Development of a Spatially-Selective, Nonlinear Refinement Algorithm for Thermal-Hydraulic Safety Analysis

By

Lewis John Lloyd

A dissertation submitted in partial fulfillment of

the requirements for the degree of

Doctor of Philosophy

(Nuclear Engineering and Engineering Physics)

at the

UNIVERSITY OF WISCONSIN–MADISON

2014

Date of final oral examination: 02/28/2014

The dissertation is approved by the following members of the Final Oral Committee:

Michael Corradini, Professor, Nuclear Engineering and Engineering Physics
Gregory Moses, Professor, Nuclear Engineering and Engineering Physics
Robert Witt, Associate Professor, Nuclear Engineering and Engineering Physics
Christopher Rutland, Professor, Mechanical Engineering
David Aumiller, Senior Advisor Engineer, Bettis Laboratory

Acknowledgments

This research was performed under appointment to the Rickover Fellowship Program in Nuclear Engineering sponsored by Naval Reactors Division of the U.S. Department of Energy.

Table of Contents

Page

Page

List of Tables

List of Figures

Figure Page

Figure Page

List of Algorithms

Nomenclature

Acronyms

10CFR	Title 10 of the Code of Federal Regulations
10CFR50	Part 50 of 10CFR
CFL	Courant-Friedrichs-Lewy
GE	General Electric
LOCA	Loss-of-Coolant Accident
LWR	Light Water Reactor
NBC	Nonlinear-Boundary Continuity
NPP	Nuclear Power Plant
NRC	Nuclear Regulatory Commision
PDE	Partial Differential Equation
PVM	Parallel Virtual Machine
SAR	Safety Analysis Report
SETS	Stability Enhancing Two-Step

Greek Symbols

Symbol	**Description**	**Units**
α	Volume fraction.	
$\delta\mathbf{\Psi}$	Flow rate updates.	
$\delta\mathbf{P}$	Pressure updates for a domain.	
$\delta\mathbf{x}$	Vector of updates to the independent parameters.	

δ_{tol}	Convergence tolerance with respect to the relative update vector.	
Δt	Timestep between time t^n and t^{n+1}.	s
Δt_{vol}	Timestep as calculated by volume CFL limit.	s
Δt_{crnt}	Timestep as calculated by CFL limit.	s
Δt_{MAX}	Maximum timestep allowed for a simulation.	s
Δx	Axial length of a volume.	ft
η	Apportionment factor for inter-phase mass transfer.	
Γ	Inter-phase source or sink of mass.	lb_m
$\Gamma \mathbf{u}'$	Momentum transfer due to the exchange of mass between the aqueous phases.	$\text{lb}_\text{f}\,\text{s}$
$\Gamma h'_\phi$	Energy transfer rate due to aqueous phase change.	BTU
Ω	The domain; the set of volumes that comprises the domain.	
Ψ	A given flow rate.	
ρ	Density.	$\text{lb}_\text{m}\,\text{ft}^{-3}$
$\tau'_{i,\phi_1\phi_2}$	Shear force from contact between the field or phase ϕ_1 and ϕ_2.	$\text{lb}_\text{f}\,\text{ft}^{-3}$
$\tau'_{w,\phi}$	Shear force from contact between the channel wall and fluid ϕ.	$\text{lb}_\text{f}\,\text{ft}^{-3}$
Υ	Inter-field field source or sink of mass.	lb_m
$\Upsilon \mathbf{u}'$	Momentum transfer due to the exchange of mass between the two liquid fields.	$\text{lb}_\text{f}\,\text{s}$
$\mathbf{\Psi}$	Vector of flow rates.	
$\mathbf{\Xi}$	Matrix of coefficients that converts the momentum vector to the flow rate vector.	

Roman Symbols

Symbol	Description	Units
$\dot{m}(t)$	Specified mass flow at boundary.	$\text{lb}_\text{m}\,\text{s}^{-1}$
$\dot{m}_\phi$	The conserved quantity for a momentum field or phase equation.	$\text{lb}_\text{m}\,\text{s}^{-1}$
$k_{\,\text{max}}$	Maximum iteration count for nonlinear solver.	
$\mathcal{L}_2$	Euclidean norm of a given vector.	
$\mathcal{L}_\infty$	Infinity norm of a given vector.	
$\dot{\mathbf{m}}$	Vector of three momenta fields.	$\text{lb}_\text{m}\,\text{s}^{-1}$

$\mathbf{A}$	Pressure matrix.	
$\mathbf{a}$	Vector of constants from right-hand-side of momentum system.	
$\mathbf{B}$	Matrix representing pressure matrix inter-domain coupling coefficients.	
$\mathbf{b}$	Vector of pressure-update coefficients from momentum system.	
$\mathbf{C}$	Vector of unknowns for a given continuity volume.	
$\mathbf{E}$	Vector of spatially discrete conservation equations.	
$\mathbf{e}$	Vector of conservation equations, except the temporal derivatives.	
$\mathbf{E}^*$	Approximation of temporal integral of $\mathbf{E}(\mathbf{y}(\mathbf{x}))$.	
$\mathbf{F}$	Vector of nonlinear residuals.	
$\mathbf{g}$	Gravitational acceleration vector.	$\mathrm{ft\,s^{-2}}$
$\mathbf{I}$	Identity matrix.	
$\mathbf{J}$	A given Jacobian matrix.	
$\mathbf{K}$	Matrix representing inter-continuity volume pressure coupling coefficients.	
$\mathbf{L}$	Lower triangular matrix from $\mathbf{LU}$ decomposition.	
$\mathbf{Q}$	Matrix representing inter-domain flow rate coefficients.	
$\mathbf{res}$	Right hand side of pressure system.	
$\mathbf{r}$	Vector of right-hand sides for a given continuity volume.	
$\mathbf{S}$	Vector of operator based scale factors.	
$\mathbf{U}$	Upper triangular matrix from $\mathbf{LU}$ decomposition.	
$\mathbf{u}$	Vector of phasic velocities for a given flow path.	$\mathrm{ft\,s^{-1}}$
$\mathbf{W}$	Matrix representing global inter-domain coupling coefficients.	
$\mathbf{w}$	Vector representing inter-domain coupling coefficients for a given NBC volume.	
$\mathbf{x}$	Vector of nine independent variables used by COBRA.	
$\mathbf{y}$	Vector of conserved quantities.	
$\mathbf{Z}$	Matrix representing a linear domain continuity volume's linear system.	
c	Local speed of sound.	$\mathrm{ft\,s^{-1}}$
C_1	Coefficient in phase transition function.	

e_p	Error in calculated pressure	psia
$f(x)$	A generic function.	
F_{tol}	Convergence tolerance with respect to the scaled residual vector.	
h	Specific enthalpy.	BTU lb_m^{-1}
$K(t)$	Effective time dependent loss coefficient in valve problem.	$\text{lb}_\text{f}\,\text{s}\,\text{ft}^{-1}$
$K_{i,\phi_1\phi_2}$	Effective coefficient for drag between fluids ϕ_1 and ϕ_2.	$\text{lb}_\text{f}\,\text{s}\,\text{ft}^{-1}$
K_o	Base loss coefficient in valve problem.	$\text{lb}_\text{f}\,\text{s}\,\text{ft}^{-1}$
$K_{w,\phi}$	Effective coefficient for drag between wall and fluid ϕ.	$\text{lb}_\text{f}\,\text{s}\,\text{ft}^{-1}$
$m_{k,j}^{n+1}$	Flow of mass for phase k in RELAP5-3D.	$\text{lb}_\text{m}\,\text{s}^{-1}$
$n_{g,j}^{n+1}$	Flow of non-condensable gas mass in RELAP5-3D.	$\text{lb}_\text{m}\,\text{s}^{-1}$
N_{CPL}	Number of COBRA – RELAP5-3D coupling interfaces.	
N_{lin}	Number of continuity volumes in the linear domain.	
N_{NBC}	Set of inter-domain connection for a linear continuity volumes.	
N_{nln}	Number of continuity volumes in the nonlinear domain.	
N_c	Set of continuity volumes connected to a given flow path.	
N_f	Set of flow paths connected to a given continuity volume.	
N_n	Number of inter-domain connections in the global domain.	
N_s	Number of stages in a multi-stage temporal integration scheme.	
N_t	Number of successful timesteps in a simulated time span.	
n_t	Number of time step refinements in timestep generation function.	
N_u	Number of variables in a residual or update vector.	
P	Pressure.	psia
$q_{i,\phi}$	Energy transferred between the saturated interface and a phase ϕ.	BTU
$q_{n,l}$	Energy transferred between the non-condensable gas field and the liquid water phase.	BTU
$q_{w,\phi}$	Energy transferred between a solid structure and a fluid ϕ.	BTU
r_f	Refinement factor for timestep generation function.	
S	Scale factor for an equation.	
s_h	Source or sink of external energy.	BTU

s_m	Source or sink of external mass.	$\mathrm{lb_m}$
s_p	Source or sink of external momentum.	$\mathrm{lb_f\,s}$
T	End time of simulation.	s
t	Time.	s
T_{CPU}	CPU time taken for a simulation to run.	s
u	A generic velocity.	$\mathrm{ft\,s^{-1}}$
$u_{k,j}^{n+1}$	Flow of internal energy for phase k in RELAP5-3D.	$\mathrm{BTU\,s^{-1}}$
V	Volume of a given continuity volume.	$\mathrm{ft^3}$
v	Specific volume.	$\mathrm{ft^3\,lb_m}$
$w_{k,j}^{n+1}$	Volumetric flow for phase k in RELAP5-3D.	$\mathrm{ft^3\,s^{-1}}$
x	Spatial location.	ft
$\tilde{A}$	Effective cross-sectional area between two flow paths.	$\mathrm{ft^2}$
A	Cross-sectional area.	$\mathrm{ft^2}$
$A(t)_r$	Time dependent area ratio in valve problem.	

Subscripts

ϕ	A generic field or phase depending upon context.
*	An intermediate or tentative value.
a	Averaged quantity.
ana	Analytic.
c	Relating to a continuity volume.
car	Carrier phase.
cbv	Continuity boundary volume between RELAP5-3D and COBRA.
d	Donored quantity.
dep	Depleting phase.
e	Indicating the entrained liquid field.
g	Indicating the gaseous phase.
h	Relating to a continuity volume's energy equations.
i	Index representing a generic coordinate.

j	The coordinate of a particular location in a given problem.
l	Indicating either the continuous liquid field or the total liquid phase.
lin	Relating to the linear domain or solver.
m	Relating to a continuity volume's mass equations.
n	Indicating the non-condensable gas component of the gaseous phase.
nln	Relating to the nonlinear domain or solver.
o	Base value.
o(i)	Coordinate of other continuity volume connected via a flow path at coordinate i.
p	Relating to a momentum flow path.
r	Relative.
s(i)	Coordinate of current continuity volume connected via a flow path at coordinate i.
v	Indicating the vapor component of the gaseous phase.

Superscripts

‴	Per unit volume.
C_2	Coefficient in phase transition function.
.	Per second, a denotation of a rate.
b	The time point at which the donored quantities are evaluated in the donoring operator.
c	The time point at which the donoring velocity is evaluated in the donoring operator.
k	Newton iteration index.
n	Time index.
p	Generic exponent.
s	Temporal integral stage index.
	Scaled vector.

Chapter 1

Introduction

In the United States, the goal of Nuclear Power Plant (NPP) designers, builders, operators, and regulators is to ensure the safety of the public during both normal operations and postulated accident scenarios. It is the responsibility of the Nuclear Regulatory Commission (NRC) to issue licenses for the construction and operation of nuclear reactors. Chapter 1 of Title 10 of the Code of Federal Regulations (10CFR) details the regulatory procedures that govern the NRC. Part 50 of 10CFR (10CFR50) lays out the process by which an applicant can obtain both construction and operating licenses for NPPs. One of the documents required prior to the issuance of any license under 10CFR50 is a safety analysis report (SAR). In part, SARs for Light Water Reactors (LWRs) require the applicant to provide an evaluation of their emergency core cooling systems during postulated loss-of-coolant accidents (LOCA). This evaluation must conform with section 46 of 10CFR50, which requires the applicant to perform analyses for “a number of postulated loss-of-coolant accidents of different sizes, locations, and other properties sufficient to provide assurance that the most severe postulated loss-of-coolant accidents are calculated" [1]. This requires designers and operators of NPPs to model the thermal-hydraulic behavior within the core of a reactor during postulated LOCAs. The diverse physical conditions experienced by the reactor during postulated accidents necessitate the inclusion of a range of complex physics during safety analyses. Each of these physics has dedicated pieces of software that are under continual development to improve their predictive capabilities. The work that follows is concerned with the mathematical formulation and solution of the equations governing the thermal-hydraulic behavior within the reactor core.

1.1 Design Motivation

All commercial reactors operating within the United States are of an LWR design. The safety analyses required for licensing necessitate the modeling of water in both liquid and gaseous phases. This has driven the development of safety software that can model the behavior of water under an extensive range of thermodynamic states, including multiple phases. Several codes are widely available for simulating the thermal-hydraulic response of an NPP during postulated accidents. These safety codes can be divided into two large categories: system analysis codes and subchannel analysis codes. While there is considerable overlap between the capabilities of these two categories, each has its own particular strengths and weaknesses. Three well-known system-level safety analysis software packages are RELAP5-3D [42], TRACE [46], and MELCOR [40]. Two well-known subchannel analysis software packages are COBRA [43] and VIPRE. A brief description of these two types of software is provided below.

The system analysis software has historically been used for complete plant analysis. These software systems have extensive collections of empirical models for pumps, valves, accumulators, etc. These models allow for a macroscopic description of the balance of plant and its interaction with the reactor core. The reactor core itself is normally a low-fidelity representation that operates primarily as a heat source for the balance of plant model. While the system software can be successfully used to model the balance of plant, more detailed physical models and hydrodynamics are required to accurately provide details of in-core behavior.

To address the need for accurate simulations of accident scenarios, software capable of modeling the dominant physics of interest within a reactor core was developed. This is the second type of software: subchannel analysis. The use of subchannel analysis software stems from physical arguments about dominant flow within a LWR. To motivate these arguments, a brief discussion of the geometry within the nuclear reactor is presented. The basic geometry of commercial nuclear reactor cores in the United States is a vertical, cylindrical array of fuel bundles, control rods, and instrumentation. Within a fuel bundle cylindrical fuel rods are arranged into a lattice. The presence of concentric geometric structures of axially aligned units creates natural subchannels for coolant

flow. This is the basis for one of the underlying assumptions of subchannel analysis software: the axial flow is the dominant flow. This assumption allows for reduced complexity when modeling the inter-subchannel transverse flow. In the work that follows the governing physics and the computational framework of interest will be taken from a variant of the aforementioned COBRA subchannel analysis code, which shall be referred to generically as COBRA.

1.2 Research Objectives

The methods used to simulate thermal-hydraulic behavior in NPPs are characterized by the manner in which the governing conservation equations are integrated over space and time and how their nonlinearities are resolved. Each of the methods outlined in Section 2.4 has one thing in common: the governing equations are solved over the global domain, either through a single linear approximation or through iterative, nonlinear convergence.

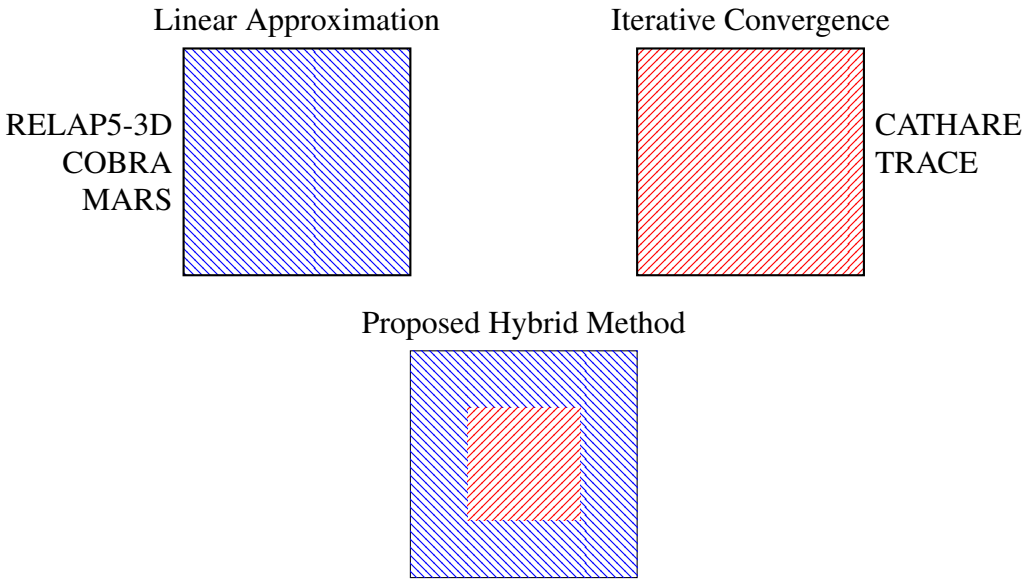

Figure 1.1 Current and proposed paradigms for thermal-hydraulic safety analysis.

The advantage of the linear approximation is the reduction in computational costs; however, the accuracy of this approximation at large timestep sizes in regions of highly nonlinear physics

is suspect. This limitation has traditionally been mitigated by restrictions placed upon the maximum permissible change in the independent parameters during a given timestep, which can in turn lead to excessively small timestep sizes. The benefit of an iterative solver is that the nonlinear physics are resolved at each timestep by using multiple iterations. In trade, the computational cost of those multiple iterations is high. This research seeks to find a balance between the competing incentives of computational cost and accuracy by creating a hybrid method, as illustrated in Figure 1.1. Specifically, if there are parts of the domain with a high degree of nonlinear physics that are spatially isolable, then the computational cost of solving the entire domain's governing equations iteratively may be unnecessary. Instead, by restricting the multiple iterations to only those portions of the global domain where they are necessary, the computational cost will be reduced while maintaining accuracy.

The objectives of this research were the design, implementation, and evaluation of a novel, spatially selective, nonlinear solution method for nuclear thermal-hydraulic safety analysis. Isolation of a specified subdomain where nonlinearities are high was achieved by developing a new variant of the code-coupling algorithm discussed in Section 2.5. For this work, the subdomain to be isolated was comprised of geometric components predetermined by the user. Upon isolation, the nonlinear subdomain was subjected to Newton's method to resolve any nonlinearities at every timestep. This converged solution was then communicated via coupling coefficients to the remainder of the domain for use in finishing its single Newton step. This unique use of domain decomposition for selective nonlinear-refinement via semi-implicit coupling provides a route for obtaining nonlinearly converged, timestep-size insensitive solutions for traditional two-phase flow methods with a lower computational cost than that of a globally iterative Newton's method. The following chapters will provide an overview of: relevant background material and research, Chapter 2; the work that was done to allow this research to be carried out, Chapter 3; the domain decomposition algorithm, Chapter 4; and the results of numerical experiments, Chapter 5.

Chapter 2

Background Material

This chapter provides a review of mathematical and theoretical material pertinent to this research. First, the geometric discretization used in this work is discussed. Second, the two-phase flow equations are detailed. Third, the numerical approximations with respect to space, time, and nonlinearities are covered. Next, the various solution methods currently used to solve the system of discrete governing equations are presented. Last, an outline of the various means by which domain coupling of thermal-hydraulic safety analysis software has traditionally been achieved is provided.

2.1 Geometric Discretization

The first step in any simulation of the thermal-hydraulic behavior of the core of a nuclear reactor is to create a discrete computational representation of the domain. For this work the computational grid used to represent the physical model is a staggered mesh. With a staggered mesh the domain is divided into two overlapping meshes. There is one mesh comprised of continuity volumes and another comprised of momentum flow paths, as shown in Figure 2.1. Thermodynamic variables are defined as constant over the continuity volumes, while momenta are defined as constant over momentum flow paths. By definition the momentum flow paths do not contain mass or energy. The boundaries of the continuity volumes align with the center of the momentum flow paths.

The geometric modeling framework in this work involves two primary components: sections and subchannels. The total axial height of the problem is divided into sections. Each section is defined by two bounding elevations and a spatial discretization of that span into discrete non-overlapping axial segments that represent the axial spacing of the continuity volumes. The total

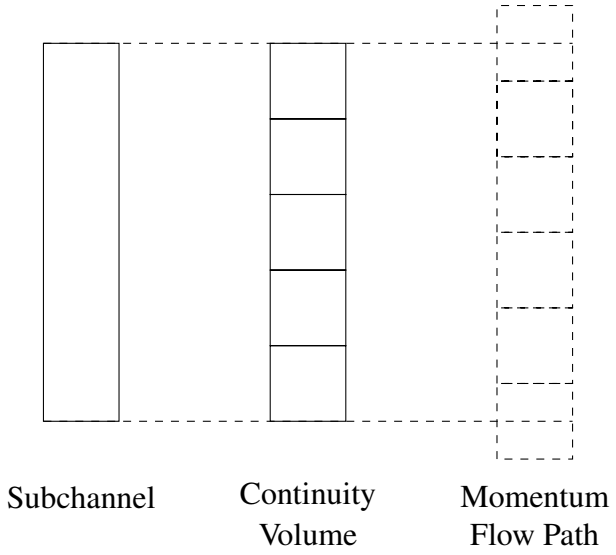

Figure 2.1 A representation of the staggered mesh.

number of continuity volumes in a given subchannel is one less than the number of momentum flow paths in that subchannel. The particular axial discretization of a given section is independent from that of the other sections.

Within a given section, subchannels are defined. A subchannel inherits the axial discretization of its parent section. The number of subchannels in a given section is independent of the number of subchannels in other the sections. Each continuity volume and momentum flow path has an associated cross-sectional area, A_c and A_p respectively. These cross-sectional areas can vary from volume to volume and from flow path to flow path.

It is also possible to model multi-dimensional flow through the definition of transverse flow paths that connect two subchannels within a given section. The transverse flow paths are orthogonal to the axial flow direction. Additionally, it is permissible to have the axial flow from a single subchannel split into multiple subchannels in an adjacent section. This flow splitting is only permitted at section boundaries. By using these features geometrically complex models can be created. An example of a model that includes these different geometric characteristics is given in Figure 2.2.

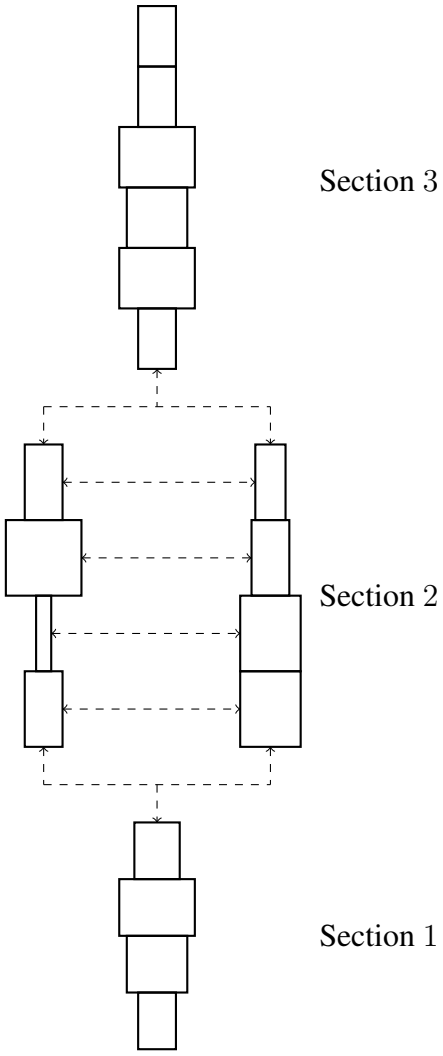

Figure 2.2 A model with transverse flow paths and complex geometric characteristics.

2.2 Two-Phase Flow Equations

The primary purpose of subchannel analysis codes is to determine fuel integrity via evaluation of effective core cooling during postulated NPP accidents, such as a LOCA. This requires accurate modeling of the heat transfer between the coolant and the fuel. During postulated accidents, the coolant, H_2O, will undergo phase-change. There are several formulations of the governing conservation equations of fluid mechanics used to predict the thermal-hydraulic response of the nuclear reactor core to transient plant conditions. To model the complex phenomenon of phase-change,

the governing equations for the fluid mechanics within the core will be those of a multicomponent fluid [14]. In particular, they are a subcategory of two-phase flow [19, 39, 44]. This section will discuss both the assumptions used and the final governing equations obtained for modeling two-phase flow.

2.2.1 Assumptions

The governing equations of fluid mechanics provide extensive information about fluid behavior. However, one of the basic assumptions of subchannel analysis is that the exact spatial behavior of the coolant is not of direct interest; as such, the average behavior of the fluid is what is modeled. Given the multi-phase nature of the coolant, the interactions between the various phases can be modeled in various ways. For the conditions encountered in a commercial nuclear reactor core, certain assumptions can be made to reduce the complexity of the physics involved. This section will discuss the technique used to obtain equations for averaged quantities, the manner in which the multi-phase nature of the coolant is dealt with, and the general assumptions used to reduce the complexity of the physics.

The first simplification is that the exact topology of the interface between the coolant phases need not be known to model the fluid mechanics. Since the exact deterministic behavior of the phasic-interface is no longer necessary, the governing equations can be subjected to averaging procedures to produce conservation laws for averaged quantities. Several averaging techniques have been used to produce the conservation laws for two-phase flow: spatial, temporal, and ensemble averaging [14, 44]. Each of these techniques has its own physical interpretation and mathematical formulation. The formulation used in this work is area averaging, a particular form of spatial averaging. In the area-averaged formulation the governing equations are averaged over the cross-sectional area.

The second simplification regards the manner in which the multi-phase nature of the coolant is treated. To simulate the behavior of in-core fluid dynamics during accident scenarios more accurately, the liquid and the gaseous phases are each divided into two distinct fields. The two fields of the liquid phase are a continuous liquid field and an entrained liquid droplet field. The gaseous

phase is composed of a mixture of a non-condensable gas field and a water-vapor field. This ability to track the different fields within a phase provides two important benefits to safety analysis: the ability to account for the effects of non-condensable gases on condensation and the ability to model the effects of the entrained liquid droplets on heat transfer. The benefits of modeling the entrained liquid field have inspired the developers of the French CATHARE software to migrate to a similar three-field formulation in the next version of their software [16].

Given the four fields of interest, there are twelve conservation equations in axial flow, one each for the mass, momentum, and energy of each of the four fields. Additional closure relationships are necessary to describe the interactions between the various fields and their interfacial transfer terms. These equations allow for a description of the time-dependent behavior of the in-core fluid. However, assumptions are made to reduce the number of required conservation laws and the number of corresponding closure relationships. The following is a list of the assumptions that are made to reduce the complexity of the governing equations.

- Thermodynamic and pressure equilibrium exists between the continuous liquid field and the entrained liquid field. The basis for this assumption is that, while the entrained droplets are being modeled by a separate set of governing equations than those of the continuous liquid field, the droplets are constantly entraining from and depositing to the continuous liquid field.
- The liquid and gaseous phases are assumed to be in pressure equilibrium with the interface between the phases. The basis for this assumption is that the inter-phase dynamics at the interface are negligible when compared to the bulk flow dynamics.
- The non-condensable gas and steam components of the gaseous phase are in mechanical equilibrium.
- The two components of the gaseous phase obey Dalton's Law.
- The two components of the gaseous phase are in thermal equilibrium.
- The viscous dissipation of momentum in the axial flow direction and the associated generation of energy are neglected. This assumption is made because inertial, not viscous, forces

dominate the simulations of interest so that the neglected terms would be small compared to other terms in the calculations.

- The wall-shear effects of viscosity are accounted for via empirically based friction correlations.
- The primary positive flow direction is counter to the gravity vector, which will be referred to as the axial flow direction. This vertically oriented coordinate system was chosen to accommodate the vertical core design of NPPs in the United States.
- The mechanical energy of the phases is neglected in the conservation of energy. The assumption is that in the simulations of interest the mechanical energy term is negligible.
- The effects of turbulence are only incorporated through the closure relationships used for wall friction and wall heat transfer.

2.2.2 Governing Equations

Following the application of the above assumptions, nine governing partial differential equations (PDEs) for axial flow remain: four for mass conservation, two for energy conservation, and three for momentum conservation. The details of this system of PDEs are discussed below.

2.2.2.1 Conservation of Mass Equations

There are four equations that represent the conservation of mass, (2.1) – (2.4), one each for the non-condensable gas, continuous liquid, entrained liquid, and vapor fields respectively.

$$\frac{\partial\left(\alpha_g\rho_n\right)}{\partial t}+\nabla\cdot\left(\alpha_g\rho_n\mathbf{u}_g\right)=\dot{s}'''_{m,n} \tag{2.1}$$

$$\frac{\partial\left(\alpha_l\rho_l\right)}{\partial t}+\nabla\cdot\left(\alpha_l\rho_l\mathbf{u}_l\right)=-(1-\eta)\dot{\Gamma}'''-\dot{\Upsilon}'''+\dot{s}'''_{m,l} \tag{2.2}$$

$$\frac{\partial\left(\alpha_e\rho_l\right)}{\partial t}+\nabla\cdot\left(\alpha_e\rho_l\mathbf{u}_e\right)=-\eta\dot{\Gamma}'''+\dot{\Upsilon}'''+\dot{s}'''_{m,e} \tag{2.3}$$

$$\frac{\partial\left(\alpha_g\rho_v\right)}{\partial t}+\nabla\cdot\left(\alpha_g\rho_v\mathbf{u}_g\right)=\dot{\Gamma}'''+\dot{s}'''_{m,v} \tag{2.4}$$

The left-hand sides of (2.1) – (2.3) represent the Lagrangian derivative for the given field. The terms on the right-hand side represent the volumetric rate of the inter-field ($\dot{\Upsilon}'''$), inter-phase ($\dot{\Gamma}'''$), and external ($\dot{s}'''_{m,\phi}$) mass transfer, where ϕ represents the relevant phase or field. Since there are two liquid fields, the rate of mass transfer between the water-vapor field and the liquid fields, $\dot{\Gamma}'''$, is apportioned between the continuous liquid field and the entrained liquid field. This relationship is shown in (2.5), where η is an apportionment factor.

$$\dot{\Gamma}''' = \eta\dot{\Gamma}''' + (1-\eta)\dot{\Gamma}''' \tag{2.5}$$

The inter-field transfer of mass occurs only between the continuous and entrained liquid fields, (2.6).

$$\dot{\Upsilon}'''_l + \dot{\Upsilon}'''_e = 0 \tag{2.6}$$

Within the conservation of mass equations several assumptions from Section 2.2.1 are evident. The mechanical equilibrium of the non-condensable gas and the vapor field manifests itself in the singular velocity for the two gaseous fields: $\mathbf{u}_g$, where the g subscript denotes the total gaseous phase. Dalton's Law allows the two components of the gaseous phase to occupy the same volume, thus providing for a single volume fraction, α_g. The thermodynamic equilibrium of the two liquid fields results in only one liquid density, ρ_l.

2.2.2.2 Conservation of Energy Equations

In addition to the conservation of mass equations there are conservation of energy equations for each of the two phases, (2.7) – (2.8).

$$\begin{aligned} \frac{\partial\left(\alpha_g\rho_g h_g\right)}{\partial t} + \nabla\cdot\left(\alpha_g\rho_g h_g\mathbf{u}_g\right) = \\ \dot{\Gamma}''' h_v' + \dot{q}'''_{i,v} + \dot{q}'''_{n,l} + \dot{q}'''_{w,g} + \alpha_g\frac{\partial P}{\partial t} + \dot{s}'''_{h,g} \\ \frac{\partial\left((1-\alpha_g)\rho_l h_l\right)}{\partial t} + \nabla\cdot\left(\alpha_l\rho_l h_l\mathbf{u}_l\right) + \nabla\cdot\left(\alpha_e\rho_l h_l\mathbf{u}_e\right) = \end{aligned} \tag{2.7}$$

$$-\dot{\Gamma}'''h_l' + \dot{q}_{i,l}''' - \dot{q}_{n,l}''' + \dot{q}_{w,l}''' + (1-\alpha_g)\frac{\partial P}{\partial t} + \dot{s}_{h,l}''' \tag{2.8}$$

The conservation of energy equations used in this work are formulated such that the conserved quantities are the phasic enthalpies, $\alpha_\phi \rho_\phi h_\phi$. Under the assumption of thermodynamic equilibrium for the two liquid fields, there is a single enthalpy for the two liquid fields. The gaseous phasic enthalpy, however, is defined according to (2.9).

$$\rho_g h_g = \rho_v h_v + \rho_n h_n \tag{2.9}$$

The various terms on the right hand sides of (2.7) and (2.8) are defined as follows:

- $\dot{\Gamma}'''h_\phi'$: energy transfer rate due to the phase change of water. The effective enthalpies, h_ϕ', are dependent upon the mechanism of phase change.
- $\dot{q}_{i,\phi}'''$: energy transfer rate between the liquid and vapor phases and the saturated interface.
- $\dot{q}_{n,l}'''$: energy transfer rate between the liquid phase and the non-condensable gases.
- $\dot{q}_{w,\phi}'''$: energy transfer rate between the solid-structures and a given phase.
- $\alpha_\phi \frac{\partial P}{\partial t}$: pressure work done by a given phase ϕ. The liquid volume fraction is the sum of the volume fractions of the continuous and the entrained liquid fields.
- $\dot{s}_{h,\phi}'''$: energy transfer rate between an external source and a given phase.

2.2.2.3 Conservation of Momentum Equations

Finally, there are three governing equations for the conservation of momentum: the continuous liquid field (2.10), the gaseous phase (2.11), and the entrained liquid droplet field (2.12). These equations are expressed in vector notation; however, only axial flow will be detailed below. The assumed mechanical equilibrium between the two gaseous fields enables the use of a single momentum conservation equation for the net gaseous phase.

$$\begin{aligned} &\frac{\partial\left(\alpha_l \rho_l \mathbf{u}_l\right)}{\partial t} + \nabla \cdot \left(\alpha_l \rho_l \mathbf{u}_l \mathbf{u}_l\right) = \\ &-\alpha_l \nabla P + \alpha_l \rho_l \mathbf{g} - \tau_{w,l}' + \tau_{i,gl}' - (1-\eta)\dot{\Gamma}'''\mathbf{u}' - \dot{\Upsilon}'''\mathbf{u}' + \dot{s}_{p,l}''' \end{aligned} \tag{2.10}$$

$$\frac{\partial\left(\alpha_g\rho_g\mathbf{u}_g\right)}{\partial t}+\nabla\cdot\left(\alpha_g\rho_g\mathbf{u}_g\mathbf{u}_g\right)= -\alpha_g\nabla P+\alpha_g\rho_g\mathbf{g}-\tau'_{w,g}-\tau'_{i,gl}-\tau'_{i,ge}+\dot{\Gamma}'''\mathbf{u}'+\dot{s}'''_{p,g} \tag{2.11}$$

$$\frac{\partial\left(\alpha_e\rho_l\mathbf{u}_e\right)}{\partial t}+\nabla\cdot\left(\alpha_e\rho_l\mathbf{u}_e\mathbf{u}_e\right)= -\alpha_e\nabla P+\alpha_e\rho_l\mathbf{g}-\tau'_{w,e}+\tau'_{i,ge}-\eta\dot{\Gamma}'''\mathbf{u}'+\dot{\Upsilon}'''\mathbf{u}'+\dot{s}'''_{p,l} \tag{2.12}$$

The ρ_g used in (2.11) is defined by (2.13).

$$\rho_g=\rho_n+\rho_v \tag{2.13}$$

The left-hand sides of (2.10) – (2.12) represent the material derivative of the fluid momentum. The right-hand sides represent various surface, boundary, and body forces that act upon the fluid. The various terms in (2.10) – (2.12) are described below.

- $\alpha_\phi\nabla P$: pressure gradient acting on field or phase ϕ.
- $\alpha_\phi\rho_\phi\mathbf{g}$: gravity body force acting upon field or phase ϕ.
- $\tau'_{w,\phi}$: shear forces from contact between field or phase ϕ and the subchannel walls.
- $\tau'_{i,\phi_1\,\phi_2}$: shear forces from the interface between one field or phase ϕ_1 and another ϕ_2.
- $\dot{\Gamma}'''\mathbf{u}'$: momentum contribution from the exchange of mass due to the phase change of water. The $\mathbf{u}'$ term depends upon the net inter-phase transfer of mass.
- $\dot{\Upsilon}'''\mathbf{u}'$: momentum contribution from the exchange of mass between the two liquid fields. The $\mathbf{u}'$ term depends upon the net inter-field transfer of mass.
- $\dot{s}'''_{p,\phi}$: momentum transfer rate between an external source and a given field ϕ.

Note that the momentum conservation equations are formulated such that the temporal derivatives are of the conserved quantities, which is referred to as a conservative formulation. This formulation will be retained during the numerical discretization process. Other common system analysis codes [42, 46] use different, non-conservative variants of the above equations for their momentum conservation laws. Both formulations are known to have computational strengths and weaknesses [33].

2.2.3 Summary

It is useful to refer to the collection of continuous conservation laws, (2.1) – (2.3), (2.7) – (2.8), and (2.10) – (2.12), as a vector of equations. To accomplish this, (2.14) defines such a system. The vector $\mathbf{e}$ represents the conservation equations minus the temporal derivatives of the conserved quantities.

$$\frac{\partial \mathbf{y}}{\partial t} = \mathbf{e}\left(\mathbf{y}\left(t\right)\right) \tag{2.14}$$

(2.15) defines the vector, $\mathbf{y}$, of conserved quantities.

$$\mathbf{y} = [\alpha_g \rho_n,\ \alpha_g \rho_v,\ \alpha_l \rho_l, \alpha_e \rho_l, \alpha_g \rho_g h_g, \alpha_l \rho_l h_l,\ \alpha_g \rho_g \mathbf{u}_g,\ \alpha_l \rho_l \mathbf{u}_l,\ \alpha_e \rho_l \mathbf{u}_e]^T \tag{2.15}$$

2.3 Numerical Approximations

The set of conservation laws, (2.14), governing the fluid mechanics within the geometry of interest does not have a general closed form solution. As such, methods need to be selected to approximate the spatial, temporal, and nonlinear behavior of its numerical solution.

2.3.1 Spatial Approximations

In this work the governing equations are discretized using the finite-volume method [24]. The discussion that follows is specific to COBRA, but other safety analysis software use similar procedures [42, 46].

The labeling scheme for the finite volumes is shown in Figure 2.3, which is a segment of a subchannel. Two momentum flow paths, $j + \frac{1}{2}$ and $j - \frac{1}{2}$, spatially overlap a continuity volume, j. Variables are indexed by the mesh on which their conservation equations are defined. For example, the velocity $u_{j-\frac{1}{2}}$ would be spatially located at center of the momentum flow path $j - \frac{1}{2}$ and at the boundary between the continuity volumes j and $j - 1$.

Recalling the staggered mesh from Section 2.1, the six scalar conservation laws, (2.1) – (2.8), are each integrated over the continuity volumes. The assumption in these integrals is that the value

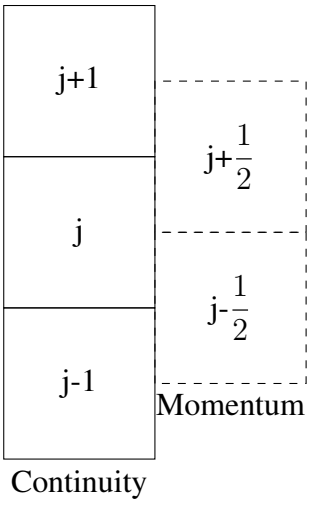

Figure 2.3 Illustration of indexing scheme.

of the conserved quantities and all thermodynamically related variables are a constant, average value within a given continuity volume. Figure 2.4 shows a graphical representation of this idea for a generic function $f(x)$ over several spatial continuity volumes.

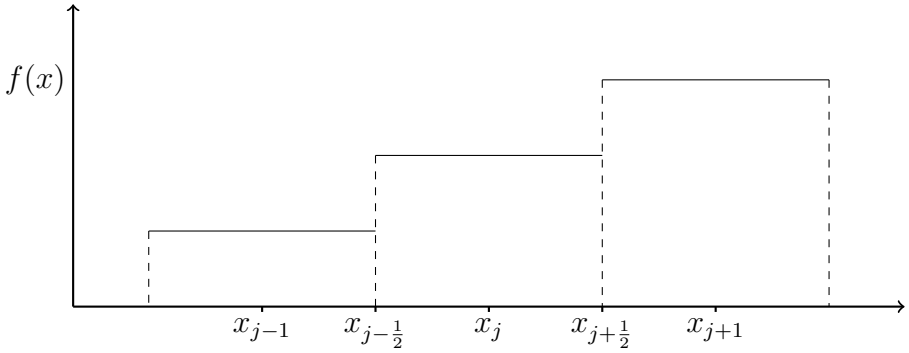

Figure 2.4 Constant variable values within computational volumes.

Within a subchannel, a given continuity volume of length Δx has a constant cross-sectional area, A_c, over its length. The cross-sectional area of the boundary between two continuity volumes is given by the cross-sectional area of the momentum path at that boundary, A_p.

For illustrative purposes (2.2) will be integrated over a simplified volume, V_j, consisting of singly connected axial flow as shown in Figure 2.5. This volume-integrated equation is given by (2.16). The same general procedure is used for the other five scalar conservation equations.

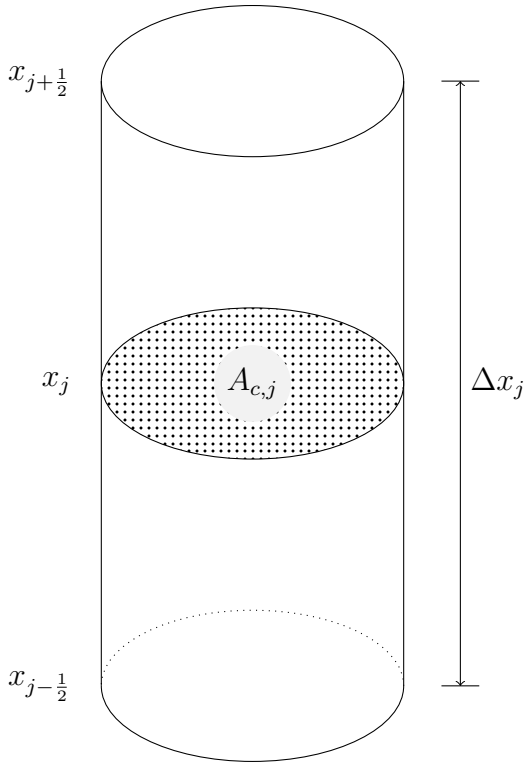

Figure 2.5 Single continuity volume over which the conservation equations are integrated.

$$
\begin{aligned}
\int_{V_j} \frac{\partial\left(\alpha_l \rho_l\right)}{\partial t} \; + \nabla \cdot \left(\alpha_l \rho_l u_l\right) \mathrm{d}V &= \int_{V_j} \left(-(1-\eta)\dot{\Gamma}''' - \dot{\Upsilon}''' + \dot{s}'''_{m,l}\right) \mathrm{d}V \\
V_j \frac{\partial\left(\alpha_{l,j} \rho_{l,j}\right)}{\partial t} &= -\int_{V_j} \nabla \cdot \left(\alpha_l \rho_l u_l\right) \mathrm{d}V - (1-\eta_j)\dot{\Gamma}_j - \dot{\Upsilon}_j + \dot{s}_{m,l,j} \\
V_j \frac{\partial\left(\alpha_{l,j} \rho_{l,j}\right)}{\partial t} &= -\left[\alpha_l \rho_l u_l A_p\right]_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} - (1-\eta_j)\dot{\Gamma}_j - \dot{\Upsilon}_j + \dot{s}_{m,l,j} \\
V_j \frac{\partial\left(\alpha_{l,j} \rho_{l,j}\right)}{\partial t} &= -\left(\langle \alpha_l \rho_l \rangle_{\mathrm{d},j+\frac{1}{2}} u_{l,j+\frac{1}{2}} A_{p,j+\frac{1}{2}} - \langle \alpha_l \rho_l \rangle_{\mathrm{d},j-\frac{1}{2}} u_{l,j-\frac{1}{2}} A_{p,j-\frac{1}{2}}\right) \\
&\quad -(1-\eta)\dot{\Gamma}_j - \dot{\Upsilon}_j + \dot{s}_{m,l,j}
\end{aligned}
\tag{2.16}
$$

For the mass-flux terms evaluated on the continuity volume edge, the advected quantity is evaluated using a 1st order upwind method [41]. The velocity and cross-sectional area utilized in the continuity flux terms, $u_{j\pm\frac{1}{2}}$ and $A_{p,j\pm\frac{1}{2}}$, have the values defined in the momentum flow path that aligns with the boundary of continuity volumes. The sign of the velocity at the volume

boundary determines the value of the donored quantity, $\langle a \rangle_{\mathrm{d},j\pm\frac{1}{2}}$. A more generic formulation for this scheme is given by (2.17). The superscripts, b and c, denote the discrete points in time at which the variables are evaluated. The particular time superscripts in (2.17) are for illustrative purposes and are not fixed.

$$\langle a^b \rangle^c_{\mathrm{d},j-\frac{1}{2}} = \begin{cases} a^b_{j-1} & u^c_{j-\frac{1}{2}} \geq 0 \\ a^b_j & u^c_{j-\frac{1}{2}} < 0 \end{cases} \tag{2.17}$$

The three momentum conservation equations, (2.10) – (2.12), are integrated over their momentum flow path. The cross-sectional area for a momentum path, A_p, can be defined independently from the two cross-sectional areas of the adjoining continuity volumes. The momentum flux terms, (2.18), are treated similarly to the flux terms in the continuity equations.

$$- \left[\langle \alpha_l \rho_l u_l \rangle_{\mathrm{d}} \langle u_l \rangle_{\mathrm{a}} \tilde{A} \right]_{x_j}^{x_{j+1}} \tag{2.18}$$

There are two primary differences. First, the area in the flux term, $\tilde{A}$, is taken as the minimum of two areas from the adjoining momentum flow paths, (2.19).

$$\tilde{A}_j = \min \left(A_{p,j-\frac{1}{2}}, A_{p,j+\frac{1}{2}} \right) \tag{2.19}$$

Second, the phasic velocity that is used to determine the donored quantities is the arithmetic mean of the velocities from the two adjacent momentum flow paths, (2.20).

$$\langle u \rangle_{\mathrm{a},\phi,j} = \frac{u_{\phi,j-\frac{1}{2}} + u_{\phi,j+\frac{1}{2}}}{2} \tag{2.20}$$

The ordering of the governing equations within a given continuity volumes is as follows:

1. Conservation of the non-condensable gas field mass.
2. Conservation of the continuous liquid field mass.
3. Conservation of the gaseous phase energy.
4. Conservation of the liquid phase energy.
5. Conservation of the entrained liquid field mass.

6. Conservation of the vapor field mass.

The conservation of momentum equations within a given momentum flow path are ordered as follows:

1. Conservation of the continuous liquid field momentum.
2. Conservation of the gaseous phase momentum.
3. Conservation of the entrained liquid field momentum.

2.3.2 Temporal Approximations

Once the governing conservation equations have been spatially discretized utilizing the method outlined above, the temporal derivatives need to be approximated numerically. The continuous conservation equations, (2.14), are now a spatially-discrete, temporally-continuous set of equations given by (2.21), where $\mathbf{E}$ now represents the spatially discrete approximation of $\mathbf{e}$.

$$\frac{\partial \mathbf{y}}{\partial t} = \mathbf{E}\left(\mathbf{y}\left(t\right)\right) \tag{2.21}$$

Given that there are nine conservation equations, nine independent parameters need to be chosen. The choice of the nine variables is another distinguishing characteristic of safety analysis software. This work uses the following set of nine independent variables (2.22).

$$\mathbf{x} = [\alpha_g P_n, \alpha_g, \alpha_g h_v, (1-\alpha_g) h_l, \alpha_e, P, \dot{m}_g, \dot{m}_l, \dot{m}_e]^T \tag{2.22}$$

(2.23) gives the definition of the conserved momentum quantity for a given phase, $\dot{m}_\phi$.

$$\dot{m}_\phi = \langle \alpha_\phi \rho_\phi \rangle_{\mathrm{a}} u_\phi A_p \tag{2.23}$$

The averaging operator $\langle a \rangle_{\mathrm{a}}$ provides the average of a quantity from the two adjoining continuity volumes, (2.24).

$$\langle a \rangle_{\mathrm{a},j+\frac{1}{2}} = \frac{a_j + a_{j+1}}{2} \tag{2.24}$$

The use of the momenta and the product of phasic volume fractions and phasic enthalpies, such as $\alpha_g h_v$, as independent parameters represents a unique choice. These independent parameters are not the only option for safety analysis software. Other software, such as TRACE and RELAP5-3D, use variants of these parameters. These variations include the use of velocities instead of momenta, and temperatures or internal energies instead of enthalpies [42, 46].

A distinguishing feature of the numerical method in COBRA is its treatment of the temporal derivative for the conservation of momentum. In the work that follows the temporal derivative of the conserved momentum equations is directly discretized, which is known as the conservative form. The other option, the non-conservative form, expands the temporal derivative analytically via the chain rule, (2.25), and these equations are then temporally discretized.

$$\frac{\partial \alpha_\phi \rho_\phi u_\phi}{\partial t} = \alpha_\phi \rho_\phi \frac{\partial u_\phi}{\partial t} + u_\phi \frac{\partial \alpha_\phi \rho_\phi}{\partial t} \tag{2.25}$$

In this work the temporal derivative is approximated by a one-step difference scheme where the continuous time variables are now evaluated at discrete points, $t^0, t^1, \ldots, t^{N_t}$. The notations t^0 and t^{N_t} represent the initial and final time, respectively. The term “one-step" refers to the fact that the temporal derivative involves only two consecutive points in time. The integral over a time interval, $\Delta t = t^{n+1} - t^n$, is show in (2.26).

$$\begin{aligned}
\int_{t^n}^{t^{n+1}} \frac{\partial \mathbf{y}\left(\mathbf{x}\right)}{\partial t} \mathrm{d}\tau &= \int_{t^n}^{t^{n+1}} \mathbf{E}\left(\mathbf{y}\left(\mathbf{x}\right)\right) \mathrm{d}\tau \\
\mathbf{y}\left(\mathbf{x}^{n+1}\right) - \mathbf{y}\left(\mathbf{x}^{n}\right) &= \int_{t^{n+1}}^{t^{n}} \mathbf{E}\left(\mathbf{y}\left(\mathbf{x}\right)\right) \mathrm{d}\tau \\
\mathbf{y}\left(\mathbf{x}^{n+1}\right) - \mathbf{y}\left(\mathbf{x}^{n}\right) &= \Delta t \mathbf{E}\left(\mathbf{y}\left(\mathbf{x}^{*}\right)\right) \\
\frac{\mathbf{y}\left(\mathbf{x}^{n+1}\right) - \mathbf{y}\left(\mathbf{x}^{n}\right)}{\Delta t} &= \mathbf{E}\left(\mathbf{y}\left(\mathbf{x}^{*}\right)\right)
\end{aligned} \tag{2.26}$$

The choice of how to approximate the temporal integral of the sources and sinks of the system, $\mathbf{E}\left(\mathbf{y}\left(\mathbf{x}^{*}\right)\right)$, hereafter referred to as $\mathbf{E}^{*}$, is a factor that defines the eventual solution algorithm. There are two subcategories for solving this one-step temporal-integration problem, single-stage and multi-stage [25, 38]. Algorithm 2.1 shows how a multi-stage temporal integration scheme would work.

Require: $\mathbf{y}^{\,0}$ and t^0
1: **set:** $n = 0$
2: **loop** Transient Loop
3: $\quad t^{n+1} := t^n + \Delta t$
4: $\quad$ **for** $s = 1 \rightarrow N_s$ **do** Stage Loop
5: $\quad\quad$ **Black Box:** Solve $\dfrac{\mathbf{y}^{\,s} - \mathbf{y}^{\,n}}{\Delta t} = \mathbf{E}^{\,*,s}$ for $\mathbf{y}^{\,s}$.
6: $\quad$ **end for**
7: $\quad n = n + 1$
8: **end loop**

Algorithm 2.1 Multi-stage temporal integration scheme.

The final-stage conserved variables will be the new-time variables, $\mathbf{y}^{\,N_s} = \mathbf{y}^{\,n+1}$. At each stage the choice of how to approximate the driving function, $\mathbf{E}^{\,*,s}$, can change in both its functional dependence upon the stage values of $\mathbf{y}^{\,s}$ and in which components of $\mathbf{E}^{\,*}$ are included. By excluding certain portions of $\mathbf{E}^{\,*}$ at certain stages, a time-splitting algorithm is developed. Time-splitting algorithms are also known as operator-splitting algorithms. By changing the functional dependencies of terms within $\mathbf{E}^{\,*}$ at different stages, predictor-corrector methods, also called stabilizing correction methods, are generated. A single-stage method is the degenerate multi-stage case of $N_s = 1$.

For this work the conserved quantities within a continuity volume (e.g., $\alpha_g \rho_g$, $\left(1 - \alpha_g\right) \rho_l h_l$) are nonlinear functions of the chosen independent parameters, (2.27), regardless of the approximation chosen for $\mathbf{E}^{\,*}$.

$$\mathbf{y}^{\,n+1} = \mathbf{y}\left(\mathbf{x}^{\,n+1}\right) \tag{2.27}$$

This nonlinearity necessitates the use of a nonlinear solver at every timestep. Any dependence of $\mathbf{E}^{\,*}$ on the new-time parameters creates additional nonlinearities. The discrete formulation, (2.26), can be expressed as a nonlinear residual that is a function of $\mathbf{x}^{\,n+1}$, (2.28).

$$\mathbf{F}\left(\mathbf{x}^{\,n+1}\right) = \mathbf{y}\left(\mathbf{x}^{\,n+1}\right) - \mathbf{y}\left(\mathbf{x}^{\,n}\right) - \Delta t \mathbf{E}^{\,*} \tag{2.28}$$

The temporal integration method used has an associated temporal accuracy that depends upon the approximation of $\mathbf{E}^{\,*}$ and the number of stages. The temporal accuracy is a way of quantifying

the behavior of the solution as the timestep is reduced. Order of accuracy estimates for temporal integration techniques are proportionality statements between the numerical error and a power of the timestep, $\mathcal{O}(\Delta t^p)$ [25]. However, it has been shown that if the nonlinear problem, (2.28), is not solved at every timestep, the temporal accuracy of a method can be degraded [21, 29].

2.3.3 Nonlinear Approximations

The nonlinear residual is a function of the new-time unknowns, $\mathbf{x}^{n+1}$. The method used to solve (2.28) for $\mathbf{x}^{n+1}$ in this work is Newton's method [12, 13]. Newton's method is an iterative procedure to obtain $\mathbf{x}^{n+1,k}$ such that $\mathbf{F}\left(\mathbf{x}^{n+1,k}\right) = 0$, as such two superscripts are required. The nonlinear iterate superscript will be k. Successive linearization of the nonlinear problem, (2.29), generates an iterative solution method.

$$0 = \mathbf{F}\left(\mathbf{x}^{n+1,k} + \delta\mathbf{x}^{k}\right) \approx \mathbf{F}\left(\mathbf{x}^{n+1,k}\right) + \mathbf{J}\left(\mathbf{x}^{n+1,k}\right) \cdot \delta\mathbf{x}^{k} \tag{2.29}$$

The algorithm is then one of finding successive updates, $\delta\mathbf{x}^{k} = \mathbf{x}^{n+1,k+1} - \mathbf{x}^{n+1,k}$, by solving (2.30).

$$\mathbf{J}\left(\mathbf{x}^{n+1,k}\right) \cdot \delta\mathbf{x}^{k} = -\mathbf{F}\left(\mathbf{x}^{n+1,k}\right) \tag{2.30}$$

Since the system of nonlinear equations being solved represents a transient simulation, the iterations at each timestep start with an initial guess for the new-time variables that is equal to the old-time variables, $\mathbf{x}^{n+1,0} = \mathbf{x}^{n}$. The underlying assumption of this initial value is that the independent parameters will not change greatly over a timestep, and the old-time variables will provide an initial vector that is within the radius of convergence of Newton's method. Algorithm 2.2 provides an overview of a transient simulation using Newton's method for a single-stage temporal integration scheme. The algorithm generalizes to a multi-stage temporal integration method by enclosing the Newton loop within a stage loop.

In Algorithm 2.2, there is a black box step, the calculation of the loop termination criteria. In multi-stage temporal integration methods the loop termination criteria may vary between stages.

Require: $\mathbf{x}^0$ and t^0
1: **set:** $n = 0$
2: **loop** Transient Loop
3: **set:** $t^{n+1} = t^n + \Delta t$
4: **set:** $k = 0$
5: **set:** $\mathbf{x}^{n+1,k} = \mathbf{x}^n$
6: **loop** Newton Loop
7: **calculate:** $\mathbf{F}\left(\mathbf{x}^{n+1,k}\right)$ and $\mathbf{J}\left(\mathbf{x}^{n+1,k}\right)$
8: **calculate:** $\delta\mathbf{x}^k = -\mathbf{J}^{-1} \cdot \mathbf{F}$
9: **calculate:** $\mathbf{x}^{n+1,k+1} = \mathbf{x}^{n+1,k} + \delta\mathbf{x}^k$
10: **set:** $k \mathrel{+}= 1$
11: **Black Box:** Loop Termination Criteria
12: **end loop**
13: **set:** $n \mathrel{+}= 1$
14: **end loop**

Algorithm 2.2 Local Newton's method for single-stage temporal integration.

In the case where the loop is terminated after only a single iterate the resultant method is labeled a single-shot linearization.

2.4 Solution Methods

Section 2.3 provided a framework for characterizing the different solution methods used in thermal-hydraulic safety codes. Each method can be defined by the manner in which the temporal integration is carried out and the manner in which the nonlinearities are resolved. The methods described below form the core of available techniques that have been developed for two-phase safety analysis codes. While each of the following methods may have many subtly varying algorithmic implementations that have appeared in safety analysis software, the algorithms detailed below are general enough to encompass these variants.

2.4.1 Fully Explicit Method

The least computationally expensive method on a per timestep basis for temporally integrating (2.26) is a single-stage, fully explicit method. In the fully explicit method, the driving function is approximated as $\mathbf{E}\left(\mathbf{x}^n\right)$. (2.31) gives the nonlinear vector notation formulation for this method.

$$\mathbf{F}\left(\mathbf{x}^{n+1}\right) = \mathbf{y}\left(\mathbf{x}^{n+1}\right) - \mathbf{y}^{n} - \Delta t \mathbf{E}\left(\mathbf{y}\left(\mathbf{x}^{n}\right)\right) \tag{2.31}$$

Given the choice of independent parameters in this work, there are nonlinearities present in (2.31). The algorithmic implementation shown in Algorithm 2.3 assumes a linearized solution technique where only a single Newton step is taken.

Require: $\mathbf{x}^{0}$ and t^{0}
1: **set:** $n = 0$
2: **loop** Take a Timestep
3: $t^{n+1} := t^{n} + \Delta t$
4: **calculate:** $\mathbf{F}\left(\mathbf{x}^{n}\right)$ and $\mathbf{J}\left(\mathbf{x}^{n}\right)$
5: **calculate:** $\delta\mathbf{x} = -\mathbf{J}^{-1}\mathbf{F}$
6: **calculate:** $\mathbf{x}^{n+1} = \mathbf{x}^{n} + \delta\mathbf{x}$
7: **end loop** $n \mathrel{+}= n + 1$

Algorithm 2.3 Single-stage, fully explicit, linearized method.

While this particular method is the least computationally expensive on a per timestep basis of those discussed, it has a severe weakness. That weakness is the Courant-Friedrichs-Lewy (CFL) limit imposed upon the timestep size, Δt. The CFL limit is a relationship between the spatial and temporal discretization and the characteristic velocities of information propagation in the problem of interest [25, 41]. For the case of (2.31), the CFL limit is given in (2.32).

$$\Delta t_i \lesssim \frac{\Delta x_i}{|u_{\phi,i}| + |c_i|} \tag{2.32}$$

In (2.32), c_i and u_i are the speed of sound and the magnitude of a given phasic velocity, respectively, at a given location within the domain. These two velocities, for most applications of interest, can be of drastically different magnitude. For example, in single-phase gaseous flow, the ratio of the local phasic velocity to the speed of sound is typically much less than one (≈ 0.01) for problems of interest. Using the local fluid conditions, Δt_i is calculated at every point within the domain, Ω. The Δt chosen for the $t^{n} \rightarrow t^{n+1}$ timestep is the most restrictive calculated value over the domain, (2.33).

$$\Delta t = \min_{i \in \Omega} \Delta t_i \tag{2.33}$$

Since the CFL limit is based upon the local speed of sound, it is referred to as the sonic Courant limit. While the explicit method may provide the lowest computational cost on a per-timestep basis, the number of timesteps required for a given problem may be much greater than the following methods due to this restrictive CFL limitation. This limitation of the explicit method prompted the development of alternative methods that were capable of exceeding this sonic Courant limit.

2.4.2 Fully Implicit Method

One alternative method for integrating (2.21) is to use a fully implicit discretization of $\mathbf{E}^*$ [6, 17]. The fully implicit method temporally approximates $\mathbf{E}^*$ as a function of new-time parameters only, (2.34).

$$\mathbf{F}\left(\mathbf{x}^{n+1}\right) = \mathbf{y}\left(\mathbf{x}^{n+1}\right) - \mathbf{y}^{n} - \Delta t \mathbf{E}\left(\mathbf{y}\left(\mathbf{x}^{n+1}\right)\right) \tag{2.34}$$

This method has the advantage of not being limited by a CFL number. While this allows for greatly increased timestep sizes, the numerical scheme introduces nonphysical diffusion into the solution [29]. Additionally, the solution of (2.34) is the most computationally expensive on a per timestep basis of the methods considered. This computational expense comes from the full inter-volume coupling of the Jacobian matrix during the nonlinear iterations. The computational implementation of a fully implicit method is presented in Algorithm 2.4. The black-box step in Algorithm 2.4 will be discussed in Section 3.2.3.

2.4.3 Semi-Implicit Method

Another alternative method for integrating (2.21) is the semi-implicit method. It was developed to overcome the sonic Courant limitations of the explicit method while avoiding both the high computational cost and excessive diffusivity of the fully implicit method [27]. The two distinguishing characteristics of the semi-implicit method are the use of new-time variables in the evaluation of both the pressure gradient in the momentum conservation equations and the advecting velocities in

Require: $\mathbf{x}^0$ and t^0
set: $n = 0$
loop Transient Loop
    set: $t^{n+1} := t^n + \Delta t$
    set: $k = 0$
    set: $\mathbf{x}^k = \mathbf{x}^n$
    loop Newton Loop
        calculate: $\mathbf{F}(\mathbf{x}^k)$ and $\mathbf{J}(\mathbf{x}^k)$
        calculate: $\delta\mathbf{x}^k = -\mathbf{J}^{-1} \cdot \mathbf{F}$
        calculate: $\mathbf{x}^{k+1} = \mathbf{x}^k + \delta\mathbf{x}^k$
        set: $k \mathrel{+}= 1$
        Black Box: Loop Termination Criteria
    end loop
    set: $n \mathrel{+}= 1$
end loop

Algorithm 2.4 Fully implicit method.

the mass and energy equations. The implicit evaluation of the pressure gradient in the momentum equations is what leads to a CFL limit known as the material Courant limit. Similar to the sonic Courant limit discussed in Section 2.4.1, the material Courant limit dictates the largest Δt that can be achieved while maintaining a stable solution algorithm. (2.35) gives the Courant limit for this method.

$$\Delta t_i \lesssim \frac{\Delta x_i}{|u_{\phi,i}|} \tag{2.35}$$

The characteristic velocity used in the calculation of the material Courant limit is based on the phasic velocity only. This allows for larger, but still limited, timestep sizes than the fully explicit method. The limit on timestep size is due to the explicit evaluation of the donored quantities in the flux terms in the conservation equations. As stated, the flux of mass and energy include new-time velocities, u_ϕ^{n+1}. Since this work uses momenta, $\dot{m}_\phi$, as independent parameters, these velocities are derived quantities whose functional form is given in (2.36). The $\langle \alpha_\phi \rho_\phi \rangle_{\mathrm{a}}^n$ represents the arithmetic average of macroscopic densities from adjoining continuity volumes. The averaging operator $\langle a \rangle_{\mathrm{a}}^n$ is an extension of (2.17). There is a mismatch between the averaged macroscopic density and the momentum in (2.36). This mismatch will be discussed in more detail in Section 3.4.3.

$$u_{\phi,j\pm\frac{1}{2}}^{n+1} = \frac{\dot{m}_{\phi,j\pm\frac{1}{2}}^{n+1}}{A_{p,j\pm\frac{1}{2}} \langle \alpha_\phi \rho_\phi \rangle_{\mathrm{a},j\pm\frac{1}{2}}^{n}} \tag{2.36}$$

An implementation of the semi-implicit method that takes only a single Newton step is shown in Algorithm 2.5s.

Require: $\mathbf{x}^0$ and t^0
1: **set:** $n = 0$
2: **loop** Transient Loop
3: **set:** $t^{n+1} := t^n + \Delta t$
4: **calculate:** $\mathbf{F}(\mathbf{x}^n)$ and $\mathbf{J}(\mathbf{x}^n)$
5: **calculate:** $\delta\mathbf{x} = -\mathbf{J}^{-1} \cdot \mathbf{F}$
6: **calculate:** $\mathbf{x}^{n+1} = \mathbf{x}^n + \delta\mathbf{x}$
7: **set:** $n \mathrel{+}= 1$
8: **end loop**

Algorithm 2.5 Semi-implicit method.

2.4.4 Stability-Enhancing Two-Step Method

In order to overcome the material Courant limit placed upon simulations by the semi-implicit method, the Stability Enhancing Two-Step (SETS) method was developed [28]. While not as stable as the fully implicit method, the SETS method allows for timesteps to exceed the material Courant limit. It overcomes the material Courant limit by taking a multi-stage approach to the temporal integration. While there are variants of the SETS method [46], Algorithm 2.6 reflects the original publication. The outline below uses velocities as independent parameters, not momenta.

There are three stages in the original SETS method. Stage one is a linearized solution of the momentum equations where only the velocity terms in the momentum equations of $\mathbf{E}^*$ are evaluated implicitly. This first stage only involves the solution of the momentum equations; the mass and energy equations are not solved during this step. The lack of implicit dependence upon any continuity variables allows for coupling only between momentum flow paths. This step results in predicted u_ϕ^* phasic velocities.

In stage two, the momentum and continuity equations are solved using a traditional semi-implicit scheme with two exceptions. The first is that the momentum flux terms are evaluated

using the stage-one predicted velocities as the advecting velocities instead of explicitly evaluating them. The second is that the nonlinearities in the momentum equations, arising from interfacial and wall drag, are subject to a single-shot linearization. The single-shot nature of the momentum equations allows for the reduction of the system into coupled continuity volumes as in the semi-implicit method. The nonlinear continuity equations are then solved via Newton's method. Stated another way, the Jacobians for the momentum equations are fixed at their first Newton iterate value, while the Jacobians for the mass and energy equations are updated at each iterate.

In the third stage the mass and energy equations are solved such that all continuity variables are implicitly evaluated, but the velocities are from stage two. These velocities, $\mathbf{u}^{\,2}$, are the only results from stage two that are used in stage three. The resulting system of coupled continuity equations is then solved. This final stage allows timestep sizes that exceed the material Courant limit. For each stage the residual and Jacobian matrix will be denoted by $\mathbf{F}^{\,s}$ and $\mathbf{J}^{\,s}$, with $s = 1 \rightarrow 3$.

Require: $\mathbf{x}^{\,0}$ and t^0
1: **set:** $n = 0$
2: **loop** Transient Loop
3: **set:** $t^{n+1} := t^n + \Delta t$
4: **calculate:** $\mathbf{F}^{\,1}$ and $\mathbf{J}^{\,1}$
5: **calculate:** $\mathbf{u}^{\,*}$ from $\delta\mathbf{u}^{\,*} = -\left[\mathbf{J}^{\,1}\right]^{-1}\mathbf{F}^{\,1}$
6: **loop** Newton Loop
7: **calculate:** $\mathbf{F}^{\,2}$ and $\mathbf{J}^{\,2}$
8: **calculate:** $\delta\mathbf{x} = -\left[\mathbf{J}^{\,2}\right]^{-1}\mathbf{F}^{\,2}$
9: **calculate:** $\mathbf{x}^{\,n+1} = \mathbf{x}^{\,n} + \delta\mathbf{x}$
10: **Black Box:** Loop Termination Criteria
11: **end loop**
12: **calculate:** $\mathbf{x}^{\,n+1}$ from $\mathbf{F}^{\,3}$.
13: **set:** $n \mathrel{+}= 1$
14: **end loop**

Algorithm 2.6 SETS method.

This algorithm was designed to allow existing software utilizing the semi-implicit method to be embedded within a larger framework. The black-box loop termination criterion is a choice that varies with implementation. The TRACE code uses an $\mathcal{L}_\infty$ norm of the unscaled Newton updates for the continuity variables to determine convergence.

2.4.5 Nearly-Implicit Method

Another possible method used to overcome the material Courant limit without incurring the same computational cost as the fully implicit method is the nearly-implicit method [42, 45]. The nearly-implicit method is a multistage temporal integration scheme. In the first stage the approximation of $\mathbf{E}^*$ is one in which the mass and energy terms are implicit except for the donored values in the flux terms. In addition, the chain rule is applied to the continuous temporal derivatives, resulting in mass equations similar to (2.37). The energy temporal derivatives are similarly expanded.

$$\frac{\partial \alpha_\phi \rho_\phi}{\partial t} = \alpha_\phi \frac{\partial \rho_\phi}{\partial t} + \rho_\phi \frac{\partial \alpha_\phi}{\partial t} \tag{2.37}$$

The temporal discretization of (2.37) is given by (2.38).

$$\alpha_\phi \frac{\partial \rho_\phi}{\partial t} + \rho_\phi \frac{\partial \alpha_\phi}{\partial t} = \alpha_\phi^n \frac{\rho_\phi^{n+1} - \rho_\phi^n}{\Delta t} + \rho_\phi^n \frac{\alpha_\phi^{n+1} - \alpha_\phi^n}{\Delta t} \tag{2.38}$$

The momentum equations are implicit in their momentum flux, pressure, and interfacial exchange terms.

The nearly-implicit method is a three-stage solution process. The first stage is similar to the semi-implicit method. However, as opposed to the semi-implicit method where the momentum variables are eliminated from the equations being solved, the nearly-implicit method uses the mass and energy equations to eliminate the pressure terms from the momentum equations. The resulting matrix is a linear system of coupled velocities. The stage-one velocities are taken to be the new-time velocities. Once the stage-one velocities are obtained, the corresponding stage-one continuity variables are obtained analogously to the new-time momentum in the semi-implicit method. Stage one is a single linearization of the semi-implicit method. In the second stage, only the continuity equations are solved and they are in conservative form. The interfacial exchange terms in the mass and energy equations are evaluated using the stage-one continuity variables. The resulting system of equations is linear in the conserved quantities. This linear system is solved for the conserved quantities. In stage three, the nonlinear relationship between the conserved quantities and the

independent parameters is then approximated via a single linearization of the equations of state. Algorithm 2.7 shows the three-stage process.

Require: $\mathbf{x}^{\,0}$ and t^0
set: $n = 0$
loop Transient Loop
  set: $t^{n+1} := t^n + \Delta t$
  calculate: $\mathbf{F}^{\,1}$ and $\mathbf{J}^{\,1}$
  calculate: $\mathbf{x}^{\,*}$ and $\mathbf{u}^{\,n+1}$ from $\delta\mathbf{x}^{\,*} = -\left[\mathbf{J}^{\,1}\right]^{-1}\mathbf{F}^{\,1}$
  calculate: $\mathbf{F}^{\,2}$
  calculate: $\mathbf{x}^{\,**}$ from $\mathbf{F}^{\,2}$
  calculate: $\mathbf{x}^{\,n+1}$ from $\mathbf{F}^{\,3}$.
  set: $n \mathrel{+}= 1$
end loop

Algorithm 2.7 The nearly-implicit method.

2.5 Domain Coupling

COBRA possesses models of the physics necessary to simulate the complicated in-core flow patterns and heat transfer encountered during a postulated LOCA. However, it lacks the detailed models for NPP components (pumps, valves, accumulators, pressurizers) that are traditionally available from system analysis codes. When modeling full NPP transients, it is advantageous to use sub-channel software for the core and system analysis software for the rest of the NPP. The combination of different specialized software is a common practice. This coupling of capabilities can be accomplished in two ways: the two pieces of software can be merged or they can be coupled via data exchange. Examples of the first method are MARS [20], COBRA/TRAC [43], and TRACE [46]. Coupling software via data exchange is a more common option [2, 3, 5, 30, 35, 47], and it will be discussed first.

When coupling via data exchange, the use of explicit coupling can lead to a sonic Courant limit imposed at the boundary of the two coupled systems [2, 34]. To circumvent the sonic Courant limit, a semi-implicit method for the coupling of software via data exchange has been developed for thermal-hydraulic simulations [3, 47]. This coupling technique allows for the material Courant

limit to be uniformly applied across the two pieces of software. This technique is of direct interest to the proposed research.

The coupling of the different software involves the use of a third-party message-passing interface. The program used for the software coupling in the papers of interest is the Parallel Virtual Machine (PVM) [18]. PVM allows the data exchange to occur between the coupled software. As originally formulated, the software coupling technique involves breaking the coupled problem into two pieces, the master and the slave components. In this respect, the problem can be thought of as being a single domain broken in two: one subdomain controlled by the master process, and one subdomain being controlled by the slave process. The presentation of this method is in terms of the RELAP5-3D conservation equations, which are different from those in COBRA. The coupling method first developed presented RELAP5-3D as both the master and slave software [47]. Later work showed how this base method could be extended to allow coupling between RELAP5-3D and the COBRA subchannel analysis software [3]. This reformulation was designed to allow for a consistent transition between a two-phase, two-field based solver and a two-phase, three-field based solver. A short outline of the method as formulated for RELAP5-3D coupling follows.

The semi-implicit software coupling is accomplished via the staggered mesh formulation. The master domain is truncated on a continuity volume. The fluxes of conserved continuity quantities at the boundary of the coupled volumes are retained as unknowns during the formulation of the global pressure matrix. Upon forming the pressure matrix, the Newton update for the pressures will be expressed as a linear combination of all of the unknown mass and energy fluxes through the coupled boundaries of the domain, (2.39) [47].

$$\begin{aligned}\delta P_j^{n+1} = a_j + &\sum_{i=1}^{N_{\text{CPL}}} b_{j,i} n_{g,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} c_{j,i} u_{g,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} d_{j,i} u_{f,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} e_{j,i} m_{g,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} f_{j,i} m_{f,i}^{n+1} \\ &+ \sum_{i=1}^{N_{\text{CPL}}} g_{j,i} w_{g,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} h_{j,i} w_{f,i}^{n+1} \end{aligned} \tag{2.39}$$

(2.39) represents change in pressure in every volume in the computational domain as a linear combination of the fluxes of mass ($n_{g,j}^{n+1}$, $m_{g,j}^{n+1}$, and $m_{f,j}^{n+1}$), volume ($w_{f,j}^{n+1}$ and $w_{g,j}^{n+1}$), and internal energy ($u_{g,j}^{n+1}$ and $u_{f,j}^{n+1}$) through the N_{CPL} coupled boundaries.

From the point-of-view of the slave process, the momentum flow path between the master domain and the slave domain is a traditional momentum flow path. Since the interface boundary between the master and slave domain is composed entirely of momentum flow paths, the only implicit unknowns from the master domain used in the slave domain are the pressures from the coupled continuity volumes. These implicit pressures are expressed in terms of the old-time pressures and the updates to those pressures from the old-time values and the new-time values. The pressure update for a given continuity volume at the boundary of the master domain, $\delta P_{\text{cbv}}^{n+1}$, is expressed in terms of the unknown velocities in the slave momentum flow paths adjacent to the boundary of the master continuity volumes, (2.40) [47].

$$\begin{aligned}\delta P_{\text{cbv}}^{n+1} = a_{\text{cbv}} &+ \sum_{i=1}^{N_{\text{CPL}}} b_{\text{cbv},i} \langle \alpha_g^n \rho_n^n \rangle_{\text{d}}^n A_i v_{g,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} c_{\text{cbv},i} \langle \alpha_g^n \rho_g^n u_g^n \rangle_{\text{d}}^n A_i v_{g,i}^{n+1} \\ &+ \sum_{i=1}^{N_{\text{CPL}}} d_{\text{cbv},i} \langle \alpha_f^n \rho_f^n u_f^n \rangle_{\text{d}}^n A_i v_{f,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} e_{\text{cbv},i} \langle \alpha_g^n \rho_g^n \rangle_{\text{d}}^n A_i v_{g,i}^{n+1} \\ &+ \sum_{i=1}^{N_{\text{CPL}}} f_{\text{cbv},i} \langle \alpha_f^n \rho_f^n \rangle_{\text{d}}^n A_i v_{f,i}^{n+1} + \sum_{i=1}^{N_{\text{CPL}}} g_{\text{cbv},i} \langle \alpha_g^n \rangle_{\text{d}}^n A_i v_{g,i}^{n+1} \\ &+ \sum_{i=1}^{N_{\text{CPL}}} h_{\text{cbv},i} \langle \alpha_f^n \rangle_{\text{d}}^n A_i v_{f,i}^{n+1} \end{aligned} \tag{2.40}$$

This formulation allows for the updated pressure in the master domain to be calculated within the slave domain without inter-software communication. The intent of this formulation was to enable the semi-implicit method to be used for software coupling.

The coupling of system analysis codes and sub-channel codes can also be viewed through the following lens. The global problem, comprised of the balance of plant (e.g., RELAP5-3D) and the in-core region (e.g., COBRA), can be considered as part of a global domain. Each piece of software can then be viewed as solving a subset of the global domain. The use of more detailed physics for the in-core thermal-hydraulic behavior is a particular application of model reduction [32].

If the semi-implicit coupling methodology is used to couple two pieces of software that use the semi-implicit method, then the boundaries between the two domains no longer represent a

disparity in the global method used. This fact enables the use of the material Courant limit at coupling boundaries. Consequently, this software coupling method allows each software process to use consistent boundary information in its calculations.

Additionally, if the boundary values are explicitly evaluated, then the decomposition of the global problem could be viewed as an application of an additive Schwarz domain decomposition algorithm. In an additive Schwarz algorithm the boundary values for each subdomain are functions only of the old-time values in the other domains. From this viewpoint, the sonic Courant limit observed at software coupling boundaries follows naturally [2].

The additive Schwarz method has seen extensive use as a nonlinear preconditioner for fully implicit computational fluid dynamics calculations [8, 9]. In particular, a nonlinearly convergent solution within each subdomain is obtained at each timestep. It has been shown that the treatment of localized nonlinearities via domain decomposition allows globalization strategies to be used in the global Newton step that would otherwise exhibit stalled convergence if the full domain were initially subjected to a globalization strategy [10]. This work is based upon the concept of nonlinear domain decomposition and nonlinear elimination [15, 23]. However, these applications do not worry about consistency at the boundaries of the subdomains since the obtained subdomain solutions are only used as a preconditioned initial guess for a global Newton-Krylov-Schwarz algorithm [11].

The two paradigms of discrete software coupling and domain decomposition are complementary views of the same process. The software coupling is the algorithmic implementation of the domain decomposition. Using the semi-implicit software coupling as a basis for domain decomposition within a single code provides a mean of isolating subdomains that provides a consistent formulation for the nonlinear problem.

Chapter 3

Nonlinear COBRA

The original COBRA software utilized the single step of the traditional semi-implicit method. In order to investigate the effects of nonlinear convergence upon the determination of the temporal convergence for a given solution, COBRA was modified to enable an iterative global Newton's method. Then, for effective use of the nonlinear solver, an operator-based scaling method was developed to provide a physically meaningful measure of convergence. This chapter will discuss each of these phases of development, as well as several implementation-specific practicalities that were identified and resolved.

3.1 Linear COBRA

COBRA utilizes the semi-implicit method as outlined in Section 2.4.3. The governing equations in the linearized semi-implicit method are not viewed through the lens of a single Newton step for a system of nonlinear equations. Instead, the view that the linearization is from $\mathbf{x}^n$ to $\mathbf{x}^{n+1}$, as opposed to $\mathbf{x}^{n+1,k}$ to $\mathbf{x}^{n+1,k+1}$, is standard in the literature of linearized safety analysis software. For that reason, the following discussion will present the algorithm from the point of view of a single linearization. An outline of the solution algorithm is given in Algorithm 3.1. This section will outline the different steps within a single timestep of the linear solver.

Following the initialization of data, the first step in a purely hydrodynamic simulation in COBRA is a loop over all momentum equations in the domain. The nonlinear momentum equations, neglecting spatial discretization notation and external sources, are given in (3.1) – (3.3).

Require: $\mathbf{x}^0$ and t^0
1: **set:** $n = 0$
2: **loop** Transient Loop
3: **set:** $t^{n+1} := t^n + \Delta t$
4: **algorithm:** Assemble Linear Pressure Matrix ▷ Algorithm 3.2
5: **solve:** $\mathbf{A}_{\text{lin}}\delta\mathbf{P}_{\text{lin}} = \mathbf{res}_{\text{lin}}$
6: **algorithm:** Update Linear Variables ▷ Algorithm 3.3
7: **set:** $n \mathrel{+}= 1$
8: **end loop**

Algorithm 3.1 Linear COBRA algorithm.

$$\dot{m}_l^{n+1} - \dot{m}_l^n = \frac{\Delta t}{\Delta x}\left(- \sum_{i \in N_c} \left(\langle \alpha_l \rho_l u_l \rangle_{\text{d}} \langle u \rangle_{\text{a,l}} \tilde{A} \right)_i^n - \langle \alpha_l \rangle_{\text{a}}^n \nabla P^{\,n+1} + g \langle \alpha_l \rho_l \rangle_{\text{a}}^n - K_{wl}^n (\dot{m}_l^{n+1})^2 \right.$$
$$\left. + \; K_{i,gl}^n (\dot{m}_l^{n+1} - \dot{m}_g^{n+1})^2 - \left[(1-\eta)\dot{\Gamma} u' + \dot{\Upsilon} u' \right]^n \right) \tag{3.1}$$

$$\dot{m}_g^{n+1} - \dot{m}_g^n = \frac{\Delta t}{\Delta x}\left(- \sum_{i \in N_c} \left(\langle \alpha_g \rho_g u_g \rangle_{\text{d}} \langle u \rangle_{\text{a},g} \tilde{A} \right)_i^n - \langle \alpha_g \rangle_{\text{a}}^n \nabla P^{\,n+1} + g \langle \alpha_g \rho_g \rangle_{\text{a}}^n - K_{wg}^n (\dot{m}_g^{n+1})^2 \right.$$
$$\left. - \; K_{i,gl}^n (\dot{m}_l^{n+1} - \dot{m}_g^{n+1})^2 - K_{i,ge}^n (\dot{m}_e^{n+1} - \dot{m}_g^{n+1})^2 + \left[\dot{\Gamma} u' \right]^n \right) \tag{3.2}$$

$$\dot{m}_e^{n+1} - \dot{m}_e^n = \frac{\Delta t}{\Delta x}\left(- \sum_{i \in N_c} \left(\langle \alpha_e \rho_l u_e \rangle_{\text{d}} \langle u \rangle_{\text{a},e} \tilde{A} \right)_i^n - \langle \alpha_e \rangle_{\text{a}}^n \nabla P^{\,n+1} + g \langle \alpha_e \rho_l \rangle_{\text{a}}^n - K_{we}^n (\dot{m}_e^{n+1})^2 \right.$$
$$\left. + \; K_{i,ge}^n (\dot{m}_e^{n+1} - \dot{m}_g^{n+1})^2 - \left[\eta \dot{\Gamma} u' - \dot{\Upsilon} u' \right]^n \right) \tag{3.3}$$

In the above equations the flux terms are summed over N_c, which indicates the set of continuity volumes to which a given momentum flow path connects. The K terms represent effective wall and interfacial friction coefficients calculated using parameters only from time n. The transfer of momentum associated with the interfacial transfer of mass is also explicitly evaluated. The vector notation for the three momenta, $\dot{\mathbf{m}}$, is defined by (3.4).

$$\dot{\mathbf{m}} = \begin{bmatrix} \dot{m}_l \\ \dot{m}_g \\ \dot{m}_e \end{bmatrix} \tag{3.4}$$

The terms in the momentum equations that are evaluated at their new-time values are then linearized about their old-time values. This includes the momenta, $\dot{\mathbf{m}}^{\,n+1}$, and the pressure, P^{n+1}, giving (3.5) - (3.7).

$$\begin{aligned} \dot{m}_l^{n+1} - \dot{m}_l^n = \frac{\Delta t}{\Delta x} \Bigg(& - \sum_{i \in N_c} \Big(\langle \alpha_l \rho_l u_l \rangle_{\mathrm{d}} \langle u \rangle_{\mathrm{a},l} \tilde{A} \Big)_i^n - \langle \alpha_l \rangle_{\mathrm{a}}^n \nabla P^{\,n} - \langle \alpha_l \rangle_{\mathrm{a}}^n \delta(\nabla P) + g \langle \alpha_l \rho_l \rangle_{\mathrm{a}}^n \\ & + K_{wl}^n (\dot{m}_l^n)^2 - 2 K_{wl}^n \dot{m}_l^n \dot{m}_l^{n+1} - K_{i,gl}^n (\dot{m}_l^n - \dot{m}_g^n)^2 \\ & + 2 K_{i,gl}^n (\dot{m}_l^n - \dot{m}_g^n)(\dot{m}_l^{n+1} - \dot{m}_g^{n+1}) - \Big[(1-\eta) \dot{\Gamma} u' + \dot{\Upsilon} u' \Big]^n \Bigg) \end{aligned} \tag{3.5}$$

$$\begin{aligned} \dot{m}_g^{n+1} - \dot{m}_g^n = \frac{\Delta t}{\Delta x} \Bigg(& - \sum_{i \in N_c} \Big(\langle \alpha_g \rho_g u_g \rangle_{\mathrm{d}} \langle u \rangle_{\mathrm{a},g} \tilde{A} \Big)_i^n - \langle \alpha_g \rangle_{\mathrm{a}}^n \nabla P^{\,n} - \langle \alpha_g \rangle_{\mathrm{a}}^n \delta(\nabla P) + g \langle \alpha_g \rho_g \rangle_{\mathrm{a}}^n \\ & + K_{wg}^n (\dot{m}_g^n)^2 + K_{i,gl}^n (\dot{m}_l^n - \dot{m}_g^n)^2 + K_{i,ge}^n (\dot{m}_e^n - \dot{m}_g^n)^2 + \Big[\dot{\Gamma} u' \Big]^n \\ & - 2 K_{i,gl}^n (\dot{m}_l^n - \dot{m}_g^n)(\dot{m}_l^{n+1} - \dot{m}_g^{n+1}) - 2 K_{wg}^n \dot{m}_g^n \dot{m}_g^{n+1} \\ & - 2 K_{i,ge}^n (\dot{m}_e^n - \dot{m}_g^n)(\dot{m}_e^{n+1} - \dot{m}_g^{n+1}) \Bigg) \end{aligned} \tag{3.6}$$

$$\begin{aligned} \dot{m}_e^{n+1} - \dot{m}_e^n = \frac{\Delta t}{\Delta x} \Bigg(& - \sum_{i \in N_c} \Big(\langle \alpha_e \rho_e u_e \rangle_{\mathrm{d}} \langle u \rangle_{\mathrm{a},e} \tilde{A} \Big)_i^n - \langle \alpha_e \rangle_{\mathrm{a}}^n \nabla P^{\,n} - \langle \alpha_e \rangle_{\mathrm{a}}^n \delta(\nabla P) + g \langle \alpha_e \rho_e \rangle_{\mathrm{a}}^n \\ & + K_{we}^n (\dot{m}_e^n)^2 - 2 K_{we}^n \dot{m}_e^n \dot{m}_e^{n+1} - K_{i,ge}^n (\dot{m}_e^n - \dot{m}_g^n)^2 \\ & + 2 K_{i,ge}^n (\dot{m}_e^n - \dot{m}_g^n)(\dot{m}_e^{n+1} - \dot{m}_g^{n+1}) - \Big[\eta \dot{\Gamma} u' - \dot{\Upsilon} u' \Big]^n \Bigg) \end{aligned} \tag{3.7}$$

The momentum equations, (3.5) - (3.7), are now linear in the unknown variables $\dot{\mathbf{m}}^{\,n+1}$. These three equations are then simultaneously solved for the three unknown new-time momenta, $\dot{\mathbf{m}}^{\,n+1}$. The differential of the pressure gradient , $\delta\,(\nabla P)$, should be read as the difference between the pressure updates for the two continuity volumes attached to this momentum flow path. The δP

terms are left as unknowns, creating additional unknowns on the right-hand side of the linear system. The resulting system is of the form $\mathbf{J}_{p,j}\dot{\mathbf{m}}_j^{n+1} = \mathbf{a}_j + \sum_{i \in N_c} \mathbf{b}_{j,i}\,\delta P_i$. The solution of this linear system is expressed in (3.8).

$$\dot{\mathbf{m}}_j^{n+1} = \dot{\mathbf{m}}_j^{*} + \sum_{i \in N_c} \frac{\partial \dot{\mathbf{m}}_j}{\partial P_i}\delta P_i \tag{3.8}$$

The definition for the new-time momenta is a function of the as-of-yet unknown changes in pressures that occur in the continuity volumes connected by the momentum flow path. The $\dot{\mathbf{m}}^*$ vector on the right-hand side of (3.8) contains the tentative new time momenta. Once all of the momentum equations have been solved in the above manner, providing estimated new-time momenta for each momentum flow path, the domain's continuity volumes are then traversed.

The continuity equations used in this work, omitting external sources, are shown in (3.9) – (3.10). The new-time velocities used for the N_f advection terms in the continuity equations are defined with the new-time momenta, (3.8), in conjunction with (2.36). N_f represents the set of momentum flow paths that connect to a given continuity volume, and the summation index, i, represents the index of a given momentum flow path.

$$V_c\left[(\alpha_g\rho_n)^{n+1} - (\alpha_g\rho_n)^n\right] = -\Delta t \sum_{i \in N_f}\left(\langle\alpha_g^n\rho_n^n\rangle_d^{n+1}u_g^{n+1}A_p\right)_i \tag{3.9}$$

$$\begin{aligned} V_c\left[(\alpha_l\rho_l)^{n+1} - (\alpha_l\rho_l)^n\right] &= -\Delta t \sum_{i \in N_f}\left(\langle\alpha_l^n\rho_l^n\rangle_d^{n+1}u_l^{n+1}A_p\right)_i \\ &\quad - \left[(1-\eta)\Gamma + \Upsilon\right]^{n+1} \end{aligned} \tag{3.10}$$

$$\begin{aligned} V_c\left[\left(\alpha_g\rho_g h_g\right)^{n+1} - \left(\alpha_g\rho_g h_g\right)^n - \alpha_g^n(P^{n+1} - P^n)\right] &= \left[q_{i,v} + \Gamma h_v^{'} + q_{gl}\right]^{n+1} + q_{wg}^n \\ &\quad - \Delta t \sum_{i \in N_f}\left(\langle\alpha_g^n\rho_g^n h_g^n\rangle_d^{n+1}u_g^{n+1}A_p\right)_i \end{aligned} \tag{3.11}$$

$$\begin{aligned} V_c\left[(\alpha_l\rho_l h_l)^{n+1} - (\alpha_l\rho_l h_l)^n - \alpha_l^n(P^{n+1} - P^n)\right] &= \left[q_{i,l} - \Gamma h_l^{'} - q_{gl}\right]^{n+1} + q_{wl}^n \\ &\quad - \Delta t \sum_{i \in N_f}\left(\langle\alpha_l^n\rho_l^n h_l^n\rangle_d^{n+1}u_l^{n+1}A_p + \langle\alpha_e^n\rho_l^n h_l^n\rangle_d^{n+1}u_e^{n+1}A_p\right)_i \end{aligned} \tag{3.12}$$

$$V_c\left[(\alpha_e\rho_l)^{n+1} - (\alpha_e\rho_l)^n\right] = -\Delta t \sum_{i \in N_f}\left(\langle\alpha_e^n\rho_l^n\rangle_d^{n+1}u_e^{n+1}A_p\right)_i + \left[\Upsilon - \eta\Gamma\right]^{n+1} \tag{3.13}$$

$$V_c \left[\left(\alpha_g \rho_v\right)^{n+1} - \left(\alpha_g \rho_v\right)^{n} \right] = -\Delta t \sum_{i \in N_f} \left(\langle \alpha_g^n \rho_v^n \rangle_d^{n+1} u_g^{n+1} A_p \right)_i + \Gamma^{n+1} \tag{3.14}$$

Unlike the momentum equations, the continuity equations are linearized with variables other than their conserved quantities, (2.22). This creates a more complicated linearization than that used for the momentum equations. The new-time variables in (3.9) – (3.10) are linearized about the old-time variables with respect to the continuity variables in (2.22). The use of (3.8) in defining the new-time velocities introduces inter-continuity volume coupling through the momentum equations' dependency upon the changes in pressure of both the continuity volume being solved, δP, and those of the continuity volumes to which the N_f flow paths connect , $\delta P_{o(i)}$.

$$\mathbf{Z}_c \, \delta \mathbf{C}_c = \mathbf{r}_c \tag{3.15}$$

With the addition of the unknown pressure updates from the connected flow paths, $\delta P_{o(i)}$, the six linearized continuity equations now form the linear system (3.15). Additionally, while the momentum equations were fully expanded to solve for the new-time conserved quantities directly, the continuity equations instead solve for their associated updates, (3.16).

$$\delta \mathbf{C}_c \equiv \begin{bmatrix} \delta(\alpha_g P_n) \\ \delta \alpha_g \\ \delta(\alpha_g h_v) \\ \delta((1-\alpha_g) h_l) \\ \delta \alpha_e \\ \delta P \\ \delta P_{o(1)} \\ \vdots \\ \delta P_{o(N_f)} \end{bmatrix} = \begin{bmatrix} (\alpha_g P_n)^{n+1} - (\alpha_g P_g)^n \\ \alpha_g^{n+1} - \alpha_g^n \\ (\alpha_g h_v)^{n+1} - (\alpha_g h_v)^n \\ ((1-\alpha_g) h_l)^{n+1} - ((1-\alpha_g) h_l)^n \\ \alpha_e^{n+1} - \alpha_e^n \\ P^{n+1} - P^n \\ P_{o(1)}^{n+1} - P_{o(1)}^n \\ \vdots \\ P_{o(N_f)}^{n+1} - P_{o(N_f)}^n \end{bmatrix} \tag{3.16}$$

Each continuity volume's linearized system of equations, (3.15), is subjected to partial LU decomposition without pivoting. The resulting lower-triangular matrix has a unit diagonal. The sixth equation in the upper-triangular system is then scaled by its diagonal. This upper-triangular,

rectangular system for each continuity volume is then stored for later back-substitution. Since the pressure update corresponds to the last row in the system of equations, this allows for the isolation of the pressure updates. The last row from each continuity volume's linear system, (3.15), is then collated into a global, linear pressure matrix, $\mathbf{A}_{\text{lin}}$, and its associated right-hand side, $\mathbf{res}_{\text{lin}}$.

The two loops, one over the momentum flow paths and one over the continuity volumes, form a single group of operations that act upon a given domain. This grouping will be known hereafter as assembling the pressure matrix for a given domain. Algorithm 3.2 shows the two loops and their associated actions. Note that a discussion of vectors $\mathbf{F}_p$ and $\mathbf{F}_c$ will be in Section 3.2.3, and one of vectors $\mathbf{S}_p$ and $\mathbf{S}_c$ will be in Section 3.3.

1: **loop** Momentum Flow Paths
2: **calculate:** $\mathbf{J}_{p,j}$, $\mathbf{a}_j$, and $\mathbf{b}_j$
3: **calculate:** $\dot{\mathbf{m}}^*$ and $\frac{\partial \dot{\mathbf{m}}}{\partial P}$
4: **calculate:** $\mathbf{F}_{p,j}$
5: **calculate:** $\mathbf{S}_{p,j}$
6: **end loop**
7: **loop** Continuity Volumes
8: **calculate:** $\mathbf{Z}_{c,j}$ and $\mathbf{r}_{c,j}$
9: **calculate:** $\mathbf{F}_{c,j}$
10: **calculate:** $\mathbf{S}_{c,j}$
11: **calculate:** $\mathbf{U}_{c,j} = \mathbf{L}_{c,j}^{-1}\mathbf{Z}_{c,j}$
12: **calculate:** $\mathbf{r}_{c,j}^* = \mathbf{L}_{c,j}^{-1}\mathbf{r}_{c,j}$
13: **set:** Scale $\mathbf{U}[6,:]_{c,j}$ and $\mathbf{r}[6]_{c,j}^*$ by $\mathbf{U}[6,6]_{c,j}$
14: **set:** $\mathbf{A}_{\text{lin},i,:} = \mathbf{U}[6,:]_{c,j}$
15: **set:** $\mathbf{res}_{\text{lin},i} = \mathbf{r}[6]_{c,j}^*$
16: **end loop**

Algorithm 3.2 Assembling the Pressure Matrix

$$\mathbf{A}_{\text{lin}}\delta\mathbf{P}_{\text{lin}} = \mathbf{res}_{\text{lin}} \tag{3.17}$$

The system, (3.17), is then solved to determine the pressure updates for the whole domain. A loop over each continuity volume is then performed so that the pressure update can be used in solving its associated upper-triangular system for the updates to the other five continuity variables in (3.16). As the loop over the continuity volume is progressing, any momentum flow path connected to the given continuity volume will have its momenta updated according to (3.8) to include

the contribution from that continuity volumes' change in pressure. After the continuity volume loop has been completed, the $\dot{\mathbf{m}}^*$ variables are assigned to $\dot{\mathbf{m}}^{n+1}$. This update loop is shown in Algorithm 3.3.

1: **loop** Continuity Volumes
2: **solve:** $\mathbf{U}_{c,j}\delta\mathbf{C}_{c,j} = \mathbf{r}^*_{c,j}$ using $\delta\mathbf{P}_{\text{lin}}$
3: **loop** Connected Flow Paths
4: **set:** $\dot{\mathbf{m}}^*_i \mathrel{+}= \frac{\partial \dot{\mathbf{m}}_i}{\partial P_j}\delta P_j$
5: **end loop**
6: **end loop**
7: **set:** $\dot{\mathbf{m}}^{n+1} \Leftarrow \dot{\mathbf{m}}^*$

Algorithm 3.3 Updating Continuity and Momentum Variables

Upon completion of this process, a single timestep is considered to have been taken. The solution obtained via this process is subjected to several physical and computational limits discussed in Section 3.4.

3.2 Nonlinear COBRA

As obtained, the COBRA software used the timestep of the semi-implicit method as outlined in Section 3.1. In order to evaluate the proposed domain decomposition algorithm, COBRA needed to be modified to be able to take multiple Newton steps within a given timestep. To achieve this goal, several objectives were met. First, the software routines that calculated portions of $\mathbf{F}\left(\mathbf{x}^k\right)$ and $\mathbf{J}\left(\mathbf{x}^k\right)$ were evaluated and transitioned to the proper nonlinear discretization. Second, a method to identify the active portion of the domain was developed to correctly construct $\mathbf{x}^k$ and $\mathbf{F}^k$. Lastly, a Newton loop algorithm was implemented. This section will describe work done to meet the above objectives.

3.2.1 Evaluation of Discretization

Given the formation of the governing equations from Section 3.1, several modifications were required to implement the nonlinear solver. Using an iterative nonlinear solver requires that the governing equations be linearized about a tentative new-time value $\mathbf{x}^{n+1,k}$; this differs from the

linear case in which the linearization occurs about the old-time values $\mathbf{x}^n$. First, the correct linearization about a new-time guess value as opposed to the old-time value needed to be implemented. Each Newton step is an incremental change in the new-time variable, not a change in the independent parameters over a timestep. Second, memory saving techniques had been employed in the construction of the linear solver that precluded more than one Newton step. In particular, there had been the implicit assumption within the software that the first Newton iterate, $\mathbf{x}^{n+1,0}$, was the old-time variable $\mathbf{x}^n$. This design decision required that all subroutines involved with the evaluation of the components of both the nonlinear residual and its Jacobian be vetted for accurate usage of variables. Third, the assumption that $\mathbf{x}^{n+1,k} = \mathbf{x}^n$ produced source code that was inconsistent with an iterative Newton method. The source code was modified to reflect the intended discretization of the governing conservation laws. To do this, areas had to be identified where: there were implicit cancellation of terms, the new-time variables were used in place of old-time variables, and the old-time variables were used in place of new-time variables. This required each of the Fortran procedures involved with the calculation of the various physical parameters and operators to be evaluated to determine if the variables being used were correct. Where appropriate, the software was changed to reflect the distinction between old-time and tentative new-time variables and to introduce terms that had been assumed equal to zero. Finally, upon verification of the proper discretization of the governing equations, several architectural changes were also made to expedite the implementation of the nonlinear solver.

3.2.2 Active Domain Determination

In COBRA the smallest geometric component that can be input is the subchannel. As mentioned in Section 2.1 subchannels inherit their geometric properties from the section in which they reside. To create geometrically complicated problems, it is possible to enforce zero momentum in a given momentum flow path, decoupling adjacent continuity volumes by eliminating the flow paths that connect them. This constraint is known as a no-flow boundary condition, and it allows for more complicated models to be created by isolating portions of the domain. If an isolated domain is not connected to an external boundary condition, then it is marked as an inactive region.

The inactive regions are thermodynamically isolated and form a closed system. These areas of the domain do not influence the solution in the portion of the domain that has boundary values, known as the active region. After evaluating several prototypical models, it was determined that a non-trivial portion of the domain of these problems were thermodynamically isolable. Figure 3.1 shows an example of a domain with inactive regions, where the inactive portions of the domain are marked with grey, the active with white, the external boundary conditions with black, and the inter-subchannel flow paths are marked with dashed lines.

Originally, COBRA would solve the momentum and continuity equations at every defined location in the problem. The pressure matrix, $\mathbf{A}$, was sized based upon the number of volumes in the domain, not the number of active volumes. During this work it was determined that those continuity volumes and momentum flow paths which were in the inactive region would be pruned from the simulation. While the additional costs of including these volumes in the simulation have historically been acceptable, the use of an iterative solver would amplify the computational time spent in those regions to an unacceptable degree. Removing these inactive volumes saved computational time by precluding the evaluation of the residuals at those locations and by modifying the size of the pressure matrix (3.17). Additionally, since the inactive regions of the problem are interpreted as not being part of the system being modeled, including their equations in the residual would not provide an accurate measure of convergence.

To determine which areas of the domain were active and which ones were inactive, a geometry traversing routine was written. This routine parses the COBRA input file and creates an adjacency list data structure to represent the active portion of the domain. The data structure allowed the active volumes to be treated independently from the memory structure of the software. Once the active volumes were identified, the way in which the domain information was stored in memory was changed.

3.2.3 Nonlinear Algorithm

During this research, the COBRA software was transitioned from being restricted to using only a linear solver, as outlined in Section 3.1, to being able to use either a linear solver or a

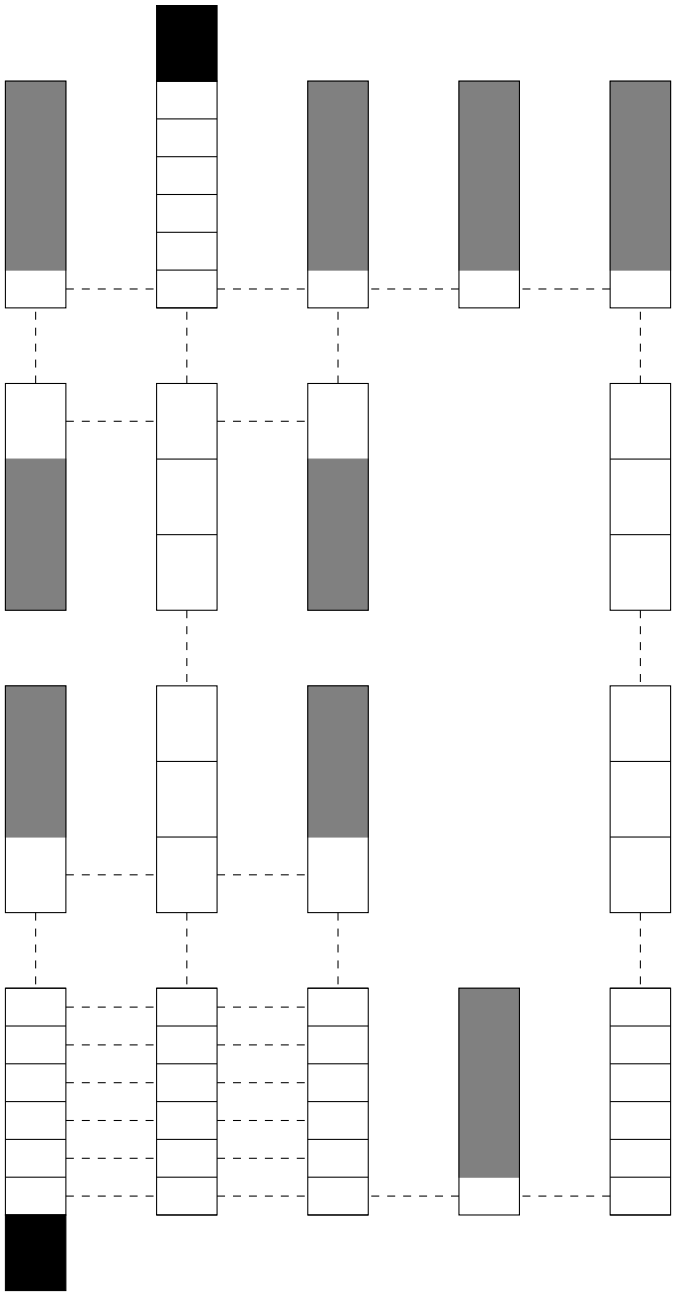

Figure 3.1 A Domain with Inactive Regions

nonlinearly convergent solver. The solution method selected for the nonlinear system was the iterative Newton's method. Newton's method uses successive linearizations to find a solution vector that adequately satisfies the system of discrete nonlinear equations. The algorithmic outline for the nonlinear solver is presented in Algorithm 3.4.

Require: $\mathbf{x}^0$ and t^0
1: **set:** $n = 0$
2: **loop** Transient Loop
3: **set:** $t^{n+1} := t^n + \Delta t$
4: **set:** $k = 0$
5: **algorithm:** Assemble Nonlinear Pressure Matrix ▷ Algorithm 3.2
6: **solve:** $\mathbf{A}^k \delta \mathbf{P}^k = \mathbf{res}^k$
7: **algorithm:** Update Nonlinear Variables ▷ Algorithm 3.3
8: **loop** Newton Loop
9: **algorithm:** Assemble Nonlinear Pressure Matrix ▷ Algorithm 3.2
10: **algorithm:** Convergence Determination ▷ Algorithm 3.5
11: **if end** Newton loop **then**
12: **break** Newton Loop
13: **end if**
14: **set:** $k \mathrel{+}= 1$
15: **solve:** $\mathbf{A}^k \delta \mathbf{P}^k = \mathbf{res}^k$
16: **algorithm:** Update Nonlinear Variables ▷ Algorithm 3.3
17: **end loop**
18: **set:** $n \mathrel{+}= 1$
19: **end loop**

Algorithm 3.4 Nonlinear COBRA algorithm.

The following section will roughly parallel Algorithm 3.4. First, the method by which the nonlinear residuals are evaluated and calculated is presented. This portion will be an extension of the procedure discussed in Section 3.1 and outlined in Algorithm 3.2. Second, the process through which the Newton vector is updated, Algorithm 3.3, is expounded upon and discussed. Lastly, the methods by which convergence is determined are discussed and presented in Algorithm 3.5.

A detailed discussion of the process through which the nonlinear residuals are evaluated and the pressure matrix is assembled will now be presented. Two quantities that are important to the iterative solver are the residual vector, $\mathbf{F}^k$, and the update vector, $\delta \mathbf{x}$. The update vector represents the changes in the solution vector that result from a given Newton step. The residual vector is composed of the residuals from the governing PDEs that are being solved. This includes both the residuals from the continuity volumes and the residuals from the momentum flow paths.

Every momentum flow path has associated with it three momentum equation residuals (3.18) – (3.20). The vector notation for the three momentum equations is $\mathbf{F}_p$.

$$F_{p,l}^k = \dot{m}_l^{n+1,k} - \dot{m}_l^n + \frac{\Delta t}{\Delta x}\left(\sum_{i\,\in\,N_c}\left(\langle\alpha_l\rho_l u_l\rangle_{\mathrm{d}}\langle u\rangle_{\mathrm{a},l}\tilde{A}\right)_i^n + \langle\alpha_l\rangle_{\mathrm{a}}^n\nabla P^{\,n+1,k} - g\langle\alpha_l\rho_l\rangle_{\mathrm{a}}^n\right.$$
$$\left. +\ K_{wl}^n(\dot{m}_l^{n+1,k})^2 - K_{i,gl}^n(u_l^{n+1,k} - u_g^{n+1,k})^2 + \left[(1-\eta)\dot{\Gamma}u^{'} + \dot{\Upsilon}u^{'}\right]^n\right) \tag{3.18}$$

$$F_{p,g}^k = \dot{m}_g^{n+1,k} - \dot{m}_g^n + \frac{\Delta t}{\Delta x}\left(\sum_{i\,\in\,N_c}\left(\langle\alpha_g\rho_g u_g\rangle_{\mathrm{d}}\langle u\rangle_{\mathrm{a},g}\tilde{A}\right)_i^n + \langle\alpha_g\rangle_{\mathrm{a}}^n\nabla P^{\,n+1,k} - g\langle\alpha_g\rho_g\rangle_{\mathrm{a}}^n\right.$$
$$\left. +\ K_{wg}^n(\dot{m}_g^{n+1,k})^2 + K_{i,gl}^n(u_l^{n+1,k} - u_g^{n+1,k})^2 + K_{i,ge}^n(u_e^{n+1,k} - u_g^{n+1,k})^2 - (\dot{\Gamma}u^{'})^n\right) \tag{3.19}$$

$$F_{p,e}^k = \dot{m}_e^{n+1,k} - \dot{m}_e^n + \frac{\Delta t}{\Delta x}\left(\sum_{i\,\in\,N_c}\left(\langle\alpha_e\rho_l u_e\rangle_{\mathrm{d}}\langle u\rangle_{\mathrm{a},e}\tilde{A}\right)_i^n + \langle\alpha_e\rangle_{\mathrm{a}}^n\nabla P^{\,n+1,k} - g\langle\alpha_e\rho_l\rangle_{\mathrm{a}}^n\right.$$
$$\left. +\ K_{we}^n(\dot{m}_e^{n+1,k})^2 - K_{i,ge}^n(u_e^{n+1,k} - u_g^{n+1,k})^2 + \left[\eta\dot{\Gamma}u^{'} - \dot{\Upsilon}u^{'}\right]^n\right) \tag{3.20}$$

These three momentum equations are now linearized about their tentative new-time values. The system of linear equations that results from this linearization is shown in (3.21).

$$\frac{\partial\,\mathbf{F}_{\,p}^{\,k}}{\partial\dot{m}_l^{n+1}}\delta\dot{m}_l^k + \frac{\partial\,\mathbf{F}_{\,p}^{\,k}}{\partial\dot{m}_g^{n+1}}\delta\dot{m}_g^k + \frac{\partial\,\mathbf{F}_{\,p}^{\,k}}{\partial\dot{m}_e^{n+1}}\delta\dot{m}_e^k + \sum_{i\,\in\,N_c}\frac{\partial\,\mathbf{F}_{\,p}^{\,k}}{\partial P_i}\delta P_i^k = -\mathbf{F}_{\,p}^{\,k} \tag{3.21}$$

The first three columns on the right-hand side of (3.21) represent the derivatives of the momentum residual with respect to the momentum variables. These three columns will be referred to as the Jacobian matrix for a given momentum flow path, $\mathbf{J}_{\,p}^{\,k}$, defined in (3.22).

$$\mathbf{J}_{\,p}^{\,k} = \begin{bmatrix} \frac{\partial F_{p,l}^k}{\partial\dot{m}_l^{n+1}} & \frac{\partial F_{p,l}^k}{\partial\dot{m}_g^{n+1}} & \frac{\partial F_{p,l}^k}{\partial\dot{m}_e^{n+1}} \\ \frac{\partial F_{p,g}^k}{\partial\dot{m}_l^{n+1}} & \frac{\partial F_{p,g}^k}{\partial\dot{m}_g^{n+1}} & \frac{\partial F_{p,g}^k}{\partial\dot{m}_e^{n+1}} \\ \frac{\partial F_{p,e}^k}{\partial\dot{m}_l^{n+1}} & \frac{\partial F_{p,e}^k}{\partial\dot{m}_g^{n+1}} & \frac{\partial F_{p,e}^k}{\partial\dot{m}_e^{n+1}} \end{bmatrix} \tag{3.22}$$

Using (3.4) and (3.22), (3.21) can be expressed as (3.23).

$$\mathbf{J}_{\,p}^{\,k}\delta\dot{\mathbf{m}}^{\,k} = -\mathbf{F}_{\,p}^{\,k} - \sum_{i\,\in\,N_c}\frac{\partial\,\mathbf{F}_{\,p}^{\,k}}{\partial P_i}\delta P_i^k \tag{3.23}$$

This system of equations, (3.23), is then solved for the updated new-time momentum vector via the procedure shown in (3.24).

$$
\begin{aligned}
\mathbf{J}_p^k \delta\dot{\mathbf{m}}^k &= -\mathbf{F}_p^k - \sum_{i \in N_c} \frac{\partial \mathbf{F}_p^k}{\partial P_i} \delta P_i^k \\
\mathbf{J}_p^k \left[\dot{\mathbf{m}}^{n+1,k+1} - \dot{\mathbf{m}}^{n+1,k} \right] &= -\mathbf{F}_p^k - \sum_{i \in N_c} \frac{\partial \mathbf{F}_p^k}{\partial P_i} \delta P_i^k \\
\mathbf{J}_p^k \dot{\mathbf{m}}^{n+1,k+1} &= -\mathbf{F}_p^k + \mathbf{J}_p^k \dot{\mathbf{m}}^{n+1,k} - \sum_{i \in N_c} \frac{\partial \mathbf{F}_p^k}{\partial P_i} \delta P_i^k \\
\dot{\mathbf{m}}^{n+1,k+1} &= \left[\mathbf{J}_p^k \right]^{-1} \left[-\mathbf{F}_p^k + \mathbf{J}_p^k \dot{\mathbf{m}}^{n+1,k} \right] - \sum_{i \in N_c} \left[\mathbf{J}_p^k \right]^{-1} \left[\frac{\partial \mathbf{F}_p^k}{\partial P_i} \right] \delta P_i^k \\
\dot{\mathbf{m}}^{n+1,k+1} &= \dot{\mathbf{m}}^{n+1,k+\frac{1}{2}} + \sum_{i \in N_c} \frac{\partial \dot{\mathbf{m}}^k}{\partial P_i} \delta P_i^k
\end{aligned}
\tag{3.24}
$$

The variable $\dot{\mathbf{m}}^{n+1,k+\frac{1}{2}}$, defined in (3.25) and present in (3.24), is the iterative analog to $\dot{\mathbf{m}}^*$ from (3.8).

$$
\begin{aligned}
\dot{\mathbf{m}}^{n+1,k+\frac{1}{2}} &= \left[\mathbf{J}_p^k \right]^{-1} \left[\mathbf{J}_p^k \dot{\mathbf{m}}^{n+1,k} - \mathbf{F}_p^k \right] \\
\dot{\mathbf{m}}^{n+1,k+\frac{1}{2}} &= \dot{\mathbf{m}}^{n+1,k} - \left[\mathbf{J}_p^k \right]^{-1} \mathbf{F}_p^k \\
\dot{\mathbf{m}}^{n+1,k+\frac{1}{2}} &= \dot{\mathbf{m}}^{n+1,k} + \delta\dot{\mathbf{m}}^*
\end{aligned}
\tag{3.25}
$$

For algebraic convenience, the update vector, $\delta\dot{\mathbf{m}}^k$, is defined in (3.26).

$$
\begin{aligned}
\delta\dot{\mathbf{m}}^k &= -\left[\mathbf{J}_p^k \right]^{-1} \mathbf{F}_p^k - \sum_{i \in N_c} \left[\mathbf{J}_p^k \right]^{-1} \left[\frac{\partial \mathbf{F}_p^k}{\partial P_i} \right] \delta P_i^k \\
\delta\dot{\mathbf{m}}^k &= \delta\dot{\mathbf{m}}^* + \sum_{i \in N_c} \frac{\partial \dot{\mathbf{m}}}{\partial P_i} \delta P_i^k
\end{aligned}
\tag{3.26}
$$

Likewise, each continuity volume has associated with it six residuals representing the mass and energy equations (3.27) – (3.32). For a given continuity volume, these six equations are collectively referred to as $\mathbf{F}_c$.

$$F^k_{m,n} = V_c\left[(\alpha_g\rho_n)^{n+1,k} - (\alpha_g\rho_n)^n\right] + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha^n_g\rho^n_n\rangle^{n+1,k}_d u^{n+1,k}_g A_p\right)_i \tag{3.27}$$

$$\begin{aligned} F^k_{m,l} &= V_c\left(\alpha_l\rho_l\right)^{n+1,k} - V_c\left(\alpha_l\rho_l\right)^n + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha^n_l\rho^n_l\rangle^{n+1,k}_d u^{n+1,k}_l A_p\right)_i \\ &\quad + \left[(1-\eta)\Gamma + \Upsilon\right]^{n+1,k} \end{aligned} \tag{3.28}$$

$$\begin{aligned} F^k_{h,g} &= V_c\left[\left(\alpha_g\rho_g h_g\right)^{n+1,k} - \left(\alpha_g\rho_g h_g\right)^n - \alpha^n_g(P^{\,n+1,k} - P^{\,n})\right] - q^n_{wg} \\ &\quad - \left[q_{i,v} + \Gamma h^{'}_v + q_{gl}\right]^{n+1,k} + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha^n_g\rho^n_g h^n_g\rangle^{n+1,k}_d u^{n+1,k}_g A_p\right)_i \end{aligned} \tag{3.29}$$

$$\begin{aligned} F^k_{h,l} &= V_c\left[(\alpha_l\rho_l h_l)^{n+1,k} - (\alpha_l\rho_l h_l)^n - \alpha^n_l(P^{\,n+1,k} - P^{\,n})\right] - \left[q_{i,l} - \Gamma h^{'}_l - q_{gl}\right]^{n+1,k} \\ &\quad + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha^n_l\rho^n_l h^n_l\rangle^{n+1,k}_d u^{n+1,k}_l A_p + \langle\alpha^n_e\rho^n_l h^n_l\rangle^{n+1,k}_d u^{n+1,k}_e A_p\right)_i - q^n_{wl} \end{aligned} \tag{3.30}$$

$$\begin{aligned} F^k_{m,e} &= V_c\left(\alpha_e\rho_l\right)^{n+1,k} - V_c\left(\alpha_e\rho_l\right)^n + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha^n_e\rho^n_l\rangle^{n+1,k}_d u^{n+1,k}_e A_p\right)_i \\ &\quad - \left[\Upsilon - \eta\Gamma\right]^{n+1,k} \end{aligned} \tag{3.31}$$

$$F^k_{m,v} = V_c\left[\left(\alpha_g\rho_v\right)^{n+1,k} - \left(\alpha_g\rho_v\right)^n\right] + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha^n_g\rho^n_v\rangle^{n+1,k}_d u^{n+1,k}_g A_p\right)_i - \Gamma^{n+1,k} \tag{3.32}$$

For each continuity volume, the above six equations are linearized about the tentative new-time independent parameters.

$$\begin{aligned} -\mathbf{F}^{\,k}_{\,c} = &\frac{\partial\mathbf{F}^{\,k}_{\,c}}{\partial(\alpha_g P_n)}\delta(\alpha_g P_n)^k + \frac{\partial\mathbf{F}^{\,k}_{\,c}}{\partial\alpha_g}\delta\alpha^k_g + \frac{\partial\mathbf{F}^{\,k}_{\,c}}{\partial(\alpha_g h_v)}\delta(\alpha_g h_v)^k + \frac{\partial\mathbf{F}^{\,k}_{\,c}}{\partial((1-\alpha_g)h_l)}\delta((1-\alpha_g)h_l)^k \\ &+ \frac{\partial\mathbf{F}^{\,k}_{\,c}}{\partial\alpha_e}\delta\alpha^k_e + \frac{\partial\mathbf{F}^{\,k}_{\,c}}{\partial P}\delta P^k + \sum_{i\,\in\, N_f}\frac{\partial\mathbf{F}^{\,k}_{\,c}}{\partial\dot{\mathbf{m}}_{\,i}}\delta\dot{\mathbf{m}}^{\,k}_{\,i} \end{aligned} \tag{3.33}$$

The matrix of the derivatives of the continuity residual vector with respect to the momentum vector at the boundaries of the continuity volume, $\dfrac{\partial\,\mathbf{F}_{\,c}}{\partial\dot{\mathbf{m}}_{\,i}}$, will be known as $\boldsymbol{\Xi}_{\,i}$. This matrix, (3.34), converts the momentum at a continuity volume's edge to mass and energy flow rates.

$$\Xi_i^{n+1,k} = \begin{bmatrix} 0 & \frac{\langle \alpha_g^n \rho_n^n \rangle_d^{n+1,k}}{\langle \alpha_g \rho_g \rangle_a^n} & 0 \\ \frac{\langle \alpha_l^n \rho_l^n \rangle_d^{n+1,k}}{\langle \alpha_l \rho_l \rangle_a^n} & 0 & 0 \\ 0 & \frac{\langle \alpha_g^n \rho_g^n h_g^n \rangle_d^{n+1,k}}{\langle \alpha_g \rho_g \rangle_a^n} & 0 \\ \frac{\langle \alpha_l^n \rho_l^n h_l^n \rangle_d^{n+1,k}}{\langle \alpha_l \rho_l \rangle_a^n} & 0 & \frac{\langle \alpha_e^n \rho_l^n h_l^n \rangle_d^{n+1,k}}{\langle \alpha_e \rho_l \rangle_a^n} \\ 0 & 0 & \frac{\langle \alpha_e^n \rho_l^n \rangle_d^{n+1,k}}{\langle \alpha_e \rho_l \rangle_a^n} \\ 0 & \frac{\langle \alpha_g^n \rho_v^n \rangle_d^{n+1,k}}{\langle \alpha_g \rho_g \rangle_a^n} & 0 \end{bmatrix}_i \tag{3.34}$$

The definition of the momentum update vector, (3.26), is used to eliminate the momentum update vector from (3.33). In the above equations, a given momentum flow path will be referenced by the index i. The summation over the number of continuity volumes to which a particular flow path is connected is at most two terms. Since each momentum flow path can connect to only two continuity volumes, one of those two continuity volumes will be the current continuity volume, which will be marked by the index $s(i)$, and the other continuity volume will be indexed with $o(i)$. The resulting system is shown in (3.35).

$$\frac{\partial \mathbf{F}_c^k}{\partial(\alpha_g P_n)}\delta(\alpha_g P_n)^k + \frac{\partial \mathbf{F}_c^k}{\partial \alpha_g}\delta\alpha_g^k + \frac{\partial \mathbf{F}_c^k}{\partial(\alpha_g h_v)}\delta(\alpha_g h_v)^k + \frac{\partial \mathbf{F}_c^k}{\partial((1-\alpha_g)h_l)}\delta((1-\alpha_g)h_l)^k + \frac{\partial \mathbf{F}_c^k}{\partial \alpha_e}\delta\alpha_e^k + \frac{\partial \mathbf{F}_c^k}{\partial P}\delta P^k + \Delta t \sum_{i \in N_f} \Xi_i^k \left(\delta \dot{\mathbf{m}}_i^* + \frac{\partial \dot{\mathbf{m}}_i^k}{\partial P_{s(i)}}\delta P_{s(i)}^k + \frac{\partial \dot{\mathbf{m}}_i^k}{\partial P_{o(i)}}\delta P_{o(i)}^k \right) = \mathbf{F}_c^k \tag{3.35}$$

By rearranging (3.35) and recognizing that the $\delta P_{s(i)}$ terms from the momentum equations correspond to the given volumes own δP, (3.36) is obtained.

$$\frac{\partial \mathbf{F}_c^k}{\partial(\alpha_g P_n)}\delta(\alpha_g P_n)^k + \frac{\partial \mathbf{F}_c^k}{\partial \alpha_g}\delta\alpha_g^k + \frac{\partial \mathbf{F}_c^k}{\partial(\alpha_g h_v)}\delta(\alpha_g h_v)^k + \frac{\partial \mathbf{F}_c^k}{\partial((1-\alpha_g)h_l)}\delta((1-\alpha_g)h_l)^k + \frac{\partial \mathbf{F}_c^k}{\partial \alpha_e}\delta\alpha_e^k + \left(\frac{\partial \mathbf{F}_c^k}{\partial P} + \Delta t \sum_{i \in N_f} \Xi_i^k \frac{\partial \dot{\mathbf{m}}_i^k}{\partial P_{s(i)}} \right) \delta P^k + \Delta t \sum_{i \in N_f} \Xi_i^k \frac{\partial \dot{\mathbf{m}}_i^k}{\partial P_{o(i)}}\delta P_{o(i)}^k = -\mathbf{F}_c^k - \Delta t \sum_{i \in N_f} \Xi_i^k \delta \dot{\mathbf{m}}_i^* \tag{3.36}$$

In (3.36) there are six equations and six plus the number of connecting flow paths, N_f, unknowns. The first six columns of (3.36) will be represented by $\mathbf{J}_c^k$. The next N_f columns, representing the inter-continuity volume coupling, will be denoted by $\mathbf{K}_c^k$. Notice that the right-hand side of (3.36) is the residual evaluated at the tentative new-time values plus the change in the momentum vector multiplied by $\boldsymbol{\Xi}$. Since the continuity equations are linear functions of their flow paths' momentum vectors, this combination is the equivalent of using $\dot{\mathbf{m}}^{n+1,k+\frac{1}{2}}$ for the advecting velocity in the evaluation of the continuity residuals. Due to the use of the tentative new-time momenta in the advection terms, there is now a discrepancy between the residuals, $\mathbf{F}_c^k$, for the continuity volume and the right-hand side of the linear system, $\mathbf{r}_c^k$. Note that for convergence determination, the residual, $\mathbf{F}_c^k$, is used, not the modified right-hand side in (3.36). The vector of unknowns is $\delta\mathbf{C}_c^k$, which is the iterative analog of (3.16). Using the above matrix notation, (3.36) can be represented as (3.37).

$$\left[\mathbf{J}_c^k | \mathbf{K}_c^k\right] \delta\mathbf{C}_c^k = \mathbf{r}_c^k \tag{3.37}$$

This rectangular, linear system for the nonlinear boundary volume is then subjected to partial $\mathbf{LU}$ decomposition without pivoting. This $\mathbf{LU}$ decomposition is such that the lower triangular matrix has ones along the diagonal. The final row of the linear system in (3.38) is then scaled by the sixth row's sixth column, $\mathbf{U}_c^k[6,6]$.

$$\left[\mathbf{U}_c^k | \left[\mathbf{L}_c^k\right]^{-1} \mathbf{K}_c^k\right] \delta\mathbf{C}_c^k = \left[\mathbf{L}_c^k\right]^{-1} \mathbf{r}_c^k \tag{3.38}$$

This final, scaled row of (3.38) is referred to as the pressure equation for a given continuity volume. The pressure equations for all continuity volumes are collated into the global pressure matrix according to the continuity volumes' ordinal. This pressure matrix is then inverted to solve for the pressure update for all continuity volumes. At this point all continuity volumes are looped over to solve their linear systems, (3.38), for the non-pressure portion of their update vectors, $\delta\mathbf{C}_c^k$.

The above discussion expounds upon the algorithm for a single Newton step that is given by Algorithm 3.2. It is during this process that both the residuals and the scale factors, $\mathbf{F}^k$ and $\mathbf{S}^k$, are evaluated and stored. Concurrently, the pressure matrix is being both constructed and solved.

In order to obtain the new-time momentum variables, (3.24) is used during Algorithm 3.3. For the purpose of convergence determination, the change in the momentum vectors, $\delta\dot{\mathbf{m}}$, is calculated for each flow path, (3.39). This is necessary since the momentum equations are solved for the new-time variables directly, not their updates.

$$\delta\dot{\mathbf{m}}^{\,k} = \dot{\mathbf{m}}^{\,n+1,k+\frac{1}{2}} + \sum_{i \in N_c} \frac{\partial\dot{\mathbf{m}}^{\,k}}{\partial P_i}\delta P_i^k - \dot{\mathbf{m}}^{\,n+1,k} \tag{3.39}$$

The residuals from every continuity volume and momentum flow path in the active domain are assembled into the $\mathbf{F}^{\,k}$ vector. The degree to which a given solution vector, $\mathbf{x}^{\,n+1,k}$, satisfies the governing PDEs can be measured by a norm of the residual vector of the nonlinear system. If the exact solution for the nonlinear system were known, the residual vector would be identically zero; however, in practice the residual will never be identically zero. As such it is necessary to discuss the various means by which a solution for a timestep is determined to be sufficiently accurate.

While a Newton step is being performed, the residual is being evaluated with the current solution vector, $\mathbf{x}^{\,n+1,k}$. Once the Newton update vector, $\delta\mathbf{x}^{\,k}$, has been calculated, the residual is reevaluated using the new solution vector, $\mathbf{x}^{\,n+1,k+1}$. Based upon the discussion above, the process for evaluating the residual is the same as that of assembling the pressure matrix. Using the two residuals, $\mathbf{F}^{\,k}$ and $\mathbf{F}^{\,k+1}$, the comparative accuracy of the two solution vectors can be determined. These residual vectors are used in the evaluation of the Newton loop termination criteria.

The iterative Newton loop can be terminated in three ways. One way is that the Newton loop will terminate if the $\mathcal{L}_2$ norm of the scaled residual vector, (3.40), divided by the square root of the number of unknowns, N_u, drops below a specified threshold, F_{tol}.

$$\tilde{\mathbf{F}}^{\,k} = \left[\mathbf{S}^{\,k}\right]^{-1}\mathbf{F}^{\,k} \tag{3.40}$$

Another way is to have the $\mathcal{L}_2$ norm of the relative update vector, (3.41), divided by the square root of the number of entries in the update vector, N_u, drop below a specified threshold δ_{tol}.

$$\delta\tilde{\mathbf{x}}^{\,k} = \frac{\delta\mathbf{x}^{\,k}}{\mathbf{x}^{\,n+1,k}} = \frac{\mathbf{x}^{\,n+1,k+1} - \mathbf{x}^{\,n+1,k}}{\mathbf{x}^{\,n+1,k}} \tag{3.41}$$

Finally, the Newton loop will terminate if the number of Newton iterates, k, exceeds a specified threshold, $k_{\max}$.

The scaling matrix, $\mathbf{S}^k$, will be discussed in detail in Section 3.3. Note that it is assembled and stored during the process of assembling the pressure matrix, Algorithm 3.2. The convergence determination logic for when to end the Newton loop is given in Algorithm 3.5.

1: **if** $\frac{||\tilde{\mathbf{F}}^{k+1}||_2}{\sqrt{N_u}} \leq F_{\text{tol}}$ **then**
2: **end** Newton Loop
3: **else if** $\frac{||\delta\tilde{\mathbf{x}}^k||_2}{\sqrt{N_u}} \leq \delta_{\text{tol}}$ **then**
4: **end** Newton Loop
5: **else if** $k > k_{\max}$ **then**
6: **end** Newton Loop
7: **end if**

Algorithm 3.5 Convergence Determination of Newton Loop

In order to determine convergence according to Algorithm 3.5, it is necessary to have the values of $\mathbf{F}^{k+1}$ and $\mathbf{S}^{k+1}$. This is accomplished by looping over the domain and assembling the pressure matrix again. As shown in Algorithm 3.2, the residual and scale factor are both evaluated during this process. If Algorithm 3.5 determines that the Newton loop should not end, then the pressure matrix has already been assembled and can then be solved for the new update vector.

3.3 Operator-Based Scaling

An important aspect of the nonlinear solver outlined in Section 3.2 is the convergence criteria. As part of this work a novel operator-based scaling has been developed to obtain meaningful convergence thresholds. The iterative solver depends upon the scaled residual to help determine when convergence has been achieved. In particular, various norms of the residual vector are evaluated to measure the degree to which the nonlinear system of algebraic equations is being satisfied. The evaluation of residual norms necessitates the use of a scaling matrix for the residual vector.

Without proper scaling the residual has several negative characteristics [17, 31]. The different conservation equations will have different orders of magnitude at a given point in time because of

the physical quantities that they represent. Over the course of a transient the phasic composition of a volume can vary dramatically as phases appear and disappear. When norms are taken of the unscaled residual those equations whose terms are the largest may have a residual that is orders of magnitude greater than those equations whose terms are smallest. For convergence criteria that are based upon relative convergence, the reduction of those equations with larger magnitudes would result in apparent convergence without any consideration given to those equations whose scale is orders of magnitude smaller. This would create a situation where any norm taken of the residual would be biased towards certain equations based on their units or phasic composition. To avoid this, an operator-based scaling strategy was developed and implemented.

For a given continuity volume, the nonlinear residual will have six components: four for the conservation of mass and two for the conservation of energy. For each momentum flow path, the three conservation of momentum equations will form the three components of the nonlinear residual. These residuals have the units of the conserved quantities for their corresponding PDEs; Table 3.1 shows the units for the different conservation equations.

Residual	Units
Conservation of the Non-Condensable Gas Field Mass	[lb_m]
Conservation of the Continuous Liquid Water Field Mass	[lb_m]
Conservation of the Entrained Liquid Water Field Mass	[lb_m]
Conservation of the Water Vapor Field Mass	[lb_m]
Conservation of the Gaseous Phase Enthalpy	[BTU]
Conservation of the Liquid Phase Enthalpy	[BTU]
Conservation of the Continuous Liquid Field Momentum	[$\frac{lb_m ft}{s}$]
Conservation of the Entrained Liquid Field Momentum	[$\frac{lb_m ft}{s}$]
Conservation of the Gaseous Phase Momentum	[$\frac{lb_m ft}{s}$]

Table 3.1 Residuals and their units.

For the aforementioned reasons, it is desirable to scale the residual. A challenge that has been addressed in this work is the development of a method for scaling of these residuals that is based upon the local physics of interest at any given point in the transient. In constructing this scaling factor it was determined that the following characteristics were desirable:

- $(S_i^{-1} F_i)^k \approx 1$ when $\mathbf{x}^k$ is a "poor" solution.

- $(S_i^{-1}F_i)^k \approx 0$ when $\mathbf{x}^k$ is a "good" solution.
- $(S_i^{-1}F_i)^k \to 0$ when phase i disappears.
- $0 \leq \left|S_i^{-1}F_i^k\right|^k \leq 1$ for all values of $\mathbf{x}_i^k$.

The dynamic behavior of the residual necessitates a method for scaling that adapts to relevant physical situations. The scaling method developed during this work is an operator-based approach. The governing PDEs can be viewed as a collection of operators, linear and nonlinear, acting upon the vector of independent parameters. The summation of these operators must balance to zero for the nonlinear equation to be satisfied. The scaling factor developed uses the magnitudes of these operators to determine an absolute measure of the physical processes occurring in a given volume. This is accomplished by summing the absolute value of the different discrete operators in the governing equations. This scaling creates a relative measure of the nonlinear residual when compared to the magnitude of the physics involved in the process.

$$S_{p,l}^k = \frac{\Delta t}{\Delta x}\left[\left|\frac{\Delta x\left[\dot{m}_l^{n+1,k} - \dot{m}_l^n\right]}{\Delta t}\right| + \left|\sum_{i\,\in\, N_c}\left(\langle\alpha_l\rho_l u_l\rangle_{\mathrm{d}}\langle u\rangle_{\mathrm{a},l}\tilde{A}\right)_i^n\right| + \left|\langle\alpha_l\rangle_{\mathrm{a}}^n\nabla P^{\,n+1,k}\right| + \left|g\langle\alpha_l\rho_l\rangle_{\mathrm{a}}^n\right| + \left|K_{wl}^n(\dot{m}_l^{n+1,k})^2\right| + \left|K_{i,gl}^n(u_l^{n+1,k} - u_g^{n+1,k})^2\right| + \left|\left[(1-\eta)\dot{\Gamma}u'\right]^n\right| + \left|\left[\dot{\Upsilon}u'\right]^n\right|\right] \tag{3.42}$$

$$S_{p,g}^k = \frac{\Delta t}{\Delta x}\left[\left|\frac{\Delta x\left[\dot{m}_g^{n+1,k} - \dot{m}_g^n\right]}{\Delta t}\right| + \left|\sum_{i\,\in\, N_c}\left(\langle\alpha_g\rho_g u_g\rangle_{\mathrm{d}}\langle u\rangle_{\mathrm{a},g}\tilde{A}\right)_i^n\right| + \left|\langle\alpha_g\rangle_{\mathrm{a}}^n\nabla P^{\,n+1,k}\right| + \left|g\langle\alpha_g\rho_g\rangle_{\mathrm{a}}^n\right| + \left|K_{wg}^n(\dot{m}_g^{n+1,k})^2\right| + \left|K_{i,gl}^n(u_l^{n+1,k} - u_g^{n+1,k})^2\right| + \left|K_{i,ge}^n(u_e^{n+1,k} - u_g^{n+1,k})^2\right| + \left|\left[\dot{\Gamma}u'\right]^n\right|\right] \tag{3.43}$$

$$S_{p,e}^k = \frac{\Delta t}{\Delta x}\left[\left|\frac{\Delta x\left[\dot{m}_e^{n+1,k} - \dot{m}_e^n\right]}{\Delta t}\right| + \left|\sum_{i\,\in\, N_c}\left(\langle\alpha_e\rho_l u_e\rangle_{\mathrm{d}}\langle u\rangle_{\mathrm{a},e}\tilde{A}\right)_i^n\right| + \left|\langle\alpha_e\rangle_{\mathrm{a}}^n\nabla P^{\,n+1,k}\right| + \left|g\langle\alpha_e\rho_l\rangle_{\mathrm{a}}^n\right|$$

$$+ \left| K_{we}^{n}(\dot{m}_{e}^{n+1,k})^2 \right| + \left| K_{i,ge}^{n}(u_{e}^{n+1,k} - u_{g}^{n+1,k})^2 \right| + \left| \left[\eta \dot{\Gamma} u^{'} \right]^{n} \right| + \left| \left[\dot{\Upsilon} u^{'} \right]^{n} \right| \Bigg] \tag{3.44}$$

For the momentum equations, (3.1) – (3.3), the scaled residuals are shown in (3.42) – (3.44). Likewise, the scale factors for the continuity equations, (3.9) – (3.14), are given in (3.45) – (3.50).

$$S_{m,n}^{k} = \Delta t \left[\left| \frac{V_c \left[(\alpha_g \rho_n)^{n+1,k} - (\alpha_g \rho_n)^{n} \right]}{\Delta t} \right| + \sum_{i \in N_f} \left| \langle \alpha_g^n \rho_n^n \rangle_d^{n+1,k} u_g^{n+1,k} A_p \right|_i \right] \tag{3.45}$$

$$\begin{aligned} S_{m,l}^{k} &= \Delta t \left[\left| \frac{V_c \left[(\alpha_l \rho_l)^{n+1,k} - (\alpha_l \rho_l)^{n} \right]}{\Delta t} \right| + \sum_{i \in N_f} \left| \langle \alpha_l^n \rho_l^n \rangle_d^{n+1,k} u_l^{n+1,k} A_p \right|_i \right. \\ &\quad \left. + \left| \left[(1-\eta) \dot{\Gamma} \right]^{n+1,k} \right| + \left| \dot{\Upsilon}^{n+1,k} \right| \right] \end{aligned} \tag{3.46}$$

$$\begin{aligned} S_{h,g}^{k} &= \Delta t \left[\left| \frac{V_c \left[\left(\alpha_g \rho_g h_g \right)^{n+1,k} - \left(\alpha_g \rho_g h_g \right)^{n} \right]}{\Delta t} \right| + \left| \frac{V_c \alpha_g^n (P^{n+1,k} - P^n)}{\Delta t} \right| + \left| \dot{q}_{wg}^{n} \right| \right. \\ &\quad \left. + \left| \left[\dot{\Gamma} h_v^{'} \right]^{n+1,k} \right| + \left| \dot{q}_{i,v}^{n+1,k} \right| + \left| \dot{q}_{gl}^{n+1,k} \right| + \sum_{i \in N_f} \left| \langle \alpha_g^n \rho_g^n h_g^n \rangle_d^{n+1,k} u_g^{n+1,k} A_p \right|_i \right] \end{aligned} \tag{3.47}$$

$$\begin{aligned} S_{h,l}^{k} &= \Delta t \left[\left| \frac{V_c \left[(\alpha_l \rho_l h_l)^{n+1,k} - (\alpha_l \rho_l h_l)^{n} \right]}{\Delta t} \right| + \left| \frac{V_c \alpha_l^n (P^{n+1,k} - P^{n})}{\Delta t} \right| + |\dot{q}_{wl}^{n}| \right. \\ &\quad + \left| \left[\dot{\Gamma} h_l^{'} \right]^{n+1} \right| + \left| \dot{q}_{i,l}^{n+1,k} \right| + \left| \dot{q}_{gl}^{n+1,k} \right| + \sum_{i \in N_f} \left| \langle \alpha_l^n \rho_l^n h_l^n \rangle_d^{n+1,k} u_l^{n+1,k} A_p \right|_i \\ &\quad \left. + \sum_{i \in N_f} \left| \langle \alpha_e^n \rho_l^n h_l^n \rangle_d^{n+1,k} u_e^{n+1,k} A_p \right|_i \right] \end{aligned} \tag{3.48}$$

$$S_{m,e}^{k} = \Delta t \left[\left| \frac{V_c \left[(\alpha_e \rho_l)^{n+1,k} - (\alpha_e \rho_l)^{n} \right]}{\Delta t} \right| + \sum_{i \in N_f} \left| \left(\langle \alpha_e^n \rho_l^n \rangle_d^{n+1,k} u_e^{n+1,k} A_m \right) \right|_i \right.$$
$$\left. + \left| \dot{\Upsilon}^{n+1,k} \right| + \left| \left[\eta \dot{\Gamma} \right]^{n+1,k} \right| \right] \tag{3.49}$$

$$S_{m,v}^{k} = \Delta t \left[\left| \frac{V_c \left[\left(\alpha_g \rho_v \right)^{n+1,k} - \left(\alpha_g \rho_v \right)^{n} \right]}{\Delta t} \right| + \sum_{i \in N_f} \left| \langle \alpha_g^n \rho_v^n \rangle_d^{n+1,k} u_g^{n+1,k} A_p \right|_i \right.$$
$$\left. + \left| \dot{\Gamma}^{n+1,k} \right| \right] \tag{3.50}$$

The issue of phase transition also needed to be considered during this work. Since COBRA does not actually transition the governing equations to those for single-phase flow, there will always be a nonlinear residual for those phases that are nominally absent. It was observed that the effects of maintaining a depleted phase in the system of equations when solving the nonlinear problem created spurious convergence issues when using these scale factors. The unscaled residuals would be on the order of machine round-off. The operator-based scaling factors for these residuals would also be within orders of magnitude of machine round-off. However, the residual would be unable to decrease with additional Newton steps due to parametric constraints. This created the situation where the scaled nonlinear residual for the depleted field would stagnate at approximately $\mathcal{O}(1)$. These depleted residuals would dominate the norms used to determine convergence.

To overcome this deficiency it was determined that when a phase or field began to deplete, the scaling factor itself would be scaled to create an artificial decrease in the scaled residual to counter the artificial presence of the depleted field. This scaling function is shown by (3.51).

$$S_\phi = \max \left[1.0, \left(C_1 \frac{\alpha_{\phi,\text{MIN}}}{\alpha_\phi} \right)^{C_2} \right] S_\phi \tag{3.51}$$

For this work, the constant C_1 was set equal to 100, and the exponent C_2 was set equal to 10. This particular phase-transition scaling produced the regular scaling factor when the volume fraction of a phase was at least two orders of magnitude greater than the minimum volume fraction for that phase or field, which is on the order of $1.0\text{E-}6$. This drove the scaled residuals for phases that were nominally not present to well below those residuals for equations of the phases that were present.

Another phase depletion limit was a floor placed upon the scale factor. For each equation, the scale factor was limited to the minimum possible value of the conserved quantity based upon the volume fraction limits. For the mass equations it would be the minimum macroscopic density of the given field based upon the minimum volume fraction for that field. Table 3.2 shows the floors for the various residuals. These floors are updated during the iterative process.

Equation	Minimum
Conservation of the Non-Condensable Gas Field Mass	$\alpha_{g,\text{min}}\rho_n$
Conservation of the Continuous Liquid Water Field Mass	$\alpha_{l,\text{min}}\rho_l$
Conservation of the Entrained Liquid Water Field Mass	$\alpha_{e,\text{min}}\rho_l$
Conservation of the Water Vapor Field Mass	$\alpha_{g,\text{min}}\rho_v$
Conservation of the Gaseous Phase Enthalpy	$\alpha_{g,\text{min}}\rho_g h_g$
Conservation of the Liquid Phase Enthalpy	$(1.0-\alpha_{g,\text{min}})\rho_l h_l$
Conservation of the Continuous Liquid Field Momentum	$u_l \frac{\alpha_{l,\text{min}}\langle\rho_l\rangle_\text{a}}{\langle\alpha_l\rho_l\rangle_\text{a}} A_{mom}$
Conservation of the Entrained Liquid Field Momentum	$u_e \frac{\alpha_{e,\text{min}}\langle\rho_l\rangle_\text{a}}{\langle\alpha_e\rho_l\rangle_\text{a}} A_{mom}$
Conservation of the Gaseous Phase Momentum	$u_g \frac{\alpha_{g,\text{min}}\langle\rho_g\rangle_\text{a}}{\langle\alpha_g\rho_g\rangle_\text{a}} A_{mom}$

Table 3.2 Minimum conserved quantities for conservation equations.

3.4 Additional Considerations

There are several aspects of the solution algorithms outlined in Section 3.1 and Section 3.2 that require additional information. First, the quality control used during development is discussed. Second, the method for activating the nonlinear solver is provided. Third, several necessary and practical considerations required to allow the software to simulate the transition from single-phase to multi-phase flow are outlined. Lastly, the timestep selection and acceptance criteria are detailed.

3.4.1 Quality Control

The development of the nonlinear solver within COBRA took place under strict quality assurance guidelines. These guidelines have been documented in extensive detail [4] and represent the current best practices for thermal-hydraulic safety analysis. At every step of the development, the linear solver was required to maintain the same solution. This verification was dependent upon a larger number of verification and assessment problems. The output of the unmodified COBRA and the modified COBRA software was compared to machine precision. It was required that either the results of the two simulations be identical or that the reason for the difference be identified and understood. The COBRA software has the ability to repeat a timestep. As such, it was required that the backup capabilities continued to work while nonlinear solver was being implemented. Additionally, testing was done to ensure that the ability to restart the software mid-simulation was unaffected. While adding time to the development cycle, the overhead of the quality assurance procedures ensured that the linear solver could continue to be used for design purposes.

3.4.2 Nonlinear Input File

The primary input file for COBRA is fixed format, so it was decided that a separate input file for the nonlinear solver would provide the most flexibility when running simulations. This decision allowed for existing models to be run in nonlinear mode without modification of the actual input file. The presence of the nonlinear input file, "`nwt.cob`," triggers the input processing routine for the nonlinear solver. Three parameters need to be present in the input file; in order they are $k_{\max}$, F_{tol}, and δ_{tol}, each on a separate line. If the file is present but empty, then the default values are used; they are 35, $1.0\text{E-}5$, and $1.0\text{E-}10$ respectively.

3.4.3 Phase Transition

Given that the governing conservation laws in thermal-hydraulic analyses are those of two-phase flow, phase transition is integral to accurate simulations. The ability to model a simulation as it moves from single-phase flow to multi-phase flow and back is the keystone of the software. However, the methods for addressing phase transitions within two-phase analysis software are ad

hoc procedures that vary between implementations. While every software package addresses this issue out of necessity, there is no agreed upon, systematic, or optimal way of doing so. However, there are certain physical limits that have been identified that need to be considered when dealing with phase transitions [7].

When multi-phase flow within COBRA approaches that of single-phase flow, there is a lower limit imposed upon the volume fraction of the phase that is disappearing. This means that the software does not transition between the governing equations for two-phase flow and those for single-phase flow when a phase depletes. Even during simulated single-phase flow, the non-dominant phase is still present, but it is at a small volume fraction. However, since the non-condensable gases and the vapor fields share a common volume fraction, the partial pressure of the non-condensable gases is allowed to go to zero. Since COBRA incorporates two liquid fields, the lower limit for the aggregate liquid volume fraction, $\alpha_e + \alpha_l$, is divided equally between the two liquid fields. As a phase approaches depletion, $\alpha_\phi \rightarrow \alpha_{\phi,\text{min}}$, or starts to appear, several physical limits are imposed upon the depleted phase.

First, the velocity of the depleting phase is required to approach that of the carrier phase. The following list shows how the velocities are required to behave as their respective volume fractions approach zero.

- $\alpha_e \rightarrow 0 : u_e \rightarrow u_g$
- $\alpha_g \rightarrow 0 : u_g \rightarrow u_l$
- $\alpha_l \rightarrow 0 : u_l \rightarrow u_g$

These velocity equilibrium constraints are imposed by artificially increasing the interfacial drag between the two phases to force equilibrium between the new-time velocities. However, there is a subtle issue created by this increase in the interfacial drag. If the new-time velocities are not equal, then the increased interfacial drag will dominate the residual for the momentum equation of the phase that is not disappearing. For example, Table 3.3 contains the magnitudes of the various terms in the gaseous momentum equation from a nominally single-phase gaseous flow simulation. The term for the interfacial shear between the gaseous phase and the entrained liquid field, $\tau_{g,e}$, is an order of magnitude greater than the other terms.

Δm_g	$\langle\alpha_g\rangle_a^n \nabla P$	$\tau_{g,l}$	$\tau_{g,e}$	τ_w	$\nabla\cdot\left(\alpha_g\rho_g u_g u_g\right)$	$g\langle\alpha_g\rho_g\rangle_a^n$
$3.76E{-}2$	0.12	$-2.72E{-}5$	-3.22	$-7.23E{-}3$	$-4.22E{-}2$	-0.11

Table 3.3 Relative magnitude of terms in a gaseous momentum equation.

This is due to the combination of the increased interfacial drag coefficient and the poor definitions of velocities in COBRA. In a given timestep the artificially large interfacial drag coefficient will force the new-time velocities to be equal. During the next timestep, however, the old-time velocities are not only no longer equal to each other, but also no longer equal to the new-time velocities calculated at the end of the previous timestep.

$$u_{\phi,i}^n = \frac{\dot{m}_{\phi,i}^n}{A_{p,i}\langle\alpha_\phi\rho_\phi\rangle_{\mathrm{a},i}^n} \tag{3.52}$$

By comparing the definition of the old-time velocity, (3.52), with the definition of the new-time velocity, (2.36), the discrepancy between the two definitions is evident, as illustrated in (3.53).

$$\underbrace{\frac{\dot{m}_{\phi,i}^{n+1}}{A_{p,i}\langle\alpha_\phi\rho_\phi\rangle_{\mathrm{a},i}^n}}_{\text{new-time: } t^n\rightarrow t^{n+1}} \neq \underbrace{\frac{\dot{m}_{\phi,i}^{n+1}}{A_{p,i}\langle\alpha_\phi\rho_\phi\rangle_{\mathrm{a},i}^{n+1}}}_{\text{old-time: } t^{n+1}\rightarrow t^{n+2}} \tag{3.53}$$

It is the use of the momenta as the independent parameters for the conservation of momentum equations in conjunction with the use of semi-implicit method that causes the velocities to be poorly defined. The discrepancy in the definition of the velocities creates a non-zero relative velocity between the depleted phase and the present phase at the beginning of every timestep. For the linear solver, this was acceptable; however, the iterative nature of the nonlinear solver exacerbated this problem. This created a situation where multiple Newton steps might have been required to drive the new-time relative velocity to zero even though the previous timestep's new-time relative velocity was zero. To remove this problem, if the new-time relative velocity of the previous timestep is below a certain threshold, $u_{r,min}$, then the initial linearization point for the new-time momentum of the depleted phase, $\dot{m}_{\text{dep}}$, is modified so that the new-time relative velocity is zero. The new linearization point is equal to the carrier phase momentum, $\dot{m}_{\text{car}}$, multiplied by the ratio of the depleted phase's to the carrier phase's macroscopic densities, (3.54).

$$\dot{m}_{\text{dep},i}^{n+1} = \dot{m}_{\text{car},i}^{n+1} \frac{\left\langle \alpha_{\text{dep}} \rho_{\text{dep}} \right\rangle_{\text{a},i}^{n+1}}{\left\langle \alpha_{\text{car}} \rho_{\text{car}} \right\rangle_{\text{a},i}^{n+1}} \tag{3.54}$$

The second restriction is placed upon the thermodynamic state of the depleting phase. Below a certain threshold, the thermodynamic state of the depleted phase is set to be equal to the saturation properties associated with the thermodynamic state of the carrier phase. Additionally, this issue applied to the presence or absence of the non-condensable gas field. Within COBRA, only the non-condensable gas field is capable of being fully depleted; the partial pressure of the non-condensable gas field is allowed to go to zero. To determine if the initial linear point for P_n^{n+1} is appropriate, information from the explicit portion of the continuity equations is used to identify situations where the linearization point may be poor.

3.4.4 Timestep Selection and Acceptance

Upon completion of a timestep within COBRA, there are certain constraints that are imposed upon the independent parameters. These constraints are designed to deal with the possibility that the obtained solution may not be an accurate one. After the single Newton step, the updated parameters are evaluated to determine their validity. There are two ways of resolving potentially invalid solutions: parameter limiting and timestep failure. Stated another way, the limiting procedure can either truncate the updated parameter so that it falls within a valid range, or the timestep can be considered a failure. When a predicted volume fraction falls outside of its valid range, it is truncated to obey the constraint of equation (3.55).

$$\alpha_{\phi,\text{min}} \leq \alpha_{\phi} \leq \alpha_{\phi,\text{max}} \tag{3.55}$$

There are also limits placed upon the changes of the thermodynamic parameters within a timestep in COBRA. The constrained thermodynamic parameters and the limits imposed upon them are:

- Change of phasic enthalpy cannot be greater than 45 [$\frac{\text{BTU}}{\text{lb}_\text{m}}$].
- Change in pressure cannot be greater than 20 [psia].
- Change in partial pressure of the non-condensable gas field cannot be greater than 20 [psia].

These three limits are an attempt to mitigate an initial guess that may be outside of the Newton step's radius of convergence. If any of the above limits are exceeded, the timestep is considered a failure and is repeated with a smaller timestep size.

An adaptive timestep selection algorithm determines the Δt at each timestep. This algorithm utilizes the maximum permissible timestep calculated from the material Courant limit, Δt_{crnt}. The Δt for any timestep is given by (3.56).

$$\Delta t^{n \to n+1} = \max \left[\Delta t_{\text{MIN}}, \min \left[1.2 \Delta t^{n-1 \to n}, 0.85 \Delta t_{\text{crnt}}, \Delta t_{\text{MAX}}, \Delta t_{\text{vol}} \right] \right] \tag{3.56}$$

Δt_{crnt} is the most restrictive of Δt_{vol} and the material Courant limit from both axial and transverse flows. Δt_{vol} is the most restrictive timestep based upon an estimation of the time required to remove all of the mass from a given continuity volume. The most restrictive timestep is used as $\Delta t^{n \to n+1}$.

Chapter 4

Domain Decomposition

As discussed in Section 2.5, domain coupling in thermal-hydraulic, safety-analysis software is an established method to allow different physics to be represented in different domains. This research extends the domain-coupling framework to allow different domains to be consistently coupled while subject to different mathematical treatments. The following chapter describes the mathematical formulation of this novel domain decomposition algorithm, several architectural changes that were made to COBRA to allow for the algorithm to be implemented properly, and a description of its implementation in the software.

4.1 Mathematical Formulation

The governing partial differential equations described in Section 2.2.2 contain nonlinearities that are distributed in both space and time. In the absence of nonlinear physics, a single Newton step is adequate to accurately solve the governing set of discrete algebraic equations. However, if the number of Newton steps is limited to one for a given spatial mesh when nonlinearities are present, then the only way to resolve those nonlinearities is to refine the temporal discretization. If the number of Newton steps is not limited to one, then the nonlinearities may be resolved for a fixed temporal discretization. When the nonlinearities are isolated to a given spatial portion of the domain, the additional Newton steps do not improve the solution in the remainder of the domain. The ability to resolve the nonlinear physics only where they occur could potentially provide a way to reduce the computational cost associated with resolving the nonlinear error in a problem at a given timestep.

With the selective nonlinear refinement algorithm, the domain is split into two segments, a linear domain and a nonlinear domain. There is no requirement that these two domains be contiguous. Each continuity volume and momentum flow path is characterized as either existing in the linear domain or the nonlinear domain. Each momentum flow path is associated with a single domain; the momentum equations are not modified, regardless of the domain in which they exist. The coupling between the two domains occurs at the continuity volume's boundaries connecting the linear domain to the nonlinear domain. Those continuity volumes that are in the linear domain, but have flow paths connecting them to the nonlinear domain, are considered nonlinear boundary continuity (NBC) volumes. All other continuity volumes have associated with them the nonlinear continuity equations, (3.9) – (3.14), regardless of the domain in which they exist. The NBC volumes have a modified set of conservation equations that will now be discussed.

The starting place for the formulation of the NBC volumes' continuity equations will be (3.9) – (3.14). In these normal nonlinear continuity equations the advection terms are formulated in terms of new-time velocities and donored quantities evaluated using the new-time velocities. However, in an NBC volume, the flow paths that connect linear continuity volumes to nonlinear continuity volumes are formulated in terms of phasic mass and energy flow rates. This formulation introduces six new unknowns per domain connection, the flow rates. The index i represents the spatial coordinate of the flow rate. The definition of the six unknown flow rates for a given location is shown in (4.1).

$$\boldsymbol{\Psi}_{i}^{n+1,k} = \begin{bmatrix} \Psi_{m,n} \\ \Psi_{m,l} \\ \Psi_{h,g} \\ \Psi_{h,l} \\ \Psi_{m,e} \\ \Psi_{m,v} \end{bmatrix}_{i}^{n+1,k} = \begin{bmatrix} \frac{\langle \alpha_g^n \rho_n^n \rangle_d^{n+1,k}}{\langle \alpha_g \rho_g \rangle_a^n} \dot{m}_g^{n+1,k} \\ \frac{\langle \alpha_l^n \rho_l^n \rangle_d^{n+1,k}}{\langle \alpha_l \rho_l \rangle_a^n} \dot{m}_l^{n+1,k} \\ \frac{\langle \alpha_g^n \rho_g^n h_g^n \rangle_d^{n+1,k}}{\langle \alpha_g \rho_g \rangle_a^n} \dot{m}_g^{n+1,k} \\ \frac{\langle \alpha_l^n \rho_l^n h_l^n \rangle_d^{n+1,k}}{\langle \alpha_l \rho_l \rangle_a^n} \dot{m}_l^{n+1,k} + \frac{\langle \alpha_e^n \rho_l^n h_l^n \rangle_d^{n+1,k}}{\langle \alpha_e \rho_l \rangle_a^n} \dot{m}_e^{n+1,k} \\ \frac{\langle \alpha_e^n \rho_l^n \rangle_d^{n+1,k}}{\langle \alpha_e \rho_l \rangle_a^n} \dot{m}_e^{n+1,k} \\ \frac{\langle \alpha_g^n \rho_v^n \rangle_d^{n+1,k}}{\langle \alpha_g \rho_g \rangle_a^n} \dot{m}_g^{n+1,k} \end{bmatrix}_{i} \tag{4.1}$$

Using (3.34) and (3.4), the flow rates, (4.1), can be expressed as (4.2).

$$\mathbf{\Psi}_{\ i}^{n+1,k} = \mathbf{\Xi}_{\ i}^{n+1,k} \cdot \dot{\mathbf{m}}_{\ i}^{n+1,k} \tag{4.2}$$

The corresponding continuity equations for a given NBC volume in residual form are given by (4.3) – (4.8). In these equations, the set of flow paths connecting an NBC volume to other continuity volumes within the linear domain is given by N_f. The set of inter-domain flow paths connecting an NBC volume to the nonlinear domain is given by N_{NBC}.

$$\begin{aligned}
F_{m,n}^{k} &= V_c\left[(\alpha_g\rho_n)^{n+1,k} - (\alpha_g\rho_n)^{n}\right] + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha_g^n\rho_n^n\rangle_d^{n+1,k} u_g^{n+1,k} A_p\right)_i \\
&\quad + \Delta t \sum_{i\,\in\, N_{\text{NBC}}}\left(\Psi_{m,n}^{n+1,k}\right)_i \qquad (4.3)\\
F_{m,l}^{k} &= V_c\left(\alpha_l\rho_l\right)^{n+1,k} - V_c\left(\alpha_l\rho_l\right)^{n} + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha_l^n\rho_l^n\rangle_d^{n+1,k} u_l^{n+1,k} A_p\right)_i \\
&\quad + \left[(1-\eta)\Gamma + \Upsilon\right]^{n+1,k} + \Delta t \sum_{i\,\in\, N_{\text{NBC}}}\left(\Psi_{m,l}^{n+1,k}\right)_i \qquad (4.4)\\
F_{h,g}^{k} &= V_c\left[\left(\alpha_g\rho_g h_g\right)^{n+1,k} - \left(\alpha_g\rho_g h_g\right)^{n} - \alpha_g^n(P^{\,n+1,k} - P^{\,n})\right] - \Delta t q_{wg}^n \\
&\quad - \Delta t\left[q_{i,v} + \dot{\Gamma} h_v^{'} + q_{gl}\right]^{n+1,k} + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha_g^n\rho_g^n h_g^n\rangle_d^{n+1,k} u_g^{n+1,k} A_p\right)_i \\
&\quad + \Delta t \sum_{i\,\in\, N_{\text{NBC}}}\left(\Psi_{h,g}^{n+1,k}\right)_i \qquad (4.5)\\
F_{h,l}^{k} &= V_c\left[(\alpha_l\rho_l h_l)^{n+1,k} - (\alpha_l\rho_l h_l)^{n} - \alpha_l^n(P^{\,n+1,k} - P^{\,n})\right] - \Delta t\left[q_{i,l} - \dot{\Gamma} h_l^{'} - q_{gl}\right]^{n+1,k} \\
&\quad + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha_l^n\rho_l^n h_l^n\rangle_d^{n+1,k} u_l^{n+1,k} A_p + \langle\alpha_e^n\rho_l^n h_l^n\rangle_d^{n+1,k} u_e^{n+1,k} A_p\right)_i \\
&\quad + \Delta t \sum_{i\,\in\, N_{\text{NBC}}}\left(\Psi_{h,l}^{n+1,k}\right)_i \qquad (4.6)\\
F_{m,e}^{k} &= V_c\left(\alpha_e\rho_l\right)^{n+1,k} - V_c\left(\alpha_e\rho_l\right)^{n} + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha_e^n\rho_l^n\rangle_d^{n+1,k} u_e^{n+1,k} A_p\right)_i \\
&\quad - \left[\Upsilon - \eta\Gamma\right]^{n+1,k} + \Delta t \sum_{i\,\in\, N_{\text{NBC}}}\left(\Psi_{m,e}^{n+1,k}\right)_i \qquad (4.7)\\
F_{m,v}^{k} &= V_c\left[\left(\alpha_g\rho_v\right)^{n+1,k} - \left(\alpha_g\rho_v\right)^{n}\right] + \Delta t \sum_{i\,\in\, N_f}\left(\langle\alpha_g^n\rho_v^n\rangle_d^{n+1,k} u_g^{n+1,k} A_p\right)_i - \Gamma^{n+1,k}
\end{aligned}$$

$$+ \Delta t \sum_{i \in N_{\text{NBC}}} \left(\Psi_{m,v}^{n+1,k} \right)_i \tag{4.8}$$

By treating the boundary flow rates as independent parameters, the linearized system given in (3.33) is modified to be (4.9).

$$\frac{\partial \mathbf{F}_{\ c}^{k}}{\partial(\alpha_g P_n)} \delta(\alpha_g P_n)^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial \alpha_g} \delta \alpha_g^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial(\alpha_g h_v)} \delta(\alpha_g h_v)^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial((1-\alpha_g) h_l)} \delta((1-\alpha_g) h_l)^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial \alpha_e} \delta \alpha_e^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial P} \delta P^k + \sum_{i \in N_f} \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial \dot{\mathbf{m}}_{\ i}} \delta \dot{\mathbf{m}}_{\ i}^{k} + \sum_{i \in N_{\text{NBC}}} \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial \mathbf{\Psi}_{\ i}} \delta \mathbf{\Psi}_{\ i}^{k} = -\mathbf{F}_{\ c}^{k} \tag{4.9}$$

Since the continuity equations are linear in the boundary flow rates, the matrix of derivatives of the NBC volumes' continuity equations with respect to a given boundary flow rate vector, $\frac{\partial \mathbf{F}_{\ c}^{k}}{\partial \mathbf{\Psi}_{\ i}}$, is the identity matrix times the timestep size, $\Delta t \mathbf{I}$. For algorithmic convenience, the flow rates in the continuity equations' residuals are kept as unknown parameters when solving the local linear system. However, the residual used for convergence includes this term. Utilizing (3.34) and (3.26), the linear system for the NBC volume, (4.9), can be expressed as (4.10) in a manner analogous to that outlined in Section 3.2.

$$\begin{aligned} &\frac{\partial \mathbf{F}_{\ c}^{k}}{\partial(\alpha_g P_n)} \delta(\alpha_g P_n)^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial \alpha_g} \delta \alpha_g^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial(\alpha_g h_v)} \delta(\alpha_g h_v)^k + \frac{\partial \mathbf{F}_{\ c}^{k}}{\partial((1-\alpha_g) h_l)} \delta((1-\alpha_g) h_l)^k + \\ &\frac{\partial \mathbf{F}_{\ c}^{k}}{\partial \alpha_e} \delta \alpha_e^k + \left(\frac{\partial \mathbf{F}_{\ c}^{k}}{\partial P} + \Delta t \sum_{i \in N_f} \mathbf{\Xi}_{\ i}^{k} \frac{\partial \dot{\mathbf{m}}_{\ i}^{k}}{\partial P} \right) \delta P^k + \Delta t \sum_{i \in N_f} \mathbf{\Xi}_{\ i}^{k} \frac{\partial \dot{\mathbf{m}}_{\ i}^{k}}{\partial P_{o(i)}} \delta P_{o(i)}^k + \\ &\Delta t \sum_{i \in N_{\text{NBC}}} \delta \mathbf{\Psi}_{\ i}^{k} = - \left(\mathbf{F}_{\ c}^{k} + \Delta t \sum_{i \in N_f} \mathbf{\Xi}_{\ i}^{k} \delta \dot{\mathbf{m}}_{\ i}^{*} - \Delta t \sum_{i \in N_{\text{NBC}}} \mathbf{\Psi}_{\ i}^{k} \right) - \Delta t \sum_{i \in N_{\text{NBC}}} \mathbf{\Psi}_{\ i}^{k} \end{aligned} \tag{4.10}$$

As outlined in Section 3.2, the advection terms in the residuals of (4.10) are evaluated using $\dot{\mathbf{m}}^{\,n+1,k+\frac{1}{2}}$. The first six columns of (4.10) will be represented by $\mathbf{J}_{\,n}$. The next N_f inter-continuity volume coupling columns multiplying their respective pressure changes, $\delta P_{o(i)}$, will be represented by $\mathbf{K}_{\,c}$. The final $6 * N_{\text{NBC}}$ columns representing the inter-domain coupling coefficients, which are multiplied by the change in flow rates, $\delta \mathbf{\Psi}_{\,i}$, will be collectively referred to as $\mathbf{Q}_{\,c}$. This matrix, $\mathbf{Q}_{\,c}$,

is also present with the opposite sign on the right-hand side of (4.10). Using this matrix notation, (4.10) can be represented as (4.11).

$$\left[\mathbf{J}_c|\mathbf{K}_c|\mathbf{Q}_c\right]\delta\mathbf{C}_c = \mathbf{r}_c - \mathbf{Q}_c\mathbf{\Psi}_c \tag{4.11}$$

The vector of unknowns, $\mathbf{C}_c$, is defined in (4.12).

$$\delta\mathbf{C}_c \equiv \begin{bmatrix} \delta(\alpha_g P_n) \\ \delta\alpha_g \\ \delta(\alpha_g h_v) \\ \delta((1-\alpha_g)h_l) \\ \delta\alpha_e \\ \delta P \\ \delta P_{o(1)} \\ \vdots \\ \delta P_{o(N_f)} \\ \delta\mathbf{\Psi}_1 \\ \vdots \\ \delta\mathbf{\Psi}_{N_{\text{NBC}}} \end{bmatrix} = \begin{bmatrix} (\alpha_g P_n)^{n+1} - (\alpha_g P_g)^n \\ \alpha_g^{n+1} - \alpha_g^n \\ (\alpha_g h_v)^{n+1} - (\alpha_g h_v)^n \\ ((1-\alpha_g)h_l)^{n+1} - ((1-\alpha_g)h_l)^n \\ \alpha_{e,j}^{n+1} - \alpha_e^n \\ P^{n+1} - P^n \\ P_{o(1)}^{n+1} - P_{o(1)}^n \\ \vdots \\ P_{o(N_f)}^{n+1} - P_{o(N_f)}^n \\ \mathbf{\Psi}_1^{n+1} - \mathbf{\Psi}_1^n \\ \vdots \\ \mathbf{\Psi}_{N_{\text{NBC}}}^{n+1} - \mathbf{\Psi}_{N_{\text{NBC}}}^n \end{bmatrix} \tag{4.12}$$

(4.11) is then subjected to partial $\mathbf{LU}$ decomposition without pivoting, producing (4.13).

$$\left[\mathbf{U}_c|\mathbf{L}_c^{-1}\mathbf{K}_c|\mathbf{L}_c^{-1}\mathbf{Q}_c\right]\delta\mathbf{C}_c = \mathbf{L}_c^{-1}\mathbf{r}_c - \mathbf{L}_c^{-1}\mathbf{Q}_c\mathbf{\Psi}_c \tag{4.13}$$

However, since the matrix product $\mathbf{L}_c^{-1}\mathbf{Q}_c$ occurs twice in (4.13), only a single instance is included in the solution of this system. The $\mathbf{L}_c^{-1}\mathbf{Q}_c$ is omitted from the right-hand side of (4.11) during the $\mathbf{LU}$ decomposition. This $\mathbf{LU}$ decomposition is such that the lower triangular matrix has ones along the diagonal. The N_{NBC} identity matrices that comprise $\mathbf{Q}_c$ are now themselves the inverse of the lower triangular matrix multiplied by Δt . The final row of the linear system in (4.13) is then scaled by the sixth row's sixth column entry, $\mathbf{U}_c[6,6]$.

The linear pressure matrix, $\mathbf{A}_{\text{lin}}$, is assembled from the sixth row of each continuity volume's linear system in the linear domain including the NBC volumes. There will be N_{lin} rows in the system, which correspond to each of the linear pressure updates. Let N_n represent the total number of interfaces between the linear and the nonlinear domains. Since there can be multiple interfaces from a single NBC volume to the nonlinear domain, N_n is greater than or equal to the number of NBC volumes. There will be an additional $6 * N_n$ right-hand sides for the linear pressure matrix that correspond to the coefficient matrices, $\mathbf{L}_c^{-1}\mathbf{Q}_c$, for the unknown flow rates in each of the NBC volumes, represented as $\mathbf{B}_{\text{lin}}$. The portion the pressure system that is from $\mathbf{L}_c^{-1}\mathbf{r}_c$ will be referred to as $\mathbf{res}_{\text{lin}}$. The resulting system is shown in (4.14).

$$\mathbf{A}_{\text{lin}}\delta\mathbf{P}_{\text{lin}} + \mathbf{B}_{\text{lin}}\delta\mathbf{\Psi}_{\text{lin}} = \mathbf{res}_{\text{lin}} - \mathbf{B}_{\text{lin}}\mathbf{\Psi}_{\text{lin}} \tag{4.14}$$

The linear system in (4.14) is then multiplied by $\mathbf{A}_{\text{lin}}^{-1}$, resulting in (4.15).

$$\begin{aligned}\delta\mathbf{P}_{\text{lin}} + \mathbf{A}_{\text{lin}}^{-1}\mathbf{B}_{\text{lin}}\delta\mathbf{\Psi}_{\text{lin}} &= \mathbf{A}_{\text{lin}}^{-1}\mathbf{res}_{\text{lin}} - \mathbf{A}_{\text{lin}}^{-1}\mathbf{B}_{\text{lin}}\mathbf{\Psi}_{\text{lin}} \\ \delta\mathbf{P}_{\text{lin}} + \mathbf{W}_{\text{lin}}\delta\mathbf{\Psi}_{\text{lin}} &= \delta\mathbf{P}_{\text{lin}}^{*} - \mathbf{W}_{\text{lin}}\mathbf{\Psi}_{\text{lin}}\end{aligned} \tag{4.15}$$

Once (4.14) has been inverted, the resulting pressure updates, $\delta\mathbf{P}_{\text{lin}}$, for the linear domain are functions of all of the unknown flow rates between the two domains and their updates. The constant portion of (4.15) is shown as $\delta\mathbf{P}_{\text{lin}}^{*}$, representing the portion of the change in $\delta\mathbf{P}_{\text{lin}}$ that comes from the linear domain. The matrix, $\mathbf{W}_{\text{lin}}$, represents the matrix of all inter-domain pressure coupling coefficients. A representative row of (4.15) is shown in (4.16). The vectors $\mathbf{w}_{j,i}$ represent the coupling coefficients for each NBC volume's pressure update.

$$\delta P_j + \sum_{i \in N_n} \mathbf{w}_{j,i}^T \cdot \delta\mathbf{\Psi}_i = \delta P_j^* - \sum_{i \in N_n} \mathbf{w}_{j,i}^T \cdot \mathbf{\Psi}_i^k \tag{4.16}$$

In the linear domain a single Newton step is taken to be the change from the old-time solution to the new-time solution, not an iterate. This approach means that (4.16) is interpreted not as an iterative solution but as a change from old-time to new-time values. As a result, the pressure update

is taken to be $\delta P_j = P_j^{n+1} - P_j^n$, the flow rates on the right-hand side are evaluated as $\mathbf{\Psi}^{\,n}$, and the flow rate updates are defined as $\delta\mathbf{\Psi} = \mathbf{\Psi}^{\,n+1} - \mathbf{\Psi}^{\,n}$.

Once the linear system has been inverted, the nonlinear pressure matrix is finalized. Recall that the nonlinear continuity volumes' governing equations are not modified by the domain decomposition. The nonlinear domain is composed of N_{nln} continuity volumes. Additionally, for each NBC volume there will be an extra row in the nonlinear pressure matrix, $\mathbf{A}_{\text{nln}}$, that corresponds to (4.16). However, within the nonlinear domain, the flow rates are not kept as unknowns; instead, the flow rates in (4.16) are expressed in terms of unknown pressures by using (3.26), (4.2), and (4.17).

$$\delta\mathbf{\Psi} = \mathbf{\Xi}^{\,k} \cdot \delta\dot{\mathbf{m}} \tag{4.17}$$

The resulting equations for the pressure updates of the NBC volumes as viewed by the nonlinear domain are given by (4.18)

$$\begin{aligned}
\delta P_j + \sum_{i \in N_n} \mathbf{w}_{j,i}^{\,T} \mathbf{\Xi}_i^{\,k} \delta\mathbf{m}_i^{\,k} &= \delta P_j^* - \sum_{i \in N_n} \mathbf{w}_{j,i}^{\,T} \mathbf{\Xi}_i^{\,k} \dot{\mathbf{m}}_i^{\,k} \\
\delta P_j + \sum_{i \in N_n} \mathbf{w}_{j,i}^{\,T} \left[\mathbf{\Xi}_i^{\,k} \frac{\partial \dot{\mathbf{m}}_i^{\,k}}{\partial P_{s(i)}} \delta P_{s(i)} + \mathbf{\Xi}_i^{\,k} \frac{\partial \dot{\mathbf{m}}_i^{\,k}}{\partial P_{o(i)}} \delta P_{o(i)} \right] &= \delta P_j^* - \sum_{i \in N_n} \mathbf{w}_{j,i}^{\,T} \left[\mathbf{\Xi}_i^{\,k} \dot{\mathbf{m}}_i^{\,k} + \mathbf{\Xi}_i^{\,k} \delta\dot{\mathbf{m}}_i^{\,*} \right] \\
\delta P_j + \sum_{i \in N_n} \mathbf{w}_{j,i}^{\,T} \left[\mathbf{\Xi}_i^{\,k} \frac{\partial \dot{\mathbf{m}}_i^{\,k}}{\partial P_{s(i)}} \delta P_{s(i)} + \mathbf{\Xi}_i^{\,k} \frac{\partial \dot{\mathbf{m}}_i^{\,k}}{\partial P_{o(i)}} \delta P_{o(i)} \right] &= \delta P_j^* - \sum_{i \in N_n} \mathbf{w}_{j,i}^{\,T} \mathbf{\Xi}_i^{\,k} \dot{\mathbf{m}}_i^{\,k+\frac{1}{2}}
\end{aligned} \tag{4.18}$$

The flux terms on the right-hand side of (4.18) are evaluated using $\dot{\mathbf{m}}^{\,n+1,k+\frac{1}{2}}$. The subscript $s(i)$ represents the NBC volume to which the flow path i is connected. The subscript $o(i)$ represents the nonlinear continuity volume to which the flow path i is connected. (4.18) shows that every NBC volume has a functional dependence upon the change in pressure of not only every other NBC volume but also every nonlinear continuity volume that is connected to an NBC volume. The final nonlinear pressure matrix will have $N_{\text{nln}} + N_n$ equations and unknowns and is represented by (4.19).

$$\mathbf{A}_{\text{nln}}^{\,k} \delta\mathbf{P}_{\text{nln}}^{\,k} = \mathbf{res}_{\text{nln}}^{\,k} \tag{4.19}$$

This system is then solved to obtain the pressure updates for the nonlinear domain. The continuity volumes and flow paths in the nonlinear domain are updated using $\delta \mathbf{P}^{\,k}_{\text{nln}}$. The NBC volumes' pressures are not updated because the old iterate, k, of a given NBC volume is always the old-time value to maintain consistency with the linear domain. The nonlinear domain is subjected to multiple Newton iterates as outlined in Section 3.2. A deviation from the convergence procedure for a single Newton domain is that the norms used for convergence determination are formulated using the residuals and the updates from only that portion of the domain that is subject to the nonlinear solver. These two vectors, the residual and the update, do not include terms from the NBC volumes.

Upon termination of the Newton iterations for the nonlinear domain, all of the inter-domain flow rates are now known. These flow rates are used to obtain the linear domain's pressure updates. The linear domain's pressure updates, δP_j, have their functional dependence resolved by using (4.20).

$$\delta P_j = \delta P_j^* - \sum_{i \,\in\, N_n} \mathbf{w}^{\,T}_{j,i} \boldsymbol{\Xi}^{\,k}_{i} \left[\dot{\mathbf{m}}^{\,k}_{i} + \delta \dot{\mathbf{m}}_{\,i} \right] \tag{4.20}$$

Once the pressure update vector for the linear domain is obtained, all of linear domain's continuity volumes and flow paths are looped over. During this loop, the new-time linear variables are calculated. The completion of the linear update marks the completion of a single timestep.

4.2 Implementation in COBRA

In this section, several architectural changes that were necessary to integrate the domain decomposition algorithm from Section 4.1 into the COBRA software are addressed. First, the modification of the data structures for the pressure matrices and their solvers is discussed. Second, the object-oriented, volume data structures for domain decomposition are detailed. Next, the implementation of domain decomposition algorithm will be shown in detail. Lastly, the dual domain input file is described.

4.2.1 Pressure Matrix and Solver Data Structures

As seen in Section 3.1, the largest linear system that is solved during a Newton step is the pressure matrix that is used to obtain the pressure update vector. COBRA provides the user with three options for the linear algebra method to be used when solving for the pressure updates: the Gauss elimination routine, the SuperLU solver [26], and the Pardiso solver [36, 37]. The SuperLU and the Pardiso solvers utilize sparse matrix storage and the Gauss elimination routine utilizes full matrix storage. The sparsity of the matrices typically encountered is such that use of the Gauss elimination routine is discouraged.

The method that the user selects via the COBRA input file, "`deck.inp`," determines the data storage structure of the pressure matrix. Each of the three methods utilizes a different memory format for matrix storage. Given that the software as obtained was based upon procedural programming practices and written in a mixed Fortran dialect (FORTRAN 77, Fortran 90, and Fortran 95), the static memory structures for each of the three possible pressure matrices were predefined in different modules. However, this procedural paradigm precluded the existence of two separate pressure matrices, one for the linear domain and one for the nonlinear domain. This procedural approach for the pressure matrices was deemed inappropriate for the work being done, and an object-based approach was introduced instead. To implement this switch, the way in which the solver was stored and accessed was modified to both enable multiple solvers and reduce the logical complexity of accessing the proper matrix.

The object-oriented features in the Fortran 2003 and Fortran 2008 standards provided an alternative method of handling the pressure matrix. The matrix storage structure was first replaced with an abstract class, `matrix`. Figure 4.1 shows the class diagram for the `matrix` hierarchy. The `matrix` class contains the number of right-hand sides anticipated, `nrhs`, as well as the total number of pressure updates for a given domain, `n`. Additionally, this class would provide interfaces for the following procedures:

- `put`: a procedure that sets $\mathbf{A}_{i,j}$ to a given value.
- `get`: a procedure that returns $\mathbf{A}_{i,j}$.
- `rescale`: a procedure for scaling $\mathbf{A}_{i,:}$ by a given value.

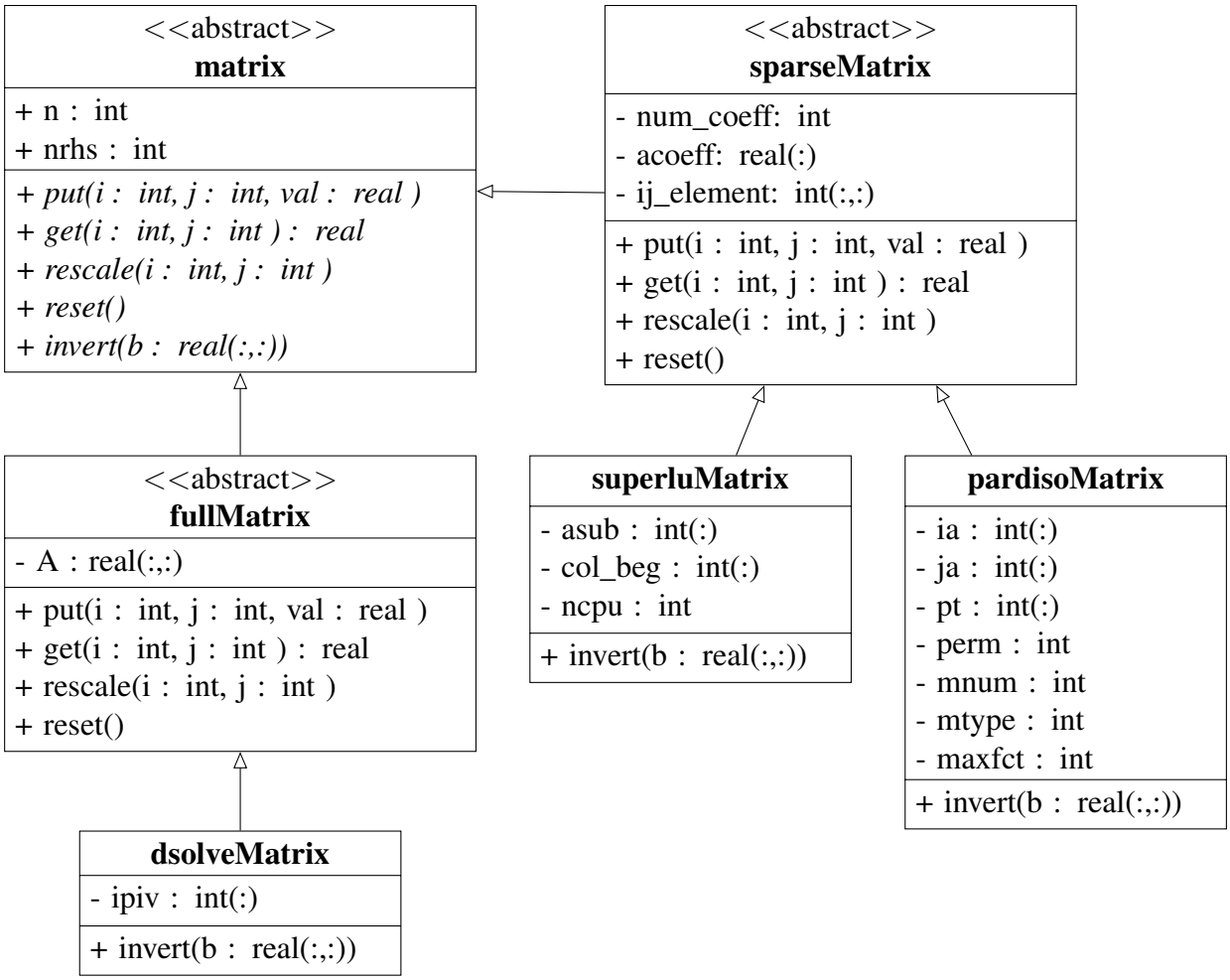

Figure 4.1 Matrix Class Diagram

- `reset`: a procedure for setting $\mathbf{A} = \mathbf{0}$.
- `invert`: a procedure that returns $\mathbf{x}$ from $\mathbf{A}\,\mathbf{x} = \mathbf{b}$.

Given that the SuperLU and the Pardiso solvers utilize sparse matrices and the Gauss elimination routine uses a full matrix, there are two further abstract classes that inherit from `matrix`; they are `fullMatrix` and `sparseMatrix`. The `fullMatrix` class contains an array representing $\mathbf{A}$. The `sparseMatrix` class contains the information required to construct a sparse matrix; however, the exact storage format is not specified in `sparseMatrix`. Both `fullMatrix` and `sparseMatrix` contain procedural implementations of the interfaces defined in `matrix`. There are three concrete classes that correspond to the three supported solvers: `dsolveMatrix`, `superluMatrix`,

and pardisoMatrix; all three implement their own linear system solving routines under the invert interface. dsolveMatrix is a concrete class inheriting from fullMatrix that includes pivoting information. superluMatrix and pardisoMatrix are concrete classes inheriting from sparseMatrix that implement the matrix storage schemes for their respective solvers.

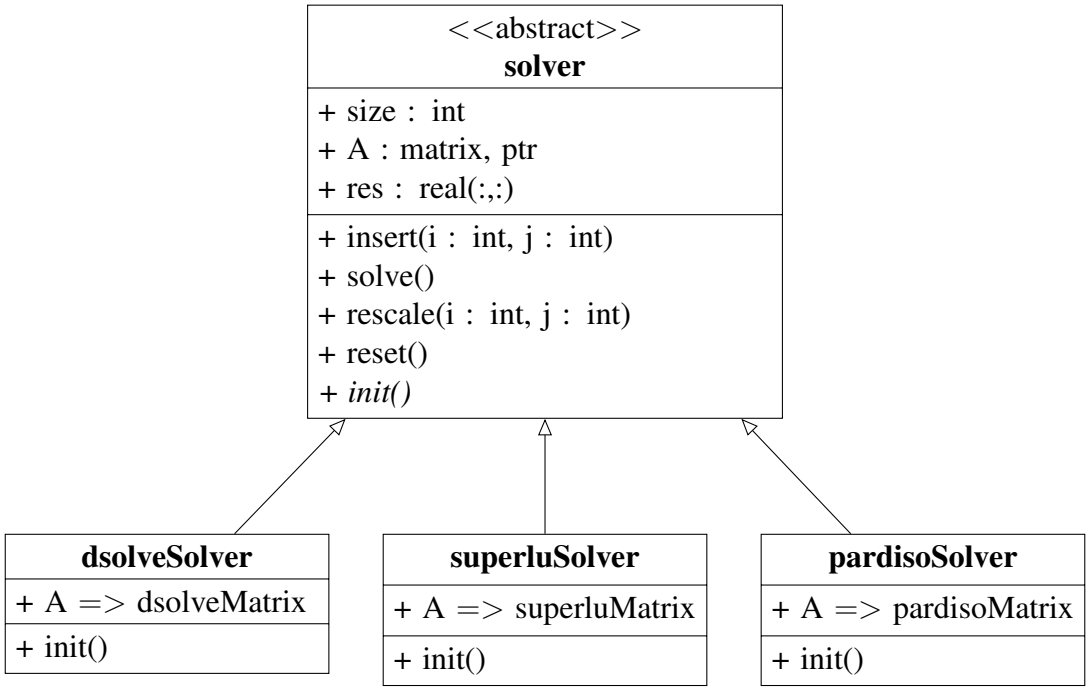

Figure 4.2 Solver Class Diagram

Once the matrix hierarchy was developed, an abstract solver class was designed. Figure 4.2 shows the class diagram for the solver hierarchy. The solver class contains the size of the system, size; a pointer to a matrix object, A; and the right-hand side of the linear system, res. Within the solver class there is an interface for the initialization routine, init. Additionally, there is a procedure named insert that will take the sixth row of a continuity volume's linear system and properly insert it into the correct row of the domain's pressure matrix, $\mathbf{A}_{j,:}$, as outlined in Section 4.1. The following procedures for manipulating the linear system are also defined within the solver object:

- solve: an interface to call the solver associated with the matrix.
- rescale: a procedure for scaling both $\mathbf{A}_{i,:}$ and $\mathbf{res}_i$.

- `reset`: a procedure for setting both $\mathbf{A} = \mathbf{0}$ and $\mathbf{res} = \mathbf{0}$.

There are three concrete classes that inherit from the abstract solver class: `dsolveSolver`, `superluSolver`, and `pardisoSolver`. Each of these concrete classes specifies an initialization routine that takes an adjacency list data structure, as discussed in Section 3.2.1, and instantiates the `matrix` pointer to the appropriate concrete subtype, and sets the appropriate parameters. By abstracting the matrix storage strategies and the solvers in this way, a consistent interface is provided to the rest of the software.

4.2.2 Volume Data Structures

A difficulty that had to be overcome during this work was the proper mapping of the continuity volumes' memory indices to the proper pressure matrix indices. The memory storage format for many variables was based upon a universal ordinal system for the continuity volumes. This ordinal system was based upon the premise that there was a single pressure matrix. It was determined that the programmatic complexity of trying to maintain a single ordinal system for the dual domains was greater than that of redesigning the memory architecture. As part of the redesign, the software was partially transitioned to an object-based mesh to allow for multiple pressure matrices with different ordinal systems to co-exist.

A continuity volume was chosen as the basic unit for mesh representation. Given that the mesh is decomposed into dual domains, there were two types of continuity volumes that needed to be distinguished. The first type of volume, `baseVolume`, encompasses all continuity volumes in the nonlinear domain and those continuity volumes in the linear domain that do not have a flow path connecting them to the nonlinear domain. The second type of volume, `nbcVolume`, is an extension of the `baseVolume` type. The set of `nbcVolumes` includes those continuity volumes in the linear domain that do have a flow path connecting them to the nonlinear domain, the NBC volumes. The volume class diagram for COBRA is outlined in Figure 4.3.

The `baseVolume` class contains information for the most basic type of continuity volumes. Each `baseVolume` has a pointer to the `solver` object associated with its domain. A `baseVolume` that corresponds to a continuity volume in the linear domain has a `solver` pointer to the linear

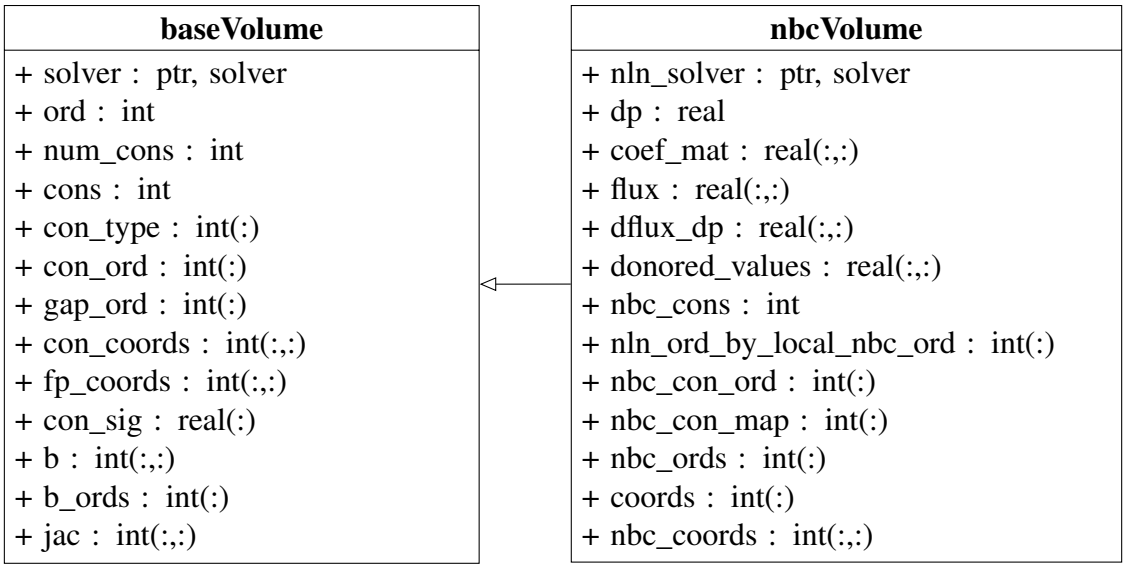

Figure 4.3 Volume Class Diagram

pressure matrix solver. The nonlinear domain's continuity volumes have a `solver` pointer to the nonlinear pressure matrix solver. This allows each continuity volume to be able to access the correct pressure matrix without having to reference a logical map or perform a calculation. This class also stores the volume's ordinal, `ord`. This variable provides indexing information for the volume's pressure matrix.

Additionally, information regarding the connectivity of the volume to other volumes is stored in several variables: `num_cons`, `cons`, `con_type`, `con_ord`, `gap_ord`, `con_coords`, `fp_coords`, and `con_sig`. These variables represent the information necessary to properly reference those flow paths to which the continuity volume is connected, as well as the continuity volumes on the other end of those flow paths. This data structure has the drawback of storing redundant flow path information. Each of the continuity volumes connected by a particular flow path contains all of the information about that flow path in separate memory locations.

The `baseVolume` class also contains volume's linear system from Section 3.1. The volume's linear system is stored as `jac`. The dimensions of `jac` are dictated by the connectivity of the volume and its volume type, both of which are determined during input processing. The right-hand side of the continuity volume's linear system is stored as `b`. Once again, the dimensions of `b` are

dictated by the type of the volume; an `nbcVolume` will have more columns in `jac` and `b` than a comparable `baseVolume`. The variable `b_ords` provides a mapping between the columns of `b` and the columns of the pressure matrix of the domain to which the volume belongs.

Those linear continuity volumes that are connected to the nonlinear domain via a flow path have their own `nbcVolume` class. This class, as shown in Figure 4.3, is a derived class and contains additional information required to perform the domain decomposition. First, a second `solver` pointer is present, `nln_solver`. This allows for proper indexing into the nonlinear pressure matrix when needed. Second, the `nbcVolume` also contains a separate collection of variables describing those flow paths that connect it to the nonlinear domain: `nbc_cons`, `nbc_con_ord`, `nbc_con_map`, `nbc_ords`, `nbc_coords`, `coords`, and `nln_ord_by_local_nbc_ord`. Lastly, the `nbcVolume` class contains the pressure coefficients, `coef_mat`; the flow rate associated with the boundary, `flux`; the derivative of the flow rate with respect to the pressure on either side of the flow path, `dflux_dp`; the matrix Ξ in compressed storage, `donored_values`; and the right-hand side of the linear domain's pressure solution, `dp`.

Each continuity volume is represented by one of the above described volume classes. This polymorphic approach allows for domain-agnostic procedures and references within the software.

4.2.3 Domain Decomposition Algorithm

The implementation of the domain decomposition's mathematical algorithm provided in Section 4.1 will now be covered. Algorithm 4.1 provides a high-level overview of the software as implemented. The steps in this algorithm will be detailed in the following section.

First, the nonlinear domain decomposition input file described in Section 4.2.4 is parsed to determine to which domain each subchannel belongs. Second, each continuity volume is designated as either a `baseVolume` or an `nbcVolume`. Then the `solver` objects are instantiated according to the COBRA input file. Lastly, after the input processing and state initialization, the temporal integration loop begins.

In a dual domain problem, a single timestep is now broken into two parts: the linear and the nonlinear portions. At the beginning of a timestep, the entire domain is traversed to assemble the

Require: Input Processing
1: **set:** $n = 0$
2: **loop** Transient Loop
3: **set:** $t^{n+1} := t^n + \Delta t$
4: **algorithm:** Assemble Nonlinear and Linear Pressure Matrices ▷ Algorithm 3.2
5: **if** `LinearSolver` **then solve** $\mathbf{A}_{\text{lin}}\delta\mathbf{P}_{\text{lin}} = \mathbf{res}_{\text{lin}}$
6: **if** `NonlinearSolver` **then**
7: **set:** $k = 0$
8: **algorithm:** Get Nonlinear Coefficients ▷ Algorithm 4.2
9: **algorithm:** Set Nonlinear Boundary Values ▷ Algorithm 4.3
10: **solve:** $\mathbf{A}^{k}_{\text{nln}}\delta\mathbf{P}^{k}_{\text{nln}} = \mathbf{res}^{k}_{\text{nln}}$
11: **algorithm:** Update Nonlinear Variables ▷ Algorithm 3.3
12: **loop** Newton Loop
13: **algorithm:** Assemble Nonlinear Pressure Matrix ▷ Algorithm 3.2
14: **algorithm:** Nonlinear Convergence Determination ▷ Algorithm 3.5
15: **if end** Newton loop **then**
16: **break** Newton Loop
17: **end if**
18: **set:** $k \mathrel{+}= 1$
19: **algorithm:** Set Nonlinear Boundary Values ▷ Algorithm 4.3
20: **solve:** $\mathbf{A}^{k}_{\text{nln}}\delta\mathbf{P}^{k}_{\text{nln}} = \mathbf{res}^{k}_{\text{nln}}$
21: **algorithm:** Update Nonlinear Variables ▷ Algorithm 3.3
22: **end loop**
23: **end if**
24: **if** `LinearSolver` **then algorithm** Update Linear Variables ▷ Algorithm 3.3
25: **set:** $n \mathrel{+}= 1$
26: **end loop**

Algorithm 4.1 Dual domain COBRA algorithm.

pressure matrices for both the linear domain and the first iterate of the nonlinear domain. This process is similar to the algorithm outlined in 3.1 for assembling the pressure matrix. However, there are differences when an `nbcVolume` is encountered.

The `nbcVolume` objects are treated slightly different from the regular `baseVolume` objects. The flow rate terms from the equations are not included in the residual at this point, but are instead treated as unknowns. These flow rates are treated as independent parameters, which decreases the number of entries in the $\mathbf{K}_c$ matrix. However, the derivatives of the residuals with respect these flow rates, $\mathbf{Q}_c$, are added to the linear system. If a linear continuity volume is attached through more than one surface to the nonlinear domain, then $\mathbf{Q}_c$ will contain multiple identity

```
1:  for i = 1, N_nln do
2:      set: b ⇒ nbcVolume[i].solver.res(nbcVolume[i].ord,:)
3:      for j = 1, N_nln do
4:          for  k = 1, nbcVolume[j].nbc_cons do
5:              set: g_ord = nbcVolume[j].nbc_con_ord(k)
6:              set: s_ord = 2 + 6 ∗ (g_ord - 1)
7:              set: e_ord = s_ord + 5
8:              set: nbcVolume[i].coef_mat(1:6, g_ord) = b(s_ord:e_ord)
9:          end for
10:     end for
11: end for
```

Algorithm 4.2 Obtain NBC Volume Coefficients.

matrices. The resulting system will be subjected to the same $\mathbf{LU}$ decomposition as outlined for a regular volume. The last equation with the additional right-hand sides will be assembled into the linear domain's pressure matrix. The linear system for the linear domain is then inverted by the chosen solver. The additional coefficients, $\mathbf{W}_{\text{lin}}$, represent the coefficients for the boundary flow rate between the linear and the nonlinear domain. After the inversion of the pressure matrix, these coefficients for the NBC volumes are collected into the `nbcVolume` objects for ease of manipulation later. This is shown in Algorithm 4.2.

Once the pressure coefficients have been collected, and the linear domain has been inverted, the additional entries in the nonlinear pressure matrix that represent the NBC volume's pressure equations are populated. This procedure is shown in detail in Algorithm 4.3. This process is a system of three nested loops. The first, outer, loop is over all the NBC volumes in the domain, N_{nln}. From Section 4.1 and (4.16), each of these outer NBC volumes has an associated pressure equation, (4.18). This equation represents the inter-dependency of all of the NBC volumes upon each other. Evaluating (4.18) requires a second, inner loop over all of the NBC volumes for each of the outer NBC volumes. For each of these inner loop NBC volumes, their N_{NBC} flow paths that connect them to nonlinear domain are then traversed, creating a third and final nested loop. For each connecting flow path, the contribution is calculated and stored in the row of $\mathbf{A}_{\text{nln}}$ associated with the outer NBC volume.

```
1: for i = 1, N_nln do
2:     set: A ⇒ nbcVolume[i].nln_solver.A
3:     set: i_ord = nbcVolume[i].nln_ord
4:     for j = 1, N_nln do
5:         set: j_ord = nbcVolume[j].nln_ord
6:         for  k = 1, nbcVolume[i].nbc_cons do
7:             set: g_ord = nbcVolume[j].nbc_con_ord(k)
8:             set: k_ord = nbcVolume[j].nln_ord_by_local_nbc_ord(k)
9:             set: dp = nbcVolume[i].coef_mat(:, g_ord) · nbcVolume[j].dflux_dp(:, k)
10:            set: A(i_ord, k_ord) += dp
11:            set: A(i_ord, j_ord) −= dp
12:        end for
13:    end for
14:    set: res = nbcVolume[i].solver.res(nbcVolume[i].ord,:) · nbc_flux
15:    set: nbcVolume[i].nln_solver.res(i_ord, 1) = res
16:    set: A(i_ord, i_ord) += 1.0
17:    call: nbcVolume[i].nln_solver.rescale(i_ord)
18: end for
```

Algorithm 4.3 Set NBC Volume Pressure Equations Into Nonlinear Pressure Matrix.

Once the pressure equations for the NBC volumes have been inserted into the nonlinear pressure matrix, this matrix is solved. The resulting pressure update vector is then used to update the nonlinear continuity volumes' variables according to Algorithm 3.3. This inversion and update represents the first iterate for the nonlinear domain. The Newton loop is now entered. This loop is similar to that discussed in Section 3.2, with the addition of a step to set the NBC volumes' equations prior to inverting $\mathbf{A}_{\text{nln}}$. The loop termination criteria for this Newton loop are similar to those outlined in Section 3.2. The difference is that these convergence metrics depend only upon the nonlinear subdomain. Upon termination of the Newton loop, the linear domain's variables are then updated.

4.2.4 Dual Domain Input File

Since the domain decomposition algorithm uses the nonlinear solver, the nonlinear convergence tolerances and the iteration limit are input as described in Section 3.4.2. These parameters only apply to the nonlinear domain. Additional information is appended to the end of the basic nonlinear input file to activate the domain decomposition algorithm. The nonlinear input file needs to have

the IDs of the subchannels that are going to be subjected to additional Newton steps specified. The presence of the character string "'begin-nln-channels'" after the nonlinear input signals that there will be two domains. After that string, the IDs of the subchannels are listed, one per line, in no particular order. Once all subchannels to be in the nonlinear domain have been listed, the character string "'end-nln-channels'" needs to be on a separate line. The following is an example of the dual domain input segment of the "nwt.cob" input file.

```
'begin-nln-channels'
<chanID_1>
<chanID_2>
<chanID_3>
...
<chanID_N>
'end-nln-channels'
```

In the above example, <chanID_N> is the subchannel ID that will be included in the nonlinear domain. The input processing routine has extensive error checking and user feedback for proper formatting of this input. Note that there is no requirement that the subchannels specified to be in the nonlinear domain form a contiguous domain.

Chapter 5

Numerical Results

There were six test problems developed for this work. Two of these problems evaluated the implementation and the efficacy of the nonlinear solver. A third problem with extensive geometric complexity was designed to evaluate the implementation of the domain decomposition algorithm. The fourth problem demonstrated that selective nonlinear refinement could produce the nonlinear solver's solution with less computational effort. The fifth problem showed the impact of spatially isolable nonlinearities upon the solution in the remainder of the problem domain. Finally, the last problem demonstrated the value of using selective nonlinear refinement when simulating a physically relevant situation to reactor safety. This problem showed that use of the domain decomposition algorithm could obtain results that are more reflective of the nonlinear solver, with less computational time. The goal of the tests, the methods used, and the models will be discussed in each section.

5.1 Single-Phase Flow and Flashing Problems

The two problems in this section were developed to evaluate the implementation and the efficacy of the nonlinear solver. The first problem discussed demonstrated that the nonlinear solver reproduces the linear solution in the absence of strong nonlinearities. The second problem illustrated that the use of the nonlinear solver on a problem with strong nonlinearities produces a solution that is more temporally consistent than that produced with the linear solver. Additionally, this second problem demonstrated that temporal convergence using the linear solver might be misleading.

5.1.1 Model

The two models developed for these tests were thought problems with generic input conditions. For both of these problems, the same computational geometry was used; Figure 5.1 represents the model geometry. Each block represents a single continuity volume with a fixed Δx of 4 [in]. Each continuity cell has a cross-sectional area of 4 [in^2]. The solid block at the top of the subchannel represents a boundary cell where the pressure and enthalpy are specified. It represents an infinite reservoir filled with a fluid at a specified thermodynamic state. The solid triangle represents a specified flow at the bottom edge of the first continuity volume.

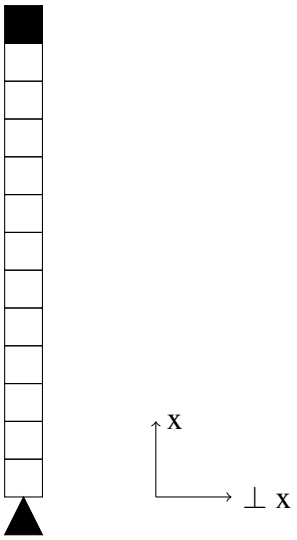

Figure 5.1 Geometry for single-phase and flashing problems.

The two problems, while having the same geometry, are different in their dominant physics. One problem was designed to simulate single-phase, single-field continuous liquid flow in a subchannel. This problem will be referred to as the single-phase problem. The second problem was designed such that high-pressure liquid would flash into steam as it entered a subchannel initially filled with saturated vapor at a much lower pressure, known hereafter as the flashing problem.

Table 5.1 provides the initial conditions for the two problems. The initial pressure, enthalpy, and volume-fractions for the different fields allowed for a complete description of the problem's thermodynamics state. The initial velocities for both problems were set to zero.

	Pressure [psia]	Enthalpy $[\frac{\text{BTU}}{\text{lb}_\text{m}}]$	α_g [-]	α_l [-]	α_e [-]
Single-Phase	200.0	355.5	0.0	1.0	0.0
Flashing	200.0	1198.3	1.0	0.0	0.0

Table 5.1 Initial conditions for the single-phase and flashing problems.

Each of the problems had a specified pressure-enthalpy boundary condition at the outlet and a flow-enthalpy boundary condition at the inlet of the domain. Table 5.2 contains the pressure, enthalpy, and composition of the pressure-enthalpy reservoir.

	Pressure [psia]	Enthalpy $[\frac{\text{BTU}}{\text{lb}_\text{m}}]$	α_g [-]	α_l [-]	α_e [-]
Single-Phase	200.0	355.5	0.0	1.0	0.0
Flashing	200.0	1198.3	1.0	0.0	0.0

Table 5.2 The outlet boundary conditions for the single-phase and flashing problems.

The flow-enthalpy boundary condition described the thermodynamic state of the in-flowing fluid and its flow rate. Table 5.3 describes the inlet boundary conditions for the two problems. The thermodynamic state of the inlet liquid for the flashing problem was unstable. There was an 87.1 $[\frac{\text{BTU}}{\text{lb}_\text{m}}]$ difference between the inlet enthalpy and the saturation enthalpy at 200 [psia]. This difference drove the liquid's phase change into vapor.

	Pressure [psia]	Enthalpy $[\frac{\text{BTU}}{\text{lb}_\text{m}}]$	α_g [-]	α_l [-]	α_e [-]
Single-Phase	200.0	355.5	0.0	1.0	0.0
Flashing	1000.0	542.6	0.0	1.0	0.0

Table 5.3 The flow-enthalpy boundary conditions for the single-phase and flashing problems.

The specified mass flow rates, $\dot{m}(t)$, at the bottom of the subchannels were the same for both problems. This time-dependent function is given by (5.1).

$$\dot{m}(t) = \begin{cases} 0.0 & [\frac{\text{lb}_\text{m}}{\text{s}}] \quad , & t \leq 1 \quad [\text{s}] \\ 0.5(t-1) & [\frac{\text{lb}_\text{m}}{\text{s}}] \quad , \quad 1\,[\text{s}] < & t \leq 2 \quad [\text{s}] \\ 0.5 & [\frac{\text{lb}_\text{m}}{\text{s}}] \quad , & t > 2 \quad [\text{s}] \end{cases} \tag{5.1}$$

Both problems adjusted their initial pressure distribution to account for hydrostatic head.

5.1.2 Results

The two models described above were subjected to timestep-sensitivity studies. An automated procedure was used to conduct the timestep sensitivity study. Given an initial Δt_{MAX} , Δt_0, a refinement factor, r_f, and the number of timestep size refinements to be taken, n_t, the simulation was run with both the linear solver and the nonlinear solver for each Δt_{MAX} in the set, calculated according to (5.2).

$$\Delta t_{\text{MAX}} \in \bigcup_{i=0}^{n_t-1} \frac{\Delta t_0}{r_f^i} \tag{5.2}$$

For these two problems, the three parameters, Δt_0, r_f, and n_t, were 1.0, 10.0, and 6, respectively. Each of the two problems were run with both the linear and the nonlinear solver at each of the six different Δt_{MAX} ; in total there were 24 simulations run. The nonlinear convergence criteria used for the nonlinear simulations were k_{max} = 35, F_{tol} = 1.0E-6 , and δ_{tol} = 1.0E-8 .

The results from the simulations were analyzed to determine the impact of nonlinear convergence upon time-step size sensitivity.

The single-phase case was designed to test if the linear solver produced a result that was equivalent to that produced by the nonlinear solver. More specifically, it was designed to show that the linear solver provided as accurate a solution, as the nonlinear solver for problems where the physics of interest have relatively low nonlinearities. Figure 5.2 and Figure 5.3 show the solution produced by both the nonlinear and linear solvers of COBRA. The solutions produced by both solvers were qualitatively equivalent at $\Delta t_{\text{MAX}} = 1.0\text{E-}0$ [s] and $\Delta t_{\text{MAX}} = 1.0\text{E-}5$ [s]. This indicates that the linear solver is adequate in regions where the physics of interest are approximately linear and that the nonlinear solver is capable of reproducing these results.

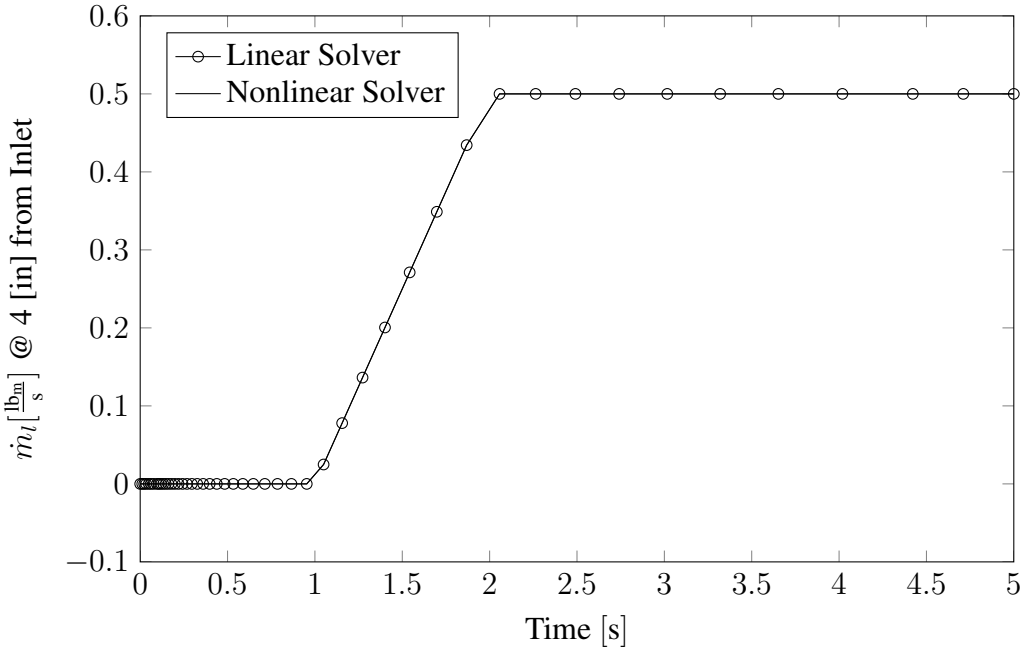

Figure 5.2 Single-phase solution with $\Delta t_{\text{MAX}} = 1.0\text{E-}0$ [s].

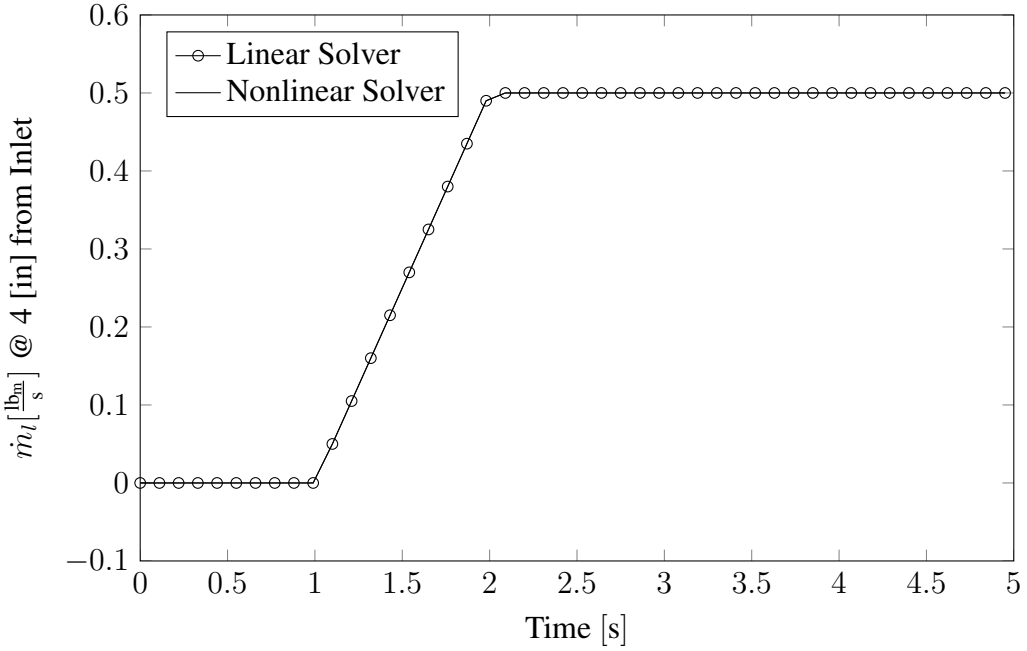

Figure 5.3 Single-phase solution with $\Delta t_{\text{MAX}} = 1.0\text{E-}5$ [s].

For the flashing problem, the parameter of interest for temporal convergence testing was α_g at 2 [in] from the inlet of the subchannel. This parameter was chosen because it represents the region where the nonlinearities should have the largest impact on the solution as this is the location where the flashing occurs. As such, this location will be sensitive to the rate of vapor formation from the thermodynamically unstable liquid that enters the cell. Figure 5.4 shows the gaseous volume fraction 2 [in] from the inlet of the subchannel as a function of time for a Δt_{MAX} of 1.0E-1 [s] for both the linear and the nonlinear solvers.

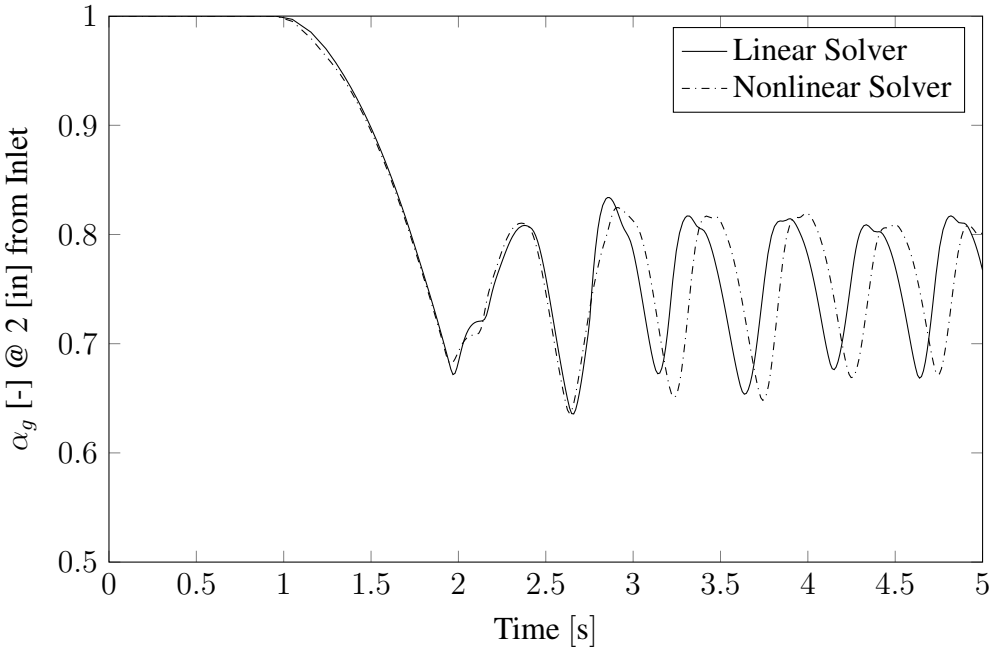

Figure 5.4 Flashing solution at $\Delta t_{\text{MAX}} = 1.0\text{E-}1$ [s].

Figure 5.4 shows that the nonlinearly resolved solution is qualitatively different from the linear solution. Even as the Δt_{MAX} was reduced, this discrepancy does not disappear, as shown in Figure 5.5. Both Figure 5.6 and Figure 5.7 show that as the timestep size was reduced the two different solutions became more timestep-size insensitive. However, the solution produced by the nonlinear solver, Figure 5.7, qualitatively varied less as the Δt_{MAX} was reduced than that produced by the linear solver, Figure 5.6.

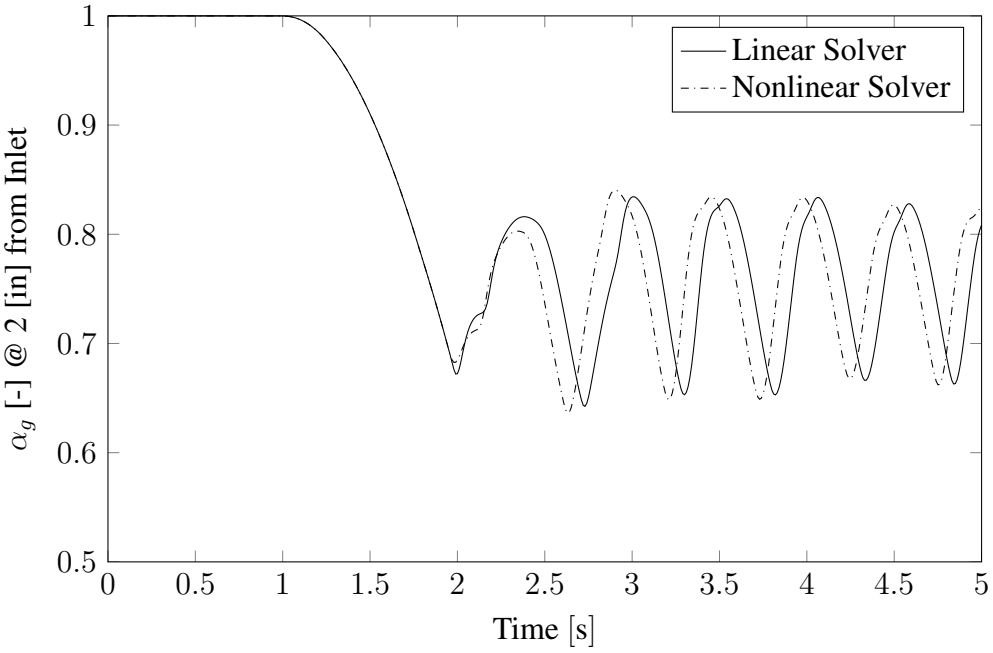

Figure 5.5 Flashing solution at $\Delta t_{\text{MAX}} = 1.0\text{E-}5$ [s].

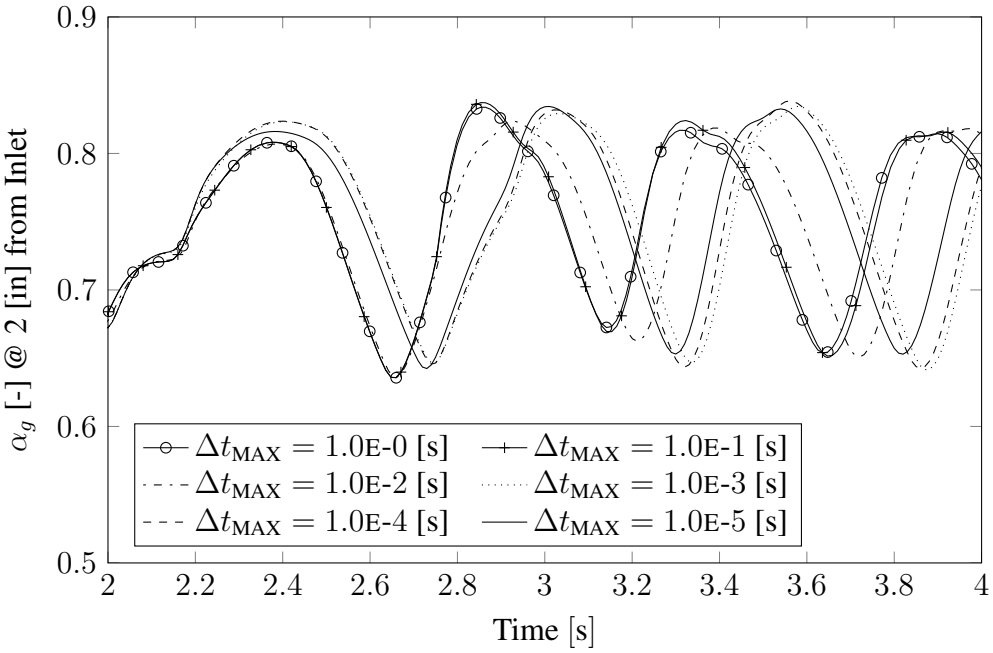

Figure 5.6 Linear solver flashing solutions.

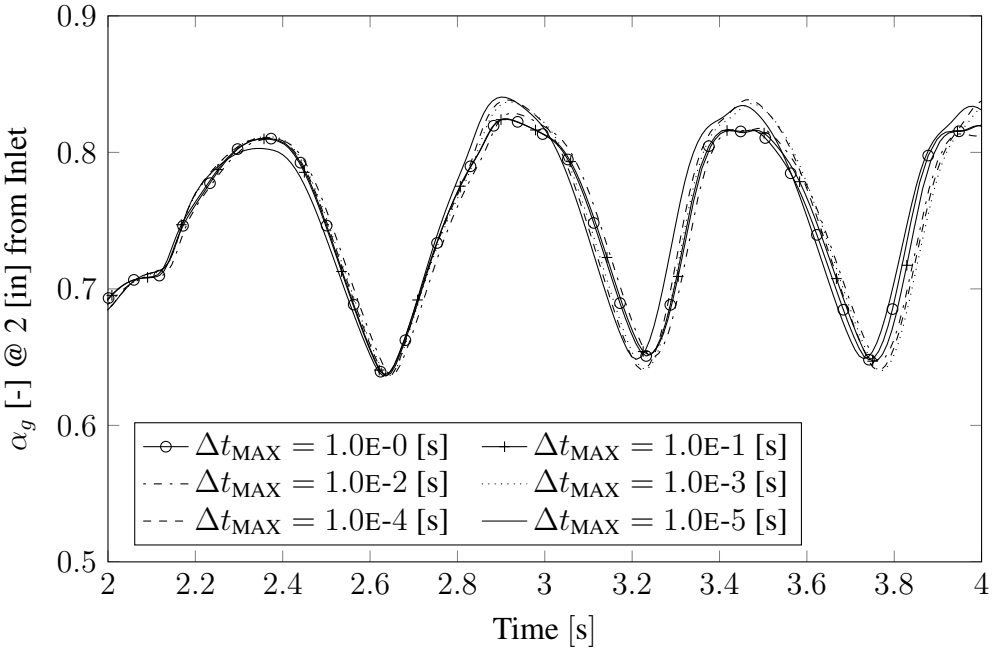

Figure 5.7 Nonlinear solver flashing solutions.

The two solvers, when applied to the same problem that contains highly nonlinear physics, produced two different timestep-size invariant solutions. These two solutions achieved qualitative timestep-size invariance at different timestep sizes. The parameter of interest in the solution produced by the nonlinear solver with $\Delta t_{\text{MAX}} = 1.0\text{E-}0$ [s] was qualitatively close to that produced with $\Delta t_{\text{MAX}} = 1.0\text{E-}1$ [s], Figure 5.8. The same level of qualitative timestep-size invariance was not evident in the linear solver's solutions until the solutions with a Δt_{MAX} of $1.0\text{E-}3$ [s] and $1.0\text{E-}4$ [s] were compared, as shown in Figure 5.9.

The timestep-size insensitive solution produced by the nonlinear solver occurred at a maximum timestep size three orders of magnitude greater than that achieved by the linear solver. An examination of the nonlinear residual over the course of the transient provided insight into why this behavior was observed. For the linear runs, the scaled residuals were evaluated after a single Newton step, $\tilde{\mathbf{F}}^{\,1}$. For the nonlinear runs, the scaled residuals were evaluated at the end of the Newton-loop, $\tilde{\mathbf{F}}^{\,k}$. The linear solver's solution's scaled residual indicated that, even for small timestep sizes, the solution obtained did not satisfy the discrete nonlinear equations, Figure 5.10.

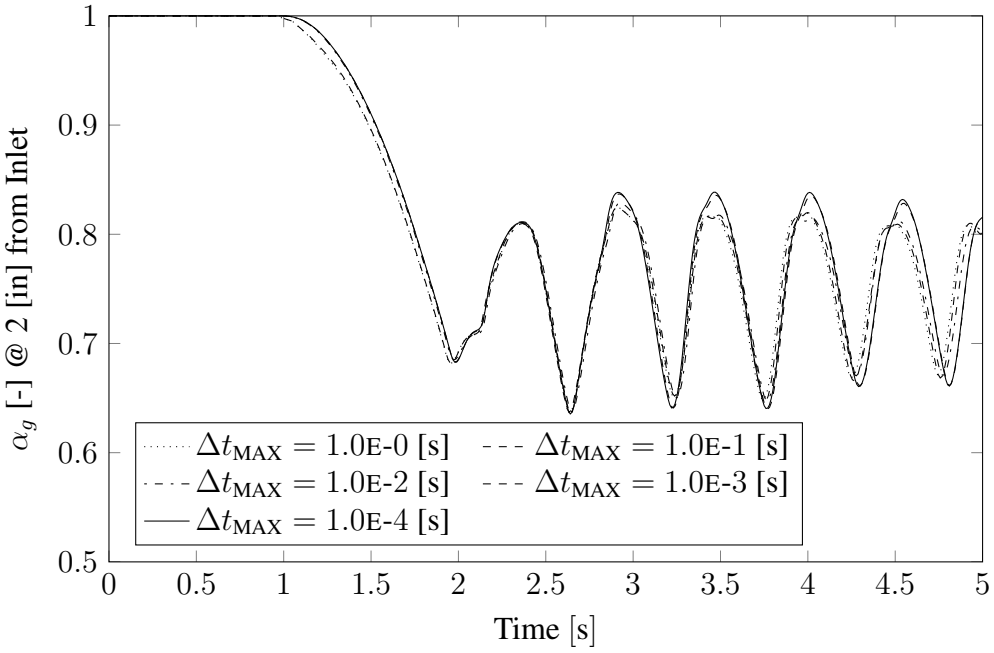

Figure 5.8 Nonlinear solver timestep-size insensitive flashing solution.

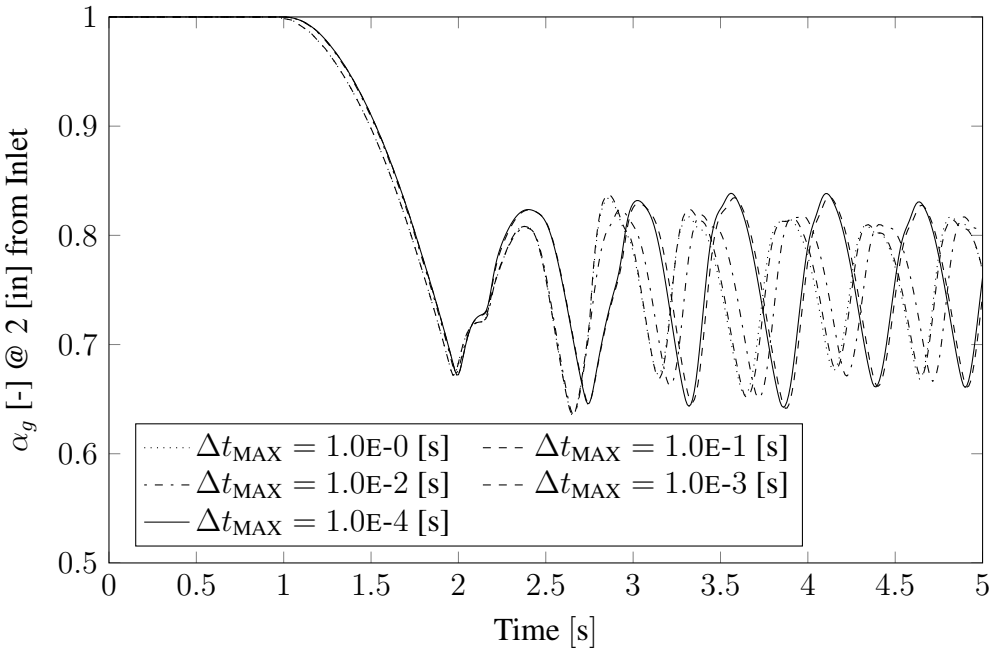

Figure 5.9 Linear solver timestep-size insensitive flashing solution.

This was in contrast to the nonlinear solver's solution, which showed a lower residual over the course of the simulation, Figure 5.11.

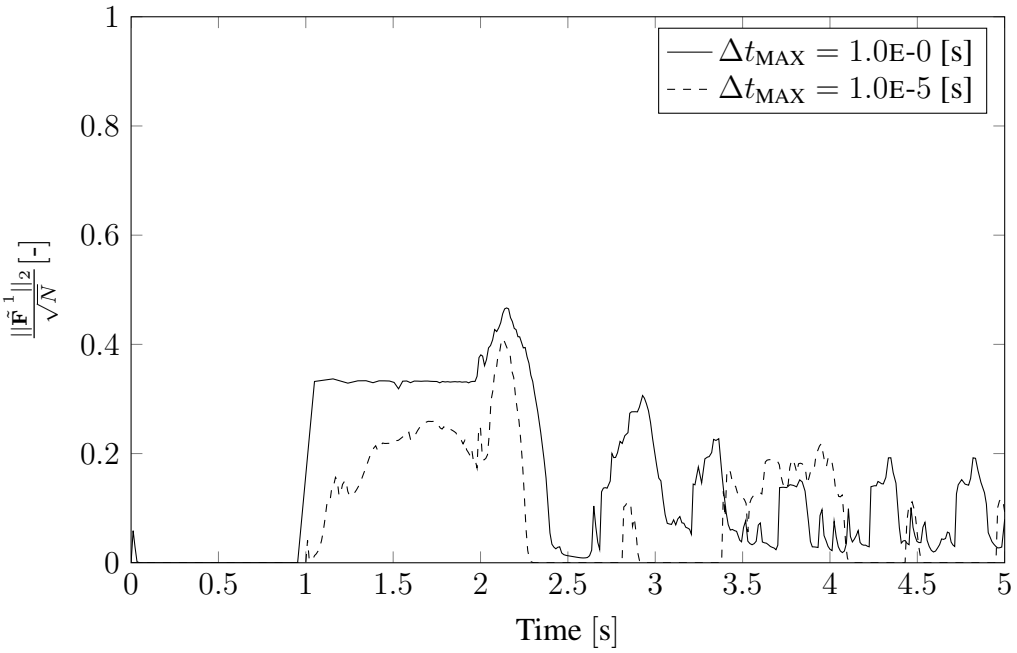

Figure 5.10 Residual of the flashing solution for the linear solver.

The reduction in the residual produced by the linear solver, Figure 5.10, shows that the reduction of the maximum allowable timestep size serves two purposes. The first is that as Δt_{MAX} is reduced the nonlinear physics are being better resolved; however, even for small Δt_{MAX} , the residuals are still large compared to those of the nonlinear solution, Figure 5.11. The second purpose in reducing Δt_{MAX} is to decrease the error due to the discrete approximation of the temporal integral of the governing equations. While the reduction of the maximum timestep size in the linear solver serves the purpose of reducing the residual, the nonlinear solver performs this task naturally. Therefore, in the nonlinear solver the reduction of the maximum timestep size primarily serves to reduce the error from the approximate discrete temporal integral.

Taken together, the above results indicate that the nonlinear solver was implemented correctly and performed as intended.

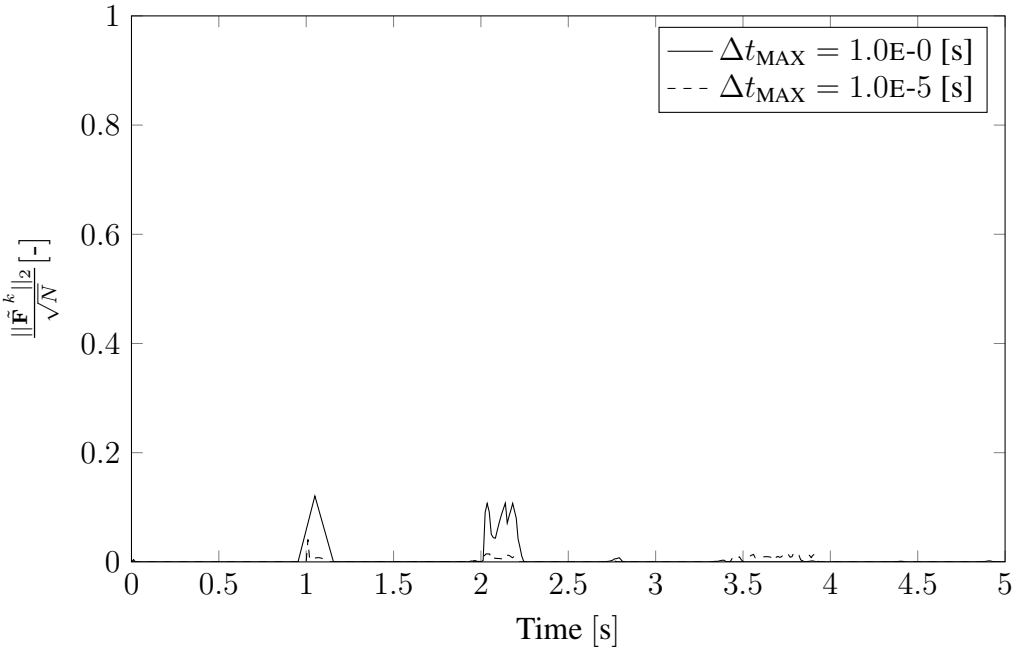

Figure 5.11 Residual of the flashing solution for the nonlinear solver.

5.2 Complex Geometry Problem

The complex geometry problem tested whether the domain decomposition algorithm was implemented properly. To properly test the domain decomposition algorithm many different permutations of the composition of the nonlinear domain were required. The presence of both axial momentum flow paths and transverse momentum flow paths were also necessary. If the nonlinear domain of the decomposed problem were subjected to a single Newton step, then the resulting dual domain solution should match within numerical round off that of a single domain with a single Newton step regardless of the composition of the nonlinear domain. A model of the General Electric (GE) nine-rod subchannel experimental facility [22] suited the requirements of this test.

5.2.1 Model

The GE experiments were conducted in an experimental facility with a nine-rod bundle representative of a boiling water reactor subchannel. The exact geometry of the experimental facilities can be found in the GE technical report [22]. The basic layout of the portion of the facility that was

of interest to this work was as follows: there was an inlet plenum that opened up into a nine-rod bundle whose cross-section is depicted in Figure 5.12; above the rod-bundle there was an outlet plenum.

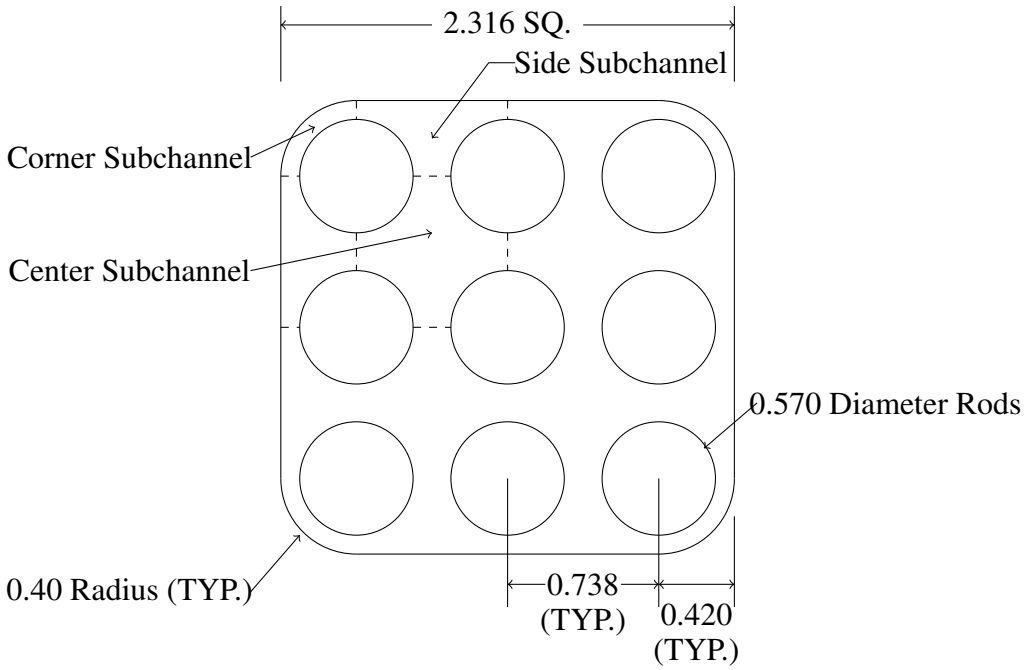

Figure 5.12 Cross-section of rod bundle geometry adapted from GE experiment [22].

The computational model was broken into three sections. The first section was the inlet plenum, while the third section was the outlet plenum. The inlet plenum was 12 [in] tall with a cross-sectional area of 2.93 $[\text{in}^2]$. The outlet plenum was identically dimensioned. The second section represented the rod-bundle. This section was 121.5 [in] tall. The rod-bundle was approximated by a four-by-four grid of sixteen subchannels. There were the four corner subchannels, the eight side subchannels, and the four center subchannels in the model. The dimensions for the various subchannels are shown in Figure 5.12.

The primary purpose of this problem was to test the ability of the domain decomposition algorithm to accurately recover the linear solution when the nonlinear domain was restricted to a single Newton step. For that reason the details of the boundary and initial conditions are omitted here.

There were eighteen subchannels in this model: the 16 subchannels in the rod bundle, the inlet plenum, and the outlet plenum. This meant that there were 2^{18} possible ways to decompose this problem with the domain decomposition algorithm. The time required to run all possible permutations was prohibitive, so only a subset were simulated here. To test if the domain decomposition algorithm was implemented correctly, 256 different permutations were selected at random and run.

5.2.2 Results

The intent of this problem was to determine if the domain decomposition algorithm was properly implemented. When using the domain decomposition algorithm, if only a single Newton step is permitted in the nonlinear domain, then solution obtained should be equivalent to taking only a single Newton step over the entire domain. This provided the basis for the following comparison. Regardless of how the 2^{18} subchannels of the problem were allocated between the linear and the nonlinear domains, the solutions produced by the domain decomposition algorithm should have been within round-off error to that produced by the single domain linear solver. However, the different linear algebra involved with the domain decomposition should not produce the same floating-point solution as that produced by the linear solver. This floating-point variance should not influence the solution above the level of numerical round-off error. However, the traditional COBRA graphics file was created with single precision variables. These single precision variables were incapable of capturing the error present in the simulation since, to single precision, the results were always the same between the domain decomposed and the non-decomposed solutions. A special version of COBRA was compiled that produced the variables of interest in double precision.

There were 256 different simulations run: one where the entire domain was linear, one where the entire domain was nonlinear and 254 where a random number of random subchannels were nonlinear. This collection was a representative subset of the 2^{18} possible decompositions. For each of these simulations, the pressures at two locations were recorded: the pressures near the top and near the bottom of the upper left corner subchannel of Figure 5.12 were chosen. The differences between the pressures at each of these two points and the pressures produced by the linear solver were summed over the course of the transient. (5.3) shows this value for a given

transient simulation, k, at a given location in the simulation, j, over the course of a transient with N_t timesteps.

$$\text{Error}_{j,k} = \sum_{i=1}^{N_t} \left| P_{j,1} - P_{j,k} \right| \tag{5.3}$$

The bar graph in Figure 5.13 shows the distribution of these errors at both the top and bottom of the subchannel.

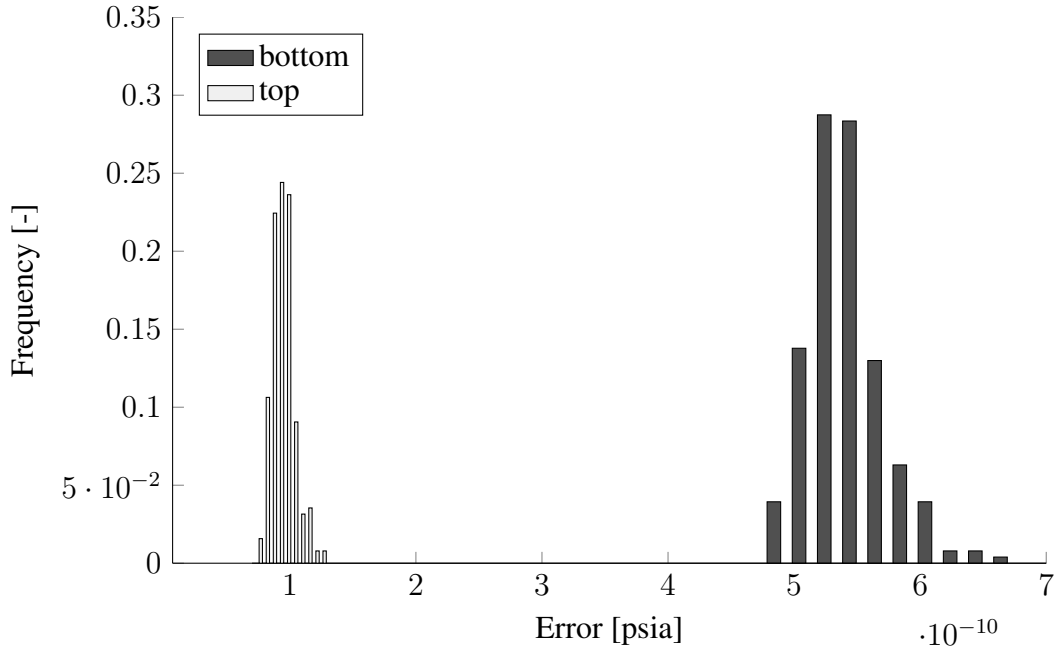

Figure 5.13 Histogram of pressure error at two locations.

These results indicate that the domain decomposition algorithm was implemented correctly.

5.3 Valve-Type Problem

This problem was designed to demonstrate the ability of the domain decomposition algorithm to resolve isolated nonlinearities at a lower computational cost than the fully nonlinear solver. First, the model will be discussed and initial and boundary conditions provided. Second, the timestep sensitivity study of the solution for both the linear and the nonlinear solver will be detailed. Next, the portion of the domain with the nonlinearities will be isolated via the domain decomposition

algorithm and another timestep sensitivity study will be conducted. Finally, the solution and run time data from the three methods will be compared.

5.3.1 Model

The system being modeled was that of four vertical subchannels, Figure 5.14.

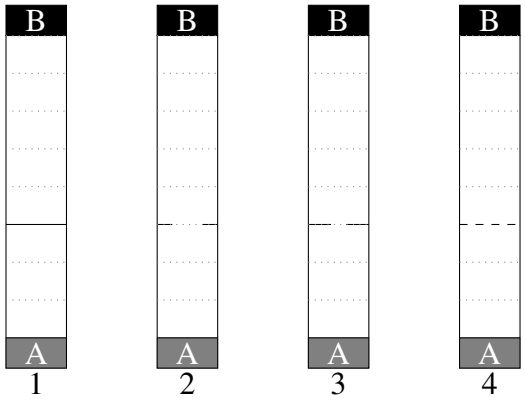

Figure 5.14 Model of valve problem.

Each subchannel was modeled as having eight continuity volumes, each with a height, Δx, of 5 [in] and a cross-sectional area, A_c, of 16 [in^2]. Subchannel 1 had a complete obstruction that prevented flow. Subchannels 2 and 3 each had a fixed obstruction, manifested as loss coefficients, along their lengths. Subchannel 4 had a boundary condition that imitated the behavior of a valve by using a time dependent loss coefficient. Initially fully open, the valve slowly closed. While closing, the valve stopped twice at cross-sectional areas that were equivalent to the two loss coefficients of the two partially obstructed subchannels. After fully closing, the valve returned to its first stopping point.

The subchannels were initially full of subcooled liquid at 800 [psia]. The bottoms and the tops of the subchannels were open to reservoirs. The bottom reservoir, denoted by the grey squares marked with the letter A, was full of subcooled liquid with a time variant pressure. The pressure in the bottom reservoir started at 802 [psia] and increased to 818 [psia] over the first second, staying there for the duration of the problem. The top reservoir, denoted by the black squares marked with the letter B, was full of subcooled liquid at 800 [psia]. This pressure did not change over the course

of the transient. The pressure difference that existed between these two reservoirs was what drove the flow in this problem.

The model used was a single section. Within that section there were four subchannels. These subchannels were hydrodynamically isolated from each other. The two partially obstructed subchannels had user-defined loss coefficients placed at an elevation of 15 [in]. The two coefficients were 40 $[-]$ and 160 $[-]$. The fully obstructed subchannel had a no-flow boundary condition at 15 [in]. The fourth subchannel had a time dependent area ratio multiplier, $A(t)_r$, at 15 [in]. This multiplier acted to reduce the area of the momentum flow path, A_m, used in the determination of form losses and wall drag. However, these four subchannels had their wall drag terms artificially reduced. Reducing the wall drag allowed the impact of the loss coefficients upon the solution to be isolated. Since there was no wall drag in this problem, the area ratio could be related to the effective loss coefficient, $K(t)$, by (5.4).

$$K(t) = \frac{K_o}{A(t)_r{}^2} \tag{5.4}$$

The base loss coefficient, K_o, for subchannel four was 10 [-]. Using (5.4), the equivalent area ratios for the loss coefficients of the two partially obstructed subchannels, 40 [-] and 160 [-], were 0.5 and 0.25 respectively. The time dependent area ratio multiplier for subchannel four is shown in Figure 5.15.

The simulation was run to an end time of 19 seconds. The functional form of the transient area boundary condition was chosen so that the solution of the transient area subchannel would mimic each of the three other subchannels for portions of the transient.

5.3.2 Results

The parameter of interest was the flow rate of liquid at 15 [in] above the inlet. This location corresponded to the location of the loss coefficients, the no-flow boundary condition, or the variable momentum-area boundary condition in each of the respective subchannels. For this study, Δt_0, r_f, and n_t used in (5.2) were 1.0, 2.0, and 10, respectively. Upon examining the solution, it was determined that timestep size was Δt_{crnt} limited for a portion of the transient until Δt_{MAX}

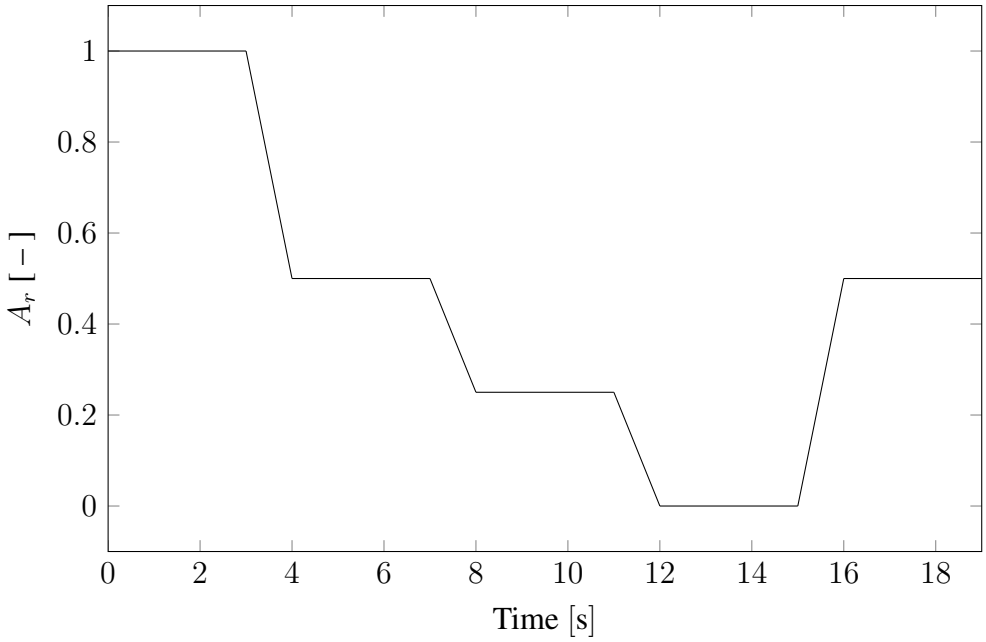

Figure 5.15 Transient area ratio for valve-type problem.

= 1.56E-2 [s] was reached. The timestep histories of the simulations from $\Delta t_{\text{MAX}} = 1.0\text{E-}0$ [s] through $\Delta t_{\text{MAX}} = 6.25\text{E-}2$ [s] were identical for both the linear and the nonlinear solution. As such, the first four simulation results for each solver were dropped from the data sets since they represented duplicate results. The solution obtained with $\Delta t_{\text{MAX}} = 6.25\text{E-}2$ [s] with the linear solver is shown in Figure 5.16.

This solution shows that the there is a non-physical spike in the mass flow rate when the closed valve starts to open at 15 [s]. This behavior was observed for the linear solver's solutions at successively smaller max timesteps through the smallest Δt_{MAX} , 1.95E-3 [s].

The presence of the non-physical behavior in the flow rate in the simulation with $\Delta t_{\text{MAX}} =$ 1.56E-2 [s], a Δt_{MAX} that produced a Δt history that was never Δt_{crnt} limited, necessitated that additional timestep refinements be examined for a smooth solution. Subsequent timestep refinements produced results where the flow-rate spikes became progressively smaller; however, the time-step refinement study did not yield a solution that was free of this perturbation, as shown in Figure 5.17. The lack of a linear solution that was free from this perturbation was not an

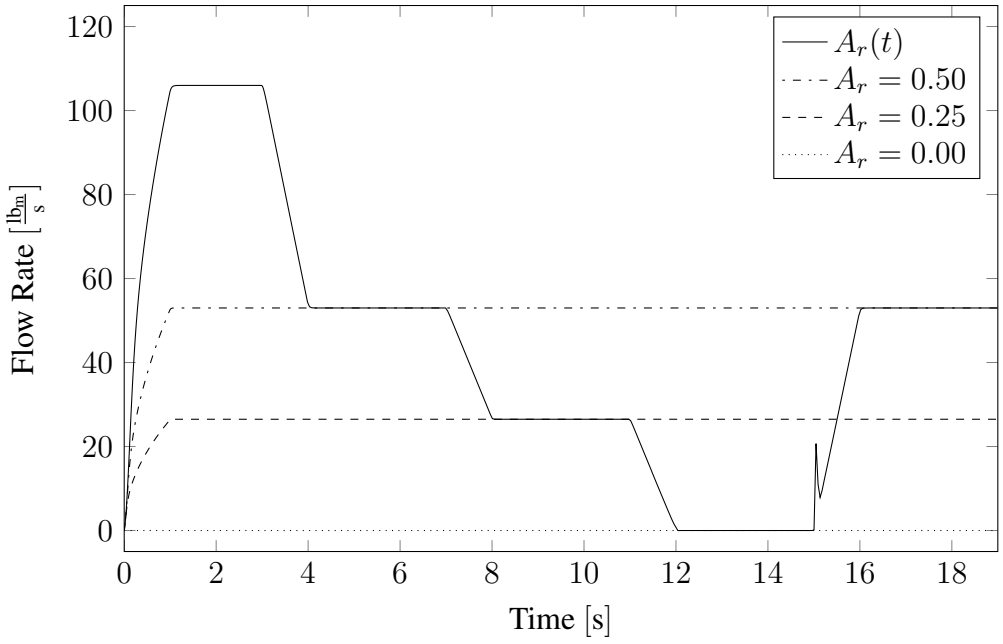

Figure 5.16 Valve problem with $\Delta t_{\text{MAX}} = 6.25\text{E-}2$ [s] with the linear solver.

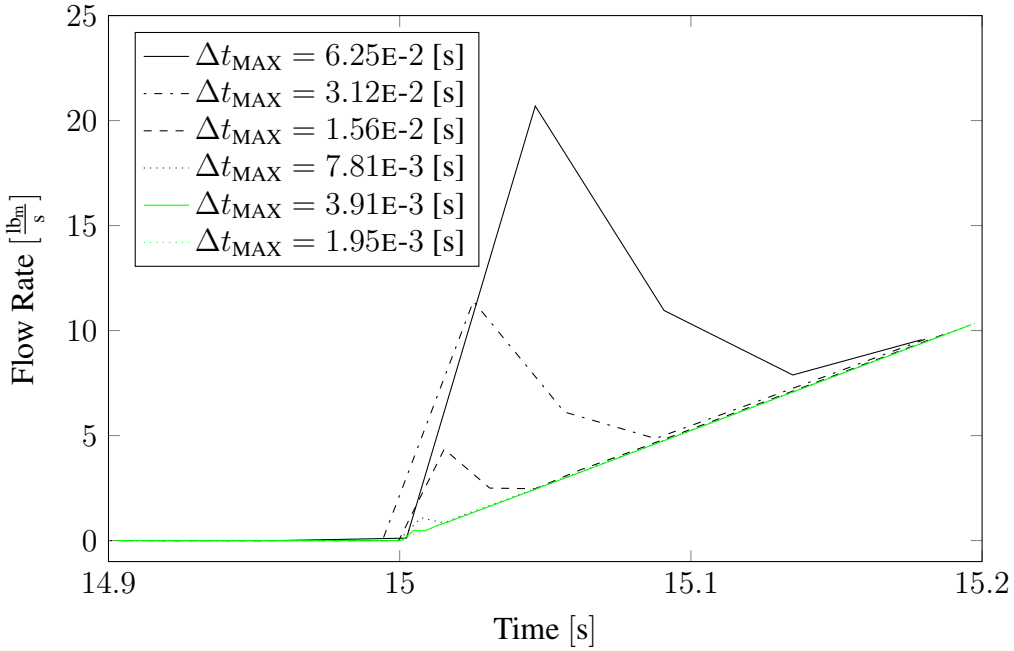

Figure 5.17 Zoom of non-physical behavior at 15 [s] in linear solutions.

impediment to this study since the purpose was not to find a Δt_{MAX} that produces such a solution. Instead, Table 5.4 shows the run time statistics for the linear solver's solutions.

Δt_{MAX} [s]	N_t [$-$]	T_{CPU} [s]	$\frac{T_{\text{CPU}}}{N_t}$ [s]
$6.3E{-}2$	539	1.388	$2.575E{-}3$
$3.1E{-}2$	680	1.836	$2.700E{-}3$
$1.6E{-}2$	1247	4.524	$3.628E{-}3$
$7.8E{-}3$	2458	7.257	$2.952E{-}3$
$3.9E{-}3$	4883	13.037	$2.670E{-}3$
$2.0E{-}3$	9742	24.578	$2.523E{-}3$

Table 5.4 Run time data for the valve problem using the linear solver.

In Table 5.4 the first column shows the maximum timestep size for the simulation. The second column shows the number of successful timesteps taken, N_t. The third column is the CPU time required to complete simulation, T_{CPU}. The final column is the CPU time per timestep for a given simulation. On average, the CPU time per timestep was 2.8E-3 [s] for the linear solution.

Next, the nonlinear solver was used to perform the same timestep sensitivity study. The same set of Δt_{MAX} was run. The solution at Δt_{MAX} = 6.25E-2 [s] is shown in Figure 5.18

The solution obtained with a Δt_{MAX} = 6.25E-2 [s] did not exhibit the nonphysical spike in the flow rate that the linear solution displayed. This solution was also obtained at the largest Δt run in the study, 1.0E-0 [s]. Reducing the Δt_{MAX} produced solutions that were qualitatively identical to that shown above. Figure 5.19 shows the same zoomed view of at 15 [s], when the valve opens.

These results indicate that the linear solver's timestep refinement serves the purpose of trying to resolve an isolated linearization error in a single subchannel of the problem. By using the nonlinear solver, that error was removed and subsequent timestep reductions were not necessary to obtain the physical solution. However, Table 5.5 shows that the nonlinear solver takes more CPU time per timestep than the linear solver.

Given that the nonlinearities in this problem were clearly isolated in a single subchannel, using the domain decomposition algorithm provided an ideal test to determine if a nonlinearly converged solution could be obtained at a lower computational cost than the full nonlinear solver. The use

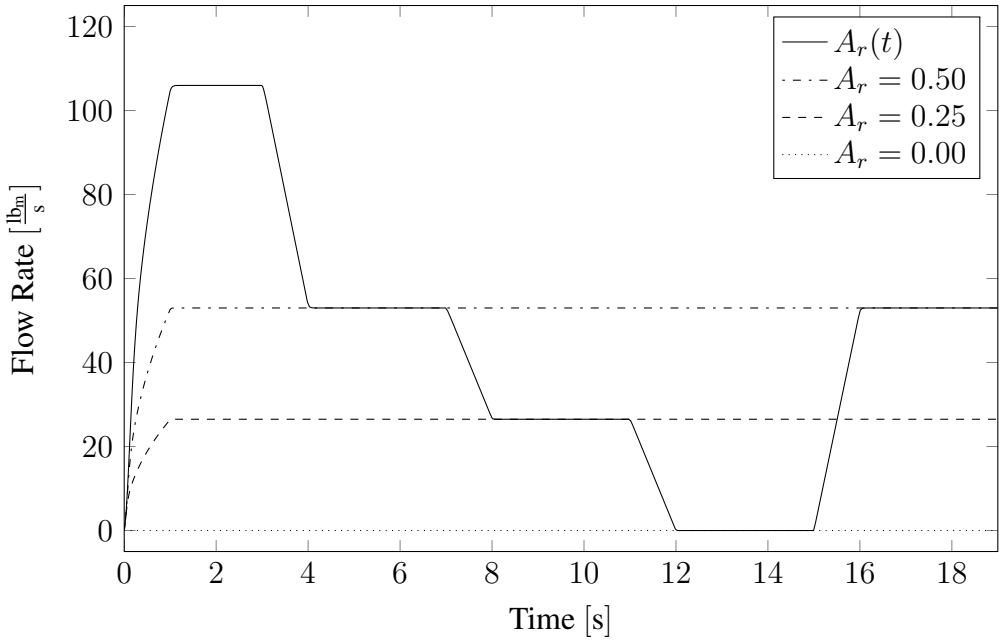

Figure 5.18 Valve problem with Δt_{MAX} = 6.25E-2 [s] with the nonlinear solver.

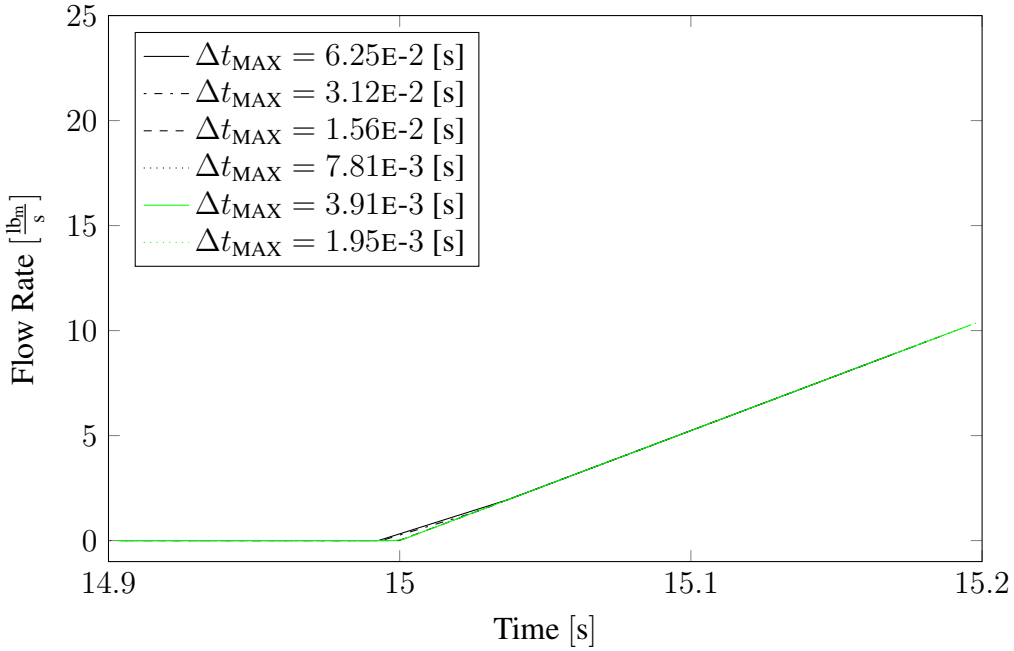

Figure 5.19 Zoom of nonlinear solutions at 15 [s].

Δt_{MAX} [s]	N_t [$-$]	T_{CPU} [s]	$\frac{T_{\text{CPU}}}{N_t}$ [s]
$6.3E{-}2$	536	4.028	$7.515E{-}3$
$3.1E{-}2$	680	4.78	$7.030E{-}3$
$1.6E{-}2$	1247	6.784	$5.441E{-}3$
$7.8E{-}3$	2458	12.385	$5.039E{-}3$
$3.9E{-}3$	4883	18.593	$3.808E{-}3$
$2.0E{-}3$	9742	32.686	$3.355E{-}3$

Table 5.5 Run time data for the valve problem using the nonlinear solver.

of the domain decomposition algorithm in this problem was restricted to the subchannel with the valve. The other three subchannels were not included in the nonlinear domain. The same set of timesteps was run to collect data. Figure 5.20 shows the solution as obtained with Δt_{MAX} = 6.25E-2 [s]; the results in this figure are qualitatively indistinguishable from the results of the nonlinear solver.

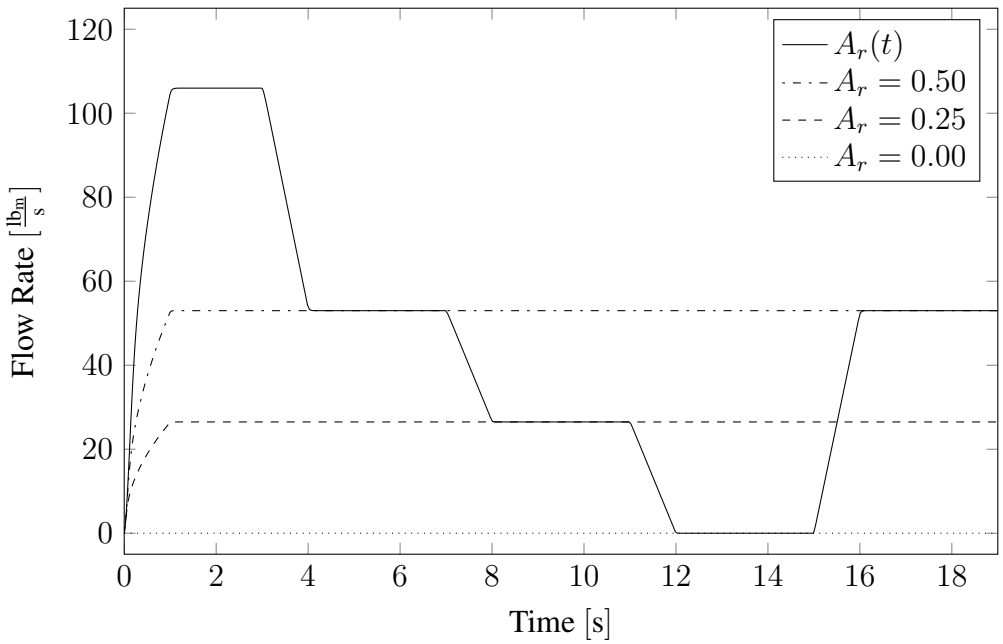

Figure 5.20 Valve problem with Δt_{MAX} = 6.25E-2 [s] and domain decomposition active.

The results for subsequently smaller timesteps were indistinguishable from those presented in Figure 5.19. This result indicates that the nonlinear solution can be obtained with the domain

decomposition algorithm. Since the nonlinearities were designed to be isolated, this result was expected. However, the run time statistics, Table 5.6, provided additional information about the domain decomposition algorithm's performance.

Δt_{MAX} [s]	N_t [$-$]	T_{CPU} [s]	$\frac{T_{\text{CPU}}}{N_t}$ [s]
$6.3E{-}2$	536	2.052	$3.829E{-}3$
$3.1E{-}2$	680	2.548	$3.747E{-}3$
$1.6E{-}2$	1247	3.572	$2.865E{-}3$
$7.8E{-}3$	2458	6.784	$2.760E{-}3$
$3.9E{-}3$	4883	12.693	$2.599E{-}3$
$2.0E{-}3$	9742	23.605	$2.423E{-}3$

Table 5.6 Run time data for the valve problem using domain decomposition.

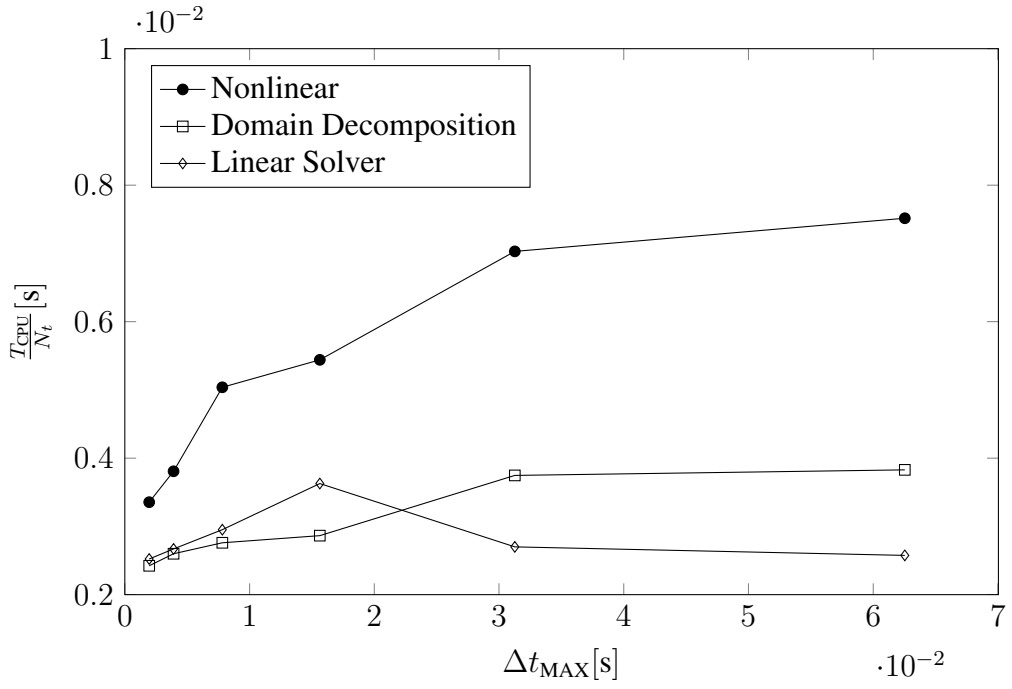

Figure 5.21 CPU time per timestep for the nonlinear and the domain decomposition runs.

The CPU time per time step of the domain decomposition was lower than that obtained by the nonlinear solver. Figure 5.21 shows that the domain decomposition algorithm is capable of

producing the same solution as the nonlinear solver at less than half the cost per timestep than the nonlinear solver.

As the Δt_{MAX} was reduced, the cost of the nonlinear solver decreased because there were fewer nonlinear iterates required per timestep. This reduction in the number of Newton iterates per timestep was also evident in the domain decomposition run-time plot. This problem was developed to show that for a problem with isolable nonlinearities, the application of the domain decomposition algorithm to resolve those nonlinearities would result in a smaller computational cost than the global nonlinear solver while producing results that were qualitatively as accurate.

5.4 Filling Pipe Problem

This problem was developed to illustrate that using the domain decomposition algorithm to resolve localized nonlinearities produced a global solution that was in-line with the nonlinear solution. First, the differences in the solutions obtained by the linear and the nonlinear solvers at large Δt_{MAX} will be shown. Second, the solution at smaller Δt_{MAX} will be shown to be in agreement with the analytic solution for both the linear and the nonlinear solver. Third, the solution produced by the domain decomposition algorithm will be shown as a function of timestep size. Finally, the impact of resolving or not resolving local nonlinearities on the global solution will be discussed.

5.4.1 Model

A vertical pipe slowly filling with saturated water at atmospheric pressure was the basis for this simulation. The pipe was 3 [ft] tall, having a 36 [in^2] cross-sectional area and a wetted perimeter of 24 [in]. The pipe was initially full of saturated steam at 14.7 [psia]. The top of the pipe was open to an infinite reservoir of saturated steam at 14.7 [psia]. Saturated water flowed into the bottom of the pipe, displacing the steam. The boundary and initial conditions are contained in Table 5.7.

The pressure at a given point in the pipe had a simple, closed-form, analytic solution under the assumptions listed below. The first assumption was that the variations in the liquid's specific volume, v_l, within the range of pressures experienced during the course of the simulation could be neglected. The liquid flow rate into the problem is given by (5.5).

	Pressure [psia]	Enthalpy $[\frac{\text{BTU}}{\text{lb}_\text{m}}]$	α_g [-]	α_l [-]	α_e [-]
Initial Conditions	14.7	1150.3	1.0	0.0	0.0
Outlet Conditions	14.7	1150.3	1.0	0.0	0.0
Inlet Conditions	14.7	180.2	0.0	1.0	0.0

Table 5.7 Initial and boundary conditions for the fill problem.

$$\dot{m}_l(t) = \begin{cases} \dot{m}_{l,\text{max}} \cdot t \quad [\frac{\text{lb}_\text{m}}{\text{s}}] \quad , \quad 0\ [\text{s}] < \quad t \quad \leq 1 \quad [\text{s}] \\ \dot{m}_{l,\text{max}} \quad [\frac{\text{lb}_\text{m}}{\text{s}}] \quad , \qquad\qquad t \quad > 1 \quad [\text{s}] \end{cases} \tag{5.5}$$

The maximum flow rate, $\dot{m}_{l,\text{max}}$, was 0.406503 $[\frac{\text{lb}_\text{m}}{s}]$. The volumetric flow rate is given by (5.6).

$$\dot{V}(t) = v_l \dot{m}_l(t) \tag{5.6}$$

The volumetric flow rate could be expressed as $\dot{V}(t) = A_c \dot{h}(t)$, where $\dot{h}(t)$ and A_c were the rate of change in height as a function of time and the cross-sectional area of the pipe, respectively. Integrating (5.6) produced the total volume of liquid in the pipe, $V(t)$, as a function of time, (5.7), which could represented as a function of height, $h(t)$.

$$V(t) = A_c h(t) = \begin{cases} v_l \dot{m}_{l,\text{max}} \frac{t^2}{2} \quad [\text{ft}^3] \quad , \quad 0\ [\text{s}] < \quad t \quad \leq 1 \quad [\text{s}] \\ v_l \dot{m}_{l,\text{max}} \left(t - \frac{1}{2}\right) \quad [\text{ft}^3] \quad , \qquad\qquad t \quad > 1 \quad [\text{s}] \end{cases} \tag{5.7}$$

Neglecting the hydrostatic head contribution from the vapor column, the pressure at any height and point in time in the problem, $P(t, h(t))$, is given by (5.8). If the water level had yet to reach a given point, the pressure was 14.7 [psia], P_o. If the water level had reached a given point, then the pressure at that point was given by the hydrostatic pressure dictated by the height of the liquid column above that point, (5.8).

$$P(t, h(t)) = \begin{cases} P_o \quad [\text{psia}] \quad , \quad 0\ [\text{s}] < \quad t \quad \leq t^*(h) \quad [\text{s}] \\ P_o + g\frac{h(t)-h}{v_l} \quad [\text{psia}] \quad , \qquad\qquad t \quad > t^*(h) \quad [\text{s}] \end{cases} \tag{5.8}$$

The function $t^*(h)$ in (5.8) is given by (5.9) and represents the time it takes the water level to reach a given height.

$$t^*(h) = \begin{cases} \sqrt{\frac{2A_c h}{v_l \dot{m}_{l,\max}}} \quad [\text{s}] \quad , \quad 0\ [\text{ft}] < \quad h \quad \leq \frac{v_l \dot{m}_{l,\max}}{2A_c} \quad [\text{ft}] \\ \frac{A_c h}{v_l \dot{m}_{l,\max}} + \frac{1}{2} \quad [\text{s}] \quad , \qquad\qquad\qquad\quad h \quad > \frac{v_l \dot{m}_{l,\max}}{2A_c} \quad [\text{ft}] \end{cases} \tag{5.9}$$

This analytic function for pressure, evaluated at six different axial heights from 0.0 [s] to 75.0 [s], is shown in Figure 5.22.

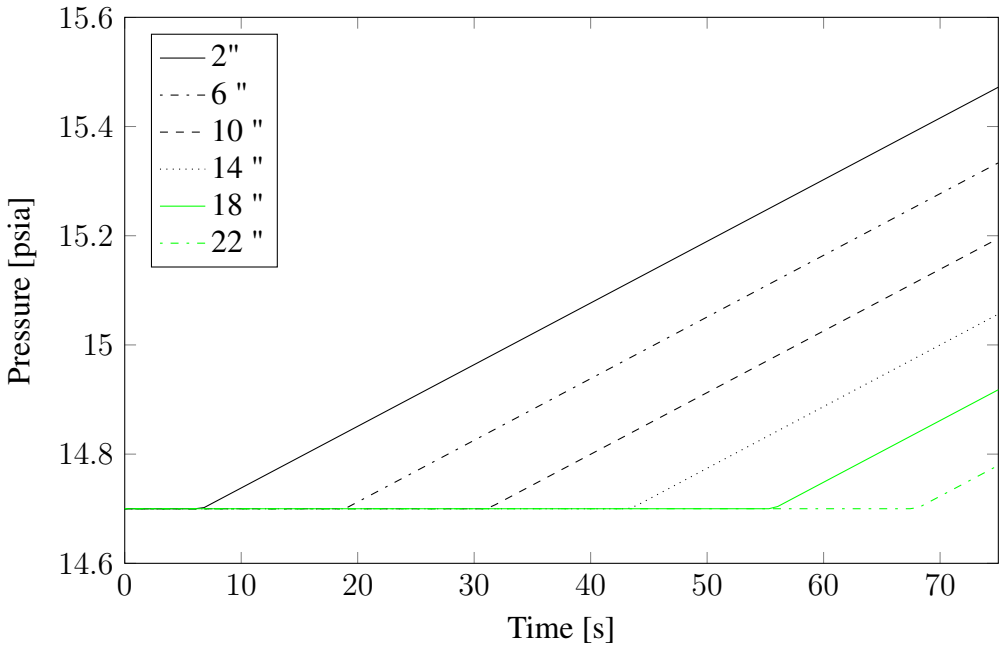

Figure 5.22 Analytic solution to the filling problem.

The model of this pipe was composed of two sections, each with a single subchannel. While it was not necessary for so simple a problem to have multiple sections, it was done to allow the domain decomposition algorithm to run, since the algorithm uses subchannels to decompose the domain. The subchannel in the bottom section was 1 [ft] tall and was broken into three continuity volumes, each 4 [in] high. The top section's subchannel was 2 [ft] tall and was broken into six continuity volumes, each 4 [in] high. Figure 5.23 shows the COBRA model for this problem.

First, the simulation was run once with each of the linear and the nonlinear solvers. The nonlinear convergence parameters were $k_{\max}$ = 35, F_{tol} = 1.0E-6, and δ_{tol} = 1.0E-8. Each

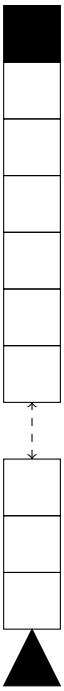

Figure 5.23 COBRA model for the filling problem.

simulation was run for 75 [s]. The Δt_{MAX} was reduced several times over two orders of magnitude to determine timestep sensitivity.

For each simulation, a time-averaged error measure was calculated. The error at a given point was determined by taking the square root of the integral of the squared difference between the analytic pressure, P_{ana}, and the calculated pressure divided by the total simulation run time, T. This error at the center of the lower six continuity volumes was summed to produce a measure of the error in the calculated result, (5.10). The bottom six continuity volumes were selected since the simulation was run only until they were filled.

$$e_p = \frac{1}{\sqrt{T}} \sum_{i=1}^{6} \sqrt{\int_0^T \left(P_{i,\text{ana}}(t) - P_i(t) \right)^2 \mathrm{d}t} \tag{5.10}$$

Next, the domain decomposition algorithm was activated. There were two simulations in this run: one was with the bottom subchannel in the nonlinear domain and the other with the top subchannel in the nonlinear domain. The Δt_{MAX} for these runs was chosen to be 1.0E-1 [s]. The

solutions of these runs are compared to the solutions produced by the single domain solvers in the following section.

5.4.2 Results

The simulation was first run with the linear solver. The Δt_{MAX} was chosen to be 1.0E-1 [s]. The pressure at the center of the bottom six continuity volumes in the problem are shown in Figure 5.24.

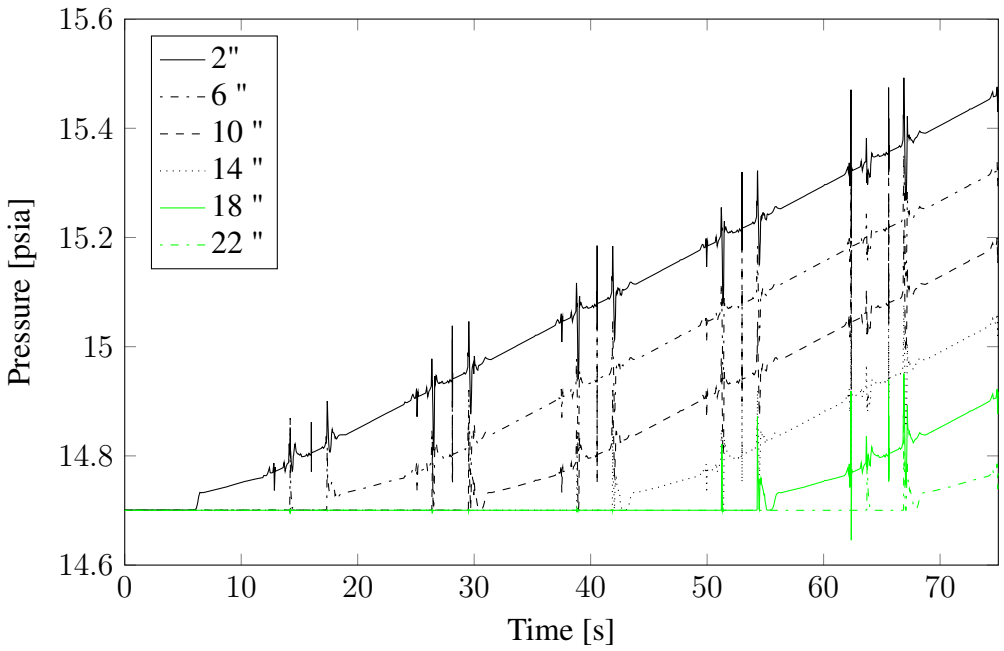

Figure 5.24 Filling problem with Δt_{MAX} = 1.0E-1 [s] with the linear solver.

The solution at this Δt_{MAX} was noisy and when the actual timestep sizes were examined, Δt_{MAX} was infrequently reached due to restrictive Δt_{crnt} limits. The measure of error, e_p, for this simulation was 7.5E-2 [psia].

The solution from the simulation with Δt_{MAX} = 5.0E-2 [s], Figure 5.25, was more in line with the analytic solution, Figure 5.22. The slight spikes visible in Figure 5.25 are characteristic of the COBRA software. They cannot be removed by either resolution of nonlinearities or reduction in timestep size. Table 5.8 shows the run time data for the linear cases that were run.

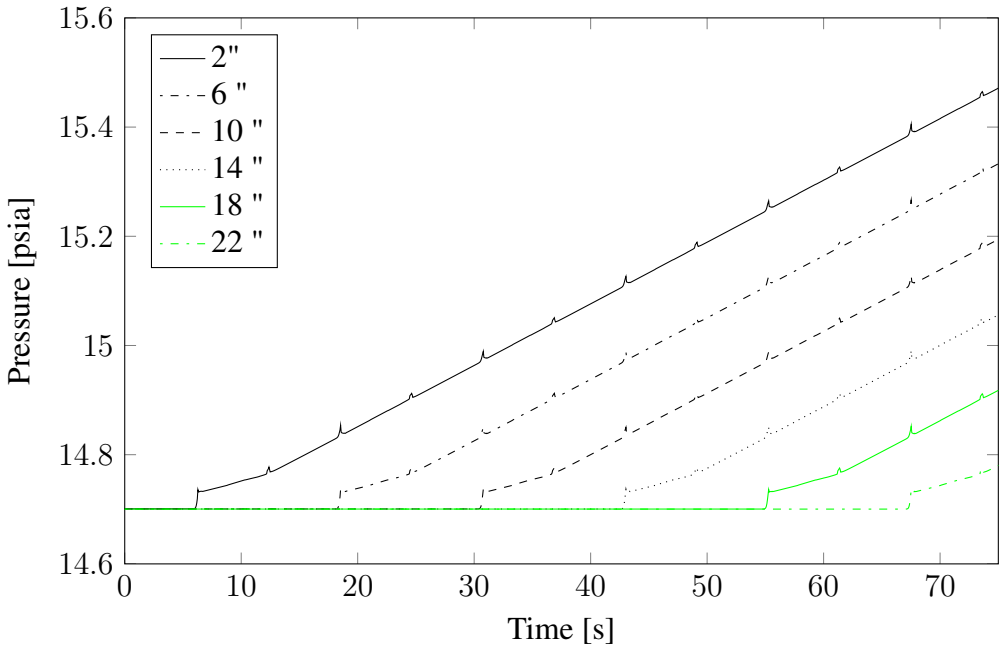

Figure 5.25 Filling problem with Δt_{MAX} = 5.0E-2 [s] using the linear solver.

Δt_{MAX} [s]	N_t [$-$]	T_{CPU} [s]	$\frac{T_{\text{CPU}}}{N_t}$ [s]
$1.0E{-}1$	1367	1.136	$8.311E{-}4$
$5.0E{-}2$	1565	1.296	$8.282E{-}4$
$2.5E{-}2$	3058	2.492	$8.150E{-}4$
$1.3E{-}2$	6051	4.972	$8.217E{-}4$
$6.3E{-}3$	12046	9.861	$8.186E{-}4$
$3.1E{-}3$	24039	19.685	$8.189E{-}4$
$1.6E{-}3$	48036	38.782	$8.074E{-}4$

Table 5.8 Linear solver's data for the fill problem.

With the nonlinear solver active, the same timestep sizes were run. The pressures from the Δt_{MAX} = 1.0E-1 [s] case are shown in Figure 5.26. The measure of error, e_p, for this simulation was 3.5E-2 [psia]. This error measure was half that of the linear case at the same Δt_{MAX} .

The solution obtained by the nonlinear solver at this large Δt_{MAX} was comparable to those obtained by the linear solver with the smaller timestep sizes, Δt_{MAX} = 5.0E-2 [s] and 1.5E-3 [s]. Examining the actual timestep sizes showed that the nonlinear simulation with Δt_{MAX} = 5.0E-2

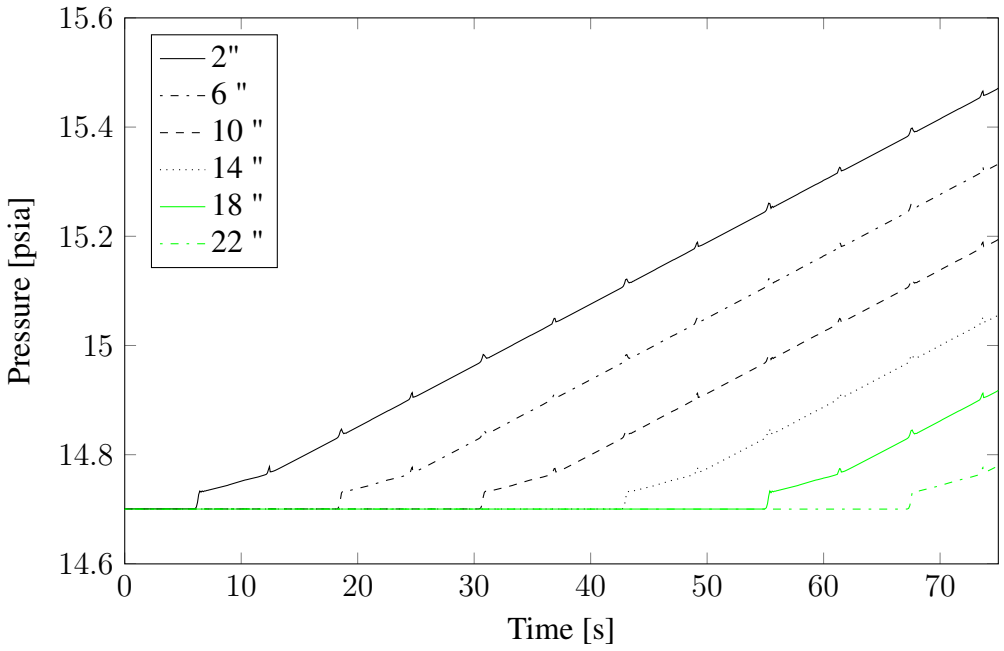

Figure 5.26 Filling problem with Δt_{MAX} = 1.0E-1 [s] with the nonlinear solver active.

[s] actually ran at Δt_{MAX} for the majority of timesteps. This indicates that a more accurate solution can be obtained with fewer timesteps for the same Δt_{MAX} as a result of resolving the nonlinearities in a problem.

The nonlinear solver took fewer total timesteps than the linear solver in this simulation. This was possible because the calculated Δt_{crnt} limit was different for the nonlinear case. The larger Δt_{crnt} was possible because the nonphysical oscillations in the linear solution's pressures were no longer present. These deviations had induced large velocities, which had reduced Δt_{crnt} below Δt_{MAX} causing COBRA to reduce the timestep size repeatedly. The nonlinear solver produced qualitatively and quantitatively similar solutions at more restrictive Δt_{MAX} . At the smaller Δt_{MAX} both the linear and the nonlinear solver's solutions had similar error measures, approximately 3.5E-2 [psia].

Reducing the nonlinear error at each timestep allowed COBRA to run at larger Δt without a reduction in the accuracy of the solution. However, the use of the nonlinear solver imposed additional computational costs. These costs are evident in Table 5.9. The run time per timestep

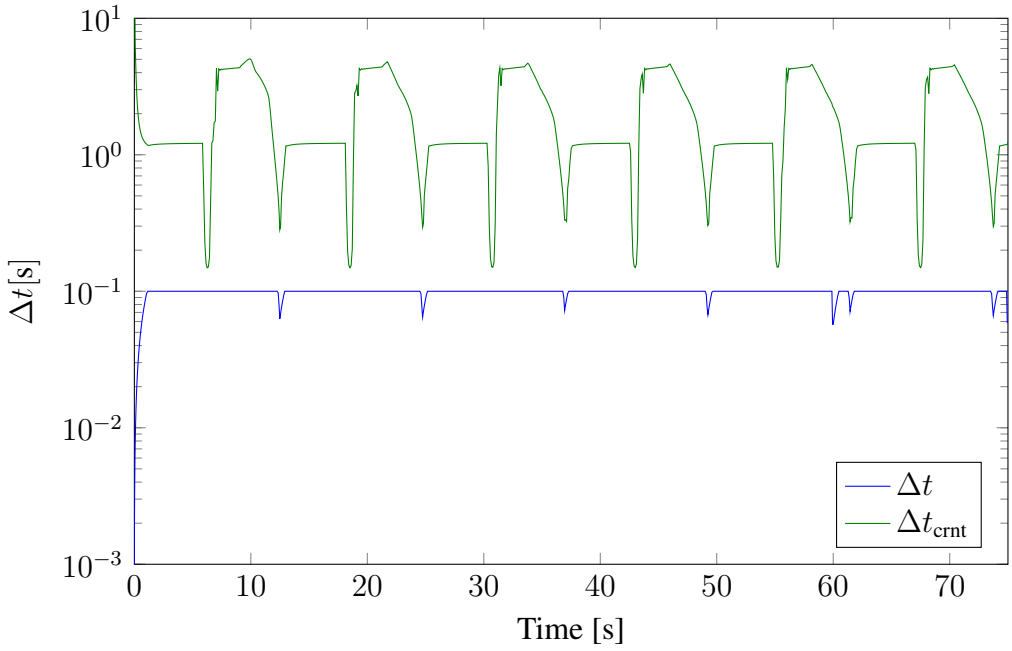

Figure 5.27 Comparison of Δt to Δt_{crnt} for the nonlinear solver with Δt_{MAX} = 1.0E-1 [s].

of the nonlinear solver for this problem was on average 5.5 times that of the linear solver. These results indicate that using the nonlinear solver at large timestep sizes can produce solutions that are comparable to those from the linear solver at smaller timesteps.

Δt_{MAX} [s]	N_t [$-$]	T_{CPU} [s]	$\frac{T_{\text{CPU}}}{N_t}$ [s]
$1.0E{-}1$	830	4.952	$5.967E{-}3$
$5.0E{-}2$	1565	9.001	$5.751E{-}3$
$2.5E{-}2$	3058	16.685	$5.456E{-}3$
$1.3E{-}2$	6051	30.55	$5.049E{-}3$
$6.3E{-}3$	12046	56.932	$4.726E{-}3$
$3.1E{-}3$	24039	103.41	$4.302E{-}3$
$1.6E{-}3$	48036	192.71	$4.012E{-}3$

Table 5.9 Nonlinear solver's data for the fill problem.

The application of the domain decomposition algorithm provided more insight into the impact of unresolved local nonlinearities. The domain decomposition simulations were run with Δt_{MAX}

= 1.0E-1 [s]. The first simulation to be examined will be the one with the bottom subchannel in the nonlinear domain, Figure 5.28.

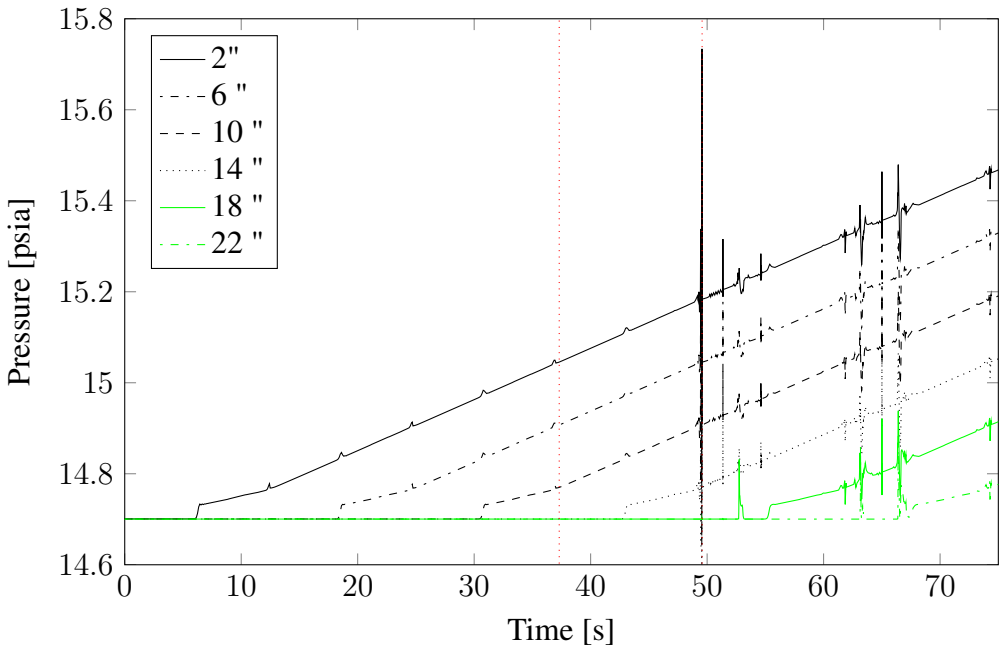

Figure 5.28 Filling problem with the bottom subchannel in the nonlinear domain.

While the water front was in the nonlinear domain, the pressures everywhere were in alignment with the analytic solution. However, once the water front entered the linear domain, the unresolved nonlinearities in the linear domain induced nonphysical pressure oscillations in both domains. This result was mirrored in the case where the top subchannel was in the nonlinear domain, Figure 5.29.

Figure 5.29 shows that when the water level was in the bottom subchannel, the nonphysical solution propagated globally. In addition, once the water level entered the nonlinear domain, the solution was out of sync with the analytic solution. This discrepancy was due to the time lag introduced by the linear solver while the water level was in the linear domain.

This problem highlighted the impact of unresolved nonlinearities on the global solution. Isolating places where the nonlinearities were present and solving those regions with the nonlinear solver allowed for a global solution that more accurately solved the nonlinear equations.

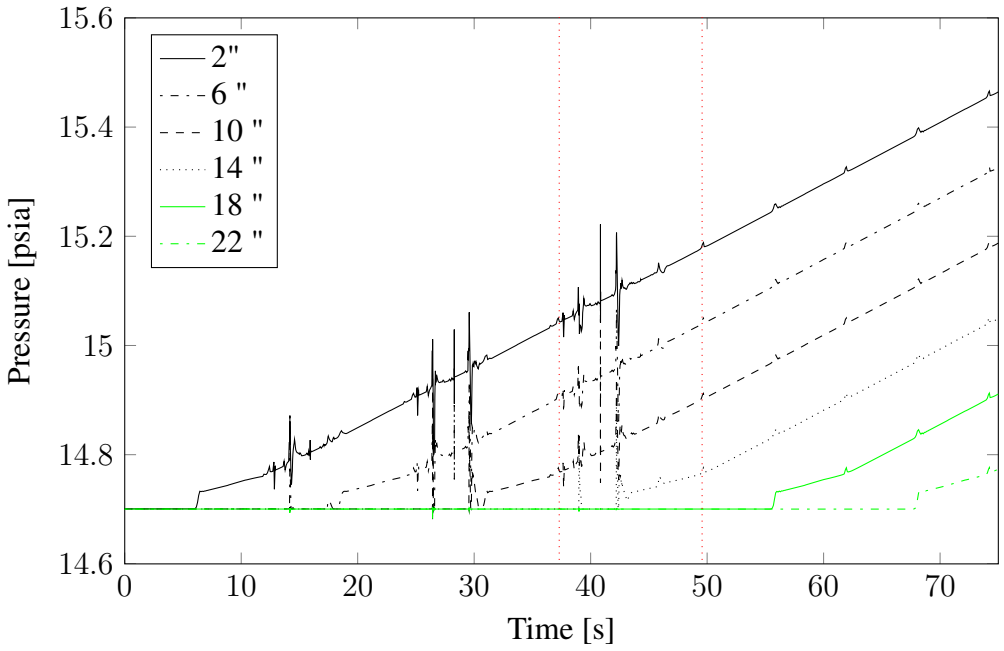

Figure 5.29 Filling problem with the top subchannel in the nonlinear domain.

5.5 Refill Problem

The final test problem was meant to demonstrate the value of using selective nonlinear refinement to simulate a scenario relevant to LWR safety. The model was a simplified pressurized water reactor core with upper-head safety injection that was going through the refill phase of an accident scenario. The intent of this refill problem was to bring together the individual assertions from the previous test problems. In particular, if there were localized nonlinearities, then the use of the domain decomposition algorithm could produce results that not only were more accurate than those produced by the linear solver, but that were also less computationally expensive.

5.5.1 Model

This problem was a model of a generic nuclear reactor vessel. Figure 5.30 contains a schematic of the discrete representation of this problem. The cross-sectional view on the left of Figure 5.30 contains the lateral connection information within a given section. The subchannel view on the right of Figure 5.30 contains the axial connections between sections. There were three sections

in the problem. In Section 1 there was the inlet plenum to the core and the two hemispheres of the downcomer. The downcomers were connected to each other and the inlet plenum, as indicated by the dotted lines. Section 2 contained six subchannels: two representing the active core, one representing the core-bypass flow paths, one representing the reflector, and two representing the downcomer. One of the active core subchannels represented the inner ring of the core, denoted by point C. Another subchannel represented the outer ring of the active core, marked by point D. And, a third subchannel represented the flow bypass paths, as denoted by the small circle adjoining subchannels C and D in the cross-sectional view. The reflector subchannel was situated between the active core subchannels and the downcomer subchannels. The subchannels in Section 2 were hydrodynamically isolated from each other, as indicated by the solid lines. Section 3 represented the upper head of the reactor. In this model, there were transverse flow paths connecting all adjacent subchannels in the upper head, as indicated by the dotted lines in the cross-sectional view. However, the transverse flow paths that connected the reflector subchannels to the two downcomer subchannels were modeled by a specialized set of PDEs meant to represent leakage past the hot-leg nozzle seals, as represented by the red dashed lines in the cross-sectional view. These equations were specialized physical models for leakage paths and used a homogeneous-flow assumption. The use of the leakage path models to represent the connections between the reflector subchannels and the downcomer subchannels allowed for a pressure differential to exist between the upper head of the core and the top of the downcomer. In Section 3 there was also a safety injection inlet at point B, and a pressure connection to containment at point A.

This simulation started after the blowdown phase of a LOCA. The lower plenum and the bottom of the downcomer in Section 1 were initially full of saturated water, as indicated by the blue subchannels. The downcomer in Sections 2 and 3, however, had a mixture of saturated steam and a non-condensable gas representing air, as indicated by the grey subchannels. The mole fraction of the non-condensable gases in these four subchannels started at 25%. The pressure boundary condition at point A was connected to an infinite reservoir of this non-condensable gas-steam mixture at 45 [psia]. Both the four subchannels of the core region in Section 2 and the seven subchannels of the upper head in Section 3 were full of saturated steam, as indicated by the white

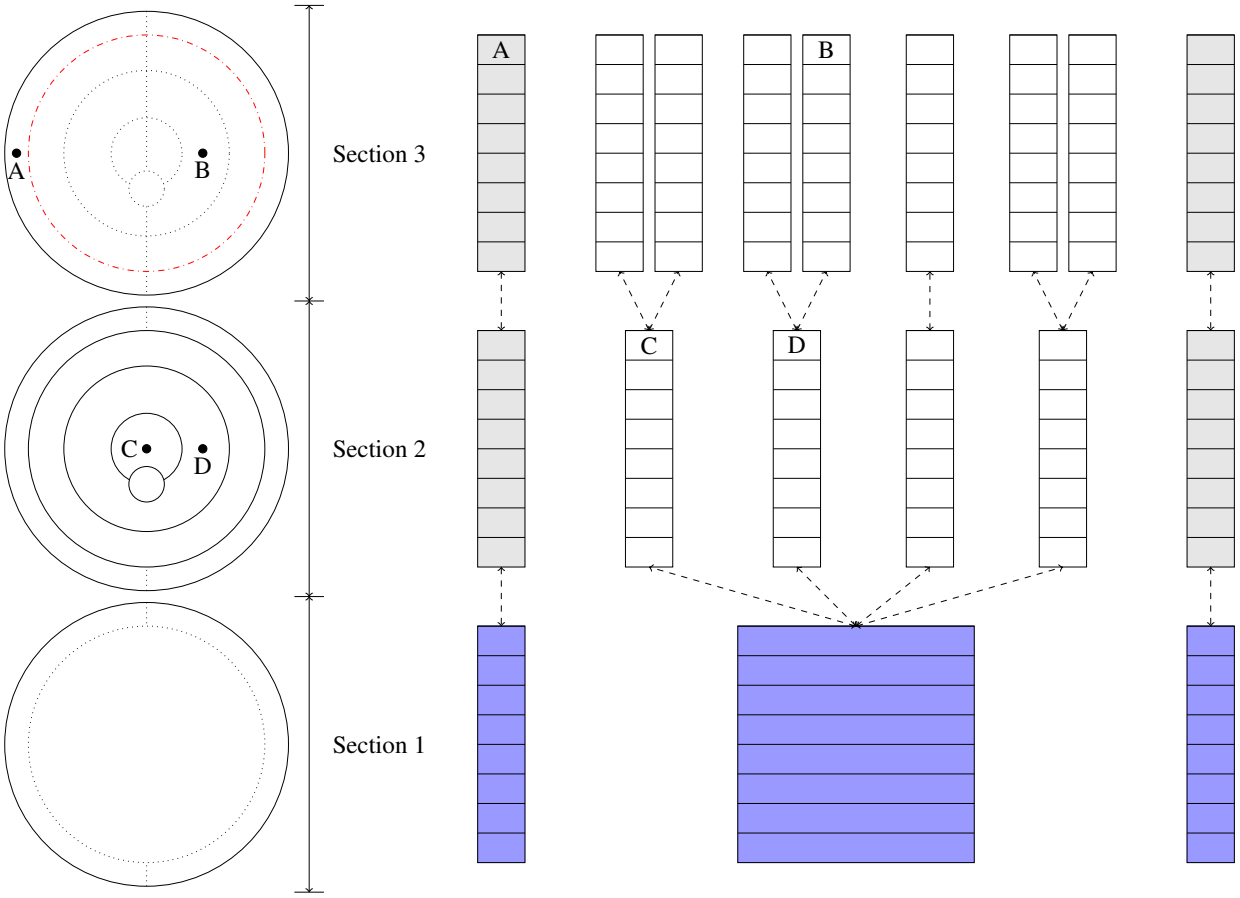

Figure 5.30 COBRA model for the refill problem.

subchannels. Of note, this model did not have solid fuel structures to produce heat in the reactor core. The fuel rods were omitted from this work because the evaluation of the impact of the nonlinear solver on wall heat was beyond the scope of this work. However, to simulate the presence of these fuel rods, two sources, points C & D in Figure 5.30, produced steam within the two subchannels representing the active core. These sources were located near the top of Section 2. The transient behavior of these sources is shown in Figure 5.31. The steam generations rates were chosen such that the steam mass fluxes per core subchannel were equal.

This steam rose into the upper head and condensation was caused by the safety injection inlet flow. This safety injection flow was composed of saturated water at 45 [psia] and was initiated at 20 [s] into the transient. This 20 [s] initialization period allowed the initial conditions to stabilize

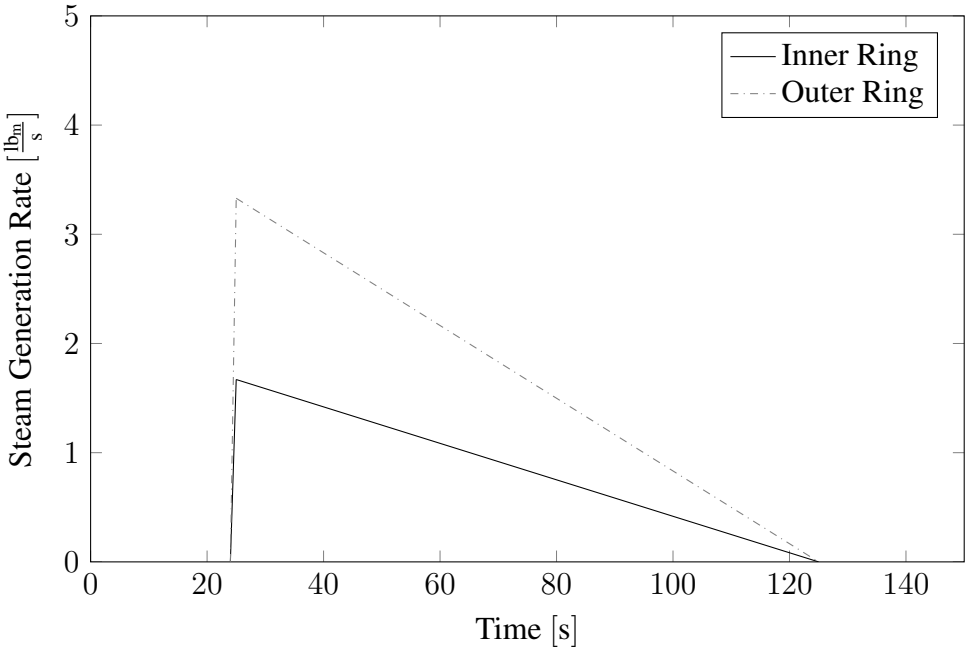

Figure 5.31 Refill problem steam generation rates.

to prevent the initial stagnant conditions from influencing the transient solution. Once initiated this safety injection continued throughout the course of the transient. Figure 5.32 shows this time dependent behavior.

5.5.2 Results

Like previous problems, this simulation was subject to timestep sensitivity studies. The set of timestep sizes analyzed is given by (5.2). For this problem, Δt_0, r_f, and n were 1.28, 2.0, and 11 respectively. These parameters reduced the Δt_{MAX} by three orders of magnitude. Additionally, the nonlinear convergence parameters $k_{\max}$, F_{tol}, and δ_{tol} were 35, 1.0E-6 , and 1.0E-8 , respectively.

The macroscopic transient behavior of the simulation was as follows. Initially, the safety injection nozzle started to condense the steam present in the core, dropping the pressure in the upper head. This drop in pressure started to draw a mixture of liquid, air, and steam into the core from the downcomer. Once the steam generation was initiated, the water seal in the inlet plenum prevented the steam from escaping to containment and the pressure in the upper head increased, causing the

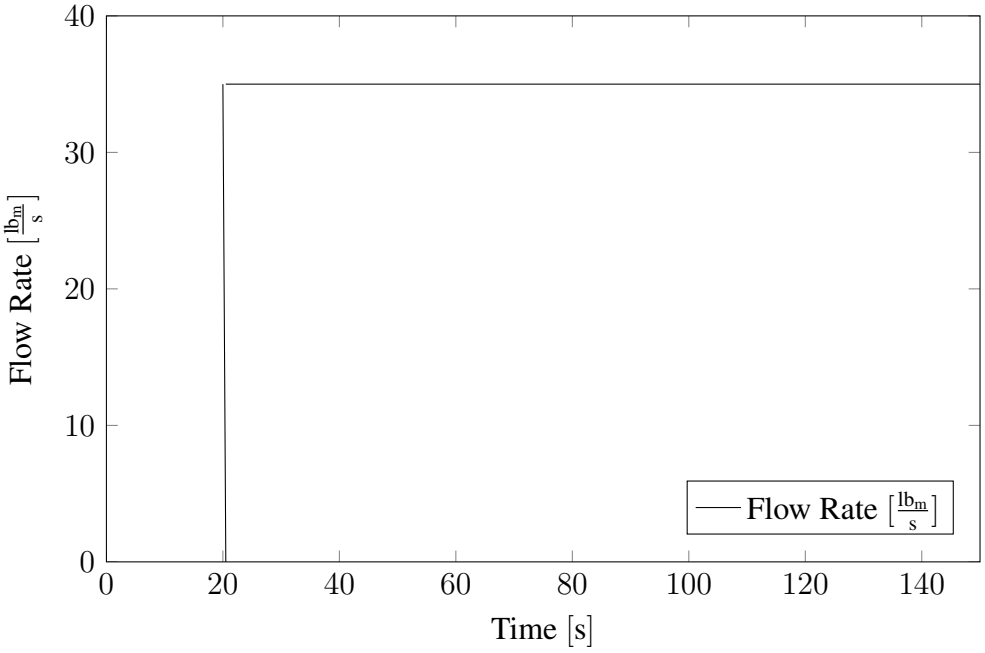

Figure 5.32 Refill problem safety injection rates.

liquid water to be pushed out the bottom of the core and back up into the downcomer. During this period, there were manometer-like oscillations between the downcomer and the core as they filled with water. The pressurization of the upper head prevented the core from filling in parallel with the downcomer. The downcomer filled entirely while the upper head was still pressurized with steam. Concurrent with the filling of the downcomer, the steam flow rate continued to decrease. Eventually, the safety injection water condensed the remaining steam. This caused a rapid drain of the downcomer into the core region. During this rapid draw down, air and steam were drawn back into the core. This slug of gas traveled through the core, collapsing and rising its way into the upper head. At that point, the downcomer started to fill again. Figure 5.33 illustrates this macroscopic behavior as calculated by a simulation with Δt_{MAX} = 2.0E-2 [s].

Since this model simulated a core refill event, the parameter of interest was related to core cooling. The primary variable of engineering interest in this problem was the time-dependent behavior of the condensation in the subchannel containing the safety injection inlet and the net condensation over the course of the transient.

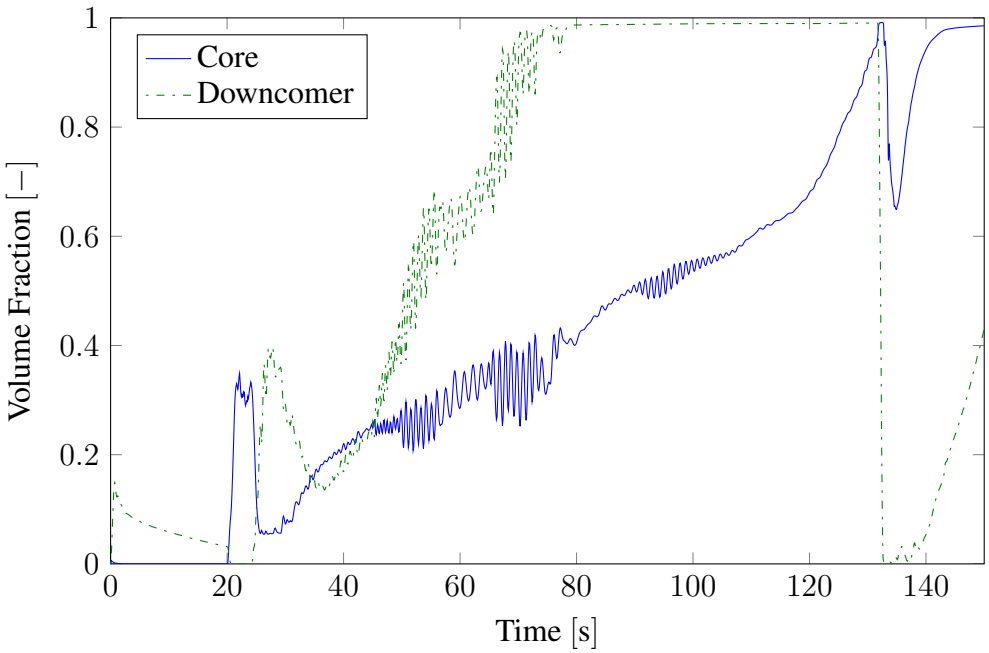

Figure 5.33 Effective volume fraction in downcomer and core.

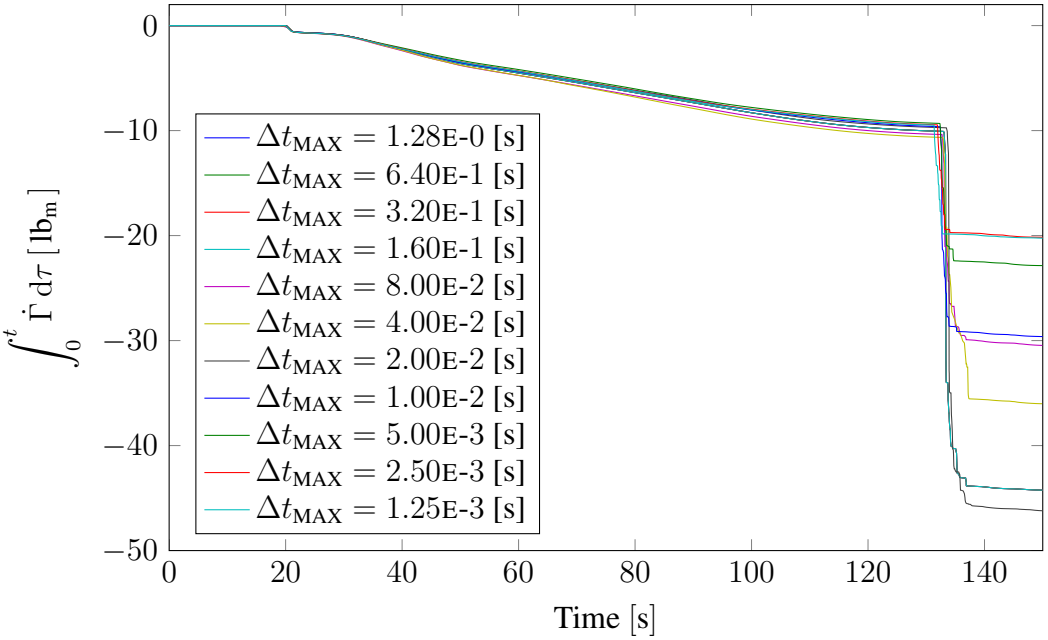

Figure 5.34 Condensation in upper head as calculated by the linear solver.

Figure 5.34 shows the upper-head, cumulative condensation from the linear solver for each different Δt_{MAX}. There was drastic variability in the integrated condensation in the upper head. However, the two solutions with the smallest Δt_{MAX} appeared to have reached a timestep-size insensitive solution. More precisely, there was less variability in integrated condensation between the last two simulations than between any previous two simulations.

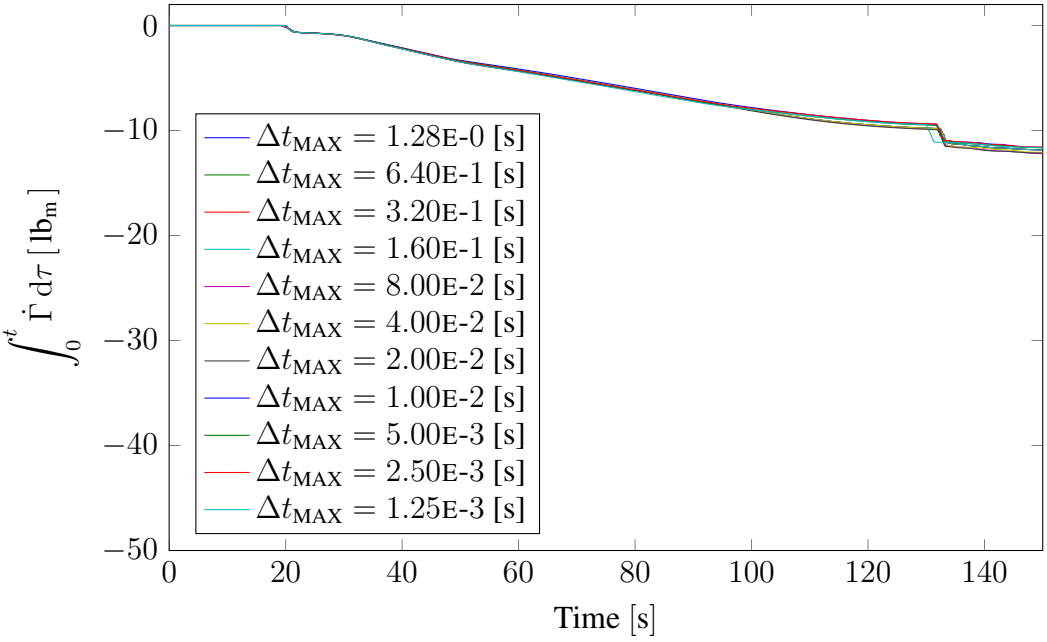

Figure 5.35 Condensation in upper head as calculated by the nonlinear solver.

Next, Figure 5.35 contains the cumulative condensation in the upper head for each of the different Δt_{MAX} using the nonlinear solver for the entire solution. The axes of this graph are the same as those in Figure 5.34. There was a dramatic decrease in both the magnitude and the variability of the quantity as a result of using the nonlinear solver. These results indicate that the nonlinear solver is capable of producing solutions that are not only more consistent than those produced by the linear solver, but that are also different in value.

Using selective nonlinear refinement produced the integrated condensations shown in Figure 5.36. The two subchannels in Section 2 that represent the active core region and the four subchannels in Section 3 above the core region comprised the nonlinear domain in this problem. These results, while not as consistent over the entire range of timesteps analyzed as those produced

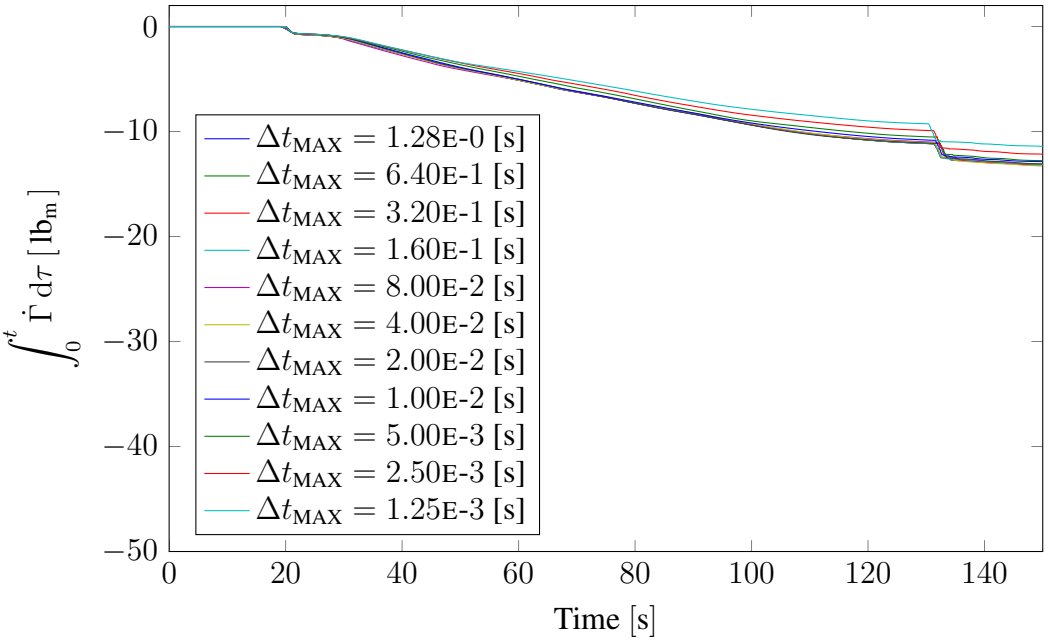

Figure 5.36 Condensation in upper head as calculated with domain decomposition algorithm.

by the global nonlinear solver, were more consistent and more accurate than those produced by the linear solver.

Next, the net condensation for each of the three solution techniques is shown in Figure 5.37.

The linear solver failed to produce results that were comparable with the nonlinear solver even at small timestep sizes. Using the selective nonlinear refinement algorithm produced solutions that were quantitatively and qualitatively consistent with the nonlinear solution.

Next, the computational cost of the nonlinear and the domain decomposition solution methods are compared in Figure 5.38. This figure shows the ratio of the full nonlinear and the domain decomposition run times to the linear solver's run time. While the nonlinear solver is approximately eight times more computationally expensive than the linear solver, the domain decomposition is only three times more computationally expensive than the linear solver. That is a consistent run time savings of greater than 50% while using the domain decomposition algorithm as compared to the full nonlinear solver.

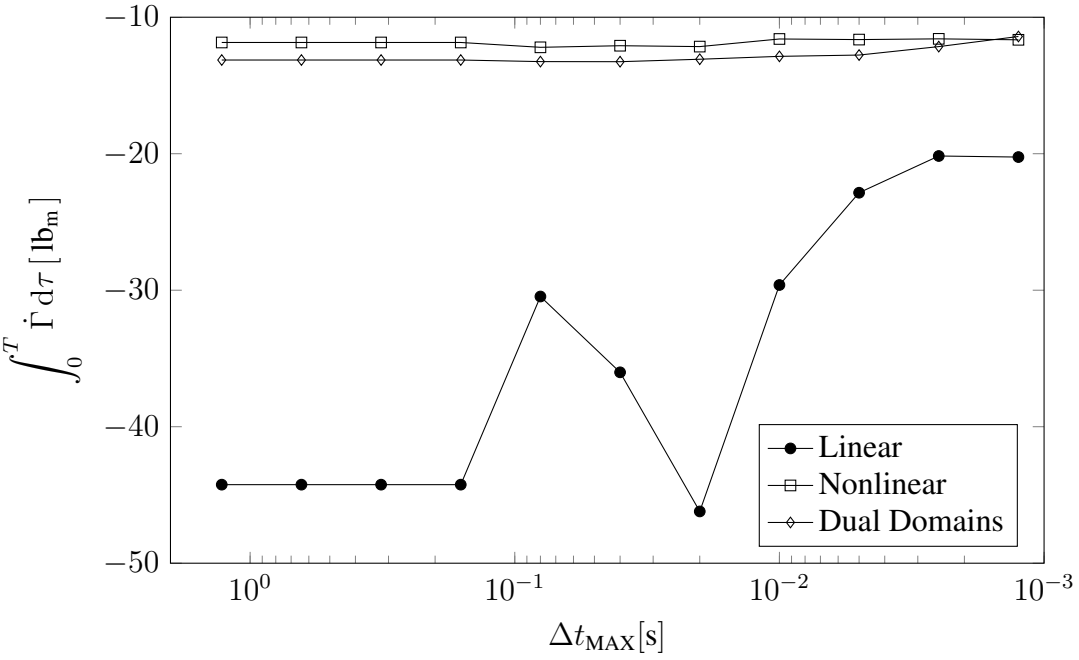

Figure 5.37 Net condensation in upper head for all three algorithms.

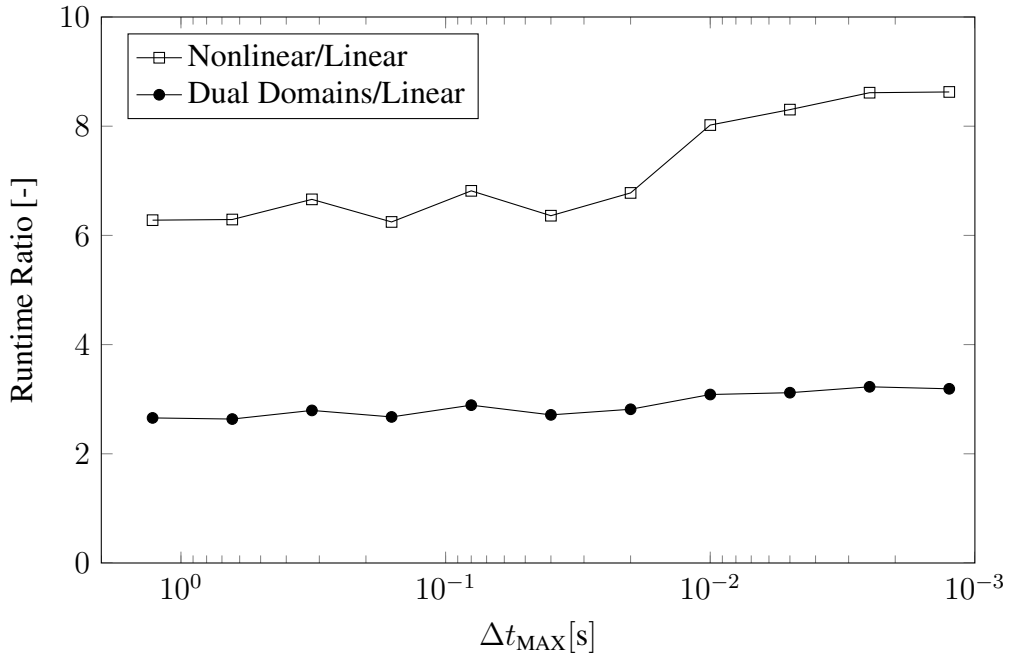

Figure 5.38 Ratios of nonlinear and dual-domain run times to linear run time.

Chapter 6

Concluding Remarks

6.1 Summary of Current Work

This dissertation has focused on developing a novel method for solving the nonlinear PDEs associated with thermal-hydraulic safety analysis software. Traditional methods used in thermal-hydraulic safety analysis software involve solving large systems of nonlinear equations. One class of methods resolves those nonlinearities by using an iterative nonlinear refinement technique. Another class linearizes the nonlinear equations and attempts to minimize the nonlinear truncation error with timestep size refinement. However, this nonlinear truncation error can cause a solution to remain unchanged with reduction in timestep size. This insensitivity of the solution to timestep size refinement is often taken as an indicator of a converged solution. Until the nonlinear truncation error is eliminated by using an iterative solver, temporal convergence cannot be assured. However, the application of a nonlinear solver can be excessively computationally expensive. These two paradigms stand at the opposite ends of a spectrum, and the middle ground had yet to be investigated. This research has developed a means of providing that middle ground: a spatially-selective, nonlinear refinement algorithm.

During the course of this work, the two-phase, three-field software COBRA was converted from a linearized semi-implicit solver to a nonlinearly convergent solver. As part of that development, an operator-based scaling that provides a physically meaningful convergence measure was developed and implemented. This operator-based approach allows for a scaling of the residual that eliminates inherent bias due to the order of magnitude of the terms in the equations; as such, this localized

scale factor normalizes its residual equation to between zero and one. The use of this scale factor is integral to the effective use of the nonlinear solver in subsequent analyses.

Two problems were used to determine the impact of nonlinearities upon the efficacy of the linear and the nonlinear solver. It was found that not resolving the nonlinearities present in a simulation might result in situations where timestep-size insensitivity is not a result of temporal convergence, but rather is an artifact brought about by the degraded order of temporal accuracy caused by linearizing the discrete nonlinear equations. Resolving the nonlinearities at every timestep not only provided a more consistent solution during temporal convergence studies, but also produced different solutions than obtained by taking only a single Newton step during each timestep. However, in simulations where nonlinearities were expected to be low, it was found that the solution produced by the linear solver was as accurate as that produced by the nonlinear solver.

A problem where nonlinearities were isolated to a given portion of the domain was run to determine the impact of resolving or not resolving those localized nonlinearities upon the global solution. When using a linear solver, the solution exhibited nonphysical behavior in all portions of the domain. Use of the nonlinear solver eliminated the nonphysical behavior from the solution. Through the use of the domain decomposition algorithm developed for this work, it was found that eliminating the spatially isolable nonlinearity produced a global solution that more accurately reflected the analytic solution. When the nonlinearities were not part of the nonlinear subdomain, the entire domain still exhibited spurious nonphysical behavior at large timestep sizes. These results emphasized the need to resolve nonlinearities and the usefulness of being able to resolve localized nonlinearities.

Resolving the local nonlinearities required the development of the domain decomposition algorithm. To test the implementation of the domain decomposition algorithm, a geometrically complex problem was developed. When the domain decomposition algorithm was used to subject the nonlinear subdomain to only a single linearization, the solution obtained should analytically match that from the traditional linear solver. By running a sample of simulations with random domain decompositions, it was shown that the obtained solutions matched those of the linear solver

to numerical round-off. These results indicated that the mathematical formulation of the domain decomposition algorithm was accurate and that the implementation was carried out correctly.

As a final evaluation of the domain decomposition algorithm, a simple LWR model was developed. This simulation modeled the refill portion of an accident scenario. When observing engineering parameters of interest, such as condensation from the safety injection nozzle in the upper head, the nonlinear solver demonstrated a more temporally converged solution than the linear solver. The linear solution was also shown to converge to a different solution than that obtained by the nonlinear solver. The domain decomposition algorithm was capable of generating a solution that was more temporally consistent than that obtained by the linear solver, and one that was much closer to solution provided by the nonlinear solver, with only approximately one-third the computational effort. This problem demonstrated that the domain decomposition algorithm might be useful in obtaining nonlinearly and temporally converged safety simulations with less computational cost than traditional nonlinear solvers.

In summary, a nonlinear solver can assist in achieving a temporally converged simulation at larger timestep sizes during timestep sensitivity studies. Unfortunately, nonlinear solvers are computationally expensive when multiple nonlinear iterates are required to resolve the nonlinearities. However, in problems where the spatial location of the nonlinearities can be determined by engineering judgment, the use of the domain decomposition algorithm is warranted. By selecting those areas of the domain where the nonlinearities are expected to be high and subjecting only them to multiple nonlinear iterations, the accuracy of the nonlinear solver may be obtained without its associated computational cost.

6.2 Areas for Future Research

During this work several opportunities for follow-on research presented themselves. These research opportunities include the development and implementation of a spatially and temporally adaptive version of the domain decomposition algorithm, the evaluation of theoretical computational costs for the domain decomposition, the investigation of the interaction of the nonlinear solver and the solid structures within COBRA, and the ability to dynamically switch governing

equations for a given spatial location during the transient. The following section will outline each of the above-mentioned avenues of research in turn.

The current research produced a domain decomposition framework that requires engineering judgment to select portions of the domain where nonlinear convergence may require multiple Newton steps. A possible extension of this work would be the development of an algorithm that identifies areas where additional Newton steps would be advantageous. Since, over the course of a given simulation, the spatial location with the greatest nonlinearities may shift, being able to change the nonlinear domain as the transient progressed would provide an additional reduction in the computational cost. Once a viable determination of when and where additional nonlinear iterates might be advantageous can be made, a method for dynamically generating the pressure matrices would need to be developed. Currently, the matrices are preallocated and of fixed size through the simulation. If the two domains were to shift during the course of the transient, the pressure matrices and the corresponding ordinals of continuity volumes would need to change to reflect the new domains.

The current operator-based scaling method has conflicting implementations for determining the magnitude of the divergence operators. For the momentum equations the net divergence is used as the operator for the scale factor, while the mass and energy equations consider each discrete portion of the surface integral to be an independent operator. Studies looking at the efficiency of the two different interpretations are needed to clarify which is the better option.

Another topic requiring additional research is the computational cost of the domain decomposition algorithm in the presence of spatially dispersed nonlinearities. The computational cost of using the domain decomposition algorithm is a function of the number of domain interfaces, the average number of Newton steps required, and the relative size of the two domains. There would need to be an investigation into when it becomes computationally less expensive to solve the global domain with the nonlinear solver as opposed to solving the dual-domain problem.

Additionally, this work was centered on the hydrodynamics of the reactor core, without consideration of the fluid-wall heat transfer. In the presence of fluid-wall heat transfer, the use of the nonlinear solver produces non-convergent behavior in some test problems. A detailed study and

analysis of the interaction of the explicit heat transfer and the iterative nonlinear solver for the hydrodynamics needs to be conducted.

Finally, the COBRA software possesses the ability to switch governing differential equations for the momentum equations if certain criteria are met. The primary use of this feature is the implementation of a counter-current flow limit boundary condition, which switch governing equations based upon flow regimes. During the course of this work, the interaction between the iterative solver and the decision of when to switch equations was found to produce excessive equation switching in some models. The reasons for this are as of yet unknown, and additional research is required to identify the root cause.

List of References

[1] *10 C. F. R. §50.46*.

[2] D. L. Aumiller, E. T. Tomlinson, and R. C. Bauer. A coupled RELAP5-3D/CFD methodology with a proof-of-principle calculation. *Nuclear Engineering and Design*, 205 (1-2):83 – 90, 2001. ISSN 0029-5493. doi: 10.1016/S0029-5493(00)00370-8.

[3] D. L. Aumiller, E. T. Tomlinson, and R. C. Bauer. Incorporation of COBRA-TF in an integrated code system with RELAP5-3D using semi-implicit coupling. In *2002 RELAP5 International Users Seminar*, September 2002.

[4] D. L. Aumiller, G. W. Swartele, J. W. Lane, F. X. Buschman, and M. J. Meholic. Development of verification testing capabilities for safety codes. In *The 15^{th} International Topical Meeting on Nuclear Reactor Thermal-Hydraulics, NURETH-15*, number 145, Pisa, Italy, May 2013.

[5] M. Avramova, D. Cuervo, and K. Ivanov. Improvements and applications of COBRA-TF for stand-alone and coupled LWR safety analyses. In *PHYSOR-2006, ANS Topical Meeting on Reactor Physics*. Canadian Nuclear Society, September 2006.

[6] F. Barre and M. Bernard. The CATHARE code strategy and assessment. *Nuclear Engineering and Design*, 124(3):257 – 284, 1990. ISSN 0029-5493. doi: 10.1016/0029-5493(90)90296-A.

[7] D. Bestion. The phase appearance and disapperance in the CATHARE code. In *Trends in Numerical and Physical Modeling for Industrial Multiphase Flows*, September 2000.

[8] X. C. Cai. Nonlinear overlapping domain decomposition methods. In M. Bercovier, M. J. Gander, R. Kornhuber, and O. Widlund, editors, *Domain Decomposition Methods in Science and Engineering XVIII*, volume 70 of *Lecture Notes In Computational Science And Engineering*, pages 217–224. Springer Heidelberg, 2009. doi: 10.1007/978-3-642-02677-5.

[9] X. C. Cai and D. Keyes. Nonlinearly preconditioned inexact newton algorithms. *SIAM Journal on Scientific Computing*, 24(1):183–200, 2002. doi: 10.1137/S106482750037620X.

[10] X. C. Cai and X. Li. Inexact newton methods with restricted additive schwarz based nonlinear elimination for problems with high local nonlinearity. *SIAM Journal on Scientific Computing*, 33(2):746–762, 2011. doi: 10.1137/080736272.

[11] T. Chan and K. Jackson. Nonlinearly preconditioned krylov subspace methods for discrete newton algorithms. *SIAM Journal on Scientific and Statistical Computing*, 5(3):533–542, 1984. doi: 10.1137/0905039.

[12] J. E. Dennis and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 1996. doi: 10.1137/1.9781611971200.

[13] P. Deuflhard. *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms*, volume 35 of *Springer Series in Computational Mathematics*. Springer-Verlag Berlin Heidelberg, 2nd edition, 2006.

[14] D. A. Drew and S. L. Passman. *Theory of Multicomponent Fluids*, volume 135 of *Applied Mathematical Sciences*. Springer-Verlag New York, Inc. 175 Fifth Avenue New York, NY 10010 USA, 1st edition, 1998.

[15] M. Dryja and W. Hackbusch. On the nonlinear domain decomposition method. *BIT*, 37(2): 296–311, 1997.

[16] P. Emonot, A. Souyri, J. L. Gandrille, and F. Barre. CATHARE-3: A new system code for thermal-hydraulics in the context of the neptune project. *Nuclear Engineering and Design*, 241(11):4476 – 4481, 2011. ISSN 0029-5493. doi: 10.1016/j.nucengdes.2011.04.049. 13th International Topical Meeting on Nuclear Reactor Thermal Hydraulics (NURETH-13).

[17] C. Frepoli, J. H. Mahaffy, and K. Ohkawa. Notes on the implementation of a fully-implicit numerical scheme for a two-phase three-field flow model. *Nuclear Engineering and Design*, 225(2-3):191 – 217, 2003. ISSN 0029-5493. doi: 10.1016/S0029-5493(03)00159-6.

[18] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM: Parallel Virtual Machine A User's Guide and Tutorial for Networked Parallel Computing*. Massachusetts Institute of Technology, Cambridge, Massachusetts, 1994.

[19] M. Ishii and K. Mishima. Two-fluid model and hydrodynamic constitutive relations. *Nuclear Engineering and Design*, 82(2-3):107 – 126, 1984. ISSN 0029-5493. doi: 10.1016/0029-5493(84)90207-3.

[20] J. J. Jeong, H. Y. Yoon, I. K. Park, H. K. Cho, and J. Kim. A semi-implicit numerical scheme for transient two-phase flows on unstructured grids. *Nuclear Engineering and Design*, 238 (12):3403 – 3412, 2008. ISSN 0029-5493. doi: 10.1016/j.nucengdes.2008.08.017.

[21] D. A. Knoll, W. J. Rider, and G. L. Olson. Nonlinear convergence, accuracy, and time step control in nonequilibrium radiation diffusion. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 70(1):25 – 36, 2001. ISSN 0022-4073. doi: 10.1016/S0022-4073(00)00112-6.

[22] R. T. Lahey, B. S. Shiralkar, and D. W. Radcliffe. Two-phase flow and heat transfer in multirod geometries: Subchannel and pressure drop measurements in a nine-rod bundle for diabatic and adiabatic conditions. AEC Research and Development Report GEAP-13049, Atomic Power Equipment Department, General Electric, Atomic Power Equipment Department General Electric Company San Jose, California 95125, March 1970.

[23] P. Lanzkron, D. Rose, and J. Wilkes. An analysis of approximate nonlinear elimination. *SIAM Journal on Scientific Computing*, 17(2):538–559, 1996. doi: 10.1137/S106482759325154X.

[24] R. J. LeVeque. *Finite Volume Methods for Hyperbolic Problems*. Cambridge Texts in Applied Mathematics. Cambridge University Press, 32 Avenue of the Americas New York, NY 10013-2473 USA, 2002.

[25] R. J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations*. Society for Industrial and Applied Mathematics, 2007.

[26] X. S. Li, J. W. Demmel, J. R. Gilbert, L. Grigori, M. Shao, and I. Yamazaki. SuperLU user's guide. LBNL 44289, Ernest Orlando Lawrence Berkeley National Laboratory, September 1999.

[27] D. R. Liles and Wm. H. Reed. A semi-implicit method for two-phase fluid dynamics. *Journal of Computational Physics*, 26(3):390 – 407, 1978. ISSN 0021-9991. doi: 10.1016/0021-9991(78)90077-3.

[28] J. H. Mahaffy. A stability-enhancing two-step method for fluid flow calculations. *Journal of Computational Physics*, 46(3):329 – 341, 1982. ISSN 0021-9991. doi: 10.1016/0021-9991(82)90019-5.

[29] J. H. Mahaffy. Numerics of codes: stability, diffusion, and convergence. *Nuclear Engineering and Design*, 145:131 – 145, 1993. ISSN 0029-5493. doi: 10.1016/0029-5493(93)90063-F.

[30] Y. Makihara. Use and development of coupled computer codes for the analysis of accidents at nuclear power plants: proceedings of a technical meeting held in Vienna, 26-28 November 2003. Technical report, International Atomic Energy Agency, 2003.

[31] P. R. McHugh. An investigation of newton-krylov algorithms for solving incompressible and low mach number compressible fluid flow and heat transfer problems using finite volume discretization. Technical Report INEL-95/0118, Idaho National Engineering Laboratory, 1995.

[32] M. Paraschivoiu, X. Cai, M. Sarkis, D. P. Young, and D. E. Keyes. Multi-domain multi-model formulation for compressible flows: Conservative interface coupling and parallel implicit solvers for 3D unstructured meshes. In *Unstructured Meshes, AIAA Paper 99-0784, American Institute of Aeronautics and Astronautics*, pages 99–0784, 1999.

[33] I.K. Park, H.K. Cho, H.Y. Yoon, and J.J. Jeong. Numerical effects of the semi-conservative form of momentum equations for multi-dimensional two-phase flows. *Nuclear Engineering and Design*, 239(11):2365 – 2371, 2009. ISSN 0029-5493. doi: DOI:10.1016/j.nucengdes.2009.06.011.

[34] J. C. Ragusa and V. S. Mahadevan. Consistent and accurate schemes for coupled neutronics thermal-hydraulics reactor analysis. *Nuclear Engineering and Design*, 239(3):566 – 579, 2009. ISSN 0029-5493. doi: DOI:10.1016/j.nucengdes.2008.11.006.

[35] S. B. Rodriguez. Using the coupled MELCOR-RELAP5 codes for simulation of the Edward's Pipe. Technical report, Sandia National Laboratories, 2002.

[36] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite systems. *Electronic Transactions on Numerical Analysis*, 23:158–179, 2006.

[37] O. Schenk, A. Wächter, and M. Hagemann. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Computational Optimization and Applications*, 36(2-3):321–341, April 2007.

[38] H. B. Stewart. Fractional step methods for thermohydraulic calculation. *Journal of Computational Physics*, 40(1):77 – 90, 1981. ISSN 0021-9991. doi: 10.1016/0021-9991(81)90200-X.

[39] H. B. Stewart and B. Wendroff. Two-phase flow: Models and methods. *Journal of Computational Physics*, 56(3):363 – 409, 1984. ISSN 0021-9991. doi: 10.1016/0021-9991(84)90103-7.

[40] R. M. Summers, R. K. Cole, R. C. Smith, D.S. Stuart, S. L. Thompson, S. A. Hodge, C. R. Hyman, and R. L. Sanders. MELCOR computer code manuals, 1994.

[41] J. C. Tannehill, D. A. Anderson, and R. H. Pletcher. *Computational Fluid Mechanics and Heat Transfer*. Taylor & Francis Group, 2nd edition, 1997.

[42] The RELAP5-3D Code Development Team. RELAP5-3D code manual volume I: Code structure, system models, and solution methods. INEEL-EXT 98-00834, Idaho National Laboratory, Idaho Falls, Idaho, June 2012.

[43] M. J. Thurgood, J. Kelly, T. E. Guidotti, R. J. Kohrt, and K. R. Crowell. COBRA/TRAC - a thermal-hydraulics code for transient analysis of nuclear reactor vessels and primary coolant systems, 1983.

[44] N. E. Todreas and M. S. Kazimi. *Nuclear Systems Volume I: Thermal Hydraulic Fundamentals*. Taylor & Francis Group, 6000 Broken Sound Parkway NW, Suite 300 Boca Raton, FL 33487-2742, 2nd edition, 2011.

[45] J. A. Trapp and R. A. Riemke. A nearly-implicit hydrodynamic numerical scheme for two-phase flows. *Journal of Computational Physics*, 66(1):62 – 82, 1986. ISSN 0021-9991. doi: 10.1016/0021-9991(86)90054-9.

[46] *TRACE V5.0 Theory Manual Field Equations, Solution Manual, and Physical Models*. United States Nuclear Regulatory Commission, Washington, DC.

[47] W. L. Weaver, D. L. Aumiller, and E. T. Tomlinson. A generic semi-implicit coupling methodology for use in RELAP5-3D. *Nuclear Engineering and Design*, 211(1):13 – 26, 2002. ISSN 0029-5493. doi: 10.1016/S0029-5493(01)00422-8.