finite element residual)

$$R(\mathbf{u}, \mathbf{v}) = \int_{\Omega_{\vec{z}}} \mathbf{v} \cdot \mathbf{r}(\mathbf{u})\ d\Omega_{\vec{z}} \tag{3.35}$$

where $\mathbf{v}$ is the weight or test function. The adjoint or dual state, $\boldsymbol{\psi}$, is defined as the solution to

$$R'[\mathbf{u}](\mathbf{u}, \boldsymbol{\psi}) = K'[\mathbf{u}](\mathbf{u}) \tag{3.36}$$

The bracket notation indicates Fréchet linearization. The adjoint varies in the physical and parameter space just like $\mathbf{u}$. Indeed, we approximate it with the same active subspace model

$$\boldsymbol{\psi}(\vec{x}, \vec{z}) = \sum_{i=1}^{n_{\text{term}}} \overline{\boldsymbol{\psi}}_i(\vec{z})\ \left(\vec{d}_i^T \vec{x}\right)^{p_i} \tag{3.37}$$

Again, the coefficients $\overline{\boldsymbol{\psi}}_i(\vec{z})$ are defined in $\Omega_{\vec{z}}$, like solutions to Equation 3.36, but do not satisfy that equation. The coefficients are computed from samples, $\boldsymbol{\psi}_{\text{true}}(\vec{x}_k)$, which are presumably available at each sample point $\vec{x}_k$. We assume that whenever a physical solution $\mathbf{u}_{\text{true}}$ is calculated, we also compute the associated adjoint $\boldsymbol{\psi}_{\text{true}}$. For linear physical problems,

this is a doubling of the amount of work required, but for non-linear problems the adjoint is usually less expensive to compute than $\mathbf{u}$ since it satisfies a linear equation (Eq. 3.36)[15].

The output error of a single simulation, ΔK, can be estimated with the dual-weighed residual approach (details can be found in Section 4.4 and in [114]).

$$\Delta K = \int_{\Omega_{\vec{z}}} \frac{\partial k}{\partial \mathbf{u}} \Delta \mathbf{u} \; d\Omega_{\vec{z}} = \int_{\Omega_{\vec{z}}} \boldsymbol{\psi} \cdot \mathbf{r}(\mathbf{u}) \; d\Omega_{\vec{z}} = R(\mathbf{u}, \boldsymbol{\psi}) \tag{3.38}$$

The physical discretization error can be estimated by computing this integral on a fine physical mesh. For each sample, the resulting $\Delta K(\mathbf{u}_{\text{true}})$ is akin to "sampling error" from the perspective of the UQ method. For the approximated solution, $\Delta K(\mathbf{u})$ estimates stochastic errors. This form of error metric is therefor advantageous because physical and stochastic errors can be separated. Separating the sources of error enables anisotropic adaptation which can be much more efficient than isotropic adaptation, depending on the problem.

The error metric for adaptation in the parameter space becomes

$$\Delta J(\vec{d}) = 2^{n_d - 1} \frac{1}{n_{\text{samp}}} \sum_{k=1}^{n_{\text{samp}}} \int_{s_{\min}}^{s_{\max}} \frac{\partial j}{\partial K} \Delta R \left(\mathbf{u}(\vec{x}_k + s\vec{d}), \boldsymbol{\psi}(\vec{x}_k + s\vec{d}) \right) ds \tag{3.39}$$

The term ΔR integrates over the physical domain and in full form is

$$\Delta R \left(\mathbf{u}(\vec{x}_k + s\vec{d}), \boldsymbol{\psi}(\vec{x}_k + s\vec{d}) \right) = \Delta K(\vec{x}_k + s\vec{d}) - \Delta K(\vec{x}_k) \tag{3.40}$$

$$= \int_{\Omega_{\vec{z}}} \boldsymbol{\psi}(\vec{x}_k + s\vec{d}) \cdot \mathbf{r}(\mathbf{u}(\vec{x}_k + s\vec{d})) \; d\Omega_{\vec{z}} - \int_{\Omega_{\vec{z}}} \boldsymbol{\psi}(\vec{x}_k) \cdot \mathbf{r}(\mathbf{u}(\vec{x}_k)) \; d\Omega_{\vec{z}} \tag{3.41}$$

Evaluating the error metric amounts to computing adjoint-based error estimates at various points in the stochastic domain (new parameter values) based on the interpolated solution

[15]The adjoint is usually required to be calculated on a refined spatial discretization. A full adjoint solve on the fine space can be very expensive. However, an interpolated or smoothed version of the original (coarse) space adjoint is often sufficient for error estimation and adaptation.

$\mathbf{u}$ and the interpolated adjoint $\boldsymbol{\psi}$. The term $\partial j/\partial K$ is akin to $\partial j/\partial u$ in the scalar case – it can be computed analytically for several classes of functionals or can be computed via a stochastic domain integral.

CHAPTER 4

Multiphase Flow Simulation

In this chapter we focus on solving the drift-flux multiphase flow equations in the physical domain. The goal is to simulate water and steam flowing in a pipe under various conditions including depressurization and heating. This is a relatively simple model of coolant flow inside of a nuclear reactor core. A realistic 3D flow can be quite complicated due to mixing vanes and radiative heating from the nuclear fuel rods, among other phenomena. The one-dimensional transient simulation developed here is complicated enough to capture some of the relevant physics, but still simple enough to be a good platform for testing the UQ method developed previously (see Chapter 3).

In terms of notation, this section will deal with the physical solution $\mathbf{u}$ which consists of state variables (e.g. density, pressure, velocity) defined over space and time. The spatial domain is identified by z (e.g. Ω_z, z_k) while temporal is identified by t. Generally, scalar quantities will be denoted with normal script (e.g. ρ, p) while vector (or matrix) quantities will be in bold (e.g. $\mathbf{u}, \mathbf{R}$).

4.1 Drift-Flux Formulation of the Governing Equations

The area and time averaged drift-flux equations for flow in a one-dimensional channel are derived in [57]. The four equations are conservation of mass for the mixture, conservation of mass for the gas phase alone, and conservation of momentum and energy for the mixture.

The equations as used in this work are:

$$\frac{\partial \rho_m}{\partial t} + \frac{\partial}{\partial z}(\rho_m v_m) = 0 \tag{4.1}$$

$$\frac{\partial (\alpha \rho_g)}{\partial t} + \frac{\partial}{\partial z}(\alpha \rho_g v_m) = \dot{m}_g - \frac{\partial}{\partial z}\left(\frac{\alpha \rho_g \rho_l}{\rho_m} V_{dj}\right) \tag{4.2}$$

$$\begin{aligned}\frac{\partial (\rho_m v_m)}{\partial t} + \frac{\partial}{\partial z}(\rho_m v_m^2) = &-\frac{\partial}{\partial z} p_m + \frac{\partial}{\partial z}\left(\mu_m \frac{\partial v_m}{\partial z}\right) - \rho_m g_z - \frac{f_m}{4 r_{\text{pipe}}} \rho_m v_m^2 \\ &- \frac{\partial}{\partial z}\left(\frac{\alpha \rho_g \rho_l}{(1-\alpha)\rho_m} V_{dj}^2\right) - \frac{\partial}{\partial z}\text{COV}_m\end{aligned} \tag{4.3}$$

$$\begin{aligned}\frac{\partial (\rho_m h_m)}{\partial t} + \frac{\partial}{\partial z}(\rho_m h_m v_m) = &-\frac{\partial}{\partial z}\left(\frac{\lambda_m}{c_{pm}} \frac{\partial h_m}{\partial z}\right) + \frac{q_w P}{A} + \frac{\partial}{\partial t} p_m \\ &- \frac{\partial}{\partial z}\left(\frac{\alpha \rho_g \rho_l}{\rho_m}\left(h_g - h_l\right) V_{dj}\right) + \left(v_m + \frac{\alpha(\rho_l - \rho_g)}{\rho_m} V_{dj}\right) \frac{\partial p_m}{\partial z}\end{aligned} \tag{4.4}$$

The subscript $_m$ denotes a mixture-averaged quantity, while $_g$ and $_l$ denote the gas phase and liquid phase, respectively. The void fraction, or volume fraction of gas, is denoted by α. The gravitational acceleration along the channel is $-g_z$. Thermodynamic properties (λ_l, λ_g, c_{pg}, c_{pl}, μ_l, μ_g) are fixed to a reference state. The mixture-averaged properties are defined as

$$\lambda_m = \alpha \lambda_g + (1-\alpha)\lambda_l \quad c_{pm} = \alpha c_{pg} + (1-\alpha) c_{pl} \quad \mu_m = \mu_l (1+\alpha) \tag{4.5}$$

The separate gas and liquid velocities can be obtained from the mixture velocity and the void fraction by inverting the averaging procedure to obtain

$$v_g = v_m + \frac{\rho_l}{\rho_m} V_{dj} \tag{4.6}$$

$$v_l = v_m + \frac{\alpha}{1-\alpha}\frac{\rho_g}{\rho_m} V_{dj} \tag{4.7}$$

where V_{dj} is the drift velocity. For the enthalpy, it is assumed that each individual bubble has approximately the same enthalpy, h_{gas}. This occurs if heat sources from the wall contribute to evaporating liquid but not to subsequently heating the gas, which is reasonable when α

is small. The gas and liquid enthalpies are then obtained from

$$h_g = h_{\text{gas}} \tag{4.8}$$

$$h_l = \frac{\rho_m h_m - \alpha \rho_g h_{\text{gas}}}{(1-\alpha)\rho_l} \tag{4.9}$$

Calculation of the gas and liquid densities is discussed in Section 4.1.1.

The wall friction term $f_m \rho_m v_m^2 / 4 r_{\text{pipe}}$ is modeled in a very approximate way by neglecting the void fraction distribution in the pipe's cross section. Depending on the flow regime, the void fraction may peak at the wall (upward, heated flow) or may go to zero there (downward, non-heated flow), strongly affecting the wall friction. For a rough approximation, however, the mixture properties can be used in a standard wall friction model. Here, the Darcy friction factor $f_m(\text{Re}_m, \epsilon / r_{\text{pipe}})$ is calculated via an explicit equation given in [99].

$$\begin{aligned}\frac{1}{\sqrt{f_m}} = -2.0 \log \Bigg(& \frac{\epsilon/2r_{\text{pipe}}}{3.7065} - \frac{5.0272}{\text{Re}_m} \log \Bigg(\frac{\epsilon/2r_{\text{pipe}}}{3.827} - \frac{4.567}{\text{Re}_m} \\ & \times \log \Bigg(\left(\frac{\epsilon/2r_{\text{pipe}}}{7.7918} \right)^{0.9924} + \left(\frac{5.3326}{208.815 + \text{Re}_m} \right)^{0.9345} \Bigg) \Bigg) \Bigg)\end{aligned} \tag{4.10}$$

The mixture Reynolds number along the pipe is taken as $\text{Re}_m(z) = \rho_m |v_m| z / \mu_m$. Note that r_{pipe} is the pipe radius and ϵ is a roughness factor.

The covariance term COV_m arises due to the averaging procedure, and relates the average of a product to the product of the averages (in this case of velocities). In most cases, the term can be approximated well via [57]

$$\text{COV}_m = (C_{vm} - 1) \left[\rho_m v_m^2 + \frac{\rho_l \rho_g \alpha}{(1-\alpha)\rho_m} V_{dj}^2 \right] \tag{4.11}$$

$$C_{vm} = 1 + 0.3 \left(1 - \sqrt{\rho_g / \rho_l} \right) \left[1 - e^{-18\alpha(1-\alpha)} \right] \tag{4.12}$$

The wall heat flux term $q_w P / A$ varies along the channel with some prescribed function $q_w(z)$ (in Watts/m^2). Here, P is the (heated) perimeter of the pipe, and A is the cross-

sectional area. The above equations are fully specified once models for pressure (p_m), drift velocity (V_{dj}), and boiling mass flux ($\dot{m}_g$) are specified. These will be detailed below. Equations 4.1 - 4.4 can be written in a general conservation form where the unknowns are the conserved quantities $\rho_m, (\alpha\rho_g), (\rho_m v_m), (\rho_m h_m)$. Denoting these unknowns as a vector $\mathbf{u}$, the equations become

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial}{\partial z}\left[\mathbf{F}^{\text{inv}}(\mathbf{u}) + \mathbf{F}^{\text{vis}}\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial z}\right)\right] = \mathbf{S}\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial z}, \frac{\partial \mathbf{u}}{\partial t}\right) \tag{4.13}$$

Here the inviscid fluxes, which depend only on the state, are separated from the viscous fluxes that also depend on the gradient of the state $\partial\mathbf{u}/\partial z$. The two types of fluxes will be treated separately because they have quite different mathematical structures. The source term requires special treatment as well because it depends on the gradient of the state.

4.1.1 Kieffer/Tait Equation of state

A formulation for two-phase equations of state, which matches well with some experiments, is given in [59]. The gas phase is assumed to be isentropic, while the liquid utilizes an equation of state similar to the famous one by Tait [50], which ignores temperature changes. The equations of state are

$$\rho_g = \rho_{g,ref}\left(\frac{p}{p_{ref}}\right)^{1/\gamma}, \quad \rho_l = \rho_{l,ref}\ \exp\left(\frac{p - p_{ref}}{K_l}\right), \tag{4.14}$$

where K_l is the bulk modulus of the liquid (2.2×10^9 Pa for water). If one assumes that void fraction is known, it is simple to compute all of the other quantities from this and the state variables ρ_m and $(\alpha\rho_g)$:

$$\rho_l(\alpha) = \frac{\rho_m - (\alpha\rho_g)}{1-\alpha}, \quad p(\alpha) = p_{ref}\left(1 + \frac{K_l}{p_{ref}}\log\left[\frac{\rho_l}{\rho_{l,ref}}\right]\right), \quad \rho_g(\alpha) = \rho_{g,ref}\left(\frac{p}{p_{ref}}\right)^{1/\gamma} \tag{4.15}$$

However, finding the void fraction from the input states requires a numerical root finder. Therefore a relaxed Newton method is set up to solve for the void fraction. The equation being solved is

$$R^{eos}(\alpha) = \left[\alpha \cdot \rho_g(\alpha) - (\alpha\rho_g)\right] \frac{p^*(\alpha)}{p_{ref}} = 0 \tag{4.16}$$

where the first term is computed from the guessed void fraction. Normally a solution is found in which $\alpha \cdot \rho_g(\alpha) = (\alpha\rho_g)$ for some α in the range $[0, 1]$. The residual is scaled by the pressure to ensure that in the limit of $p = 0$, a solution is found with $R^{eos} = 0$. The pressure p^* is a modified version of the pressure calculated from α. If the input to the equation of state has $(\alpha\rho_g) < 0$, the pressure can be negative or even imaginary. So, we use the modified pressure

$$p^*/p_{ref} = 10^{-3} \log\left(1 + \exp\left(10^3 p/p_{ref}\right)\right) \tag{4.17}$$

A similar transformation is used in [86] to ensure a positive value of turbulent working variable[1].

The relaxation factor for the Newton iterations is set to enforce positivity of the pressure and to ensure that $0 \leq \alpha \leq 1$. Since analytic derivatives are desired for the solver, the derivatives of the computed quantities (pressure, void fraction, etc) with respect to the state must take into account the fact that $R^{eos} = 0$ as follows:

$$\begin{aligned} \frac{d\alpha}{d\mathbf{u}} &= -\left(\frac{\partial R^{eos}}{\partial \alpha}\right)^{-1} \frac{\partial R^{eos}}{\partial \mathbf{u}}, \\ \frac{d(\cdot)}{d\mathbf{u}} &= -\frac{\partial(\cdot)}{\partial \alpha}^T \left(\frac{\partial R^{eos}}{\partial \alpha}\right)^{-1} \frac{\partial R^{eos}}{\partial \mathbf{u}} + \frac{\partial(\cdot)}{\partial \mathbf{u}} = \frac{\partial(\cdot)}{\partial \alpha}^T \frac{d\alpha}{d\mathbf{u}} + \frac{\partial(\cdot)}{\partial \mathbf{u}} \end{aligned} \tag{4.18}$$

[1]The scale factor 10^3 results in modifications of less than 1% for $p/p_{ref} > 4 \times 10^{-3}$ and less than 0.01% for $p/p_{ref} > 8 \times 10^{-3}$. The changes in the solution due to the modified pressure are very small because the lowest pressure encountered in this work is around 20×10^{-3}. For numerical stability, it is necessary to set $p^* = p$ for $p/p_{ref} \gtrsim 10^{-2}$.

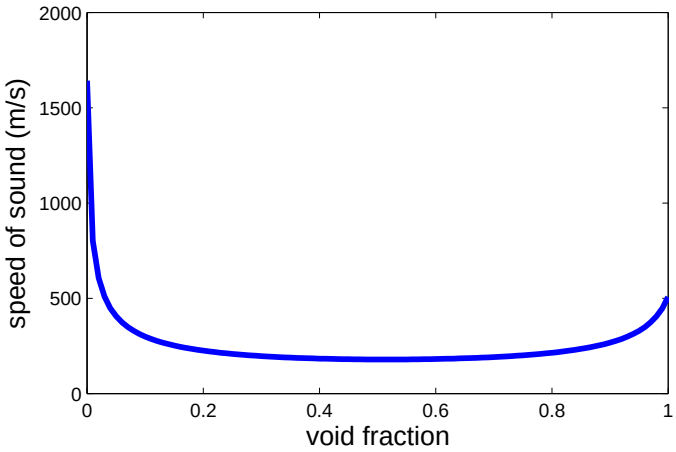


Figure 4.1: Two-phase speed of sound calculated with the Kieffer-Tait equation of state.

The iterations required for this method make it somewhat costly, but it is necessary for an accurate solution. The state variables resulting from the iterations are needed at each integration point and at the element interfaces. The iterations are generally stable since the iterate (void fraction) is known to lie in the interval $[0, 1]$. The speed of sound computed by this method is shown in Figure 4.1 and is in good agreement with some of the experimental data in [59]. However, it is important to note that in that paper there is a complex range of behavior for similar models at very low void fractions ($\alpha < 0.1$) and even the "true" physical behavior there is not well understood.

4.1.2 Drift Velocity

Of the many drift velocity correlations that have been studied, this work will focus on just three representative ones. Some authors pair drift velocity correlations with boiling correlations but here they are treated as separate. A drift velocity correlation relates V_{dj} and C_0 only to flow parameters. C_0 is called the distribution parameter – it is a property of the flow that arises from averaging.

The first and simplest model is essentially no model at all. The drift velocity is assumed zero ($V_{dj} = 0$) and the distribution parameter is assumed unity ($C_0 = 1$). These values are

actually attained when the fluid is static.

The second model is due to Ishii and is relatively coarse. The drift velocity assumes a fully developed vertical flow with spherical bubbles and V_{dj} is modeled directly. The distribution parameter is basically constant except at low void fractions, where it reduces to zero.

$$V_{dj} = \frac{(C_0 - 1)v_m + \frac{2}{9}\frac{g}{\mu_l}d_b^2(1-\alpha)^3(\rho_l - \rho_g)}{1 - (C_0 - 1)(1 - \rho_l/\rho_m)}, \quad C_0 = \left(1.2 - 0.2\sqrt{\frac{\rho_g}{\rho_l}}\right)\left(1 - e^{-18\alpha}\right) \tag{4.19}$$

Here, d_b is the bubble diameter, which, in this work, is assumed constant for the entire flow.

The third model is due to Chexal and Lellouche and is quite involved; a full description can be found in [80] and in Appendix A. Analytic derivatives were taken and confirmed to be accurate via finite-difference testing.

4.1.3 Flashing Model

The most basic type of boiling in pipes occurs when the pressure reduces below the saturation pressure or the temperature rises above the boiling point. This is called flashing because the change from liquid to gas can happen quite quickly. Other boiling pheonomena, including subcooled boiling, nucleation, and the critical heat flux, are not modeled here for simplicity. The model presented here is derived from [9].

The flashing model employs heat transfer correlations to determine heat flux from the gas to the liquid and from the liquid to gas. These are

$$\mathrm{Nu}_g = 26, \qquad \mathrm{Nu}_l = 2 + \left(0.4\mathrm{Re}^{1/2} + 0.06\mathrm{Re}^{2/3}\right)\mathrm{Pr}^{0.4} \tag{4.20}$$

The Reynolds number pertains to the gas bubbles in bulk liquid

$$\mathrm{Re} = \frac{\rho_l V_{dj}^2 d_b}{\mu_l} \tag{4.21}$$

Note, in the inviscid limit we take $\mathrm{Nu}_l = 2$. The heat transfer coefficients are then computed

from the Nusselt numbers as HTC $= \mathrm{Nu}\lambda/d_b$, where λ is the thermal conductivity. In addition to the heat transfer from the liquid and gas, a third "flashing coefficient" is used in [9] to stabilize the model. The flashing coefficient causes a large amount of gas to be generated when the liquid enthalpy exceeds saturation. We use a modified form with a cosine (instead of a hyperbolic tangent)

$$\mathrm{FC} = \begin{cases} 0, & h_l < h_{l,sat} \\ 10^4 \ \frac{1}{2}\left(1 - \cos\left(\frac{\pi}{\delta h}(h_l - h_{l,sat})\right)\right), & h_{l,sat} \le h_l < h_{l,sat} + \delta h \\ 10^4, & h_l \ge h_{l,sat} + \delta h \end{cases} \tag{4.22}$$

This form allows direct control over the distance over which flashing occurs. The numerical solution will be difficult to converge if the enthalpy grows from $h_{l,sat}$ to $h_{l,sat} + \delta h$ in one grid cell or less. Thus, δh can be set to

$$\delta h > \frac{\partial h}{\partial z}\Delta z \tag{4.23}$$

for a given grid cell or to the maximum value of $(\partial h/\partial z)\Delta z$ over all spatial cells. In this work, we will use a safe value of $\delta h = 10\mathrm{kJ/kg}$ for all solutions. The value was chosen by experimentation and observation of the mixture enthalpy. Problems with high heat transfer rates may require a larger constant or another way of setting δh.

The total heat transfers $q_g, q_l, q_{\text{flash}}$ are then computed[2]

$$q_g = A_i \ \mathrm{HTC}_g \ \frac{h_g - h_{g,sat}}{c_{pg}}, \quad q_l = A_i \ \mathrm{HTC}_l \ \frac{h_l - h_{l,sat}}{c_{pl}}, \quad q_{\text{flash}} = \mathrm{FC} \ \frac{h_l - h_{l,sat}}{c_{pl}} \tag{4.24}$$

Here, A_i is the interficial area density. A simple correlation for this assumes spherical bubbles:

[2]These are defined as the heat transfer from the gas (or liquid) to the interface (some will be negative).

$A_i = 6\alpha/d_b$. Finally, the mass transfer is related to the heat transfer imbalance

$$\dot{m}_g = \frac{q_g + q_l + q_{\text{flash}}}{\Delta h} \tag{4.25}$$

The normalizing factor Δh is the difference between the enthalpies of the gas and liquid during boiling or condensation. As documented in [9], for boiling the term accounts for the extra heat needed to bring the liquid to saturation, and vice versa

$$\Delta h = \begin{cases} h_{g,sat} - h_l, & h_l > h_{l,sat} \quad \text{boiling} \\ h_g - h_{l,sat}, & h_l \leq h_{l,sat} \quad \text{condensation} \end{cases} \tag{4.26}$$

The above model for flashing generally performs well except at low void fractions. When α is very small (i.e. $10^{-5}, 10^{-6}$), a tight feedback between the gas generation and temperature equations causes uncontrolled oscillations which can lead to $\alpha < 0$ and a lack of convergence. In reality, hardly any boiling or condensation occurs with $\alpha \lesssim 10^{-5}$, so it makes sense to simply turn off the source term in this case. We turn off the gas generation gradually by reducing A_i. The modified interficial area density is

$$A_i = \frac{6\mathcal{Z}(a_1, a_2, \alpha)}{d_b} \tag{4.27}$$

The function $\mathcal{Z}(a_1, a_2, \alpha)$ is equal to zero for $\alpha < a_1$ (causing $A_i = 0$) and equal to the void fraction for $\alpha > a_2$ (reproducing $A_i = 6\alpha/d_b$). A cubic polynomial interpolates between a_1 and a_2, matching the value and derivative at the endpoints. This is shown schematically in Figure 4.2. For this work, we will use the values $a_1 = 10^{-5}, a_2 = 10^{-4}$.

4.1.4 Physical Properties

Approximate equations of state are used for most calculations (see Section 4.1.1) which are referenced to water at saturation at 5 MPa. The physical properties of water and steam

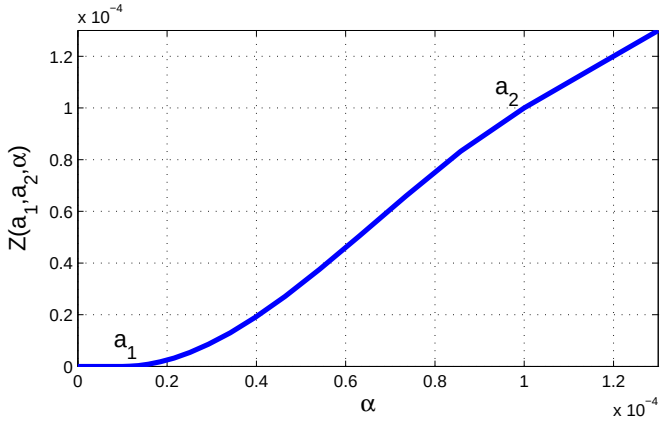


Figure 4.2: Modification for interficial area density to force zero mass transfer for $\alpha < a_1$.

are imported from NIST data [73]. Thermal conductivity, specific heat, and viscosity are kept constant at their reference values since they do not affect the solution significantly. It is important to have an accurate saturation curve, though, so this is imported from NIST data ranging from 0.05 to 12 MPa. The saturation curve should be smooth and differentiable because otherwise it can cause oscillations in the solution. The oscillations can cause the adaptive method to focus on resolving the saturation curve rather than the output of interest, and in the extreme case can prevent convergence. Therefore, a cubic spline is fit to the gas and liquid enthalpies as functions of the pressure, $h_{g,sat}(p), h_{l,sat}(p)$. For this, the Matlab® spline utility is used [83]. The derivatives of the fit are also available ($dh_{g,sat}/dp$ and $dh_{l,sat}/dp$) for calculating analytic derivatives. Using 75 data points for the fit, the maximum relative error for both the gas and liquid enthalpy fits is 0.047%.

4.2 Spatial Discretization

We use the Discontinuous Galerkin Finite Element Method to solve the governing equations because the method lends itself particularly well to the adaptation scheme presented later. The domain Ω_z is first divided into $N_{e,z}$ segments or elements, Ω_k. Each element is the segment between $z = z_{k-1}$ and $z = z_k$, as shown in Figure 4.3. Note, the element sizes need not be uniform. For now the discussion will focus on convective problems with simple

source terms. Section 4.2.1 will describe the treatment of the viscous terms. Sections 4.2.3 and 4.2.8.1 will describe the treatment of the complex sources that depend on $\partial\mathbf{u}/\partial z$ and $\partial\mathbf{u}/\partial t$.

The method assumes a solution which is a polynomial within each element. The solution is allowed to jump from one element to the next, as shown in Figure 4.3. Mathematically, we say that the solution is a member of the function space $\mathcal{V} = \{\mathbf{u}|_{\Omega_k} \in [\mathcal{P}^p(\Omega_k)]^4,\ 1 \leq k \leq N_{e,z}\}$, where $\mathcal{P}^p$ are p^{th} order polynomials. The solution is represented as a set of coefficients of elementary functions which, when linearly combined, yield the desired polynomial. One example of such elementary, or basis, functions is the monomial basis $\{1, z, z^2, z^3, \cdots\}$. A Legendre basis is used here because it results in better conditioning of the resulting discrete equations.

The finite element method finds the solution which best satisfies the governing equations by doing a series of tests. Given a guess of the solution, each test consists of multiplying the governing equations by a test function, say $\mathbf{v}(z)$, and integrating over the domain. This is called the weak form of the governing equation, and the scalar value of this test is called a residual:

$$R = \int_{\Omega_z} \mathbf{v}^T \left[\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{u})}{\partial z} - \mathbf{S}(\mathbf{u}) \right] dz \tag{4.28}$$

The portion in brackets should be zero for the correct solution $\mathbf{u}$, no matter what $\mathbf{v}$ is used (i.e. $R(\mathbf{u}, \mathbf{v}) = 0\ \forall \mathbf{v} \in \mathcal{V}$). In fact, we test with a set of functions $\mathbf{v}_i$ that span the space $\mathcal{V}$ and get a vector of residuals $\mathbf{R} = \{R_i\}$. The solution that best satisfies the governing equations is found by driving the residuals to zero with a root finding algorithm:

$$\mathbf{R}(\mathbf{u}, \mathbf{v}) = \{\mathbf{0}\} \tag{4.29}$$

A unique solution is obtained by choosing as many test functions $\mathbf{v}_i$ as there are degrees of freedom of the solution $\mathbf{u}$. For example, if $\mathbf{u}$ consists of quadratic polynomials ($p = 2$) on $N_{e,z} = 10$ elements with four state variables, there are $(2 + 1) \times 10 \times 4 = 120$ degrees

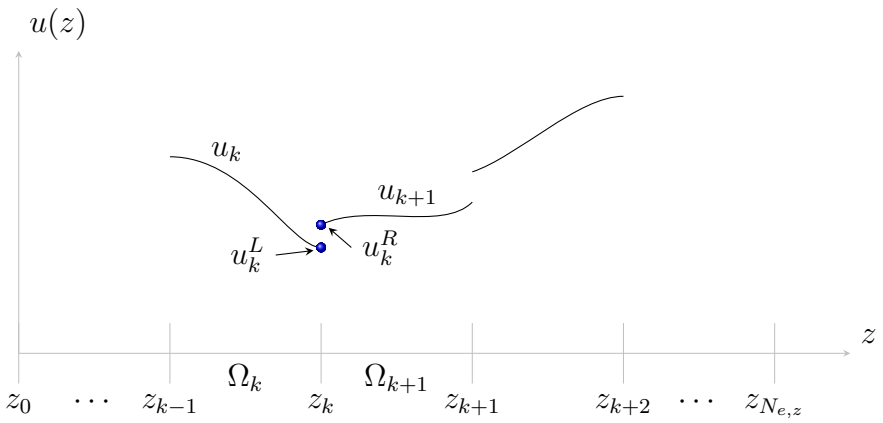


Figure 4.3: The discontinuous finite element approximation is a polynomial within each element but may jump between elements.

of freedom for $\mathbf{u}$, so we need 120 test functions $\mathbf{v}_i$. Since the solution is broken up into separate pieces on each element, we choose test functions that are localized to each element. Mathematically, a Galerkin method is obtained by choosing $\mathbf{v}_i$ to be the same basis that is used to represent the solution $\mathbf{u}$. This is a simple choice which results in well conditioned system.

In the finite element method, the integrals in Eqn. 4.28 are broken up into separate integrals over each element for ease of computation:

$$R_i = \sum_{k=1}^{N_{e,z}} \int_{\Omega_k} \mathbf{v}_i^T \left[\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{u})}{\partial z} - \mathbf{S}(\mathbf{u}) \right] \, dz$$

Then, integration by parts is performed on the spatial flux term, yielding

$$R_i = \sum_{k=1}^{N_{e,z}} \left[\int_{\Omega_k} \mathbf{v}_i^T \frac{\partial \mathbf{u}}{\partial t} \, dz - \int_{\Omega_k} \frac{\partial \mathbf{v}_i}{\partial z}^T \mathbf{F}(\mathbf{u}) \, dz + \mathbf{v}_i^T \hat{\mathbf{F}}|_{z_{k-1}} - \mathbf{v}_i^T \hat{\mathbf{F}}|_{z_k} - \int_{\Omega_k} \mathbf{v}_i^T \mathbf{S}(\mathbf{u}) \, dz \right] \tag{4.30}$$

Almost all of the terms above are straightforward to compute. The test functions $\mathbf{v}_i$ are defined analytically, so their derivatives are known. The source and flux functions can be evaluated simply and integrated using Gaussian integration rules on the elements. What

remains to be defined are the interface fluxes $\hat{\mathbf{F}}$. In general, the interface flux is a function of the two values of the solution at the interface, see Figure 4.3

$$\hat{\mathbf{F}}|_{z_k} = \hat{\mathbf{F}}\left(\mathbf{u}_k^L, \mathbf{u}_k^R\right) \tag{4.31}$$

This flux should be stablized and consistent. Upwind methods are commonly used and give good accuracy. Due to the nonlinear nature of the problem, many choices are possible for stabilized, upwind interface fluxes. In general, these are all approximate solutions to the Riemann problem at the interface; a full solution would require iterations.

4.2.0.1 Riemann Solver Method

One formulation of an upwind flux uses an approximate Riemann solver to define a single set of waves [70]

$$\hat{\mathbf{F}}|_{z_k} = \frac{1}{2}\left[\mathbf{F}(\mathbf{u}_k^L) + \mathbf{F}(\mathbf{u}_k^R)\right] - \frac{1}{2}V|\Lambda|V^{-1}\left[\mathbf{u}_k^R - \mathbf{u}_k^L\right] \tag{4.32}$$

Here, V and Λ make up the eigen-decomposition of the average flux Jacobian $\partial\mathbf{F}/\partial\mathbf{u}$, and $|\Lambda|$ denotes the absolute value of the eigenvalues λ_i.

$$V\Lambda V^{-1} = \mathbf{F}'\left(\frac{\mathbf{u}_k^L + \mathbf{u}_k^R}{2}\right), \qquad \Lambda = \text{diag}(\lambda_i), \qquad |\Lambda| = \text{diag}(|\lambda_i|) \tag{4.33}$$

The stabilized flux can also be thought of as composed of right-moving waves from the left state and left-moving waves from the right state.

4.2.0.2 Double-upwind Method

Another possible definition of an upwind method uses the flux computed from two "middle," upwinded states $\hat{\mathbf{u}}'$ and $\hat{\mathbf{u}}''$:

$$\hat{\mathbf{F}}|_{z_k} = \frac{\mathbf{F}\left(\hat{\mathbf{u}}|^*_{z_k}\right) + \mathbf{F}\left(\hat{\mathbf{u}}|^{**}_{z_k}\right)}{2}, \quad \hat{\mathbf{u}}|^*_{z_k} = \mathbf{u}^L_k + VH(-\Lambda)V^{-1}\left(\mathbf{u}^R_k - \mathbf{u}^L_k\right)$$
$$\hat{\mathbf{u}}|^{**}_{z_k} = \mathbf{u}^R_k + VH(\Lambda)V^{-1}\left(\mathbf{u}^L_k - \mathbf{u}^R_k\right) \tag{4.34}$$

Here, V and Λ make up the eigen-decomposition of the average flux Jacobian. $H(\Lambda)$ is a matrix of ones wherever $\Lambda > 0$ (H is used to denote the Heavyside function). The matrix of ones picks out the right-moving waves.

$$V\Lambda V^{-1} = \mathbf{F}'\left(\frac{\mathbf{u}^L_k + \mathbf{u}^R_k}{2}\right), \qquad \Lambda = \text{diag}(\lambda_i), \qquad H(\Lambda) = \text{diag}(H(\lambda_i)) \tag{4.35}$$

The middle states are composed of right-moving waves from the left side of the interface and left-moving waves from the right side. The two middle states may not identical because the governing equations are non-linear and not homogeneous.

The two methods for the upwind flux are identical for linear problems. Both perform adequately, though in test runs the second was found to result in fewer oscillations. In this work, the double-upwind method is used.

For convective problems, the method is fully defined up to boundary conditions. Correct treatment of boundary conditions can be found in the literature [25] and details on the current implementation are in Section 4.2.4. It should be noted that the eigen-decomposition of $\mathbf{F}'$ can be done quickly because there are only four states and there is a closed form solution for the eigen-decomposition of a 4×4 matrix. In addition, it has been reported [47, 16] that multiphase problems can lose hyperbolicity, meaning that the eigenvalues become imaginary. If this is the case, we switch to the Lax-Friedrichs method, which essentially assumes that

all waves travel at the fastest wave speed

$$\hat{\mathbf{F}}|_{z_k} = \frac{1}{2}\left[\mathbf{F}\left(\mathbf{u}_k^L\right) + \mathbf{F}\left(\mathbf{u}_k^R\right)\right] - \frac{1}{2}|\Lambda|_{\max}\left[\mathbf{u}_k^R - \mathbf{u}_k^L\right] \tag{4.36}$$

where $|\Lambda|_{\max}$ is the eigenvalue matrix where all values are replaced by the maximum absolute eigenvalue. In our experience, however, the Lax-Friedrichs method is rarely, if ever, required.

4.2.1 Extension to Viscous Problems

There are many successful extensions of the Discontinuous Galerkin Finite Element method to viscous problems. We will use the Local Discontinuous Galerkin (LDG) method [25], which has good properties for our adaptation method. In the viscous setting, the fluxes are functions of both the solution and its gradient, so the governing equations are

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial}{\partial z}\mathbf{F}\left(\mathbf{u}, \frac{\partial \mathbf{u}}{\partial z}\right) - \mathbf{S}(\mathbf{u}) = \mathbf{0} \tag{4.37}$$

The LDG method begins by converting this second order system to a larger first order system. The solution gradient is a new unknown $\mathbf{q}$ that will be solved for at the same time as $\mathbf{u}$,

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{u}, \mathbf{q})}{\partial z} - \mathbf{S}(\mathbf{u}) &= \mathbf{0} \\ \mathbf{q} - \frac{\partial \mathbf{u}}{\partial z} &= \mathbf{0} \end{aligned} \tag{4.38}$$

In particular, we assume that $\mathbf{u}, \mathbf{q} \in \mathcal{V}$. Now, we treat this entire first order system in the same manner as before - multiply by test functions, integrate, and perform integration by parts. The test functions for the first equation are denoted $\mathbf{w}_i$, while those for the second are still $\mathbf{v}_i$ (the $\mathbf{w}_i$ are simply copies of $\mathbf{v}_i$, since the solutions to both equations lie in the

same space $\mathcal{V}$). Two residuals result from the equations:

$$R_i^{\mathbf{u}} = \sum_{k=1}^{N_{e,z}} \left[\int_{\Omega_k} \mathbf{v}_i^T \frac{\partial \mathbf{u}}{\partial t}\, dz - \int_{\Omega_k} \frac{\partial \mathbf{v}_i}{\partial z}^T \mathbf{F}(\mathbf{u})\, dz + \mathbf{v}_i^T \hat{\mathbf{F}}|_{z_{k-1}} - \mathbf{v}_i^T \hat{\mathbf{F}}|_{z_k} - \int_{\Omega_k} \mathbf{v}_i^T \mathbf{S}(\mathbf{u})\, dz \right] \tag{4.39}$$

$$R_i^{\mathbf{q}} = \sum_{k=1}^{N_{e,z}} \left[\int_{\Omega_k} \mathbf{w}_i^T \mathbf{q}\, dz + \int_{\Omega_k} \frac{\partial \mathbf{w}_i}{\partial z}^T \mathbf{u}\, dz - \mathbf{w}_i^T \hat{\mathbf{u}}|_{z_{k-1}} + \mathbf{w}_i^T \hat{\mathbf{u}}|_{z_k} \right] \tag{4.40}$$

In this formulation, inviscid fluxes (terms that only depend on $\mathbf{u}$) are computed exactly as for the convective case above. For viscous fluxes, the LDG method defines the interface flux as a function of interface states $\hat{\mathbf{F}} = \mathbf{F}(\hat{\mathbf{u}}, \hat{\mathbf{q}})$. The method computes the interface states $\hat{\mathbf{u}}$ and $\hat{\mathbf{q}}$ as

$$\begin{aligned} \hat{\mathbf{q}}|_{z_k} &= \frac{1}{2}\left(\mathbf{q}_k^R + \mathbf{q}_k^L\right) - C_{11}\left(\mathbf{u}_k^R - \mathbf{u}_k^L\right) + C_{12}\left(\mathbf{q}_k^R - \mathbf{q}_k^L\right) \\ \hat{\mathbf{u}}|_{z_k} &= \frac{1}{2}\left(\mathbf{u}_k^R + \mathbf{u}_k^L\right) - C_{12}\left(\mathbf{u}_k^R - \mathbf{u}_k^L\right) - C_{22}\left(\mathbf{q}_k^R - \mathbf{q}_k^L\right) \end{aligned} \tag{4.41}$$

Here, the constants C_{11}, C_{12} and C_{22} are defined by the LDG method and arise from a specific choice of a 2×2 "C" matrix. Taking $C_{22} = 0$ leads to a convenient method where the solution procedure can be split into two stages: 1) solve Equations 4.40 for $\mathbf{q}(\mathbf{u})$, and 2) solve Equations 4.39 for $\mathbf{u}$. For this 1D problem, we can then simply take $C_{12} = 0.5$ and $C_{11} = 0$ except on the boundary, where $C_{11} = 1/\Delta z$ and Δz is a measure of the element size. For 2D and 3D problems, the choice becomes slightly more complicated, see [25, 92].

Now that all the terms are defined, we can use a root finding algorithm to solve for both unknowns $\mathbf{u}$ and $\mathbf{q}$. In fact, the choice of the LDG method simplifies the problem because the $\hat{\mathbf{u}}$ only depend on $\mathbf{u}$, but not on $\mathbf{q}$. This allows us to symbolically solve the first equation for $\mathbf{q} = \mathbf{q}(\mathbf{u})$ and substitute it into the second equation, so that $\mathbf{u}$ is the only unknown.

4.2.2 Variable Area Formulation

For pipes with variable area $A(z)$, we have modified governing equations in conservative form

$$\frac{\partial(A\mathbf{u})}{\partial t} + \frac{\partial(A\mathbf{F}(\mathbf{u}))}{\partial z} - A\mathbf{S}(\mathbf{u}) = 0 \tag{4.42}$$

Assuming the area does not change in time, we can divide by the area and only the flux term remains affected

$$\frac{\partial \mathbf{u}}{\partial t} + \frac{1}{A}\frac{\partial(A\mathbf{F}(\mathbf{u}))}{\partial z} - \mathbf{S}(\mathbf{u}) = 0 \tag{4.43}$$

Solving this form results in an asymptotically conservative method. Note, however, that the flux Jacobian (and therefore the speed of sound) remains correct. When multiplied by test functions $\mathbf{v}$ and integrated, the modified flux term becomes

$$\begin{aligned}
\int_{\Omega_k} \mathbf{v}_i^T \frac{1}{A}\frac{\partial(A\mathbf{F}(\mathbf{u}))}{\partial z}\, dz &= -\int_{\Omega_k} \frac{\partial(\mathbf{v}_i^T/A)}{\partial z} A\mathbf{F}(\mathbf{u})\, dz + \mathbf{v}_i^T\hat{\mathbf{F}}|_{z_{k-1}} - \mathbf{v}_i^T\hat{\mathbf{F}}|_{z_k} \\
&= -\int_{\Omega_k} \left[\frac{\partial \mathbf{v}_i}{\partial z}^T \mathbf{F}(\mathbf{u}) - \left(\frac{1}{A}\frac{\partial A}{\partial z}\right)\mathbf{v}_i^T\mathbf{F}(\mathbf{u}) \right] dz + \mathbf{v}_i^T\hat{\mathbf{F}}|_{z_{k-1}} - \mathbf{v}_i^T\hat{\mathbf{F}}|_{z_k} \\
&= -\int_{\Omega_k} \frac{\partial \mathbf{v}_i}{\partial z}^T \mathbf{F}(\mathbf{u})\, dz + \underbrace{\int_{\Omega_k} \left(\frac{1}{A}\frac{\partial A}{\partial z}\right)\mathbf{v}_i^T\mathbf{F}(\mathbf{u})\, dz}_{\text{new term}} + \mathbf{v}_i^T\hat{\mathbf{F}}|_{z_{k-1}} - \mathbf{v}_i^T\hat{\mathbf{F}}|_{z_k}
\end{aligned}$$

There is one additional term that arises in the variable area formulation of the momentum equation. The modified momentum equation actually includes one additional term due to the way in which pressure is treated as a "flux" (the actual area-weighted pressure term in conservative form is $A\partial p/\partial z$, not $\partial(Ap)/\partial z$). The extra term, after dividing by the area and

integrating, is added to the residual:

$$\int_{\Omega_k} \left(-\frac{p}{A}\frac{\partial A}{\partial z} \right) v_{i,MOM} F_{MOM}(\mathbf{u})\ dz \tag{4.44}$$

where $v_{i,MOM}$ and F_{MOM} are the test functions and fluxes for the momentum equation.

4.2.3 Auxiliary Variables

When the source terms depend on gradients of the solution, they require special treatment because they act very much like fluxes. We treat these source terms using a "mixed" formulation (details can be found in [90]). The source terms depend on the pressure gradient $\partial p/\partial z$. In the mixed formulation, the pressure gradient is computed in the same way as the solution gradient $\mathbf{q}$. The intermediate approximation to the gradient, p', solves the auxiliary equation

$$p' - \frac{\partial}{\partial z} p(\mathbf{u}) = 0 \tag{4.45}$$

Here, $p(\mathbf{u})$ is computed via the equation of state. The equation is discretized in the same way as the $\mathbf{q}$ equation (see Section 4.2.1) and is solved along with it. The auxiliary variable p' is a member of the same space $\mathcal{V}$ and is multiplied by test functions w_i and integrated:

$$R_i^p = \sum_{k=1}^{N_{e,z}} \left[\int_{\Omega_k} w_i p'\ dz + \int_{\Omega_k} \frac{\partial w_i}{\partial z} p(\mathbf{u})\ dz - w_i \hat{p}|_{z_{k-1}} + w_i \hat{p}|_{z_k} \right] \tag{4.46}$$

The interface quantity $\hat{p}$ is calculated via the natural extension to the LDG method, $\hat{p} = p(\hat{\mathbf{u}})$. Driving the residuals to zero results in the auxiliary variable p'. The residuals are linear and local, so this can be done quickly and in parallel. The gradient p' is then used in the

source terms in the governing equations. Symbolically, we are solving for $p'(\mathbf{u})$ and $\mathbf{q}(\mathbf{u})$ and substituting these into the equations for $\mathbf{u}$.

4.2.4 Boundary Conditions

Generally, boundary conditions are imposed by ghost states and gradients. That is, the boundary is like any interior interface between elements except that the left and right states are replaced by interior and ghost states. The LDG method [25] specifies the boundary quantities

$$\hat{\mathbf{u}} = \mathbf{u}_{\text{ghost}}, \qquad \hat{\mathbf{q}} = \mathbf{q}_{\text{interior}} - C_{11}(\mathbf{u}_{\text{interior}} - \mathbf{u}_{\text{ghost}}) \qquad \text{not supersonic outflow} \tag{4.47}$$

$$\hat{\mathbf{u}} = \mathbf{u}_{\text{interior}}, \qquad \hat{\mathbf{q}} = \mathbf{q}_{\text{ghost}} \qquad \text{supersonic outflow} \tag{4.48}$$

Equation 4.47 is used whenever there are some characteristics entering the domain and $\mathbf{u}_{\text{ghost}}$ specifies the values of those characteristics. The value of $\mathbf{u}_{\text{ghost}}$ depends on the particular type of boundary condition, as described in the following sections. Equation 4.48 is used only for supersonic (or choked) outflow where no information enters the domain. In this case, a value of the gradient must be specified. For this work we take the simplest case $\mathbf{q}_{\text{ghost}} = 0$.

4.2.5 Inflow Void Fraction and Enthalpy

In order to set the void fraction and enthalpy in $\mathbf{u}_{\text{ghost}}$, we first calculate the inflow pressure either from the interior (if $\dot{m}_{\text{spec}}$ is specified) or from a specified inlet pressure. From this known pressure and the boundary values α_{spec} and $h_{m,\text{spec}}$, the equation of state is used to determine the density of the ghost state. This determines three of the components of the ghost state, namely $\rho_m, (\alpha\rho_g)$ and $\rho_m h_m$. The final ghost state component $\rho_m v_m$ is taken as the current guess of $\dot{m}_{\text{in}}$ if the inflow mass flow is specified (see Section 4.2.5.1) or taken from the interior if the inflow pressure is specified.

4.2.5.1 Inflow Mass Flow

For a subsonic inflow boundary condition three independent quantities need to be specified for each of the three incoming characteristic waves. The void fraction and enthalpy are always specified. The third quantity can be either the mass flow or the pressure. We will focus on a specified mass flow, but the same ideas can be used for the pressure. The boundary condition is enforced via a modified ghost state method. We extend the entire system to include as an extra unknown the inflow mass flow ($\dot{m}_{\text{in}}$). For the system to be solvable, an extra residual equation is added

$$R^{\text{bc}} = \dot{m}_{\text{in}} - \dot{m}_{\text{spec}} \tag{4.49}$$

here, $\dot{m}_{\text{spec}}$ is the user-specified mass flow. When R^{bc} is driven to zero (with all of the other residuals), the specified mass flow will be achieved. For a similar iterative procedure for enforcing mass flow, see [31]. The ghost state is then computed from the specified void fraction and enthalpy; the current guess of the mass flow; and the interior state (for the characteristic wave coming from the interior)

$$\mathbf{u}_{\text{ghost}} = \mathbf{u}_{\text{ghost}}(\alpha_{\text{spec}}, h_{\text{spec}}, \dot{m}_{\text{in}}, \mathbf{u}_{\text{interior}}) \tag{4.50}$$

The reason we do not use the specified mass flow directly in $\mathbf{u}_{\text{ghost}}$ is that it can cause the system to be under-specified when the flow is choked[3]. The ghost state is used as a Dirichlet boundary condition in the standard LDG method [92].

To simplify the solution of the extended system, the Schur complement decomposition is used to solve for $\dot{m}_{\text{in}}$ and substitute it back [29, 42]. For a single Newton step, we are interested in computing the update $\delta\mathbf{u} = -(\partial\mathbf{R}/\partial\mathbf{u})^{-1}\mathbf{R}(\mathbf{u})$. If we separate out the residual

[3]When the flow is choked, the mass flow is *always* constant, so it cannot be used to set inflow conditions. Instead an inflow pressure must be used. Indeed, if the inflow pressure is specified, we can just use this directly in $\mathbf{u}_{\text{ghost}}$ without adding an extra unknown to the system

and unknown due to the boundary condition, the system to solve, in block form, is

$$\begin{bmatrix} \frac{\partial \mathbf{R}}{\partial \mathbf{u}} & \frac{\partial \mathbf{R}}{\partial \dot{m}_{\text{in}}} \\ \frac{\partial R^{\text{bc}}}{\partial \mathbf{u}} & \frac{\partial R^{\text{bc}}}{\partial \dot{m}_{\text{in}}} \end{bmatrix} \begin{bmatrix} \delta \mathbf{u} \\ \delta \dot{m}_{\text{in}} \end{bmatrix} = - \begin{bmatrix} \mathbf{R} \\ R^{\text{bc}} \end{bmatrix} \tag{4.51}$$

Now we can formally solve for the boundary state

$$\delta \dot{m}_{\text{in}} = - \frac{\partial R^{\text{bc}}}{\partial \dot{m}_{\text{in}}}^{-1} \left(R^{\text{bc}} + \frac{\partial R^{\text{bc}}}{\partial \mathbf{u}} \delta \mathbf{u} \right) \tag{4.52}$$

This is substituted back to get a matrix equation for the interior states only

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{u}} - \frac{\partial \mathbf{R}}{\partial \dot{m}_{\text{in}}} \left(\frac{\partial R^{\text{bc}}}{\partial \dot{m}_{\text{in}}} \right)^{-1} \frac{\partial R^{\text{bc}}}{\partial \mathbf{u}} \right] \delta \mathbf{u} = - \left[\mathbf{R} - \frac{\partial \mathbf{R}}{\partial \dot{m}_{\text{in}}} \frac{\partial R^{\text{bc}}}{\partial \dot{m}_{\text{in}}}^{-1} R^{\text{bc}} \right] \tag{4.53}$$

Forming this equation requires only matrix-vector multiplications and inversion of the a scalar. From Equation 4.49, we have $\partial R^{\text{bc}} / \partial \dot{m}_{\text{in}} = 1$ and $\partial R^{\text{bc}} / \partial \mathbf{u} = 0$, giving

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \right] \delta \mathbf{u} = - \left[\mathbf{R} - \frac{\partial \mathbf{R}}{\partial \dot{m}_{\text{in}}} R^{\text{bc}} \right] \tag{4.54}$$

Thus, the residual Jacobian does not change and all that is needed is a simple modification of the residual vector.

4.2.5.2 Pressure Outflow

A fixed pressure (subsonic) outflow boundary condition is set with the ghost cell method. A ghost state is determined from the specified pressure (for the one characteristic wave entering the domain) and the interior state

$$\mathbf{u}_{\text{ghost}} = \mathbf{u}_{\text{ghost}}(p_{\text{spec}}, \mathbf{u}_{\text{interior}}) \tag{4.55}$$

Normally the ghost state would be set by examining the eigen-decomposition of the state to separate the characteristic waves. For the multiphase equations with a complex drift-velocity correlation, this decomposition is not known. Instead, the ghost state can be set iteratively. Three of the ghost states are set to the interior and one is adjusted to match the specified pressure. The pressure is only a function of the first two states ρ_m and $\alpha\rho_g$. In this work, we find the mean density ρ_m^* that enforces

$$R^{\text{pout}}(\rho_m^*) = p_{\text{spec}} - p(\rho_m^*, \mathbf{u}_{\text{interior}}) = 0 \tag{4.56}$$

This is done using Newton's method, with under-relaxation to ensure positive pressure[4]. The ghost state is then computed as

$$\mathbf{u}_{\text{ghost}} = \mathbf{u}_{\text{ghost}}(\rho_m^*, \mathbf{u}_{\text{interior}}) \tag{4.57}$$

That is, the first component of $\mathbf{u}_{\text{ghost}}$, the average density, is taken as ρ_m^*, while the other three components are taken from the interior state.

4.2.5.3 Choked, Supersonic Outflow

A choked outflow boundary condition is simple to set when the outflow is supersonic, since all information comes from the interior. The ghost state is simply

$$\mathbf{u}_{\text{ghost}} = \mathbf{u}_{\text{interior}} \tag{4.58}$$

One difficulty with choked flow is detecting choking. One way to detect choking is to compute the wave speeds (the eigenvalues of $\partial\mathbf{F}/\partial\mathbf{u}$) throughout the domain. The wave speeds

[4]Here, $p(\rho_m^*, \mathbf{u}_{\text{interior}})$ denotes the pressure computed by upwinding the ghost state (i.e. the current guess of it) and interior state. "Upwinding" means computing a "middle" state that takes the right-moving waves from the state to the left of the boundary and vice-versa. Investigations showed that linearizing the flux about the external state for the upwinding produced the least oscillations in the adjoint, though other linearizations still produced convergent methods. When applying the boundary conditions, the interface flux $\mathbf{F}(\mathbf{u}_{\text{interior}}, \mathbf{u}_{\text{ghost}})$ is also computed by linearizing about the external state for consistency.

are approximately of the form $v-a, v, v, v+a$ where v is the local velocity and a is the local speed of sound. This is only approximate because the drift velocity model affects the eigenvalues [40, 17]. The flow is choked if the wavespeeds are all positive at any point. In practice, the bulk speed of sound is approximated instead by

$$a_{\text{sound}} = \sqrt{\frac{\partial p}{\partial \rho_m}} \tag{4.59}$$

which is inspired by the single-phase relationship. This formula for the sound speed was found to be much more reliable and stable than using the eigenvalues of the flux Jacobian[5]. The flow is considered choked whenever $v_m \geq a_{\text{sound}}$, which is checked at all of the integration points.

4.2.6 Sonic Outflow

Although rare, there are cases where a precisely sonic condition should be set at the outflow. Sonic outflow is detected when the velocity is close to the speed of sound, specifically when

$$\left| \frac{v_m - a_{\text{sound}}}{v_m} \right| \leq \tau_{\text{sonic}} \tag{4.60}$$

The sonic condition is enforced by an iterative procedure for calculating $\mathbf{u}_{\text{ghost}}$, similar to the procedure for the outflow pressure. For a sonic condition, the residual to be satisfied is

$$R^{\text{sonic}} = a_{\text{sound}}(\rho_m^*, \mathbf{u}_{\text{interior}}) - v_m(\mathbf{u}_{\text{interior}}) \tag{4.61}$$

[5]The minimum eigenvalue in practice never reaches zero even when the flow is sonic or supersonic. The minimum eigenvalue peaks at around -50, which is small compared to the liquid speed of sound (e.g. 1088 m/s at the reference condition). One must choose a tolerance for the minimum eigenvalue, and this tolerance was found to have a strong effect on the solution.

The density ρ_m^* which satisfies $R^{\text{sonic}} = 0$ is used in the ghost state

$$\mathbf{u}_{\text{ghost}} = \mathbf{u}_{\text{ghost}}(\rho_m^*, \mathbf{u}_{\text{interior}}) \tag{4.62}$$

A small value of the tolerance τ_{sonic} results in the sonic condition being applied sparingly, which can lead to rapid switching between sub- and super-sonic conditions. This can occur because of the discontinuous nature of the approximation in time. The switching results in large oscillations in the adjoint and hence degrades the adaptation metrics. A large value of τ_{sonic} results in the sonic condition being applied for a possibly large time interval. While this may not be physically accurate, it results in a smoother adjoint and better adaptation metrics. Ideally, the value of τ_{sonic} would depend on the range of velocities in the solution and the grid size. In this work, a value of $\tau_{\text{sonic}} = 0.05$ was found to give good results for all grids.

4.2.6.1 Reflecting Wall

A solid wall boundary condition can be enforced by using the mass flow condition above and setting $\dot{m}_{\text{spec}} = 0$. Another possibility is to set the boundary fluxes directly, which avoids adding the extra unknown to the system. At a reflecting wall, all fluxes are zero except for the momentum flux, which is equal to the local pressure.

4.2.6.2 Gradient and Auxiliary Variable Boundary Conditions

The initial solve for $\mathbf{q}(\mathbf{u})$ also requires boundary conditions which are specified by the LDG method (see Equations 4.47 and 4.48). The auxiliary variable solve also requires boundary conditions. It was found that using just the interior state is sufficient for good convergence.

4.2.7 Physicality Penalization

Numerical multiphase flow problems can suffer from oscillations that cause the solution to be unphysical and degrade or prohibit convergence. This can occur when the initial guess is far from the steady state solution. Oscillations can also occur if the grid is too coarse. One solution is to guide the Newton iterations away from unphysical regions of the state space. This can done by including a physicality penalty in the residuals [20]. In the two-phase flow context, a number of constraints could be imposed. The following four perform well in the current method

$$\alpha \leq 1, \quad \alpha \geq 0, \quad p > 0, \quad \rho_m > 0 \tag{4.63}$$

These are normalized and converted to the form $c_i(u) > 0$, where c_i is a constraint

$$1 - \alpha > 0, \quad \alpha > 0, \quad \frac{p}{p_{ref}} > 0, \quad \frac{\rho_m}{\alpha\rho_g + (1-\alpha)\rho_l} > 0 \tag{4.64}$$

The final penalty term on an element is

$$P_{k,phys} = \mu_{pen} \sum_{i=1}^{4} \sum_{j=1}^{n_g} \frac{1}{c_i(\mathbf{u}(z_j))} \tag{4.65}$$

here, j indexes the n_g integration points inside an element, z_j. The residuals in element k are scaled by the penalty term $P_{k,phys}$. For more details on the method, see [20].

4.2.8 Transient Simulations

Time-accurate transient simulations can be computed using a variety of time stepping methods. The basis for these is the coupled set of ordinary differential equations which arise from the finite element discretization in space. We begin by restating the test function

weighted governing equations for a transient problem

$$\int \mathbf{v}(z) \left[\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{u})}{\partial z} - \mathbf{S}(\mathbf{u}) \right] \, dz = 0 \tag{4.66}$$

Here, the $\mathbf{v}$ are spatial basis functions. For a steady state computation, we assume that $\partial \mathbf{u}/\partial t = \mathbf{0}$ and set the resulting residuals to zero. For a test function $\mathbf{v}_i$, the corresponding residual is

$$R_i^{ss}(\mathbf{u}, \mathbf{v}_i) = \int \mathbf{v}_i(z) \left[\frac{\partial \mathbf{F}(\mathbf{u})}{\partial z} - \mathbf{S}(\mathbf{u}) \right] \, dz = 0 \tag{4.67}$$

For a transient computation, we separate the time and space dependence:

$$\int \mathbf{v}_i(z) \frac{\partial \mathbf{u}}{\partial t} \, dz + \int \mathbf{v}_i(z) \left[\frac{\partial \mathbf{F}(\mathbf{u})}{\partial z} - \mathbf{S}(\mathbf{u}) \right] \, dz = \int \mathbf{v}_i(z) \frac{\partial \mathbf{u}}{\partial t} \, dz + R_i^{ss}(\mathbf{u}, \mathbf{v}_i) = 0 \tag{4.68}$$

We split the solution $\mathbf{u}(z,t)$ into time and space parts in a tensor product fashion,

$$\mathbf{u}(z,t) = \sum_{m,n} g_{m,n} \mathbf{v}_m(z) \boldsymbol{\nu}_n(t) \tag{4.69}$$

where $\mathbf{v}_m$ are spatial basis function and $\boldsymbol{\nu}_n$ are temporal basis functions. This splitting simplifies the first term in Equation 4.68 to

$$\int \mathbf{v}_i(z) \frac{\partial \mathbf{u}}{\partial t} \, dz = \sum_{m,n} g_{m,n} \frac{d\boldsymbol{\nu}_n(t)}{dt} \int \mathbf{v}_i(z) \mathbf{v}_m(z) \, dz \tag{4.70}$$

Next, we apply the LDG method to the time domain. The time span of interest is broken up into elements Ω_l bounded by t_{l-1} and t_l. Temporal basis functions, $\boldsymbol{\nu}_j(t)$ are essentially the same as for the spatial discretization. The governing equations are multiplied by both spatial and temporal basis functions and integrated. This yields the transient residuals $R_{i,j}^{tr}$

for spatial test function $\mathbf{v}_i$ and temporal test function $\boldsymbol{\nu}_j$

$$\begin{aligned} R^{tr}_{i,j} &= \int_{\Omega_t}\int_{\Omega_z} \mathbf{v}_i \boldsymbol{\nu}_j \left[\frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{u})}{\partial z} - \mathbf{S}(\mathbf{u})\right] \, dz \, dt \\ &= \int_{\Omega_t} \boldsymbol{\nu}_j(t) \sum_{m,n} g_{m,n} \frac{d\boldsymbol{\nu}_n(t)}{dt} \int_{\Omega_z} \mathbf{v}_i(z)\mathbf{v}_m(z) \, dz \, dt + \int_{\Omega_t} \boldsymbol{\nu}_j(t) R^{ss}_i \, dt \end{aligned} \tag{4.71}$$

The notation is simplified by taking $\sum_{m,n} g_{m,n}\boldsymbol{\nu}_n(t) \int_{\Omega_k} \mathbf{v}_i(z)\mathbf{v}_m(z) \, dz = \mathbf{M}_{z,i,:}\mathbf{g}(t)$. The vector $\mathbf{M}_{z,i,:}$ essentially the i^{th} row of the spatial mass matrix[6] and $\mathbf{g}(t) = \{\sum_{m,n} g_{m,n}\boldsymbol{\nu}_n(t)\}$ can be interpreted as a time-dependent vector of coefficients of the spatial basis functions $\mathbf{v}_m$. Integrating by parts in time and restricting to the space element Ω_k and time element Ω_l yields

$$\begin{aligned} R^{tr}_{i,j} = \sum_{k,l} \Bigg[&- \int_{\Omega_l} \frac{d\boldsymbol{\nu}_j(t)}{dt} \mathbf{M}_{z,i,:}\mathbf{g}(t) \, dt + \int_{\Omega_l} \boldsymbol{\nu}_j(t) R^{ss}_i \, dt \\ &+ \boldsymbol{\nu}_j \left[\mathbf{M}_{z,i,:}\mathbf{g}(t)\right]_{t_{l-1}} - \boldsymbol{\nu}_j \left[\mathbf{M}_{z,i,:}\mathbf{g}(t)\right]_{t_l} \Bigg] \end{aligned} \tag{4.72}$$

The quantities evaluated at the temporal interfaces t_{l-1} and t_l can be simply upwinded from the previous time point because the simulation must be deterministic

$$\left[\mathbf{M}_{z,i,:}\mathbf{g}(t)\right]_{t_l} = \mathbf{M}_{z,i,:}\mathbf{g}^L(t_l) \tag{4.73}$$

This fully specifies the time integration scheme. The full set of space-time residuals, $\mathbf{R}^{tr} = \{R^{tr}_{i,j}\}$, is driven to zero via Newton's method

$$\mathbf{R}^{tr} = \mathbf{0} \tag{4.74}$$

Due to the pure upwinding in the time axis, it is possible to drive the residuals to zero in a

[6] The 1D spatial mass matrix is $\mathbf{M}_z$ with elements $\mathbf{M}_{z,i,j} = \int \mathbf{v}_i\mathbf{v}_j dz$. The version used here is replicated for each temporal point.

sequential manner starting with the first time element. This may have to be done in order to keep memory requirements low. For smaller problems, we can solve for the entire time span at once.

An effective way to initialize the solution is with a time stepping procedure based on the discontinuous Galerkin discretization. Since each time element only depends on the solution at the end of the previous time element, the whole space-time solution can be built up one time element at a time. Converging each time element requires Newton iterations. The initial guess for the solution in an element is just the value of the solution at the end of the previous element (i.e. assume a constant in the current element). The advantage of this procedure is that the time steps can be adaptively increased or decreased. This generates a space-time solution that is convergent; getting an *accurate* solution requires further adjoint-based adaptation of the spatial and temporal grids.

The time step Δt is adaptively increased or decreased based on the convergence of the Newton iterations within the time step. If n_{dtgrowth} time steps converge successfully[7], the time step is increased ($\Delta t \leftarrow f_{\text{dtgrowth}} \Delta t$). If the Newton iterations fail to converge, the time step is decreased ($\Delta t \leftarrow f_{\text{dtshrink}} \Delta t$) and the Newton iterations are restarted. For this work, it suffices to take $f_{\text{dtgrowth}} = 5, n_{\text{dtgrowth}} = 5, f_{\text{dtshrink}} = 0.2$. After initializing with the time stepping procedure, the full space-time solution is checked for residual tolerance and the adjoint can be calculated.

4.2.8.1 Auxiliary Variables in Time

The governing equations include a source term in the enthalpy equation, Eqn 4.4, which depends on $\partial p/\partial t$. This term must be calculated in a similar way to how the $\partial p/\partial z$ term is calculated (see Section 4.2.3) in order to preserve adjoint consistency. In this case, the

[7] Successful convergence means that either residuals reached small values or the iterations stalled out, see Section 4.2.9

auxiliary variable $\tilde{p}'$ solves the equation

$$\tilde{p}' - \frac{\partial}{\partial t} p(\mathbf{u}) = 0 \tag{4.75}$$

The equation is multiplied by test functions and integrated in time and space to yield the residual

$$R^{\tilde{p}}_{i,j} = \sum_{k,l} \left[\int_{\Omega_l} \int_{\Omega_k} w_i \omega_j \tilde{p}' \; dz \; dt + \int_{\Omega_l} \int_{\Omega_k} w_i \frac{\partial \omega_j}{\partial t} p(\mathbf{u}) \; dz \; dt - \int_{\Omega_k} w_i \omega_j \hat{p}|_{t_{l-1}} dz + \int_{\Omega_k} w_i \omega_j \hat{p}|_{t_l} dz \right] \tag{4.76}$$

Here $w_i(z)$ is a spatial basis function and $\omega_j(t)$ is a temporal basis function. The interface quantity $\hat{p}$ is computed as $p(\hat{\mathbf{u}})$. In this case, the value of $\hat{\mathbf{u}}$ on the borders of the time elements is simply the value at the previous time. The gradient $\tilde{p}$ is computed by driving the residuals to zero. The residuals are linear and local, so this can be done quickly and in parallel. The solution $\tilde{p}$ is then used in the source term in the energy equation.

4.2.9 Numerical Root-Finding with Newton-Raphson

A few details about the Newton iterations are as follows. For a residual $\mathbf{R}(\mathbf{u}) = 0$, the k^{th} Newton step is defined as [7]

$$\mathbf{u}^{k+1} = \mathbf{u}^k - \omega \left(\frac{\partial \mathbf{R}^k}{\partial \mathbf{u}^k} \right)^{-1} \mathbf{R}^k(\mathbf{u}^k) \tag{4.77}$$

A full update corresponds to $\omega = 1$, and a relaxed update has $\omega < 1$. The residual is measured by $|\mathbf{R}|$

$$|\mathbf{R}| = \max_i \frac{||\mathbf{R}_i(\mathbf{u}_i)||_2}{||\mathbf{u}_i||_2} \tag{4.78}$$

where i indexes over the equations and states. For example, $i = 1$ corresponds to the conservation of mass equation and state ρ_m. The measure $|\mathbf{R}|$ compensates for the different

scales in each equation. Newton iterations start with $\mathbf{u}^0, \mathbf{R}^0$ and convergence is found if $|\mathbf{R}|/|\mathbf{R}^0| < \tau_{newtonrel}$ or $|\mathbf{R}| < \tau_{newtonabs}$. Usually the residuals will stall out before reaching machine precision because of the wide range of scales[8]. If the residual stays almost constant for a few iterations, stalling is detected and the iterations are stopped. Specifically, stalling occurs if $\max(\mathbf{R}^k) - \min(\mathbf{R}^k) < \tau_{newtonstall}$ for k ranging over the last $n_{newtonstall}$ iterations. In addition, the Newton relaxation factor ω is be reduced by $d\omega$ during iterations if $|\mathbf{R}|/|\mathbf{R}^0| > \tau_{newtonrelax}$ or increased if $|\mathbf{R}^k|/|\mathbf{R}^{k-1}| < \tau_{newtonunrelax}$. The total number of Newton steps is capped at $n_{newtonmax}$. Finally, failure is reported if the residual grows too much, $|\mathbf{R}|/|\mathbf{R}^0| > \tau_{newtonbig}$. In this work, values for the various tolerances and criteria were chosen by trial and error. They are: $\tau_{newtonrel} = 10^{-8}, \tau_{newtonabs} = 10^{-6}, d\omega = 0.1, \tau_{newtonstall} = 10^2, n_{newtonstall} = 6, \tau_{newtonrelax} = 5.0, \tau_{newtonunrelax} = 10^{-2}, n_{newtonmax} = 16, \tau_{newtonbig} = 10^3$.

4.3 Adjoint Discretization

The term adjoint as used here is essentially the same as that used in the neutronics community for perturbation analysis. There, the adjoint is usually derived in the continuous framework. This leads to adjoint operators that complement "regular" operators (e.g. the adjoint of "grad" is "-grad"). Sensitivities are then computed by discretizing the adjoint operators. The same analysis can be performed on the governing equations of multiphase fluid flow. However, another approach is to derive the adjoint operators after the equations have been discretized. Suppose the (linear) governing equations $\mathcal{L}u = f$ are discretized, yielding the matrix equation $\mathbf{L}_h\mathbf{u}_h = \mathbf{f}_h$. Then, the discrete adjoint operator can be shown to be $\mathbf{L}_h^\dagger = \mathbf{L}_h^T$, simply the transpose of the original matrix. If the discretization is an adjoint consistent one (the LDG method can be shown to be, subject to appropriate boundary conditions, see [6]), then $\mathbf{L}_h^T$ is a good (consistent and stable) discretization of the continuous adjoint operator $\mathcal{L}^\dagger$. The derivation is slightly more involved for a nonlinear discrete system, and is presented in Section 1.2.3. A sketch of the derivation for a continuous system is used

[8]The equations are not normalized for ease of implementing the empirical models

in 3.4.2 to motivate the stochastic error estimate.

The canonical discontinuous Galerkin method is adjoint consistent, so the simple discrete approach can be used to solve for the adjoint. That is, for a given output $K(\mathbf{u})$, the adjoint satisfies

$$\frac{\partial \mathbf{R}}{\partial \mathbf{u}}^T \boldsymbol{\psi} = -\frac{\partial K}{\partial \mathbf{u}}^T \tag{4.79}$$

Note, variations on the above development, for example different treatment of the pressure gradient or boundary conditions, can lead to an adjoint *in*consistent method. In fact, the method for computing the pressure gradient is only asymptotically adjoint consistent [90]. The solution of the adjoint equations results in an adjoint with four components, one associated with each equation being solved. These are the mixture-mass component, gas-mass component, mixture-momentum component, and the mixture-enthalpy component. Each component represents the sensitivity of the output to errors in satisfying the associated equation (i.e. residuals of that equation). Finally, note that in the ideal case, the adjoint solution converges to the true solution to the continuous adjoint equations at the same rate as $\mathbf{u}$, in the case of DG FEM the error decays as $O(\Delta z^{p+1})$.

4.4 Error Estimation

The adjoint can be used in a special way to estimate discretization errors, or errors associated with the grid resolution. Essentially, the adjoint computes the sensitivity to the "parameter" of grid resolution. When the sensitivity is sufficiently small, we can say that the quantity of interest has converged and perhaps no further grid refinement is necessary. In order to do this, we compute the difference between the output on the current grid and on a grid that is uniformly refined, called δK_H^{unif}. Information on the "fine" grid (denoted

by h) can uncover inaccuracies in the original or "coarse" grid solution (denoted by H).

$$\begin{aligned} \mathbf{u}_H \text{ satisfies} \quad & \mathbf{R}_H(\mathbf{u}_H) = 0 \\ \boldsymbol{\psi}_H \text{ satisfies} \quad & \frac{\partial \mathbf{R}_H}{\partial \mathbf{u}_H}^T \boldsymbol{\psi}_H = -\frac{\partial K_H}{\partial \mathbf{u}_H}^T \\ \text{error due to resolution:} \quad & K_H(\mathbf{u}_H) - K_h(\mathbf{u}_h) \approx \delta K_H^{\text{unif}} = \boldsymbol{\psi}_h^T \mathbf{R}_h(\mathcal{I}_h^H \mathbf{u}_H) \end{aligned} \tag{4.80}$$

Here, we are looking at the error of the coarse solution ($\mathbf{u}_H$) as seen by the fine grid ($\boldsymbol{\psi}_h$ and $\mathbf{R}_h$). The injection operator $\mathcal{I}_h^H$ simply generates a representation of the coarse solution $\mathbf{u}_H$ on the fine grid (this can sometimes be done by sampling or in general by projection). It is straightforward to compute the second term $\mathbf{R}_h(\mathcal{I}_h^H \mathbf{u}_H)$ by setting up the refined grid and taking the solution $\mathbf{u}_H$ as, e.g., a first guess of the fine solution.

Eqn. 4.80, though, also requires the adjoint solution on the fine grid, $\boldsymbol{\psi}_h$. In general, we would have to solve for the full fine grid solution $\mathbf{u}_h$ in order to then solve for $\boldsymbol{\psi}_h$, making this option potentially quite time consuming. However, it can be shown that it is often sufficient to simply interpolate and smooth the coarse adjoint $\boldsymbol{\psi}_H$ as an approximation of $\boldsymbol{\psi}_h$. This assumes that the adjoint is smooth and converges to the continuous adjoint as the grid is refined. The discontinuous Galerkin method used here is asymptotically adjoint consistent, so the rate of convergence of the adjoint to the continuous adjoint approaches $p+1$ (assuming space and time are discretized to order p). When the adjoint has sharp changes, the interpolation results in a less accurate error estimate. Still, for the current problem interpolation was found to have sufficient accuracy for driving the adaptive process[9]. For the purposes of obtaining an highly accurate estimate of discretization error, the full fine adjoint would be more appropriate.

In multiple dimensions, interpolating the coarse adjoint can be done with various reconstruction or interpolation methods. In this work, a patch reconstruction method is used [95]. The solution in element k and its two neighbors is projected to the higher-order space

[9]Some inaccuracy in the error estimate is tolerated when using it to drive adaptation because, in the end, the information is used to decided which elements to be refined.

$\mathcal{P}^{p+1}(\Omega_k)$. The higher-order solution is then projected via least squares onto the mesh of interest[10].

4.5 Adaptation

If grid convergence is defined by the user to be an absolute error tolerance on some part of the solution (or quantity of interest, K_h), then the adjoint-based error estimation technique can be used to reliably measure grid convergence. By the same token, the grid can be dynamically refined (adapted) such that the quantity K_h is quickly computed to high accuracy by targeting regions that contribute most to the error. Notice that the error estimate is an inner product between two field quantities, $\boldsymbol{\psi}_h$ and $\mathbf{R}_h$. The value of $\boldsymbol{\psi}_h^T \mathbf{R}_h$ in each element can be interpreted as how much error that element contributes to the overall error:

$$\epsilon_k = \sum_{i \in k} \psi_{h,i} R_{h,i}(\mathcal{I}_h^H \mathbf{u}_H) \quad \text{or} \quad \sum_{i \in k} |\psi_{h,i} R_{h,i}(\mathcal{I}_h^H \mathbf{u}_H)| \tag{4.81}$$

Here, the notation $i \in k$ picks out the fine-grid elements i that lie within the coarse grid element k. The second expression is a more conservative one in which error cancellation does not occur. Here, the sum is over all components relevant to an element.

Next we must choose a strategy for adapting the elements. One of the simplest strategies is to uniformly refine a fixed percentage of the elements (those with the highest error) at each adaptation step. That is, the elements are sorted by ϵ_k and the top γ_{num} percent are flagged for refinement. The strategy works well when the error is moderately concentrated in one part of the domain.

When errors are highly concentrated in, e.g., one element, then the above strategy is slow because excess refinement will occur outside of that one "bad" element. There are two ways to remedy this problem. One strategy is multi-level refinement, where some elements

[10] At cell interfaces, the projected solutions from both sides are averaged.

are flagged for multiple uniform refinements (all in a single adaptation step). This requires another heuristic to decide the level of refinement. A second strategy is to reduce the number of flagged elements to those that contain γ_{err} percent of the total error. That is, we refine either a fixed fraction of the number of elements, $\gamma_{num} \times N_{e,z}$, or a fixed fraction of the total error, $\gamma_{err} \times \delta K_H$, whichever is smaller. This strategy also results in single "bad" elements being refined many times (over many adaptation steps), but requires evaluating the error estimate more often. In the results below we use the second strategy.

4.5.1 Anisotropic Adaptation

Grid adaptation is most beneficial when small regions of the domain require a large number of elements to resolve the solution due to complex behavior. In multiple dimensions, one must consider element size in each dimension. An element's aspect ratio is a measure of how the element's size varies with direction[11]. If the region is near a single point, then isotropic refinement, where element aspect ratios are kept approximately constant, is good at capturing the complex behavior. However, if the region is, say, a line (e.g. a boundary layer), then isotropic refinement will lead to a sub-optimal grid. In the case of a line, the solution has complex behavior perpendicular to the line, but not along it. An efficient grid therefore has long, thin elements along the line. This is called anisotropic refinement because different amounts of refinement are needed in different directions [68].

In the context of space-time (and a tensor-product grid), anisotropic adaptation requires a decision to be made between spatial and temporal refinement. The same basic method can be used to decide between refinement in the physical and stochastic domains. Indeed, our approximation to $\mathbf{u}$ can be thought of as a tensor product between the stochastic and physical spaces.

We use the anisotropic adaptation scheme presented in [39]. The output error is computed on semi-refined spaces and compared. The regular isotropic error metric and the total error

[11] One way to define the aspect ratio of an element is the ratio of the largest to smallest principle axes of the smallest enclosing ellipsoid of the element.

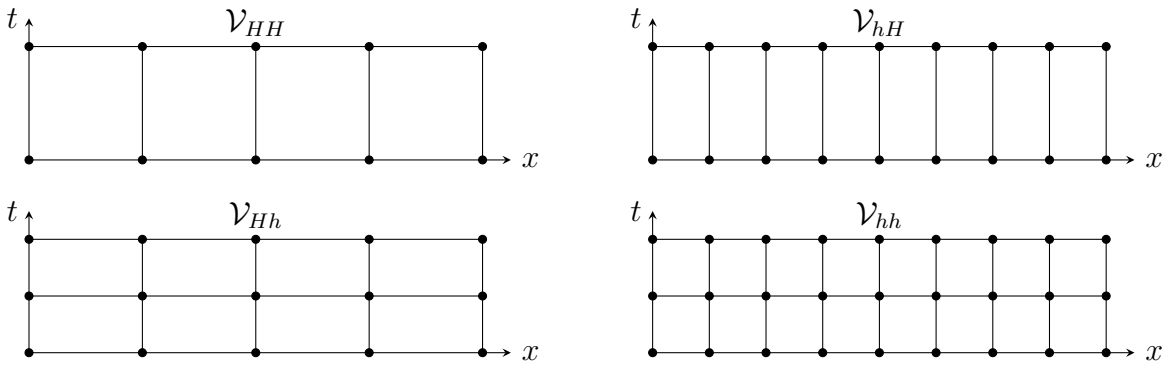


Figure 4.4: Diagram of original coarse space $\mathcal{V}_{HH}$ (upper left), full fine space $\mathcal{V}_{hh}$ (lower right), and semi-refined spaces in space $\mathcal{V}_{hH}$ (upper right) and semi-refined in time $\mathcal{V}_{Hh}$ (lower left).

are

$$\epsilon_{hh}^{kl} = \sum_{i \in (k,l)} \boldsymbol{\psi}_{hh,i} \mathbf{R}_{hh,i} (\mathcal{I}_{HH}^{hh} \mathbf{u}), \qquad \delta K_{HH} = \sum_{k} \sum_{l} \epsilon_{hh}^{kl} \tag{4.82}$$

where $(\cdot)_{hh}$ indicates a quantity evaluated on a grid refined both in time and space. The notation $i \in (k,l)$ indicates the components of $\boldsymbol{\psi}$ and $\mathbf{R}$ relevant to the spatial element k and temporal element l. The semi-refined spaces are denoted $\mathcal{V}_{hH}$ (spatial refinement only) and $\mathcal{V}_{Hh}$ (temporal refinement only). A diagram of the various spaces is shown in Figure 4.4. The error estimates on these spaces are

$$\epsilon_{hH}^{kl} = \sum_{i \in (k,l)} \boldsymbol{\psi}_{hH,i} \mathbf{R}_{hH,i} (\mathcal{I}_{HH}^{hH} \mathbf{u}) \qquad \epsilon_{Hh}^{kl} = \sum_{i \in (k,l)} \boldsymbol{\psi}_{Hh,i} \mathbf{R}_{Hh,i} (\mathcal{I}_{HH}^{Hh} \mathbf{u}) \tag{4.83}$$

The adjoint is easily interpolated onto either of these spaces using one-dimensional patch-reconstruction. Next, fractions of error are constructed by comparing the semi-refined error estimates. The grid is a tensor-product between space and time, so refinement can occur in space (a spatial element is refined for all time) or in time (a temporal refinement is the same for all spatial elements). The error fractions are normalized to remove any scaling differences and the result is a measure of anisotropy in the solution as described in Table 4.1. The

Table 4.1: Anisotropy measures for adaptation.

	Error due to spatial resolution	Error due to temporal resolution
In space element k :	$f^k_{\text{space}} = \frac{\sum_l \epsilon^{kl}_{hH}}{\sum_l \left(\epsilon^{kl}_{hH}+\epsilon^{kl}_{Hh}\right)}$	$f^k_{\text{time}} = \frac{\sum_l \epsilon^{kl}_{Hh}}{\sum_l \left(\epsilon^{kl}_{hH}+\epsilon^{kl}_{Hh}\right)}$
In time element l :	$f^l_{\text{space}} = \frac{\sum_k \epsilon^{kl}_{hH}}{\sum_k \left(\epsilon^{kl}_{hH}+\epsilon^{kl}_{Hh}\right)}$	$f^l_{\text{time}} = \frac{\sum_k \epsilon^{kl}_{Hh}}{\sum_k \left(\epsilon^{kl}_{hH}+\epsilon^{kl}_{Hh}\right)}$

fraction f^l_{space}, for example, denotes the fraction of the error in (time) element l due to spatial inaccuracy. This is a slight modification from the method presented in [39], which keeps the error fractions separate for each space-time element. Next we compute the error that is targeted for each refinement choice.

$$\epsilon^k = f^k_{\text{space}} \sum_{l=1}^{N_{e,t}} \epsilon^{kl}_{hh} \qquad \epsilon^l = f^l_{\text{time}} \sum_{k=1}^{N_{e,z}} \epsilon^{kl}_{hh} \tag{4.84}$$

An error-per-cost metric is used to decide where to refine. Assuming the polynomial orders are p_z and p_t in space and time, respectively, the cost of refining a spatial element by splitting it in half is the number of additional unknowns, $4(p_t+1)N_{e,t}$ and similarly for splitting a temporal element[12]. The overall refinement metric for a spatial or temporal element is

$$E^k = \frac{\epsilon^k}{4(p_t+1)N_{e,t}} \qquad E^l = \frac{\epsilon^l}{4(p_z+1)N_{e,z}} \tag{4.85}$$

The values E^k and E^l are sorted. The largest values correspond to elements that contain a large amount of error but that are not expensive to refine. For fixed-fraction refinement, the first $\gamma_{num}(N_{e,z}+N_{e,t})$ elements are flagged for refinement. For fixed-error refinement, elements are greedily chosen based on the refinement metric until a fraction of the overall error, $\gamma_{err}(\delta K_{HH})$, has been targeted.

[12] In this work we only consider splitting elements, or h-refinement. The polynomial orders will remain fixed for the entire simulation

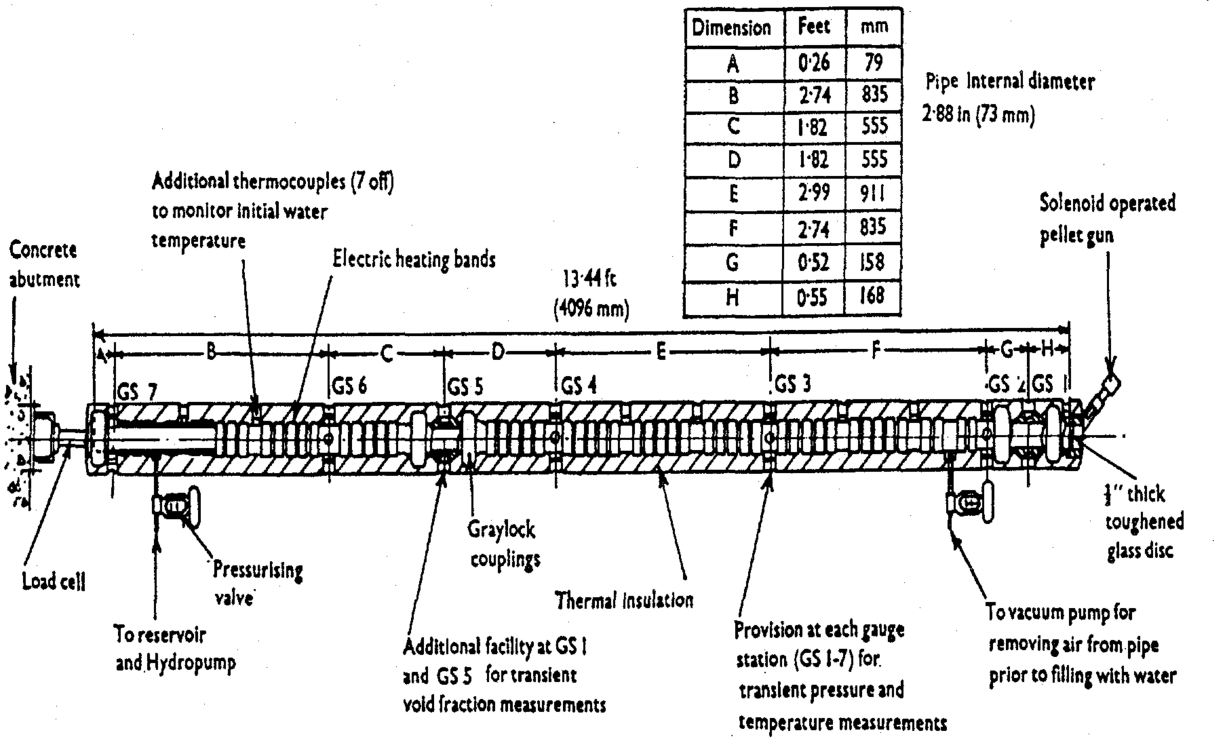


Figure 4.5: Diagram of the Edwards' blowdown experiment [36].

4.6 Edwards' Blowdown Problem

The multiphase model is applied to the Edwards' blowdown problem, illustrated in Figure 4.5. Experiments were carried out in [36] and the data have been used as a benchmark problem to validate multiphase simulations [112]. The setup consists of a long pipe filled with hot, pressurized water. At time $t = 0$, a glass disk at one end of the pipe is burst and the water begins to flow out. During the depressurization, the water in the pipe boils (or flashes) and eventually only steam is left by $t \approx 0.6$ seconds. The experiment is used as a benchmark for simulations codes because the same phenomena are at play in a loss of coolant accident (LOCA). It is important to know how quickly the depressurization takes place so that safety mechanisms like pressure sensors and pumps can be properly designed.

The following settings are used to simulate the blowdown problem.

- Liquid is set to water with reference and saturation data imported from NIST
- Gravity is off, walls are set to adiabatic. Pipe wall roughness is set to 0.001. All other source terms are on.
- Gas enthalpy is fixed to slightly above saturation, $h_{\text{gas}} = (1 + \tau_{\text{hgas}})h_{g,sat}$
- Covariance term is turned off to enhance stability
- Drift velocity is set to the Chexal-Lellouche model

The gas enthalpy parameter τ_{hgas} is set to 10^{-4}, similar to the value used in Nphase [75]. The tube radius is set to $r_{\text{pipe}} = 0.0366$m except at the outlet. Reports from the experiment indicated a 10-15% blockage from a piece of the burst disk at the pipe outlet. We model this with a radius that decreases to $A_{\text{block}} = 12\%$ using a cosine from the point $z = z_{\text{block}} = 3.8$m to the outlet $z = 4.096$m. The radius as a function of z is

$$r_{\text{pipe}}(z) = \begin{cases} 0.0366, & 0 \leq z < z_{\text{block}} \\ 0.0366 - 0.0366 A_{\text{block}} \left[1 - \cos\left(\pi \dfrac{z - z_{\text{block}}}{4.096 - z_{\text{block}}}\right)\right], & z_{\text{block}} \leq z \leq 4.096 \end{cases} \tag{4.86}$$

The initial condition is water at rest with enthalpy $h_l = 980.67$kJ/kg, pressure $p = 7$Mpa, and void fraction $\alpha = 10^{-6}$. Although the initial temperature distribution is known to be slightly non-uniform, for simplicity we take an average value. The small positive void fraction is required to keep the multiphase model stable and the value is chosen so that the gas generation source term will initially be inactive (see Section 4.1.3). The boundary conditions are set to a reflecting wall at the left end of the pipe. The right end (outlet) is set to a pressure outflow if the flow is unchoked or a supersonic outflow if the flow is choked. The pressure outflow condition should simulate the action of breaking the burst disk. We adopt a simple boundary condition which exponentially decreases from the initial pressure (7 MPa) to atmospheric pressure (0.1 MPa). The time over which the decrease occurs is

controlled by a time constant τ_{pout}

$$p_{\text{spec}} = 0.1 + (7 - 0.1)e^{-\frac{t}{\tau_{\text{pout}}}} \tag{4.87}$$

Initially we set $\tau_{\text{pout}} = 500^{-1}$s, which gives a decrease to atmospheric pressure in about 10 milliseconds. The long time constant was chosen due to numerical difficulties with shorter time constants. According to data in [107], measurements showed a decrease to atmospheric pressure in about 1 millisecond.

The Edwards' problem results in nearly all of the water initially in the pipe flowing out. At some point, the pressure at the outlet reduces to atmospheric and even below. Water (actually steam since $\alpha \approx 1$) begins to flow back into the pipe until a steady state is reached. In order to make the boundary conditions easier to implement, we do not allow the outlet to become an "inlet" at this point. Rather, the simulation is aborted when the mass flow at the outlet becomes negative and only the region with positive outflow is retained.

4.6.1 Outputs

Two outputs of the Edwards blowdown problem that are of interest to nuclear engineers are the time to reach a fixed pressure, t_{pfix} and the total mass flow out of the pipe up to that time, $\dot{m}_{\text{pfix}}$. During a loss of coolant accident, pressure sensors will trigger once the pressure has dropped to some fixed pressure p_{fix}, after which pumps may be activated to re-fill the pipe with water. The pumps are sized based on the amount of water required to re-fill the pipe. The outputs are computed by examining the outlet pressure over time. A few iterations are used to pinpoint the time t_{pfix} by driving the following residual to zero

$$R^{\text{out}}(t_{\text{pfix}}) = p(\mathbf{u}(z = 4.096, t = t_{\text{pfix}})) - p_{\text{fix}} \tag{4.88}$$

where p_{fix} is the fixed pressure. In this work we take $p_{\text{fix}} = 0.5$MPa so that the time t_{pfix} is about halfway through the blowdown transient. For this output, it is difficult to construct the

output linearization $\partial t_{\text{pfix}}/\partial \mathbf{u}$ since the output is only implicitly a function of the solution. Thus, the linearization is computed via finite differences. We assume that only the values of $\mathbf{u}$ in the space-time cell in which t_{pfix} was found have an effect on t_{pfix}[13]. Therefore, the finite difference calculation is quite fast. The second output is computed by integration

$$\dot{m}_{\text{pfix}} = \int_0^{t_{\text{pfix}}} \dot{m}(\mathbf{u}(z = 4.096, t))\ dt \tag{4.89}$$

The full linearization is

$$\frac{d\dot{m}_{\text{pfix}}}{d\mathbf{u}} = \frac{\partial \dot{m}_{\text{pfix}}}{\partial \mathbf{u}} + \frac{\partial \dot{m}_{\text{pfix}}}{\partial t_{\text{pfix}}} \frac{\partial t_{\text{pfix}}}{\partial \mathbf{u}} \tag{4.90}$$

The first term is easy to compute via analytic differentiation since it assumes that t_{pfix} remains constant. The second term can be computed using finite differences[14]. Since $\partial t_{\text{pfix}}/\partial \mathbf{u}$ is only non-zero in the space-time cell in which $p = p_{\text{fix}}$, the finite differences calculation of the second term can again be computed quickly.

4.7 Results

We solve the Edwards' blowdown problem to demonstrate the multiphase phenomena that the drift-flux formulation can simulate and to demonstrate the grid adaptation scheme. The spatial discretization has 21 elements[15] with $p_z = 2$ and the adaptive time stepping scheme (using $p_t = 1$) resulted in 33 time steps. Newton iterations on the full space-time grid were then performed to obtain a fully converged solution. Figures 4.6, 4.7, 4.8, and 4.9

[13] This assumption rules out "mulit-modal" $p(t)$ curves, where t_{pfix} could jump between two or more places in the time domain when the solution is perturbed. In fact, the solution does have two dips in the pressure, but analyzing this case would require far more finite differences, and thus more time to compute, while only yielding a more accurate adjoint when the location of t_{pfix} is about to jump.

[14] In the finite difference calculation, $\dot{m}_{\text{pfix}}$ is re-calculated with a modified $\mathbf{u}$. Thus, we actually compute the entire linearization (not just the second term) for the space-time cell in which $p = p_{\text{fix}}$ with finite differences. A fixed step size of 0.01 is used.

[15] This was constructed by taking 16 evenly spaced elements followed by splitting the last element in half twice and the second to last element into three to capture the outlet behavior. The grid size and polynomial order was decided based on trial and error and computational effort.

show the pressure, void fraction, and enthalpy of the solution in space-time. The solution is smooth but exhibits strong gradients at the initial time ($t \lesssim 10$ms) and near the outlet ($z \gtrsim 4$m). The result is a good candidate for adaptive grid refinement.

The output is set to $\dot{m}_{\text{pfix}}$, the mass flux leaving the pipe until $p = p_{\text{fix}}$. The point in time at which the pressure reaches $p_{\text{fix}} = 0.5$ MPa is $t_{\text{pfix}} \approx 0.4$ seconds. Figures 4.10, 4.11, 4.12, and 4.13 show the four components of the adjoint with respect to this output. The color scales indicate that the second component of the adjoint, corresponding to the conservation of mass for the gas phase, is the most important for the output of interest. This confirms that accuracy of the boiling model directly impacts the accuracy of the output $\dot{m}_{\text{pfix}}$.

It should be noted that there has been relatively little attention paid to solving for the adjoint in the multiphase-flow context. We demonstrate here that smooth, realistic adjoints can be obtained for multiphase problems. While the amount of coding required can be large, the major expense is computing analytic derivatives. Analytic derivatives, though, are quite useful for implicit solution methods which are often need for the stiff problems encountered in multiphase flow. In addition, the computational cost is relatively small once the code has been implemented and automatic differentiation can reduce the coding burden. Adjoints enable faster evaluation of sensitivities and error estimates and are useful for driving adaptation.

Figures 4.14, 4.15, and 4.16 compare the time histories of pressure and void fraction with experimental data [107, 36]. The initial value of $\tau_{\text{pout}} = 500^{-1}$s gave poor agreement for short times near the outlet. As a comparison, the code is run with a faster time constant $\tau_{\text{pout}} = 2000^{-1}$s which gave good agreement with the data for the first few milliseconds. The plots compare the two values with the experimental data. The shorter time constant only significantly affects the behavior at the outlet for the first few milliseconds; otherwise, the simulation does not fit the data any better than with the longer time constant. In general, the level of agreement with experiments is only approximate. This is expected due to the uncertainties, approximations, and model fidelity used. The agreement is acceptable since

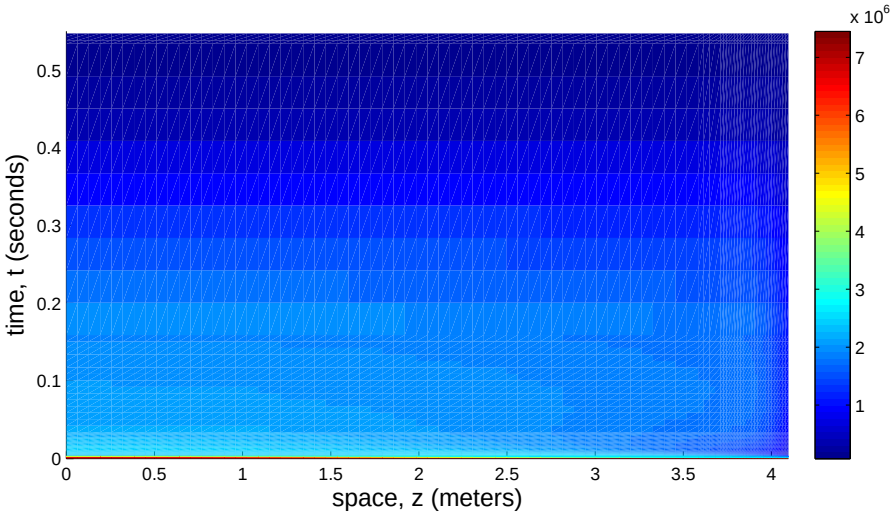


Figure 4.6: Solution to the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. Pressure is plotted.

the goal is not to faithfully replicate the experiment, but rather to demonstrate the capability of simulating realistic multiphase flow with adaptive methods.

The adaptive methodology presented in Section 4.5 was used to automatically refine the grid. The adaptive thresholds were set at $\gamma_{num} = 0.15$ and $\gamma_{err} = 0.9$ based on trial and error. The original grid from the adaptive time stepping procedure and the adjoint-adapted grids are shown in Figure 4.17. A large amount of refinement is focused on the initial time, the outlet, and a few other points in time. Interestingly, the region near $t = [0.15, 0.2]$s has a large value of the adjoint (see Figure 4.11) and is refined, even though this is not near t_{pfix}. This demonstrates the potentially non-intuitive nature of what regions of the domain affect a given output and the ability of the adjoint to detect these. Figure 4.18 shows the convergence of the output over the adaptive iterations. Overall, the error converges at an increasing rate as the grid is refined. The error is much smaller than is achieved with uniform refinement. Only one uniform refinement could be performed due to memory constraints. The adaptive method resulted in a significantly better solution (over an order of magnitude lower output error) with fewer degrees of freedom (about one half) for approximately the

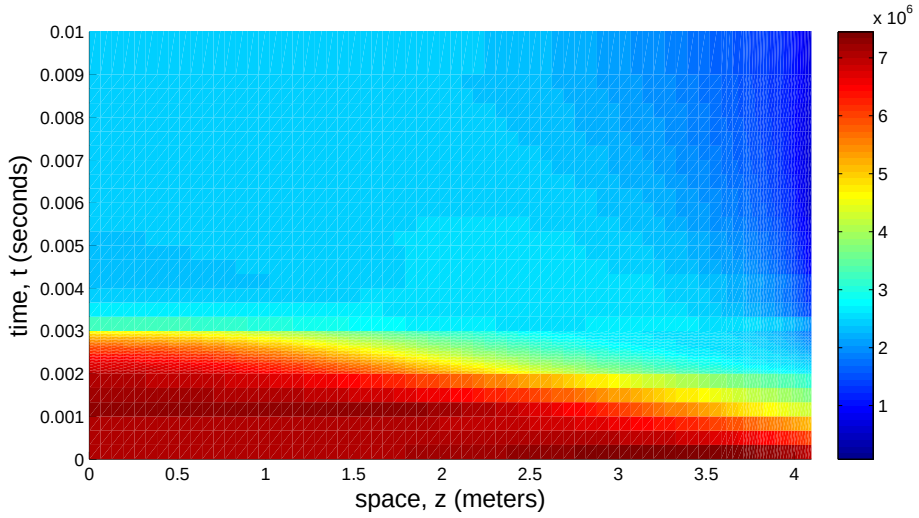


Figure 4.7: Solution to the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. Pressure for the first 10 milliseconds is plotted.

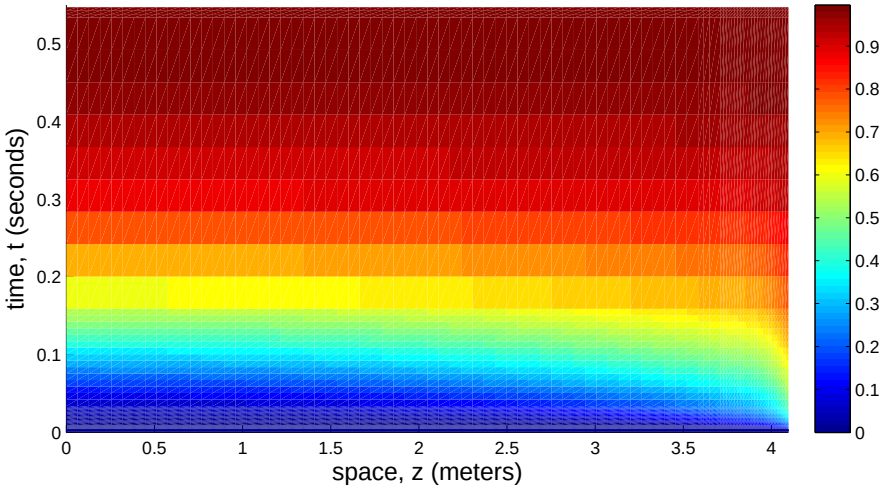


Figure 4.8: Solution to the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. Void fraction is plotted.

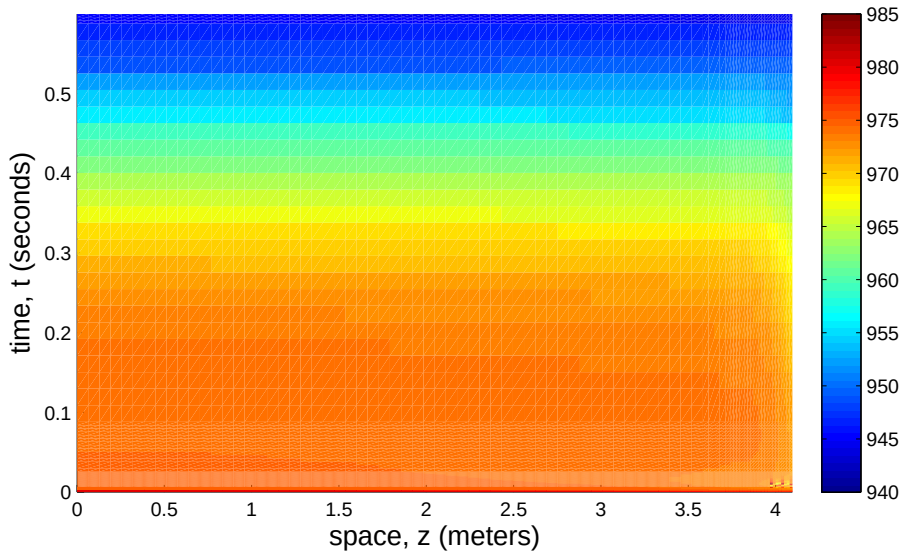


Figure 4.9: Solution to the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. Enthalpy is plotted.

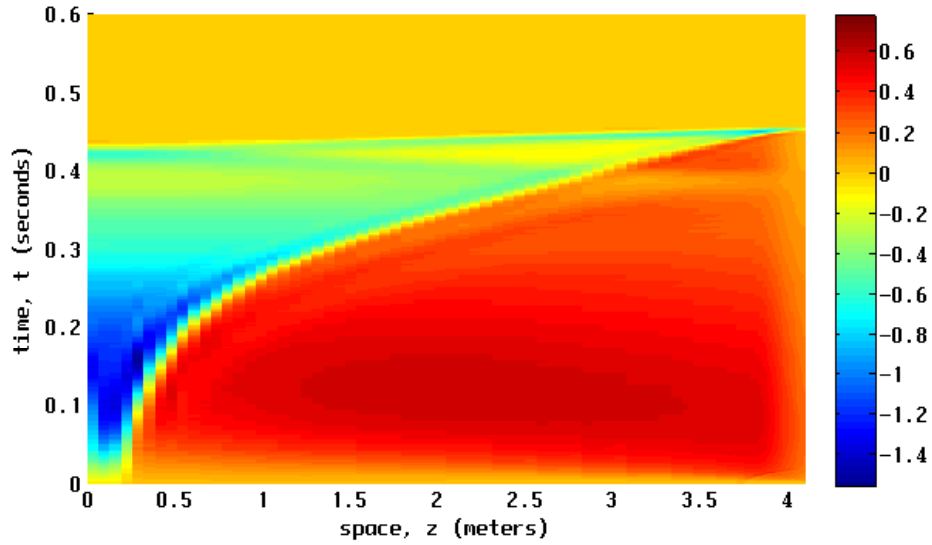


Figure 4.10: Adjoint for the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. The output is the total mass flux until $p = p_{\text{fix}}$. The mixture-mass component of the adjoint is plotted.

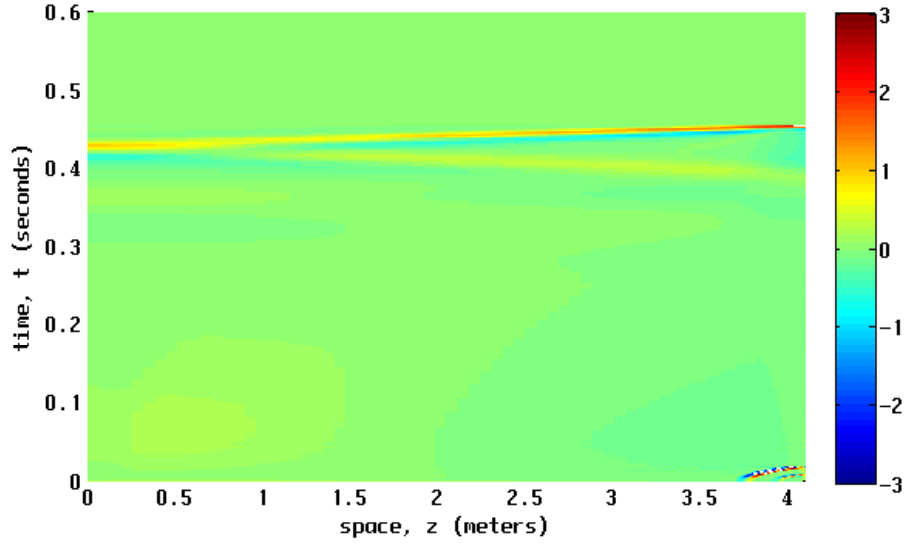


Figure 4.11: Adjoint for the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. The output is the total mass flux until $p = p_{\text{fix}}$. The gas-mass component of the adjoint is plotted.

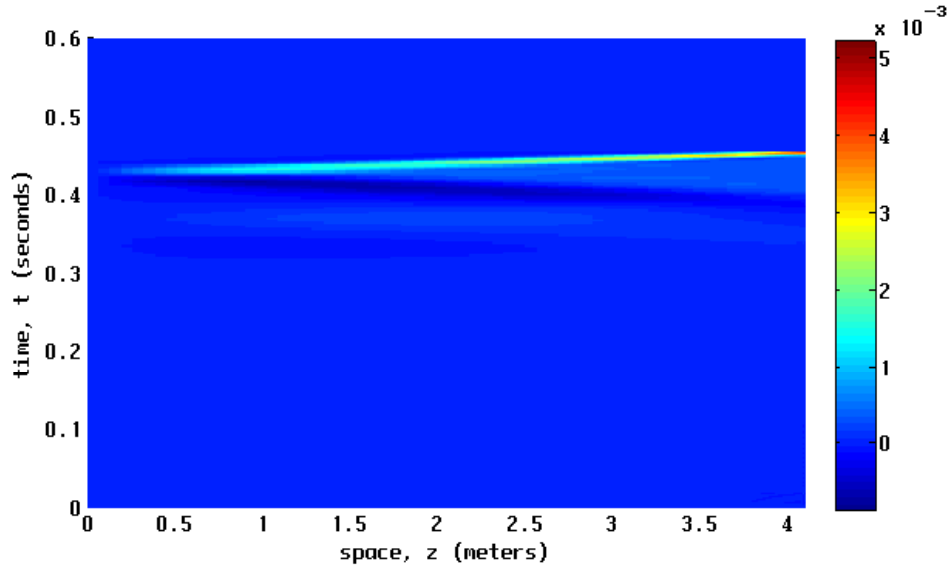


Figure 4.12: Adjoint for the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. The output is the total mass flux until $p = p_{\text{fix}}$. The mixture-momentum component of the adjoint is plotted.

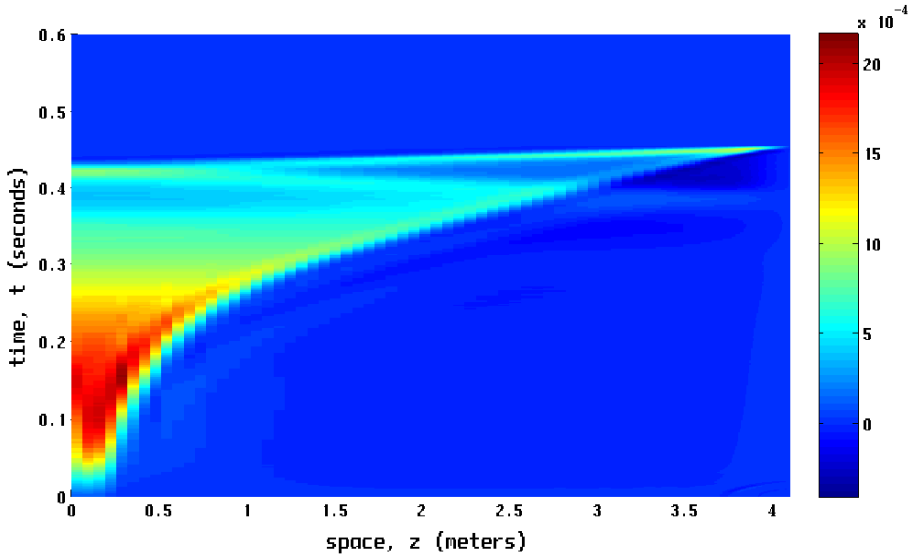


Figure 4.13: Adjoint for the Edwards' blowdown problem. Space-time plots are shown; the pipe is horizontal with the outlet on the right (at $z = 4.096$), time increases upward. The output is the total mass flux until $p = p_{\text{fix}}$. The mixture-enthalpy component of the adjoint is plotted.

same computational time compared to standard uniform refinement[16].

The outflow boundary is particularly interesting in this simulation. The outlet is the point at which the area is minimum, so it is in a sense a throat. As the water depressurizes, the outlet transitions from subsonic to supersonic. In addition, the mass flow at the outlet is the quantity of interest. Figure 4.19 shows the outflow velocity for the initial (black) and adapted (blue) grids. Clearly, the adaptation uncovered a much smoother solution with significant differences from the initial grid. First, the peak velocity moves from $t \approx 0.2$s to $t \approx 0.3$s and becomes much sharper. This change has a significant impact on the output of interest, and this is why refinement of the temporal grid is seen around this time interval in Figure 4.17. In addition, we clearly see the the transition from subsonic to supersonic (the speed of sound on the adapted grid is plotted in red), which is clearer in the close-up in Figure 4.20. The speed of sound plummets as the pressure reduces below saturation and eventually meets the increasing velocity at $t \approx 0.01$s. Interestingly, the two curves follow each

[16]Both the adaptive and uniform refinement took about one day (in parallel on 40 processors).

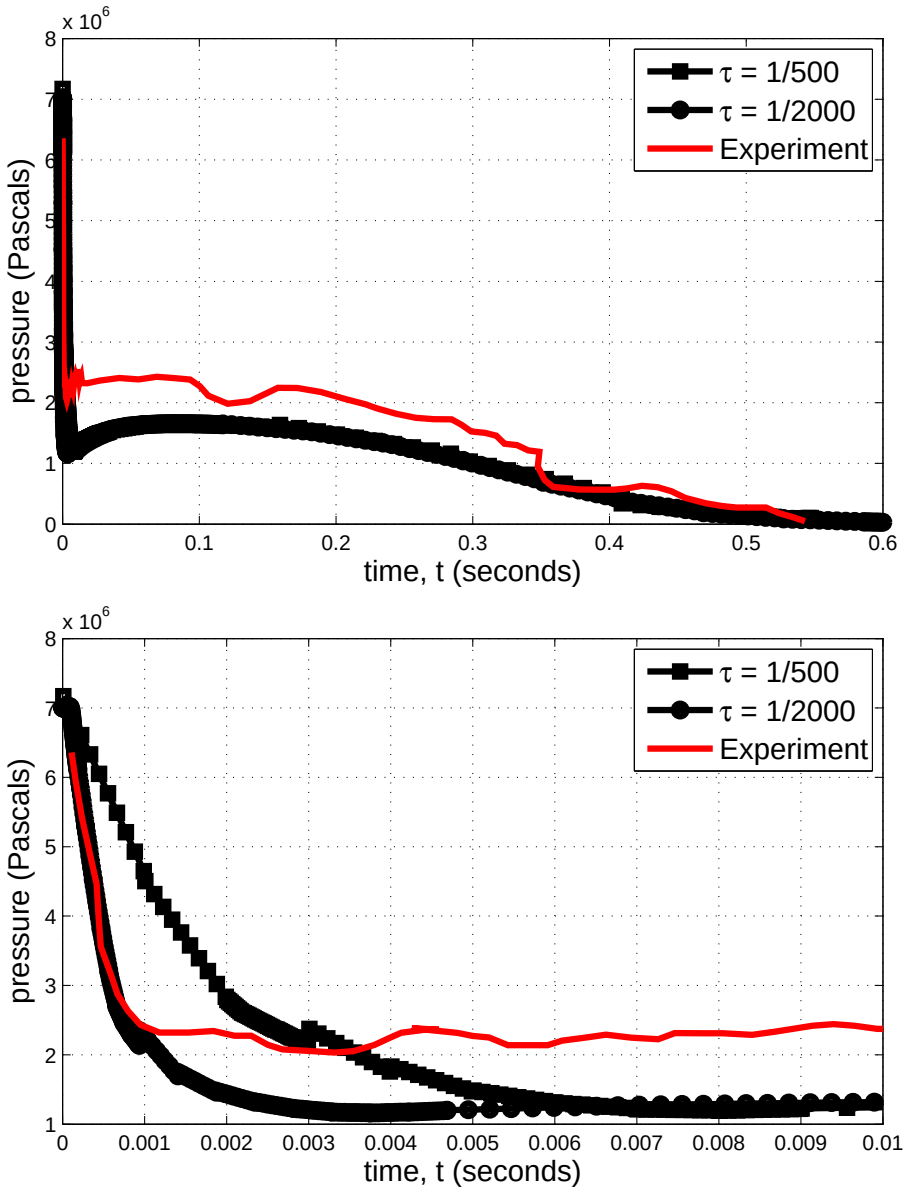


Figure 4.14: Solution to the Edwards' blowdown problem. Comparison of experimental data with two values of τ_{pout}. Pressure at the outlet (GS1). Below, a close-up of the first 15 milliseconds.

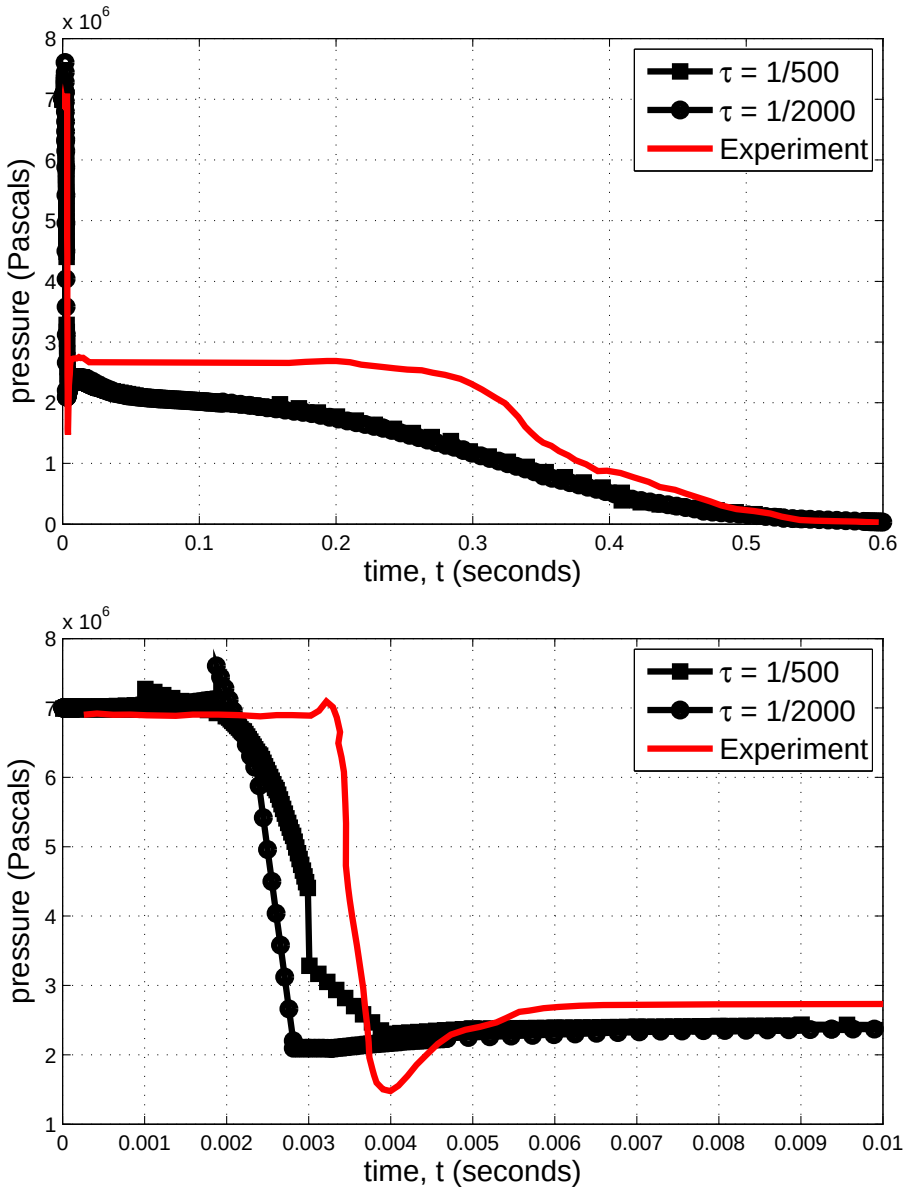


Figure 4.15: Solution to the Edwards' blowdown problem. Comparison of experimental data with two values of τ_{pout}. Pressure at the closed end (GS7). Below, a close-up of the first 15 milliseconds.

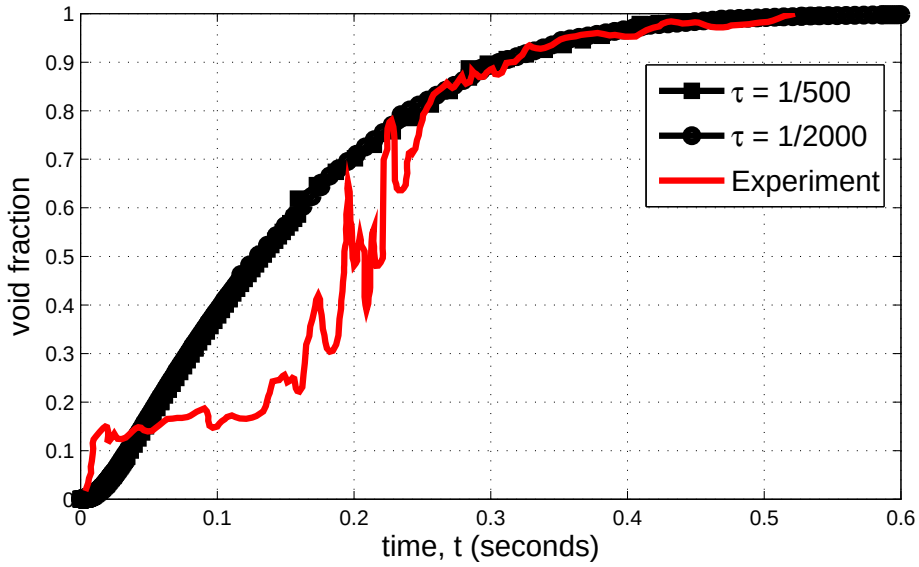


Figure 4.16: Solution to the Edwards' blowdown problem. Comparison of experimental data with two values of τ_{pout}. Void fraction in the center of the pipe (GS5).

other for a finite time interval of about 0.01 seconds. During this time interval, the outflow velocity increases but the outflow remains at the sonic condition. Further investigation is needed to develop a physical explanation for this behavior.

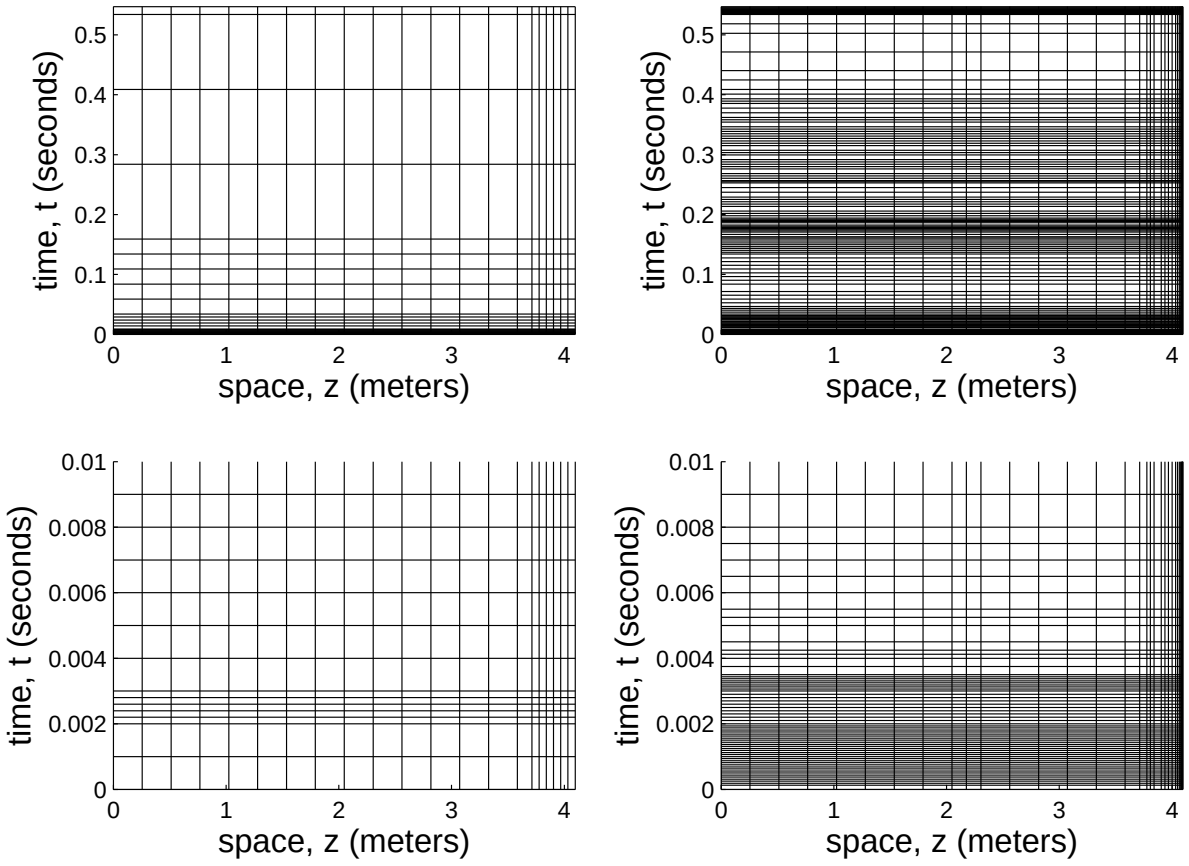


Figure 4.17: Initial grid from time step initialization procedure (left) and adapted grid (right). The bottom plots are close-ups of the first 10 milliseconds.

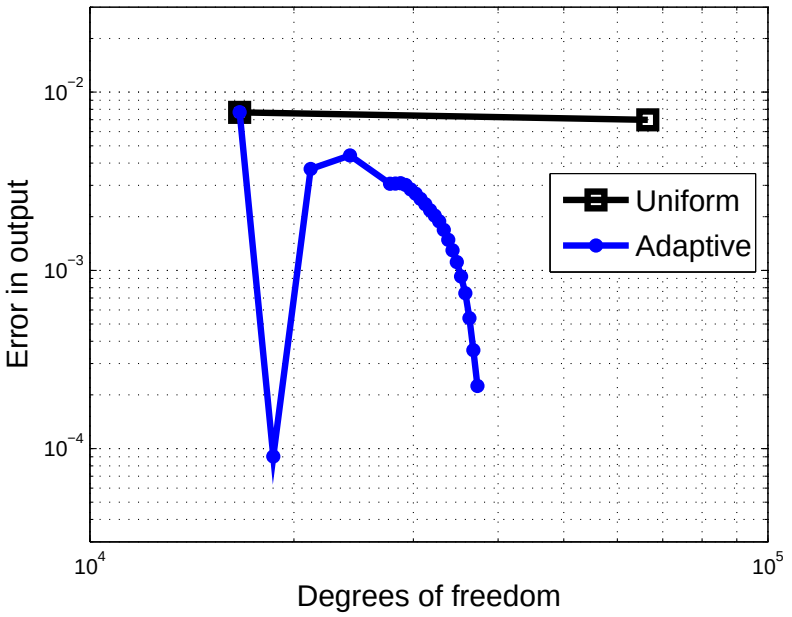


Figure 4.18: Convergence of the output for the adaptive grid example compared to uniform refinement. The output is the net mass flow out of the pipe until the pressure reaches 0.5 MPa. (Note, this is the un-corrected output.)

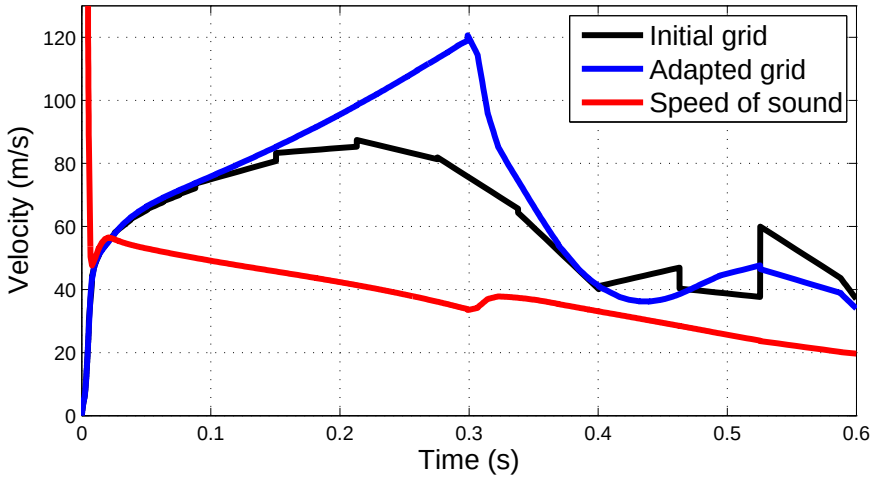


Figure 4.19: Plots of the velocity at the outlet of the pipe. Choked conditions occur when the velocity (blue or black) is higher than the speed of sound (red). The black line is the velocity for the initial grid and the blue line is for the final adapted grid.

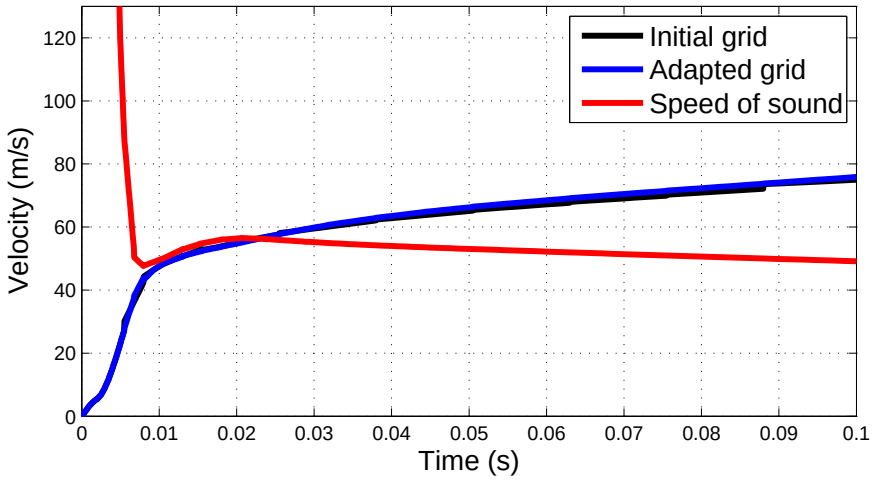


Figure 4.20: Plots of the velocity at the outlet of the pipe. Choked conditions occur when the velocity (blue or black) is higher than the speed of sound (red). The black line is the velocity for the initial grid and the blue line is for the final adapted grid. Note how sonic conditions are maintained for a finite time interval.

CHAPTER 5

Combined Adaptivity in Stochastic and Physical Domains

5.1 Introduction

In this chapter we combine the adaptive UQ method developed in Chapter 3 with the adaptive multiphase flow simulation developed in Chapter 4. The UQ study requires a number of samples, each of which can be a dynamically adapted simulation. During the UQ study, stochastic errors must be balanced with discretization errors in the physical space. This leads to a fully adaptive framework where a set of simulations is evolved to yield an accurate output of the UQ study by adding more simulations ("refinement in stochastic space") and adaptively refining existing simulations ("refinement in physical space"). The goal is to equally distribute the stochastic and physical errors over all elements in all simulations, giving an output with the best ratio of error per cost.

The overall algorithm for fully adaptive UQ is shown in Figure 5.1. The "true" subscript denotes samples in the stochastic domain, i.e. simulations with a fixed set of parameters. The regular u denotes the stochastic approximation over both the stochastic and physical spaces. In general, the "flow solves" block is the most expensive part, but it is also the easiest to do in parallel since the simulations are all independent. The stochastic model fitting usually requires matrix operations with matrices of size $n_d \times n_d$, where n_d is the

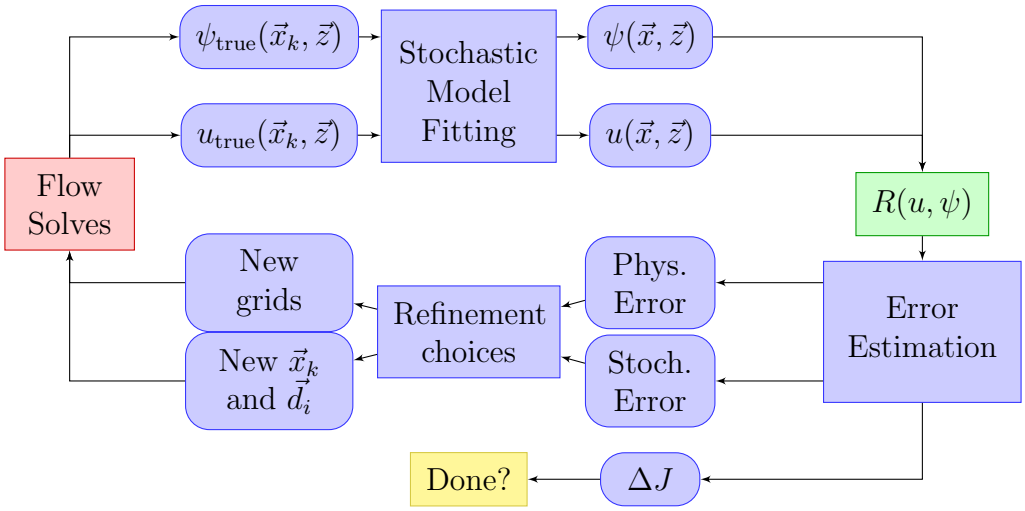


Figure 5.1: Diagram of solution process for fully adaptive UQ applied to a numerical simulation of a PDE.

number of stochastic variables. This block synthesizes the results from all of the simulations into a model over the entire stochastic space. Next, the residual block $R(u, \psi)$ involves some extra analysis but no full solutions. This block is also easily done in parallel and is therefore relatively fast. The block yields an overall error estimate (which is used for the convergence criteria) and separate error estimates for the various types of refinement (stochastic, spatial, temporal). These are fed into a heuristic "refinement choices" block which attempts to target the most error with the least added expense. The simple two-dimensional example in Section 5.2 will use this block diagram with simplified methods in each block. The Edwards blowdown example in Section 5.3 uses the same procedure with more complicated flow solves and stochastic models.

5.2 Two Dimensional Example

We begin with a simple example of equidistribution of errors where the physical and stochastic spaces each have only one dimension. In physical space, the 1D model problem is

a convection-diffusion-reaction problem

$$-x_0 \frac{\partial^2 u}{\partial z^2} + c\frac{\partial u}{\partial z} = S(z, x_0) \tag{5.1}$$

Here, the physical dimension is z and the stochastic parameter is x_0. The source term is chosen such that the exact solution is

$$u_{\text{true}} = \sin\left(x_0 \frac{z^2}{5}\right) + \ln x_0 \tag{5.2}$$

The ranges are taken as $1 \leq z \leq 5$, $1 \leq x_0 \leq 3$ to provide a complex function in both physical and parameter space. The physical output is chosen to be the integral

$$K(u) = \int_2^3 u(z)\, dz \tag{5.3}$$

The physical solution is found with the Discontinuous Galerkin method as in Section 4.2, which can be considerably simplified for this simple, scalar problem. The discrete adjoint equations are also solved using the discontinuous Galerkin discretization. An example solution with $c = 5, x_0 = 2.0$ is shown in Figure 5.2. The full stochastic-physical solution is shown in Figure 5.3.

For simplicity we will use linear interpolation to represent the solution in stochastic space. The stochastic grid initially consists of evenly distributed points, $x_{0,j}$. As refinement proceeds, new points in stochastic space are added adaptively. Note, this method works well with only one stochastic dimension, but creating such a "grid" in the high-dimensional spaces often encountered in UQ can be prohibitively expensive. This example serves to demonstrate the benefits of combining adaptivity in the physical and stochastic spaces, while later we will use the direction-based approximation in the stochastic space for a more realistic problem.

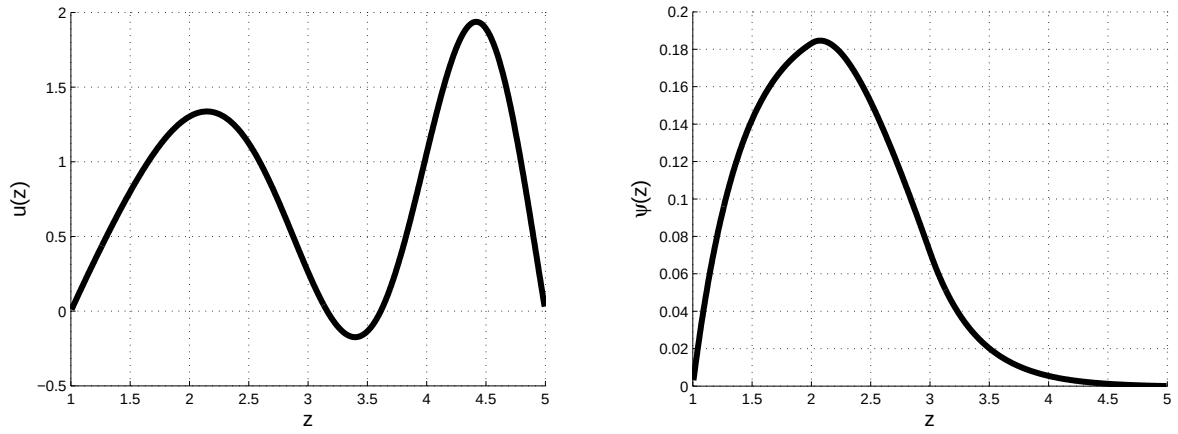


Figure 5.2: Example solution and adjoint for the convection-diffusion-reaction problem. The convection speed is $c = 5$ and the stochastic diffusion coefficient is $x_0 = 2.0$. The state variable $u(z)$ (left) and the adjoint $\psi(z)$ (right).

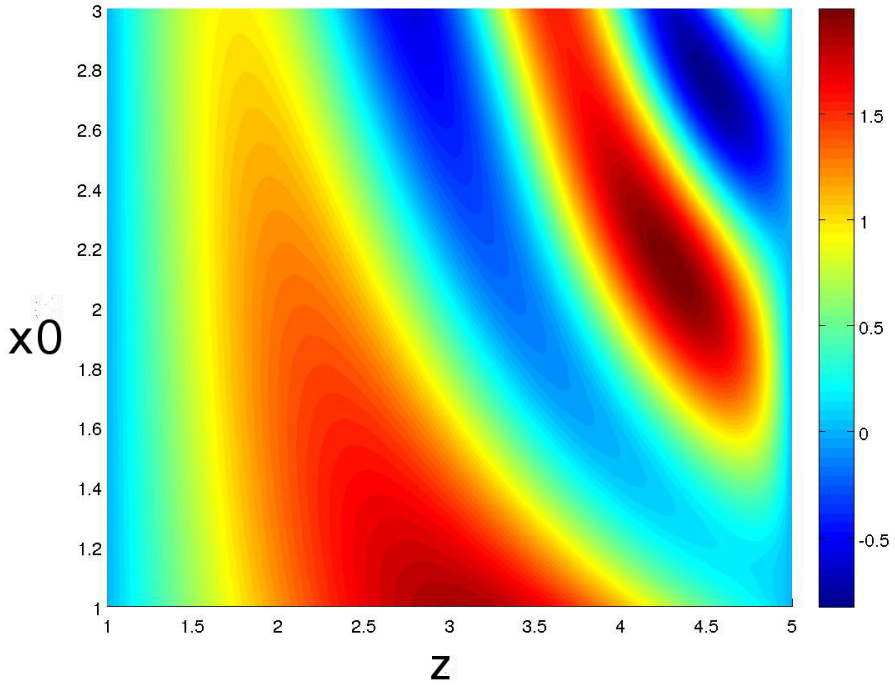


Figure 5.3: Exact solution for the example problem. Horizontal axis is the physical space, vertical axis is the stochastic space. Color denotes the value of $u(z, x_0)$.

5.2.1 Error Estimation

The dual-weighted residual approach is used to drive the adaptation. The physical residual from the Discontinuous Galerkin discretization of Equation 5.1 is deonted $R_H^{cdr}(u_H; x_0) = 0$. The subscript $_H$ denotes the physical grid and the superscript ^{cdr} denotes the convection-diffusion-reaction equation. The discrete adjoint is defined by

$$\frac{\partial R_H^{cdr}}{\partial u_H}^T \psi_H = -\frac{\partial K_H}{\partial u_H}^T \tag{5.4}$$

where $K_H(u_H)$ is an approximation of the output, computed by Gaussian integration. For a single simulation, the physical-space error estimate is

$$\Delta K_H = \psi_h^T R_h^{cdr}(\mathcal{I}_h^H u_H) \tag{5.5}$$

where ψ_h is either computed exactly or is a higher-order, smoothed interpolation of ψ_H. As discussed in Section 4.4, taking ψ_h as an smoothed interpolation of ψ_H may degrade the accuracy of the error estimate if the discretization is not fully adjoint consistent or there are sharp changes in ψ_H. In this case, the discretization is fully adjoint-consistent and the adjoint is smooth, so interpolation is a good option. In addition, the error estimate is only used to drive adaptation, so some inaccuracy can be tolerated.

The injection operator in Equation 5.5, $\mathcal{I}_h^H$, produces a representation of u_H on the fine grid h. That is, $\mathcal{I}_h^H u_H$ is exactly the same function as u_H, just represented on a different grid. Thus, the residual measures the errors in u_H that would be resolved by obtaining a solution on the fine grid.

Now we add an extra subscript to denote the discretization level in stochastic space. For example, u_{hH} denotes the solution on the refined spatial grid and the coarse stochastic grid.

In addition, we define the overall output of the UQ study as

$$J = \int j(K)\ dx_0 \tag{5.6}$$

For the example in this section, we simply take $j(K) = K$. The output is computed with Gaussian integration in stochastic space for a given stochastic grid: $J_{HH} = \int j_H(K_{HH})dx_0$. This leads to the adjoint-based error estimate

$$\Delta J_{HH} = \int \frac{\partial j_H}{\partial K_{HH}} \sum_i \psi_{hh,i} R^{cdr}_{hh,i}(\mathcal{I}^{HH}_{hh} u_{HH})\ dx_0 \tag{5.7}$$

The index i runs over all components of the adjoint and residual. The error estimate is integrated in stochastic space with a Legendre-Gausss-Lobatto (LGL) integration rule applied within each interval $[x_{0,j-1}, x_{0,j}]$. The integration points of this rule include the endpoints of the interval, so the adjoint and residual evaluations are easy and accurate there (no interpolation in x_0 space is necessary). At a general point x_0, the solution $u_{HH}(x_0)$ is simply the value from linear interpolation. The adjoint ψ_{hh} is interpolated in physical space and stochastic space.

For the physical space interpolation, we fit a polynomial of degree $p_z + 1$ to $p_z + 2$ of the solution nodes (each element has $p_z + 1$ equally spaced solution nodes). This produces a "sliding window" type of interpolation. Where these polynomials overlap, the average is taken. For stochastic space, we use four neighboring points $(x_{0,j-1}, x_{0,j}, x_{0,j+1}, x_{0,j+2})$ to create a quadratic function. If the point in stochastic space is near a boundary, only the three nearest simulations are used. This is done separately for each region $[x_{0,j}, x_{0,j+1}]$. Note, many types of interpolation are possible and equivalent for the purpose here, the methods used here were chosen for ease of implementation.

A common splitting of the error estimate is based on two adjoints, first the coarse adjoint ψ_{HH} and then the fine, interpolated adjoint ψ_{hh} [114, 38]. The splitting reflects the potentially much higher cost in computing fine adjoint compared to the coarse, and the po-

tential gain in accuracy in the resulting error estimates. Since the coarse adjoint is typically available at little extra cost, it is used in the "computable correction" ΔJ^c_{HH}

$$\Delta J^c_{HH} = \int \frac{\partial j_H}{\partial K_{HH}} \sum_i \mathcal{I}^{HH}_{hh} \psi_{HH,i} R^{cdr}_{hh,i}(\mathcal{I}^{HH}_{hh} u_{HH}) \, dx_0 \tag{5.8}$$

The fine adjoint requires extra work (interpolation or a full solve) and it contributes to the "remaining error" ϵ_{hh}

$$\epsilon_{hh} = \int \frac{\partial j_H}{\partial K_{HH}} \sum_i \left(\psi_{hh_i} - \mathcal{I}^{HH}_{hh} \psi_{HH,i}\right) R^{cdr}_{hh,i}(\mathcal{I}^{HH}_{hh} u_{HH}) \, dx_0 \tag{5.9}$$

Since the computable correction is easy to compute (at least for one stochastic dimension), we assume that the output can always be corrected as

$$J^c_{HH} = J_{HH} + \Delta J^c_{HH} \tag{5.10}$$

and the corrected output is reported in the results below. The remaining error is used to drive the adaptation. We use the notation $\delta\psi_{hh} = \psi_{hh} - \mathcal{I}^{HH}_{hh}\psi_{HH}$ to indicate the adjoint increment associated with the remaining error.

The error estimate can be localized as before

$$\epsilon^{k,j+1/2}_{hh} = \int_{x_{0,j}}^{x_{0,j+1}} \frac{\partial j_H}{\partial K_{HH}} \sum_{i \in (k)} \delta\psi_{hh,i} R^{cdr}_{hh,i}(\mathcal{I}^{HH}_{hh} u_{HH}) \, dx_0 \tag{5.11}$$

Here, the sum in physical space is done for all degrees of freedom i within a coarse-grid element k. That is, the coarse element k is subdivided to get the fine grid and $i \in (k)$ indexes over the subelements of k. The error estimate is treated as separate for each stochastic region $[x_{0,j}, x_{0,j+1}]$.

As the adaptation proceeds, some physical grids (at certain $x_{0,j}$) may become more refined than others. If this is the case, then the solutions on two neighboring grids cannot

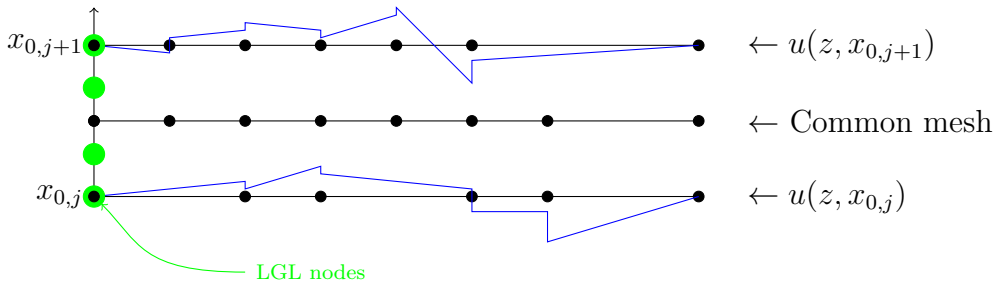


Figure 5.4: Diagram of construction of a common grid. Physical space is horizontal, stochastic space is vertical. Two physical grids are shown at $x_{0,j}$ and $x_{0,j+1}$.

be directly compared since they have different numbers of unknowns. However, we must have some way to interpolate among various physical grids. To that end, we introduce the notion of a "common grid," i.e. a grid that is used for such comparisons. Each physical solution is projected (without smoothing) onto the common grid and comparisons are done on the common grid. A diagram of this is shown in Figure 5.4. In general, the common grid should be at least as fine as any of the individual physical grids so that all of the physical solutions can be precisely represented on it. If the physical grid starts out the same for all $x_{0,j}$ at the beginning of the adaptive process, it is easy to keep track of the refinements (just successive splitting of the elements) and to combine the refinements of all physical grids to get a common grid. The common grid is what is denoted by H, while a uniformly refined version of it is the fine physical grid, denoted by h.

In general, a separate common grid could be constructed for each comparison based on the relevant physical grids that are being compared. Alternatively, a single common grid can be used for the entire UQ study. In this section, comparisons are only done between two physical grids at a time, so a global common grid is not needed. A new common grid is constructed for each pair of simulations. When more stochastic dimensions are considered, however, it is more convenient to use a single common grid for the entire stochastic space.

5.2.2 Anisotropic Adaptation

The same framework for space-time anisotropic adaptation presented in Section 4.5 can be used for the stochastic and physical spaces here. The error estimate in two "partially refined" spaces is compared to decide between stochastic and physical refinement. The error estimate for physical space is

$$\epsilon_{hH}^{k,j+1/2} = \frac{1}{2}\left[\frac{\partial j_H}{\partial K_{hH}}\sum_{i\in(k)}\delta\psi_{hH,i}R_{hH,i}^{cdr}(\mathcal{I}_{hH}^{HH}u_{HH})|_{x_{0,j}} + \frac{\partial j_H}{\partial K_{hH}}\sum_{i\in(k)}\delta\psi_{hH,i}R_{hH,i}^{cdr}(\mathcal{I}_{hH}^{HH}u_{HH})|_{x_{0,j+1}}\right] \tag{5.12}$$

This is the physical error on physical element k for the stochastic region $[x_{0,j}, x_{0,j+1}]$. It measures the amount of physical discretization error that is due to the physical element Ω_k (on the common grid). It is not affected by the stochastic parameter x_0 because it is evaluated at locations where the effect of x_0 is known (i.e. the sample locations $x_{0,j}$).

In contrast, the error estimate for the stochastic space is

$$\epsilon_{Hh}^{k,j+1/2} = \sum_q w_q \frac{\partial j_h}{\partial K_{Hh}}\sum_{i\in(k)}\delta\psi_{Hh,i}R_{Hh,i}^{cdr}(\mathcal{I}_{Hh}^{HH}u_{HH})|_{x_{0,q}} \tag{5.13}$$

Here, the points $x_{0,q}$ are quadrature points for the LGL integration in stochastic space, and w_q are the associated weights. This measures the amount of error due to inaccurate representation of u and ψ in the stochastic space. The solution and adjoint are evaluated on the coarse common grid and (linearly) interpolated to the points $x_{0,q}$. The injection operator $\mathcal{I}_{Hh}^{HH}$ performs the linear interpolation in stochastic space[1]. The adjoint increment is the difference between the higher-order smoothed adjoint (ψ_{Hh}) and the linearly interpolated coarse adjoint: $\delta\psi_{Hh} = \psi_{Hh} - \mathcal{I}_{Hh}^{HH}\psi_{HH}$. The residual at those points is zero if the parameter x_0 has only a linear effect on the solution. If the parameter affects the solution in a higher-order way, there will be non-zero residuals and hence a nonzero $\epsilon_{Hh}^{k,j+1/2}$. Since the error estimate is evaluated on the coarse common grid, errors due to the physical discretization

[1] In a sense, we take the stochastic "discretization" to be interpolation, either linear (coarse, H) or quadratic (fine, h).

are not measured.

The fractions of error in the stochastic and physical spaces are computed using the partially-refined error estimates. In this example, we use a slightly simpler approach than the one presented for space-time adaptation. A fraction of stochastic error is averaged over the entire space to give an overall stochastic-fraction and physical-fraction of the error

$$f_{\text{phys}} = \frac{\sum_{k,j} \epsilon_{hH}^{k,j+1/2}}{\sum_{k,j} \left(\epsilon_{hH}^{k,j+1/2} + \epsilon_{Hh}^{k,j+1/2} \right)}, \qquad f_{\text{stoch}} = 1 - f_{\text{phys}} \tag{5.14}$$

Next, we must decide which elements of existing simulations to refine and where to add new simulations[2]. We focus on fixed-fraction refinement, so at each adaptive step there is a budget of new degrees of freedom which are allocated to different refinement choices. This is a classic problem in computer science called the knapsack problem [13]. There are two goals here. The first is to distribute the error evenly in both stochastic and physical space, resulting in the fewest degrees of freedom for a given level of error. The second is to reduce the number of simulations, since starting up and running a simulation incurs overhead. The second goal helps to reduce the simulation time for a given amount of error. The overall cost (in terms of expected computation time) of a given set of refinements, then, is non-linearly related to the error targeted by that set of refinements. The overhead cost significantly complicates the knapsack problem and renders an exact solution intractable[3]. Instead, a heuristic method is used here with a tuning parameter γ_{oh} that is related to the overhead cost of starting a simulation. A large value of γ_{oh} assumes significant overhead cost and will refine many elements in only a few simulations. A small value of γ_{oh} will distribute refined elements over many simulations (assuming the error is actually spread out like that). If

[2]In this example, we consider only splitting the x_0 intervals in half by adding a new simulation at $x_{0,j+1/2}$. New simulations are assumed to be computed on the grid used to initialize the simulations (not the common grid). This allows for a variety of coarse and fine grids, which can yield a more efficient approximation of the stochastic space behavior than requiring a minimum level of refinement everywhere in the stochastic space.

[3]For space-time adaptation, there is no additional overhead cost for refining in space versus time. Thus, the knapsack problem remains linear and is easy to solve. In Section 4.5, the problem is solved approximately by refining the elements with the highest error-per-cost (a greedy method).

much of the error is concentrated in only a few simulations, then the parameter will have little or no effect.

The heuristic method uses a standard fixed-fraction parameter γ_{num} to determine the overall increase in degrees of freedom. The steps are:

1. Create new simulations where $\sum_k \epsilon_{Hh}^{k,j+1/2}$ is largest, using up to $f_{\text{stoch}}\gamma_{num}N_{dof}$ new degrees of freedom.

2. Refine elements of existing simulations where $\epsilon_{hH}^{k,j}$ is largest, using up to $(1-\gamma_{oh})f_{\text{phys}}\gamma_{num}N_{dof}$ new degrees of freedom.

3. Add further refinements to existing simulations that have at least one element already flagged for refinement from step 2, using up to $\gamma_{oh}f_{\text{phys}}\gamma_{num}N_{dof}$ new degrees of freedom.

The total added degrees of freedom is $\gamma_{num}N_{dof}$, where N_{dof} is the number of degrees of freedom in at the previous iteration.

An example of the adaptive solution process is shown in Figure 5.5. In the figure, simulations are single rows and black dots represent degrees of freedom. The background color shows the error. The heuristic method has chosen to add two new simulations (rows of red dots) near the top and bottom where errors are large. In a few spots, large errors are seen in red and existing simulations in that area are targeted for refinement.

5.2.3 Results

The results of the method are shown as error as a function of degrees of freedom and compute time. The compute time is specific to the machine used (2xQuadcore Intel Xeon E5630, 16Gb RAM), but the relative comparison between uniform and adaptive refinement is useful. The error is measured by the difference between the corrected output and the true output, computed on a very fine grid[4]. We compare two strategies for computing

[4]In this case, the "true" solution was computed using the adaptive method with $p_z = 3$ and 7,168 total degrees of freedom. The difference in the output between the final two adaptive iterations in this run was $< 3 \times 10^{-8}$.

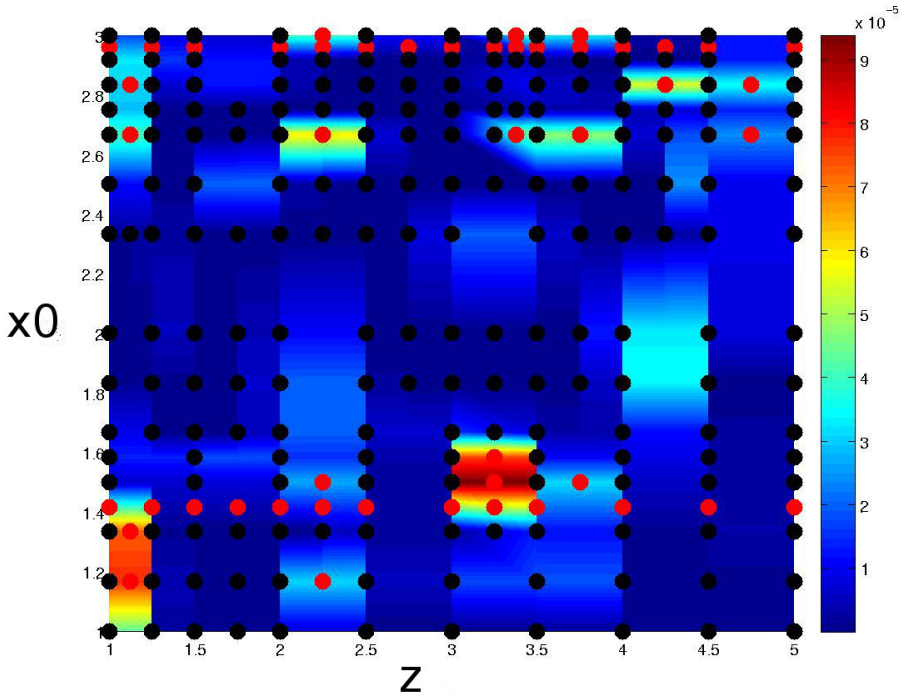


Figure 5.5: Example of stochastic and physical space adaptation. Horizontal axis is pnysical space, vertical axis is stochastic space. Each row is a single physical simulation. Black dots represent degrees of freedom at the previous iteration. Red dots represent added degrees of freedom, either by adding new simulations (rows) or by refining elements of existing simulations. In this case, two simulations are added and seven existing simulations are refined. The background color is the adjoint-weighted-residual error $\epsilon_{hh}^{k,j+1/2}$. Note how each simulation (row) has a different grid. The initial grid had eight physical elements and three simulations at $x_0 = 1, 2, 3$. The overhead factor is $\gamma_{oh} = 0.5$.

the integrals in stochastic space (specifically the quadrature in Equation 5.13). The first strategy uses only information at a single new point which is the midpoint between two existing simulations, essentially integrating with the midpoint rule. This is inspired by collocation methods in uncertainty quantification, which focus on point evaluations. The second strategy uses a high order LGL integration that requires more residual evaluations. This strategy incurs extra cost but brings in more information from the equations and could result in better error estimates. The LGL integration is inspired by stochastic finite-element methods, which attempt to perform domain integrations in stochastic space. Figure 5.6 shows the performance of the adaptive method compared to uniform refinement in stochastic and physical space. Note, the overhead parameter γ_{oh} is set to 0.5 and the refinement parameter $\gamma_{num} = 0.2$. The corrected output converges like $O(N_{dof}^p)$ with $p = -1$, consistent with linear approximation in stochastic space and $p_z = 1$. If we look at the stochastic-physical element size $h \propto 1/\sqrt{N_{dof}}$, then the convergence is like $O(h^2)$. Note, however, that a single element size or degree-of-freedom count hides the different costs of stochastic vs. physical refinements. From the plot, the adaptive method performs better than uniform refinement with respect to computation time, but it is not more efficient in terms of error per degree of freedom. This is because the error is mostly evenly distributed in the physical and stochastic domains (see the true solution in Figure 5.3). Nonetheless, the benefits of the current adaptation scheme are twofold. First, sometimes a full uniform refinement is not possible for very large simulations. The adaptive method creates an intermediate grid which has the same optimal error distribution. Second, the adaptive method is clearly faster in terms of compute time, despite the extra expense of solving the adjoint equation and calculating the error metrics.

The adjoint-based adaptive indicators require extra computational resources and it is important to assess the benefit of using those resources. To that end, we compare the adjoint-based indicators with adaptive indicators based on interpolation error, jumps in the solution, and unweighted residuals. The interpolation error is the difference between an interpolated solution (in stochastic and/or physical space) and the current solution. Third

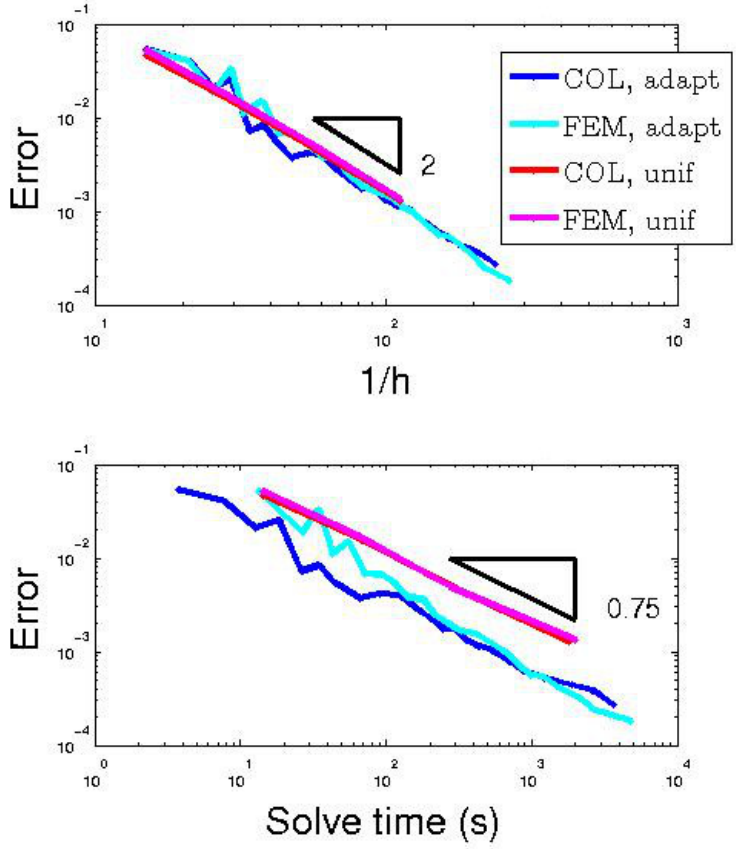


Figure 5.6: Convergence of the adaptive method for two integration strategies ("col" for collocation, "fem" for full LGL integration), comparing adaptive to uniform refinement. Since there are two dimensions (one physical, one stochastic), the horizontal axis is $1/h = N_{dof}^{1/2}$. The spatial polynomial order is $p_z = 1$ and $\gamma_{oh} = 0.5$.

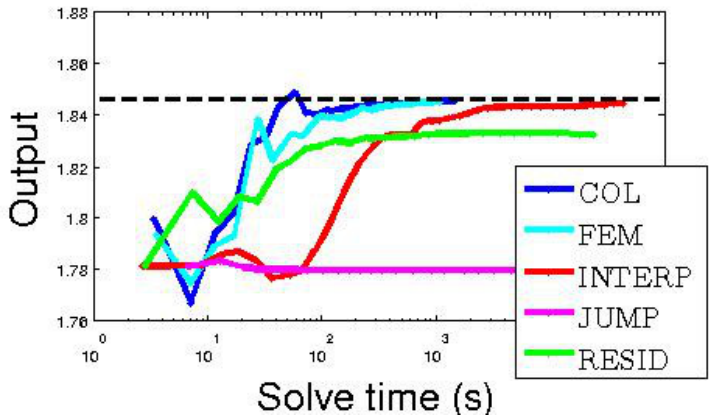


Figure 5.7: Convergence of the adaptive method for other adaptive indicators. Comparison between collocation and LGL integration of the adjoint-based indicator with indicators based on interpolation error, equation residuals, and jumps in the solution across physical elements. The black dashed line indicates the "true" value of the output. The spatial polynomial order is $p_z = 1$ and $\gamma_{oh} = 0.5$.

order interpolation was used in the physical and stochastic spaces (see Section 5.2.1 for a description of the interpolation procedure). The jump-based indicator takes the average of the jumps on the left and right sides of a physical element as the indicator for that element. For stochastic regions, the jumps in the two nearest simulations are averaged. The residual indicator is simply the sum of the residuals in the physical element. For stochastic regions, we use the residual of the interpolated solution at the midpoint $x_{0,j+1/2}$. The results are shown in Figure 5.7. The output-based methods perform the best even in terms of run-time. The interpolation error indicator eventually converges to the correct value, but the jump and residual-based indicators converge to the incorrect value. This behavior has been reported before in [38] and shows the importance of using an adaptive indicator that is tied to the output of interest.

Finally, we investigate the impact of the overhead factor γ_{oh}. Three different settings are shown in Figure 5.8, $\gamma_{oh} = 0.3, 0.5$, and 0.8. The difference in performance is generally small, though the large value of 0.8 leads to more erratic convergence because it concentrates extra physical refinement on only a few simulations.

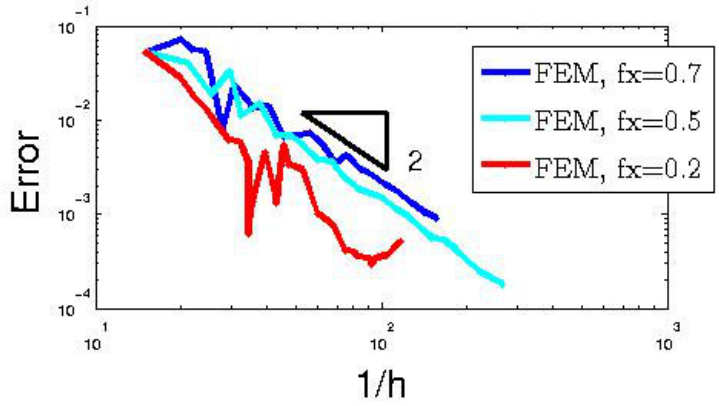


Figure 5.8: Convergence of the adaptive method different values of $\gamma_{oh} = 1-\text{fx} = 0.3, 0.5,$ and 0.8. LGL integration is used to compute the error estimates. Since there are two dimensions (one physical, one stochastic), the horizontal axis is $1/h = N_{dof}^{1/2}$. The spatial polynomial order is $p_z = 1$.

This relatively simple study of combined adaptivity in physical and stochastic spaces demonstrated the advantages of the current method. First, using output-based, anisotropic, adaptive indicators results in faster convergence (in terms of computation time) compared to simpler indicators. Second, a heuristic for deciding how to refine in the stochastic and physical spaces is sufficient. Third, utilizing a common grid for comparing solutions on different grids was acceptable and enables a multi-fidelity UQ study. That is, some simulations (samples) can be relatively coarse while others may be more highly refined. We expect that for problems with more anisotropy, the methods developed here will be even more beneficial.

5.3 Full UQ Study

5.3.1 Parameters and Ranges

The full UQ study of the Edward's pipe problem is a straightforward combination of the UQ method developed in Chapter 3 and the multiphase flow simulation developed in Chapter 4. Due to time and resource constraints, a demonstration run is done with six active parameters as shown in Table 5.1. The parameter choices are based on some of the known

uncertainties in the experimental setup as discussed in [36] and [112]. These include the blockage area and the initial enthalpy and pressure. The roughness and gravitational acceleration (simulating a tilted pipe) are not at all known for the experiment, but representative ranges are chosen. The large roughness value is included to help simulate valves and other obstructions in the pipe.

The blockage location and outlet pressure time constant are known to some extent, but the current model cannot accommodate more realistic values. The blockage location should be at the very end of the pipe ($z_{\text{block}} = 4.096$). However, the model has only one spatial dimension and the discontinuous Galerkin discretization assumes smoothness of the solution, so the blockage must be "smeared out" over a finite length. The range of the blockage location was chosen so that a solution could be obtained with the same spatial mesh for all values[5]. The outlet pressure boundary condition is assumed to be an exponential decay. As discussed in Section 4.6, the true outlet pressure behavior is not known, and the experimental data are approximately consistent with the value $\tau_{\text{pout}} = 2000^{-1}$s. The range chosen for this study is consistent with the values tested in Chapter 4.

Finally, the Prandtl number and gas enthalpy parameter are values that should not affect the simulation much and many models assume these are constant. The Prandtl number varies by about 10%, which is a convenient range that is loosely related to the actual variation in the domain. At saturation, the Prandtl number ranges from 0.834 to 0.983 for pressures between 1 and 10 MPa, which is representative of the pressures in the solution. The gas enthalpy parameter does not have a known range, so a simple (and arbitrary) doubling of τ_{hgas} is tested.

5.3.2 Discretization

In order to reduce computational expense, the UQ study is focused just on the initial transient of the Edwards blowdown problem. The temporal domain is restricted to $[0, 3 \times$

[5] An alternative would be to specify some hueristic or a robustness-based adaptive procedure to adapt the spatial mesh for a non-converged solution

Table 5.1: Parameters and ranges for the full UQ study.

Parameter	Symbol	Range	Notes
Roughness	ϵ/r_{pipe}	$[10^{-3}, 1]$	logarithmic scale
Blockage area	A_{block}	$10 - 15\%$	
Blockage location	z_{block}	$[3.5, 3.8]$ m	pipe length is 4.096m
Initial enthalpy	$h_{l,\text{init}}$	$[980.67, 985.67]$ kJ/kg	
Outlet pressure time constant	τ_{pout}	$[2000^{-1}, 500^{-1}]$ s^{-1}	logarithmic scale
Initial pressure	p_{init}	7 MPa $\pm$5 Pa	
Gravitational acceleration	g_z	$[-0.171, 0.171]$ m/s^2	simulates a -1 to 1 degree tilt
Prandtl number	Pr	$[0.7337, 0.9337]$	
Gas enthalpy parameter	τ_{hgas}	$[10^{-4}, 2 \times 10^{-4}]$	

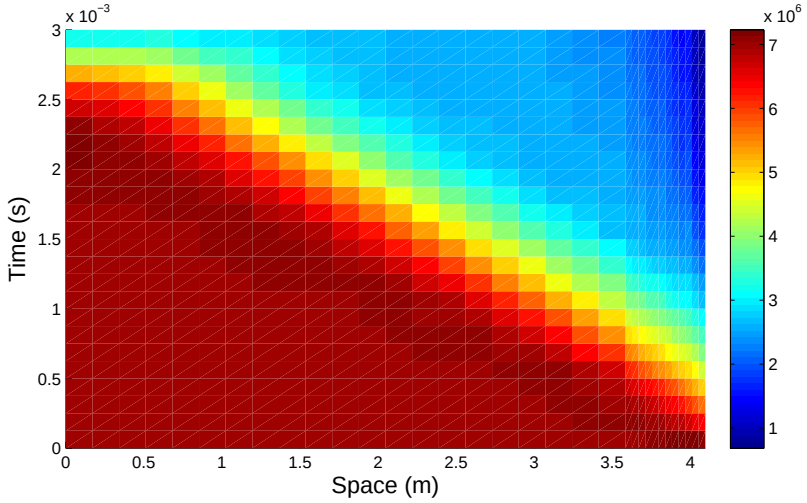


Figure 5.9: Example of solution of the truncated Edwards blowdown problem for the UQ study. Pressure is plotted.

$10^{-3}]$ seconds. This domain encompasses the initial sharp depressurization and is small enough to enable inexpensive residual evaluations. An example of the pressure solution is shown in Figure 5.9. The truncated time domain includes some strongly non-linear behavior. In addition, Figure 4.20 shows that the domain also encompasses the switch from un-choked to choked flow.

The spatial discretization is chosen to capture relevant flow features and reduce the expense of solutions and residual evaluations. The spatial domain is first divided into eight equal elements and the final element is then subdivided into four elements to resolve the

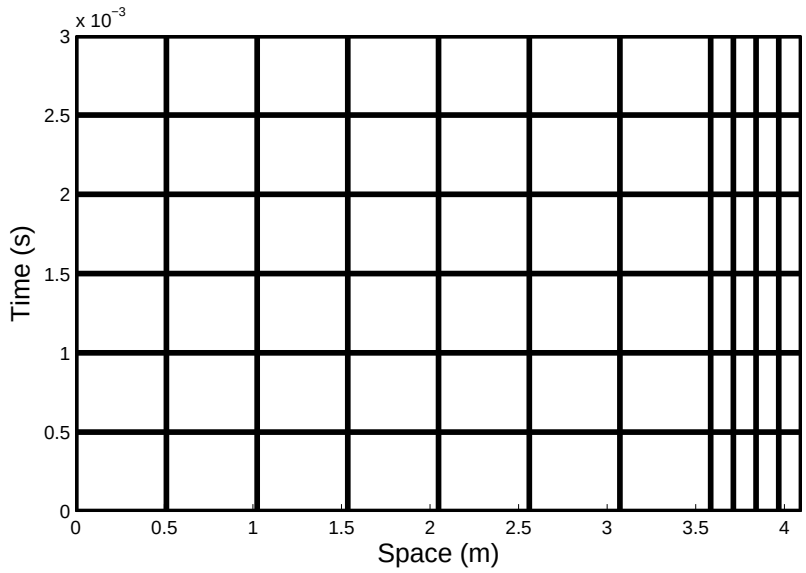


Figure 5.10: Example initial mesh for the truncated Edwards blowdown problem for the UQ study. Each box is a linear space-time element.

geometry of the outlet. The initial temporal discretization is determined by the adaptive time-stepping procedure described in Section 4.2.8. All elements are linear, $p_z = p_t = 1$. This initial grid has a resolution low enough to enable the UQ study on modest resources but high enough to capture interesting physics and converge to a solution. A residual evaluation on the truncated domain takes less than 20 seconds on a single processor compared to the full domain used in Section 4.7 (with $p_z = p_t = 2$), which takes 8.7 minutes. This savings of over an order of magnitude made the full UQ study feasible with the available computational resources.

Each simulation (at each sample point in the stochastic domain) is initialized with the same grid. The first solution is obtained at the point $\vec{x} = \mathbf{0}$ and that solution is used to initialize the other samples. If a sample does not converge from the initial guess, then the adaptive time-stepping procedure is restarted for that sample (using the same spatial grid). This results in a new temporal discretization which is usually finer in some parts than the original one. The study is initialized with 20 samples in the stochastic space from a latin-hypercube design, in addition to the point $\vec{x} = \mathbf{0}$. In this study, three of the initial samples

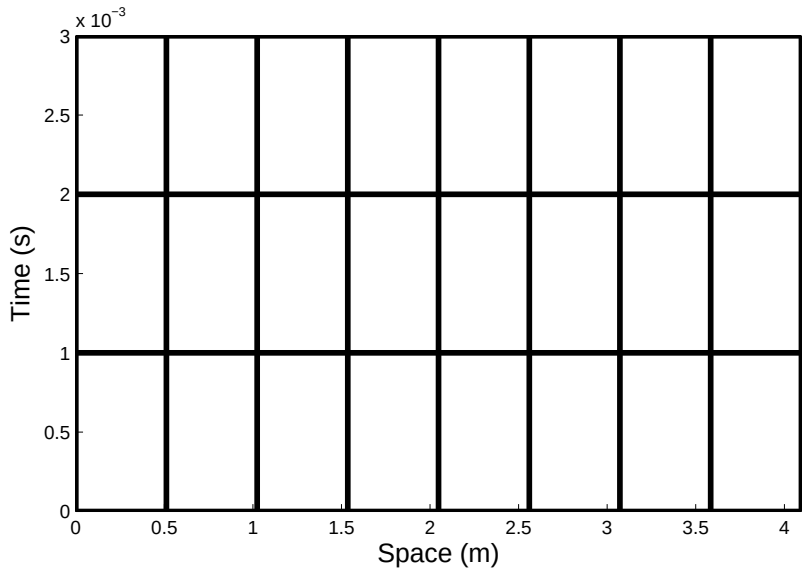


Figure 5.11: Common mesh for the truncated Edwards blowdown problem. Each box is a linear space-time element.

required restarting the adaptive time-stepping, each arriving at a finer temporal grid.

A common grid is used for comparing solutions between simulations and evaluating stochastic errors. Although the common grid should be at least as fine as the finest simulation, this was found to be prohibitively expensive for the current study. Instead, a coarsened grid is used to reduce cost. During the study the common grid was found to be sufficient to evaluate stochastic errors accurately enough to enable anisotropic adaptation, though it is too coarse to allow a convergent solution. The grid is shown in Figure 5.11. Residuals are evaluated on the common grid hundreds of times during the optimization of the directions in the stochastic model, so it is important to keep this step inexpensive.

As for the stochastic discretization, the only difference is a modified procedure for adding samples. As discussed above, 20 latin-hypercube samples are taken to initialize the adaptive process. Later, when new samples are needed, it is necessary to ensure that samples are spread out along the important directions $\vec{d}$ (not just along the original dimensions as latin-hypercube ensures). For ease of implementation, a simple strategy is developed. Suppose the stochastic approximation is adding a term along the direction $\vec{d}_i$ while adding n_{add} samples.

The procedure begins by dividing up the vector $\vec{d_i}$ into 10 divisions[6] Each current sample point is projected onto $\vec{d_i}$ and binned into the divisions. In order to spread out the new samples along $\vec{d_i}$, the next new sample is placed in the division with the fewest number of samples in it. The value within the division is a uniformly distributed random number, say r. This fixes the component of the new point, $\vec{x}_{k+1}$, along $\vec{d_i}$, so $\vec{x}_{k+1}^T \vec{d_i} = r$. The components perpendicular to $\vec{d_i}$ are set randomly. This is done by generating latin-hypercube samples and projecting them onto the complement of $\vec{d_i}$, resulting in $\vec{x}_{\perp i}$. The final sample is then $\vec{x}_{k+1} = r\vec{d_i} + \vec{x}_{\perp i}$[7]. The entire procedure is repeated for each new sample, so the divisions have approximately equal numbers of samples.

Ensuring that samples are spread out along the directions $\vec{d_i}$ is important because these directions are precisely the directions in which the solution of the simulation changes most. That is, the solution with $\vec{x}^T \vec{d_i} \approx -1$ is very different from the solution with $\vec{x}^T \vec{d_i} \approx 1$, compared to other directions in the stochastic domain. The potentially complex behavior along $\vec{d_i}$ cannot be accurately modeled without samples that are well-spread out along it. And, regular latin-hypercube sampling cannot give samples spread out along an arbitrary direction. Indeed, experience showed that the samples are generally clustered in the center of the domain. This makes sense because as the dimension increases, the relative volume in the corners of the domain increases, while latin-hypercube does not generate more samples in corners of the domain. In fact, this approach can be thought of as a special case of the orthogonal array sampling approach [108]. Orthogonal arrays are an improvement on latin-hypercube designs where the samples are evenly spread out for all r-level interactions (where r is the "strength" of the orthogonal array).

In order to reduce computational time, only least-squares fitting is used to compute $\overline{\mathbf{u}}$ (see Section 3.2.1.2). While output-based fitting (see Section 3.2.2.4) results in better fits, it is much more expensive and requires much more working memory (storing the residual and

[6]The latin-hypercube procedure also uses 10 divisions per dimension.

[7]It is possible that this point will not lie within the stochastic domain $[-1, 1]^{n_d}$. If this occurs, the procedure is attempted again with a new random number r and a different latin-hypercube sample until a point is found within the domain.

residual Jacobian for every simulation).

5.3.3 Adaptation Scheme

The combined UQ study adapts both the physical grids to reduce discretization errors and the stochastic approximation to reduce stochastic errors. The discretization error is estimated using δK_{hh} at the samples, where hh refers to a refined version of the grid associated with the sample (each sample as its own, unique, adapted grid). The discretization error is only based on the solution at the samples, so it is not contaminated by stochastic error. Separately, the stochastic error is estimated using $\delta K(\mathbf{u}(\vec{x}_k + s\vec{d}))$, which measures the error of the surrogate model $\mathbf{u}$ at points in the stochastic space *away* from the samples. The surrogate model is defined on the common grid, so the error estimate is calculated on the common grid. The error estimate could be contaminated by discretization errors if the common grid is finer than the grids at the samples[8], but this is expected to be small compared to the true stochastic error for the current example. By evaluating δK_{hh} and $\delta K(\mathbf{u}(\vec{x}_k + s\vec{d}))$, the discretization and stochastic errors are split[9]. This results in a fraction of error due to the two spaces and allocating degrees of freedom to enrich each space proportionately.

Anisotropic adaptation is enabled by computing separate error estimates for partially refined discretizations in stochastic and deterministic space, analogous to the method used for space-time adaptation in Section 4.5.1. The semi-refined error estimates are combined into an overall global fraction of error due to the stochastic and deterministic discretizations. While in Section 4.5.1 it was possible to create separate fractions for each element, there is no

[8] In addition, some discretizations, including the discontinuous Galerkin method used here, are so-called "p-dependent," meaning that the error estimate calculated on the same grid but with a different polynomial order will be non-zero (the error estimate on the same grid, with the same order is zero because $\mathbf{R}_H(\mathbf{u}_H) = 0$). Thus, if the common grid is fine or has a different polynomial order than the sample grids, some discretization error will enter into the stochastic error estimate. The magnitude of this error is expected to be small, especially for the current example where the stochastic error is in fact large. Estimating the magnitude of the contamination and its effect on the adaptation scheme is left for future work.

[9] We perform essentially just a first order error analysis, and assume that errors from the two spaces do not interact much. That is, we do not analyze how discretization errors vary over the stochastic space, or vice-versa. This is done for simplicity and produces reasonable results. For some problems, a more involved error analysis may be warranted.

way to associate error with a certain "part" of the stochastic domain. All samples contribute equally to the stochastic approximation during the fitting procedure (Section 3.2.1.2). Once the global fraction is specified, added degrees of freedom are allocated to either new samples (refinement in stochastic space) or refinement of existing simulations.

The error estimate for refinement in stochastic space is the regular direction-based error estimate developed in Section 3.2.2. First, a new direction is found by optimization. In order to reduce computational time, a modest reduction in the direction optimization is employed. The population is reduced to $n_{\text{pso,pop}} = n_d = 20$, the number of samples $n_{\text{pso,samp}}$ is fixed to 5, the number of iterations is reduced to $n_{\text{pso,maxiter}} = 50$, an additional optimization with SQP (Section 3.2.3.2) is not performed[10]. The stochastic error estimates associated with the optimal directions inside and outside of the current subspace are $\Delta J(\vec{d}^{*\Gamma})$ and $\Delta J(\vec{d}^{*\Gamma^c})$. The direction to be added to the approximation is the one with the higher error estimate (denoted $\vec{d}^*$).

Next, the error estimate for refinement in deterministic space is computed. In order to eliminate error contamination, errors are evaluated at the simulation-level on each individual grid (not on the common grid). The adjoint-weighted-residual error for the local output K is computed on a grid with a higher polynomial order ($p_z = p_t = 2$). The errors are integrated in the stochastic domain in a similar way to how ΔJ is computed. When computing $\Delta J(\vec{d})$, the error is evaluated at a sample point and along the direction $\vec{d}$. For the deterministic error, $\Delta J_{\vec{z}}(\vec{d})$, we assume that there is no change in the stochastic domain, so the deterministic error is constant along $\vec{d}$. Thus the stochastic integration is quite simple and fast. For comparison with the stochastic error, the stochastic integration is computed along the direction that will be added to the approximation. The deterministic error is denoted $\Delta J_{\vec{z}}(\vec{d}^*)$. During the stochastic error calculation, the errors are also organized into semi-refined error estimates for adaptation in space and time.

With the overall stochastic error $\Delta J(\vec{d}^*)$ and the deterministic error $\Delta J_{\vec{z}}(\vec{d}^*)$ known, an

[10] See Section 3.2.3 for a discussion of these parameters, including plots comparing values of $n_{\text{pso,samp}}$.

overall fraction of error due to the stochastic discretization is computed[11]

$$f_{\text{stoch}} = \frac{\Delta J(\vec{d}^*)}{\Delta J(\vec{d}^*) + \Delta J_{\vec{z}}(\vec{d}^*)} \tag{5.15}$$

This fraction is used to allocate new degrees of freedom to stochastic refinement (new simulations/samples) or deterministic refinement (space-time refinement of existing simulations). In order to simplify the implementation, only fixed-fraction refinement is considered. The number of samples is allowed to increase by a factor $\gamma_{\text{stoch,num}}$ at each step and the deterministic degrees of freedom are allowed to increase by a factor $\gamma_{\text{det,num}}$. More specifically, the number of samples added is

$$n_{\text{stoch,add}} = \lfloor f_{\text{stoch}} \gamma_{\text{stoch,num}} n_{\text{samp}} \rfloor \tag{5.16}$$

and the number of deterministic degrees of freedom added is

$$n_{\text{det,add}} = \lfloor (1 - f_{\text{stoch}}) \gamma_{\text{det,num}} n_{\text{det}} \rfloor \tag{5.17}$$

where n_{det} is the total number of degrees of freedom in the current simulations.

When a new sample is requested, the point in stochastic space $\vec{x}_{k+1}$ is generated by the modified latin-hypercube method described in Section 5.3.2. The spatial-temporal grid for the sample is taken from the nearest existing sample. The initial guess of the solution is taken from the stochastic approximation $\mathbf{u}(\vec{x}_{k+1})$, interpolated onto the chosen grid. This often results in good convergence, though sometimes the adaptive time-stepping procedure needs to be restarted.

When deterministic refinement is requested, the error-per-cost metric developed in Section 4.5.1 is used to determine which elements (spatial or temporal) are to be refined. All refinement options (every element from every existing simulation) are sorted based on the

[11] Note that the local output error ΔK is computed with absolute value signs, so $\Delta K = \sum |\boldsymbol{\psi} \cdot \mathbf{R}|$. See Section 4.5 for details.

error-per-cost and refinements are taken until the maximum new degrees of freedom, $n_{\text{det,add}}$, are exhausted. Note, for simplicity the overhead factor introduced in Section 5.2.2 is not used in this study, though this may somewhat reduce the efficiency of the method in terms of compute time. In addition, for simplicity, we do not compute error fractions for spatial and temporal refinement, taking

$$\epsilon^k = \sum_{l=1}^{N_{e,t}} \epsilon_{hh}^{kl} \qquad \epsilon^l = \sum_{k=1}^{N_{e,z}} \epsilon_{hh}^{kl} \tag{5.18}$$

This reduces computational expense by not computing the semi-refined error estimates for space and time.

5.3.4 Output

The local output of interest is based on the output used in Section 4.6.1. Due to the truncated time-domain, we take the fixed outlet pressure as $p_{\text{fix}} = 0.9786$ MPa, which occurs around $t = 2.18 \times 10^{-3}$ seconds for the simulation at $\mathbf{0}$. The output K is the integrated mass flow up to the point at which $p = p_{\text{fix}}$, denoted $\dot{m}_{\text{pfix}}$. Depending on the values of the parameters, the outlet pressure may never reach p_{fix}. In this case, we take K equal to the integral of the mass flow over the entire time domain. The stochastic output J, for simplicity, is taken as the average of K.

5.3.5 Results

The fully adaptive UQ method is compared with uniform refinement, Monte-Carlo, and a simple heuristic method. The methods are described in the following sections along with some results for each one.

5.3.5.1 Fully Adaptive UQ Method

The fully adaptive uncertainty quantification study, with the output of interest J, ran for 13 iterations. After these iterations, there are a total of 53 sample locations, each with grids ranging from 1,584 to 4,608 degrees of freedom, for a total of 110,688 degrees of freedom. An example of a refined grid is shown in Figure 5.12. For this particular simulation, the pressure reached p_{fix} at $t_{\text{pfix}} = 1.07 \times 10^{-3}$s. The refinement is clearly focused around this time and at the end of the pipe in space. Figure 5.13 shows the gas-mass component of the adjoint which is active in the same region. Since the adjoint is concentrated around a small region in space-time, an adaptive unstructured discretization would perform even better than the current structured discretization. The structured discretization requires refinement of a time element for all of space, resulting in possibly unneeded refinement for $x \lesssim 3.5$.

Of the 20 dimensions, the first nine corresponded to parameters that were varied. Of those, only the first six produced measurable changes in the solution (the rest were removed from the approximation, see Section 3.2.4). The stochastic approximation produced by the method is active in a two-dimensional subspace ($\dim(\Gamma) = 2$). While running the adaptive studies, it was found that increasing the number of optimization iterations often leads to a larger active subspace (the method is more likely to expand the active subspace). Quantifying this effect is left for future work. A larger number of optimization iterations would likely have led to a larger active subspace, though it is likely that the error behavior would be similar because most of the accuracy would be focused on the important parameters.

In this particular study, the 2D active subspace was found to represent the behavior of the solution well. A basis of the active subspace is shown in Table 5.2. The parameters ϵ/r_{pipe}, A_{block}, $h_{l,\text{init}}$, and τ_{pout} were found to be most important (the basis vectors have large components in these dimensions). The friction parameter, when large, restricts the flow velocity and flow rate, having a large impact on the mass flow $\dot{m}_{\text{pfix}}$. The blockage area controls what the choked mass flow rate is, also heavily impacting the net mass flow. The initial enthalpy affects how quickly the water flashes. Higher void fraction effectively chokes

Table 5.2: Basis vectors for the 2D active subspace Γ discovered by the fully adaptive UQ method.

x_0	x_1	x_2	x_3	x_4	x_5	x_6 thru x_{19}
ϵ/r_{pipe}	A_{block}	z_{block}	$h_{l,\text{init}}$	τ_{pout}	p_{init}	-
0.0512	0.5517	-0.2573	0.5881	0.5296	0.0234	0
0.5247	0.5069	0.2988	0.1144	-0.5696	0.2023	0

the liquid flow, reducing the net mass flow. The outlet pressure parameter τ_{pout} strongly affects how quickly the depressurization takes place. This has a strong affect on the net mass flow when the output switches to the total mass flow over the entire time period (when the pressure never reaches p_{fix}). Otherwise, there is less of an effect because a delay in depressurization just delays the location of $p = p_{\text{fix}}$, but does not necessarily alter the mass flow up to that point. The blockage location z_{block} does not have a strong effect because the shape of the pipe is not important compared to the area ratios [12]. The initial pressure does not have a strong effect because the variation in p_{init} (1 MPa) is small compared to the overall depressurization (e.g. a drop of 6.9 MPa). Interestingly, the gravitational acceleration, Prandtl number, and gas enthalpy parameter had essentially no effect on the simulation at all.

The current method shows that modeling variation along just two combinations of the four important parameters is sufficient to describe much of the variation of the output. In a sense, the current method has reduced the dimensionality of the problem from 20 to just two.

The method uses a relatively high order approximation within the two-dimensional active subspace. The importance of a high order approximation can be seen by comparing the pressure and adjoint values for two sample points at either end of the direction $\vec{d}_1$ (the first direction in Table 5.2). Figure 5.14 shows the pressure and Figure 5.15 shows the adjoint

[12] This is especially true since the flashing model assumes equilibrium between the phases. In a non-equilibrium model, the flashing is dependent on the time it takes the flow to accelerate (proportional to z_{block}) compared to the time it takes bubbles to form.

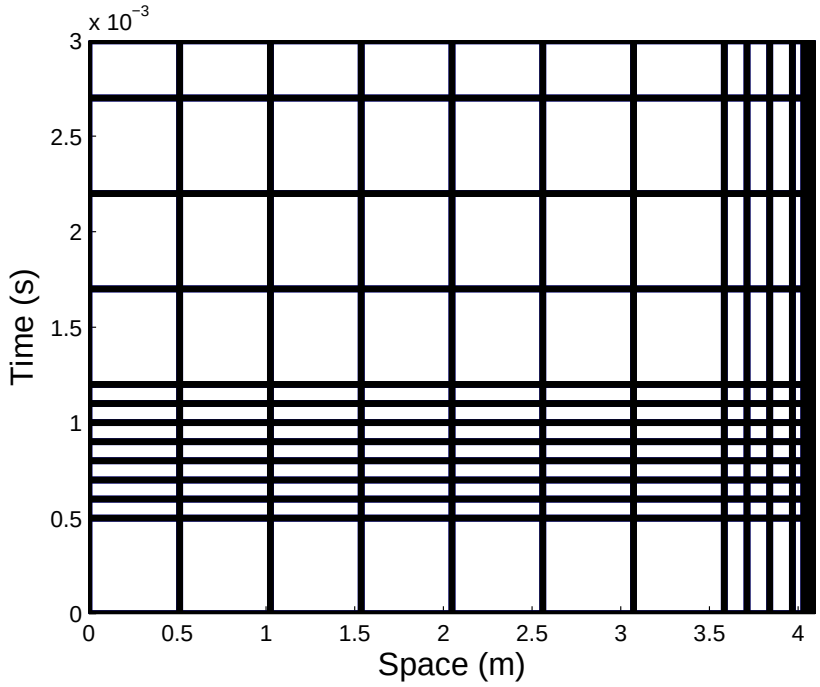


Figure 5.12: An example of a refined mesh from the UQ study. Each box is a linear space-time element.

for points with $\vec{x}^T \vec{d}_1 = -1.43$ (left plot) and 1.51 (right plot). The stochastic approximation must smoothly interpolate between these two different solutions. Some points in the domain, when viewed "along" $\vec{d}_1$, show a high-order variation. For example, the pressure drop shifts up between the two samples. Thus, a point above the pressure drop on the left plot (e.g. $x = 3.5$m, $t = 10^{-3}$s) is below the pressure drop on the right plot. Even more complicated behavior can be seen in the adjoint (note the difference in the scales of the plots). Thus, the stochastic approximation focuses on building the order of the approximation (up to $p = 4$).

The range of solutions found in the samples is large but does not necessarily encompass the experimental data. Figure 5.16 shows the maximum and minimum pressure profiles found in the samples for the outlet and the closed end of the pipe. At the outlet, the range of the prescribed exponential decrease in pressure is clearly visible[13]. The range of simulations

[13]The "baseline" solution here is taken as the parameter settings used to first solve the blowdown problem in Chapter 4. The high and low limits (dotted lines) are taken from the solutions at the samples evaluated on the common grid. The differences in grids between the high/low solutions and the baseline solution accounts for the baseline solution being slightly above the high limit for outlet pressure at $t \lesssim 0.25$ms.

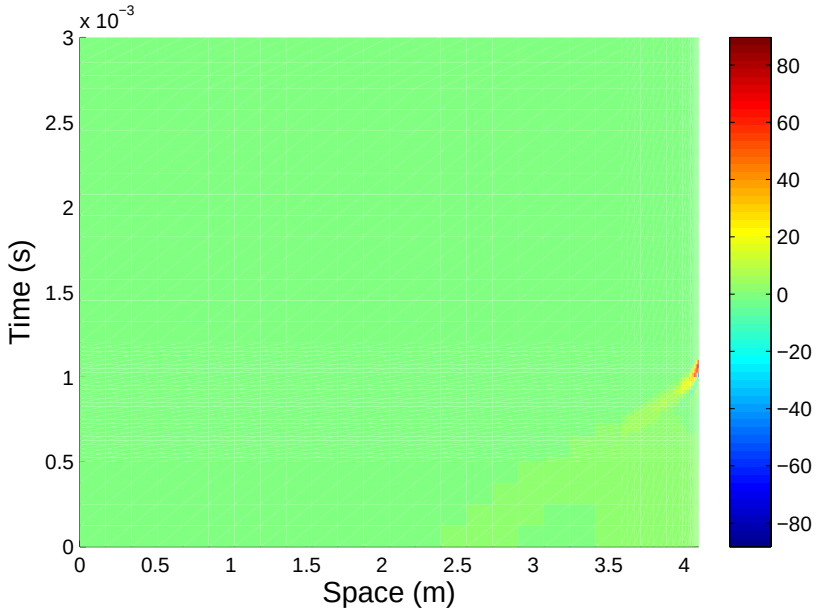


Figure 5.13: An example of the gas-mass component of the adjoint from the UQ study. The output is the integrated mass flow up to $t = t_{\text{pfix}} = 1.07 \times 10^{-3}$.

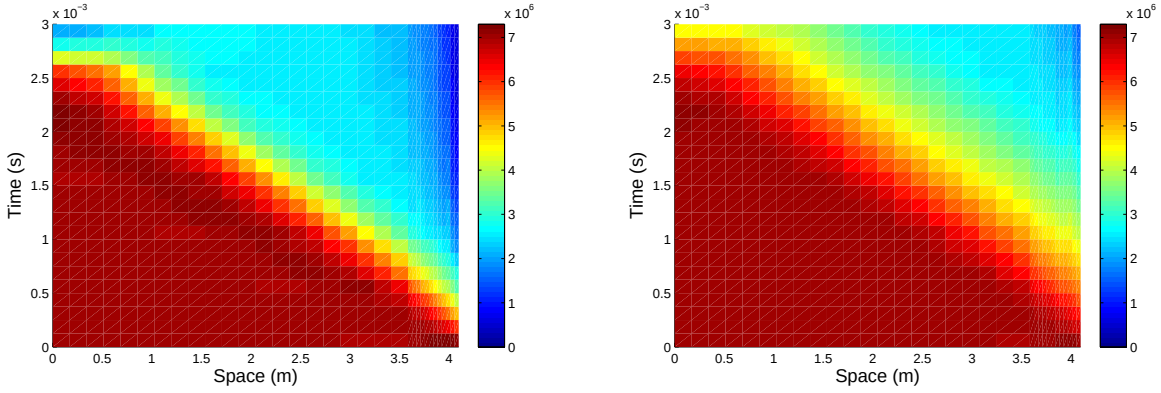


Figure 5.14: Two examples of the solution for sample points with large and small components along $\vec{d}_1$. Pressure is plotted.

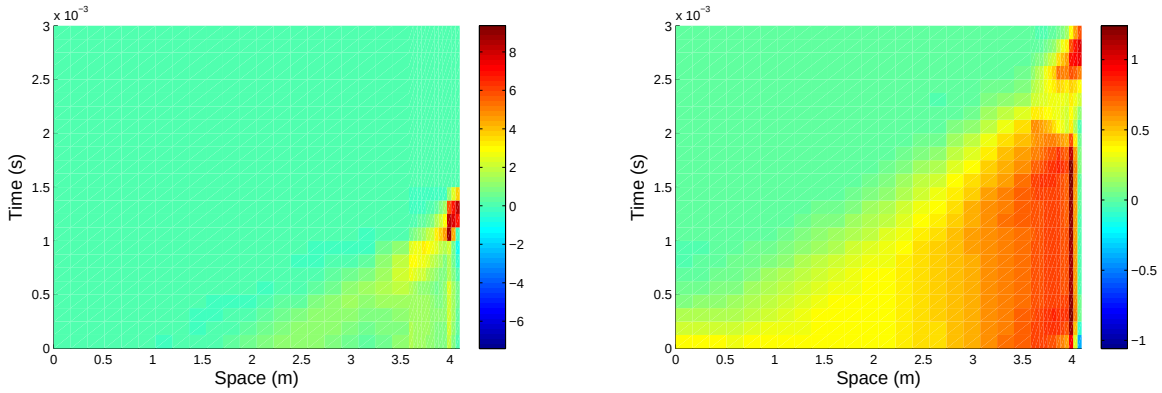


Figure 5.15: Two examples of the adjoint for sample points with large and small components along $\vec{d}_1$. The gas-mass component of the adjoint is plotted.

nearly encloses the experimental data. At the closed end, the simulations do not enclose the experimental data, which has the pressure wave encountering the closed end around $t = 3.2$ms. The fact that no values of the parameters can correctly predict this behavior suggests two possibilities. First, there may be additional parameters (or wider ranges) that, when added to the UQ study, could result in behavior more consistent with the experiment. Second, the physical models may be too simplistic to produce more consistent behavior. The current UQ study is meant to demonstrate the feasibility and cost savings of combined adaptivity, so a full investigation comparing the simulations with experimental data is left for future work.

The values of the stochastic error fraction, f_{stoch}, serves as an indication of how well the method is doing at balancing stochastic and deterministic errors. An error fraction of $f_{\text{stoch}} = 0.5$ indicates well balanced errors so that refinement in both spaces results in an improved solution. The error fraction is plotted in Figure 5.17 and shows good behavior. For the first nine iterations or so, the error is more concentrated in the deterministic space, and more degrees of freedom are allocated to grid refinement (see, e.g., Figure 5.12). At this point, the (relative) stochastic error grows and more samples are added (lower plot). Since the error is not highly concentrated in either space, this plot indicates that the fixed-fraction growth is sufficient for this study.

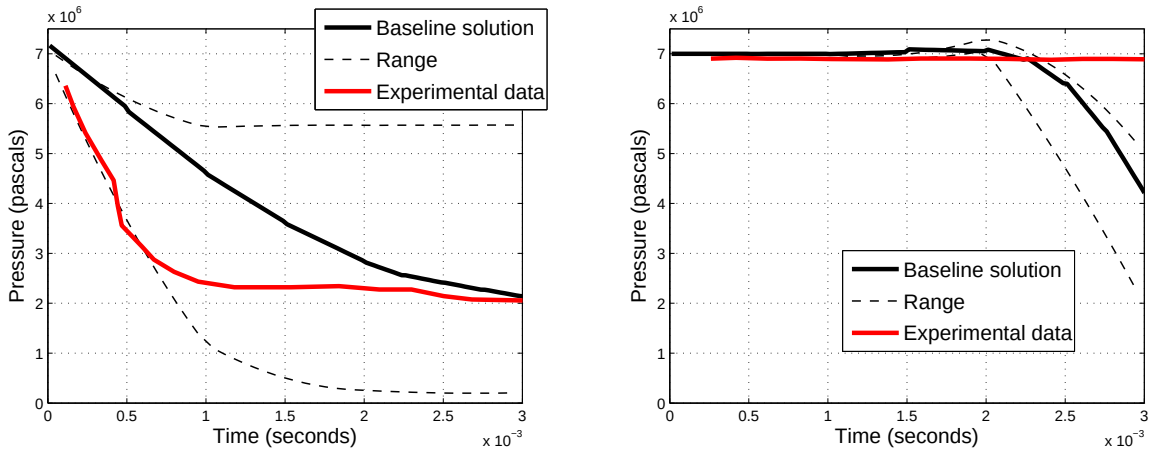


Figure 5.16: Baseline solution and range of the samples from the uncertainty quantification study compared to the experimental data. Left, pressure at the outlet. Right, pressure at the closed end.

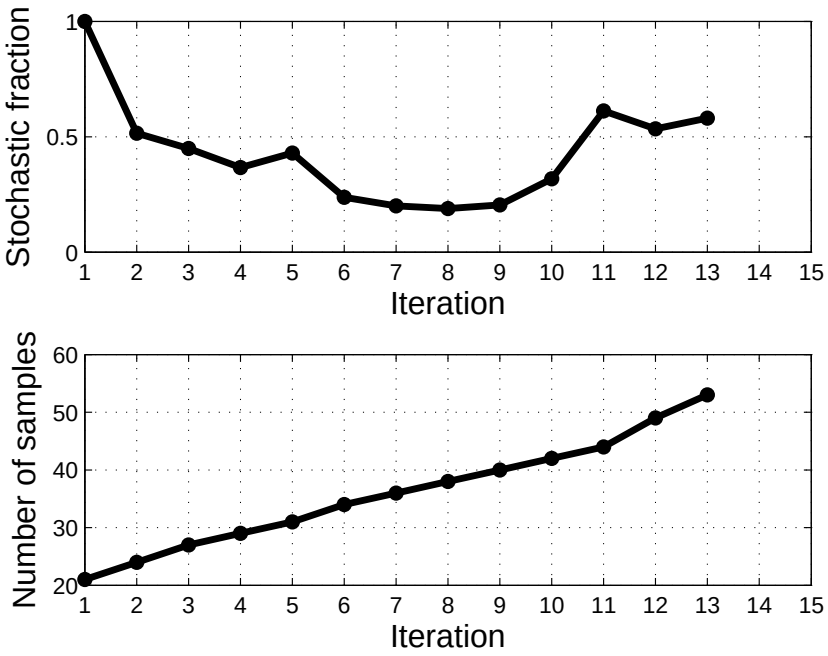


Figure 5.17: The stochastic fraction of the error f_{stoch} (upper plot) and the number of samples (lower plot) for the UQ study.

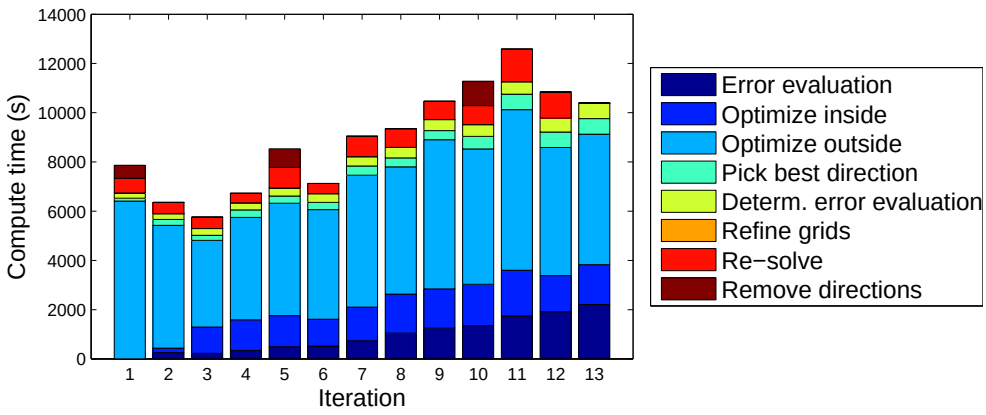


Figure 5.18: Timing data for the fully adaptive UQ study. Each bar represents one iteration, with phases of the iteration in separate colors. The run shown here used 40 processors in parallel. Note that 10,000s≈2.8 hours.

The timing data from the fully adaptive UQ method is shown in Figure 5.18. The code was run in parallel on 40 processes (Intel® Xeon® E5), each with up to 4Gb of memory[14]. Clearly, finding the optimal direction $\vec{d}^*$ is the most time-consuming part of the calculation, requiring hundreds of residual evaluations on the common grid. For the first three iterations, the size of the active subspace Γ grows. Since the time taken to optimize the direction is proportional to the size of the space being explored, a larger active subspace shifts the computational burden from optimizing $\vec{d}^{*,\Gamma^c}$ (outside) to $\vec{d}^{*,\Gamma}$ (inside). Thereafter, the burden remains approximately evenly distributed because the subspaces remain the same with $\dim(\Gamma) = 2$ and $\dim(\Gamma^c) = 4$. It is noteworthy that re-solving, which includes adding new samples (new simulations) and re-solving existing ones with refined grids, takes relatively little time. The adaption process could probably be made more efficient by being more aggressive (i.e. increasing $\gamma_{\text{det,num}}$ and $\gamma_{\text{stoch,num}}$).

[14]It was found that a process could only store about 500Mb of raw data due to memory overhead.

5.3.5.2 Uniform Refinement

In one step of uniform refinement, the deterministic grids are all fully refined. The stochastic approximation is initially constant everywhere with 21 samples (Note, $n_d = 20$). The stochastic approximation is enriched by doubling the number of samples and by incrementing the order of the approximation. The approximation includes all possible terms up to the specified order (for example, 20 directions for a linear approximation). Due to memory constraints, it was only possible to run one step of uniform refinement.

5.3.5.3 Monte-Carlo Method

For the Monte-Carlo method, we assume that the deterministic grid is "pre-converged" at the nominal point $\vec{x} = \mathbf{0}$, and the resulting grid is used for all samples. To find the grid, a grid-convergence study was performed. The output for three uniform refinements is shown in Figure 5.19. The output from the second grid was deemed suitably converged, with a relative difference of 1.2×10^{-4} compared to the output from the third grid. The Monte-Carlo method could be considered converged if the stochastic error is less than 10^{-4}, since at that point further Monte-Carlo samples cannot increase accuracy beyond the deterministic errors. The stochastic output J is easily computed as the average of the sample outputs K (sample mean). The convergence of the Monte-Carlo method, compared to the best guess it returns, is shown in Figure 5.20. The convergence is "noisy" because it is only a single Monte-Carlo run (ideally one would replicate the run multiple times and average the results). Overall, there were 95 samples with a total of 164,736 degrees of freedom. The convergence behavior suggests that the stochastic error is still large compared to the deterministic error, and more samples are required to get a good estimate of J.

5.3.5.4 Heuristic Method

For the heuristic method, we consider interpolation error which is faster to calculate than the adjoint-based error estimate. Separate deterministic and stochastic error estimates are

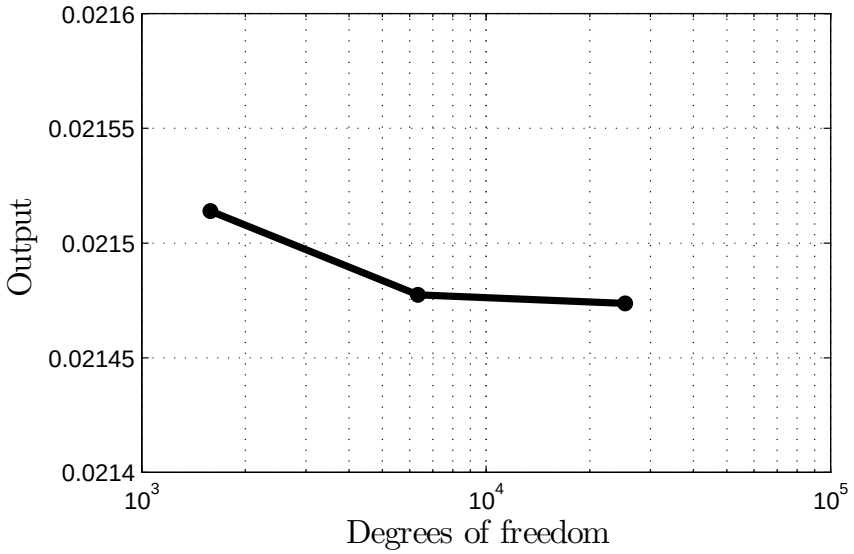


Figure 5.19: Mesh convergence study for the Monte Carlo method. Three grids are shown, each a uniform refinement of the previous. The stochastic point under consideration is $\vec{x} = \mathbf{0}$.

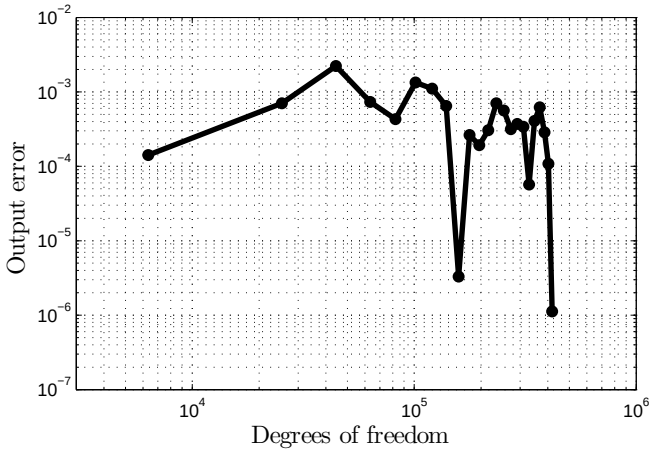


Figure 5.20: Convergence of the Monte Carlo method. Error in the stochastic output is shown, compared to the best value computed by the largest Monte-Carlo run. Each point represents one additional sample, and the total number of samples is 95 with a total of 164,736 degrees of freedom.

compared. The approximation is refined in either deterministic space if the deterministic error is higher and vice versa. The deterministic interpolation error is computed on a grid with a higher polynomial order (denoted h). The original solution $\mathbf{u}_H$ is injected and interpolated onto the higher-order grid (see Section 4.4), resulting in $\mathcal{I}_h^H \mathbf{u}_H$ and $\mathbf{u}_h$, respectively. That is, the solution is transferred to a fine grid (h) first without alteration (i.e. $\mathcal{I}_h^H \mathbf{u}_H$ is just a different representation of the exact same function $\mathbf{u}_H$) and second with smoothing ($\mathbf{u}_h$). The interpolation error is approximated as the difference between the unaltered and smoothed versions. For the i^{th} sample, the deterministic interpolation error is

$$\epsilon_i^{\text{det}} = \frac{||\mathcal{I}_h^H \mathbf{u}_H - \mathbf{u}_h||_2}{||\mathbf{u}_H||_2} \tag{5.19}$$

Note, since each state is not directly comparable, the interpolation error is computed separately for each state ($\rho, (\alpha\rho_g)$, etc) and the maximum value among the states is taken as ϵ_i^{det}. Finally, the overall deterministic error is the average among all samples, $\epsilon^{\text{det}} = 1/n_{\text{samp}} \sum_i \epsilon_i^{\text{det}}$.

The stochastic interpolation error uses the mean-squared-error of the output at the sample points. It is defined as

$$\epsilon^{\text{stoch}} = \left\{ \sum_i \left[K(u(\vec{x}_i)) - K(u_{\text{true}}(\vec{x}_i)) \right]^2 \right\}^{1/2} \tag{5.20}$$

Here, $u(\vec{x}_i)$ is the approximation of the solution at the sample point $\vec{x}_i$, evaluated on the common grid, and $u_{\text{true}}(\vec{x}_i)$ is the actual solution at the sample point on the sample's grid. Note that since the grids are different, this stochastic error estimate is somewhat "contaminated" by deterministic error. The heuristic method is meant to be simple, though, and this form is used because it is easy to compute.

In the heuristic method, refinement of the deterministic space is a simple uniform refinement of all deterministic grids. Refinement of the stochastic space is accomplished by increasing the order of the approximation and adding as many directions as possible within

the active subspace (resulting in a full-order approximation of the active subspace, similar to Russi's method [100]). The number of samples is increased to satisfy a heuristic for the appropriate number of samples,

$$n_{\text{samp}} = \max\left(2(n_{\text{term}} - 1), n_d + 1\right) \tag{5.21}$$

This adds two samples for each new direction (beyond the initial $\vec{d}_0$ with $p_0 = 0$). Note that the approximation requires $n_{\text{samp}} > n_{\text{term}}$ to be able to solve for $\overline{\mathbf{u}}$.

For this problem, the heuristic method resulted in only deterministic refinement because ϵ^{det} was larger than ϵ^{stoch} at each step. Only three steps of the method were possible before memory constraints were encountered. For this study, the stochastic error heuristic is too simplistic to reveal the underlying stochastic errors. For example, none of the sample points have a large value of $\vec{x}^T\vec{d}_1$, so the "corners" of the domain where $\vec{d}_1$ is large[15] are not explored, though that is where the largest errors are. Without sufficient exploration inherent in the error calculation, the heuristic cannot discover poorly modeled parts of the stochastic domain. The advantage of the fully adaptive UQ method is an error metric that does a better job exploring the large stochastic domain, though it is more expensive.

5.3.5.5 Comparison of Methods

The four UQ methods are compared in terms of error in the stochastic output J. The value of J for a given stochastic approximation is expensive to compute since it requires a stochastic domain integral[16]. This integral is approximated by taking a sample average of K from a set of sample points. The set of sample points consists of two sub-sets: the samples

[15] In this study, $\vec{d}_1$ has large components along x_2,x_4, and x_5, corresponding to the parameters A_{block}, h_l, and τ_{pout}. Each of these parameters are important, so there is large variation due to the combined effects of the parameters. Further, the fully adaptive UQ results show high-order variation along $\vec{d}_1$, suggesting interaction effects beyond the simple additive main effects of each parameter. Thus, it is important to have samples at points $\vec{x}$ such that $\vec{x}^T\vec{d}_1$ is large and small to capture the interaction effects.

[16] The solution and adjoint, $\mathbf{u}$ and $\boldsymbol{\psi}$, are modeled in the stochastic domain, but the output K is not. Thus, a full integral in stochastic space requires evaluating $\mathbf{u}$ at integration points in the six-dimensional active subspace. The analytic method in Section 3.2.6.1 cannot be used since $K(\mathbf{u})$ is not a linear functional.

at which a full simulation is available[17], and new sample points added along the relevant directions[18] as discussed in Section 5.3.2. The total number of samples used for the output calculation, n_J, is heuristically set to $n_J = 4n_{\text{samp}}$[19] For the heuristic method, no directions were added to the approximation, so the extra samples are simply latin-hypercube samples. The same is true for uniform refinement, since for the single step taken here, the added directions are simply $\vec{e}_0$ through $\vec{e}_5$. For the Monte-Carlo method, we just take the sample average of the Monte-Carlo samples since there is no implicit stochastic approximation of $\mathbf{u}$ to use.

The four methods are compared to a "true" value, J_{true}, computed on the finest discretization available. All of the samples computed by all of the methods were combined into a single approximation (duplicate sample points were removed). The stochastic approximation is taken as the finest approximation from the fully adaptive UQ method with additional directions to create a full fourth order approximation of the 2D active subspace, plus four additional linear directions to ensure a good linear representation of the six active parameters. The value of J from this approximation, also computed using n_J total samples, was considered to be J_{true}.

Figure 5.22 shows the output error for the four methods. The Monte-Carlo method seems to be converging to a different value than the other three. This is likely due to the fact that classic latin-hypercube samples are not good at sampling the "corners" of a high-dimensional domain [108]. While more advanced sampling methods try to sample these corners, one still encounters the problem that there are so many corners in a hypercube that a huge number of samples is still needed to explore them all. To see this, one can look at the component of Monte-Carlo samples $\vec{x}$ along the important directions discovered by the fully adaptive UQ method, for example $\vec{d}_1$. For the 95 Monte-Carlo samples, the value $\vec{x}^T\vec{d}_1$ ranges between

[17]Since a full simulation is available, the deterministic output K is calculated from the full solution at this point.

[18]The value of K at these new sample points is computed by evaluating the stochastic approximation of $\mathbf{u}$ at this point and computing the output. This is done on the common grid

[19]The results are qualitatively the same for $n_J \gtrsim 2.5n_{\text{samp}}$. The parameter may be problem dependent.

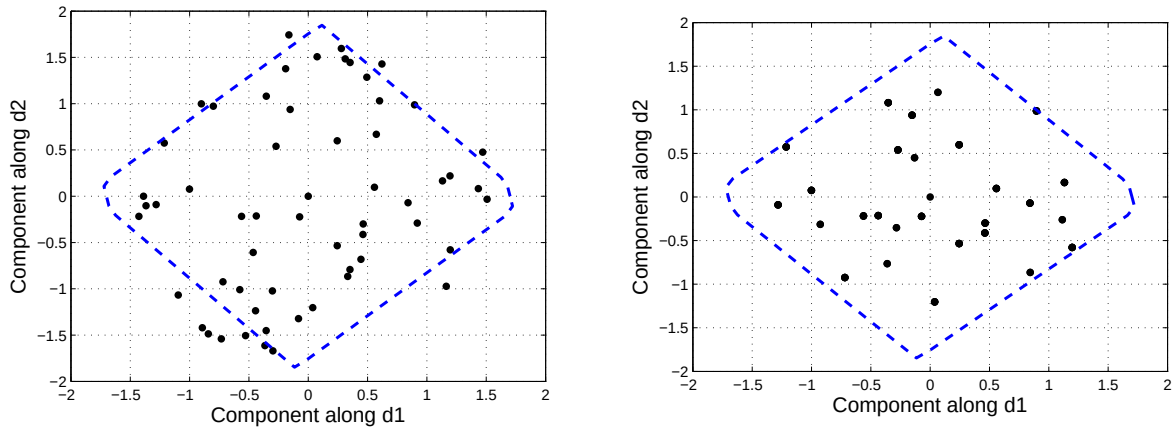


Figure 5.21: Components of samples along basis vectors $\vec{d}_1$ and $\vec{d}_2$ of the active subspace Γ. Left, samples from the fully adaptive UQ method. Right, samples from Monte-Carlo. The blue outline approximates the maximum values of $\vec{x}^T\vec{d}$.

-1.20 and 1.28. The fully adaptive UQ method, in contrast, has samples with $\vec{x}^T\vec{d}_1$ ranging from -1.43 to 1.51. The samples from both methods, in terms of their components along $\vec{d}_1$ and $\vec{d}_2$, are shown in Figure 5.21. The fully adaptive UQ method does a better job at exploring the far corners of the domain[20].

The heuristic method generates successively better approximations of J, but the improvement is slow compared to the number of degrees of freedom it uses. Again, the stochastic error heuristic does not take into account the corners of the domain and therefore focuses on deterministic errors.

The current method, while somewhat "noisy", does reduce the error in the output and does so more efficiently than uniform refinement. The compact active subspace reduces the number of samples needed, resulting in an efficient method in terms of error for a given number of degrees of freedom. For about the same computational time, the current method results in approximately an order of magnitude lower error and an order of magnitude fewer degrees of freedom compared to the other methods.

[20]The blue dotted outline is the outline of the domain ($[-1,1]^{n_d}$) in the plane defined by $\vec{d}_1$ and $\vec{d}_2$. The full projection of the hypercube onto this plane, though, is a larger outline, hence some of the points lie outside of the blue line.

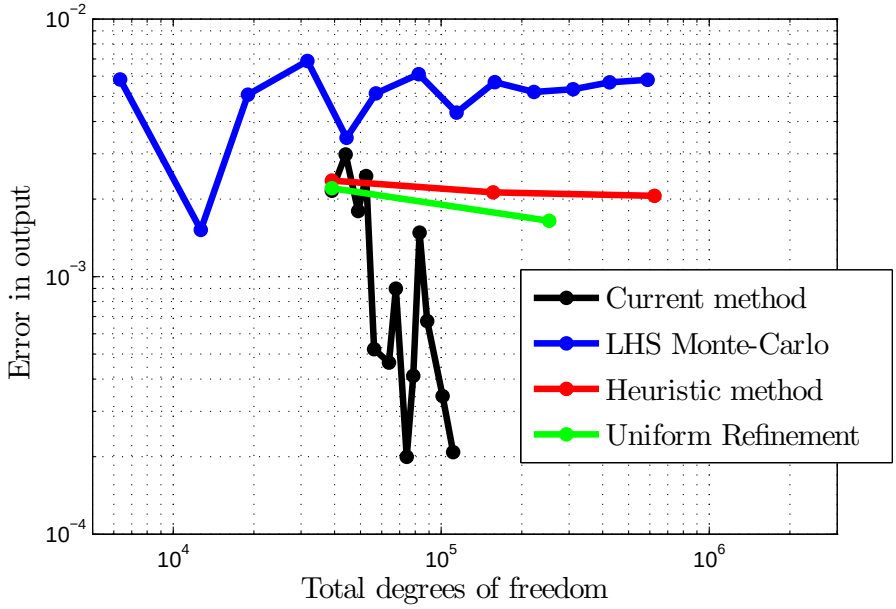


Figure 5.22: Comparison of convergence of the four UQ methods.

CHAPTER 6

Conclusions

In practice, most simulations are run without quantifying errors from the discretization or parameters. Often convergence is judged by how smooth a solution looks and parameter dependence is characterized by vague percentages. With the increasing use of simulations for important engineering decisions, accurate predictions are needed and errors must be quantified. Simple methods for quantifying and reducing errors, as has been shown, can lead to good results. These include grid convergence studies and parameter sensitivity studies. However, these simple methods often result in overly expensive calculations and slow convergence.

More advanced methods tailor the discretization (in both the deterministic and stochastic spaces) to the problem at hand. This can, in some cases, significantly reduce the computational burden. The process also naturally produces intermediate results at slowly increasing levels of fidelity, allowing discretizations that are tailored to the resources available.

In the deterministic space, grids that have low error are often created by experienced users and require much human input for each new problem. In addition, some features in the solution may not be visible or deemed relevant at the outset, resulting in costly iterations between grid generation and solution. The grid adaptation method presented here automates this procedure at a fine-grained scale, allowing highly tailored grids to be generated for any given problem. In addition, discretization error is quantified at each step,

resulting in grids with a known accuracy. The cost is essentially writing the code needed to generate refined grids, compare solutions between grids, and possibly compute derivatives. Such an investment can be costly, but these features are already available in many simulation codes and are useful for more than just adaptive UQ studies.

In the stochastic space, the biggest obstacle is the curse of dimensionality. That is, the stochastic space that needs to be explored is typically vast, requiring a huge number of costly simulations for simple methods. Given the large space, it is understandably difficult to model behavior in that space with limited sampling. However, in some cases there is a small "inherent" dimensionality to the stochastic space. That is, only a few parameters (or combinations of them) have a strong effect on the simulation. Some advanced uncertainty quantification (UQ) methods search for the inherent dimensionality with good success. A second problem, though, is that the functional model of the behavior in stochastic space is often very complex, requiring many samples just to fit the model. This often occurs when high-level interaction effects are considered, since there are exponentially many of them. The stochastic approximation used in this work is very compact since it groups interaction effects together into a small number of active directions. These directions span an active subspace, which is a small space compared to the full stochastic space. In this way, the stochastic approximation reduces the number of samples required for high accuracy. Finally, the stochastic approximation is built up adaptively, like the deterministic grid, resulting in a sequence of approximations with increasing accuracy and cost.

A third problem with conventional methods for error quantification is that the methods used for the deterministic and stochastic spaces are often not consistent with each other. For example, one might use interpolation error in deterministic space and least-squares fitting error in stochastic space. While these two measures seem similar, in fact their scales are not necessarily consistent. In addition, anisotropic adaptation (deciding which space to refine more) requires error estimates that are well separated for each space. This is not trivial and many error estimates conflate errors from the two spaces. In this work, consistent error

metrics are developed by targeting them to a single, scalar stochastic output of interest. Any error estimate is then simply the effect that a certain grid size or sample has on that output, so all errors are comparable to each other. In addition, care has been taken to isolate errors from each space by using semi-refined error estimates. This results in a successful splitting up of refinement between the two spaces.

The new uncertainty quantification method developed here produced good results for test functions with small inherent dimensionality. While other methods loose efficiency when other factors change, for example when high-level interactions are included or when the behavior is anisotropic, the current method still gave good results. This is because the method takes advantage of low-cost error estimates and tools from the field of optimization to explore the stochastic space efficiently. This reduces the space that is modeled, resulting in higher accuracy with fewer samples.

A few directions for future work in the UQ method are as follows. First, more realistic problems require more complex outputs (i.e. $J(u)$) based on statistical measures. Second, a more thorough study of the effects of the optimization parameters (number of iterations, population size, number of samples, etc) on the resulting stochastic approximation and size of the active subspace is warranted. Third, the method could be combined with parameter optimization for solving the optimization under uncertainty problem.

In this work, a one-dimensional multiphase model was solved with the discontinuous Galerkin method. This required developing smoothed versions of some relations and special boundary conditions for convergence and adjoint consistency. The solution was of good quality and a smooth adjoint solution was found, which is rare for multiphase flow problems. In addition, the anisotropic adaptation resulted in good error reduction and targeting time intervals that strongly affect the output. The adapted grid was quite specialized to the output; error in the output was far smaller than for a large, uniformly refined grid. Some of the locations where the method generated high-fidelity were not initially obvious as requiring it. It is unlikely that a human would have been able to generate a grid with as good an error

to cost ratio. The results promise to be even more exaggerated for 2D and 3D simulations with unstructured grids. In such complex multiphase flows, features can be highly localized so a well tailored grid can be very efficient. In addition, outputs as complex as the one used in this work are rarely encountered as targets for adaptation, especially in the multiphase context. The output, though, is important for engineers and good results were found for the adjoint and error convergence with adaptation.

Some directions for further work in this area are as follows. First, applying the same adaptive methods to more complex (e.g. full two-fluid), 2D and 3D simulations should reveal even more savings. Further, multi-scale multiphase simulations would make the physics adaptable as well, possibly further increasing accuracy while not reducing efficiency. In such extensions, one must always be aware of extra code needed for derivatives (automatic differentiation can help here) and possible problems with adjoint consistency (though these can, to some extent, be smoothed out after the fact). Second, it is unclear why the flow at the outlet of the pipe remains sonic for a finite time before becoming supersonic. This may be an interesting problem for theoretical two-phase flow.

An uncertainty quantification study was performed that was adaptive in both the deterministic space (by adapting the physical grids) and the stochastic space (by choosing new sample points). This was done while balancing deterministic and stochastic errors. Such a study has rarely been carried out but promises a substantially reduced computational burden and increased the accuracy. The adaptive method discovered parts of the physical domain where increased accuracy was necessary (near the time and location where the pressure reached its critical value) and parts of the stochastic domain (the interaction between four of the parameters). It built compact deterministic and stochastic models of the solution, requiring fewer overall degrees of freedom and resulting in lower error than other methods. And, this was accomplished with some modifications that sped up the computation, despite some loss in accuracy.

There is much room for further work in the area of combined adaptivity in deterministic

and stochastic spaces. First, how coarse or fine should the common grid be? Could there be different common grids for different parts of the domain or different calculations (e.g. multi-fidelity optimization of the direction)? Second, how aggressively should degrees of freedom be added? Can a fixed-error type of growth be devised? How fine should the physical grid of new samples be? Third, could one use separate approximations for the solution and adjoint? Should the direction optimization be separate? Would it be useful to build an approximation of the error or the output in stochastic space as well? There are many more questions to ask. For now, we have shown that the potential exists for cheaper, more accurate UQ studies.

Appendix A: Chexal-Lellouche Drift Velocity Model

This model computes the a weighted drift velocity, $\langle\langle V_{dj}\rangle\rangle$, and the distribution parameter C_0. The model can be found in various sources including [80].

$$
\begin{aligned}
C_0 &= \frac{L}{K_0 + (1-K_0)\alpha^r} \\
\langle\langle V_{dj}\rangle\rangle &= 1.41\left(\frac{g_z\sigma\Delta\rho}{\rho_l^2}\right)^{1/4} C_2C_3C_4C_9 \\
L &= \frac{1-e^{-C_1\alpha}}{1-e^{-C_1}} \qquad C_1 = \frac{4p_{crit}^2}{p(p_{crit}-p)} \\
K_0 &= B_1 + (1-B_1)\left(\frac{\rho_g}{\rho_l}\right)^{1/4} \qquad r = \frac{1.0 + 1.57\rho_g/\rho_l}{1-B_1} \\
B_1 &= \min\left(0.8, \frac{1}{1+e^{-\mathrm{Re}/6000}}\right), \qquad \mathrm{Re} = \max\left(\mathrm{Re}_l, \mathrm{Re}_g\right) \\
C_2 &= \begin{cases} 0.4757\left(\ln\left(\dfrac{\rho_l}{\rho_g}\right)\right)^{0.7}, & \dfrac{\rho_l}{\rho_g} \le 18 \\ \begin{cases} 1, & C_5 \ge 1 \\ \left(1-e^{C_5/(1-C_5)}\right)^{-1}, & C_5 < 1 \end{cases} & \dfrac{\rho_l}{\rho_g} > 18 \end{cases} \\
C_3 &= \max\left(0.5, 2e^{-|\mathrm{Re}|/60000}\right) \qquad (6.1) \\
C_4 &= \begin{cases} 1 & C_7 \ge 1 \\ \left(1-e^{-C_8}\right)^{-1} & C_7 < 1 \end{cases} \qquad C_8 = \frac{C_7}{1-C_7} \\
C_5 &= \sqrt{150\frac{\rho_g}{\rho_l}} \qquad C_7 = \left(\frac{0.09144}{2r_{\mathrm{pipe}}}\right)^{0.6} \qquad C_9 = (1-\alpha)^{B_1} \qquad (6.2)
\end{aligned}
$$

σ is the surface tension of water, here taken as constant at the reference condition $\sigma_{ref} = 0.02276$ N/m. p_{crit} is the critical pressure of water, 22.0640 MPa.

From Ishii [57], the relation

$$V_{dj} = \langle\langle V_{dj} \rangle\rangle \left[1 - (C_0 - 1)\left(v_m + \frac{\alpha\,(\rho_c - \rho_d)}{\rho_m}\right)\right] \tag{6.3}$$

is used to form the drift velocity V_{dj} which appears in the governing equations. This relation can be derived from the following definitions

$$V_{dj} = \langle\langle V_{dj} \rangle\rangle + (C_0 - 1)\, j \tag{6.4}$$

$$j = v_m + \frac{\alpha\,(\rho_c - \rho_d)}{\rho_m} V_{dj} \tag{6.5}$$

The following modifications were made to ease implementation and correct for limiting cases.

- C_2 is computed using the reference densities.
- The Reynolds number is computed using mixture quantities, rather than the maximum of the two phases. It is also capped at 10^{10} so that the model is valid in the inviscid limit.
- Computation of C_9 and L must be corrected for the limits $\alpha = 1$ and $\alpha = 0$, where $C_9 = 0$ and $L = 1$ respectively.
- The derivative of C_0 with respect to r is set to zero for $\alpha = 0$ since it is undefined there.

Appendix B: A Survey of Multiphase Models

B.1 Appendix Nomenclature

$()_g, ()_v, ()_d$	gas, dispersed phase
$()_l, ()_f, ()_c$	liquid, continuous phase
$()_r$	relative (between phases), gas - liquid
$()_i$	interface value
d_b	bubble diameter
μ^t	turbulent viscosity
μ^{eff}	effective viscosity $= \mu + \mu^t$
A_i	interficial area concentration
A_g	area fraction of wall in contact with gas
$\theta = \frac{T - T_{\text{sat}}}{T_{\text{wall}} - T_{\text{sat}}}, (\overline{\theta})$	non-dimensional temperature (mean)
$r^* = \frac{r}{R}$	non-dimensional radial distance (bubble)
Re_d	dispersed phase Reynolds number (using $\mathbf{v}_r$ and bubble diameter)
Δh_{gl}	latent enthalpy between liquid and gas (at saturation temperature T_{sat})
λ	thermal conductivity
k, ϵ	turbulent kinetic energy and dissipation rate
$\Delta T_{\text{super}} = T_{\text{wall}} - T_{\text{sat}}$	wall superheat
$\Delta T_{\text{sub}} = T_{\text{sat}} - T_l$	liquid subcooling
$T^+ = (T - T_{\text{wall}}) \frac{\rho C_p u^*}{q_{\text{wall}}}$	non-dimensional temperature for use in wall functions
y_{wall}	distance to nearest wall
$\vec{n}_{\text{wall}}$	vector normal to wall (outward facing)
$\vec{t}_{\text{wall}}$	tangential vector along wall
$\text{Eo}_d = \frac{g(\rho_l - \rho_g) d_b^2}{\sigma}$	bubble Eötvös number
σ	surface tension

This chapter outlines the differences between the multiphase flow models in Star-CD, Star-CCM+, and Nphase, and gives references for other possible models in some cases. In general, all three codes are in the Eulerian-Eulerian framework, where both liquid and gas phases are considered to be interpenetrating continua. The codes generally solve the conservation of mass, momentum, and energy for the liquid or continuous phase and at least conservation of mass and momentum for the gas or dispersed phase. Turbulence equations are solved for the liquid or continuous phase only. The small differences in implementation of the three codes can lead to significant differences in the solutions.

B.2 Governing Equations

B.2.1 Dispersed Phase Stress

Nphase	$\nabla \cdot (\mu_g \nabla \mathbf{v}_l)$	[75] and talking with developers
Star-CCM+	$\nabla \cdot (\mu_g \nabla \mathbf{v}_g)$	[1]
Star-CD	$\nabla \cdot (\mu_g \nabla \mathbf{v}_g)$	[2]

B.2.2 Turbulent Prandtl Number for Energy Equation

Nphase	0.91	standard in code
Star-CCM+	0.90	standard in code
Star-CD	0.90	standard in code

B.2.3 Turbulence

The RANS equations are used to model turbulence. All codes have the capability of using the high-Reynolds number $k - \epsilon$ model, which was used for the calculations and is compared between the codes below. The $k - \epsilon$ model is a two-equation model. The codes solve for two extra states, the turbulent kinetic energy (k) and dissipation rate (ϵ), along with the other conservation equations.

Kinetic energy

Nphase [75]

$$\frac{D_c(\alpha_c\rho_c k_c)}{Dt} = \nabla\cdot\left(\left(\mu_c + \frac{\mu_c^t}{\sigma_c^k}\right)\alpha_c\nabla k_c\right) + P - \alpha_c\rho_c\epsilon_c$$

$$P = \mu_c^{\text{eff}}\nabla\mathbf{v}_c\cdot(\nabla\mathbf{v}_c + \mathbf{v}_c\nabla) - \tfrac{2}{3}(\nabla\cdot\mathbf{v}_c)(\rho_c k_c + \mu_c^{\text{eff}}\nabla\cdot\mathbf{v}_c)$$

Star-CCM+ [1]

$$\frac{D_r(\alpha_c\rho_c k_c)}{Dt} = \nabla\cdot\left(\left(\mu_c + \frac{\mu_c^t}{\sigma_c^k}\right)\alpha_c\nabla k_c\right) + \alpha_c P - \alpha_c\rho_c\epsilon_c + \alpha_c S_c^k + \dot{m}_i k_i$$

$$P = \mu_c^t\nabla\mathbf{v}_c\cdot(\nabla\mathbf{v}_c + \mathbf{v}_c\nabla) - \tfrac{2}{3}(\nabla\cdot\mathbf{v}_c)(\rho_c k_c + \mu_c^t\nabla\cdot\mathbf{v}_c)$$

Star-CD [2]

$$\frac{D_c(\alpha_c\rho_c k_c)}{Dt} = \nabla\cdot\left(\left(\frac{\mu_c+\mu_c^t}{\sigma_c^k}\right)\alpha_c\nabla k_c\right) + \alpha_c P - \alpha_c\rho_c\epsilon_c + S_c^k + \dot{m}_i k_i$$

$$P = \mu_c^t\nabla\mathbf{v}_c\cdot(\nabla\mathbf{v}_c + \mathbf{v}_c\nabla) - \tfrac{2}{3}(\nabla\cdot\mathbf{v}_c)(\rho_c k_c + \nabla\cdot\mathbf{v}_c)$$

$$S_c^k = -A_i\frac{\nu_c^t}{\alpha_c\alpha_d\sigma_c}\mathbf{v}_r\cdot\nabla\alpha_d + 2A_i(C_t - 1)k_c$$

Note, the source term in Star-CD is their Turbulent Dispersion Force.

Dissipation rate

Nphase [75]

$$\frac{D_c(\alpha_c\rho_c\epsilon_c)}{Dt} = \nabla\cdot\left(\left(\mu_c + \frac{\mu_c^t}{\sigma_c^\epsilon}\right)\alpha_c\nabla\epsilon_c\right) + C_1\alpha_c\frac{\epsilon_c}{k_c}P - C_2\alpha_c\rho_c\frac{\epsilon_c}{k_c}$$

$$P = \mu_c^{\text{eff}}\nabla\mathbf{v}_c\cdot(\nabla\mathbf{v}_c + \mathbf{v}_c\nabla) - \tfrac{2}{3}(\nabla\cdot\mathbf{v}_c)(\rho_c k_c + \mu_c^{\text{eff}}\nabla\cdot\mathbf{v}_c)$$

Star-CCM+ [1]

$$\frac{D_r(\alpha_c\rho_c\epsilon_c)}{Dt} = \nabla\cdot\left(\left(\mu_c + \frac{\mu_c^t}{\sigma_c^\epsilon}\right)\alpha_c\nabla\epsilon_c\right) + C_1\alpha_c\frac{1}{T}P - C_2\alpha_c\rho_c\frac{\epsilon_c}{T} + \alpha_c\frac{1}{T}S_c^\epsilon + \dot{m}_i\epsilon_i$$

$$P = \mu_c^t\nabla\mathbf{v}_c\cdot(\nabla\mathbf{v}_c + \mathbf{v}_c\nabla) - \tfrac{2}{3}(\nabla\cdot\mathbf{v}_c)(\rho_c k_c + \mu_c^t\nabla\cdot\mathbf{v}_c)$$

$$T = \max\left(\frac{k}{\epsilon}, C_t\sqrt{\frac{\nu}{\epsilon}}\right)$$

Star-CD [2]

$$\frac{D_c(\alpha_c\rho_c\epsilon_c)}{Dt} = \nabla\cdot\left(\left(\frac{\mu_c+\mu_c^t}{\sigma_c^\epsilon}\right)\alpha_c\nabla\epsilon_c\right) + C_1\alpha_c\frac{\epsilon_c}{k_c}P - C_2\alpha_c\rho_c\frac{\epsilon_c^2}{k_c} + S_c^\epsilon + \dot{m}_i\epsilon_i$$

$$P = \mu_c^t\nabla\mathbf{v}_c\cdot(\nabla\mathbf{v}_c + \mathbf{v}_c\nabla) - \tfrac{2}{3}(\nabla\cdot\mathbf{v}_c)(\rho_c k_c + \nabla\cdot\mathbf{v}_c)$$

$$S_c^\epsilon = 2A_i(C_t - 1)\epsilon_c$$

The Star-CCM+ manual does not explicitly include the C_t model in the equations. The formulation in the manual says that these source terms S_c^k and S_c^ϵ are "user-specified".

B.2.4 Turbulence Closure Coefficients

	C_μ	C_1	C_2	σ_c^k	σ_c^ϵ	
Nphase	0.09	1.44	1.92	1.0	1.3	standard in code
Star-CCM+	0.09	1.44	1.92	1.0	1.3	standard in code
Star-CD	0.09	1.44	1.92	1.0	1.219	standard in code

B.2.4.1 Ranges

- C_μ: formula from [66] in Nphase DEBORA solution gives $C_\mu \in [0.02, 0.09]$, with most of the domain at 0.09. Near the wall it goes down to around 0.08, and there is a small area near the inlet where it goes down to 0.02. Formulas from [67] and [65, 98] give similar results. Note, the "realisable" k-epsilon model was created to address this problem and has $C_\mu = f(k, \epsilon, \mathbf{v})$.
- C_1: no information found.
- C_2: formula from [66] in Nphase DEBORA solution gives $C_2 = 1.92$ everywhere because of the large turbulent Reynolds number $R_T \in [30, 8800]$.
- σ_c^k: no information found.
- σ_c^ϵ: no information found.

B.3 Heat Transfer

B.3.1 Nusselt Number - Interface to Gas

Nphase	$-2 \frac{\partial \theta}{\partial r^*}\big\vert_{r^*=1} (\overline{\theta})$	see Figure 6.1. [102] (see ref for how to calculate $\mathrm{Nu}_g(\overline{\theta})$)
Star-CCM+	26	can be set to field function
Star-CD	26	[77]

B.3.1.1 Ranges

- Nu_g: in Nphase DEBORA solution, $\mathrm{Nu}_g \in [0, 23]$. However, here the enthalpy is non-dimensionalized with an arbitrary value h_i (the fixed wall enthalpy, which is set but never used in the solution since there is a known heat flux). By changing this value, any range of Nu_g can be obtained. The "correct" non-dimensionalization is unclear from the paper [102], since the formula is derived from unsteady bubble growth/collapse and h_i is the initial "average" enthalpy in some sense. Also, [89] has $\mathrm{Nu}_g = 10$.

B.3.2 Nusselt Number - Interface to Liquid

The Ranz-Marshall correlation

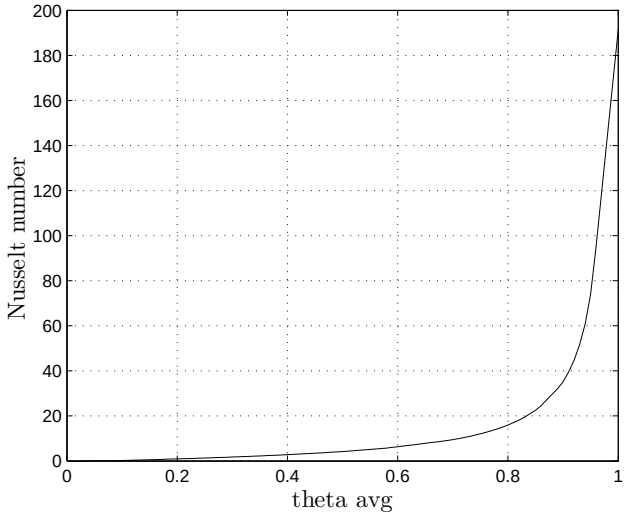


Figure 6.1: Nphase correlation for interface to gas heat transfer.

Nphase	$2 + (0.4\mathrm{Re}_d^{0.5} + 0.06\mathrm{Re}^{2/3})\mathrm{Pr}_c^{0.4}$	Modified Ranz-Marshall (standard in code), [71]
Star-CCM+	$2 + 0.6\mathrm{Re}_d^{0.5}\mathrm{Pr}_c^{0.3}$	[1, 96]
Star-CD	$2 + 0.6\mathrm{Re}_d^{0.5}\mathrm{Pr}_c^{0.3}$	[2, 96]

B.3.2.1 Ranges

- Nu_l: There are many other correlations besides the two given above. In the Nphase DEBORA solution, with the Nphase model we have $\mathrm{Nu}_l \in [2, 33]$.
  - $\mathrm{Nu}_l = 2 + 0.6\mathrm{Re}_d^{0.6}\mathrm{Pr}_c^{0.5}$ from [84, 22] which correlates with experimental data for Nu_l to $\pm 15 - 20\%$
  - $\mathrm{Nu}_l = 2 + 0.185\mathrm{Re}_d^{0.7}\mathrm{Pr}_c^{0.5}$ from [84, 22] which correlates with experimental data for Nu_l to $\pm 70\%$
  - $\mathrm{Nu}_l = 1/(2\pi)\mathrm{Re}_d^{0.5}\mathrm{Pr}_c^{1/3}$ from [54]
  - $\mathrm{Nu}_l = 2.09\mathrm{Re}_d^{0.61}\alpha_d^{0.328}\mathrm{Ja}^{-0.308}$ from [123], Ja = Jacob number
  - $\mathrm{Nu}_l = 0.6\mathrm{Re}_d^{0.5}\mathrm{Pr}_c^{1/3}\left[1 - 1.20\mathrm{Ja}^{9/10}\mathrm{Fo}_d^{2/3}\right]$ from [118] which correlates with experimental data for Nu_l to $\pm 28\%$.
  - see also [123] for yet more models

B.3.3 Wall Heat Partitioning

Heat flux q_{wall} must be divided between liquid and gas phases. In all three codes, the heat that goes into the gas phase only generates gas at saturation enthalpy and does not go into heating the gas. Care must be taken that the resulting volume fraction of gas does not exceed 1.

Nphase: User input (boundary condition is specified gas mass flux) is the standard in code.

Star-CCM+ and Star-CD: T_{wall} is chosen to enforce $q_l + q_g = (q_c + q_q) + (q_e) = q_{\text{wall}}$, and must be found iteratively.

B.3.3.1 Convective Heat Flux

$$q_c = \text{HTC}_c(1 - A_g)(T_{\text{wall}} - T_l)$$

The convective heat transfer coefficient HTC_c is calculated from wall functions.

$$\text{HTC}_c = \frac{\rho_l C_{p,l} u_l^*}{T_l^+}$$

Reference [1]

B.3.3.2 Quenching Heat Flux

$$\begin{aligned} q_q &= \text{HTC}_q A_g (T_{\text{wall}} - T_l) \\ \text{HTC}_q &= 2K_q f \sqrt{\frac{t_w \rho_l C_{p,l} \lambda_l}{\pi}} \end{aligned}$$

References [2, 1]

B.3.3.3 Evaporative Heat Flux

$$q_e = \frac{\pi d_w^3}{6} \rho_g \Delta h_{gl} f n''$$

B.3.3.4 Auxiliary Relations

$$K_q = \begin{cases} 1 & \text{Star-CD} \\ F_A \frac{\pi d_w^2}{4} n'' & \text{Star-CCM+} \end{cases}$$

With $F_A = 2$ is taken as the standard. It is noted that some authors use $F_A = 4$. Star-CD reference [2, 77]. Star-CCM+ references [1, 62].

$$n'' = (m \Delta T_{\text{super}})^p, \quad m = 185 \quad p = 1.805$$

In Star-CD, if $\Delta T_{\text{super}} < 0$, then no boiling is assumed to occur and $n'' = 0$ (standard in code). In Star-CCM+, ΔT_{super} is restricted to lie in the interval $[0, \Delta T_{\max}]$, with a default $\Delta T_{\max} = 20.0K$. References [1, 77, 69, 93].

$$f = \sqrt{\frac{4}{3} \frac{g(\rho_l - \rho_g)}{d_w \rho_l}}$$

References [1, 77, 27]

$$d_w = d_{w,0} \exp\left(-\frac{\Delta T_{\text{sub}}}{\Delta T_0}\right), \quad d_{w,0} = 0.6mm \quad \Delta T_0 = 45K$$

In Star-CCM+, d_w is restricted to lie in the interval $[0.025, 1.4]mm$. References [1, 77, 115].

$$t_w = 0.8/f$$

References [77, 1, 62].

B.4 Dispersed Phase

B.4.1 Bubble Diameter, Interficial Area Concentration

	d	A_i	
Nphase	user-specified constant	$6\frac{\alpha_d}{d_b}$	[75], standard in code
Star-CCM+	constant or same as Star-CD	$6\frac{\alpha_d}{d_b}$	[1]
Star-CD	$\frac{d_{b,1}(\Delta T_{\text{sub}}-\Delta T_0)+d_{b,0}(\Delta T_1-\Delta T_{\text{sub}})}{\Delta T_1-\Delta T_0}$	$6\frac{\min(\alpha_d,(1-\alpha_d))}{d_b}$	[2]

In Star-CD, $d_{b,0}$ and $d_{b,1}$ are the min and max bubble diameters. The formula is a linear curve fit of $d_b(\Delta T_{\text{sub}})$. Standard parameters are

$$d_{b,0} = 1.5 \times 10^{-4}m \quad d_{b,1} = 2 \times 10^{-3}m \quad \Delta T_0 = 13.5K \quad \Delta T_1 = -5K$$

Also in Star-CD, in the code we had (see user routine `uevar.f`), the formula for A_i that was implemented was

$$A_i = 6\frac{\alpha_d(1-\alpha_d)}{d_b}$$

Star-CD also has the ability to solve extra field equations for the void fraction distribution. This is called the $S-\Gamma$ model.

B.4.1.1 Ranges

- Star-CD DEBORA solution has $d_b \in [0, 15.5\times10^{-4}]$m, with the average of the non-zero values around 7×10^{-4}m.
- Star-CD DEBORA solution with the $S-\Gamma$ model has $d_b \in [0, 13.7 \times 10^{-4}]$m.
- Star-CCM+ DEBORA solution has $d_b \in [1.5 \times 10^{-4}, 15.7 \times 10^{-4}]$m.
- Nphase DEBORA solution with fixed $d_b = 7 \times 10^{-4}$m. If the temperature field is used to calculate $d_b(T)$ (i.e. the Star-CD formula), it gives $d_b \in [1.5 \times 10^{-4}, 18.46 \times 10^{-4}]$m with an average of 6.35×10^{-4}m.
- Experimental data reported in [122] has $d_b \in [3.75 \times 10^{-4}, 11 \times 10^{-4}]$m, with most of the pipe at 10×10^{-4}m. Experimental error was around 12%.

B.5 Interficial Forces

B.5.1 Drag Force

Nphase	$-\frac{1}{8}C_D\rho_c\lvert\mathbf{v}_r\rvert\mathbf{v}_r A_i$	[75]
Star-CCM+	$-\frac{1}{8}C_D\rho_c\lvert\mathbf{v}_r\rvert\mathbf{v}_r A_i$	[1]
Star-CD	$-\frac{1}{8}C_D\rho_c\lvert\mathbf{v}_r\rvert\mathbf{v}_r 6\frac{\alpha_d}{d_b}$	[2]

Note, in Star-CD a different correlation for $A_i(\alpha_d)$ is used for other quantities (e.g. mass and energy transfer), though here it is implicitly defined as $A_i = 6\alpha_d/d_b$.

B.5.2 Drag Coefficient

All codes are set up to use a range of curve fits for $C_D(\mathrm{Re})$, in this case the Wang curve fit for drag on a single bubble is used [117]

$$C_D(\mathrm{Re}) = \exp\left[a + b\ln\left(\mathrm{Re}_d\right) + c\ln^2\left(\mathrm{Re}_d\right)\right]$$

	a	b	c
$\mathrm{Re}_d \leq 1$	$\ln(24)$	-1	0
$1 \leq \mathrm{Re}_d \leq 450$	2.699467	-0.33581596	-0.07135617
$450 \leq \mathrm{Re}_d \leq 4000$	-51.771717	13.1670725	-0.8235592
$\mathrm{Re}_d \geq 4000$	$\ln\left(\frac{8}{3}\right)$	0	0

B.5.2.1 Ranges

- C_D: Access to the original paper by Wang was not available, and no comparison to experiments has been found. However, other curve fits for bubble drag suggest that $\pm 30\%$ is a reasonable range (e.g. [58]).

B.5.3 Bubble Swarm

This factor is in the Star-CD ufiles (see `uedrag.f`) and multiplies C_D to modify it for large void fractions

$$(1-\alpha_d)\left[1-\min\left(\alpha_d,\alpha_0\right)\right]^{r_{\mathrm{swarm}}}, \qquad \alpha_0 = 0.4325, \quad r_{\mathrm{swarm}} = 3.0$$

The values above were implemented in Star-CD and in Nphase for the DEBORA calculations. Figure 6.2 shows some other models in the literature.

Star-CD	$(1-\alpha_d)\left[1-\min\left(\alpha_d,\alpha_0\right)\right]^{r_{\text{swarm}}}$
Notes: $\alpha_0 = 0.4325$, $r_{\text{swarm}} = 3.0$, standard in code	
Simmonet et al.	$(1-\alpha_d)\left[(1-\alpha_d)^m + \left(4.8\frac{\alpha_d}{1-\alpha_d}\right)^m\right]^{-2/m}$
Notes: $m = 25$, [104]. valid for $0.15 \lesssim \alpha_d \lesssim 0.30$	
Dijkhuizen et al.	$\left(1+\frac{18}{\text{Eo}}\alpha_d\right)(1-\alpha_d)$
Notes: [64]. Eo= $\frac{\Delta\rho g d_b^2}{\sigma}$. Correlation from DNS, errors up to $\approx 25\%$	
Marucci et al.	$\frac{1-\alpha_d^{5/3}}{(1-\alpha_d)^2}$
Notes: [81]. Exact result from laminar, irrotational flow near spheres (Re$\lesssim$ 300).	
Tomiyama et al.	$(1-\alpha_d)^{3-2\ell}$
Notes: $\ell = 1.75$, [52, 110] Correlation with experiments.	

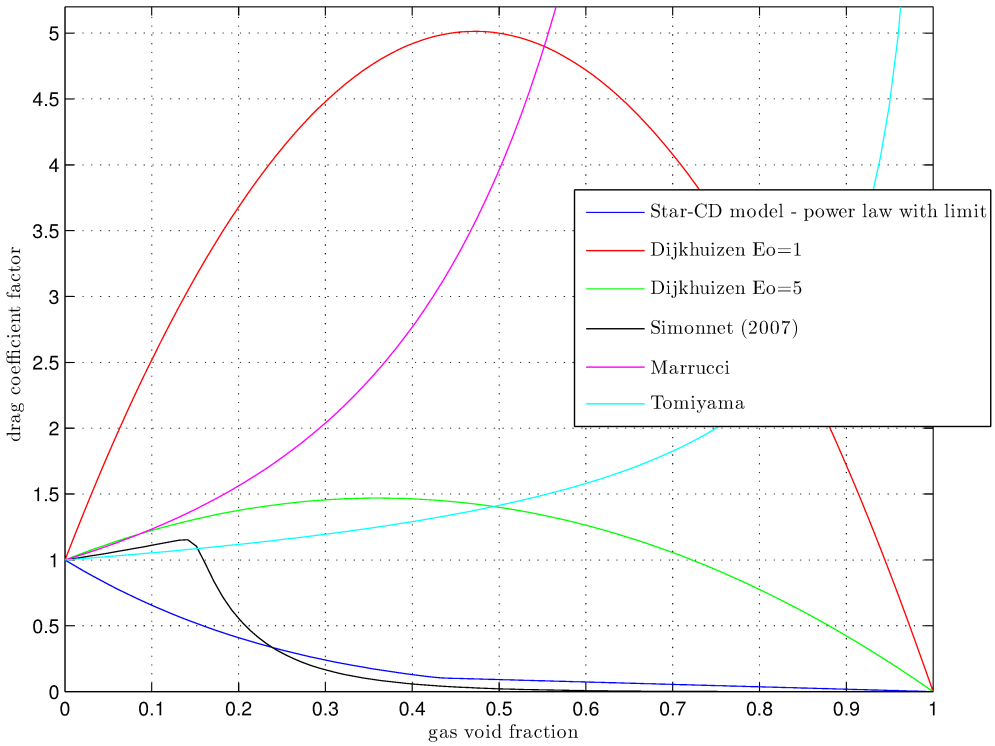


Figure 6.2: Various bubble swarm models (drag coefficient multipliers for large void fractions)

B.5.4 Lift Force

Force on dispersed phase/bubbles.

Nphase	$-C_L\rho_c\alpha_d\mathbf{v}_r \times (\nabla \times \mathbf{v}_c)$	[75, 33]
Star-CCM+	$-C_L\rho_c\alpha_d\mathbf{v}_r \times (\nabla \times -\mathbf{v}_r)$	[1]
Star-CD	$-C_L\rho_c\alpha_d\mathbf{v}_r \times (\nabla \times \mathbf{v}_c)$	[2, 33]

B.5.5 Lift Coefficient

Nphase

Originally, Nphase had the following lift coefficient

$$C_L = \begin{cases} 0 & , \quad y_{\text{wall}} \leq d_b \\ -0.03\,(y_{\text{wall}}/d_b - 1) & , \quad d_b \leq y_{\text{wall}} \leq 2d_b \\ -0.03 & , \quad y_{\text{wall}} \geq 2d_b \end{cases}$$

The lift coefficient goes to zero near the wall, which simulates the wall lubrication force. In addition, C_L was set to zero below a void fraction of 10^{-3}.

Built-in to Nphase are modified versions of the lift coefficient that set it to zero at high values of void fraction and liquid vorticity.

Star-CCM+

There was no baseline in Star-CCM+. During testing, many values were attempted. The baseline from Star-CD of -0.03 did not converge with positive virtual mass coefficient, and gave very large void fractions in the center of the pipe. A value of -0.002 was found to give void fractions similar to experimental results. Convergence was also obtained for a user-code implementation of the Nphase model above (i.e. setting C_L to zero near the wall).

Star-CD

CD-Adapco suggested using $C_L = -0.03$ for the DEBORA case (constant everywhere in the flow). A parameter sweep (with all other parameters at baseline) was performed for $C_L = [-0.3, 0.3]$ as suggested by Tomiyama [111]. Convergence was obtained for $-0.1 \leq C_L \leq 0.3$. However, for $C_L \geq 0$, the residuals "stalled out" at large values and some aspects of the solution varied significantly between iterations. Figures 6.4 and 6.3 shows how the heat partitioning calculated by Star-CD varies with C_L (only a few representative positive values of C_L are shown).

None of the attempts at using the Tomiyama lift coefficient correlation for $C_L(\text{Eo}_d, \text{Re}_d)$ converged. The correlation is

$$C_L = \begin{cases} \min\left[0.288\tanh\left(0.121\text{Re}_d\right), f\left(\text{Eo}_d\right)\right] & , \quad \text{Eo}_d \leq 4 \\ f\left(\text{Eo}_d\right) & , \quad 4 \leq \text{Eo}_d \leq 10.7 \end{cases}$$

$$f\left(\text{Eo}_d\right) = 0.00105\text{Eo}_d^3 - 0.0159\text{Eo}_d^2 - 0.0204\text{Eo}_d + 0.474$$

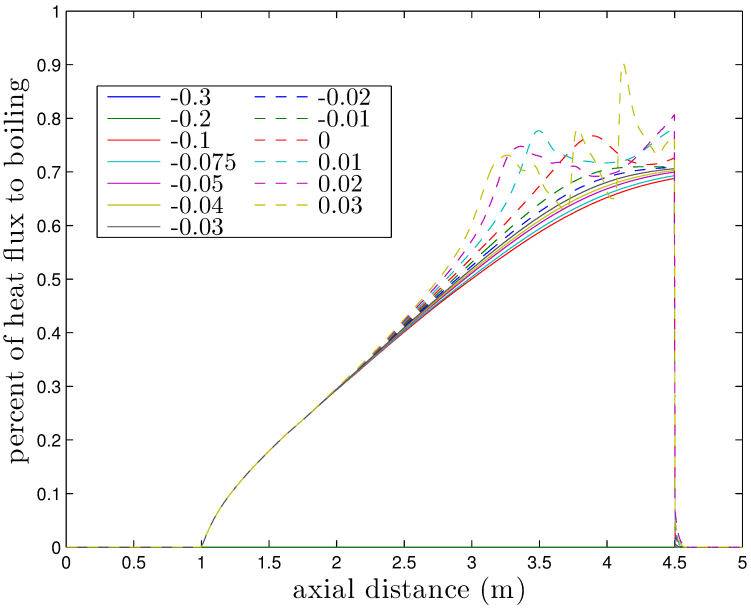


Figure 6.3: Star-CD heat flux partitioning with various C_L. Note, the solution for the two lowest values of C_L did not converge.

B.5.5.1 Ranges

From the above discussions, it is natural to take $C_L \in [-0.01, 0.1]$.

B.5.6 Wall Force

This force is available in Nphase and Star-CD, but was not used for the DEBORA case. Below is the force on the dispersed phase.

Nphase	$C_{\text{wall}} f^1(y_{\text{wall}}) \alpha_d \rho_c \lvert \mathbf{v}_r \cdot \vec{t}_{\text{wall}} \rvert^2 \vec{n}_{\text{wall}}$	[75]
Star-CCM+	-	
Star-CD	$C_{\text{wall}} f^2(y_{\text{wall}}) \alpha_d \rho_c \lvert \mathbf{v}_r \cdot \vec{t}_{\text{wall}} \rvert^2 \vec{n}_{\text{wall}}$	[2]

$$f^1(y_{\text{wall}}) = \begin{cases} \frac{1}{d_b}\left[1 + \left(\frac{y_{\text{wall}}}{d_b}\right)^2 \left(\frac{1}{4}\frac{y_{\text{wall}}}{d_b} - \frac{3}{4}\right)\right] & , \quad y_{\text{wall}}/d_b \le 2 \\ 0 & , \quad y_{\text{wall}}/d_b \ge 2 \end{cases}$$

$$f^2(y_{\text{wall}}) = \max\left[\frac{-0.0167}{d_b} + \frac{0.0667}{y_{\text{wall}}}, 0\right]$$

By default, Nphase and Star-CD take $C_{\text{wall}} = 1.0$. The f function causes the wall force to decrease to zero by $y_{\text{wall}} \le 2d_b$ for Nphase and $y_{\text{wall}} \lesssim 4d_b$ for Star-CD. The two functions are plotted in Figure 6.5.

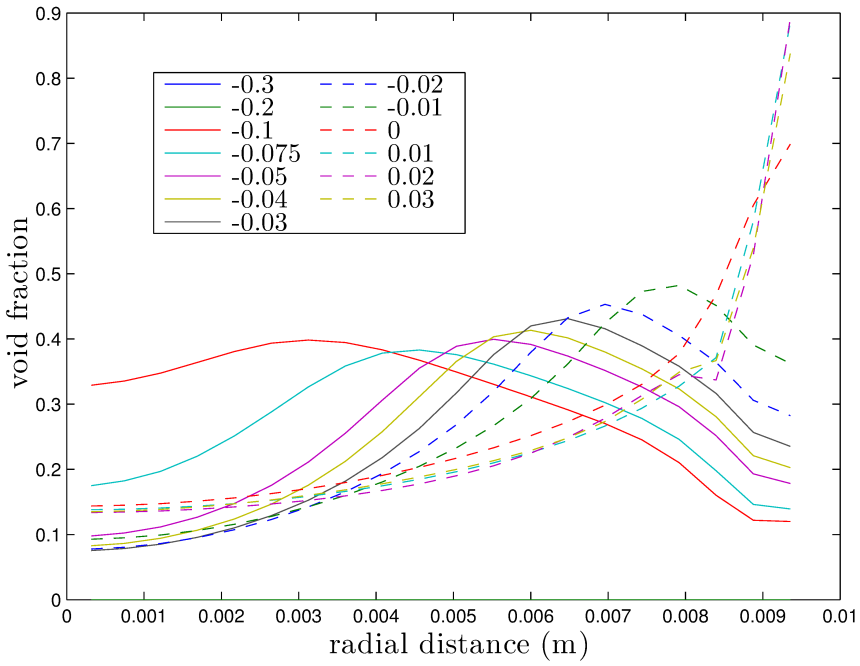


Figure 6.4: Star-CD void fraction profiles at the end of the heated section with various C_L. Note, the solution for the two lowest values of C_L did not converge.

B.5.6.1 Ranges

- maximum y_{wall}/d_b: for Star-CD and in [52, 111], this is set to 4.
- maximum y_{wall}/d_b: varies in $[1, 1.4]$ in [5].

B.5.7 Virtual Mass Force

Force on dispersed phase.

Nphase	$C_{VM}\rho_c\alpha_d\left(\frac{D_c\mathbf{v}_c}{Dt} - \frac{D_d\mathbf{v}_d}{Dt}\right)$	[75, 33]
Star-CCM+	$-C_{VM}\rho_c\alpha_d\left(\frac{D_c\mathbf{v}_c}{Dt} - \frac{D_d\mathbf{v}_d}{Dt}\right)$	[1]
Star-CD	$C_{VM}\rho_c\alpha_d\left(\frac{D_c\mathbf{v}_c}{Dt} - \frac{D_d\mathbf{v}_d}{Dt}\right)$	[2, 33]

Force was developed theoretically for motion of a body in an inviscid fluid in [8]. This paper suggests that the force should be

$$F_{VM} = \rho_c\mathcal{V}\left[(1 + C_{VM})\frac{D_c\mathbf{v}_c}{Dt} - C_{VM}\frac{D_d\mathbf{v}_d}{Dt}\right]$$

with $C_{VM} = 0.5$ for a sphere. Here, $\mathcal{V}$ is the volume of the particle/bubble (thus, for the force per unit volume in a multiphase flow $\mathcal{V}$ is replaced by α_d).

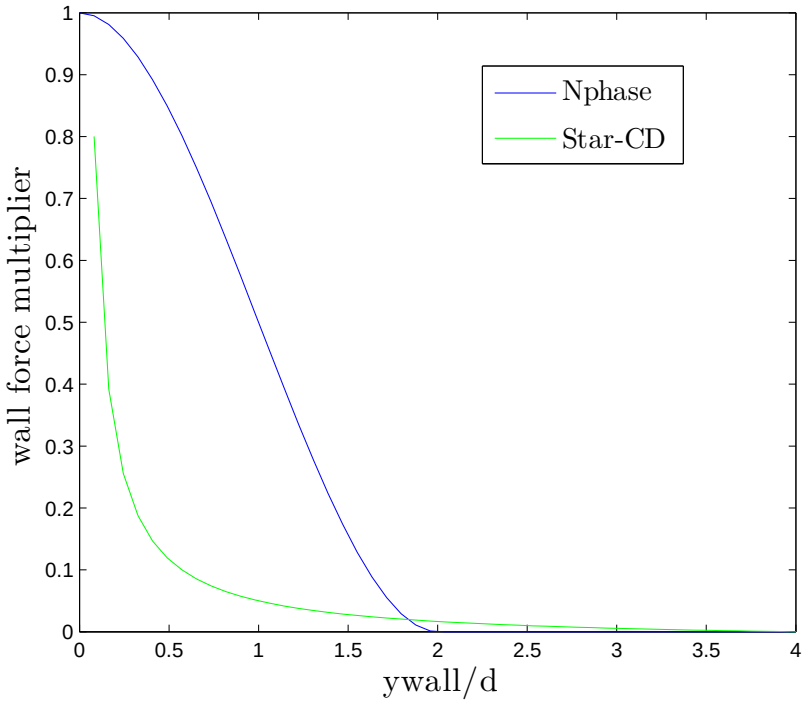


Figure 6.5: Wall force coefficient as a function of non-dimensional distance from the wall.

B.5.8 Virtual Mass Coefficient

Nphase	1.0	[75]
Star-CCM+	0.5	[1, 33]
Star-CD	0.5	[2, 33]

B.5.8.1 Ranges

- C_{VM}: Bertonado [78] suggests that $C_{VM} \in [1.2, 3.4]$ from previous experimental and theoretical work.

B.5.9 Turbulent Dispersion Force

Nphase	$-C_{\mathrm{TD}}\rho_c k \nabla \alpha_d$	[75, 52]
Star-CCM+	$-A^D \frac{\nu_c^t}{\sigma_\alpha}\left(\frac{\nabla\alpha_d}{\alpha_d} - \frac{\nabla\alpha_c}{\alpha_c}\right)$	[1]
Star-CD	$-A^D \frac{\nu_c^t}{\sigma_\alpha}\left(\frac{\nabla\alpha_d}{\alpha_d} - \frac{\nabla\alpha_c}{\alpha_c}\right)$	[2]

B.5.10 Turbulent Dispersion Coefficient and Prandtl Number

	C_{TD}	σ_α	
Nphase	2/3	–	[75]
Star-CCM+	$-A^D$	1.0	[1]
Star-CD	$-A^D$	1.0	[2]

A^D is the "linearized drag coefficient":

$$A^D = \frac{3}{4}\frac{\alpha_d \rho_c C_D}{d_b}|\mathbf{v}_r|$$

B.5.10.1 Ranges

- C_{TD}: experimental data in [122] suggests $C_{\text{TD}} \in [1, 2.5]$.
- C_{TD}: the Nphase DEBORA solution, using the model from [52], gives $C_{\text{TD}} \in [0, 0.63]$.

B.6 Other Codes and References

See [52] for another multiphase code with some more complicated momentum transfer models based on 6 bubble regimes/populations and correlations for non-spherical bubbles.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Star-CCM+ User Manual, star-ccm+ version 6.06.011. Technical report, CD-Adapco.

[2] Star-CD User Manual, star-cd version 4.14.011. Technical report, CD-Adapco.

[3] Brian M Adams, William Eugene Hart, Michael Scott Eldred, Daniel M Dunlavy, Patricia Diane Hough, Anthony Andrew Giunta, Joshua D Griffin, Monica L Martinez-Canales, Jean-Paul Watson, Tamara Gibson Kolda, et al. *DAKOTA, a Multilevel Parellel Object-oriented Framework for Design Optimization, Parameter Estimation, Uncertainty Quantification, and Sensitivity Analysis: Version 4.0 Uers's Manual*. United States. Department of Energy, 2006.

[4] Mark Ainsworth and J Tinsley Oden. A posteriori error estimation in finite element analysis. *Computer Methods in Applied Mechanics and Engineering*, 142(1):1–88, 1997.

[5] SP Antal, RT Lahey Jr, and JE Flaherty. Analysis of phase distribution in fully developed laminar bubbly two-phase flow. *International Journal of Multiphase Flow*, 17(5):635–652, 1991.

[6] Douglas N Arnold, Franco Brezzi, Bernardo Cockburn, and L Donatella Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM journal on numerical analysis*, 39(5):1749–1779, 2002.

[7] Uri M Ascher and Linda R Petzold. *Computer methods for ordinary differential equations and differential-algebraic equations*, volume 61. Siam, 1998.

[8] T.R. Auton, J.C.R. Hunt, and M. Prud'homme. The force exerted on a body in inviscid unsteady non-uniform rotational flow. *J. Fluid Mechanics*, 197:241–257, 1988.

[9] S Bajorek et al. Trace v5. 0 theory manual, field equations, solution methods and physical models. *United States Nuclear Regulatory Commission*, 2008.

[10] Youngsuk Bang, Hany S Abdel-Khalik, and Jason M Hite. Hybrid reduced order modeling applied to nonlinear models. *International Journal for Numerical Methods in Engineering*, 91(9):929–949, 2012.

[11] Bernhard Beckermann. The condition number of real vandermonde, krylov and positive definite hankel matrices. *Numerische Mathematik*, 85(4):553–577, 2000.

[12] Géraud Blatman and Bruno Sudret. An adaptive algorithm to build up sparse polynomial chaos expansions for stochastic finite element analysis. *Probabilistic Engineering Mechanics*, 25(2):183–197, 2010.

[13] Kurt M Bretthauer and Bala Shetty. The nonlinear knapsack problem–algorithms and applications. *European Journal of Operational Research*, 138(3):459–472, 2002.

[14] A.D. Burns, T. Frank, I. Hamill, and J.-M. Shi. The Favre averaged drag model for turbulence dispersion in Eulerian multi-phase flows. 5th international conference on multiphase flow, ICMF2004, Yokohama, Japan, 2004.

[15] Emmanuel J Candès and David L Donoho. Ridgelets: A key to higher-dimensional intermittency? *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 357(1760):2495–2509, 1999.

[16] C. E. Castro and E. F. Toro. New numerical approaches to multiphase flows modeling. *International Journal of Energetic Materials and Chemical Propulsion*, 6(5):609–627, 2007.

[17] CE Castro and EF Toro. A Riemann solver and upwind methods for a two-phase flow model in non-conservative form. *International journal for numerical methods in fluids*, 50(3):275–307, 2006.

[18] CD-adapco. STAR-CD version 4.14 methodology manual, chapter 13, 2010.

[19] CD-adapco. STAR-CD version 4.14.011 methodology manual, 2010.

[20] Marco Ceze and Krzysztof J Fidkowski. A robust adaptive solution strategy for high-order implicit CFD solvers. In *20th AIAA Computaional Fluid Dynamics Conference*, 2011.

[21] John C Chen. Correlation for boiling heat transfer to saturated fluids in convective flow. *Industrial & Engineering Chemistry Process Design and Development*, 5(3):322–329, 1966.

[22] YM Chen and F. Mayinger. Measurement of heat transfer at the phase interface of condensing bubbles. *International journal of multiphase flow*, 18(6):877–890, 1992.

[23] B Chexal, J Horowitz, and GS Lellouche. An assessment of eight void fraction models. *Nuclear engineering and design*, 126(1):71–88, 1991.

[24] Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002.

[25] Bernardo Cockburn and Chi-Wang Shu. The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463, 1998.

[26] Paul Coddington and Rafael Macian. A study of the performance of void fraction correlations used in the context of drift-flux two-phase flow models. *Nuclear Engineering and Design*, 215(3):199–216, 2002.

[27] R. Cole. A photographic study of pool boiling in the region of the critical heat flux. *AIChE Journal*, 6:533–542, 1960.

[28] Paul G Constantine, Eric Dow, and Qiqi Wang. Active subspace methods in theory and practice: applications to kriging surfaces. *arXiv preprint arXiv:1304.2070*, 2013.

[29] Richard W Cottle. Manifestations of the schur complement. *Linear Algebra and its Applications*, 8(3):189–211, 1974.

[30] Wei Dai and Olgica Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *Information Theory, IEEE Transactions on*, 55(5):2230–2249, 2009.

[31] John D Denton. The calculation of three-dimensional viscous flow through multistage turbomachines. *Journal of turbomachinery*, 114(1):18–26, 1992.

[32] David L Donoho and Yaakov Tsaig. Fast solution of-norm minimization problems when the solution may be sparse. *Information Theory, IEEE Transactions on*, 54(11):4789–4812, 2008.

[33] D. A. Drew and R. R. Lahey Jr. The virtual mass and lift force of a sphere in rotating and straining inviscid flow. *International Journal of Multiphase Flow*, 13(1):113–121, 1987.

[34] T.J. Drzewiecki, I.M. Asher, T.P. Grunloh, V.E. Petrov, K.J. Fidkowski, A. Manera, and T.J. Downar. Parameter sensitivity study of boiling and two-phase flow models in CFD. *Journal of Computational Multiphase Flow*, 4(4):411–425, 2012.

[35] Karthik Duraisamy and Praveen Chandrashekar. Goal-oriented uncertainty propagation using stochastic adjoints. *Computers & Fluids*, 66:10–20, 2012.

[36] AR t Edwards and TP O'Brien. Studies of phenomena connected with the depressurization of water reactors. Technical report, United Kingdom Atomic Energy Authority, Risley, Eng., 1970.

[37] Joe Faith. Interactive data exploration with targeted projection pursuit. *Leonardo Electronic Almanac*, 16(6-7), 2009.

[38] Krzysztof Fidkowski and David Darmofal. Output error estimation and adaptation in computational fluid dynamics: Overview and recent results. In *47th AIAA Aerospace Sciences Meeting, Orlando, FL, 5-8 Jan*, 2009.

[39] Krzysztof J Fidkowski and Yuxing Luo. Output-based space–time mesh adaptation for the compressible Navier–Stokes equations. *Journal of Computational Physics*, 230(14):5753–5773, 2011.

[40] Tore Flatten and Svend Tollak Munkejord. The approximate Riemann solver of Roe applied to a drift-flux two-phase flow model. *ESAIM-Mathematical Modelling and Numerical Analysis*, 40(4):735–764, 2006.

[41] Jerome H Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76(376):817–823, 1981.

[42] Jean Gallier. The schur complement and symmetric positive semidefinite (and definite) matrices. *December*, 44:1–12, 2010.

[43] J. Garnier, E. Manon, and G. Cubizolles. Local measurement and flow boiling of refrigerant 12 in a vertical tube. *Multiphase Science and Technology*, 13:1–111, 2001.

[44] J Garnier, E Manon, and G Cubizolles. Local measurements on flow boiling of refrigerant 12 in a vertical tube. *Multiphase Science and Technology*, 13(1&2), 2001.

[45] Thomas Gerstner and Michael Griebel. Dimension–adaptive tensor–product quadrature. *Computing*, 71(1):65–87, 2003.

[46] Roger Ghanem. Ingredients for a general purpose stochastic finite elements implementation. *Computer Methods in Applied Mechanics and Engineering*, 168(1):19–34, 1999.

[47] Hervé Guillard, Fabien Duval, Jean Claude Latche, and Roxana Panescu. Numerical multiphase modeling of bubbly flows. *ANNALI DELL'UNIVERSITA'DI FERRARA*, 53(2):243–253, 2007.

[48] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

[49] Ralf Hartmann. Adjoint consistency analysis of discontinuous Galerkin discretizations. *SIAM Journal on Numerical Analysis*, 45(6):2671–2696, 2007.

[50] Alan Thomas Joseph Hayward. Compressibility equations for liquids: a comparative study. *British Journal of Applied Physics*, 18(7):965, 1967.

[51] Takashi Hibiki and Mamoru Ishii. One-dimensional drift-flux model and constitutive equations for relative motion between phases in various two-phase flow regimes. *International Journal of Heat and Mass Transfer*, 46(25):4935–4948, 2003.

[52] S. Hosokawa and A. Tomiyama. Multi-fluid simulation of turbulent bubbly pipe flows. *Chemical Engineering Science*, 64:5308–5318, 2009.

[53] Peter J Huber. Projection pursuit. *The annals of Statistics*, pages 435–475, 1985.

[54] J. Isenberg and S. Sideman. Direct contact heat transfer with change of phase: bubble condensation in immiscible liquids. *International Journal of Heat and Mass Transfer*, 13(6):997–1011, 1970.

[55] M. Ishii and K. Mishima. Two-fluid model and constitutive relations. *Nuclear Engineering and Design*, 82:107–126, 1984.

[56] Mamoru Ishii. One-dimensional drift-flux model and constitutive equations for relative motion between phases in various two-phase flow regimes. Technical report, Argonne National Lab., Ill.(USA), 1977.

[57] Mamoru Ishii and Takashi Hibiki. *Thermo-fluid dynamics of two-phase flow*. Springer, 2011.

[58] D.G. Karamanev. Rise of gas bubbles in quiescent liquids. *AIChE journal*, 40(8):1418–1421, 1994.

[59] Susan Werner Kieffer. Sound speed in liquid-gas mixtures: Water-air and water-steam. *Journal of Geophysical research*, 82(20):2895–2904, 1977.

[60] Andreas Klimke and Barbara Wohlmuth. Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in matlab. *ACM Transactions on Mathematical Software (TOMS)*, 31(4):561–579, 2005.

[61] E. Krepper and R. Rzehak. CFD for subcooled flow boiling: Simulation of DEBORA experiments. *Nuclear Engineering and Design*, 241:3851–3866, 2011.

[62] N. Kurul and M.Z. Podowski. Multidimensional effects in sub-cooled boiling. In *Proceedings of the 9th Heat Transfer Conference, Jerusalem*, 1990.

[63] N. Kurul and M.Z. Podowski. On the modeling of multidimensional effects in boiling channels. ANS proceedings, national heat transfer conference, 1991.

[64] Y.M. Lau, I Roghair, N.G. Deen, M. Van Sint Annaland, and J.A.M. Kuipers. Numerical investigation of the drag closure for bubbles in bubble swarms. 66:3309–3316, 2011.

[65] B. E. Launder, A. Morse, W. Rodi, and D. B. Spalding. Prediction of free shear flows – a comparison of the performance of six turbulence models. Technical report, NASA, 1973. Document ID: 19730019433.

[66] BE Launder and BI Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters Heat Mass Transfer*, 1:131–137, 1974.

[67] BE Launder and DB Spalding. The numerical computation of turbulent flows. *Computer methods in applied mechanics and engineering*, 3(2):269–289, 1974.

[68] Tobias Leicht and Ralf Hartmann. Anisotropic mesh refinement for discontinuous Galerkin methods in two-dimensional aerodynamic flow simulations. *International journal for numerical methods in fluids*, 56(11):2111–2138, 2008.

[69] M. Lemmert and J.M. Chawla. *Influence of Flow Velocity on Surface Boiling Heat Transfer Coefficient*. Academic Press and Hemisphere, New York, 1977.

[70] Randall J LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[71] D. Levi-Hevroni, A. Levy, and I. Borde. Mathematical modeling of drying of liquid/solid slurries in steady state one-dimensional flow. *Drying Technology*, 13(5-7):1187–1201, 1995. cited By (since 1996) 24.

[72] R Li and R Ghanem. Adaptive polynomial chaos expansions applied to statistics of extremes in nonlinear random vibration. *Probabilistic engineering mechanics*, 13(2):125–136, 1998.

[73] P. J. Linstrom and W. G. Mallard, editors. *NIST Chemistry WebBook, NIST Standard Reference Database Number 69*. National Institute of Standards and Technology, Gaithersburg MD, 20899, 2013. (retrieved November 3, 2013).

[74] Inerphase Dynamics LLC. Nphase-CMFD version 3.1.19 user manual, 2009.

[75] Interphase Dynamics LLC. Nphase-cmfd user manual, nphase version 3.1.19. Technical report, Rensselaer Polytechnic Institute, Sept 2009.

[76] S. Lo, A. Splawski, and B.J. Yun. The importance of correct modelling of bubble size and condensation prediction of sub-cooled boiling flows. In *Fourteenth International Topical Meeting on Nuclear Reactor Thermal Hydraulics*, 2011.

[77] Simon Lo. Eulerian multiphase flows. Technical report, CD-Adapco, Dec 2008.

[78] M. Lopez de Bertodano, RT Lahey, and OC Jones. Phase distribution in bubbly two-phase flow in vertical ducts. *International Journal of Multiphase Flow*, 20(5):805–818, 1994.

[79] Dmitry M Malioutov, Müjdat Cetin, and Alan S Willsky. Homotopy continuation for sparse signal representation. In *Acoustics, Speech, and Signal Processing, 2005. Proceedings.(ICASSP'05). IEEE International Conference on*, volume 5, pages v–733. IEEE, 2005.

[80] Annalisa Manera, Horst-Michael Prasser, and Tim HJJ VAN DER HAGEN. Suitability of drift-flux models, void-fraction evolution, and 3-D flow pattern visualization during stationary and transient flashing flow in a vertical pipe. *Nuclear technology*, 152(1):38–53, 2005.

[81] G. Marrucci. Rising velocity of a swarm of spherical bubbles. *Industrial and Engineering Chemistry Fundamentals*, 4(2):224–225, 1965.

[82] Joaquim RRA Martins, Peter Sturdza, and Juan J Alonso. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[83] MATLAB. *version 7.14.0.739 (R2012a)*. The MathWorks Inc., Natick, Massachusetts, 2012.

[84] F. Mayinger and YM Chen. Heat transfer at the phase interface of condensing bubbles. In *Proc. of the 8th Int. Heat Transfer Conf*, volume 4, pages 1913–1918, 1986.

[85] Michael D McKay, Richard J Beckman, and William J Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[86] D Moro, NC Nguyen, and J Peraire. Navier-stokes solution using hybridizable discontinuous Galerkin methods. *AIAA paper*, 3407:2011, 2011.

[87] Deanna Needell and Joel A Tropp. Cosamp: iterative signal recovery from incomplete and inaccurate samples. *Communications of the ACM*, 53(12):93–100, 2010.

[88] Deanna Needell and Roman Vershynin. Signal recovery from incomplete and inaccurate measurements via regularized orthogonal matching pursuit. *Selected Topics in Signal Processing, IEEE Journal of*, 4(2):310–316, 2010.

[89] RI Nigmatulin, NS Khabeev, and FB Nagiev. Dynamics, heat and mass transfer of vapour-gas bubbles in a liquid. *International Journal of Heat and Mass Transfer*, 24(6):1033–1044, 1981.

[90] Todd A Oliver and David L Darmofal. Analysis of dual consistency for discontinuous Galerkin discretizations of source terms. *SIAM Journal on Numerical Analysis*, 47(5):3507–3525, 2009.

[91] Carsten Othmer. Adjoint methods for car aerodynamics. *Journal of Mathematics in Industry*, 4(1):6, 2014.

[92] Jaime Peraire and P-O Persson. The compact discontinuous Galerkin (CDG) method for elliptic problems. *SIAM Journal on Scientific Computing*, 30(4):1806–1824, 2008.

[93] M.Z. Podowski and R.M. Podowski. Mechanistic multidimensional modeling of forced convection boiling heat transfer. *Science and Technology of Nuclear Installations*, 2009. Article ID 387020.

[94] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[95] Rolf Rannacher. Adaptive Galerkin finite element methods for partial differential equations. *Journal of Computational and Applied Mathematics*, 128(1):205–233, 2001.

[96] W.E. Ranz and W.R. Marhsall. Evaporation from drops – part i and ii. *Chemical Engineering Progress*, 48, 1952.

[97] W.E. Ranz and W.R. Marshall. Evaporation from drops – parts I and II. *Chemical Engineering Progress*, 48(3):141, 1952.

[98] W. Rodi. *The prediction of free turbulent boundary layers by use of a two-equation model of turbulence*. PhD thesis, University of London, 1972.

[99] Eva Romeo, Carlos Royo, and Antonio Monzón. Improved explicit equations for estimation of the friction factor in rough and smooth pipes. *Chemical engineering journal*, 86(3):369–374, 2002.

[100] Trent Michael Russi. *Uncertainty quantification with experimental data and complex system models*. PhD thesis, University of California, Berkeley, 2010.

[101] Wilhelmus HA Schilders, Henk A Van der Vorst, and Joost Rommes. *Model order reduction: theory, research aspects and applications*, volume 13. Springer, 2008.

[102] D.R. Shaver, S.P. Antal, M.Z. Podowski, and D. Kim. Direct steam condensation modeling for a passive pwr safety system. In *OECD Nuc. Energy Agency and IAEA Workshop, Sept. 2010, Washington D.C.*, 2010.

[103] Jie Shen. Efficient spectral-Galerkin method i. direct solvers of second-and fourth-order equations using Legendre polynomials. *SIAM Journal on Scientific Computing*, 15(6):1489–1505, 1994.

[104] M. Simonnet, C. Gentric, E. Olmos, and N. Midoux. Experimental determination of the drag coefficient in a swarm of bubbles. *Chemical Engineering Science*, 62:858–866, 2007.

[105] Jin Ho Song and M Ishii. The well-posedness of incompressible one-dimensional two-fluid model. *International Journal of Heat and Mass Transfer*, 43(12):2221–2231, 2000.

[106] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural processing letters*, 9(3):293–300, 1999.

[107] Takashi Takata and Akira Yamaguchi. Numerical approach to the safety evaluation of sodiumwater reaction. *Journal of Nuclear Science and Technology*, 40(10):708–718, 2003.

[108] Boxin Tang. Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993.

[109] V.I. Tolubinsky and D.M. Kostanczuk. Vapour bubbles growth rate and heat transfer intensity at subcooled water boiling. 4th international heat transfer conference, volume 5, 1970.

[110] A. Tomiyama, I. Kataoka, T. Fukuda, and T. Sakaguchi. Drag coefficients of bubbles: 2nd report, drag coefficient for a swarm of bubbles and its applicability to transient flow. *Transactions of JSME*, 61(588):2810–2817, 1995.

[111] Akio Tomiyama, Hidesada Tamai, Iztok Zun, and Shigeo Hosokawa. Transverse migration of single bubbles in simple shear flows. *Chemical Engineering Science*, 57:1849–1858, 2002.

[112] ET Tomlinson and DL Aumiller. An assessment of relap5-3d using the edwards-o'brien blowdown problem. Technical report, Bettis Atomic Power Lab., West Mifflin, PA (United States). Funding organisation: US Department of Energy (United States), 1999.

[113] Balakrishnan Varadarajan, Sanjeev Khudanpur, and Trac D Tran. Stepwise optimal subspace pursuit for improving sparse recovery. *Signal Processing Letters, IEEE*, 18(1):27–30, 2011.

[114] David A Venditti and David L Darmofal. Adjoint error estimation and grid adaptation for functional outputs: Application to quasi-one-dimensional flow. *Journal of Computational Physics*, 164(1):204–227, 2000.

[115] Tolubinsky V.I. and Kostanchuk D.M. Vapour bubbles growth rate and heat transfer intensity at subcooled water boiling. In *4th International Heat Transfer Conference, Paris*, 1970. Reprinted in Heat Transfer, 1970, Paper No. B-2.8.

[116] D.M. Wang. Modeling of bubbly flow in a sudden pipe expansion. Technical report II-34, BRITE/EuRam project BE-4098, 1994.

[117] D.M. Wang. Modelling of bubbly flow in a sudden pipe expansion. Technical report, BRITE/EuRam Project BE-4098, 1994. Report II-34.

[118] G.R. Warrier, N. Basu, and V.K. Dhir. Interfacial heat transfer during subcooled flow boiling. *International journal of heat and mass transfer*, 45(19):3947–3959, 2002.

[119] Dongbin Xiu and George Em Karniadakis. The wiener–askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644, 2002.

[120] Shuyuan Yang, Min Wang, and Licheng Jiao. Ridgelet kernel regression. *Neurocomputing*, 70(16):3046–3055, 2007.

[121] Shuyuan Yang, Min Wang, and Licheng Jiao. A linear ridgelet network. *Neurocomputing*, 73(1):468–477, 2009.

[122] W. Yao and C. Morel. Volumetric interfacial area prediction in upward bubbly two-phase flow. *International Journal of Heat and Mass Transfer*, 47(2):307–328, 2004.

[123] O. Zeitoun, M. Shoukri, and V. Chatoorgoon. Interfacial heat transfer between steam bubbles and subcooled water in vertical upward flow. *Journal of heat transfer*, 117:402, 1995.