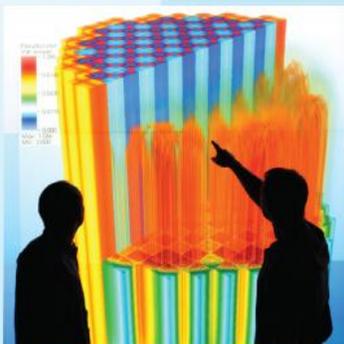


Power updates  
and plant life extension

CASL-I-2012-0155-000/c



Engineering design  
and analysis



L3:THM.CFD.P5.01

Mark Christon

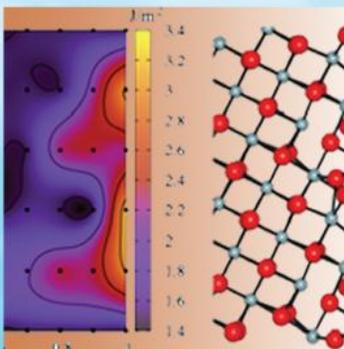
LANL

Completed: 9/29/2012

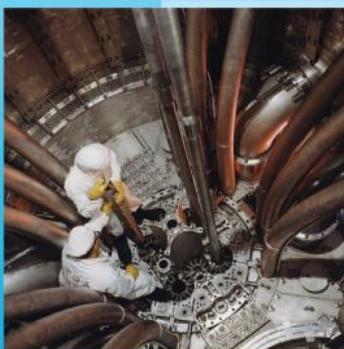
Science-enabling  
high performance  
computing



Fundamental science



Plant operational data



U.S. DEPARTMENT OF  
**ENERGY**

**Nuclear Energy**

INL/EXT-12-27187

# **Solution Algorithms for Multi-Fluid-Flow Averaged Equations**

**Robert Nourgaliev, Mark Christon**

**September 20, 2012**



*This Page is Intentionally Left Blank*

# **Solution Algorithms for Multi-Fluid-Flow Averaged Equations**

ROBERT NOURGALIEV\*, MARK CHRISTON\*\*

**\*Nuclear Science & Technology  
Idaho National Laboratory  
P.O. Box 1625, Idaho Falls, ID 83415-3840, USA**

**\*\*Computational Physics Group  
Computer, Computational and Statistical Sciences Division  
Los Alamos National Laboratory  
Los Alamos, NM 87545, USA**



**INL/EXT-12-27187 (151 p.)  
September 20, 2012**

*This Page is Intentionally Left Blank*

# Abstract

**T**HIS document summarizes the state-of-the-art algorithms for solving effective-field (averaged) multi-fluid equations for multiphase flows, used in reactor-safety codes. The main purpose is to outline and analyze all the major solution algorithms developed in the past 40 years, widely used in legacy reactor thermal-hydraulics codes (RELAP5, TRAC/TRACE, CATHARE) and commercial CFD codes (SIMPLE-based). This will provide the base for implementation of the efficient solution strategy in Hydra-TH code, that is being developed under the Consortium for Advanced Simulation of Light-Water Reactors (CASL) in Los Alamos National Laboratory.

*This Page is Intentionally Left Blank*

## Acknowledgements

**T**HIS work has been authored by Battelle Energy Alliance, LLC under contract No. DE-AC07-05ID14517 (INL/EXT-12-27187) with the U.S. Department of Energy. The United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manual, or allow others to do so, for United States Government purposes.

*This Page is Intentionally Left Blank*

# Contents

<b>Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>Contents</b>	<b>vii</b>
<b>1 Introduction</b>	<b>2</b>
<b>2 Governing Equations</b>	<b>6</b>
2.1 Ensemble-Averaging . . . . .	6
2.2 Balance Equations . . . . .	7
2.3 Averaged variables . . . . .	8
2.4 Closures . . . . .	9
2.5 Entropy inequality . . . . .	12
2.6 Two-fluid equations . . . . .	13
2.6.1 Bulk pressure difference and interfacial forces . . . . .	15
2.6.2 Variations . . . . .	16
<b>3 Fully-compressible, Explicit Algorithms</b>	<b>20</b>
3.1 Explicit Runge-Kutta scheme . . . . .	20
3.2 Primitive variables . . . . .	21
3.3 Remarks . . . . .	22
<b>4 Operator-Splitting Algorithms</b>	<b>24</b>
4.1 Semi-Implicit Algorithms . . . . .	25
4.1.1 Predictor discretization . . . . .	26
4.1.2 Pressure-Helmholtz equation . . . . .	28
4.1.3 Corrector . . . . .	31
4.1.4 Remarks . . . . .	32
4.2 Fractional step based variations . . . . .	34

4.2.1	Stability-Enhancing Two-Step method (SETS)	34
4.2.2	Nearly-implicit algorithm	37
4.2.3	Remarks	40
4.3	Incremental form of Semi-Implicit-based algorithms	41
4.3.1	Linearization of sources	42
4.3.2	Linearization of conservation laws	43
4.3.3	Diagonalization of momentum equations	48
4.3.4	Mass P'HE	52
4.3.5	Energy P'HE	55
4.3.6	Pressure correction P'HE	60
4.3.7	Mass and energy stabilizers	64
4.3.8	Outline of the fractional step version	68
4.3.9	P'HE for single-fluid formulation	70
4.3.10	P'HE for 2-fluid formulation	73
<b>5</b>	<b>Segregated (Picard-Iteration) Algorithms</b>	<b>84</b>
5.1	SIMPLE-based algorithms	84
5.1.1	Mass Conservation-Based Algorithms (MCBA)	86
5.1.2	Geometric Conservation-Based Algorithms (GCBA)	96
5.2	“Semi-implicit”-based algorithm	101
5.3	NPHASE Algorithm	102
<b>6</b>	<b>Fully-Implicit, Newton-Based Algorithms</b>	<b>104</b>
6.1	Implicit time discretizations	104
6.1.1	Backward Euler	105
6.1.2	BDF	105
6.1.3	Crank-Nicholson	105
6.1.4	ESDIRK	105
6.2	Newton method	106
6.3	CATHARE Algorithm	110
6.4	Fully-compressible, Fully-implicit, JFNK-based	111
6.4.1	Krylov subspace iterations (GMRES)	112
6.4.2	Jacobian-free implementation	112
6.4.3	Inexact Newton	113
6.5	Preconditioning	114
6.5.1	Matrix-based preconditioning	115
6.5.2	Physics-(process)-based preconditioning (PBP)	115

<b>CONTENTS</b>	<b>IX</b>
<b>7 Phase appearance and disappearance</b>	<b>118</b>
7.1 CATHARE strategy . . . . .	118
7.1.1 Void fraction bounding and thermal conditioning . . . . .	119
7.1.2 Velocity conditioning . . . . .	122
7.1.3 Discussion . . . . .	122
7.2 Paillère at al. treatment . . . . .	123
<b>8 Concluding Remarks</b>	<b>126</b>
 <b>APPENDICES</b>	 <b>129</b>
<b>A Cartesian Vector Calculus</b>	<b>130</b>
<b>B SNES JFNK-PBP pseudo-code example</b>	<b>132</b>
B.1 Configuration . . . . .	133
B.1.1 Configure SNES . . . . .	133
B.1.2 Configure KSP . . . . .	134
B.2 Execution stage . . . . .	136
B.3 Residual evaluation routine: <b>FormFunction</b> . . . . .	137
B.4 Preconditioning setup routine: <b>Precond_SetUp</b> . . . . .	138
B.5 Preconditioning routine: <b>PBP_precondition</b> . . . . .	138
<b>Bibliography</b>	<b>146</b>
<b>Index</b>	<b>151</b>

*This Page is Intentionally Left Blank*

# List of Figures

1.1	On relation of the discussed algorithms. . . . .	3
5.1	On basic concept of segregated algorithms. . . . .	85

*This Page is Intentionally Left Blank*

# Chapter 1

## Introduction

**I**N this document, we review the solution strategies for solving the system of governing equations for effective-field formulation of multi-fluid/multi-phase flows. These equations were developed in late 60s [Del68, IH06, Nig90, DP99], and are widely used in nuclear reactor safety and thermalhydraulics analysis, as well as for other multiphase-flow applications in oil and aerospace industries, geoscience, advanced weaponry, etc. The main motivation for this study is to develop the best solution algorithm for implementation  $N$ -fluid model in Hydra-TH code, being developed at Los Alamos National Laboratory under the auspices of the Consortium for Advanced Simulation of Light-Water Reactors (CASL) for thermal-hydraulics applications in nuclear industry.

The manuscript is organized as follows.

First, we summarize the governing equations for  $N$ -fluid multiphase flows, in Chapter 2. The focus is placed on the PDE part of governing equations, developed using homogenization with ensemble-averaging. This is the most numerically challenging part of the model, which directly affect the choice of the solution strategy, including space and time discretization. We presume that the set of closure models is defined, and we leave the discussion of the proper constitutive physics to other future study.

We divide the algorithms on four major distinct groups, Figure 1.1:

- I. Explicit algorithms (Chapter 3),
- II. Operator-Splitting algorithms (Chapter 4),

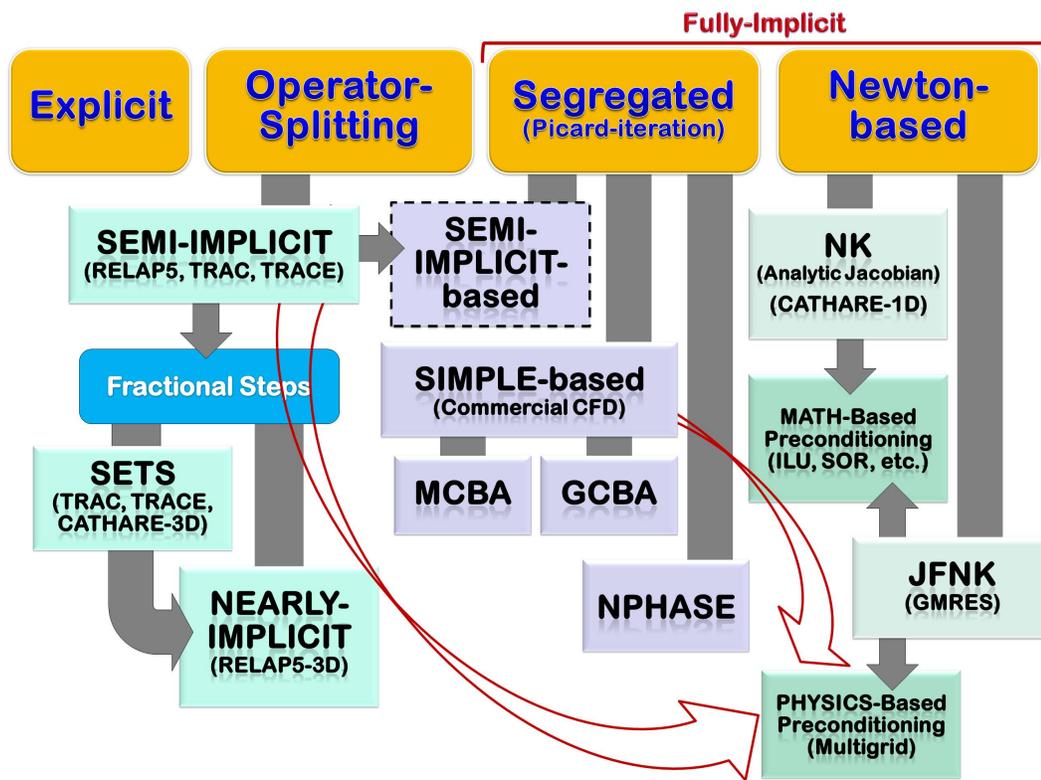


Fig. 1.1 : On relation of the discussed algorithms.

III. Segregated (fully-implicit) algorithms (Chapter 5), and

IV. Newton-based algorithms (Chapter 6).

The algorithms discussed in Chapter 4 (operator-splitting, “semi-implicit”) are the closest to what is currently used in single-phase models of Hydra-TH, and this is our first choice for multi-phase flow implementation. This provides very important building block for fully-implicit algorithms within either Picard-iteration (Chapter 5) or Newton-based (Chapter 6) solution strategies. The innovative part is the development of incremental form of the “semi-implicit” method, which introduced in Section 4.3. Here, we provide the detail derivation, as well as preliminary analysis for solvability and well-posedness.

Finally, we discuss and review another challenging issue – the solution strategy for phase appearance and disappearance, in Chapter 7. While no definite and fully-satisfactory strategy for this problem currently exists, the provided discussion and analysis is an important step in the devising the one. This will be the focus of future study.

*This Page is Intentionally Left Blank*

# Chapter 2

## Governing Equations

IN this chapter, we summarize the governing averaged equations for  $N$ -fluid formulation. We will follow the ensemble-averaging procedure described by Drew and Passman in [DP99]. The other homogenization (time- [IH06], volume- [Del68, Nig90] averaging) techniques result in the similar set of governing equations.

### 2.1 Ensemble-Averaging

The average variables are defined using the following ensemble-averaging operator

$$\bar{\Phi}(\mathbf{x}, t) = \int_{\mathcal{E}} \Phi(\mathbf{x}, t; \mu) dm(\mu) \quad (2.1)$$

where  $dm(\cdot)$  is the density for the measure (probability) on the set of all processes  $\mathcal{E}$ . Another important definition is the *component indicator function* (or *characteristic function*),  $\mathcal{X}_k(\mathbf{x}, t; \mu)$ :

$$\mathcal{X}_k(\mathbf{x}, t; \mu) = \begin{cases} 1 & \text{if } \mathbf{x} \in k \text{ in realization } \mu \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

## 2.2 Balance Equations

Ensemble-averaged balance equations for multi-component fluids ( $k=0, \dots, K-1$ ) are<sup>1,2</sup> [DP99]:

**Mass:**

$$\frac{\partial \alpha_k \bar{\rho}_k}{\partial t} + \nabla \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k) = \boxed{\Gamma_k} \quad (2.3)$$

**Momentum:**

$$\begin{aligned} \frac{\partial \alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k}{\partial t} + \nabla \cdot (\alpha_k [\bar{\rho}_k \tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k + \bar{p}_k]) &= \alpha_k \bar{\rho}_k \boxed{\tilde{\mathbf{b}}_k} + \boxed{\mathbf{v}_{ki}^m} \boxed{\Gamma_k} + \\ &+ \nabla \cdot \left( \alpha_k \left[ \boxed{\bar{\boldsymbol{\tau}}_k} + \boxed{\mathbf{T}_k^{Re}} \right] \right) \quad (2.4) \\ &+ \left( \boxed{p_{ki}} - \boxed{\tau_{ki}} \right) \nabla \alpha_k + \boxed{\mathbf{M}'_k} \end{aligned}$$

**Total energy:**

$$\begin{aligned} \frac{\partial}{\partial t} (\alpha_k \bar{\rho}_k \tilde{e}_k) + \nabla \cdot (\alpha_k [\bar{\rho}_k \tilde{e}_k + \bar{p}_k] \tilde{\mathbf{v}}_k) &= \alpha_k \bar{\rho}_k \left( \boxed{\tilde{r}_k} + \boxed{\tilde{\mathbf{b}}_k} \cdot \tilde{\mathbf{v}}_k \right) + \\ &\nabla \cdot \left( \alpha_k \left[ \tilde{\mathbf{v}}_k \left( \boxed{\bar{\boldsymbol{\tau}}_k} + \boxed{\mathbf{T}_k^{Re}} \right) - \boxed{\bar{\mathbf{q}}_k} - \boxed{\mathbf{q}_k^{Re}} \right] \right) \quad (2.5) \\ &+ \boxed{\Gamma_k} \left( \boxed{u_{ki}} + \frac{\left( \boxed{v_{ki}^e} \right)^2}{2} \right) + \boxed{E_k} \\ &+ \boxed{W'_k} + \boxed{\mathbf{M}'_k} \cdot \tilde{\mathbf{v}}_k + \left( \boxed{p_{ki}} - \boxed{\tau_{ki}} \right) \tilde{\mathbf{v}}_k \cdot \nabla \alpha_k \end{aligned}$$

**Fluctuation kinetic energy:**

$$\begin{aligned} \frac{\partial (\alpha_k \bar{\rho}_k \tilde{k}_k)}{\partial t} + \nabla \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k \tilde{k}_k) &= \alpha_k \boxed{\mathbf{T}_k^{Re}} : \nabla \tilde{\mathbf{v}}_k - \nabla \cdot \alpha_k \left( \boxed{\mathbf{q}_k^K} + \boxed{\mathbf{q}_k^T} \right) + \quad (2.6) \\ &+ \frac{1}{2} \left[ \left( \boxed{v_{ki}^e} \right)^2 + (|\tilde{\mathbf{v}}_k|)^2 - \tilde{\mathbf{v}}_k \cdot \boxed{\mathbf{v}_{ki}^m} \right] \boxed{\Gamma_k} + \boxed{W'_k} + \alpha_k \left( \boxed{D_k} - \boxed{P_k} \right) \end{aligned}$$

Complemented by equations of state for each phase  $\mathcal{EOS}_k$  and constitutive physics for terms shown in boxes (see Section 2.4), these balance equations define a complete closed form for multi-fluid dynamics of inter-penetrating continua.

<sup>1</sup>The terms requiring closure are shown in boxes.

<sup>2</sup>On vector notation, see Chapter A.

## 2.3 Averaged variables

The averaged variables in eqs.(2.3)-(2.6) are defined as follows.

**Volume fraction:**

$$\alpha_k = \overline{\mathcal{X}'_k} \quad (2.7)$$

**Phasic density:**

$$\bar{\rho}_k = \frac{\overline{\mathcal{X}'_k \rho}}{\alpha_k} \quad (2.8)$$

**Phasic velocity:**

$$\tilde{\mathbf{v}}_k = \frac{\overline{\mathcal{X}'_k \rho \mathbf{v}}}{\alpha_k \bar{\rho}_k} \quad (2.9)$$

**Phasic specific internal energy:**

$$\tilde{u}_k = \frac{\overline{\mathcal{X}'_k \rho u}}{\alpha_k \bar{\rho}_k} \quad (2.10)$$

**Phasic fluctuation (Reynolds) kinetic energy:**

$$\tilde{\kappa}_k = \frac{\overline{\mathcal{X}'_k \rho |\mathbf{v}'_k|^2}}{2\alpha_k \bar{\rho}_k}, \quad \mathbf{v}'_k = \mathbf{v} - \tilde{\mathbf{v}}_k \quad (2.11)$$

where by  $(\tilde{\cdot})$  we denote *mass-weighted* (or Fevré) averaged quantities.

**Phasic total energy:**

$$\tilde{e}_k = \tilde{u}_k + \frac{|\tilde{\mathbf{v}}_k|^2}{2} + \tilde{\kappa}_k \quad (2.12)$$

**Phasic pressure.** Pressures are given by equations of state formulated in terms of phasic quantities:

$$\bar{p}_k = \mathcal{EOS}(\bar{\rho}_k, \tilde{u}_k) \quad (2.13)$$

which together with *compatibility equation*

$$\sum_k \alpha_k = 1 \quad (2.14)$$

completes problem formulation.

## 2.4 Closures

The system of eqs.(2.3)-(2.6) requires closure in terms of constitutive physics for the terms shown in boxes. These are the following.

**Interfacial mass generation source:**

$$\Gamma_k = \overline{\rho(\mathbf{v} - \mathbf{v}_i) \cdot \nabla \mathcal{X}_k} \quad (2.15)$$

where  $\mathbf{v}_i$  is velocity of the interface.

**Phasic body force:**

$$\tilde{\mathbf{b}}_k = \frac{\overline{\mathcal{X}_k \rho \mathbf{b}}}{\alpha_k \bar{\rho}_k} \quad (2.16)$$

**Interfacial momentum source:**

$$\mathbf{v}_{ki}^m = \frac{\overline{\rho \mathbf{v}(\mathbf{v} - \mathbf{v}_i) \cdot \nabla \mathcal{X}_k}}{\Gamma_k} \quad (2.17)$$

**Interfacial internal energy source:**

$$u_{ki} = \frac{\overline{\rho u(\mathbf{v} - \mathbf{v}_i) \cdot \nabla \mathcal{X}_k}}{\Gamma_k} \quad (2.18)$$

**Interfacial heat source:**

$$E_k = \overline{\mathbf{q} \cdot \nabla \mathcal{X}_k} \quad (2.19)$$

**Interfacial kinetic energy source:**

$$u_{ki}^e = \sqrt{\frac{\overline{\rho |\mathbf{v}_k|^2 (\mathbf{v} - \mathbf{v}_i) \cdot \nabla \mathcal{X}_k}}{\Gamma_k}} \quad (2.20)$$

**Phasic viscous stress:**

$$\bar{\tau}_k = \frac{\overline{\mathcal{X}_k \tau}}{\alpha_k} \quad (2.21)$$

**Interfacial pressure:**

$$p_{ki} = \frac{\overline{p \mathbf{n}_k \cdot \nabla \mathcal{X}_k}}{a_i} \quad (2.22)$$

where  $\mathbf{n}_k$  and  $a_i$  are interfacial unit normal and area.

**Interfacial viscous stress:**

$$\tau_{ki} = \frac{\overline{\tau_k \mathbf{n}_k \cdot \nabla \mathcal{X}_k}}{a_i} \quad (2.23)$$

**Phasic Reynolds stress:**

$$\mathbf{T}_k^{Re} = -\frac{\overline{\mathcal{X}_k \rho \mathbf{v}'_k \mathbf{v}'_k}}{\alpha_k} \quad (2.24)$$

**Interfacial extra momentum source:**

$$\mathbf{M}'_k = -\overline{\mathbf{T}'_{ki} \cdot \nabla \mathcal{X}_k} \quad (2.25)$$

where

$$\mathbf{T}'_{ki} = -(p - p_{ki}) \mathbf{I} + (\tau - \tau_{ki}) \quad (2.26)$$

**Phasic energy source:**

$$\tilde{\gamma}_k = \frac{\overline{\mathcal{X}_k \rho r}}{\alpha_k \bar{\rho}_k} \quad (2.27)$$

where  $r$  is volumetric energy source.

**Phasic energy flux:**

$$\bar{\mathbf{q}}_k = \frac{\overline{\mathcal{X}_k \mathbf{q}}}{\alpha_k} \quad (2.28)$$

**Phasic fluctuation (Reynolds) energy flux:**

$$\mathbf{q}_k^{Re} = \hat{\mathbf{q}}_k^{Re} + \mathbf{q}_k^T + \mathbf{q}_k^K \quad (2.29)$$

## 2.4. CLOSURES

11

**Phasic fluctuation (Reynolds) kinetic energy flux:**

$$\mathbf{q}_k^K = \frac{\overline{\mathcal{X}_k \rho \mathbf{v}'_k \frac{|\mathbf{v}'_k|^2}{2}}}{\alpha_k} \quad (2.30)$$

**Phasic fluctuation (Reynolds) shear working:**

$$\mathbf{q}_k^T = -\frac{\overline{\mathcal{X}_k \mathbf{T} \cdot \mathbf{v}'_k}}{\alpha_k} \quad (2.31)$$

**Phasic fluctuation (Reynolds) internal energy flux:**

$$\hat{\mathbf{q}}_k^{Re} = \frac{\overline{\mathcal{X}_k \rho \mathbf{v}'_k u'_k}}{\alpha_k} \quad (2.32)$$

**Interfacial extra working:**

$$W'_k = -\overline{\mathbf{T} \cdot \mathbf{v}'_k \cdot \nabla \mathcal{X}_k} \quad (2.33)$$

**Phasic internal dissipation:**

$$D_k = \frac{\overline{\mathcal{X}_k \tau : \nabla \mathbf{v}'_k}}{\alpha_k} \quad (2.34)$$

**Phasic pressure working:**

$$P_k = \frac{\overline{\mathcal{X}_k p \nabla \cdot \mathbf{v}'_k}}{\alpha_k} \quad (2.35)$$

Typically, a set of explicit algebraic formulas - “closure laws”, for eqs. (2.15)-(2.35) must be provided to complete mathematical formulations. This set must also be constrained by the following (“jump”) compatibility conditions:

$$\sum_k \Gamma_k = 0 \quad (2.36)$$

$$\sum_k (\mathbf{M}'_k + \mathbf{v}_{ki}^m + (p_{ki} - \tau_{ki}) \nabla \alpha_k) = \mathbf{m} \quad (2.37)$$

and

$$\sum_k \left( E_k + W'_k + \mathbf{M}_k \cdot \tilde{\mathbf{v}}_k + \left( u_{ki} + \frac{(v_{ki}^e)^2}{2} \right) \Gamma_k \right) = \epsilon \quad (2.38)$$

where  $\mathbf{m}$  and  $\epsilon$  are the surface tension source and the interfacial energy source, correspondingly.

## 2.5 Entropy inequality

Entropy inequality is defined by the following equation:

$$\frac{\partial(\alpha_k \bar{\rho}_k \tilde{s}_k)}{\partial t} + \nabla \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k \tilde{s}_k) \geq \nabla \cdot \left( \alpha_k \left( \bar{\phi}_k + \phi_k^{Re} \right) \right) + \alpha_k \bar{\rho}_k \tilde{\Sigma}_k + S_k + s_{ki} \Gamma_k \quad (2.39)$$

the averaged quantities are defined as follows.

**Phasic entropy:**

$$\tilde{s}_k = \frac{\overline{\mathcal{X}_k \rho s}}{\alpha_k \bar{\rho}_k} \quad (2.40)$$

**Phasic entropy flux:**

$$\bar{\phi}_k = \frac{\overline{\mathcal{X}_k \mathbf{q} / \theta}}{\alpha_k} \quad (2.41)$$

where  $\theta$  is temperature.

**Phasic fluctuation (Reynolds) kinetic energy:**

$$\phi_k^{Re} = - \frac{\overline{\mathcal{X}_k \rho s \mathbf{v}'_k}}{\alpha_k} \quad (2.42)$$

**Phasic entropy source:**

$$\tilde{\Sigma}_k = \frac{\overline{\mathcal{X}_k \rho r / \theta}}{\alpha_k \bar{\rho}_k} \quad (2.43)$$

**Interfacial entropy source:**

$$S_k = \frac{\overline{\mathbf{q}}}{\theta} \cdot \nabla \mathcal{X}_k \quad (2.44)$$

**Interfacial entropy source:**

$$s_{ki} = \frac{\overline{\rho s (\mathbf{v} - \mathbf{v}_i) \cdot \nabla \mathcal{X}_k}}{\Gamma_k} \quad (2.45)$$

## 2.6 Two-fluid equations

Without loss of generality, in the following Chapters, we will discuss two-field formulation ( $k=0,1$ ), ignoring turbulence models and viscous stresses, and focus on PDE part of the most common formulation:

**Mass:**

$$\begin{aligned} \frac{\partial(\alpha\bar{\rho}_0)}{\partial t} + \nabla \cdot (\alpha\bar{\rho}_0\tilde{\mathbf{v}}_0) &= \mathcal{S}_{\text{mass},0}(\mathbf{U}) \\ \frac{\partial((1-\alpha)\bar{\rho}_1)}{\partial t} + \nabla \cdot ((1-\alpha)\bar{\rho}_1\tilde{\mathbf{v}}_1) &= \mathcal{S}_{\text{mass},1}(\mathbf{U}) \end{aligned} \quad (2.46)$$

**Momentum:**

$$\begin{aligned} \frac{\partial(\alpha\bar{\rho}_0\tilde{\mathbf{v}}_0)}{\partial t} + \nabla \cdot (\alpha[\bar{\rho}_0\tilde{\mathbf{v}}_0 \otimes \tilde{\mathbf{v}}_0 + \bar{p}]) &= \\ \left[ \bar{p} - \underbrace{\xi(\mathbf{U})\bar{\rho}_c\Delta U^2}_{\text{Interfacial dynamic pressure}} \right] \nabla\alpha - \underbrace{\mu(\mathbf{U})\bar{\rho}_c\frac{d_d\Delta\vec{U}}{dt}}_{\text{Added mass}} + \mathcal{S}_{\text{mom},0}(\mathbf{U}) \\ \frac{\partial((1-\alpha)\bar{\rho}_1\tilde{\mathbf{v}}_1)}{\partial t} + \nabla \cdot \left( (1-\alpha) \left[ \bar{\rho}_1\tilde{\mathbf{v}}_1 \otimes \tilde{\mathbf{v}}_1 + \bar{p} - \underbrace{\Delta\bar{p}_{(01)}(\mathbf{U})}_{\text{Bulk pressure difference}} \right] \right) &= \\ - \left[ \bar{p} - \underbrace{\xi(\mathbf{U})\bar{\rho}_c\Delta U^2}_{\text{Interfacial dynamic pressure}} \right] \nabla\alpha + \underbrace{\mu(\mathbf{U})\bar{\rho}_c\frac{d_d\Delta\vec{U}}{dt}}_{\text{Added mass}} + \mathcal{S}_{\text{mom},1}(\mathbf{U}) \end{aligned} \quad (2.47)$$

**Energy:**

$$\begin{aligned}
\frac{\partial(\alpha\bar{\rho}_0\tilde{e}_0)}{\partial t} + \nabla \cdot (\alpha\tilde{\mathbf{v}}_0 [\bar{\rho}_0\tilde{e}_0 + \bar{p}]) &= \tilde{\mathbf{v}}_0 \left[ \bar{p} - \underbrace{\xi(\mathbf{U})\bar{\rho}_c\Delta U^2}_{\text{Interfacial dynamic pressure}} \right] \nabla\alpha + \\
&\quad - \underbrace{\tilde{\mathbf{v}}_0\mu(\mathbf{U})\bar{\rho}_c\frac{d_d\Delta\vec{U}}{dt}}_{\text{Added mass}} + \mathcal{S}_{\text{ene},0}(\mathbf{U}) \\
\frac{\partial((1-\alpha)\bar{\rho}_1\tilde{e}_1)}{\partial t} + \nabla \cdot \left( (1-\alpha)\tilde{\mathbf{v}}_1 \left[ \bar{\rho}_1\tilde{e}_1 + \bar{p} - \underbrace{\Delta\bar{p}_{(01)}(\mathbf{U})}_{\text{Bulk pressure difference}} \right] \right) &= \\
-\tilde{\mathbf{v}}_1 \left[ \bar{p} - \underbrace{\xi(\mathbf{U})\bar{\rho}_c\Delta U^2}_{\text{Interfacial dynamic pressure}} \right] \nabla\alpha + \underbrace{\tilde{\mathbf{v}}_1\mu(\mathbf{U})\bar{\rho}_c\frac{d_d\Delta\vec{U}}{dt}}_{\text{Added mass}} + \mathcal{S}_{\text{ene},1}(\mathbf{U})
\end{aligned} \tag{2.48}$$

Notably, compatibility equation is already incorporated into eqs.(2.46)-(2.48).

By vector  $\mathbf{U}$  we denote field conservation variables<sup>3</sup>:

$$\mathbf{U} = \begin{bmatrix} \alpha\bar{\rho}_0 \\ (1-\alpha)\bar{\rho}_1 \\ \alpha\bar{\rho}_0\tilde{\mathbf{v}}_0 \\ (1-\alpha)\bar{\rho}_1\tilde{\mathbf{v}}_1 \\ \alpha\bar{\rho}_0\tilde{e}_0 \\ (1-\alpha)\bar{\rho}_1\tilde{e}_1 \end{bmatrix} \tag{2.49}$$

By vector  $\mathbf{V}$  we will denote primitive variables, typically – those we solve for in the solution algorithm. These are algorithm-dependent. For example, in fully-

<sup>3</sup>Even though field mass, momentum and energy are not conserved due to interfacial interaction terms, it is customary call them “conservation” variables.

## 2.6. TWO-FLUID EQUATIONS

15

implicit algorithm, the following solution vector is chosen:

$$\mathbf{V} = \begin{bmatrix} \bar{p} \\ \alpha \\ \mathbf{v}_{\text{mix}} \equiv \frac{\alpha \bar{\rho}_0 \tilde{\mathbf{v}}_0 + (1-\alpha) \bar{\rho}_1 \tilde{\mathbf{v}}_1}{\alpha \bar{\rho}_0 + (1-\alpha) \bar{\rho}_1} \\ \tilde{\mathbf{v}}_0 \\ \tilde{u}_0 \\ \tilde{u}_1 \end{bmatrix} \quad (2.50)$$

where  $k = 0$  refers to liquid phase. RELAP5-3D [cdt09] solves for

$$\mathbf{V} = \begin{bmatrix} \bar{p} \\ \alpha \\ \tilde{\mathbf{v}}_0 \\ \tilde{\mathbf{v}}_1 \\ \tilde{u}_0 \\ \tilde{u}_1 \end{bmatrix} \quad (2.51)$$

All source terms and interfacial exchange/closure terms (except those due to interfacial pressure) are presumed to be in algebraic forms, and hidden in the r.h.s.  $\mathcal{S}_{\text{mass},k}(\mathbf{U})$ ,  $\vec{\mathcal{S}}_{\text{mom},k}(\mathbf{U})$  and  $\mathcal{S}_{\text{ene},k}(\mathbf{U})$ .

### 2.6.1 Bulk pressure difference and interfacial forces

In eqs.(2.46)-(2.48), we account for difference in bulk phasic pressures, introducing

$$\Delta \bar{p}_{01}(\mathbf{U}(\mathbf{x})) \equiv \bar{p}_0 - \bar{p}_1 \quad (2.52)$$

Very often,  $\Delta \bar{p}_{01}$  is dropped when it is independent of space,  $\Delta \bar{p}_{01} = \text{const.}$

We also included interfacial forces due to *interfacial dynamic pressure*

$$\delta p_1 = \xi(\mathbf{U}) \bar{\rho}_c \Delta U^2 \quad (2.53)$$

and *virtual mass*:

$$\vec{\mathcal{F}}_{\text{vm}} = \mu(\mathbf{U}) \bar{\rho}_c \frac{d_d \Delta \vec{U}}{dt} \quad (2.54)$$

where  $\xi(\mathbf{U})$ ,  $\Delta\vec{U} = \tilde{\mathbf{v}}_0 - \tilde{\mathbf{v}}_1$  and  $\mu(\mathbf{U})$  are dynamic pressure coefficient, relative velocity and added mass coefficient, respectively. By subscripts <sub>d</sub> and <sub>c</sub> we denote dispersed and continuum phase, correspondingly. Time derivative is made following the dispersed phase,

$$\frac{d_d}{dt} \equiv \frac{\partial}{\partial t} + \tilde{\mathbf{v}}_d \cdot \nabla \quad (2.55)$$

Many of known numerical implementations ignore bulk pressure difference, interfacial dynamic pressure and added mass forces,

$$\begin{aligned} \Delta\bar{p}_{01} &= 0 \\ \xi(\mathbf{U}) &= 0 \\ \mu(\mathbf{U}) &= 0 \end{aligned}$$

leading to the so-called *single-pressure 6-equation two-fluid model*<sup>4</sup>. As shown by Stuhmiller in [Stu77], this assumption not only unphysical, but also result in loss of hyperbolicity.

To close the model, one needs to provide equations of state for each phase:

$$\bar{\rho}_k = \mathcal{EOS}(\bar{p}_k, \tilde{u}_k), \quad k = 0, 1$$

It is convenient to write governing eqs.(2.46)-(2.48) in vector form:

$$(\mathbb{I} + \mathbb{A}) \mathbf{U}_t = - \left( [\mathbf{F}]_x + [\mathbf{G}]_y + [\mathbf{H}]_z + \mathbb{F}\mathbf{U}_x + \mathbb{G}\mathbf{U}_y + \mathbb{H}\mathbf{U}_z \right) + \mathbf{S} \quad (2.56)$$

where  $\mathbf{F}$ ,  $\mathbf{G}$  and  $\mathbf{H}$  are conservative flux vectors;  $\mathbb{F}$ ,  $\mathbb{G}$  and  $\mathbb{H}$  are non-conservative flux matrices;  $\mathbb{I}$  is identity matrix,  $\mathbb{A}$  is virtual mass matrix; and  $\mathbf{S}$  is a vector of source terms.

## 2.6.2 Variations

It is important to note that in particular code implementations, governing equations (2.46)-(2.48) are often modified, adjusting to a particular solution algorithm.

---

<sup>4</sup>Among major reactor thermalhydraulics codes, these terms are ignored in TRAC-BF1 [B<sup>+</sup>92] and TRACE [cdt]. RELAP5-3D [cdt09] includes virtual mass term. CATHARE [Bes90] includes both virtual mass and interfacial dynamic pressure terms.

**JFNK-based.** For example, in fully-implicit algorithm of Chapter 6.4, the following equations are actually discretized to form residuals of JFNK:

$$\left\{ \begin{array}{l} \text{Mixture mass:} \\ \text{Gas mass:} \\ \text{Mixture momentum:} \\ \text{Liquid momentum:} \\ \text{Gas total energy:} \\ \text{Liquid total energy:} \end{array} \right\} \begin{array}{l} \rightarrow \sum_{k=0,1} \text{Eq.}(2.46) \\ \rightarrow \text{Eq.}(2.46) \text{ for } k = 1 \\ \rightarrow \sum_{k=0,1} \text{Eq.}(2.47) \\ \rightarrow \text{Eq.}(2.47) \text{ for } k = 0 \\ \rightarrow \text{Eq.}(2.48) \text{ for } k = 0 \\ \rightarrow \text{Eq.}(2.48) \text{ for } k = 1 \end{array} \quad (2.57)$$

$$\mathbf{U} = \begin{bmatrix} \alpha \bar{\rho}_0 + (1 - \alpha) \bar{\rho}_1 \\ \alpha \bar{\rho}_0 \\ \alpha \bar{\rho}_0 \tilde{\mathbf{v}}_0 + (1 - \alpha) \bar{\rho}_1 \tilde{\mathbf{v}}_1 \\ (1 - \alpha) \bar{\rho}_1 \tilde{\mathbf{v}}_1 \\ \alpha \bar{\rho}_0 \tilde{e}_0 \\ (1 - \alpha) \bar{\rho}_1 \tilde{e}_1 \end{bmatrix}$$

From the point of view of numerical discretization, eqs.(2.57) and (2.46)-(2.48) are identical. The form eqs.(2.57) is however more convenient for treatment of phase appearance and disappearance.

**RELAP5-3D.** RELAP5-3D also solves for different set of equations. Instead of phasic mass and momentum equations, “sum” and “difference” mass and momentum equations (summing and subtracting eqs.(2.46) and eqs.(2.48), respectively) are formulated. Instead of phasic total energy equations, phasic specific internal energy equations are formed. Moreover, these equations are written in a non-conservative form (called “expanded form” in [cdt09]),

$$\mathbb{A} \mathbf{V}_t = - (\mathbb{F} \mathbf{V}_x + \mathbb{G} \mathbf{V}_y + \mathbb{H} \mathbf{V}_z) + \mathbf{S} \quad (2.58)$$

where  $\mathbf{V}$  is defined by eq.(2.51), and  $\mathbb{A}$ ,  $\mathbb{F}$ ,  $\mathbb{G}$ ,  $\mathbb{H}$  and  $\mathbf{S}$  are different from those in eq.(2.56). These modifications are necessary for RELAP5-3D implementation of semi-implicit and nearly-implicit algorithms. “Sum” and “difference” mass and momentum equations are introduced for convenience to treat phase appearance and disappearance.

Apparent deficiency of non-conservative (“expanded”) form is that there is no way to get conservation of mass, momentum and energy at discrete level. This is

why the last step in RELAP5-3D algorithm is to use “unexpanded” (conservative) form of phasic mass and energy equations. There is no attempt to get momentum equation update conservatively.

*This Page is Intentionally Left Blank*

## Chapter 3

# Fully-compressible, Explicit Algorithms

**E**XPLICIT algorithms for solving effective-field equations are suitable for high-speed transients, when dynamic time scales are comparable with the fastest time of the model. The fastest time scales are usually acoustic or interfacial exchange (when rapid phase transitions occur). In these cases, time steps taken due to accuracy requirements are small and typically comparable with stability limits of explicit algorithms.

In the following discussion, we avoid any specific reference to the details of space discretization.

### 3.1 Explicit Runge-Kutta scheme

For general time integration of the system eqs.(2.46)-(2.48), a number of *Strong-Stability-Preserving (SSP)* explicit time discretization methods are available [Got05]. The *Total Variation Diminishing (TVD) Runge-Kutta* methods of Shu and Osher [SO89] (a subclass of SSP) are particularly suited for this purpose. In addition to the simplicity of the Runge-Kutta methods, they are specially designed for time integration of hyperbolic conservation laws in a way that does not create spurious oscillation in the solution. The explicit Runge-Kutta schemes are defined as follows.

### 3.2. PRIMITIVE VARIABLES

21

#### First-order Forward Euler:

$$\mathbf{U}^{(n+1)} = \mathbf{U}^{(n)} + \Delta t \cdot \mathfrak{S} \left( \mathbf{U}^{(n)} \right) \quad (3.1)$$

#### Second-order Heun:

$$\left\{ \begin{array}{l} \mathbf{U}^{(1)} = \mathbf{U}^{(n)} + \Delta t \cdot \mathfrak{S} \left( \mathbf{U}^{(n)} \right) \\ \mathbf{U}^{(n+1)} = \frac{1}{2} \mathbf{U}^{(n)} + \frac{1}{2} \left( \mathbf{U}^{(1)} + \Delta t \cdot \mathfrak{S} \left( \mathbf{U}^{(1)} \right) \right) \end{array} \right. \left| \begin{array}{l} t^{(1)} = t^{(n+1)} \\ \hline t^{(2)} = t^{(n+\frac{1}{2})} \end{array} \right. \quad (3.2)$$

#### Third-order RK-TVD:

$$\left\{ \begin{array}{l} \mathbf{U}^{(1)} = \mathbf{U}^{(n)} + \Delta t \cdot \mathfrak{S} \left( \mathbf{U}^{(n)} \right) \\ \mathbf{U}^{(2)} = \frac{3}{4} \mathbf{U}^{(n)} + \frac{1}{4} \left( \mathbf{U}^{(1)} + \Delta t \cdot \mathfrak{S} \left( \mathbf{U}^{(1)} \right) \right) \\ \mathbf{U}^{(n+1)} = \frac{1}{3} \mathbf{U}^{(n)} + \frac{2}{3} \left( \mathbf{U}^{(2)} + \Delta t \cdot \mathfrak{S} \left( \mathbf{U}^{(2)} \right) \right) \end{array} \right. \left| \begin{array}{l} t^{(1)} = t^{(n+1)} \\ \hline t^{(2)} = t^{(n+\frac{1}{2})} \end{array} \right. \quad (3.3)$$

where  $\Delta t$  is a time step, while  $\mathbf{U}$  is a vector of conservative variables, defined by eq.(2.49). The spatial/source discretization operator  $\mathfrak{S}^{(\text{rk})}$  is given by

$$\mathfrak{S}^{(\text{rk})} = \left[ \begin{array}{c} -\nabla_{\rho_k} \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^{(\text{rk})} + \mathcal{S}_{\text{mass},k}^{(\text{rk})} \\ -\nabla_{\mathbf{v}_k} \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k)^{(\text{rk})} + \alpha_k^{(\text{rk})} \nabla_{\mathbf{v}_k} \bar{p}^{(\text{rk})} + \mathcal{S}_{\text{mom},k}^{(\text{rk})} \\ -\nabla_{e_k} \cdot (\alpha_k \tilde{\mathbf{v}}_k (\bar{\rho}_k \tilde{e}_k + \bar{p}))^{(\text{rk})} + (\tilde{\mathbf{v}}_k \bar{p})^{(\text{rk})} \nabla_{e_k} \alpha_k^{(\text{rk})} + \mathcal{S}_{\text{ene},k}^{(\text{rk})} \end{array} \right] \quad (3.4)$$

## 3.2 Primitive variables

At the end of each Runge-Kutta step, primitive variables are computed from known conservative variables  $(\alpha_k \bar{\rho}_k)^{(\text{rk})}$ ,  $(\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^{(\text{rk})}$  and  $(\alpha_k \bar{\rho}_k \tilde{e}_k)^{(\text{rk})}$  as follows.

### 1. Phase velocities:

$$\tilde{\mathbf{v}}_k^{(\text{rk})} = \frac{(\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^{(\text{rk})}}{(\alpha_k \bar{\rho}_k)^{(\text{rk})}} \quad (3.5)$$

## 2. Phase specific internal energies:

$$\tilde{u}_k^{(\text{rk})} = \frac{(\alpha_k \bar{\rho}_k \tilde{e}_k)^{(\text{rk})}}{(\alpha_k \bar{\rho}_k)^{(\text{rk})}} - \frac{\left(\tilde{\mathbf{v}}_k^{(\text{rk})}\right)^2}{2} \quad (3.6)$$

3. **Pressure.** Using phasic equations of state  $\rho_k(\bar{p}, \tilde{u}_k^{(\text{rk})})$ , void fractions are defined as a function of pressure as

$$\alpha_k(\bar{p}) = \frac{(\alpha_k \bar{\rho}_k)^{(\text{rk})}}{\rho_k(\bar{p}, \tilde{u}_k^{(\text{rk})})} \quad (3.7)$$

Using “compatibility” equation,  $\left[\sum_k \alpha_k = 1\right]$ , the following non-linear equation for pressure is defined:

$$\mathcal{F}(\bar{p}) = 1 - \sum_k \frac{(\alpha_k \bar{\rho}_k)^{(\text{rk})}}{\rho_k(\bar{p}, \tilde{u}_k^{(\text{rk})})} = 0 \quad (3.8)$$

Eq.(3.8) can be solved for pressure  $\bar{p}$  using Newton’s method<sup>1</sup>.

4. **Phasic void fractions** are finally computed using  $\bar{p}$  and eq.(3.7).

## 3.3 Remarks

1. In most nuclear reactor applications, time step and stability limitations of explicit schemes are too restrictive. All current and next generation reactor safety codes use (and will be using) semi-implicit and fully-implicit algorithms, which eliminate the stiffness associated with acoustic pressure waves and interfacial exchange terms.
2. Phase appearance and disappearance is a challenging numerical issue. We will discuss this in Chapter 7.

<sup>1</sup>For some simple fluids,  $\mathcal{F}(\bar{p})$  can be solved analytically. For example, when the stiffened gas equation state is used for both phases,  $\mathcal{F}(\bar{p})$  is quadratic in  $\bar{p}$ . Though, even in this case the use of iterative Newton method is desirable from round-off errors point of view [CL07].

*This Page is Intentionally Left Blank*

## Chapter 4

# Operator-Splitting Algorithms

ALL current work-horse reactor thermalhydraulics codes (RELAP5 [cdt09], ATAC [B<sup>+</sup>92], TRACE [cdt], CATHARE [Bes90, BPB93] and RETRAN [McF81]) originate from Liles and Reed [LW78] extension of Harlow and Amsden [HA68, HA71, HA75b, HA75a] all-speed “*Implicit Continuous-fluid Eulerian (ICE)*” algorithm. Liles and Reed work was focusing on solving drift-flux (5-equation) two-fluid model, demonstrating how more tight coupling with energy equation can be achieved within the general framework of the original ICE algorithm. These ideas are straightforwardly extendable to the 6-equation model outlined in Section 2.6. The whole family of these algorithms are often referred to as “*semi-implicit*”, Section 4.1.

A couple of extensions of the “*semi-implicit*” algorithm were introduced in the beginning of 1980s, based on the fractional step method, to enhance stability of the method and to eliminate material CFL restrictions. These are the *SETS* algorithm (implemented in TRAC and TRACE, [B<sup>+</sup>92, cdt]), and *Nearly-Implicit* algorithm (implemented in RELAP5-3D, [cdt09]). We will discuss these extensions in Section 4.2.

Even though ICE-based algorithms are called “all-speed”, they are not the methods of choice for high-speed (supersonic) applications, where density-based methods are both more accurate and cost-effective. This is why we refer to ICE-based algorithms as “*weakly-compressible*”, even though the compressibility effects are fully accounted for in numerical discretizations.

All reactor thermalhydraulics codes utilize first-order finite-difference donor-

cell/upwinding based schemes, implemented on structured staggered meshes. In the following discussion though, we avoid any specific reference to the details of space discretization, allowing for straightforward extension of basic ideas to more modern and more general high-order space discretization schemes with (approximate) Riemann solvers [Tor99, Sta06], such as high-order finite-volume or Discontinuous Galerkin on unstructured collocated meshes.

Without loss of generality, we make discussion of algorithms assuming that that the vector of unknowns solved for is

$$\mathbf{V} = \begin{bmatrix} \bar{p} \\ \alpha \\ \tilde{\mathbf{v}}_0 \\ \tilde{\mathbf{v}}_1 \\ \tilde{u}_0 \\ \tilde{u}_1 \end{bmatrix} \quad (4.1)$$

while the governing equations are written in the form of phasic mass, momentum and energy conservation, eqs.(2.46)-(2.48) (with neglected virtual mass, interfacial dynamic pressure and bulk pressure differences, for simplicity). Thus, the vector of conserved variables is:

$$\mathbf{U} = \begin{bmatrix} \alpha \bar{\rho}_0 \\ (1 - \alpha) \bar{\rho}_1 \\ \alpha \bar{\rho}_0 \tilde{\mathbf{v}}_0 \\ (1 - \alpha) \bar{\rho}_1 \tilde{\mathbf{v}}_1 \\ \alpha \bar{\rho}_0 \tilde{e}_0 \\ (1 - \alpha) \bar{\rho}_1 \tilde{e}_1 \end{bmatrix} \quad (4.2)$$

## 4.1 Semi-Implicit Algorithms

The chief consideration in constructing *semi-implicit* scheme is to eliminate stability limits due to *acoustic pressure wave propagation* and due to *interfacial mass, momentum and energy exchange terms*. This is achieved by treating *implicitly* (a) convective velocities in mass and energy conservation equations, (b) pressure gradient in momentum conservation equations, and (c) all interfacial exchange terms on the r.h.s. of eqs.(2.46)-(2.48). All the rest terms are treated effectively explicitly, which imposes the following material CFL stability limit:

$$\Delta t < \frac{\Delta x}{|\tilde{\mathbf{v}}_k|} \quad (4.3)$$

There are two slightly different versions of the algorithm – the one implemented in RELAP5-3D [cdt09], and the other one in TRAC and TRACE [B<sup>+</sup>92, cdt]. In the following discussion, we will highlight the differences.

The algorithm is constructed in a two-stage “predictor-corrector” fashion.

### 4.1.1 Predictor discretization

The basic “predictor” discretization of six governing conservation equations<sup>1,2</sup> is as follows.

**Mass**,  $k=0,1$ :

$$\underbrace{\alpha_k^n \left( \widehat{\bar{\rho}}_k^{n+1} - \bar{\rho}_k^n \right) + \bar{\rho}_k^n \left( \widehat{\alpha}_k^{n+1} - \alpha_k^n \right)}_{\text{RELAP5-3D}} = \underbrace{\left[ (\alpha_k \bar{\rho}_k)^{\widehat{n+1}} - (\alpha_k \bar{\rho}_k)^n \right]}_{\text{TRAC/TRACE}} \quad (4.4)$$

$$= \Delta t \left( -\nabla_{\rho_k} \cdot \left( (\alpha_k \bar{\rho}_k)^n \tilde{\mathbf{v}}_k^{n+1} \right) + \tilde{\mathcal{S}}_{\text{mass},k}^n \cdot \underbrace{\begin{bmatrix} 1 \\ \widehat{\alpha}_k^{n+1} \\ \widehat{\rho}_{m=0,1}^{n+1} \\ \tilde{\mathbf{v}}_{m=0,1}^{n+1} \\ \tilde{u}_{m=0,1}^{n+1} \end{bmatrix}}_{\text{Linearization (if non-linear)}} \right)$$

<sup>1</sup>As mentioned above, we ignore spatial discretization operator details, focussing on time treatment.

<sup>2</sup>TRAC, TRACE, RELAP5 and CATHARE use *specific internal energy* equation instead of *total energy*. We prefer total energy equation, as this is what is conserved. Thus, there are some (hopefully minor) numerical implementation differences between our discussion and legacy T/H codes.

## 4.1. SEMI-IMPLICIT ALGORITHMS

27

**Total energy**,  $k=0,1$ :

$$\underbrace{(\alpha_k \bar{\rho}_k)^n \left[ \tilde{u}_k^{n+1} - \tilde{u}_k^n + \frac{\tilde{\mathbf{v}}_k^n}{2} \cdot (\tilde{\mathbf{v}}_k^{n+1} - \tilde{\mathbf{v}}_k^n) \right] + \tilde{e}_k^n \left[ \alpha_k^n (\widehat{\bar{\rho}}_k^{n+1} - \bar{\rho}_k^n) + \bar{\rho}_k^n (\alpha_k^{n+1} - \alpha_k^n) \right]}_{\text{RELAP5-3D}} = \quad (4.5)$$

$$\underbrace{\text{TRAC/TRACE: } \left[ (\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1} - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n \right]}_{\text{RELAP5-3D}}$$

$$= \Delta t \left( -\nabla_{e_k} \cdot \underbrace{\left( \tilde{\mathbf{v}}_k^{n+1} (\alpha_k \bar{\rho}_k)^n \left( \tilde{e}_k^n + \frac{\bar{p}^*}{\bar{\rho}_k^n} \right) \right)}_{\substack{\text{RELAP5-3D} \rightarrow \bar{p}^* = \bar{p}^n \\ \text{TRAC/TRACE} \rightarrow \bar{p}^* = \bar{p}^{n+1}}} + \tilde{\mathbf{v}}_k^{n+1} \cdot \left( \bar{p}^* \nabla_{e_k} \alpha_k^n \right) + \right.$$

$$\left. + \underbrace{\tilde{\mathbf{S}}_{\text{ene},k}^n \cdot \begin{bmatrix} 1 \\ \alpha_k^{n+1} \\ \bar{\rho}_{m=0,1}^{n+1} \\ \tilde{\mathbf{v}}_{m=0,1}^{n+1} \\ \tilde{u}_{m=0,1}^{n+1} \end{bmatrix}}_{\text{Linearization (if non-linear)}} \right)$$

**Momentum**,  $k=0,1$ :

$$(\alpha_k \bar{\rho}_k)^n \tilde{\mathbf{v}}_k^{n+1} - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n = \quad (4.6)$$

$$\Delta t \left[ -\nabla_{\mathbf{v}_k} \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k)^n + \alpha_k^n \nabla_{\mathbf{v}_k} (\bar{p}^{n+1}) + \underbrace{\mathbb{S}_{\text{mom},k}^n \begin{bmatrix} 1 \\ \tilde{\mathbf{v}}_{m=0,1}^{n+1} \end{bmatrix}}_{\text{Linearization (if non-linear)}} \right]$$

where by  $(\cdot)^{n+1}$  we denote new-time ‘‘predictor’’ values. Also,  $(\nabla_{\rho_k} \cdot)$ ,  $(\nabla_{e_k} \cdot)$  and  $(\nabla_{\mathbf{v}_k} \cdot)$  are implementation-specific discrete divergence operators<sup>3</sup>, while  $(\nabla_{\mathbf{v}_k} \cdot)$

<sup>3</sup>We are leaving the door open for non-linear flux treatment with approximate Riemann

and  $(\nabla_{e_k})$  are discrete gradient operators. Interfacial mass and energy exchange and other mass and energy source terms are linearized (if necessary) relative to new-time predictor solution vector, with linearization coefficients defined by vectors  $\vec{S}_{\text{mass},k}^n$  and  $\vec{S}_{\text{ene},k}^n$ . Source terms of phasic momentum equations are linearized only<sup>4</sup> relative to predictor velocity,  $\tilde{\mathbf{v}}_{m=0,1}^{n+1}$ , with linearization coefficients represented by matrix  $\mathbb{S}_{\text{mom},k}^n$ .

Temporal derivatives on the l.h.s. of eqs.(4.4)-(4.6) are expressed along the lines of “expanded” equation set of RELAP5-3D [cdt09]. This is necessary to produce simple linear equations<sup>5</sup>.

Eqs.(4.4)-(4.6) are constructed in a special way, so that the only spatial derivatives on new-time values  $(\cdot)^{\widehat{n+1}}$  and  $(\cdot)^{n+1}$  are for the bulk pressure  $\bar{p}^{n+1}$  and phasic velocity  $\tilde{\mathbf{v}}_k^{n+1}$ . Moreover, eq.(4.6) contains derivatives for only  $\bar{p}^{n+1}$ , which suggests the construction of *pressure-Helmholtz equation* as discussed next.

### 4.1.2 Pressure-Helmholtz equation

**Linearization of density.** The first step in construction of the pressure equation is to linearize phasic densities along the lines of ICE algorithm [HA68, HA71]:

$$\widehat{\bar{\rho}}_k^{n+1} \approx \bar{\rho}_k^\diamond + \left( \bar{p}^{n+1} - \bar{p}^\diamond \right) \left. \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \right|_{\tilde{u}_k = \text{const}}^\diamond + \left( \tilde{u}_k^{n+1} - \tilde{u}_k^\diamond \right) \left. \frac{\partial \bar{\rho}_k}{\partial \tilde{u}_k} \right|_{\bar{p} = \text{const}}^\diamond \quad (4.7)$$

where partials  $\left. \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \right|_{\tilde{u}_k = \text{const}}$  and  $\left. \frac{\partial \bar{\rho}_k}{\partial \tilde{u}_k} \right|_{\bar{p} = \text{const}}$  are available from equations of state  $\bar{\rho}_k = \mathcal{EOS}(\bar{p}, \tilde{u}_k)$  and evaluated at the state  $(\bar{p}, \tilde{u}_k)^\diamond$ . RELAP5-3D choses  $(\cdot)^\diamond = (\cdot)^n$ , which together with  $\bar{p}^* = \bar{p}^n$  in eq.(4.5) makes the final coupled mass-energy (pressure-Helmholtz) system linear. TRAC/TRACE employ iterative algorithm to account for non-linearities. We will discuss these in details in a moment.

solvers by retaining subscripts  $\rho_k, e_k$  and  $\mathbf{v}_k$ .

<sup>4</sup>This is to avoid appearance of gradient terms on variables other than velocity, when momentum equations are plugged into mass and energy equations.

<sup>5</sup>TRAC/TRACE versions are also shown in boxes below. They do not generate linear equations, which necessitates iterations on pressure-Helmholtz eq.(4.12), as discussed in Section 4.1.2

#### 4.1. SEMI-IMPLICIT ALGORITHMS

29

Next, equations (4.7) are substituted into phasic mass and energy equations (4.4) and (4.5) to eliminate “predictor” phasic densities  $\widehat{\rho}_k^{n+1}$ .

**Compatibility equation.** “Predictor” void fraction  $\alpha_1^{n+1}$  can also be eliminated by substituting the compatibility condition  $[\alpha_1 = 1 - \alpha_0]$  into eqs.(4.4) and (4.5).

**Phasic velocities.** The next step is to solve generally coupled (through the source terms<sup>6</sup>) phasic momentum equations. The result can be presented as the following vector:

$$\begin{aligned}\tilde{\mathbf{v}}_0^{n+1} &= \mathbf{a}_0^n + b_0^n \nabla_{\mathbf{v}_0} \bar{p}^{n+1} + c_0^n \nabla_{\mathbf{v}_1} \bar{p}^{n+1} \\ \tilde{\mathbf{v}}_1^{n+1} &= \mathbf{a}_1^n + b_1^n \nabla_{\mathbf{v}_0} \bar{p}^{n+1} + c_1^n \nabla_{\mathbf{v}_1} \bar{p}^{n+1}\end{aligned}\quad (4.8)$$

where  $\mathbf{a}_{0,1}^n$ ,  $b_{0,1}^n$  and  $c_{0,1}^n$  are coefficients – functions of known variables from old time step level.

The next step would be elimination of “predictor” phasic velocities, by substituting eq.(4.8) into phasic mass and energy conservation equations (4.4) and (4.5). After collecting terms, the resulting system of four equations can be presented in the following matrix form:

$$\underbrace{\begin{bmatrix} A_{\alpha_0, \alpha_0}^{n, \diamond} & A_{\alpha_0, \tilde{u}_0}^{n, \diamond} & A_{\alpha_0, \tilde{u}_1}^{n, \diamond} & A_{\alpha_0, \bar{p}}^{n, \diamond} \\ A_{\tilde{u}_0, \alpha_0}^{n, \diamond} & A_{\tilde{u}_0, \tilde{u}_0}^{n, \diamond} & A_{\tilde{u}_0, \tilde{u}_1}^{n, \diamond} & A_{\tilde{u}_0, \bar{p}}^{n, \diamond} \\ A_{\tilde{u}_1, \alpha_0}^{n, \diamond} & A_{\tilde{u}_1, \tilde{u}_0}^{n, \diamond} & A_{\tilde{u}_1, \tilde{u}_1}^{n, \diamond} & A_{\tilde{u}_1, \bar{p}}^{n, \diamond} \\ A_{\bar{p}, \alpha_0}^{n, \diamond} & A_{\bar{p}, \tilde{u}_0}^{n, \diamond} & A_{\bar{p}, \tilde{u}_1}^{n, \diamond} & A_{\bar{p}, \bar{p}}^{n, \diamond} \end{bmatrix}}_{\mathbb{A}} \underbrace{\begin{bmatrix} \widehat{\alpha}_0^{n+1} \\ \widehat{\tilde{u}}_0^{n+1} \\ \widehat{\tilde{u}}_1^{n+1} \\ \widehat{\bar{p}}^{n+1} \end{bmatrix}}_{\mathbf{X}} = \mathbf{b} \quad (4.9)$$

where  $\mathbb{A}$  is the *mass-energy* matrix with known coefficients evaluated using states  $(\cdot)^n$  and  $(\cdot)^\diamond$ .

The r.h.s. of eq.(4.9) contains four Laplacians:

$$\mathbf{b} = \begin{bmatrix} \gamma_{\bar{\rho}_0}^{n, \diamond} + \Delta t \nabla_{\rho_0} \cdot [(\alpha_0 \bar{\rho}_0)^n (b_0^n \nabla_{\mathbf{v}_0} \bar{p}^{n+1} + c_0^n \nabla_{\mathbf{v}_1} \bar{p}^{n+1})] \\ \gamma_{\bar{\rho}_1}^{n, \diamond} + \Delta t \nabla_{\rho_1} \cdot [(\alpha_1 \bar{\rho}_1)^n (b_1^n \nabla_{\mathbf{v}_0} \bar{p}^{n+1} + c_1^n \nabla_{\mathbf{v}_1} \bar{p}^{n+1})] \\ \gamma_{\bar{e}_0}^{n, \diamond} + \Delta t \nabla_{e_0} \cdot [\alpha_0^n (\bar{\rho}_0^n \bar{e}_0^n + \bar{p}^\diamond) (b_0^n \nabla_{\mathbf{v}_0} \bar{p}^{n+1} + c_0^n \nabla_{\mathbf{v}_1} \bar{p}^{n+1})] \\ \gamma_{\bar{e}_1}^{n, \diamond} + \Delta t \nabla_{e_1} \cdot [\alpha_1^n (\bar{\rho}_1^n \bar{e}_1^n + \bar{p}^\diamond) (b_1^n \nabla_{\mathbf{v}_0} \bar{p}^{n+1} + c_1^n \nabla_{\mathbf{v}_1} \bar{p}^{n+1})] \end{bmatrix} \quad (4.10)$$

<sup>6</sup>Coupling can be easily accounted for by simple algebraic manipulation.

where  $\gamma_{z=\bar{\rho}_k, \bar{\epsilon}_k}^{n, \diamond}$  are known coefficients evaluated using states  $(\cdot)^n$  and  $(\cdot)^\diamond$ .

Finally, these four Helmholtz equations can be collapsed into one by eliminating  $\widehat{\alpha}_0^{n+1}$ ,  $\widehat{u}_0^{n+1}$  and  $\widehat{u}_1^{n+1}$ . This can be achieved by inverting matrix  $\mathbb{A}$ <sup>7</sup> (this is done at each cell of the computational domain) and multiplying  $\mathbb{A}^{-1}$  on the left and right side of eq.(4.9). The resulting variable-coefficient pressure-Helmholtz equation is the last (fourth) equation in

$$\mathbb{A}^{-1}\mathbf{b} - \mathbf{X} = 0 \quad (4.11)$$

It can be presented in the following form:

$$\begin{aligned} \nabla_{\rho_0} \cdot (a \nabla_{v_0} \bar{p}^{n+1} + b \nabla_{v_1} \bar{p}^{n+1}) + \nabla_{\rho_1} \cdot (c \nabla_{v_0} \bar{p}^{n+1} + d \nabla_{v_1} \bar{p}^{n+1}) + \\ \nabla_{e_0} \cdot (e \nabla_{v_0} \bar{p}^{n+1} + f \nabla_{v_1} \bar{p}^{n+1}) + \nabla_{e_1} \cdot (g \nabla_{v_0} \bar{p}^{n+1} + h \nabla_{v_1} \bar{p}^{n+1}) = \\ + i \bar{p}^{n+1} = j \end{aligned} \quad (4.12)$$

where  $(a-j)$  are constants evaluated using states  $(\cdot)^n$  and  $(\cdot)^\diamond$ . This parabolic equation must be solved implicitly by some direct or iterative method for  $\bar{p}^{n+1}$ . There is a number of well-established Helmholtz solvers available, including different variations of algebraic and full multigrids. In RELAP5-3D, a variation of LU decomposition solver (so-called *Border-Profile Lower-Upper (BPLU)* [Mes98]) is utilized as a default.

**RELAP5-3D.** As discussed above, by choosing

$$\bar{p}^* = \bar{p}^n$$

in phasic energy equations (4.5), and

$$(\cdot)^\diamond = (\cdot)^n$$

in linearization eq.(4.7), RELAP5-3D removes non-linearity in the coupled phasic mass and energy equations, and no iterations are necessary.

**TRAC/TRACE.** These codes follow the original work by Liles and Reed [LW78], in which iterative *Newton Block Gauss Seidel (NBGS)* was introduced<sup>8</sup>. Effectively, RELAP5-3D's version is just one (the first) iteration (with initial guess

<sup>7</sup>In RELAP5-3D, a simple LU decomposition is utilized [cdt09].

<sup>8</sup>Liles and Reed [LW78] implemented this algorithm for drift-flux model.

taken at the state  $(\cdot)^n$  of this non-linear procedure. As discussed by Mahaffy in [Mah93], tightly converged non-linear iterations of TRAC/TRACE allow to eliminate systematic mass errors that occur in the RELAP5-3D's version of the algorithm.

To complete predictor step, new-time velocities  $\tilde{\mathbf{v}}_k^{n+1}$  are computed using equations (4.8) with new pressure values  $\bar{p}^{n+1}$ .

Finally, “predictor” values of void fraction  $\alpha_0^{n+1}$  and phasic specific internal energies  $\tilde{u}_{k=0,1}^{n+1}$  are computed by evaluating the first three rows of

$$\mathbb{A}^{-1}\mathbf{b}$$

using already available  $\bar{p}^{n+1}$  field.

### 4.1.3 Corrector

1. **(RELAP5-3D only)**<sup>9</sup> New-time phasic densities  $(\alpha_k \bar{\rho}_k)^{n+1}$  and energies  $(\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1}$  are obtained by solving eqs.(4.4) and (4.5), in which *conservative* (“unexpanded”)

$$[(\alpha_k \bar{\rho}_k)^{n+1} - (\alpha_k \bar{\rho}_k)^n]$$

and

$$[(\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1} - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n]$$

forms of l.h.s. (transient terms) are utilized. “Predictor” values of phasic densities, specific internal energies and void fraction are used for evaluation of the source terms. This step is necessary to alleviate mass and energy conservation errors which occur due to approximation of transient terms in “expanded” form of mass and energy equations (4.4) and (4.5).

2. Specific internal energies and densities for each phase are evaluated as

$$\tilde{u}_k^{n+1} = \frac{(\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1}}{(\alpha_k \bar{\rho}_k)^{n+1}} - \frac{(\tilde{\mathbf{v}}_k^{n+1})^2}{2}; \text{ and } \bar{\rho}_k^{n+1} = \mathcal{EOS}(\bar{p}^{n+1}, \tilde{u}_k^{n+1})$$

<sup>9</sup>This step is unnecessary for TRAC/TRACE, in which non-linearly-coupled phasic mass and energy equations are solved (Section 4.1.2).

3. Finally, new-time void fraction is evaluated using density and phasic density for one of the phase (typically, “continuous” phase is chosen<sup>10</sup>):

$$\alpha_c^{n+1} = \frac{(\alpha_c \bar{\rho}_c)^{n+1}}{\bar{\rho}_c^{n+1}}$$

while void fraction of the “dispersed” phase is computed from the “compatibility” constraint:

$$\alpha_d^{n+1} = 1 - \alpha_c^{n+1}$$

#### 4.1.4 Remarks

1. Operator-splitting in *semi-implicit* algorithm makes it only first-order-accurate in time.
2. The choice of time-level mixing in flux discretization of phasic mass and energy equations (4.4) and (4.5) is necessary to maintain simple structure of the method, with only one implicit equation (4.12) being solved.
3. Mixed time-level transient terms (“expanded” forms) of RELAP5-3D are also were caused by the necessity to maintain simplicity and linearity of the discrete form.
4. Mixing time levels in discretization of transient and flux terms suggests potential problems in achieving steady-states (this is well-known problem of RELAP5-3D).
5. It can be easily seen that even though “unexpanded” discretizations are used to update phasic mass and energy (step 1 of the corrector), the final solution is *not conservative* for dispersed phase, due to enforcing “compatibility” at the last corrector stage, thereby “chopping-off” some mass and energy. Even though this might not be seen as a serious “crime”, at long transients, these non-conservation errors tend to accumulate<sup>11</sup>. This also suggests potential problems at  $\alpha_k \rightarrow 0$ .

<sup>10</sup>RELAP5-3D uses liquid phase, [cdt09, TR86], because the liquid phase is nearly incompressible and less void error is expected using the liquid mass balance.

<sup>11</sup>In fact, RELAP5-3D uses a measure of difference between “predictor” phasic mass and final new-time phasic mass for control of time step, [cdt09].

6. The size of time step is also an issue for RELAP5-3D due to linearization of equation of state eq.(4.7), as this is done relative to known *old time-step phase density*. Even though the algorithm is stable for arbitrary acoustic CFL, operator-splitting errors due to linearization of density might be significant for large time steps.
7. The algorithm requires special treatment for phase appearance and disappearance. We discuss these issues in Chapter 7.
8. Extension of the algorithm to multi- $(N)$ -field formulation is rather straightforward. The algorithm would still require only one implicit Helmholtz solver. However, the size of the matrix  $\mathbb{A}$  in eq.(4.9) will be  $(2N \times 2N)$ , for  $N$  phase specific internal energies, pressure and  $(N - 1)$  phase void fractions. All the rest will stay essentially the same.
9. It is straightforward to incorporate transport of non-condensable gases and other passive scalars transport equations (like boron or corrosion products). In fact, both RELAP5-3D and TRACE do solve for noncondensable gases, in which case the matrix  $\mathbb{A}$  in eq.(4.9) is of size  $(5 \times 5)$ .
10. Even though we neglected added mass, interfacial dynamic pressure and phasic bulk pressure differences, these can be straightforwardly incorporated in the numerical algorithm. In fact, RELAP5-3D does include added mass term<sup>12</sup>.
11. Even though *semi-implicit* algorithm is only-first order in time and non-conservative, it seems to be very attractive for *physics-based preconditioning* of the JFNK-based algorithm discussed in Section 6.4. Reduction of size for preconditioning linear algebra solved is significant even in 1D (six times less for two-fluid formulation). It is 10 times less for 3D two-fluid formulation, and 15 times less for 3D three-fluid formulation<sup>13</sup>. With JFNK, all above discussed problems with first-order operator-splitting errors and non-conservation will be avoided. Special care however must be taken to ensure

<sup>12</sup>TRAC and TRACE ignore all these physical closure terms.

<sup>13</sup>And this is on the top of avoiding to compute full Jacobians, which are necessary for any mathematical-based preconditioners, like ILU. Besides, equation (4.12) does not need to be solved tightly, as it is used only for preconditioning. For example, only one or two cycles of multi-grid method can be involved if adequate, which is significant CPU saving.

consistency of the discretization at the JFNK solution and preconditioning stages.

12. It is straightforward to turn operator-splitting “*semi-implicit*” algorithm into a segregated algorithm, (see Section 5.2), by modifying “mixed-time” flux terms (replacing “old-time” values by “iteration” values), and building Picard-based outer iteration loop around it. This could be an alternative to fractional step algorithms discussed in Section 4.2.
13. Wall heat transfer and friction are treated explicitly in all RELAP5-3D, TRAC and TRACE, which does impose certain additional stability constraints on the algorithms. This is also the case for fractional step variations, discussed next.

## 4.2 Fractional step based variations

Time steps of “*semi-implicit*” algorithms in Section 4.1 are limited by material CFL eq.(4.3). This is rather restrictive for many reactor-related transients, including those involving long-transients and natural convection. To improve stability, Stewart [Ste81] and Mahaffy [Mah82] introduced modifications based on fractional step method [Yan71]. Mahaffy’s SETS algorithm is implemented in TRAC and TRACE, and in a three-dimensional reactor pressure vessel module of CATHARE [Mah93, Sta01, BPB93], and we will discuss it in Section 4.2.1. Trapp and Riemke introduced another variation – “*Nearly-Implicit*” algorithm in [TR86], and it will be discussed in Section 4.2.2.

The basic idea of both methods is to increase “implicitness” by effectively making convective fluxes in all phasic mass, momentum and energy equations evaluated at “new” time ( $n + 1$ ) level. Instead of moving to fully-implicit, fractional step methods achieve this by introducing additional “stabilizer” steps.

### 4.2.1 Stability-Enhancing Two-Step method (SETS)

As discussed by Mahaffy [Mah82, Mah93], there are several ways to implement fractional step algorithm. The one introduced in [Mah82] is implemented in both TRAC and TRACE, and can be described as a three-stage algorithm:

- I. Stabilizer for phasic momentum equations,

## 4.2. FRACTIONAL STEP BASED VARIATIONS

35

- II. Modified “semi-implicit” step, and
- III. Stabilizer for phasic mass and energy equations.

### Step I: Stabilizer for phasic momentum equations

The main purpose of this step is to provide “predictor” value of phasic velocities,  $\widehat{\tilde{\mathbf{v}}}_k^{n+1}$ , to be used in the “*more-implicit*” treatment of the momentum equation convective terms. No attempts are made to force these “predictor” velocity fields to satisfy mass or energy conservation. The equations solved at this stage are:

**Momentum**,  $k=0,1$ :

$$\begin{aligned}
 & (\alpha_k \bar{\rho}_k)^n \widehat{\tilde{\mathbf{v}}}_k^{n+1} - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n = \\
 & \Delta t \left[ -\nabla_{\mathbf{v}_k} \cdot \left( (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n \otimes \widehat{\tilde{\mathbf{v}}}_k^{n+1} \right) + \alpha_k^n \nabla_{\mathbf{v}_k} (\bar{p}^n) + \underbrace{\mathbb{S}_{\text{mom},k}^n \left[ \begin{array}{c} 1 \\ \widehat{\tilde{\mathbf{v}}}_k^{n+1} \\ m=0,1 \end{array} \right]}_{\text{Linearization (if non-linear)}} \right] \quad (4.13)
 \end{aligned}$$

These equations are implicit in  $\widehat{\tilde{\mathbf{v}}}_k^{n+1}$ , and can be easily solved with some appropriate direct or iterative method<sup>14</sup>.

### Step II: Modified “semi-implicit”

At this stage, “semi-implicit” algorithm of Section 4.1 (TRAC/TRACE version) is applied, with the following modified phasic momentum equation:

**Momentum**,  $k=0,1$ :

$$\begin{aligned}
 & (\alpha_k \bar{\rho}_k)^n \tilde{\mathbf{v}}_k^{n+1} - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n = \\
 & \Delta t \left[ -\nabla_{\mathbf{v}_k} \cdot \left( \underbrace{(\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n \otimes \widehat{\tilde{\mathbf{v}}}_k^{n+1}}_{\text{Modification}} \right) + \alpha_k^n \nabla_{\mathbf{v}_k} (\bar{p}^{n+1}) + \underbrace{\mathbb{S}_{\text{mom},k}^n \left[ \begin{array}{c} 1 \\ \tilde{\mathbf{v}}_k^{n+1} \\ m=0,1 \end{array} \right]}_{\text{Linearization (if non-linear)}} \right] \quad (4.14)
 \end{aligned}$$

<sup>14</sup>In general, these phasic momentum equations can be locally coupled through the interfacial exchange terms.

Modification involves convective term, in which “predictor” value of phasic velocity  $\widehat{\tilde{\mathbf{v}}_k^{n+1}}$  is used instead of  $\tilde{\mathbf{v}}_k^n$ . Since  $\widehat{\tilde{\mathbf{v}}_k^{n+1}}$  is already known from the Step I, no other modifications are required for the “semi-implicit” algorithm.

As a result of this step, new-time values of phasic velocities  $\tilde{\mathbf{v}}_k^{n+1}$  and pressure  $\bar{p}^{n+1}$  are obtained, as well as “predictor” values for phasic density, void fractions and energy,  $\widehat{\bar{\rho}}_k^{n+1}$ ,  $\widehat{\alpha}_k^{n+1}$  and  $\widehat{\tilde{e}}_k^{n+1}$ .

### Step III: Stabilizer for phasic mass and energy equations

The purpose of this step is to update phasic density and energy equations, using the following implicit equations:

**Mass**,  $k=0,1$ :

$$\begin{aligned}
 & (\alpha_k \bar{\rho}_k)^{n+1} - (\alpha_k \bar{\rho}_k)^n = \\
 & = \Delta t \left( -\nabla_{\rho_k} \cdot \left( (\alpha_k \bar{\rho}_k)^{n+1} \tilde{\mathbf{v}}_k^{n+1} \right) + \underbrace{\vec{S}_{\text{mass},k}^n \cdot \begin{bmatrix} 1 \\ (\alpha_k \bar{\rho}_k)^{n+1} \\ \tilde{\mathbf{v}}_k^{n+1} \\ \widehat{\tilde{u}}_k^{n+1} \end{bmatrix}}_{\text{Linearization (if non-linear)}} \right) \quad (4.15)
 \end{aligned}$$

and

**Total energy**,  $k=0,1$ :

$$\begin{aligned}
 & (\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1} - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n = \\
 & \Delta t \left( -\nabla_{e_k} \cdot \left( \tilde{\mathbf{v}}_k^{n+1} \left( (\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1} + \widehat{\alpha}_k^{n+1} \bar{p}^{n+1} \right) \right) + \right. \\
 & \quad \left. + \tilde{\mathbf{v}}_k^{n+1} \bar{p}^{n+1} \nabla_{e_k} \widehat{\alpha}_k^{n+1} + \underbrace{\vec{S}_{\text{ene},k}^n \cdot \begin{bmatrix} 1 \\ (\alpha_k \bar{\rho}_k)^{n+1} \\ \tilde{\mathbf{v}}_k^{n+1} \\ (\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1} \end{bmatrix}}_{\text{Linearization (if non-linear)}} \right) \quad (4.16)
 \end{aligned}$$

It can be seen that advection terms for both mass and energy equations are evaluated at the new time level  $(n + 1)$ , which together with effectively implicit treat-

## 4.2. FRACTIONAL STEP BASED VARIATIONS

37

ment of advection for momentum equations should eliminate material CFL stability limits. Eqs.(4.15) and (4.16) must be solved with some appropriate direct or iterative implicit solver.

Next, specific internal energies and densities for each phase are evaluated as

$$\tilde{u}_k^{n+1} = \frac{(\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1}}{(\alpha_k \bar{\rho}_k)^{n+1}} - \frac{(\tilde{\mathbf{v}}_k^{n+1})^2}{2}; \text{ and } \bar{\rho}_k^{n+1} = \mathcal{EOS}(\bar{p}^{n+1}, \tilde{u}_k^{n+1})$$

Finally, new-time void fraction is evaluated using density and phasic density for the “continuous” phase:

$$\alpha_c^{n+1} = \frac{(\alpha_c \bar{\rho}_c)^{n+1}}{\bar{\rho}_c^{n+1}}$$

while the void fraction of the “dispersed” phase is computed from the “compatibility” constraint:

$$\alpha_d^{n+1} = 1 - \alpha_c^{n+1}$$

### 4.2.2 Nearly-implicit algorithm

Nearly-Implicit algorithm was introduced by Trapp and Riemke in [TR86], and it is RELAP5’s counterpart of the TRAC/TRACE’s SETS algorithm. In fact, this particular fractional step variation was mentioned in the original Mahafee’s paper [Mah82] as a variation of the SETS algorithm.

Nearly-Implicit algorithm collapses the first two steps of the SETS into a single one. This is achieved as follows.

#### Step I: Implicit velocity equations (= Step I and II of SETS)

Momentum advection of phasic momentum equations is treated implicitly, by using the following linearization of  $[\tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k]$ :

$$[\tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k] \rightarrow \tilde{\mathbf{v}}_k^n \otimes \tilde{\mathbf{v}}_k^{n+1} \quad (4.17)$$

Thus, eq.(4.6) becomes:

**Momentum**,  $k=0,1$ :

$$(\alpha_k \bar{\rho}_k)^n \tilde{\mathbf{v}}_k^{n+1} - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n = \Delta t \left[ -\nabla_{\mathbf{v}_k} \cdot \left( \underbrace{(\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n \otimes \tilde{\mathbf{v}}_k^{n+1}}_{\text{Modification}} \right) + \alpha_k^n \nabla_{\mathbf{v}_k} (\bar{p}^{n+1}) + \underbrace{\mathbb{S}_{\text{mom},k}^n \begin{bmatrix} 1 \\ \tilde{\mathbf{v}}_k^{n+1} \\ m=0,1 \end{bmatrix}}_{\text{Linearization (if non-linear)}} \right] \quad (4.18)$$

With this, nice algebraic momentum equation form of the “semi-implicit” and SETS algorithms

$$\tilde{\mathbf{v}}_k^{n+1} = \mathcal{F}(\nabla_{\mathbf{v}_k} \bar{p}^{n+1})$$

is lost. Instead,

$$\tilde{\mathbf{v}}_k^{n+1} = \mathcal{F}(\nabla_{\mathbf{v}_k} \bar{p}^{n+1}, \nabla_{\mathbf{v}_k} \cdot (\tilde{\mathbf{v}}_k^n \otimes \tilde{\mathbf{v}}_k^{n+1}))$$

which prevents forming pressure-Helmholtz equation along the lines of eq.(4.12). Therefore, another route with forming coupled velocity-Helmholtz equations is followed.

Similar to “semi-implicit” algorithms, linearization eq.(4.7) (RELAP5’s version) is utilized to eliminate “predictor” phasic densities  $\widehat{\bar{\rho}}_k^{n+1}$ , together with the “compatibility equation”  $[\alpha_1 = 1 - \alpha_0]$ . Collecting terms, the following mass-energy system can be formed:

$$\underbrace{\begin{bmatrix} A_{\alpha_0, \alpha_0}^n & A_{\alpha_0, \tilde{u}_0}^n & A_{\alpha_0, \tilde{u}_1}^n & A_{\alpha_0, \bar{p}}^n \\ A_{\tilde{u}_0, \alpha_0}^n & A_{\tilde{u}_0, \tilde{u}_0}^n & A_{\tilde{u}_0, \tilde{u}_1}^n & A_{\tilde{u}_0, \bar{p}}^n \\ A_{\tilde{u}_1, \alpha_0}^n & A_{\tilde{u}_1, \tilde{u}_0}^n & A_{\tilde{u}_1, \tilde{u}_1}^n & A_{\tilde{u}_1, \bar{p}}^n \\ A_{\bar{p}, \alpha_0}^n & A_{\bar{p}, \tilde{u}_0}^n & A_{\bar{p}, \tilde{u}_1}^n & A_{\bar{p}, \bar{p}}^n \end{bmatrix}}_{\mathbb{A}} \underbrace{\begin{bmatrix} \widehat{\alpha_0^{n+1}} \\ \widehat{\tilde{u}_0^{n+1}} \\ \widehat{\tilde{u}_1^{n+1}} \\ \widehat{\bar{p}^{n+1}} \end{bmatrix}}_{\mathbf{X}} = \mathbf{b} \quad (4.19)$$

where  $\mathbb{A}$  is mass-energy matrix with known coefficients evaluated using the state  $(\cdot)^n$  (these are obviously different from those of eq.(4.9)).

## 4.2. FRACTIONAL STEP BASED VARIATIONS

39

The r.h.s. of eq.(4.19) contains velocity divergences:

$$\mathbf{b} = \begin{bmatrix} \gamma_{\bar{\rho}_0}^n + \xi_{\bar{\rho}_0}^n \cdot \tilde{\mathbf{v}}_0^{n+1} + \zeta_{\bar{\rho}_0}^n \cdot \tilde{\mathbf{v}}_1^{n+1} + \eta_{\bar{\rho}_0}^n \nabla_{\rho_0} \cdot \tilde{\mathbf{v}}_0^{n+1} \\ \gamma_{\bar{\rho}_1}^n + \xi_{\bar{\rho}_1}^n \cdot \tilde{\mathbf{v}}_0^{n+1} + \zeta_{\bar{\rho}_1}^n \cdot \tilde{\mathbf{v}}_1^{n+1} + \eta_{\bar{\rho}_1}^n \nabla_{\rho_1} \cdot \tilde{\mathbf{v}}_1^{n+1} \\ \gamma_{\bar{e}_0}^n + \xi_{\bar{e}_0}^n \cdot \tilde{\mathbf{v}}_0^{n+1} + \zeta_{\bar{e}_0}^n \cdot \tilde{\mathbf{v}}_1^{n+1} + \eta_{\bar{e}_0}^n \nabla_{e_0} \cdot \tilde{\mathbf{v}}_0^{n+1} \\ \gamma_{\bar{e}_1}^n + \xi_{\bar{e}_1}^n \cdot \tilde{\mathbf{v}}_0^{n+1} + \zeta_{\bar{e}_1}^n \cdot \tilde{\mathbf{v}}_1^{n+1} + \eta_{\bar{e}_1}^n \nabla_{e_1} \cdot \tilde{\mathbf{v}}_1^{n+1} \end{bmatrix} \quad (4.20)$$

where  $\gamma_{z=\bar{\rho}_k, \bar{e}_k}^n$ ,  $\xi_{z=\bar{\rho}_k, \bar{e}_k}^n$  and  $\zeta_{z=\bar{\rho}_k, \bar{e}_k}^n$  are known coefficients evaluated using the state  $(\cdot)^n$  ( $\gamma_{z=\bar{\rho}_k, \bar{e}_k}^n$  are obviously different from those in eq.(4.10)).

Next, similar to the “semi-implicit” algorithm,  $\widehat{\alpha}_0^{n+1}$ ,  $\widehat{u}_0^{n+1}$  and  $\widehat{u}_1^{n+1}$  can be eliminated by inverting matrix  $\mathbb{A}$  (this is done at each cell of the computational domain) and multiplying  $\mathbb{A}^{-1}$  on the left and right side of eq.(4.19). The resulting pressure is the last (fourth) equation in

$$\mathbb{A}^{-1} \mathbf{b} - \mathbf{X} = 0 \quad (4.21)$$

It can be presented in the following form<sup>15</sup>:

$$\begin{aligned} \bar{p}^{n+1} = & a + \mathbf{b} \cdot \tilde{\mathbf{v}}_0^{n+1} + \mathbf{c} \cdot \tilde{\mathbf{v}}_1^{n+1} + \\ & + e \nabla_{\rho_0} \cdot \tilde{\mathbf{v}}_0^{n+1} + f \nabla_{\rho_1} \cdot \tilde{\mathbf{v}}_1^{n+1} + g \nabla_{e_0} \cdot \tilde{\mathbf{v}}_0^{n+1} + h \nabla_{e_1} \cdot \tilde{\mathbf{v}}_1^{n+1} \end{aligned} \quad (4.22)$$

where  $(a-h)$  are constants evaluated using the state  $(\cdot)^n$ .

The last step is to take gradient of eq.(4.22) and to plug it into the phasic momentum equation (4.18), thereby eliminating pressure. The resulting velocity-Helmholtz<sup>16</sup> system can be solved (with direct or iterative solver) for  $\tilde{\mathbf{v}}_k^{n+1}$ . These new velocities can be plugged into eq.(4.22) to compute new pressure,  $\bar{p}^{n+1}$ . Finally,  $\widehat{\alpha}_0^{n+1}$ ,  $\widehat{u}_0^{n+1}$  and  $\widehat{u}_1^{n+1}$  are computed from the first three equations of (4.21). This concludes the first step of the “Nearly-Implicit” algorithm.

### Step II: Stabilizer for phasic mass and energy equations

This step is exactly the same as for the TRAC/TRACE’s SETS algorithm.

<sup>15</sup>As in the case of eq.(4.12), we keep divergence operators different, leaving opportunity to incorporate approximate Riemann solvers.

<sup>16</sup>These equations include parabolic terms.

### 4.2.3 Remarks

1. Both *SETS* and “*Nearly-Implicit*” algorithms are only first-order accurate in time. The only possible way to increase the order of time discretization is by implementing outer iteration loop in a “segregated” algorithm manner, Section 5.2, with *a*) proper time-centering terms, *b*) bringing time-differences to the consistent level, and *c*) removing operator-split “mixed-time” treatments, which are all over the place in the original algorithms. But this of course will defeat the whole purpose of using fractional steps, which aim on achieving stability for lesser cost than in the case of fully-implicit treatment.
2. Both *SETS* and “*Nearly-Implicit*” algorithms suffer from the same mass and energy conservation errors for the “dispersed” phase, as in the case of the “*semi-implicit*” algorithm.
3. Even though material CFL stability limit was eliminated, both *SETS* and “*Nearly-Implicit*” algorithms do not treat wall heat transfer and friction implicitly. This is why the methods are referred to as “Stability-enhancing” and “Nearly-Implicit”.
4. It is difficult to say which fractional-step algorithm is more efficient. On one hand, “*Nearly-Implicit*” algorithm involves  $5N$  implicit solves (in 3D  $N$ -field formulation), while *SETS* would require  $5N + N_{\text{SI-iters}}$  implicit solves (where  $N_{\text{SI-iters}}$  is the number of non-linear iterations on the pressure-Helmholtz equation of the “*semi-implicit*” step). On the other hand, implicit velocity equations of *SETS* are much simpler than velocity-Helmholtz equations of “*Nearly-Implicit*”.

### 4.3 Incremental form of Semi-Implicit-based algorithms

In the present section, we re-formulate “semi-implicit” algorithm in the *incremental form*, to make it easier to compare with SIMPLE-based algorithms described in Section 5.1, and prepare a convenient form for using it either within Picard iterations of the segregated algorithm (Section 5.2), or as “physics-based” preconditioning of Section 6.5.2.

We start with writing phasic mass, energy and momentum conservation equations in the following non-linear form:

**Mass**,  $[k=0, N-1]$ :

$$[\alpha_k^{**} \bar{\rho}_k^{**} - (\alpha_k \bar{\rho}_k)^n] = \Delta t \left( -\nabla_{\rho_k} \cdot (\alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**}) + \mathcal{S}_{\text{mass},k}^{**} \right) \quad (4.23)$$

**Total energy**,  $[k=0, N-1]$ :

$$\begin{aligned} [\alpha_k^{**} \bar{\rho}_k^{**} \tilde{e}_k^{**} - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n] = \\ = \Delta t \left( -\nabla_{e_k} \cdot (\tilde{\mathbf{v}}_k^{**} \alpha_k^{**} (\bar{\rho}_k^{**} \tilde{e}_k^{**} + \bar{p}^{**} + \Delta \bar{p}_k^{**})) + \right. \\ \left. + \tilde{\mathbf{v}}_k^{**} \cdot ((\bar{p}^{**} - \delta \bar{p}_1^{**}) \nabla_{e_k} \alpha_k^{**}) + \mathcal{S}_{\text{ene},k}^{**} \right) \end{aligned} \quad (4.24)$$

**Momentum**,  $[k=0, N-1]$ :

$$\begin{aligned} \alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**} - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n = \Delta t \left[ -\nabla_{\mathbf{v}_k} \cdot (\alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**} \otimes \tilde{\mathbf{v}}_k^{**}) + \right. \\ \left. + \alpha_k^{**} \nabla_{\mathbf{v}_k} (\bar{p}^{**} + \Delta \bar{p}_k^{**}) - (\Delta \bar{p}_k^{**} + \delta \bar{p}_1^{**}) \nabla_{\mathbf{v}_k} \alpha_k^{**} + \mathcal{S}_{\text{mom},k}^{**} \right] \end{aligned} \quad (4.25)$$

where by  $(\cdot)^{**}$  we denote new-update values, which will be represented in the following incremental form:

$$\begin{aligned} \bar{p}^{**} &= \bar{p}^* + \bar{p}' \\ \bar{\rho}_k^{**} &= \bar{\rho}_k^* + \bar{\rho}'_k \\ \tilde{\mathbf{u}}_k^{**} &= \tilde{\mathbf{u}}_k^* + \tilde{\mathbf{u}}'_k \\ \alpha_k^{**} &= \alpha_k^* + \alpha'_k \\ \tilde{\mathbf{v}}_k^{**} &= \tilde{\mathbf{v}}_k^* + \tilde{\mathbf{v}}'_k \end{aligned} \quad (4.26)$$

with  $(\cdot)^*$  denoting current (Picard- or Newton-) iteration value of the variable<sup>17</sup>.

### 4.3.1 Linearization of sources

As our next step, we define the following generic linearization form of source terms:

$$\mathcal{S}_{\text{mass},k}^{**} = \frac{1}{\Delta t} \left( \begin{array}{l} \gamma_{\rho_k}^* + \sum_{m=0}^{N-1} \mu_{\rho(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\rho(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \\ + \sum_{m=0}^{N-1} \eta_{\rho(m,k)}^* \tilde{u}'_m + \nu_{\rho_k}^* \bar{p}' \end{array} \right) \quad (4.27)$$

$$\mathcal{S}_{\text{ene},k}^{**} = \frac{1}{\Delta t} \left( \begin{array}{l} \gamma_{e_k}^* + \sum_{m=0}^{N-1} \mu_{e(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \\ + \sum_{m=0}^{N-1} \eta_{e(m,k)}^* \tilde{u}'_m + \nu_{e_k}^* \bar{p}' \end{array} \right) \quad (4.28)$$

$$\mathcal{S}_{\text{mom},k}^{**} = \frac{1}{\Delta t} \left( \begin{array}{l} \gamma_{v_k}^* + \sum_{m=0}^{N-1} \mu_{v(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{v(m,k)}^* \tilde{\mathbf{v}}'_m + \\ + \sum_{m=0}^{N-1} \eta_{v(m,k)}^* \tilde{u}'_m + \nu_{v_k}^* \bar{p}' \end{array} \right) \quad (4.29)$$

Similarly, let's define the following generic linearization of the *phasic bulk pressure differences*:

$$\Delta \bar{p}_k^{**} = \left( \begin{array}{l} \gamma_{\Delta \bar{p}_k}^* + \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \\ + \sum_{m=0}^{N-1} \eta_{\Delta \bar{p}(m,k)}^* \tilde{u}'_m + \nu_{\Delta \bar{p}_k}^* \bar{p}' \end{array} \right) \quad (4.30)$$

and *dynamic interfacial pressure*:

$$\delta \bar{p}_1^{**} = \left( \begin{array}{l} \gamma_{\delta \bar{p}_1}^* + \sum_{m=0}^{N-1} \mu_{\delta \bar{p}_1(m)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* \cdot \tilde{\mathbf{v}}'_m + \\ + \sum_{m=0}^{N-1} \eta_{\delta \bar{p}_1(m)}^* \tilde{u}'_m + \nu_{\delta \bar{p}_1}^* \bar{p}' \end{array} \right) \quad (4.31)$$

<sup>17</sup>When used in an operator-splitting mode,  $(\cdot)^* = (\cdot)^n$ .

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 43

Accounting for energy phase coupling (coefficients  $\eta_{(\times)}^*$ ) is particularly important, as this is the major contributor to the interfacial mass generation source eq.(2.15), appearing in closures for all conservation equations. In 1D reactor thermalhydraulics, this term is typically modelled as

$$\Gamma = \frac{h_{w,i} (T_w - T_{\text{sat}}(\bar{p})) - a_i \left[ \begin{array}{l} h_{i,v} (T_{\text{sat}}(\bar{p}) - \tilde{T}_v(\tilde{u}_k, \bar{p})) \\ + h_{i,l} (T_{\text{sat}}(\bar{p}) - \tilde{T}_l(\tilde{u}_k, \bar{p})) \end{array} \right]}{H_{vl}(\bar{p})} \quad (4.32)$$

This term can be linearized as

$$\Gamma^{**} = \Gamma^* + \mathfrak{J}_v^* \tilde{T}'_v + \mathfrak{J}_l^* \tilde{T}'_l \quad (4.33)$$

with

$$\tilde{T}'_k = \tilde{p}' \left. \frac{\partial \tilde{T}}{\partial \bar{p}} \right|_{\tilde{u}_k = \text{const}}^* + \tilde{u}'_k \left. \frac{\partial \tilde{T}}{\partial \tilde{u}_k} \right|_{\bar{p} = \text{const}}^* \quad (4.34)$$

#### 4.3.2 Linearization of conservation laws

The next step is to plug eqs.(4.26)-(4.31) into eqs.(4.23)-(4.25), dropping all non-linear terms<sup>18</sup>:

**Mass**,  $[k=0, N-1]$ :

$$\underbrace{\alpha_k^{**} \bar{\rho}_k^{**}}_{\substack{\alpha_k^* \bar{\rho}_k^* + \alpha_k' \bar{\rho}_k^* \\ + \alpha_k^* \bar{\rho}_k' + \alpha_k' \bar{\rho}_k'}} - (\alpha_k \bar{\rho}_k)^n = -\Delta t \nabla_{\rho_k} \cdot \underbrace{(\alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**})}_{\substack{\alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \alpha_k' \bar{\rho}_k^* \tilde{\mathbf{v}}_k' + \\ + \alpha_k^* \bar{\rho}_k' \tilde{\mathbf{v}}_k^* + \alpha_k' \bar{\rho}_k' \tilde{\mathbf{v}}_k' + \\ + \times}} + \quad (4.35)$$

$$+ \gamma_{\rho_k}^* + \sum_{m=0}^{N-1} \mu_{\rho(m,k)}^* \alpha_m' + \sum_{m=0}^{N-1} \zeta_{\rho(m,k)}^* \cdot \tilde{\mathbf{v}}_m' + \sum_{m=0}^{N-1} \eta_{\rho(m,k)}^* \tilde{u}_m' + \nu_{\rho_k}^* \tilde{p}'$$

<sup>18</sup>Linearization of phasic mass, momentum and energy source terms will be introduced later.

**Total energy**,  $[k=0, N-1]$ :

$$\begin{aligned}
 & \underbrace{\alpha_k^{**} \bar{\rho}_k^{**} \tilde{e}_k^{**}}_{\alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{\rho}_k^* \tilde{e}'_k + \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{\rho}_k^* \tilde{e}'_k} - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n = \\
 & \quad \underbrace{\tilde{u}'_k + \tilde{\mathbf{v}}_k^* \cdot \tilde{\mathbf{v}}'_k}_{\alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{\rho}_k^* \tilde{e}'_k} + \\
 & = -\Delta t \nabla_{e_k} \cdot \left( \underbrace{\tilde{\mathbf{v}}_k^{**} \alpha_k^{**} (\bar{\rho}_k^{**} \tilde{e}_k^{**} + \bar{p}^{**} + \Delta \bar{p}_k^{**})}_{\tilde{\mathbf{v}}_k^* (\alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{\rho}_k^* (\tilde{u}'_k + \tilde{\mathbf{v}}_k^* \cdot \tilde{\mathbf{v}}'_k) + \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{\rho}_k^* \tilde{e}'_k) + \tilde{\mathbf{v}}'_k \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{p}^* \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{p}^* \tilde{\mathbf{v}}'_k + \alpha_k^* \bar{p}^* \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{p}^* \tilde{\mathbf{v}}'_k + \alpha_k^* \tilde{\mathbf{v}}_k^* \gamma_{\Delta \bar{p}_k}^* + \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha'_m + \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \eta_{\Delta \bar{p}(m,k)}^* \tilde{u}'_m + \alpha_k^* \tilde{\mathbf{v}}_k^* \nu_{\Delta \bar{p}_k}^* \bar{p}' + \alpha_k^* \tilde{\mathbf{v}}_k^* \gamma_{\Delta \bar{p}_k}^* + \alpha_k^* \tilde{\mathbf{v}}'_k \gamma_{\Delta \bar{p}_k}^*} \right) + \Delta t \tilde{\mathbf{v}}_k^{**} \cdot \left( \bar{p}^{**} \nabla_{e_k} \alpha_k^{**} \right) + \\
 & \quad \tilde{\mathbf{v}}_k^* \cdot \left( \bar{p}^* \nabla_{e_k} \alpha_k^* \right) + \tilde{\mathbf{v}}'_k \cdot \left( \bar{p}^* \nabla_{e_k} \alpha_k^* \right) + \tilde{\mathbf{v}}_k^* \cdot \left( \bar{p}^* \nabla_{e_k} \alpha'_k \right) + \tilde{\mathbf{v}}_k^* \cdot \left( \bar{p}' \nabla_{e_k} \alpha_k^* \right) + \\
 & -\Delta t \tilde{\mathbf{v}}_k^{**} \cdot \left( \delta \bar{p}_I^{**} \nabla_{e_k} \alpha_k^{**} \right) + \\
 & \quad \tilde{\mathbf{v}}_k^* \cdot \left( \left[ \gamma_{\delta \bar{p}_I}^* + \sum_{m=0}^{N-1} \mu_{\delta \bar{p}_I(m)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_I(m)}^* \cdot \tilde{\mathbf{v}}'_m + \sum_{m=0}^{N-1} \eta_{\delta \bar{p}_I(m)}^* \tilde{u}'_m + \nu_{\delta \bar{p}_I}^* \bar{p}' \right] \nabla_{e_k} \alpha_k^* \right) + \\
 & \quad + \tilde{\mathbf{v}}'_k \cdot \left( \gamma_{\delta \bar{p}_I}^* \nabla_{e_k} \alpha_k^* \right) + \tilde{\mathbf{v}}_k^* \cdot \left( \gamma_{\delta \bar{p}_I}^* \nabla_{e_k} \alpha'_k \right) + \\
 & \quad + \gamma_{e_k}^* + \sum_{m=0}^{N-1} \mu_{e(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \sum_{m=0}^{N-1} \eta_{e(m,k)}^* \tilde{u}'_m + \nu_{e_k}^* \bar{p}'
 \end{aligned} \tag{4.36}$$

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

45

**Momentum**,  $[k=0, N-1]$ :

$$\begin{aligned}
& \underbrace{\alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**}}_{\alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}'_k \tilde{\mathbf{v}}_k'} - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n = \\
& \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}'_k \tilde{\mathbf{v}}_k' + \\
& + \underbrace{\alpha_k' \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}'_k \tilde{\mathbf{v}}_k'}_{\alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}'_k \tilde{\mathbf{v}}_k' \otimes \tilde{\mathbf{v}}_k^*} + \times \\
= & -\Delta t \nabla_{\mathbf{v}_k} \cdot \left( \underbrace{\alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**} \otimes \tilde{\mathbf{v}}_k^{**}}_{\alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}'_k \tilde{\mathbf{v}}_k' \otimes \tilde{\mathbf{v}}_k^*} \right) + \Delta t \underbrace{\alpha_k^{**} \nabla_{\mathbf{v}_k} (\bar{p}^{**} + \Delta \bar{p}_k^{**})}_{\alpha_k^* \nabla_{\mathbf{v}_k} \bar{p}^* + \alpha_k^* \nabla_{\mathbf{v}_k} \bar{p}'} + \\
& + \underbrace{\alpha_k' \nabla_{\mathbf{v}_k} \bar{p}^*}_{\alpha_k' \nabla_{\mathbf{v}_k} \bar{p}^*} + \cancel{\alpha_k^* \nabla_{\mathbf{v}_k} \bar{p}'} + \\
& + \alpha_k^* \nabla_{\mathbf{v}_k} \left( \begin{array}{l} \gamma_{\Delta \bar{p}_k}^* + \\ + \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha'_m + \\ + \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \\ + \sum_{m=0}^{N-1} \eta_{\Delta \bar{p}(m,k)}^* \tilde{\mathbf{u}}'_m + \\ + \nu_{\Delta \bar{p}_k}^* \bar{p}' \end{array} \right) + \\
& + \underbrace{\alpha_k' \nabla_{\mathbf{v}_k} \gamma_{\Delta \bar{p}_k}^*}_{\alpha_k' \nabla_{\mathbf{v}_k} \gamma_{\Delta \bar{p}_k}^*} + \times \\
& - \Delta t \underbrace{(\Delta \bar{p}_k^{**} + \delta \bar{p}_1^{**}) \nabla_{\mathbf{v}_k} \alpha_k^{**}}_{\left( \begin{array}{l} (\gamma_{\Delta \bar{p}_k}^* + \gamma_{\delta \bar{p}_1}^*) + \sum_{m=0}^{N-1} (\mu_{\Delta \bar{p}(m,k)}^* + \mu_{\delta \bar{p}_1(m)}^*) \alpha'_m + \\ + \sum_{m=0}^{N-1} (\zeta_{\Delta \bar{p}(m,k)}^* + \zeta_{\delta \bar{p}_1(m)}^*) \cdot \tilde{\mathbf{v}}'_m + \\ + \sum_{m=0}^{N-1} (\eta_{\Delta \bar{p}(m,k)}^* + \eta_{\delta \bar{p}_1(m)}^*) \tilde{\mathbf{u}}'_m + (\nu_{\Delta \bar{p}_k}^* + \nu_{\delta \bar{p}_1}^*) \bar{p}' \end{array} \right) \nabla_{\mathbf{v}_k} \alpha_k^*} + \\
& + \underbrace{\left( \begin{array}{l} (\gamma_{\Delta \bar{p}_k}^* + \gamma_{\delta \bar{p}_1}^*) \nabla_{\mathbf{v}_k} \alpha_k' \\ + \gamma_{\mathbf{v}_k}^* + \sum_{m=0}^{N-1} \mu_{\mathbf{v}(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\mathbf{v}(m,k)}^* \tilde{\mathbf{v}}'_m + \sum_{m=0}^{N-1} \eta_{\mathbf{v}(m,k)}^* \tilde{\mathbf{u}}'_m + \nu_{\mathbf{v}_k}^* \bar{p}' \end{array} \right)}_{\left( \begin{array}{l} (\gamma_{\Delta \bar{p}_k}^* + \gamma_{\delta \bar{p}_1}^*) \nabla_{\mathbf{v}_k} \alpha_k' \\ + \gamma_{\mathbf{v}_k}^* + \sum_{m=0}^{N-1} \mu_{\mathbf{v}(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\mathbf{v}(m,k)}^* \tilde{\mathbf{v}}'_m + \sum_{m=0}^{N-1} \eta_{\mathbf{v}(m,k)}^* \tilde{\mathbf{u}}'_m + \nu_{\mathbf{v}_k}^* \bar{p}' \end{array} \right)} + \times
\end{aligned} \tag{4.37}$$

Note, that in order to be truly consistent with the original semi-implicit algorithm (Section 4.1), the terms in boxes must be zeroed. Keeping them would make the algorithm to be of SETS (and SIMPLE) flavour (see sections 4.2.1 and 5.1).

The next step is elimination of density correction from eqs.(4.35), (4.36) and

(4.37), by plugging ICE linearization of equation of state:

$$\bar{\rho}'_k = \underbrace{\bar{p}' \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \Big|_{\tilde{u}_k = \text{const}}}_{\bar{\rho}_{\bar{p}_k}^*} + \underbrace{\tilde{u}'_k \frac{\partial \bar{\rho}_k}{\partial \tilde{u}_k} \Big|_{\bar{p} = \text{const}}}_{\bar{\rho}_{\tilde{u}_k}^*} \quad (4.38)$$

rendering

**Mass**,  $[k=0,1,\dots,N-1]$ :

$$\begin{aligned} & \alpha'_k \bar{\rho}_k^* + \bar{p}' \alpha_k^* \bar{\rho}_{\bar{p}_k}^* + \tilde{u}'_k \alpha_k^* \bar{\rho}_{\tilde{u}_k}^* + \alpha_k^* \bar{\rho}_k^* - (\alpha_k \bar{\rho}_k)^n = \\ & \Delta t \left( -\nabla_{\rho_k} \cdot \left( \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}'_k + \alpha'_k \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \left( \bar{p}' \bar{\rho}_{\bar{p}_k}^* + \tilde{u}'_k \bar{\rho}_{\tilde{u}_k}^* \right) \alpha_k^* \tilde{\mathbf{v}}_k^* \right) + \right. \\ & \quad \left. + \gamma_{\rho_k}^* + \sum_{m=0}^{N-1} \mu_{\rho(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\rho(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \sum_{m=0}^{N-1} \eta_{\rho(m,k)}^* \tilde{u}'_m + \nu_{\rho_k}^* \bar{p}' \right) \end{aligned} \quad (4.39)$$

**Total energy**,  $[k=0,N-1]$ :

$$\begin{aligned} & \alpha_k^* \bar{\rho}_k^* (\tilde{u}'_k + \tilde{\mathbf{v}}_k^* \cdot \tilde{\mathbf{v}}'_k) + \alpha'_k \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \tilde{e}_k^* \left( \bar{p}' \bar{\rho}_{\bar{p}_k}^* + \tilde{u}'_k \bar{\rho}_{\tilde{u}_k}^* \right) + \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n = \\ & = \Delta t \left[ -\nabla_{e_k} \cdot \left( \begin{aligned} & \tilde{\mathbf{v}}_k^* \left( \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{\rho}_k^* (\tilde{u}'_k + \tilde{\mathbf{v}}_k^* \cdot \tilde{\mathbf{v}}'_k) + \right. \right. \\ & \left. \left. + \alpha'_k \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \tilde{e}_k^* \left( \bar{p}' \bar{\rho}_{\bar{p}_k}^* + \tilde{u}'_k \bar{\rho}_{\tilde{u}_k}^* \right) \right) + \right. \\ & \left. + \tilde{\mathbf{v}}'_k \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* + \alpha_k^* \bar{p}' \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{p}' \tilde{\mathbf{v}}'_k + \right. \\ & \left. + \alpha'_k \bar{p}' \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{p}' \tilde{\mathbf{v}}_k^* + \right. \\ & \left. + \alpha_k^* \tilde{\mathbf{v}}_k^* \gamma_{\Delta \bar{p}_k}^* + \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha'_m + \right. \\ & \left. + \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \eta_{\Delta \bar{p}(m,k)}^* \tilde{u}'_m + \right. \\ & \left. + \alpha_k^* \tilde{\mathbf{v}}_k^* \nu_{\Delta \bar{p}_k}^* \bar{p}' + \alpha'_k \tilde{\mathbf{v}}_k^* \gamma_{\Delta \bar{p}_k}^* + \alpha_k^* \tilde{\mathbf{v}}'_k \gamma_{\Delta \bar{p}_k}^* \right) \right] + \\ & + \tilde{\mathbf{v}}_k^* \cdot \left( \bar{p}^* \nabla_{e_k} \alpha_k^* \right) + \tilde{\mathbf{v}}_k^* \cdot \left( \bar{p}^* \nabla_{e_k} \alpha'_k \right) + \tilde{\mathbf{v}}_k^* \cdot \left( \bar{p}' \nabla_{e_k} \alpha_k^* \right) + \tilde{\mathbf{v}}'_k \cdot \left( \bar{p}^* \nabla_{e_k} \alpha_k^* \right) + \\ & + \tilde{\mathbf{v}}_k^* \cdot \left[ \begin{aligned} & \gamma_{\delta \bar{p}_1}^* + \sum_{m=0}^{N-1} \mu_{\delta \bar{p}_1(m)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* \cdot \tilde{\mathbf{v}}'_m + \right. \\ & \left. + \sum_{m=0}^{N-1} \eta_{\delta \bar{p}_1(m)}^* \tilde{u}'_m + \nu_{\delta \bar{p}_1}^* \bar{p}' \right] \nabla_{e_k} \alpha_k^* + \\ & + \tilde{\mathbf{v}}'_k \cdot \left( \gamma_{\delta \bar{p}_1}^* \nabla_{e_k} \alpha_k^* \right) + \tilde{\mathbf{v}}_k^* \cdot \left( \gamma_{\delta \bar{p}_1}^* \nabla_{e_k} \alpha'_k \right) \Big] + \\ & + \gamma_{e_k}^* + \sum_{m=0}^{N-1} \mu_{e(m,k)}^* \alpha'_m + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \sum_{m=0}^{N-1} \eta_{e(m,k)}^* \tilde{u}'_m + \nu_{e_k}^* \bar{p}' \end{aligned} \quad (4.40)$$

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

47

and

**Momentum**,  $[k=0, N-1]$ :

$$\begin{aligned}
& \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k' + \alpha_k' \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \bar{p}' \alpha_k^* \bar{\rho}_{\bar{p}_k}^* \tilde{\mathbf{v}}_k^* + \tilde{u}_k' \alpha_k^* \bar{\rho}_{\tilde{u}_k}^* \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n = \\
& = \Delta t \left[ -\nabla_{\mathbf{v}_k} \cdot \left( \begin{aligned} & \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* + \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k' + \\ & + \alpha_k' \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* + \alpha_k^* \left( \bar{p}' \bar{\rho}_{\bar{p}_k}^* + \tilde{u}_k' \bar{\rho}_{\tilde{u}_k}^* \right) \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* + \\ & + \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k' \otimes \tilde{\mathbf{v}}_k^* \end{aligned} \right) + \right. \\
& \quad + \alpha_k^* \nabla_{\mathbf{v}_k} \bar{p}^* + \alpha_k^* \nabla_{\mathbf{v}_k} \bar{p}' + \alpha_k' \nabla_{\mathbf{v}_k} \bar{p}^* + \\
& \quad + \alpha_k^* \nabla_{\mathbf{v}_k} \left( \begin{aligned} & \gamma_{\Delta \bar{p}_k}^* + \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha_m' + \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \tilde{\mathbf{v}}_m' + \\ & + \sum_{m=0}^{N-1} \eta_{\Delta \bar{p}(m,k)}^* \tilde{u}_m' + \nu_{\Delta \bar{p}_k}^* \bar{p}' \end{aligned} \right) + \quad (4.41) \\
& \quad + \alpha_k' \nabla_{\mathbf{v}_k} \gamma_{\Delta \bar{p}_k}^* - \left( \gamma_{\Delta \bar{p}_k}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{\mathbf{v}_k} \alpha_k' \\
& \quad - \left( \begin{aligned} & \left( \gamma_{\Delta \bar{p}_k}^* + \gamma_{\delta \bar{p}_1}^* \right) + \sum_{m=0}^{N-1} \left( \mu_{\Delta \bar{p}(m,k)}^* + \mu_{\delta \bar{p}_1(m)}^* \right) \alpha_m' + \\ & + \sum_{m=0}^{N-1} \left( \zeta_{\Delta \bar{p}(m,k)}^* + \zeta_{\delta \bar{p}_1(m)}^* \right) \cdot \tilde{\mathbf{v}}_m' + \\ & + \sum_{m=0}^{N-1} \left( \eta_{\Delta \bar{p}(m,k)}^* + \eta_{\delta \bar{p}_1(m)}^* \right) \tilde{u}_m' + \left( \nu_{\Delta \bar{p}_k}^* + \nu_{\delta \bar{p}_1}^* \right) \bar{p}' \end{aligned} \right) \nabla_{\mathbf{v}_k} \alpha_k^* \right] + \\
& \quad + \gamma_{\mathbf{v}_k}^* + \sum_{m=0}^{N-1} \mu_{\mathbf{v}(m,k)}^* \alpha_m' + \sum_{m=0}^{N-1} \zeta_{\mathbf{v}(m,k)}^* \tilde{\mathbf{v}}_m' + \sum_{m=0}^{N-1} \eta_{\mathbf{v}(m,k)}^* \tilde{u}_m' + \nu_{\mathbf{v}_k}^* \bar{p}'
\end{aligned}$$

### 4.3.3 Diagonalization of momentum equations

To diagonalize momentum equations, we first re-write eq.(4.41), collecting and grouping the terms as:

**Momentum**,  $[k=0, N-1]$ :

$$\begin{aligned} \left( \alpha_k^* \bar{\rho}_k^* - \zeta_{\mathbf{v}_k}^* \right) \tilde{\mathbf{v}}_k' - \sum_{m=0, \neq k}^{N-1} \zeta_{\mathbf{v}_k}^* \tilde{\mathbf{v}}_m' + \Delta t \left[ \sum_{m=0}^{N-1} \left( \zeta_{\Delta \bar{p}(m,k)}^* + \zeta_{\delta \bar{p}1(m)}^* \right) \cdot \tilde{\mathbf{v}}_m' \right] \nabla_{\mathbf{v}_k} \alpha_k^* = \end{aligned} \quad (4.42)$$

$$= -\mathfrak{D}'_{\mathbf{v}_k} + \mathbf{p}'_{\mathbf{v}_k} + \mathbf{a}'_{\mathbf{v}_k} + \mathbf{u}'_{\mathbf{v}_k} - \mathbf{r}_{\mathbf{v}_k}^*$$

where

$$\mathfrak{D}'_{\mathbf{v}_k} = \Delta t \left[ \nabla_{\mathbf{v}_k} \cdot \left( \alpha_k^* \bar{\rho}_k^* \left[ \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k' + \tilde{\mathbf{v}}_k' \otimes \tilde{\mathbf{v}}_k^* \right] \right) - \alpha_k^* \nabla_{\mathbf{v}_k} \sum_{m=0}^{N-1} \left( \zeta_{\Delta \bar{p}(m,k)}^* \cdot \tilde{\mathbf{v}}_m' \right) \right] \quad (4.43)$$

$$\begin{aligned} \mathbf{p}'_{\mathbf{v}_k} &= \underbrace{\left( \nu_{\mathbf{v}_k}^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* - \Delta t \left( \nu_{\Delta \bar{p}k}^* + \nu_{\delta \bar{p}1}^* \right) \nabla_{\mathbf{v}_k} \alpha_k^* \right)}_{1/\omega_k^*} \bar{p}' + \\ &+ \Delta t \underbrace{\left[ \alpha_k^* \nabla_{\mathbf{v}_k} \left( \left( 1 + \nu_{\Delta \bar{p}k}^* \right) \bar{p}' \right) - \nabla_{\mathbf{v}_k} \cdot \left( \alpha_k^* \bar{\rho}_k^* \bar{p}' \left( \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* \right) \right) \right]}_{\mathfrak{D}p'_k} = \end{aligned} \quad (4.44)$$

$$= \boxed{\frac{\bar{p}'}{\omega_k^*} + \mathfrak{D}p'_k}$$

$$\begin{aligned} \mathbf{a}'_{\mathbf{v}_k} &= \underbrace{\left( \mu_{\mathbf{v}_k(k)}^* + \Delta t \left[ \nabla_{\mathbf{v}_k} \left( \bar{p}^* + \gamma_{\Delta \bar{p}k}^* \right) - \left( \mu_{\Delta \bar{p}(k,k)}^* + \mu_{\delta \bar{p}1(k)}^* \right) \nabla_{\mathbf{v}_k} \alpha_k^* \right] - \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right)}_{\hat{\mu}_{\mathbf{v}_k(k)}^*} \alpha_k' \\ &+ \sum_{m=0, \neq k}^{N-1} \underbrace{\left( \mu_{\mathbf{v}_k(m,k)}^* - \Delta t \left( \mu_{\Delta \bar{p}(m,k)}^* + \mu_{\delta \bar{p}1(m)}^* \right) \nabla_{\mathbf{v}_k} \alpha_k^* \right)}_{\hat{\mu}_{\mathbf{v}_k(m,k)}^*} \alpha_m' + \\ &+ \Delta t \underbrace{\left[ \alpha_k^* \nabla_{\mathbf{v}_k} \left( \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha_m' \right) - \left( \gamma_{\Delta \bar{p}k}^* + \gamma_{\delta \bar{p}1}^* \right) \nabla_{\mathbf{v}_k} \alpha_k' - \nabla_{\mathbf{v}_k} \cdot \left( \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* \right) \right]}_{\mathfrak{D}\alpha'_k} = \end{aligned} \quad (4.45)$$

$$= \boxed{\hat{\mu}_{\mathbf{v}_k(k)}^* \alpha_k' + \sum_{m=0, \neq k}^{N-1} \hat{\mu}_{\mathbf{v}_k(m,k)}^* \alpha_m' + \mathfrak{D}\alpha'_k}$$

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 49

$$\begin{aligned}
 \mathbf{u}'_{\mathbf{v}_k} &= \underbrace{\left( \hat{\eta}_{\mathbf{v}_{(k)}}^* - \alpha_k^* \bar{\rho}_{\tilde{u}_k}^* \tilde{\mathbf{v}}_k^* \right)}_{\hat{\eta}_{\mathbf{v}_{(k)}}^*} \tilde{\mathbf{u}}'_k + \sum_{m=0, \neq k}^{N-1} \underbrace{\left[ \hat{\eta}_{\mathbf{v}_{(m,k)}}^* - \Delta t \left( \eta_{\Delta \bar{p}(m,k)}^* + \eta_{\delta \bar{p}I(m)}^* \right) \nabla_{\mathbf{v}_k} \alpha_k^* \right]}_{\hat{\eta}_{\mathbf{v}_{(m,k)}}^*} \tilde{\mathbf{u}}'_m + \\
 &\quad + \Delta t \underbrace{\left[ \alpha_k^* \nabla_{\mathbf{v}_k} \left( \eta_{\Delta \bar{p}(m,k)}^* \tilde{\mathbf{u}}'_m \right) - \nabla_{\mathbf{v}_k} \cdot \left( \bar{\rho}_{\tilde{u}_k}^* \alpha_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* \tilde{\mathbf{u}}'_k \right) \right]}_{\mathbf{d}\mathbf{u}'_k} = \\
 &= \hat{\eta}_{\mathbf{v}_{(k)}}^* \tilde{\mathbf{u}}'_k + \sum_{m=0, \neq k}^{N-1} \hat{\eta}_{\mathbf{v}_{(m,k)}}^* \tilde{\mathbf{u}}'_m + \mathbf{d}\mathbf{u}'_k
 \end{aligned} \tag{4.46}$$

and

$$\begin{aligned}
 \mathbf{r}_{\mathbf{v}_k}^* &= \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* - (\alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^*)^n - \gamma_{\mathbf{v}_k}^* + \\
 &\quad + \Delta t \left[ \nabla_{\mathbf{v}_k} \cdot \left( \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \tilde{\mathbf{v}}_k^* \right) - \alpha_k^* \nabla_{\mathbf{v}_k} \left( \bar{p}^* + \gamma_{\Delta \bar{p}k}^* \right) + \left( \gamma_{\Delta \bar{p}k}^* + \gamma_{\delta \bar{p}I}^* \right) \nabla_{\mathbf{v}_k} \alpha_k^* \right]
 \end{aligned} \tag{4.47}$$

is a non-linear momentum residual vector, evaluated at state  $\mathbf{V}^*$ .

Eq.(4.42) is a system of  $(d \times N)^{19}$  coupled equations for  $\tilde{\mathbf{v}}'_k$ , where  $d$  is the number of dimensions.

We can now re-write eq.(4.42) in the following compact form:

$$\mathfrak{F} \begin{bmatrix} \dots \\ \dots \\ \tilde{\mathbf{v}}'_m \\ \dots \\ \dots \end{bmatrix} = \begin{bmatrix} \dots \\ \dots \\ \mathbf{m}'_m \\ \dots \\ \dots \end{bmatrix}_{m=0, \dots, N} \tag{4.48}$$

<sup>19</sup>We tacitly presume finite-volume (one degree of freedom per cell-variable) based discretization. Though the approach shall be straightforwardly expandable to finite-element/Discontinuous Galerkin forms as well, with several degrees of freedom per cell-variable.

where

$$\begin{aligned}
 \mathbf{m}'_m = & \frac{\bar{p}'}{\omega_m^*} + \hat{\boldsymbol{\mu}}_{\mathbf{v}(m)}^* \alpha'_m + \sum_{j=0, \neq m}^{N-1} \hat{\boldsymbol{\mu}}_{\mathbf{v}(j,m)}^* \alpha'_j + \hat{\boldsymbol{\eta}}_{\mathbf{v}(m)}^* \tilde{u}'_m + \sum_{j=0, \neq m}^{N-1} \hat{\boldsymbol{\eta}}_{\mathbf{v}(j,m)}^* \tilde{u}'_j + \\
 & + \boldsymbol{\mathcal{D}}'_{\mathbf{v}_m} (\nabla \bar{p}') + \boldsymbol{\mathcal{D}} \alpha'_m (\nabla \alpha'_j) + \boldsymbol{\mathcal{D}} \mathbf{u}'_m (\nabla \tilde{u}'_j) - \\
 & - \underbrace{\left( \boldsymbol{\mathcal{D}}'_{\mathbf{v}_m} (\nabla \cdot \tilde{\mathbf{v}}'_j) + \mathbf{r}_{\mathbf{v}_m}^* \right)}_{\text{(combined)} \rightarrow \mathbf{r}_{\mathbf{v}_m}^*}
 \end{aligned} \tag{4.49}$$

and  $\mathfrak{F}$  is  $(N \times N)$  friction coefficient matrix, with each element being  $(d \times d)$  blocks. Off-diagonal blocks represent local inter-phase momentum coupling, which can be eliminated by writing:

$$\tilde{\mathbf{v}}'_k = \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \begin{array}{l} \frac{\bar{p}'}{\omega_m^*} + \hat{\boldsymbol{\mu}}_{\mathbf{v}(m)}^* \alpha'_m + \sum_{j=0, \neq m}^{N-1} \hat{\boldsymbol{\mu}}_{\mathbf{v}(j,m)}^* \alpha'_j + \\ + \hat{\boldsymbol{\eta}}_{\mathbf{v}(m)}^* \tilde{u}'_m + \sum_{j=0, \neq m}^{N-1} \hat{\boldsymbol{\eta}}_{\mathbf{v}(j,m)}^* \tilde{u}'_j + \\ + \boldsymbol{\mathcal{D}}'_{\mathbf{v}_m} + \boldsymbol{\mathcal{D}} \alpha'_m + \boldsymbol{\mathcal{D}} \mathbf{u}'_m - \left( \boldsymbol{\mathcal{D}}'_{\mathbf{v}_m} + \mathbf{r}_{\mathbf{v}_m}^* \right) \end{array} \right) \tag{4.50}$$

where  $\mathfrak{F}_{(k,m)}^{-1}$  represents  $_{(k,m)}$ <sup>(th)</sup> block (each of size  $(d \times d)$ ) of matrix  $\mathfrak{F}^{-1}$ .

Eq.(4.50) is still non-diagonal, due to terms associated with spatial variation of velocity corrections,  $\boldsymbol{\mathcal{D}}'_{\mathbf{v}_k}$ . To eliminate these off-diagonal terms, a number of options can be implemented.

1. **OS.** Simply ignore these terms,  $\boldsymbol{\mathcal{D}}'_{\mathbf{v}_k} = 0$ . This is what would be the closest to the original semi-implicit (operator-splitting, OS) algorithm.
2. **SIMPLE-like.** For this, we replace  $\boldsymbol{\mathcal{D}}'_{\mathbf{v}_k}$  by its discrete form<sup>20</sup>:

$$\boldsymbol{\mathcal{D}}'_{\mathbf{v}_k} \rightarrow \boldsymbol{\mathcal{A}}_{\mathbf{v},c}^{(k)} \tilde{\mathbf{v}}'_k + \sum_n \boldsymbol{\mathcal{A}}_{\mathbf{v},n}^{(k)} \tilde{\mathbf{v}}'_{k(n)} \quad \text{ignore} \tag{4.51}$$

<sup>20</sup>To be sufficiently generic, we avoid discussion of specific forms of spatial discretization.

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 51

where  $\mathfrak{A}_{v_{i,j}}^{(k)}$  are  $(d \times d)$  matrices with space discretization coefficients,  $c$  denotes cell id,  $n$  – neighboring cells, while  $N$  is the number of neighboring cells involved. By ignoring contributions from neighbor cells<sup>21</sup>, we remove all off-diagonal contributions from spatial variations of velocity corrections. Coefficient  $\mathfrak{A}_{v_{c,c}}^{(k)}$  should be absorbed into the diagonal of the friction coefficient matrix  $\mathfrak{F}$ .

3. **Diagonal stabilizer.** In this option, one would solve for “predictor” value of velocity correction  $\hat{\mathbf{v}}'_k$ , using the following “explicit” sweep:

$$\hat{\mathbf{v}}'_k = - \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \mathbf{r}_{v_m}^* \quad (4.52)$$

Now,  $\hat{\mathbf{v}}'_k$  can be used to compute  $\mathfrak{D}'_{v_k} (\nabla \cdot \hat{\mathbf{v}}'_j)$ , which is absorbed into  $\mathbf{r}_{v_m}^*$  as:

$$\mathbf{r}_{v_m}^* \rightarrow \boxed{\mathbf{r}_{v_m}^* + \mathfrak{D}'_{v_k} (\nabla \cdot \hat{\mathbf{v}}'_m)} \quad (4.53)$$

4. **Implicit stabilizer.** In this option, the following PDE is solved implicitly for  $\hat{\mathbf{v}}'_k$ :

$$\hat{\mathbf{v}}'_k + \mathfrak{D}'_{v_k} (\nabla \cdot \hat{\mathbf{v}}'_k) = - \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \mathbf{r}_{v_m}^* \quad (4.54)$$

This option is close to what is done in the “velocity stabilizer” step of SETS (Section 4.2.1). If there is a viscosity operator<sup>22</sup>, this option is a must to eliminate stability limits associated with (turbulent) viscosity-Fourier number, in addition to breaking material Courant stability restrictions. Predictor velocity  $\hat{\mathbf{v}}'_k$  can now be used to compute  $\mathfrak{D}'_{v_k} (\nabla \cdot \hat{\mathbf{v}}'_k)$ , which is absorbed into  $\mathbf{r}_{v_m}^*$  as in eq.(4.53).

We can now plug eq.(4.50) into mass and energy conservation equations (4.39) and (4.40), to get a system of  $(2 \times N)$  coupled *pressure-correction Helmholtz equations (P'HE)*.

<sup>21</sup>When the method is used in an iterative algorithm – segregated- or JFNK-based, this term is zero upon convergence, and ignoring these contributions should not affect the final result.

<sup>22</sup>Ignored in the discussion here, for simplicity.

### 4.3.4 Mass P' HE

First, re-group eq.(4.39) as

**Mass**,  $[k=0,1,\dots,N-1]$ :

$$\begin{aligned}
& \frac{\bar{\rho}_k^* - \mu_{\rho(k,k)}^*}{\Delta t^2} \alpha'_k - \sum_{m=0, \neq k}^{N-1} \frac{\mu_{\rho(m,k)}^*}{\Delta t^2} \alpha'_m + \\
& + \frac{\alpha_k^* \bar{\rho}_{\tilde{u}_k}^* - \eta_{\rho(k,k)}^*}{\Delta t^2} \tilde{u}'_k - \sum_{m=0, \neq k}^{N-1} \frac{\eta_{\rho(m,k)}^*}{\Delta t^2} \tilde{u}'_m + \left( \frac{\alpha_k^* \bar{\rho}_{\bar{p}_k}^* - \nu_{\rho k}^*}{\Delta t^2} \right) \bar{p}' = \\
& = -\frac{1}{\Delta t^2} \left[ \alpha_k^* \bar{\rho}_k^* - (\alpha_k \bar{\rho}_k)^n + \Delta t \nabla_{\rho k} \cdot (\alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^*) - \gamma_{\rho k}^* \right] - \\
& \quad - \frac{1}{\Delta t} \nabla_{\rho k} \cdot \left( \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \alpha'_k + \bar{\rho}_{\tilde{u}_k}^* \alpha_k^* \tilde{\mathbf{v}}_k^* \tilde{u}'_k + \bar{\rho}_{\bar{p}_k}^* \alpha_k^* \tilde{\mathbf{v}}_k^* \bar{p}' \right) + \\
& \quad + \frac{1}{\Delta t^2} \sum_{m=0}^{N-1} \zeta_{\rho(m,k)}^* \cdot \tilde{\mathbf{v}}'_m - \frac{1}{\Delta t} \nabla_{\rho k} \cdot (\alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}'_k)
\end{aligned} \tag{4.55}$$

and plug eq.(4.50) to replace velocity corrections:

**Mass**,  $[k=0,1,\dots,N-1]$ :

$$\begin{aligned}
& \underbrace{\sigma_{(\rho, \alpha')}^{(k)} \alpha'_k + \sigma_{(\rho, u')}^{(k)} \tilde{u}'_k + \sigma_{(\rho, p')}^{(k)} \bar{p}'}_{\text{in-phase coupling}} + \underbrace{\sum_{m=0, \neq k}^{N-1} \sigma_{(\rho, \alpha')}^{(k,m)} \alpha'_m + \sum_{m=0, \neq k}^{N-1} \sigma_{(\rho, u')}^{(k,m)} \tilde{u}'_m}_{\text{inter-phase coupling}} = \\
& = -\frac{\mathbf{v}_{\alpha_k}^*}{\Delta t^2} + \underbrace{\mathcal{L}_{\rho k} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(\rho, \alpha')}^{(k)} + \mathfrak{D}_{(\rho, u')}^{(k)}}_{\text{Pressure waves}}
\end{aligned} \tag{4.56}$$

where

$$\begin{aligned}
\sigma_{(\rho, \alpha')}^{(k)} &= \frac{1}{\Delta t^2} \left[ \bar{\rho}_k^* - \mu_{\rho(k,k)}^* - \sum_{m=0}^{N-1} \zeta_{\rho(m,k)}^* \cdot \left( \mathfrak{F}_{(m,k)}^{-1} \hat{\mu}_{\mathbf{v}(k)}^* + \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \hat{\mu}_{\mathbf{v}(k,i)}^* \right) \right] \\
\sigma_{(\rho, u')}^{(k)} &= \frac{1}{\Delta t^2} \left[ \alpha_k^* \bar{\rho}_{\tilde{u}_k}^* - \eta_{\rho(k,k)}^* - \sum_{m=0}^{N-1} \zeta_{\rho(m,k)}^* \cdot \left( \mathfrak{F}_{(m,k)}^{-1} \hat{\eta}_{\mathbf{v}(k)}^* + \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \hat{\eta}_{\mathbf{v}(k,i)}^* \right) \right] \\
\sigma_{(\rho, p')}^{(k)} &= \frac{1}{\Delta t^2} \left( \alpha_k^* \bar{\rho}_{\bar{p}_k}^* - \nu_{\rho k}^* - \sum_{m=0}^{N-1} \zeta_{\rho(m,k)}^* \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \frac{1}{\omega_i^*} \right)
\end{aligned} \tag{4.57}$$

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

53

are *in-phase* void/energy/pressure coupling elements of the  $(2 \times N)$  **mass-energy wavenumber matrix**,  $\tilde{\mathbb{W}}$ ;

$$\boxed{\sum_{m=0, \neq k}^{N-1} \sigma_{(\rho, \alpha')}^{(k, m)} \alpha'_m} = - \sum_{m=0, \neq k}^{N-1} \frac{\mu_{\rho(m, k)}^*}{\Delta t^2} \alpha'_m - \quad (4.58)$$

$$- \sum_{m=0}^{N-1} \left( \frac{\zeta_{\rho(m, k)}^*}{\Delta t^2} \cdot \left[ \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \hat{\mu}_{v(i)}^* \alpha'_i + \left( \sum_{i=0}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \sum_{j=0, \neq i, \neq k}^{N-1} \hat{\mu}_{v(j, i)}^* \alpha'_j \right) \right] \right)$$

$$\boxed{\sum_{m=0, \neq k}^{N-1} \sigma_{(\rho, u')}^{(k, m)} \tilde{u}'_m} = - \sum_{m=0, \neq k}^{N-1} \frac{\eta_{\rho(m, k)}^*}{\Delta t^2} \tilde{u}'_m - \quad (4.59)$$

$$- \sum_{m=0}^{N-1} \left( \frac{\zeta_{\rho(m, k)}^*}{\Delta t^2} \cdot \left[ \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \hat{\eta}_{v(i)}^* \tilde{u}'_i + \left( \sum_{i=0}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \sum_{j=0, \neq i, \neq k}^{N-1} \hat{\eta}_{v(j, i)}^* \tilde{u}'_j \right) \right] \right)$$

are *inter-phase* void/energy coupling elements of  $\hat{\mathbb{W}}$ ,

$$\boxed{\mathbf{r}_{\alpha_k}^*} = \alpha_k^* \bar{\rho}_k^* - (\alpha_k \bar{\rho}_k)^n + \Delta t \nabla_{\rho_k} \cdot \left( \alpha_k^* \bar{\rho}_k^* \left[ \tilde{\mathbf{v}}_k^* - \sum_{m=0}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \mathbf{r}_{v_m}^* \right] \right) - \quad (4.60)$$

$$- \gamma_{\rho_k}^* - \sum_{m=0}^{N-1} \zeta_{\rho(m, k)}^* \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \mathbf{r}_{v_i}^*$$

is non-linear residual of phasic mass, evaluated at state  $\mathbf{V}^*$ ,

$$\boxed{\mathcal{L}_{\rho_k}(\nabla \bar{p}', \nabla^2 \bar{p}')} = \frac{1}{\Delta t^2} \sum_{m=0}^{N-1} \zeta_{\rho(m, k)}^* \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \partial p'_i - \quad (4.61)$$

$$- \frac{1}{\Delta t} \nabla_{\rho_k} \cdot \left( \alpha_k^* \left[ \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* + \bar{\rho}_k^* \sum_{m=0}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \frac{1}{\omega_m^*} \right] \bar{p}' + \alpha_k^* \bar{\rho}_k^* \sum_{m=0}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \partial p'_m \right)$$

is Laplacian<sup>23</sup>, and

$$\boxed{\mathcal{D}_{(\rho, \alpha')}^{(k)}} = \sum_{m=0}^{N-1} \frac{\zeta_{\rho(m, k)}^*}{\Delta t^2} \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \partial \alpha'_i - \quad (4.62)$$

$$- \nabla_{\rho_k} \cdot \left( \frac{\bar{\rho}_k^*}{\Delta t} \left[ \tilde{\mathbf{v}}_k^* \alpha'_k + \alpha_k^* \sum_{m=0}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \left( \hat{\mu}_{v(m)}^* \alpha'_m + \sum_{j=0, \neq m}^{N-1} \hat{\mu}_{v(j, m)}^* \alpha'_j + \partial \alpha'_m \right) \right] \right)$$

<sup>23</sup>Strictly speaking, this operator is not exactly Laplacian, as it also includes pressure gradient terms.

$$\begin{aligned}
\mathfrak{D}_{(\rho, u')}^{(k)} &= \sum_{m=0}^{N-1} \frac{\zeta_{\rho(m,k)}^*}{\Delta t^2} \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \mathfrak{D}u'_i - \\
& - \nabla_{\rho_k} \cdot \left( \frac{\alpha_k^*}{\Delta t} \left[ \bar{\rho}_{\tilde{u}_k}^* \tilde{\mathbf{v}}_k^* \tilde{u}'_k + \bar{\rho}_k^* \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \hat{\eta}_{\mathbf{v}(m)}^* \tilde{u}'_m + \sum_{j=0, \neq m}^{N-1} \hat{\eta}_{\mathbf{v}(j,m)}^* \tilde{u}'_j + \mathfrak{D}u'_m \right) \right] \right) \quad (4.63)
\end{aligned}$$

are terms associated with spatial variations of void fraction and specific internal energy correction, correspondingly; similar to those of momentum equation (4.42). The first obvious choice would be simply ignore them (OS mode). We will discuss other possible treatments of these terms as *stabilizers* of fractional steps, in Section 4.3.7.

For the purpose of treating phase appearance and disappearance, it is important to introduce *conservation of mixture mass* equation as:

**Mixture mass:**

$$\begin{aligned}
& \sum_{n=0}^{N-1} \left( \sigma_{(\rho, \alpha')}^{(n)} \alpha'_n + \sum_{m=0, \neq n}^{N-1} \sigma_{(\rho, \alpha')}^{(n,m)} \alpha'_m \right) + \\
& + \sum_{n=0}^{N-1} \left( \sigma_{(\rho, u')}^{(n)} \tilde{u}'_n + \sum_{m=0, \neq n}^{N-1} \sigma_{(\rho, u')}^{(n,m)} \tilde{u}'_m \right) + \sum_{n=0}^{N-1} \sigma_{(\rho, p')}^{(n)} \bar{p}' = \\
& = \sum_{n=0}^{N-1} \left( \mathcal{L}_{\rho_n} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(\rho, \alpha')}^{(n)} + \mathfrak{D}_{(\rho, u')}^{(n)} - \frac{\mathbf{r}_{\alpha_n}^*}{\Delta t^2} \right) \quad (4.64)
\end{aligned}$$

This equation will be used to replace mass conservation for “minor” phase, as discussed in Section 4.3.6.

### 4.3.5 Energy P' HE

First, re-write eq.(4.40) in the following form:

**Total energy**,  $[k=0, N-1]$ :

$$\begin{aligned}
& \frac{\bar{\rho}_k^* \tilde{e}_k^* - \mu_{e(k,k)}^*}{\Delta t^2} - \frac{\Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \mu_{\delta \bar{p}_1(k)}^*}{\Delta t^2} \alpha'_k - \sum_{m=0, \neq k}^{N-1} \frac{\mu_{e(m,k)}^* + \Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \mu_{\delta \bar{p}_1(m)}^*}{\Delta t^2} \alpha'_m + \\
& + \frac{\alpha_k^* (\bar{\rho}_k^* + \tilde{e}_k^* \bar{\rho}_{\tilde{u}_k}^*) - \eta_{e(k,k)}^*}{\Delta t^2} - \frac{\Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \eta_{\delta \bar{p}_1(k)}^*}{\Delta t^2} \tilde{u}'_k - \sum_{m=0, \neq k}^{N-1} \frac{\eta_{e(m,k)}^* + \Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \eta_{\delta \bar{p}_1(m)}^*}{\Delta t^2} \tilde{u}'_m + \\
& + \frac{\alpha_k^* \tilde{e}_k^* \bar{\rho}_{\bar{p}_k}^* - \nu_{e_k}^* - \Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) (1 + \nu_{\delta \bar{p}_1}^*)}{\Delta t^2} \bar{p}' = \\
& = -\frac{1}{\Delta t^2} \left[ \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* - (\alpha_k^* \bar{\rho}_k^* \tilde{e}_k^*)^n - \gamma_{e_k}^* - \right. \\
& \quad \left. - \Delta t \left[ (\bar{p}^* + \gamma_{\delta \bar{p}_1}^*) (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) - \nabla_{e_k} \cdot (\alpha_k^* \tilde{\mathbf{v}}_k^* (\bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^*)) \right] \right] - \\
& \quad - \frac{1}{\Delta t} \nabla_{e_k} \cdot (\alpha_k^* \tilde{\mathbf{v}}_k^* (1 + \nu_{\Delta \bar{p}_k}^* + \tilde{e}_k^* \bar{\rho}_{\bar{p}_k}^*) \bar{p}') + \tag{4.65} \\
& + \frac{1}{\Delta t} \left[ (\bar{p}^* + \gamma_{\delta \bar{p}_1}^*) (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha'_k) - \right. \\
& \quad \left. - \nabla_{e_k} \cdot (\tilde{\mathbf{v}}_k^* \left[ (\bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^*) \alpha'_k + \alpha_k^* \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha'_m \right]) \right] - \\
& \quad - \frac{1}{\Delta t} \nabla_{e_k} \cdot (\alpha_k^* \tilde{\mathbf{v}}_k^* \left[ (\bar{\rho}_k^* + \tilde{e}_k^* \bar{\rho}_{\tilde{u}_k}^*) \tilde{u}'_k + \sum_{m=0}^{N-1} \eta_{\Delta \bar{p}(m,k)}^* \tilde{u}'_m \right]) - \\
& - \frac{1}{\Delta t} \nabla_{e_k} \cdot (\alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \tilde{\mathbf{v}}'_m + \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* (\tilde{\mathbf{v}}_k^* \cdot \tilde{\mathbf{v}}'_k) + \alpha_k^* (\bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^*) \tilde{\mathbf{v}}'_k) + \\
& \quad + \frac{1}{\Delta t^2} \left( \Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right) \cdot \tilde{\mathbf{v}}'_m \\
& \quad + \frac{1}{\Delta t^2} \left( \Delta t (\bar{p}^* + \gamma_{\delta \bar{p}_1}^*) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right) \cdot \tilde{\mathbf{v}}'_k
\end{aligned}$$

and plug eq.(4.50) to replace velocity corrections:

**Total energy,  $[k=0, N-1]$ :**

$$\begin{aligned}
 & \underbrace{\sigma_{(e,\alpha')}^{(k)} \alpha'_k + \sigma_{(e,u')}^{(k)} \tilde{u}'_k + \sigma_{(e,p')}^{(k)} \tilde{p}'_k}_{\text{in-phase coupling}} + \underbrace{\sum_{m=0, \neq k}^{N-1} \sigma_{(e,\alpha')}^{(k,m)} \alpha'_m + \sum_{m=0, \neq k}^{N-1} \sigma_{(e,u')}^{(k,m)} \tilde{u}'_m}_{\text{inter-phase coupling}} = \\
 & = -\frac{\tau_{e_k}^*}{\Delta t^2} + \underbrace{\mathcal{L}_{e_k}(\nabla \tilde{p}', \nabla^2 \tilde{p}') + \mathfrak{D}_{(e,\alpha')}^{(k)} + \mathfrak{D}_{(e,u')}^{(k)}}_{\text{Pressure waves}}
 \end{aligned} \tag{4.66}$$

where

$$\sigma_{(e,\alpha')}^{(k)} = \frac{1}{\Delta t^2} \left[ \begin{aligned} & \bar{\rho}_k^* \tilde{e}_k^* - \mu_{e(k,k)}^* - \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \mu_{\delta \bar{p}_1}^* - \\ & - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right] \cdot \left( \begin{aligned} & \mathfrak{F}_{(k,k)}^{-1} \hat{\mu}_{\mathbf{v}(k)}^* + \\ & + \sum_{m=0, \neq k}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \hat{\mu}_{\mathbf{v}(k,m)}^* \end{aligned} \right) - \\ & - \left[ \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right] \cdot \left( \begin{aligned} & \mathfrak{F}_{(m,k)}^{-1} \hat{\mu}_{\mathbf{v}(k)}^* + \\ & + \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \hat{\mu}_{\mathbf{v}(k,i)}^* \end{aligned} \right) \end{aligned} \right] \tag{4.67}$$

$$\sigma_{(e,u')}^{(k)} = \frac{1}{\Delta t^2} \left[ \begin{aligned} & \alpha_k^* \left( \bar{\rho}_k^* + \tilde{e}_k^* \bar{\rho}_k^* \right) - \eta_{e(k,k)}^* - \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \eta_{\delta \bar{p}_1}^* - \\ & - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right] \cdot \left( \begin{aligned} & \mathfrak{F}_{(k,k)}^{-1} \hat{\eta}_{\mathbf{v}(k)}^* + \\ & + \sum_{m=0, \neq k}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \hat{\eta}_{\mathbf{v}(k,m)}^* \end{aligned} \right) - \\ & - \left[ \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right] \cdot \left( \begin{aligned} & \mathfrak{F}_{(m,k)}^{-1} \hat{\eta}_{\mathbf{v}(k)}^* + \\ & + \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \hat{\eta}_{\mathbf{v}(k,i)}^* \end{aligned} \right) \end{aligned} \right] \tag{4.68}$$

$$\sigma_{(e,p')}^{(k)} = \frac{1}{\Delta t^2} \left[ \begin{aligned} & \alpha_k^* \tilde{e}_k^* \bar{\rho}_{\bar{p}_k}^* - \nu_{e_k}^* - \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \left( 1 + \nu_{\delta \bar{p}_1}^* \right) - \\ & - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right] \cdot \sum_{m=0}^{N-1} \frac{\mathfrak{F}_{(k,m)}^{-1}}{\omega_m^*} - \\ & - \left[ \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right] \cdot \sum_{i=0}^{N-1} \frac{\mathfrak{F}_{(m,i)}^{-1}}{\omega_i^*} \end{aligned} \right] \tag{4.69}$$

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

57

are *in-phase* void/energy/pressure coupling elements of the  $(2 \times N)$  **mass-energy wavenumber matrix**,  $\tilde{\mathbb{W}}$ ;

$$\begin{aligned}
 \boxed{\sum_{m=0, \neq k}^{N-1} \sigma_{(e, \alpha')}^{(k, m)} \alpha'_m} &= - \sum_{m=0, \neq k}^{N-1} \frac{\mu_{e(m, k)}^* + \Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \mu_{\delta \bar{p}_1(m)}^*}{\Delta t^2} \alpha'_m - \\
 &\quad - \frac{\Delta t (\bar{p}^* + \gamma_{\delta \bar{p}_1}^*) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^*}{\Delta t^2} \cdot \sum_{m=0, \neq k}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \left( \hat{\boldsymbol{\mu}}_{\mathbf{v}(m)}^* \alpha'_m \right) - \\
 &\quad - \frac{\Delta t (\bar{p}^* + \gamma_{\delta \bar{p}_1}^*) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^*}{\Delta t^2} \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \sum_{j=0, \neq m, \neq k}^{N-1} \left( \hat{\boldsymbol{\mu}}_{\mathbf{v}(j, m)}^* \alpha'_j \right) - \\
 &\quad - \frac{\Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m, k)}^*}{\Delta t^2} \cdot \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \left( \hat{\boldsymbol{\mu}}_{\mathbf{v}(i)}^* \alpha'_i \right) - \\
 &\quad - \frac{\Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m, k)}^*}{\Delta t^2} \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \left[ \sum_{j=0, \neq i, \neq k}^{N-1} \left( \hat{\boldsymbol{\mu}}_{\mathbf{v}(j, i)}^* \alpha'_j \right) \right]
 \end{aligned} \tag{4.70}$$

$$\begin{aligned}
 \boxed{\sum_{m=0, \neq k}^{N-1} \sigma_{(e, u')}^{(k, m)} \tilde{u}'_m} &= - \sum_{m=0, \neq k}^{N-1} \frac{\eta_{e(m, k)}^* + \Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \eta_{\delta \bar{p}_1(m)}^*}{\Delta t^2} \tilde{u}'_m - \\
 &\quad - \frac{\Delta t (\bar{p}^* + \gamma_{\delta \bar{p}_1}^*) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^*}{\Delta t^2} \cdot \sum_{m=0, \neq k}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \left( \hat{\boldsymbol{\eta}}_{\mathbf{v}(m)}^* \tilde{u}'_m \right) - \\
 &\quad - \frac{\Delta t (\bar{p}^* + \gamma_{\delta \bar{p}_1}^*) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^*}{\Delta t^2} \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k, m)}^{-1} \sum_{j=0, \neq m, \neq k}^{N-1} \left( \hat{\boldsymbol{\eta}}_{\mathbf{v}(j, m)}^* \tilde{u}'_j \right) - \\
 &\quad - \frac{\Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m, k)}^*}{\Delta t^2} \cdot \sum_{i=0, \neq k}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \left( \hat{\boldsymbol{\eta}}_{\mathbf{v}(i)}^* \tilde{u}'_i \right) - \\
 &\quad - \frac{\Delta t (\tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^*) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m, k)}^*}{\Delta t^2} \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m, i)}^{-1} \left[ \sum_{j=0, \neq i, \neq k}^{N-1} \left( \hat{\boldsymbol{\eta}}_{\mathbf{v}(j, i)}^* \tilde{u}'_j \right) \right]
 \end{aligned} \tag{4.71}$$

are *inter-phase* void/energy coupling elements of  $\hat{\mathbb{W}}$ ,

$$\begin{aligned}
\boxed{\mathbf{r}_{e_k}^*} &= \alpha_k^* \bar{\rho}_k^* \tilde{e}_k^* - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n - \gamma_{e_k}^* - \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_I}^* \right) \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) + \\
&+ \left( \Delta t \left[ \bar{p}^* + \gamma_{\delta \bar{p}_I}^* \right] \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right) \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \mathbf{r}_{\mathbf{v}_m}^* + \\
&+ \left[ \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_I(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right] \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \mathbf{r}_{\mathbf{v}_i}^* + \\
&+ \Delta t \nabla_{e_k} \cdot \left( \begin{array}{l} \alpha_k^* \tilde{\mathbf{v}}_k^* \left( \bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^* \right) - \\ - \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \mathbf{r}_{\mathbf{v}_i}^* - \\ - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \left( \tilde{\mathbf{v}}_k^* \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \mathbf{r}_{\mathbf{v}_m}^* \right) - \\ - \alpha_k^* \left( \bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^* \right) \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \mathbf{r}_{\mathbf{v}_m}^* \end{array} \right) \quad (4.72)
\end{aligned}$$

is a non-linear energy residual vector, evaluated at state  $\mathbf{V}^*$ ,

$$\begin{aligned}
\boxed{\mathcal{L}_{e_k}(\nabla \bar{p}', \nabla^2 \bar{p}')} &= \frac{1}{\Delta t^2} \left( \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_I}^* \right) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right) \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \partial \mathbf{p}'_m + \\
&+ \frac{1}{\Delta t^2} \left( \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_I(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right) \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \partial \mathbf{p}'_i - \\
&- \frac{1}{\Delta t} \nabla_{e_k} \cdot \left( \begin{array}{l} \alpha_k^* \tilde{\mathbf{v}}_k^* \left( 1 + \nu_{\Delta \bar{p}_k}^* + \tilde{e}_k^* \bar{\rho}_k^* \right) \bar{p}' + \\ \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \left( \frac{\bar{p}'}{\omega_i^*} + \partial \mathbf{p}'_i \right) + \\ + \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \left( \tilde{\mathbf{v}}_k^* \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \frac{\bar{p}'}{\omega_m^*} + \partial \mathbf{p}'_m \right) \right) + \\ + \alpha_k^* \left( \bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^* \right) \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \frac{\bar{p}'}{\omega_m^*} + \partial \mathbf{p}'_m \right) \end{array} \right) \quad (4.73)
\end{aligned}$$

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

59

is Laplacian, and

$$\begin{aligned}
\boxed{\mathcal{D}_{(e,\alpha')}^{(k)}} &= \frac{1}{\Delta t^2} \left( \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right) \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \mathfrak{d}\alpha'_m + \\
&+ \frac{1}{\Delta t^2} \left( \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right) \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \mathfrak{d}\alpha'_i + \\
&+ \frac{1}{\Delta t} \left[ \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha'_k \right) - \right. \\
&\quad \left. - \nabla_{e_k} \cdot \left( \tilde{\mathbf{v}}_k^* \left[ \left( \bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^* \right) \alpha'_k + \alpha_k^* \sum_{m=0}^{N-1} \mu_{\Delta \bar{p}(m,k)}^* \alpha'_m \right] \right) \right] - \\
&- \frac{1}{\Delta t} \nabla_{e_k} \cdot \left( \begin{aligned} &\alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \left( \hat{\mu}_{v(i)}^* \alpha'_i + \sum_{j=0, \neq i}^{N-1} \hat{\mu}_{v(j,i)}^* \alpha'_j + \mathfrak{d}\alpha'_i \right) + \\ &+ \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \left( \tilde{\mathbf{v}}_k^* \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \hat{\mu}_{v(m)}^* \alpha'_m + \sum_{j=0, \neq m}^{N-1} \hat{\mu}_{v(j,m)}^* \alpha'_j + \mathfrak{d}\alpha'_m \right) \right) + \\ &+ \alpha_k^* \left( \bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^* \right) \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \hat{\mu}_{v(m)}^* \alpha'_m + \sum_{j=0, \neq m}^{N-1} \hat{\mu}_{v(j,m)}^* \alpha'_j + \mathfrak{d}\alpha'_m \right) \end{aligned} \right) \quad (4.74)
\end{aligned}$$

$$\begin{aligned}
\boxed{\mathcal{D}_{(e,u')}^{(k)}} &= \frac{1}{\Delta t^2} \left( \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_k} \alpha_k^* - \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \right) \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \mathfrak{d}u'_m + \\
&+ \frac{1}{\Delta t^2} \left( \Delta t \left( \tilde{\mathbf{v}}_k^* \cdot \nabla_{e_k} \alpha_k^* \right) \sum_{m=0}^{N-1} \zeta_{\delta \bar{p}_1(m)}^* + \sum_{m=0}^{N-1} \zeta_{e(m,k)}^* \right) \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \mathfrak{d}u'_i - \\
&\left( \begin{aligned} &\alpha_k^* \tilde{\mathbf{v}}_k^* \left[ \left( \bar{\rho}_k^* + \tilde{e}_k^* \bar{\rho}_{\tilde{u}_k}^* \right) \tilde{u}'_k + \sum_{m=0}^{N-1} \eta_{\Delta \bar{p}(m,k)}^* \tilde{u}'_m \right] + \\ &+ \alpha_k^* \tilde{\mathbf{v}}_k^* \sum_{m=0}^{N-1} \zeta_{\Delta \bar{p}(m,k)}^* \cdot \sum_{i=0}^{N-1} \mathfrak{F}_{(m,i)}^{-1} \left( \hat{\eta}_{v(i)}^* \tilde{u}'_i + \sum_{j=0, \neq i}^{N-1} \hat{\eta}_{v(j,i)}^* \tilde{u}'_j + \mathfrak{d}u'_i \right) + \\ &+ \alpha_k^* \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \left( \tilde{\mathbf{v}}_k^* \cdot \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \hat{\eta}_{v(m)}^* \tilde{u}'_m + \sum_{j=0, \neq m}^{N-1} \hat{\eta}_{v(j,m)}^* \tilde{u}'_j + \mathfrak{d}u'_m \right) \right) + \\ &+ \alpha_k^* \left( \bar{\rho}_k^* \tilde{e}_k^* + \bar{p}^* + \gamma_{\Delta \bar{p}_k}^* \right) \sum_{m=0}^{N-1} \mathfrak{F}_{(k,m)}^{-1} \left( \hat{\eta}_{v(m)}^* \tilde{u}'_m + \sum_{j=0, \neq m}^{N-1} \hat{\eta}_{v(j,m)}^* \tilde{u}'_j + \mathfrak{d}u'_m \right) \end{aligned} \right) \quad (4.75)
\end{aligned}$$

are terms associated with spatial variations of void fraction and specific internal energy correction, correspondingly; similar to those of momentum and mass equations (4.42) and (4.56). Again, the first obvious choice would be simply ignore them (OS mode), but we will introduce other possible treatments of these terms in Section 4.3.7.

Similar to eq.(4.64), we introduce the following *conservation of mixture energy* equation:

**Mixture energy:**

$$\begin{aligned}
 & \sum_{n=0}^{N-1} \left( \sigma_{(e,\alpha')}^{(n)} \alpha'_n + \sum_{m=0, \neq n}^{N-1} \sigma_{(e,\alpha')}^{(n,m)} \alpha'_m \right) + \\
 & + \sum_{n=0}^{N-1} \left( \sigma_{(e,u')}^{(n)} \tilde{u}'_n + \sum_{m=0, \neq n}^{N-1} \sigma_{(e,u')}^{(n,m)} \tilde{u}'_m \right) + \sum_{n=0}^{N-1} \sigma_{(e,p')}^{(n)} \bar{p}' = \\
 & = \sum_{n=0}^{N-1} \left( \mathcal{L}_{e_n} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(e,\alpha')}^{(n)} + \mathfrak{D}_{(e,u')}^{(n)} - \frac{\mathbf{r}_{\alpha_n}^*}{\Delta t^2} \right)
 \end{aligned} \tag{4.76}$$

### 4.3.6 Pressure correction P'HE

Eqs.(4.56) and (4.66) compose a system of  $(2 \times N)$  equations for  $(2 \times N + 1)$  unknowns:  $(\bar{p}', \alpha'_k, \tilde{u}'_k)$ ,  $k=0, \dots, N-1$ . We eliminate one unknown – void fraction for the selected phase  $e$ , by using *compatibility equation*:

$$\alpha'_e = 1 - \alpha_e^* - \sum_{m=0, \neq e}^{N-1} (\alpha_m^* + \alpha'_m) \tag{4.77}$$

For the sake of robustness when a phase appears/disappears, we choose to eliminate void fraction for a fluid with the smallest  $\alpha_e$ . Note, that nothing prohibits to choose different  $e$  for different computational cells, provided that the choice of  $e$  is frozen during non-linear iterations (to avoid “clicking” if Newton procedure is used). Also, instead of phasic mass and energy equations for phase  $e$ , we will use mixture mass and energy equations (4.64) and (4.76). This should avoid forming degenerate P'HE when  $\alpha_e \rightarrow 0$ .

Now, a closed system of  $(2 \times N)$  P'HE equations for  $(2 \times N)$  unknowns

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

61

$(\bar{p}', \alpha'_k, \tilde{u}'_m)$ ,  $[k, m=0, \dots, N-1, k \neq \epsilon]$  can be written as:

**P' HE of phasic mass conservation**,  $[k=0, 1, \dots, N-1, k \neq \epsilon]$ :

$$\begin{aligned}
 & \left( \sigma_{(\rho, \alpha')}^{(k)} - \sigma_{(\rho, \alpha')}^{(k, \epsilon)} \right) \alpha'_k + \sigma_{(\rho, u')}^{(k)} \tilde{u}'_k + \sigma_{(\rho, p')}^{(k)} \bar{p}' + \\
 & \quad + \sum_{m=0, \neq k, \neq \epsilon}^{N-1} \left( \sigma_{(\rho, \alpha')}^{(k, m)} - \sigma_{(\rho, \alpha')}^{(k, \epsilon)} \right) \alpha'_m + \sum_{m=0, \neq k}^{N-1} \sigma_{(\rho, u')}^{(k, m)} \tilde{u}'_m = \\
 & = \mathcal{L}_{\rho_k} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(\rho, \alpha')}^{(k)} + \mathfrak{D}_{(\rho, u')}^{(k)} - \sigma_{(\rho, \alpha')}^{(k, \epsilon)} \left( 1 - \sum_{m=0}^{N-1} \alpha_m^* \right) - \frac{\tau_{\alpha_k}^*}{\Delta t^2}
 \end{aligned} \tag{4.78}$$

**P' HE of phase- $\epsilon$  mass conservation:**

$$\begin{aligned}
 & \sigma_{(\rho, u')}^{(\epsilon)} \tilde{u}'_\epsilon + \sigma_{(\rho, p')}^{(\epsilon)} \bar{p}' + \sum_{m=0, \neq \epsilon}^{N-1} \left( \sigma_{(\rho, \alpha')}^{(\epsilon, m)} - \sigma_{(\rho, \alpha')}^{(\epsilon)} \right) \alpha'_m + \sum_{m=0, \neq \epsilon}^{N-1} \sigma_{(\rho, u')}^{(\epsilon, m)} \tilde{u}'_m = \\
 & = \mathcal{L}_{\rho_\epsilon} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(\rho, \alpha')}^{(\epsilon)} + \mathfrak{D}_{(\rho, u')}^{(\epsilon)} - \sigma_{(\rho, \alpha')}^{(\epsilon)} \left( 1 - \sum_{m=0}^{N-1} \alpha_m^* \right) - \frac{\tau_{\alpha_\epsilon}^*}{\Delta t^2}
 \end{aligned} \tag{4.79}$$

**P' HE of phasic total energy conservation**,  $[k=0, 1, \dots, N-1, k \neq \epsilon]$ :

$$\begin{aligned}
 & \left( \sigma_{(e, \alpha')}^{(k)} - \sigma_{(e, \alpha')}^{(k, \epsilon)} \right) \alpha'_k + \sigma_{(e, u')}^{(k)} \tilde{u}'_k + \sigma_{(e, p')}^{(k)} \bar{p}' + \\
 & \quad + \sum_{m=0, \neq k, \neq \epsilon}^{N-1} \left( \sigma_{(e, \alpha')}^{(k, m)} - \sigma_{(e, \alpha')}^{(k, \epsilon)} \right) \alpha'_m + \sum_{m=0, \neq k}^{N-1} \sigma_{(e, u')}^{(k, m)} \tilde{u}'_m = \\
 & = \mathcal{L}_{e_k} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(e, \alpha')}^{(k)} + \mathfrak{D}_{(e, u')}^{(k)} - \sigma_{(e, \alpha')}^{(k, \epsilon)} \left( 1 - \sum_{m=0}^{N-1} \alpha_m^* \right) - \frac{\tau_{e_k}^*}{\Delta t^2}
 \end{aligned} \tag{4.80}$$

**P' HE of phase- $\epsilon$  total energy conservation:**

$$\begin{aligned}
 & \sigma_{(e, u')}^{(\epsilon)} \tilde{u}'_\epsilon + \sigma_{(e, p')}^{(\epsilon)} \bar{p}' + \sum_{m=0, \neq \epsilon}^{N-1} \left( \sigma_{(e, \alpha')}^{(\epsilon, m)} - \sigma_{(e, \alpha')}^{(\epsilon)} \right) \alpha'_m + \sum_{m=0, \neq \epsilon}^{N-1} \sigma_{(e, u')}^{(\epsilon, m)} \tilde{u}'_m = \\
 & = \mathcal{L}_{e_\epsilon} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(e, \alpha')}^{(\epsilon)} + \mathfrak{D}_{(e, u')}^{(\epsilon)} - \sigma_{(e, \alpha')}^{(\epsilon)} \left( 1 - \sum_{m=0}^{N-1} \alpha_m^* \right) - \frac{\tau_{e_\epsilon}^*}{\Delta t^2}
 \end{aligned} \tag{4.81}$$

As an alternative to replace eqs.(4.79) and (4.81), one can solve for the following

mixture mass and energy equations:

**P' HE of mixture mass:**

$$\begin{aligned}
& \sum_{n=0, \neq \epsilon}^{N-1} \left[ \left( \sigma_{(\rho, \alpha')}^{(n)} + \sigma_{(\rho, \alpha')}^{(\epsilon, n)} \right) \alpha'_n + \sum_{m=0, \neq n, \neq \epsilon}^{N-1} \sigma_{(\rho, \alpha')}^{(n, m)} \alpha'_m \right] - \\
& \quad - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(\rho, \alpha')}^{(n, \epsilon)} + \sigma_{(\rho, \alpha')}^{(\epsilon)} \right) \sum_{m=0, \neq \epsilon}^{N-1} \alpha'_m + \\
& \quad + \sum_{n=0}^{N-1} \left( \sigma_{(\rho, u')}^{(n)} \tilde{u}'_n + \sum_{m=0, \neq n}^{N-1} \sigma_{(\rho, u')}^{(n, m)} \tilde{u}'_m \right) + \sum_{n=0}^{N-1} \sigma_{(\rho, p')}^{(n)} \bar{p}' = \\
& \quad = \sum_{n=0}^{N-1} \left( \mathcal{L}_{\rho_n} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(\rho, \alpha')}^{(n)} + \mathfrak{D}_{(\rho, u')}^{(n)} - \frac{\mathbf{r}_{\alpha_n}^*}{\Delta t^2} \right) - \\
& \quad - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(\rho, \alpha')}^{(n, \epsilon)} + \sigma_{(\rho, \alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{m=0}^{N-1} \alpha_m^* \right)
\end{aligned} \tag{4.82}$$

and

**P' HE of mixture energy:**

$$\begin{aligned}
& \sum_{n=0, \neq \epsilon}^{N-1} \left[ \left( \sigma_{(e, \alpha')}^{(n)} + \sigma_{(e, \alpha')}^{(\epsilon, n)} \right) \alpha'_n + \sum_{m=0, \neq n, \neq \epsilon}^{N-1} \sigma_{(e, \alpha')}^{(n, m)} \alpha'_m \right] - \\
& \quad - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(e, \alpha')}^{(n, \epsilon)} + \sigma_{(e, \alpha')}^{(\epsilon)} \right) \sum_{m=0, \neq \epsilon}^{N-1} \alpha'_m + \\
& \quad + \sum_{n=0}^{N-1} \left( \sigma_{(e, u')}^{(n)} \tilde{u}'_n + \sum_{m=0, \neq n}^{N-1} \sigma_{(e, u')}^{(n, m)} \tilde{u}'_m \right) + \sum_{n=0}^{N-1} \sigma_{(e, p')}^{(n)} \bar{p}' = \\
& \quad = \sum_{n=0}^{N-1} \left( \mathcal{L}_{e_n} (\nabla \bar{p}', \nabla^2 \bar{p}') + \mathfrak{D}_{(e, \alpha')}^{(n)} + \mathfrak{D}_{(e, u')}^{(n)} - \frac{\mathbf{r}_{e_n}^*}{\Delta t^2} \right) - \\
& \quad - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(e, \alpha')}^{(n, \epsilon)} + \sigma_{(e, \alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{m=0}^{N-1} \alpha_m^* \right)
\end{aligned} \tag{4.83}$$

Now we have two alternative (and mathematically equivalent) systems of  $(2 \times N)$  locally coupled P' HE equations to solve:

I. **Phasic system:** eqs.(4.78)-(4.81).

I. **Mixture system:** eqs.(4.78), (4.80), (4.82) and (4.83).

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 63

The mixture system is slightly better for solution algorithms, when  $\alpha_\epsilon \rightarrow 0$ . The phasic system is slightly easier to analyze from the point of view wavenumbers.

Both mixture and phasic systems can be written in the following matrix form:

$$\mathbb{W}(\mathbf{V}^*) \begin{bmatrix} \dots \\ \alpha'_k \\ \dots \\ \tilde{u}'_m \\ \dots \\ \bar{p}' \end{bmatrix} = \mathbf{f} \left( \nabla \bar{p}', \nabla^2 \bar{p}', \mathfrak{D}_{(\rho, \alpha')}^{(n)}, \mathfrak{D}_{(\rho, u')}^{(n)}, \mathfrak{D}_{(e, \alpha')}^{(n)}, \mathfrak{D}_{(e, u')}^{(n)} \right) \quad (4.84)$$

$k = 0, \dots, N-1; k \neq \epsilon$   
 $(m, n) = 0, \dots, N-1$

where  $\mathbb{W}$  is a wavenumber matrix. This system can be locally de-coupled to produce the following set of equations for phasic void fractions, internal energies and pressure:

$$\begin{bmatrix} \dots \\ \alpha'_k \\ \dots \\ \tilde{u}'_m \\ \dots \\ \bar{p}' \end{bmatrix} = \mathbb{W}^{-1} \mathbf{f} \left( \nabla \bar{p}', \nabla^2 \bar{p}', \mathfrak{D}_{(\rho, \alpha')}^{(n)}, \mathfrak{D}_{(\rho, u')}^{(n)}, \mathfrak{D}_{(e, \alpha')}^{(n)}, \mathfrak{D}_{(e, u')}^{(n)} \right) \quad (4.85)$$

$k = 0, \dots, N-1; k \neq \epsilon$   
 $(m, n) = 0, \dots, N-1$

The last equation in this system is the **pressure correction P'HE**:

$$\kappa^2 \bar{p}' - \left\{ \kappa^2 \mathbb{W}^{-1} \mathbf{f} \left( \nabla \bar{p}', \nabla^2 \bar{p}', \mathfrak{D}_{(\rho, \alpha')}^{(n)}, \mathfrak{D}_{(\rho, u')}^{(n)}, \mathfrak{D}_{(e, \alpha')}^{(n)}, \mathfrak{D}_{(e, u')}^{(n)} \right) \right\}_{2N} = 0 \quad (4.86)$$

where by  $\{\mathbf{w}\}_k$  we denote the  $k^{\text{th}}$  component of the vector  $\mathbf{w}$ , and  $\kappa$  is the *wavenumber*, typically consisting of a number of harmonics. It is completely decoupled from the rest  $(N-1)$  phasic void fraction and  $N$  internal energy correction P'HE equations, provided that  $\mathfrak{D}_{(\rho, \alpha')}^{(n)}$ ,  $\mathfrak{D}_{(\rho, u')}^{(n)}$ ,  $\mathfrak{D}_{(e, \alpha')}^{(n)}$ , and  $\mathfrak{D}_{(e, u')}^{(n)}$  are available. The options for these will be explained in Section 4.3.7.

Equation (4.86) *embeds all phasic mass, energy conservation, as well as phase compatibility condition*. Once it is solved for a pressure correction  $\bar{p}'$ , we

can compute the r.h.s. vector  $\mathbf{f}(\nabla \bar{\mathbf{p}}', \nabla^2 \bar{\mathbf{p}}')$  and the rest of unknowns:

$$\begin{aligned} \alpha'_k &= \left\{ \mathbb{W}^{-1} \mathbf{f} \left( \nabla \bar{\mathbf{p}}', \nabla^2 \bar{\mathbf{p}}', \mathfrak{D}_{(\rho, \alpha')}^{(n)}, \mathfrak{D}_{(\rho, u')}^{(n)}, \mathfrak{D}_{(e, \alpha')}^{(n)}, \mathfrak{D}_{(e, u')}^{(n)} \right) \right\}_{0, \dots, N-2} \\ \tilde{\mathbf{u}}'_m &= \left\{ \mathbb{W}^{-1} \mathbf{f} \left( \nabla \bar{\mathbf{p}}', \nabla^2 \bar{\mathbf{p}}', \mathfrak{D}_{(\rho, \alpha')}^{(n)}, \mathfrak{D}_{(\rho, u')}^{(n)}, \mathfrak{D}_{(e, \alpha')}^{(n)}, \mathfrak{D}_{(e, u')}^{(n)} \right) \right\}_{N-1, \dots, 2N-1} \end{aligned} \quad (4.87)$$

$$\begin{aligned} k &= 0, \dots, N-1; k \neq \epsilon \\ (m, n) &= 0, \dots, N-1 \end{aligned}$$

These also do satisfy mass/energy conservation and compatibility condition. Next, we compute the void fraction for eliminated phase  $\alpha'_\epsilon$  using equation (4.77). Finally, we compute  $\mathbf{p}'_{v_k}$ ,  $\mathbf{a}'_{v_k}$  and  $\mathbf{u}'_{v_k}$  with eqs.(4.44)-(4.46), and then phasic velocities  $\tilde{\mathbf{v}}'_k$  by eq.(4.50). This will complete the solution algorithm.

### 4.3.7 Mass and energy stabilizers

Here, we discuss different options for computation of  $\mathfrak{D}_{(\rho, \alpha')}^{(k)}$ ,  $\mathfrak{D}_{(\rho, u')}^{(k)}$ ,  $\mathfrak{D}_{(e, \alpha')}^{(k)}$ , and  $\mathfrak{D}_{(e, u')}^{(k)}$ .

1. **OS.** Simply ignore these terms,

$$\begin{aligned} \mathfrak{D}_{(\rho, \alpha')}^{(k)} &= 0 \\ \mathfrak{D}_{(\rho, u')}^{(k)} &= 0 \\ \mathfrak{D}_{(e, \alpha')}^{(k)} &= 0 \\ \mathfrak{D}_{(e, u')}^{(k)} &= 0 \end{aligned}$$

This is what would be the closest to the original semi-implicit (operator-splitting, OS) algorithm.

2. **SIMPLE-like.** For this, we represent these terms using the following discrete forms:

$$\begin{aligned} \mathfrak{D}_{(\rho, \alpha')}^{(k)} &\rightarrow \mathfrak{A}_{(\rho, \alpha)_{c,c}}^{(k)} \alpha'_k + \sum_n^{\mathcal{N}} \mathfrak{A}_{(\rho, \alpha)_{c,n}}^{(k)} \alpha'_{k(n)} \xrightarrow{\text{ignore}} \\ \mathfrak{D}_{(e, \alpha')}^{(k)} &\rightarrow \mathfrak{A}_{(e, \alpha)_{c,c}}^{(k)} \alpha'_k + \sum_n^{\mathcal{N}} \mathfrak{A}_{(e, \alpha)_{c,n}}^{(k)} \alpha'_{k(n)} \xrightarrow{\text{ignore}} \end{aligned} \quad (4.88)$$

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 65

and

$$\begin{aligned}
 \mathfrak{D}_{(\rho,u')}^{(k)} &\rightarrow \mathfrak{A}_{(\rho,u)c,c}^{(k)} \tilde{u}'_k + \sum_n^{\mathcal{N}} \mathfrak{A}_{(\rho,u)c,n}^{(k)} \tilde{u}'_{k(n)} \xrightarrow{\text{ignore}} \\
 \mathfrak{D}_{(e,u')}^{(k)} &\rightarrow \mathfrak{A}_{(e,u)c,c}^{(k)} \tilde{u}'_k + \sum_n^{\mathcal{N}} \mathfrak{A}_{(e,u)c,n}^{(k)} \tilde{u}'_{k(n)} \xrightarrow{\text{ignore}}
 \end{aligned} \tag{4.89}$$

where  $\mathfrak{A}_{(\times)c,c}^{(k)}$  are space discretization coefficients,  $c$  denotes cell id,  $n$  – neighboring cells, while  $\mathcal{N}$  is the number of neighboring cells involved. Ignoring contributions from neighbor cells should not affect the final result, because this term is zero upon convergence of an iterative algorithm (either segregated or Newton-based). Coefficient  $\mathfrak{A}_{(\times)c,c}^{(k)}$  should be absorbed into the diagonal of the wavenumber matrix  $\mathbb{W}$ .

3. **Diagonal stabilizer.** In this option, one would solve for “predictor” values of void fraction  $\hat{\alpha}'_k$  and internal energy  $\hat{u}'_m$  corrections, using the following “explicit” version of eq.(4.85), in which  $\bar{p}'$ ,  $\mathfrak{D}_{(\rho,\alpha')}^{(k)}$ ,  $\mathfrak{D}_{(\rho,u')}^{(k)}$ ,  $\mathfrak{D}_{(e,\alpha')}^{(k)}$ , and  $\mathfrak{D}_{(e,u')}^{(k)}$  are ignored (“operator-splitted”). First, for void fractions:

$$\hat{\alpha}'_k = \left\{ \mathbb{W}^{-1} \left[ \begin{array}{c} -\sigma_{(\rho,\alpha')}^{(i,\epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{\mathbf{r}_{\alpha_i}^*}{\Delta t^2} \\ \dots \\ -\sigma_{(e,\alpha')}^{(j,\epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{\mathbf{r}_{e_j}^*}{\Delta t^2} \\ \dots \\ \sum_{n=0}^{N-1} \left( -\frac{\mathbf{r}_{\alpha_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(\rho,\alpha')}^{(n,\epsilon)} + \sigma_{(\rho,\alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \\ \sum_{n=0}^{N-1} \left( -\frac{\mathbf{r}_{e_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(e,\alpha')}^{(n,\epsilon)} + \sigma_{(e,\alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \end{array} \right] \right\}_{0, \dots, N-2} \tag{4.90}$$

$$\begin{aligned}
 (i, k) &= 0, \dots, N-1; i \neq \epsilon \\
 j &= 0, \dots, N-1
 \end{aligned}$$

and

$$\hat{\alpha}'_\epsilon = 1 - \alpha_\epsilon^* - \sum_{m=0, \neq \epsilon}^{N-1} (\alpha_m^* + \hat{\alpha}'_m) \tag{4.91}$$

This allows to compute  $\mathfrak{D}_{(\rho, \alpha')}^{(k)}(\hat{\alpha}'_k)$  and  $\mathfrak{D}_{(e, \alpha')}^{(k)}(\hat{\alpha}'_k)$  using eqs.(4.62) and (4.74). Next, for phasic specific internal energies:

$$\hat{u}'_k = \left\{ \mathbb{W}^{-1} \left[ \begin{array}{c} \mathfrak{D}_{(\rho, \alpha')}^{(i)} - \sigma_{(\rho, \alpha')}^{(i, \epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{\tau_{\alpha_i}^*}{\Delta t^2} \\ \dots \\ \mathfrak{D}_{(e, \alpha')}^{(j)} - \sigma_{(e, \alpha')}^{(j, \epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{\tau_{e_j}^*}{\Delta t^2} \\ \dots \\ \sum_{n=0}^{N-1} \left( \mathfrak{D}_{(\rho, \alpha')}^{(n)} - \frac{\tau_{\alpha_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, n \neq \epsilon}^{N-1} \sigma_{(\rho, \alpha')}^{(n, \epsilon)} + \sigma_{(\rho, \alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \\ \sum_{n=0}^{N-1} \left( \mathfrak{D}_{(e, \alpha')}^{(n)} - \frac{\tau_{e_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, n \neq \epsilon}^{N-1} \sigma_{(e, \alpha')}^{(n, \epsilon)} + \sigma_{(e, \alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \end{array} \right\}_{0, \dots, N-2} \quad (4.92)$$

$$\begin{array}{l} i = 0, \dots, N-1; i \neq \epsilon \\ (j, k) = 0, \dots, N-1 \end{array}$$

This allows to compute  $\mathfrak{D}_{(\rho, u')}^{(k)}(\hat{u}'_k)$  and  $\mathfrak{D}_{(e, u')}^{(k)}(\hat{u}'_k)$  using eqs.(4.63) and (4.75), and, finally, to incorporate all  $\mathfrak{D}_{(\rho, \alpha')}^{(k)}(\hat{\alpha}'_k)$ ,  $\mathfrak{D}_{(e, \alpha')}^{(k)}(\hat{\alpha}'_k)$ ,  $\mathfrak{D}_{(\rho, u')}^{(k)}(\hat{u}'_k)$  and  $\mathfrak{D}_{(e, u')}^{(k)}(\hat{u}'_k)$  into the pressure correction P' HE eq.(4.86).

This option would definitely improve upon the first (OS) option, without significant CPU overhead (no implicit solves involved). However, it will not “break” material Courant (heat conduction Fourier) limits, as the fractional steps stabilizer expected to do. This will be achieved by the following “implicit” stabilizer option.

4. **Implicit stabilizer.** In this option, one would also solve for “predictor” values of void fraction  $\hat{\alpha}'_k$  and internal energy  $\hat{u}'_m$  corrections, but in this case using “implicit” version of eq.(4.85). Here, we also ignore (“operator-split”)  $\bar{p}'$ , and  $\mathfrak{D}_{(e, \alpha')}^{(k)}$   $\mathfrak{D}_{(e, u')}^{(k)}$  (the last two for mass stabilizers only). Thus,

**4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS**
**67**

for void fractions:

$$\hat{\alpha}'_k - \underbrace{\left\{ \mathbb{W}^{-1} \left[ \begin{array}{c} \delta_{(i,k)} \mathfrak{D}_{(\rho,\alpha')}^{(i)} \\ \dots \\ \delta_{(j,k)} \mathfrak{D}_{(e,\alpha')}^{(j)} \\ \dots \\ \sum_{n=0}^{N-1} \left( \delta_{(n,k)} \mathfrak{D}_{(\rho,\alpha')}^{(n)} \right) \\ \sum_{n=0}^{N-1} \left( \delta_{(n,k)} \mathfrak{D}_{(e,\alpha')}^{(n)} \right) \end{array} \right] \right\}}_{\text{Implicit operator on } \hat{\alpha}'_k} = \left\{ \mathbb{W}^{-1} \left[ \begin{array}{c} -\sigma_{(\rho,\alpha')}^{(i,\epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{\mathbf{r}_{\alpha_i}^*}{\Delta t^2} \\ \dots \\ -\sigma_{(e,\alpha')}^{(j,\epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{\mathbf{r}_{e_j}^*}{\Delta t^2} \\ \dots \\ \sum_{n=0}^{N-1} \left( -\frac{\mathbf{r}_{\alpha_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(\rho,\alpha')}^{(n,\epsilon)} + \sigma_{(\rho,\alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \\ \sum_{n=0}^{N-1} \left( -\frac{\mathbf{r}_{e_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, \neq \epsilon}^{N-1} \sigma_{(e,\alpha')}^{(n,\epsilon)} + \sigma_{(e,\alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \end{array} \right] \right\}_{0, \dots, N-2} \quad (4.93)$$

$$\begin{array}{l}
 (i, k) = 0, \dots, N-1; i \neq \epsilon \\
 j = 0, \dots, N-1
 \end{array}$$

where

$$\delta_{(i,j)} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

These  $(N-1)$  decoupled implicit equations should be solved for each  $\hat{\alpha}'_k$  in a sequence, by an appropriate iterative method. Next, compute the remaining (eliminated) void fraction  $\hat{\alpha}'_\epsilon$  with compatibility equation (4.91). After this step, we can compute  $\mathfrak{D}_{(\rho,\alpha')}^{(k)}(\hat{\alpha}'_k)$  and  $\mathfrak{D}_{(e,\alpha')}^{(k)}(\hat{\alpha}'_k)$  for  $k=0, \dots, N-1$ , using eqs.(4.62) and (4.74). These can be used in the following implicit phasic

specific energy correction equations:

$$\begin{aligned}
 \hat{u}'_k - \underbrace{\left\{ \mathbb{W}^{-1} \begin{bmatrix} \delta_{(i,k)} \mathfrak{D}_{(\rho,u')}^{(i)} \\ \dots \\ \delta_{(j,k)} \mathfrak{D}_{(e,u')}^{(j)} \\ \dots \\ \sum_{n=0}^{N-1} \left( \delta_{(n,k)} \mathfrak{D}_{(\rho,u')}^{(n)} \right) \\ \sum_{n=0}^{N-1} \left( \delta_{(n,k)} \mathfrak{D}_{(e,u')}^{(n)} \right) \end{bmatrix} \right\}}_{\text{Implicit operator on } \hat{u}'_k} &= \\
 = \left\{ \mathbb{W}^{-1} \begin{bmatrix} \mathfrak{D}_{(\rho,\alpha')}^{(i)} - \sigma_{(\rho,\alpha')}^{(i,\epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{r_{\alpha_i}^*}{\Delta t^2} \\ \dots \\ \mathfrak{D}_{(e,\alpha')}^{(j)} - \sigma_{(e,\alpha')}^{(j,\epsilon)} \left( 1 - \sum_{n=0}^{N-1} \alpha_n^* \right) - \frac{r_{e_j}^*}{\Delta t^2} \\ \dots \\ \sum_{n=0}^{N-1} \left( \mathfrak{D}_{(\rho,\alpha')}^{(n)} - \frac{r_{\alpha_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, n \neq \epsilon}^{N-1} \sigma_{(\rho,\alpha')}^{(n,\epsilon)} + \sigma_{(\rho,\alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \\ \sum_{n=0}^{N-1} \left( \mathfrak{D}_{(e,\alpha')}^{(n)} - \frac{r_{e_n}^*}{\Delta t^2} \right) - \left( \sum_{n=0, n \neq \epsilon}^{N-1} \sigma_{(e,\alpha')}^{(n,\epsilon)} + \sigma_{(e,\alpha')}^{(\epsilon)} \right) \left( 1 - \sum_{l=0}^{N-1} \alpha_l^* \right) \end{bmatrix} \right\}_{0, \dots, N-2} & \quad (4.94)
 \end{aligned}$$

$$\begin{aligned}
 i &= 0, \dots, N-1; i \neq \epsilon \\
 (j, k) &= 0, \dots, N-1
 \end{aligned}$$

Now, we can compute  $\mathfrak{D}_{(\rho,u')}^{(k)} \left( \hat{u}'_k \right)$  and  $\mathfrak{D}_{(e,u')}^{(k)} \left( \hat{u}'_k \right)$  using eqs.(4.63) and (4.75), and, finally, incorporate all  $\mathfrak{D}_{(\rho,\alpha')}^{(k)} \left( \hat{\alpha}'_k \right)$ ,  $\mathfrak{D}_{(e,\alpha')}^{(k)} \left( \hat{\alpha}'_k \right)$ ,  $\mathfrak{D}_{(\rho,u')}^{(k)} \left( \hat{u}'_k \right)$  and  $\mathfrak{D}_{(e,u')}^{(k)} \left( \hat{u}'_k \right)$  into the pressure correction P'HE eq.(4.86).

### 4.3.8 Outline of the fractional step version

In this section, we will summarize the fractional step version of the algorithm.

1. Start with given solution vector state  $\mathbf{V}^*$ . Under operator-splitting, this vector is set to  $\mathbf{V}^n$ . In the non-linear iteration algorithms (either Picard- or Newton-based), the first guess is set as  $\mathbf{V}^* = \mathbf{V}^n$ .

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 69

2. Compute non-linear momentum residual vector  $\mathbf{r}_{v_k}^*$  defined by eq.(4.47).
3. **Velocity stabilizer.** Solve  $N$  implicit equations (4.54) for  $\hat{\mathbf{v}}'_k$ . This step should break stability limits associated with material Courant and (if modeled) turbulent viscosity Fourier number. Predictor velocity  $\hat{\mathbf{v}}'_k$  can now be used to compute  $\mathcal{D}'_{v_k} (\nabla \cdot \hat{\mathbf{v}}'_k)$ , eq.(4.43), which is absorbed into  $\mathbf{r}_{v_m}^*$  as in eq.(4.53).
4. **Mass stabilizer.** Solve  $(N - 1)$  implicit equations (4.93) for  $\hat{\alpha}'_k$ , and compute the remaining (eliminated) void fraction  $\hat{\alpha}'_e$  with compatibility equation (4.91). After this step, we can compute  $\mathcal{D}_{(\rho, \alpha')}^{(k)} (\hat{\alpha}'_k)$  and  $\mathcal{D}_{(e, \alpha')}^{(k)} (\hat{\alpha}'_k)$  for  $k=0, \dots, N-1$ , using eqs.(4.62) and (4.74). This step should eliminate stability limit associated with material Courant.
5. **Energy stabilizer.** Solve  $N$  implicit equations (4.94) for  $\hat{u}'_k$ . Now, we can compute  $\mathcal{D}_{(\rho, u')}^{(k)} (\hat{u}'_k)$  and  $\mathcal{D}_{(e, u')}^{(k)} (\hat{u}'_k)$  using eqs.(4.63) and (4.75), and, finally, incorporate all  $\mathcal{D}_{(\rho, \alpha')}^{(k)} (\hat{\alpha}'_k)$ ,  $\mathcal{D}_{(e, \alpha')}^{(k)} (\hat{\alpha}'_k)$ ,  $\mathcal{D}_{(\rho, u')}^{(k)} (\hat{u}'_k)$  and  $\mathcal{D}_{(e, u')}^{(k)} (\hat{u}'_k)$  into the pressure correction P'HE eq.(4.86). This step should eliminate stability limits associated with material Courant and (if modeled) turbulent heat conduction Fourier number.
6. **Pressure correction P'HE.** Solve implicit equation (4.86) for  $\bar{p}'$ . This pressure *enforces all phasic mass, energy conservation, as well as phase compatibility condition.*
7. Compute the r.h.s. vector  $\mathbf{f} (\nabla \bar{p}', \nabla^2 \bar{p}')$  and the phasic void fraction and internal energy corrections, using eq.(4.87). These also DO satisfy mass/energy conservation and compatibility condition.
8. Next, compute the void fraction for the eliminated phase  $\alpha'_e$  using equation (4.77).
9. Compute  $\mathbf{p}'_{v_k}$ ,  $\mathbf{\alpha}'_{v_k}$  and  $\mathbf{u}'_{v_k}$  with eqs.(4.44)-(4.46), and then phasic velocities  $\tilde{\mathbf{v}}'_k$  by eq.(4.50).
10. *Newton, preconditioning:* done for preconditioning of the Newton-based algorithm. Return  $\phi'$  as  $\delta \bar{\mathcal{X}}_V^a$ , see Section 6.5.

11. Update all solution variables  $\mathbf{V}^{**}$  with eq.(4.26).
12. *Operator-splitting*: done, i.e.  $\mathbf{V}^{n+1} = \mathbf{V}^{**}$ .
13. *Picard-based algorithm*: Check convergence. If not converged, reset  $\mathbf{V}^* = \mathbf{V}^{**}$  and return to step 2. If converged,  $\mathbf{V}^{n+1} = \mathbf{V}^{**}$  and done.

### 4.3.9 P'HE for single-fluid formulation

The vector of unknowns for 1-fluid P'HE system is  $\begin{bmatrix} \tilde{u}' \\ \tilde{p}' \end{bmatrix}$ .

The friction coefficient matrix is simply

$$\mathfrak{F} = \bar{\rho}^* - \zeta_v^*$$

The wavenumber matrix  $\mathbb{W}$  becomes

$$\mathbb{W} = \begin{bmatrix} \boxed{\sigma_{(\rho, u')}} & \boxed{\sigma_{(\rho, p')}} \\ \boxed{\sigma_{(e, u')}} & \boxed{\sigma_{(e, p')}} \end{bmatrix} \quad (4.95)$$

where

$$\begin{aligned} \sigma_{(\rho, u')}^{(0)} &= \frac{1}{\Delta t^2} \left( \bar{\rho}_u^* - \eta_\rho^* - \zeta_\rho^* \cdot \frac{\eta_v^* - \bar{\rho}_u^* \tilde{v}^*}{\bar{\rho}^* - \zeta_v^*} \right) \\ \sigma_{(\rho, p')}^{(0)} &= \frac{1}{\Delta t^2} \left( \bar{\rho}_p^* - \nu_\rho^* - \zeta_\rho^* \cdot \frac{\nu_v^* - \bar{\rho}_p^* \tilde{v}^*}{\bar{\rho}^* - \zeta_v^*} \right) \\ \sigma_{(e, u')} &= \frac{1}{\Delta t^2} \left( \bar{\rho}^* + \tilde{e}^* \bar{\rho}_u^* - \eta_e^* + (\bar{\rho}^* \tilde{v}^* - \zeta_e^*) \cdot \frac{\eta_v^* - \bar{\rho}_u^* \tilde{v}^*}{\bar{\rho}^* - \zeta_v^*} \right) \\ \sigma_{(e, p')} &= \frac{1}{\Delta t^2} \left( \tilde{e}^* \bar{\rho}_p^* - \nu_e^* + (\bar{\rho}^* \tilde{v}^* - \zeta_e^*) \cdot \frac{\nu_v^* - \bar{\rho}_p^* \tilde{v}^*}{\bar{\rho}^* - \zeta_v^*} \right) \end{aligned} \quad (4.96)$$

The vector  $\mathbf{f}$  reduces to

$$\mathbf{f} = \begin{bmatrix} \boxed{\mathcal{L}_\rho (\nabla \tilde{p}', \nabla^2 \tilde{p}') + \mathfrak{D}_{(\rho, u')} - \frac{\mathbf{r}_\alpha^*}{\Delta t^2}} \\ \boxed{\mathcal{L}_e (\nabla \tilde{p}', \nabla^2 \tilde{p}') + \mathfrak{D}_{(e, u')} - \frac{\mathbf{r}_e^*}{\Delta t^2}} \end{bmatrix} \quad (4.97)$$

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

71

where

$$\boxed{\mathbf{r}_\alpha^*} = \bar{\rho}^* - \bar{\rho}^n + \Delta t \nabla_\rho \cdot \left( \bar{\rho}^* \left[ \tilde{\mathbf{v}}^* - \frac{\mathbf{r}_v^*}{\bar{\rho}^* - \zeta_v^*} \right] \right) - \gamma_\rho^* - \zeta_\rho^* \cdot \frac{\mathbf{r}_v^*}{\bar{\rho}^* - \zeta_v^*} \quad (4.98)$$

$$\begin{aligned} \boxed{\mathcal{L}_\rho(\nabla \bar{p}', \nabla^2 \bar{p}')} &= \frac{1}{\Delta t (\bar{\rho}^* - \zeta_v^*)} \zeta_\rho^* \cdot \left( \nabla_v \bar{p}' - \nabla_v \cdot \left( \bar{\rho}_p^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \bar{p}' \right) \right) - \\ &- \frac{1}{\Delta t} \nabla_\rho \cdot \left( \left[ \bar{\rho}_p^* \tilde{\mathbf{v}}^* + \bar{\rho}^* \frac{\nu_v^* - \bar{\rho}_p^* \tilde{\mathbf{v}}^*}{\bar{\rho}^* - \zeta_v^*} \right] \bar{p}' + \frac{\Delta t \bar{\rho}^*}{\bar{\rho}^* - \zeta_v^*} \left[ \nabla_v \bar{p}' - \nabla_v \cdot \left( \bar{\rho}_p^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \bar{p}' \right) \right] \right) \end{aligned} \quad (4.99)$$

$$\begin{aligned} \boxed{\mathcal{D}_{(\rho, u')}} &= -\frac{1}{\Delta t (\bar{\rho}^* - \zeta_v^*)} \zeta_\rho^* \cdot \left( \nabla_v \cdot \left( \bar{\rho}_u^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \tilde{u}' \right) \right) - \\ &- \frac{1}{\Delta t} \nabla_\rho \cdot \left( \left[ \bar{\rho}_u^* \tilde{\mathbf{v}}^* + \frac{\bar{\rho}^* \hat{\eta}_v^*}{\bar{\rho}^* - \zeta_v^*} \right] \tilde{u}' - \frac{\Delta t \bar{\rho}^*}{\bar{\rho}^* - \zeta_v^*} \left[ \nabla_v \cdot \left( \bar{\rho}_u^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \tilde{u}' \right) \right] \right) \end{aligned} \quad (4.100)$$

$$\begin{aligned} \boxed{\mathbf{r}_e^*} &= \bar{\rho}^* \tilde{e}^* - (\bar{\rho} \tilde{e})^n - \gamma_e^* - \left( \bar{\rho}^* \tilde{\mathbf{v}}^* - \zeta_e^* \right) \cdot \frac{\mathbf{r}_v^*}{\bar{\rho}^* - \zeta_v^*} + \\ &+ \Delta t \nabla_e \cdot \left( \left( \bar{\rho}^* \tilde{e}^* + \bar{p}^* \right) \left( \tilde{\mathbf{v}}^* - \frac{\mathbf{r}_v^*}{\bar{\rho}^* - \zeta_v^*} \right) - \bar{\rho}^* \tilde{\mathbf{v}}^* \left( \tilde{\mathbf{v}}^* \cdot \frac{\mathbf{r}_v^*}{\bar{\rho}^* - \zeta_v^*} \right) \right) \end{aligned} \quad (4.101)$$

$$\begin{aligned} \boxed{\mathcal{L}_e(\nabla \bar{p}', \nabla^2 \bar{p}')} &= -\frac{\bar{\rho}^* \tilde{\mathbf{v}}^* - \zeta_e^*}{\Delta t (\bar{\rho}^* - \zeta_v^*)} \cdot \left( \nabla_v \bar{p}' - \nabla_v \cdot \left( \bar{\rho}_p^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \bar{p}' \right) \right) - \\ &- \frac{1}{\Delta t} \nabla_{e_k} \cdot \left( \begin{aligned} &\left[ \left( 1 + \tilde{e}^* \bar{\rho}_p^* \right) \tilde{\mathbf{v}}^* + \frac{(\bar{\rho}^* \tilde{e}^* + \bar{p}^*)}{\bar{\rho}^* - \zeta_v^*} \left( \nu_v^* - \bar{\rho}_p^* \tilde{\mathbf{v}}^* \right) \right] \bar{p}' + \\ &+ \frac{\bar{\rho}^*}{\bar{\rho}^* - \zeta_v^*} \tilde{\mathbf{v}}^* \left[ \tilde{\mathbf{v}}^* \cdot \left( \nu_v^* - \bar{\rho}_p^* \tilde{\mathbf{v}}^* \right) \right] \bar{p}' + \\ &+ \frac{\Delta t \bar{\rho}^*}{\bar{\rho}^* - \zeta_v^*} \tilde{\mathbf{v}}^* \left( \tilde{\mathbf{v}}^* \cdot \left[ \nabla_v \bar{p}' - \nabla_v \cdot \left( \bar{\rho}_p^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \bar{p}' \right) \right] \right) \right] + \\ &+ \frac{\Delta t (\bar{\rho}^* \tilde{e}^* + \bar{p}^*)}{\bar{\rho}^* - \zeta_v^*} \left[ \nabla_v \bar{p}' - \nabla_v \cdot \left( \bar{\rho}_p^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \bar{p}' \right) \right] \end{aligned} \right) \end{aligned} \quad (4.102)$$

$$\begin{aligned} \boxed{\mathcal{D}_{(e, u')}} &= \frac{\bar{\rho}^* \tilde{\mathbf{v}}^* - \zeta_e^*}{\Delta t (\bar{\rho}^* - \zeta_v^*)} \cdot \left( \nabla_v \cdot \left( \bar{\rho}_u^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \tilde{u}' \right) \right) - \\ &- \frac{1}{\Delta t} \nabla_e \cdot \left( \begin{aligned} &\left[ \left( \bar{\rho}^* + \tilde{e}^* \bar{\rho}_u^* \right) \tilde{\mathbf{v}}^* + \frac{(\bar{\rho}^* \tilde{e}^* + \bar{p}^*)}{\bar{\rho}^* - \zeta_v^*} \left( \eta_v^* - \bar{\rho}_u^* \tilde{\mathbf{v}}^* \right) \right] \tilde{u}' + \\ &+ \frac{\bar{\rho}^*}{\bar{\rho}^* - \zeta_v^*} \tilde{\mathbf{v}}^* \left[ \tilde{\mathbf{v}}^* \cdot \left( \eta_v^* - \bar{\rho}_u^* \tilde{\mathbf{v}}^* \right) \right] \tilde{u}' + \\ &- \frac{\Delta t \bar{\rho}^*}{\bar{\rho}^* - \zeta_v^*} \tilde{\mathbf{v}}^* \left( \tilde{\mathbf{v}}^* \cdot \left[ \nabla_v \cdot \left( \bar{\rho}_u^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \tilde{u}' \right) \right] \right) \right] + \\ &- \frac{\Delta t (\bar{\rho}^* \tilde{e}^* + \bar{p}^*)}{\bar{\rho}^* - \zeta_v^*} \left( \nabla_v \cdot \left( \bar{\rho}_u^* (\tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) \tilde{u}' \right) \right) \end{aligned} \right) \end{aligned} \quad (4.103)$$

and

$$\boxed{\mathbf{r}_v^*} = \bar{\rho}^* \tilde{\mathbf{v}}^* - (\bar{\rho} \tilde{\mathbf{v}})^n - \gamma_v^* + \Delta t [\nabla_v \cdot (\bar{\rho}^* \tilde{\mathbf{v}}^* \otimes \tilde{\mathbf{v}}^*) - \nabla_v \bar{p}^*] \quad (4.104)$$

With these, the single-fluid version of the P'HE equation<sup>24</sup> (4.86) becomes:

$$\frac{\Delta t^2}{\bar{\rho}^*} \left( \sigma_{(\rho, u')} f_2 - \sigma_{(e, u')} f_1 - \bar{p}' \mathcal{D} \right) = 0 \quad (4.107)$$

where  $\mathcal{D} = \left( \sigma_{(\rho, u')} \sigma_{(e, p')} - \sigma_{(\rho, p')} \sigma_{(e, u')} \right)$  is a determinant of the matrix  $\mathbb{W}$ . The wavenumber associated with this P'HE is

$$\kappa^2 = \Delta t^2 \frac{\sigma_{(\rho, p')} \sigma_{(e, u')} - \sigma_{(\rho, u')} \sigma_{(e, p')}}{\bar{\rho}^*} \quad (4.108)$$

To simplify discussion, let's ignore the sources in mass, momentum and energy equations<sup>25</sup>. Then,

$$\kappa^2 = \frac{1}{c_{s\bar{p}}^2 \Delta t^2} \quad (4.109)$$

where the square of speed of sound is defined as  $c_{s\bar{p}}^2 \equiv \frac{1}{\bar{\rho}_{\bar{p}}^*}$ .

Note, that when compressibility of a fluid reduces,  $c_{s\bar{p}} \rightarrow \infty$  (as  $\bar{\rho}_{\bar{p}}^* \rightarrow 0$ ), and eq.(4.107) approaches the *pressure Poisson* form,  $\kappa \rightarrow 0$ .

<sup>24</sup> Strictly speaking, Helmholtz equation is defined as

$$\nabla^2 P + \kappa^2 P = 0 \quad (4.105)$$

while eq.(4.107) is more complex:

$$\nabla \cdot (a \nabla P) + \kappa^2 P + \mathbf{b} \cdot \nabla P + \nabla \cdot (\mathbf{c} P) = d \quad (4.106)$$

<sup>25</sup>It can be seen that the sources do change the wavenumber, and, therefore, the effective speed of sound.

### 4.3.10 P'HE for 2-fluid formulation

The vector of unknowns for two-fluid P'HE system is  $\begin{bmatrix} \alpha'_0 \\ \tilde{u}'_0 \\ \tilde{u}'_1 \\ \bar{p}' \end{bmatrix}$ , where we used  $\epsilon=1$ .

In the case of “mixture system” formulation, the wavenumber matrix  $\mathbb{W}$  becomes

$$\mathbb{W} = \begin{array}{|c|c|c|c|} \hline \sigma_{(\rho,\alpha')}^{(0)} - \sigma_{(\rho,\alpha')}^{(0,1)} & \sigma_{(\rho,u')}^{(0)} & \sigma_{(\rho,u')}^{(0,1)} & \sigma_{(\rho,p')}^{(0)} \\ \hline \sigma_{(e,\alpha')}^{(0)} - \sigma_{(e,\alpha')}^{(0,1)} & \sigma_{(e,u')}^{(0)} & \sigma_{(e,u')}^{(0,1)} & \sigma_{(e,p')}^{(0)} \\ \hline \sigma_{(\rho,\alpha')}^{(0)} - \sigma_{(\rho,\alpha')}^{(1)} + \sigma_{(\rho,\alpha')}^{(1,0)} - \sigma_{(\rho,\alpha')}^{(0,1)} & \sigma_{(\rho,u')}^{(0)} + \sigma_{(\rho,u')}^{(1,0)} & \sigma_{(\rho,u')}^{(1)} + \sigma_{(\rho,u')}^{(0,1)} & \sigma_{(\rho,p')}^{(0)} + \sigma_{(\rho,p')}^{(1)} \\ \hline \sigma_{(e,\alpha')}^{(0)} - \sigma_{(e,\alpha')}^{(1)} + \sigma_{(e,\alpha')}^{(1,0)} - \sigma_{(e,\alpha')}^{(0,1)} & \sigma_{(e,u')}^{(0)} + \sigma_{(e,u')}^{(1,0)} & \sigma_{(e,u')}^{(1)} + \sigma_{(e,u')}^{(0,1)} & \sigma_{(e,p')}^{(0)} + \sigma_{(e,p')}^{(1)} \\ \hline \end{array} \quad (4.110)$$

where

$$\begin{aligned} \sigma_{(\rho,\alpha')}^{(0)} &= \frac{1}{\Delta t^2} \left[ \begin{array}{l} \bar{\rho}_0^* - \mu_{\rho(0,0)}^* - \zeta_{\rho(0,0)}^* \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{v(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{v(0,1)}^* \right) - \\ - \zeta_{\rho(1,0)}^* \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{v(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{v(0,1)}^* \right) \end{array} \right] \\ \sigma_{(\rho,u')}^{(0)} &= \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_0^* \bar{\rho}_{\tilde{u}_0}^* - \eta_{\rho(0,0)}^* - \zeta_{\rho(0,0)}^* \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{v(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{v(0,1)}^* \right) - \\ - \zeta_{\rho(1,0)}^* \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{v(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{v(0,1)}^* \right) \end{array} \right] \\ \sigma_{(\rho,p')}^{(0)} &= \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_0^* \bar{p}_0^* - \nu_{\rho_0}^* - \zeta_{\rho(0,0)}^* \cdot \left( \frac{\mathfrak{F}_{(0,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(0,1)}^{-1}}{\omega_1^*} \right) - \\ - \zeta_{\rho(1,0)}^* \cdot \left( \frac{\mathfrak{F}_{(1,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(1,1)}^{-1}}{\omega_1^*} \right) \end{array} \right] \end{aligned} \quad (4.111)$$

$$\begin{aligned}
\sigma_{(\rho, \alpha')}^{(1)} &= \frac{1}{\Delta t^2} \left[ \begin{array}{l} \bar{\rho}_1^* - \mu_{\rho(1,1)}^* - \zeta_{\rho(0,1)}^* \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) - \\ - \zeta_{\rho(1,1)}^* \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) \end{array} \right] \\
\sigma_{(\rho, u')}^{(1)} &= \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_1^* \bar{\rho}_{\tilde{u}_1}^* - \eta_{\rho(1,1)}^* - \zeta_{\rho(0,1)}^* \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) - \\ - \zeta_{\rho(1,1)}^* \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) \end{array} \right] \quad (4.112) \\
\sigma_{(\rho, p')}^{(1)} &= \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_1^* \bar{\rho}_{\tilde{p}_1}^* - \nu_{\rho_1}^* - \zeta_{\rho(0,1)}^* \cdot \left( \frac{\mathfrak{F}_{(0,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(0,1)}^{-1}}{\omega_1^*} \right) - \\ - \zeta_{\rho(1,1)}^* \cdot \left( \frac{\mathfrak{F}_{(1,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(1,1)}^{-1}}{\omega_1^*} \right) \end{array} \right]
\end{aligned}$$

$$\sigma_{(e, \alpha')}^{(0)} = \frac{1}{\Delta t^2} \left[ \begin{array}{l} \bar{\rho}_0^* \tilde{e}_0^* - \mu_{e(0,0)}^* - \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \mu_{\delta \tilde{p}_1(0)}^* - \\ - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \tilde{p}_1}^* \right) \nabla_{e_0} \alpha_0^* - \alpha_0^* \bar{\rho}_0^* \tilde{\mathbf{v}}_0^* \right] \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(0,1)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \tilde{p}_1(0)}^* + \zeta_{e(0,0)}^* \right] \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(0,1)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \tilde{p}_1(1)}^* + \zeta_{e(1,0)}^* \right] \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(0,1)}^* \right) \end{array} \right] \quad (4.113)$$

$$\sigma_{(e, \alpha')}^{(1)} = \frac{1}{\Delta t^2} \left[ \begin{array}{l} \bar{\rho}_1^* \tilde{e}_1^* - \mu_{e(1,1)}^* - \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \mu_{\delta \tilde{p}_1(1)}^* - \\ - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \tilde{p}_1}^* \right) \nabla_{e_1} \alpha_1^* - \alpha_1^* \bar{\rho}_1^* \tilde{\mathbf{v}}_1^* \right] \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \tilde{p}_1(0)}^* + \zeta_{e(0,1)}^* \right] \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \tilde{p}_1(1)}^* + \zeta_{e(1,1)}^* \right] \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) \end{array} \right] \quad (4.114)$$

$$\sigma_{(e, \alpha')}^{(0)} = \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_0^* \left( \bar{\rho}_0^* + \tilde{e}_0^* \bar{\rho}_{\tilde{u}_0}^* \right) - \eta_{e(0,0)}^* - \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \eta_{\delta \tilde{p}_1(0)}^* - \\ - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \tilde{p}_1}^* \right) \nabla_{e_0} \alpha_0^* - \alpha_0^* \bar{\rho}_0^* \tilde{\mathbf{v}}_0^* \right] \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(0,1)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \tilde{p}_1(0)}^* + \zeta_{e(0,0)}^* \right] \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(0,1)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \tilde{p}_1(1)}^* + \zeta_{e(1,0)}^* \right] \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(0,1)}^* \right) \end{array} \right] \quad (4.115)$$

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

75

$$\sigma_{(e,u')}^{(1)} = \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_1^* \left( \bar{\rho}_1^* + \tilde{e}_1^* \bar{\rho}_{\tilde{u}_1}^* \right) - \eta_{e(1,1)}^* - \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \eta_{\delta \bar{p}_1(1)}^* - \\ - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_1} \alpha_1^* - \alpha_1^* \bar{\rho}_1^* \tilde{\mathbf{v}}_1^* \right] \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_1(0)}^* + \zeta_{e(0,1)}^* \right] \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_1(1)}^* + \zeta_{e(1,1)}^* \right] \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) \end{array} \right] \quad (4.116)$$

$$\sigma_{(e,p')}^{(0)} = \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_0^* \tilde{e}_0^* \bar{\rho}_{\bar{p}_0}^* - \nu_{e_0}^* - \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \left( 1 + \nu_{\delta \bar{p}_1}^* \right) - \\ - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_0} \alpha_0^* - \alpha_0^* \bar{\rho}_0^* \tilde{\mathbf{v}}_0^* \right] \cdot \left( \frac{\mathfrak{F}_{(0,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(0,1)}^{-1}}{\omega_1^*} \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \bar{p}_1(0)}^* + \zeta_{e(0,0)}^* \right] \cdot \left( \frac{\mathfrak{F}_{(0,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(0,1)}^{-1}}{\omega_1^*} \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \bar{p}_1(1)}^* + \zeta_{e(1,0)}^* \right] \cdot \left( \frac{\mathfrak{F}_{(1,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(1,1)}^{-1}}{\omega_1^*} \right) \end{array} \right] \quad (4.117)$$

$$\sigma_{(e,p')}^{(1)} = \frac{1}{\Delta t^2} \left[ \begin{array}{l} \alpha_1^* \tilde{e}_1^* \bar{\rho}_{\bar{p}_1}^* - \nu_{e_1}^* - \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \left( 1 + \nu_{\delta \bar{p}_1}^* \right) - \\ - \left[ \Delta t \left( \bar{p}^* + \gamma_{\delta \bar{p}_1}^* \right) \nabla_{e_1} \alpha_1^* - \alpha_1^* \bar{\rho}_1^* \tilde{\mathbf{v}}_1^* \right] \cdot \left( \frac{\mathfrak{F}_{(1,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(1,1)}^{-1}}{\omega_1^*} \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_1(0)}^* + \zeta_{e(0,1)}^* \right] \cdot \left( \frac{\mathfrak{F}_{(0,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(0,1)}^{-1}}{\omega_1^*} \right) - \\ - \left[ \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_1(1)}^* + \zeta_{e(1,1)}^* \right] \cdot \left( \frac{\mathfrak{F}_{(1,0)}^{-1}}{\omega_0^*} + \frac{\mathfrak{F}_{(1,1)}^{-1}}{\omega_1^*} \right) \end{array} \right] \quad (4.118)$$

$$\sigma_{(\rho,\alpha')}^{(0,1)} = -\frac{\mu_{\rho(1,0)}^*}{\Delta t^2} - \frac{\zeta_{\rho(0,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) - \\ - \frac{\zeta_{\rho(1,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) \quad (4.119)$$

$$\sigma_{(\rho,\alpha')}^{(1,0)} = -\frac{\mu_{\rho(0,1)}^*}{\Delta t^2} - \frac{\zeta_{\rho(0,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(0,1)}^* \right) - \\ - \frac{\zeta_{\rho(1,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(0,1)}^* \right) \quad (4.120)$$

$$\sigma_{(\rho,u')}^{(0,1)} = -\frac{\eta_{\rho(1,0)}^*}{\Delta t^2} - \frac{\zeta_{\rho(0,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) - \\ - \frac{\zeta_{\rho(1,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) \quad (4.121)$$

$$\begin{aligned} \sigma_{(\rho, u')}^{(1,0)} = & -\frac{\eta_{\rho(0,1)}^*}{\Delta t^2} - \frac{\zeta_{\rho(0,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(0,1)}^* \right) - \\ & -\frac{\zeta_{\rho(1,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(0,1)}^* \right) \end{aligned} \quad (4.122)$$

$$\begin{aligned} \sigma_{(e, \alpha')}^{(0,1)} = & -\frac{\mu_{e(1,0)}^* + \Delta t (\tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^*) \mu_{\delta \bar{p}_I(1)}^*}{\Delta t^2} - \\ & -\frac{\Delta t \left[ \left( \bar{p}^* + \gamma_{\delta \bar{p}_I}^* \right) \nabla_{e_0} \alpha_0^* + \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \bar{p}_I(0)}^* \right] - \alpha_0^* \bar{\rho}_0^* \tilde{\mathbf{v}}_0^* + \zeta_{e(0,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) - \\ & -\frac{\Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \bar{p}_I(1)}^* + \zeta_{e(1,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(1,0)}^* \right) \end{aligned} \quad (4.123)$$

$$\begin{aligned} \sigma_{(e, \alpha')}^{(1,0)} = & -\frac{\mu_{e(0,1)}^* + \Delta t (\tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^*) \mu_{\delta \bar{p}_I(0)}^*}{\Delta t^2} - \\ & -\frac{\Delta t \left[ \left( \bar{p}^* + \gamma_{\delta \bar{p}_I}^* \right) \nabla_{e_1} \alpha_1^* + \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_I(1)}^* \right] - \alpha_1^* \bar{\rho}_1^* \tilde{\mathbf{v}}_1^* + \zeta_{e(1,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\mu}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\mu}_{\mathbf{v}(0,1)}^* \right) - \\ & -\frac{\Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_I(0)}^* + \zeta_{e(0,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\mu}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\mu}_{\mathbf{v}(0,1)}^* \right) \end{aligned} \quad (4.124)$$

$$\begin{aligned} \sigma_{(e, u')}^{(0,1)} = & -\frac{\eta_{e(1,0)}^* + \Delta t (\tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^*) \eta_{\delta \bar{p}_I(1)}^*}{\Delta t^2} - \\ & -\frac{\Delta t \left[ \left( \bar{p}^* + \gamma_{\delta \bar{p}_I}^* \right) \nabla_{e_0} \alpha_0^* + \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \bar{p}_I(0)}^* \right] - \alpha_0^* \bar{\rho}_0^* \tilde{\mathbf{v}}_0^* + \zeta_{e(0,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) - \\ & -\frac{\Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) \zeta_{\delta \bar{p}_I(1)}^* + \zeta_{e(1,0)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(1)}^* + \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(1,0)}^* \right) \end{aligned} \quad (4.125)$$

$$\begin{aligned} \sigma_{(e, u')}^{(1,0)} = & -\frac{\eta_{e(0,1)}^* + \Delta t (\tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^*) \eta_{\delta \bar{p}_I(0)}^*}{\Delta t^2} - \\ & -\frac{\Delta t \left[ \left( \bar{p}^* + \gamma_{\delta \bar{p}_I}^* \right) \nabla_{e_1} \alpha_1^* + \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_I(1)}^* \right] - \alpha_1^* \bar{\rho}_1^* \tilde{\mathbf{v}}_1^* + \zeta_{e(1,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(1,0)}^{-1} \hat{\eta}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(1,1)}^{-1} \hat{\eta}_{\mathbf{v}(0,1)}^* \right) - \\ & -\frac{\Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) \zeta_{\delta \bar{p}_I(0)}^* + \zeta_{e(0,1)}^*}{\Delta t^2} \cdot \left( \mathfrak{F}_{(0,0)}^{-1} \hat{\eta}_{\mathbf{v}(0)}^* + \mathfrak{F}_{(0,1)}^{-1} \hat{\eta}_{\mathbf{v}(0,1)}^* \right) \end{aligned} \quad (4.126)$$

### 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

77

In the case of “*phasic system*” formulation, the wavenumber matrix  $\mathbb{W}$  is

$$\mathbb{W} = \begin{bmatrix} \sigma_{(\rho,\alpha')}^{(0)} - \sigma_{(\rho,\alpha')}^{(0,1)} & \sigma_{(\rho,u')}^{(0)} & \sigma_{(\rho,u')}^{(0,1)} & \sigma_{(\rho,p')}^{(0)} \\ \sigma_{(e,\alpha')}^{(0)} - \sigma_{(e,\alpha')}^{(0,1)} & \sigma_{(e,u')}^{(0)} & \sigma_{(e,u')}^{(0,1)} & \sigma_{(e,p')}^{(0)} \\ \sigma_{(\rho,\alpha')}^{(1,0)} - \sigma_{(\rho,\alpha')}^{(1)} & \sigma_{(\rho,u')}^{(1,0)} & \sigma_{(\rho,u')}^{(1)} & \sigma_{(\rho,p')}^{(1)} \\ \sigma_{(e,\alpha')}^{(1,0)} - \sigma_{(e,\alpha')}^{(1)} & \sigma_{(e,u')}^{(1,0)} & \sigma_{(e,u')}^{(1)} & \sigma_{(e,p')}^{(1)} \end{bmatrix} \quad (4.127)$$

Eq.(4.48) becomes:

$$\underbrace{\begin{bmatrix} \mathcal{F}_{1,1} & \mathcal{F}_{1,2} \\ \mathcal{F}_{2,1} & \mathcal{F}_{2,2} \end{bmatrix}}_{\mathfrak{F}} \begin{bmatrix} \tilde{\mathbf{v}}'_0 \\ \tilde{\mathbf{v}}'_1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}'_0 \\ \mathbf{m}'_1 \end{bmatrix} \quad (4.128)$$

where  $\mathcal{F}_{i,j}$  are operators acting on a vector  $\langle \dots \rangle$ , defined as:

$$\begin{aligned} \mathcal{F}_{1,1}(\langle \dots \rangle) &= \left( \alpha_0^* \bar{\rho}_0^* - \zeta_{\mathbf{v}(0,0)}^* \right) \langle \dots \rangle + \Delta t \left[ \left( \zeta_{\Delta \bar{p}(0,0)}^* + \zeta_{\delta \bar{p}_I(0)}^* \right) \cdot \langle \dots \rangle \right] \nabla_{\mathbf{v}_0} \alpha_0^* \\ \mathcal{F}_{1,2}(\langle \dots \rangle) &= \zeta_{\mathbf{v}(1,0)}^* \langle \dots \rangle + \Delta t \left[ \left( \zeta_{\Delta \bar{p}(1,0)}^* + \zeta_{\delta \bar{p}_I(1)}^* \right) \cdot \langle \dots \rangle \right] \nabla_{\mathbf{v}_0} \alpha_0^* \\ \mathcal{F}_{2,1}(\langle \dots \rangle) &= \zeta_{\mathbf{v}(0,1)}^* \langle \dots \rangle + \Delta t \left[ \left( \zeta_{\Delta \bar{p}(0,1)}^* + \zeta_{\delta \bar{p}_I(0)}^* \right) \cdot \langle \dots \rangle \right] \nabla_{\mathbf{v}_1} \alpha_1^* \\ \mathcal{F}_{2,2}(\langle \dots \rangle) &= \left( \alpha_1^* \bar{\rho}_1^* - \zeta_{\mathbf{v}(1,1)}^* \right) \langle \dots \rangle + \Delta t \left[ \left( \zeta_{\Delta \bar{p}(1,1)}^* + \zeta_{\delta \bar{p}_I(1)}^* \right) \cdot \langle \dots \rangle \right] \nabla_{\mathbf{v}_1} \alpha_1^* \end{aligned} \quad (4.129)$$

These operators are sufficient to define  $\mathfrak{F}_{(k,m)}^{-1}$ , which are necessary for computation of the wavenumber matrix.

#### **Case-1: Ignoring closures and pressure differences**

Lets consider a particular case when we ignore source terms (and all closures) dependence on the vector of unknowns, and also drop dynamic interfacial pressure

and phasic bulk pressure difference terms. In this case,

$$\mathfrak{F}_{(k,m)}^{-1} = \begin{cases} \frac{1}{\alpha_k^* \bar{\rho}_k^*} & \text{if } k = m \\ 0 & \text{otherwise} \end{cases} \quad (4.130)$$

$$\begin{aligned} \sigma_{(\rho,\alpha')}^{(0)} &= \frac{\bar{\rho}_0^*}{\Delta t^2}; & \sigma_{(\rho,u')}^{(0)} &= \frac{\alpha_0^* \bar{\rho}_{\tilde{u}_0}^*}{\Delta t^2}; & \sigma_{(\rho,p')}^{(0)} &= \frac{\bar{\rho}_{\bar{p}_0}^*}{\Delta t^2} \\ \sigma_{(\rho,\alpha')}^{(1)} &= \frac{\bar{\rho}_1^*}{\Delta t^2}; & \sigma_{(\rho,u')}^{(1)} &= \frac{\alpha_1^* \bar{\rho}_{\tilde{u}_1}^*}{\Delta t^2}; & \sigma_{(\rho,p')}^{(1)} &= \frac{\bar{\rho}_{\bar{p}_1}^*}{\Delta t^2} \end{aligned} \quad (4.131)$$

$$\sigma_{(e,\alpha')}^{(0)} = \frac{\bar{\rho}_0^* \tilde{e}_0^* - \left( \frac{\Delta t \bar{p}^*}{\alpha_0^* \bar{\rho}_0^*} \nabla_{e_0} \alpha_0^* - \tilde{\mathbf{v}}_0^* \right) \cdot (\Delta t \nabla_{\mathbf{v}_0} \bar{p}^* - \bar{\rho}_0^* \tilde{\mathbf{v}}_0^*)}{\Delta t^2} \quad (4.132)$$

$$\sigma_{(e,\alpha')}^{(1)} = \frac{\bar{\rho}_1^* \tilde{e}_1^* - \left( \frac{\Delta t \bar{p}^*}{\alpha_1^* \bar{\rho}_1^*} \nabla_{e_1} \alpha_1^* - \tilde{\mathbf{v}}_1^* \right) \cdot (\Delta t \nabla_{\mathbf{v}_1} \bar{p}^* - \bar{\rho}_1^* \tilde{\mathbf{v}}_1^*)}{\Delta t^2} \quad (4.133)$$

$$\sigma_{(e,u')}^{(0)} = \frac{\alpha_0^* \left( \bar{\rho}_0^* + \tilde{e}_0^* \bar{\rho}_{\tilde{u}_0}^* \right) + \left( \frac{\Delta t \bar{p}^*}{\bar{\rho}_0^*} \nabla_{e_0} \alpha_0^* - \alpha_0^* \tilde{\mathbf{v}}_0^* \right) \cdot \left( \bar{\rho}_{\tilde{u}_0}^* \tilde{\mathbf{v}}_0^* \right)}{\Delta t^2} \quad (4.134)$$

$$\sigma_{(e,u')}^{(1)} = \frac{\alpha_1^* \left( \bar{\rho}_1^* + \tilde{e}_1^* \bar{\rho}_{\tilde{u}_1}^* \right) + \left( \frac{\Delta t \bar{p}^*}{\bar{\rho}_1^*} \nabla_{e_1} \alpha_1^* - \alpha_1^* \tilde{\mathbf{v}}_1^* \right) \cdot \left( \bar{\rho}_{\tilde{u}_1}^* \tilde{\mathbf{v}}_1^* \right)}{\Delta t^2} \quad (4.135)$$

$$\sigma_{(e,p')}^{(0)} = \frac{\alpha_0^* \tilde{e}_0^* \bar{\rho}_{\bar{p}_0}^* - \Delta t \left( \tilde{\mathbf{v}}_0^* \cdot \nabla_{e_0} \alpha_0^* \right) + \left( \frac{\Delta t \bar{p}^*}{\bar{\rho}_0^*} \nabla_{e_0} \alpha_0^* - \alpha_0^* \tilde{\mathbf{v}}_0^* \right) \cdot \left( \bar{\rho}_{\bar{p}_0}^* \tilde{\mathbf{v}}_0^* \right)}{\Delta t^2} \quad (4.136)$$

$$\sigma_{(e,p')}^{(1)} = \frac{\alpha_1^* \tilde{e}_1^* \bar{\rho}_{\bar{p}_1}^* - \Delta t \left( \tilde{\mathbf{v}}_1^* \cdot \nabla_{e_1} \alpha_1^* \right) + \left( \frac{\Delta t \bar{p}^*}{\bar{\rho}_1^*} \nabla_{e_1} \alpha_1^* - \alpha_1^* \tilde{\mathbf{v}}_1^* \right) \cdot \left( \bar{\rho}_{\bar{p}_1}^* \tilde{\mathbf{v}}_1^* \right)}{\Delta t^2} \quad (4.137)$$

**4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 79**

and

$$\begin{aligned} \sigma_{(\rho,\alpha')}^{(0,1)} = 0; \quad \sigma_{(\rho,\alpha')}^{(1,0)} = 0; \quad \sigma_{(\rho,u')}^{(0,1)} = 0; \quad \sigma_{(\rho,u')}^{(1,0)} = 0; \\ \sigma_{(e,\alpha')}^{(0,1)} = 0; \quad \sigma_{(e,\alpha')}^{(1,0)} = 0; \quad \sigma_{(e,u')}^{(0,1)} = 0; \quad \sigma_{(e,u')}^{(1,0)} = 0; \end{aligned} \tag{4.138}$$

For the wavenumber analysis, we will use the “phasic system” wavenumber matrix eq.(4.127)<sup>26</sup>, in which each phase mass and energy equations are scaled by  $\frac{\Delta t^2}{\bar{\rho}_k^*}$ . It can be shown that the wavenumber of the final pressure correction P’HE equation is

$$\kappa^2 = -\frac{\bar{\rho}_0^* \mathcal{D}}{\alpha_0 \alpha_1 \Delta t^2} \tag{4.139}$$

where  $\mathcal{D}$  is the determinant<sup>27</sup> of the matrix  $\hat{\mathbb{W}}$ :

$$\hat{\mathbb{W}} = \Delta t^2 \begin{bmatrix} \frac{\sigma_{(\rho,\alpha')}^{(0)}}{\bar{\rho}_0^*} & \frac{\sigma_{(\rho,u')}^{(0)}}{\bar{\rho}_0^*} & 0 & \frac{\sigma_{(\rho,p')}^{(0)}}{\bar{\rho}_0^*} \\ \frac{\sigma_{(e,\alpha')}^{(0)}}{\bar{\rho}_0^*} & \frac{\sigma_{(e,u')}^{(0)}}{\bar{\rho}_0^*} & 0 & \frac{\sigma_{(e,p')}^{(0)}}{\bar{\rho}_0^*} \\ -\frac{\sigma_{(\rho,\alpha')}^{(1)}}{\bar{\rho}_1^*} & 0 & \frac{\sigma_{(\rho,u')}^{(1)}}{\bar{\rho}_1^*} & \frac{\sigma_{(\rho,p')}^{(1)}}{\bar{\rho}_1^*} \\ -\frac{\sigma_{(e,\alpha')}^{(1)}}{\bar{\rho}_1^*} & 0 & \frac{\sigma_{(e,u')}^{(1)}}{\bar{\rho}_1^*} & \frac{\sigma_{(e,p')}^{(1)}}{\bar{\rho}_1^*} \end{bmatrix} \tag{4.140}$$

<sup>26</sup>It can be seen that in this case the wavenumber matrix eq.(4.140) has four zero elements, in difference to the similar matrix of the “mixture system”, where only two elements are zero. Otherwise, these two system are mathematically identical.

<sup>27</sup>Note that the determinant  $\mathcal{D}$  is a product of  $\alpha_0 \alpha_1$ , which indicates a degeneracy when one of the phase disappears – i.e. the wavenumber matrix  $\mathbb{W}$  is uninvertable. This is the case for both the “phasic” and “mixture” systems. Numerically, this requires a special treatment near or at degenerate points.

After collecting terms, equation (4.139) becomes:

$$\begin{aligned} \kappa^2 &= \kappa_{\text{acou},0}^2 + \kappa_{\text{acou},1}^2 + \kappa_{\text{topo},0}^2 + \kappa_{\text{topo},1}^2 = \\ &= \frac{\omega_{\text{acou},0}^2 + \omega_{\text{int},0}^2 + (\omega_{\text{acou},1}^2 + \omega_{\text{int},1}^2) \frac{\bar{\rho}_0^*}{\bar{\rho}_1^*}}{\Delta t^2} \end{aligned} \quad (4.141)$$

where we splitted the wavenumber into four harmonics. The first two are associated with *phase acoustics*:

$$\begin{aligned} \omega_{\text{acou},0}^2 &= \frac{1}{c_{s\bar{p}_0}^2} \left( 1 + \alpha_1 \frac{u_0 \bar{\rho}_{\bar{u}_0}^*}{\bar{\rho}_0^*} \right) \\ \omega_{\text{acou},1}^2 &= \frac{1}{c_{s\bar{p}_1}^2} \left( 1 + \alpha_0 \frac{u_1 \bar{\rho}_{\bar{u}_1}^*}{\bar{\rho}_1^*} \right) \end{aligned} \quad (4.142)$$

where the (squares of) phasic speeds of sound are defined as

$$c_{s\bar{p}_k}^2 \equiv \frac{1}{\bar{\rho}_{\bar{p}_k}^*}$$

(pressure perturbation component); and  $u_k$  is defined as

$$u_k = \tilde{\epsilon}_k^* - (\tilde{\mathbf{v}}_k^*)^2$$

Note that the terms in boxes of eq.(4.142) are associated with *attenuation/accentuation of acoustic waves by the presence of another phase*. We can now define the effective speed of sound of two-phase media as

$$c_{2\phi} = \sqrt{\frac{1}{\frac{1}{c_{s\bar{p}_0}^2} \left( 1 + \alpha_1 \frac{u_0 \bar{\rho}_{\bar{u}_0}^*}{\bar{\rho}_0^*} \right) + \frac{1}{c_{s\bar{p}_1}^2} \left( 1 + \alpha_0 \frac{u_1 \bar{\rho}_{\bar{u}_1}^*}{\bar{\rho}_1^*} \right)}}} \quad (4.143)$$

The other two harmonics emerged from the modeling of interfacial forces in the form  $\sim p\nabla\alpha$ , and associated with *interfacial forces and topology*. We group

## 4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS

81

these harmonics in the following form:

$$\begin{aligned}
 \omega_{\text{int},0}^2 = & \underbrace{\frac{\Delta t \bar{\rho}_{\tilde{u}_0}^* \nabla_{e_0} \alpha_0^* \cdot \tilde{\mathbf{v}}_0^*}{\bar{\rho}_0^*}}_{\omega_{(0,0)}^2} \left( 1 - \frac{\Delta t^2 \bar{\rho}_{\tilde{u}_1}^* \bar{p}^*}{\alpha_1^* \bar{\rho}_1^{*3}} \left( \nabla_{e_0} \alpha_0^* \cdot \nabla_{v_0} \bar{p}^* \right) \right) \left( 1 + \frac{\alpha_1}{\alpha_0} \frac{\bar{p}^*}{\bar{\rho}_0^* c_{s\bar{p}_0}^2} \right) + \\
 & + \underbrace{\bar{\rho}_{\tilde{u}_0}^* \bar{\rho}_{\tilde{u}_1}^* \frac{\tilde{\mathbf{v}}_1^* \cdot \tilde{\mathbf{v}}_0^*}{\bar{\rho}_0^* \bar{\rho}_1^*}}_{\omega_{(0,1)}^2} \frac{\Delta t^2 \bar{p}^*}{\bar{\rho}_1^{*2} c_{s\bar{p}_1}^2} \frac{\alpha_0^*}{\alpha_1^*} \left( \nabla_{e_0} \alpha_0^* \cdot \nabla_{v_0} \bar{p}^* \right) \left( 1 + \frac{\alpha_1}{\alpha_0^2} \right) + \\
 & + \underbrace{\frac{\Delta t \bar{\rho}_{\tilde{u}_1}^*}{\bar{\rho}_1^{*2} c_{s\bar{p}_0}^2} \nabla_{v_0} \bar{p}^* \cdot \left( -\frac{\Delta t \bar{p}^*}{\alpha_1^* \bar{\rho}_1^*} \nabla_{e_0} \alpha_0^* - \tilde{\mathbf{v}}_1^* \right)}_{\omega_{(0,2)}^2} \left( 1 + \alpha_1 \frac{u_0 \bar{\rho}_{\tilde{u}_0}^*}{\bar{\rho}_0^*} \right)
 \end{aligned} \tag{4.144}$$

and

$$\begin{aligned}
 \omega_{\text{int},1}^2 = & - \underbrace{\frac{\Delta t \bar{\rho}_{\tilde{u}_1}^* \nabla_{e_1} \alpha_0^* \cdot \tilde{\mathbf{v}}_1^*}{\bar{\rho}_1^*}}_{\omega_{(1,0)}^2} \left( 1 + \frac{\Delta t^2 \bar{\rho}_{\tilde{u}_0}^* \bar{p}^*}{\alpha_0^* \bar{\rho}_0^{*3}} \left( \nabla_{e_1} \alpha_0^* \cdot \nabla_{v_1} \bar{p}^* \right) \right) \left( 1 + \frac{\alpha_0}{\alpha_1} \frac{\bar{p}^*}{\bar{\rho}_1^* c_{s\bar{p}_1}^2} \right) - \\
 & - \underbrace{\bar{\rho}_{\tilde{u}_0}^* \bar{\rho}_{\tilde{u}_1}^* \frac{\tilde{\mathbf{v}}_0^* \cdot \tilde{\mathbf{v}}_1^*}{\bar{\rho}_0^* \bar{\rho}_1^*}}_{\omega_{(1,1)}^2} \frac{\Delta t^2 \bar{p}^*}{\bar{\rho}_0^{*2} c_{s\bar{p}_0}^2} \frac{\alpha_1^*}{\alpha_0^*} \left( \nabla_{e_1} \alpha_0^* \cdot \nabla_{v_1} \bar{p}^* \right) \left( 1 + \frac{1}{\alpha_1} \frac{\bar{\rho}_0^* c_{s\bar{p}_0}^2}{\bar{\rho}_1^* c_{s\bar{p}_1}^2} \right) + \\
 & + \underbrace{\frac{\Delta t \bar{\rho}_{\tilde{u}_0}^*}{\bar{\rho}_0^{*2} c_{s\bar{p}_1}^2} \nabla_{v_1} \bar{p}^* \cdot \left( \frac{\Delta t \bar{p}^*}{\alpha_0^* \bar{\rho}_0^*} \nabla_{e_1} \alpha_0^* - \tilde{\mathbf{v}}_0^* \right)}_{\omega_{(1,2)}^2} \left( 1 + \alpha_0 \frac{u_1 \bar{\rho}_{\tilde{u}_1}^*}{\bar{\rho}_1^*} \right)
 \end{aligned} \tag{4.145}$$

The multipliers in boxes are dimensionless.

Eq.(4.141) can be interpreted as following. There are six waves in the eigen-structure of the two-fluid averaged equations:

- (1,2): Two *entropy waves* which propagate with material velocities  $\tilde{\mathbf{v}}_0^*$  and  $\tilde{\mathbf{v}}_1^*$ . These can be easily identified by looking at the Jacobian of governing equations using traditional linear eigenvalue analysis.
- (3,4): Two *acoustic waves* which are associated with fluid' equations of state. The wave numbers introduced by these waves are defined by eq.(4.142).
- (5,6): Two *internal (topology) waves* which are associated with interfacial forces and variations of void fractions. The wave numbers introduced by these waves are defined by eqs.(4.144) and (4.145). Importantly, these waves cannot be identified by traditional linear eigenvalue analysis of governing equations (looking at principal modes of the Jacobian), because *their wave speeds depend on multiphase flow topology* (gradients of void fraction and pressure), un-detectable during eigen-analysis of the Jacobian matrices<sup>28</sup>. By nature, these internal (topology) waves are different from the acoustic ones, as they are the carriers of pressure disturbances in two phase system, associated with (averaged) phase momentum interactions (forces) happening at material interfaces.

We argue that the model is **numerically ill-posed** when  $\kappa^2$  defined by eq.(4.141) is *negative*. Note that acoustic wave modes of  $\kappa^2$  are always positive. Internal (topology) wave modes however can be negative. When these negative modes overwhelm the acoustic modes, making  $\kappa^2 < 0$ , the pressure correction P'HE equation becomes **unsolvable**. Note that traditional linear eigen-analysis shows that only two (entropy) waves are *real*. The other four eigenvalues are *complex-conjugate pairs*<sup>29</sup>. Therefore, mathematically, the model is “*ill-posed*” and “*non-hyperbolic*”. In practice however, we do successfully use two-fluid models in reactor thermalhydraulics, for many safety-relevant transients. The analysis discussed above shows that the “*ill-posedness*” of the model should exhibit itself (numerically) only when  $\kappa^2 < 0$ . This should happen when the steep gradients of void fraction developed. These steep gradients are trouble not only when they make  $\kappa^2 < 0$ , but also when  $\kappa^2 \gg 0$ . This will induce very short waves into the solution. Under certain grid refinement, these waves become comparable to the

<sup>28</sup>Characteristic equation and corresponding eigenvalues do not contain any terms with spatial or temporal derivatives.

<sup>29</sup>When  $(\tilde{\mathbf{v}}_0^* - \tilde{\mathbf{v}}_1^*) \neq 0$  or when  $0 < \alpha_k < 1$ .

---

**4.3. INCREMENTAL FORM OF SEMI-IMPLICIT-BASED ALGORITHMS 83**


---

domain and grid scales, exhibiting themselves as oscillations emanating from void discontinuities. The typical example of this behavior is the Vic Ramson's water faucet problem [Ran87].

Finally, we note, that when approaching incompressible limit ( $\bar{\rho}_k = const$ ) by setting  $c_{s\bar{\rho}_k} \rightarrow \infty$  and  $\bar{\rho}_{\tilde{u}_k}^* \rightarrow 0$ , the acoustic and topology wavelenghtes become very large,  $\kappa^2 \rightarrow 0$ , and the P'HE reduces to the P'PE (Poisson, elliptic) form. In this case, there are no acoustic and internal (topology) waves, and the question of ill-posedness becomes irrelevant. Internal (topology) waves however do exist when  $\bar{\rho}_k = f(\tilde{T}_k)$  (e.g., when an acoustically-filtered formulation is used), as  $\bar{\rho}_{\tilde{u}_k}^* \neq 0$ .

## Chapter 5

# Segregated (Picard-Iteration) Algorithms

SEGREGATED algorithms utilize fully-implicit strategy to solve effective-field equations, using Picard type of iterations. The basic idea is to organize outer iteration loops around some operator-splitting algorithm, to successively update full solution vector, Figure 5.1. Each outer iteration is a sequence of implicit solves for one or a few coupled (“segregated”) equations (say, momentum-pressure, energy, turbulent kinetic energy, etc.), continuously updating solution vector iteratively in a Gauss-Seidel fashion. In difference to Newton-based algorithms discussed in Chapter 6, no residual slope (Jacobians) information are involved to guide outer iterations. Instead, a simple (Gauss-Seidel-like) iterative procedure with (possible) under-relaxations is utilized. Convergence of this type of fully-implicit solver is rather slow – in some occasions it might take hundreds of outer iterations to get an acceptable tolerance (if any). This is why the design of “segregated” steps is crucial for both efficiency and convergence (stability/robustness) of the algorithm.

The most popular approaches are derived from single-phase SIMPLE-based solvers [PS72, Pat80]. We will discuss these algorithms first, in Section 5.1.

### 5.1 SIMPLE-based algorithms

Probably the first SIMPLE-based algorithm for two-fluid model is the “*Inter-Phase Slip Algorithm*” (IPSA) developed by the Spalding group at Imperial College [Spa76, Spa80, Spa81, Spa83], followed by a number of variations. As doc-

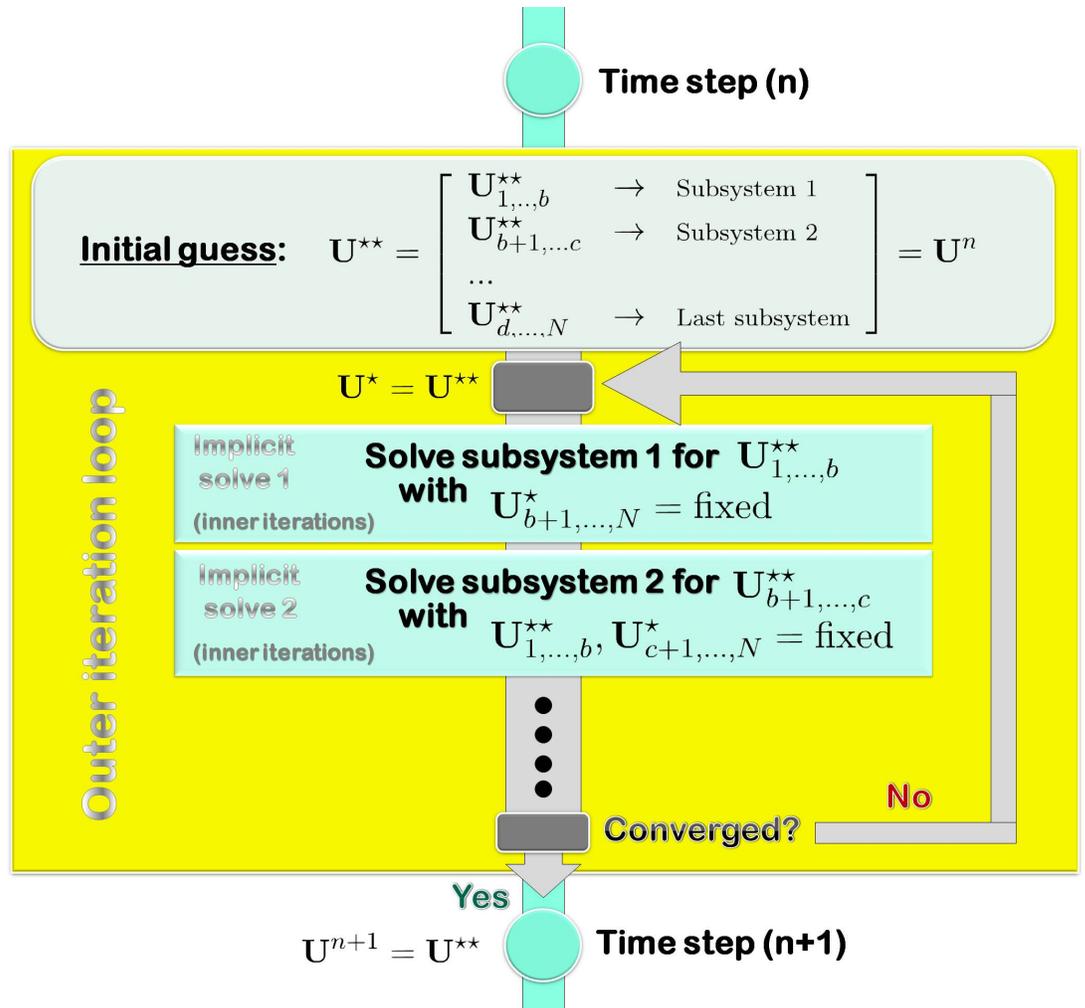


Fig. 5.1 : On basic concept of segregated algorithms.

umented by Darwish et al. [DMS01, MDS03, MD04], these algorithms can be divided into two major groups, based on the principle utilized to construct pressure correction equation: (1) *Mass Conservation-Based Algorithms (MCBA)* (Section 5.1.1) and (2) *Geometric Conservation-Based Algorithms (GCBA)* (Section 5.1.2).

Both algorithms are adaptations of SIMPLE-based single-phase flow algorithms [PS72, Pat80] to solve single-pressure effective-field models. As a consequence, the algorithms are focussed on coupling mass and momentum, following different recipes to construct pressure correction equation. Energy and turbulent transport equations are de-coupled and usually downstream in the pipeline of the segregated steps of the outer iteration loop. This is in contrast to reactor safety codes based on “semi-implicit” algorithm, Chapter 4, which derive pressure-Helmholtz equation more tightly coupled to energy equation. In reactor thermalhydraulics applications, tight coupling of mass-momentum-energy is crucial, due to importance of heat transfer on phase change (boiling/condensation). Another feature of SIMPLE-based algorithms - they are not well-suited for fully-compressible applications. There are *à la* ICE variations [DMS01], with linearizations of equation of state similar to eq.(4.7), but these will be referred to as weakly-compressible, as they are not the method of choice for high-speed (supersonic) applications, for which the methods of Chapter 3 are more appropriate and cost-effective.

### 5.1.1 Mass Conservation-Based Algorithms (MCBA)

MCBA algorithms utilize *global mass conservation equation*,

$$\sum_k \left[ \frac{\partial \left( \alpha_k \frac{\bar{\rho}_k}{\rho_k^{(\text{ref})}} \right)}{\partial t} + \nabla \cdot \left( \alpha_k \frac{\bar{\rho}_k \tilde{\mathbf{v}}_k}{\rho_k^{(\text{ref})}} \right) - \frac{\mathcal{S}_{\text{mass},k}(\mathbf{U})}{\rho_k^{(\text{ref})}} \right] = 0 \quad (5.1)$$

to derive a pressure-correction equation. Note, that each phase mass conservation equation is scaled by its reference density  $\rho_k^{(\text{ref})}$ , which is found necessary to balance [HB77] individual phase mass conservation errors, especially important when there are significant variations in phase densities (like in air-water configurations).

The outer iteration loop of Figure 5.1 is organized as following:

- 
- A. Solve implicitly phasic momentum equations. No mass mass conservation constraints are applied.
  - B. Form and solve (implicit) pressure-correction equation, based on global mass conservation eq.(5.1).
  - C. Correct phasic velocities, densities and pressure.
  - D. Solve (implicitly) phasic mass conservation equations for volume fractions.
  - E. Solve energy equations.
  - F. Solve turbulent transport equations (if any).
  - G. Check convergence.
- 

### Step A: Velocity predictor

Implicit discretization of phasic momentum equation renders the following linear discrete form at each computational cell  $(c)$ :

$$a_{c,c}^{(k)} \hat{\mathbf{v}}_{k(c)}^{**} + \underbrace{\sum_n^N a_{c,n}^{(k)} \hat{\mathbf{v}}_{k(n)}^{**}}_{\text{Coupling to neighbours}} + \underbrace{\sum_{m \neq k}^K c_c^{(k,m)} \left( \tilde{\mathbf{v}}_{k(c)}^{(?) } - \tilde{\mathbf{v}}_{m(c)}^{(?) } \right)}_{\text{Coupling to other phases (interfacial momentum exchange)}} = b_c^{(k)} \quad (5.2)$$

where all coefficients  $a_{i,j}^{(k)}$ ,  $b_i^{(k)}$  and  $c_i^{(k,m)}$  are evaluated using currently-available solution vector  $\mathbf{U}^*$  (in a Gauss-Seidel manner), as this is Picard type of iterations<sup>1</sup>. We denote phasic velocities by  $\hat{(\cdot)}$ , emphasizing that these are “predictor”

<sup>1</sup>Non-linear momentum advection terms can be linearized similar to eq.(4.17) of *Nearly-Implicit* algorithm:

$$[\bar{\rho}_k \tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k] \rightarrow \bar{\rho}_k^* \tilde{\mathbf{v}}_k^* \otimes \hat{\mathbf{v}}_k^{**} \quad (5.3)$$

velocities fields, which will be corrected at Step B to satisfy mass conservation.  $N$  is the number of neighboring cells (this and the values of  $a_{i,j}^{(k)}$  depend of the chosed space discretization scheme).  $K$  is the total number of phases. The source term  $b_i^{(k)}$  incorporate all local body forces, closure laws (those which are independent of  $\tilde{\mathbf{v}}$ ) and pressure gradient terms. Superscript  $(\cdot)^{(?)}$  on velocities in the interfacial momentum exchange terms denote different treatments, which will be discussed shortly.

In general, eq.(5.2) represents a system of  $(d \times K)$  coupled linear equations, where  $d$  is the number of dimensions. There are three variations to simplify eq.(5.2).

**I. De-coupled.** Phasic momentum equations are completely de-coupled. Thus, the following equation

$$\underbrace{\left( a_{c,c}^{(k)} + \sum_{m \neq k}^K c_c^{(k,m)} \right)}_{\tilde{a}_{c,c}^{(k)}} \hat{\mathbf{v}}_{k(c)}^{**} + \underbrace{\sum_n^N a_{c,n}^{(k)} \hat{\mathbf{v}}_{k(n)}^{**}}_{\text{Coupling to neighbours}} = b_c^{(k)} + \underbrace{\sum_{m \neq k}^K c_c^{(k,m)} \tilde{\mathbf{v}}_{m(c)}^*}_{\tilde{b}_c^{(k)}} \quad (5.4)$$

is solved for  $\hat{\mathbf{v}}_{k(c)}^{**}$ . Phasic velocities  $\tilde{\mathbf{v}}_{m \neq k(c)}$  are frozen at the current Picard iteration state.

**II. Partial Elimination Algorithm (PEA).** It is generally recognized that phasic momentum decoupling is problematic for convergence of Picard iterations. To speed-up convergence, Spalding [Spa83] introduced a better (local) coupling of phases, using the following simple algebraic manipulation for 2-fluid configuration.

Re-writing eq.(5.4) for fluid-1 as:

$$\tilde{\mathbf{v}}_{1(c)}^\diamond = \frac{b_c^{(1)} + c_c^{(1,0)} \hat{\mathbf{v}}_{0(c)}^{**} - \sum_n^N a_{c,n}^{(1)} \tilde{\mathbf{v}}_{1(n)}^*}{a_{c,c}^{(1)} + c_c^{(1,0)}} \quad (5.5)$$

and then plugging  $\tilde{\mathbf{v}}_{1(c)}^\diamond$  into eq.(5.4) written for  $k = 0$ , in place of  $\tilde{\mathbf{v}}_{1(c)}^*$ , and finally

re-arranging:

**Momentum equation for fluid-0:**

$$\begin{aligned}
 & \underbrace{\left( a_{c,c}^{(0)} + c_c^{(0,1)} \left( 1 - \frac{c_c^{(1,0)}}{a_{c,c}^{(1)} + c_c^{(1,0)}} \right) \right)}_{\tilde{a}_{c,c}^{(0)}} \hat{\mathbf{v}}_{0(c)}^{**} + \sum_n^N a_{c,n}^{(0)} \hat{\mathbf{v}}_{0(n)}^{**} = \\
 & \underbrace{b_c^{(0)} + c_c^{(0,1)} \frac{b_c^{(1)} - \sum_n^N a_{c,n}^{(1)} \tilde{\mathbf{v}}_{1(n)}^*}{a_{c,c}^{(1)} + c_c^{(1,0)}}}_{\tilde{b}_c^{(0)}}
 \end{aligned} \tag{5.6}$$

Exactly the same manipulations for fluid-1 leads to the following equation:

**Momentum equation for fluid-1:**

$$\begin{aligned}
 & \underbrace{\left( a_{c,c}^{(1)} + c_c^{(1,0)} \left( 1 - \frac{c_c^{(0,1)}}{a_{c,c}^{(0)} + c_c^{(0,1)}} \right) \right)}_{\tilde{a}_{c,c}^{(1)}} \hat{\mathbf{v}}_{1(c)}^{**} + \sum_n^N a_{c,n}^{(1)} \hat{\mathbf{v}}_{1(n)}^{**} = \\
 & \underbrace{b_c^{(1)} + c_c^{(1,0)} \frac{b_c^{(0)} - \sum_n^N a_{c,n}^{(0)} \tilde{\mathbf{v}}_{0(n)}^*}{a_{c,c}^{(0)} + c_c^{(0,1)}}}_{\tilde{b}_c^{(1)}}
 \end{aligned} \tag{5.7}$$

Equations (5.6) and (5.7) are improvement relative to eq.(5.4) because both phases in the interfacial momentum exchange terms are locally coupled (“implicit”). Spatial variations on “the other phase” velocities are treated in a Gauss-Seidel manner.

**III. Simultaneous solution of non-linearly coupled equations (SINCE).** Even better coupling of interfacial momentum exchange terms is provided by SINCE algorithm [Lo90]. SINCE is an improvement relative to PEA not only because spatial variations on “the other phase” velocities are accounted for (thereby providing a better phasic momentum coupling), but also because it is easier to implement for  $K > 2$ .

The algorithm is implemented in a predictor-corrector manner. At the first (predictor) step, equations similar to eq.(5.5):

$$\tilde{\mathbf{v}}_{k(c)}^\diamond = \frac{b_c^{(k)} + \sum_{m \neq k}^K c_c^{(k,m)} \tilde{\mathbf{v}}_{m(c)}^* - \sum_n^N a_{c,n}^{(k)} \tilde{\mathbf{v}}_{k(n)}^*}{a_{c,c}^{(k)} + \sum_{m \neq k}^K c_c^{(k,m)}} \quad (5.8)$$

are solved at each computational cell. These are local equations, treating spatial variations “explicitly”.

At the second (corrector) step, equations similar to eq.(5.6) and (5.7) are solved:

$$\underbrace{\left( a_{c,c}^{(k)} + \sum_{m \neq k}^K \left[ c_c^{(k,m)} \left( 1 - \frac{c_c^{(m,k)}}{a_{c,c}^{(m)} + \sum_{j \neq m}^K c_c^{(m,j)}} \right) \right] \right)}_{\tilde{a}_{c,c}^{(k)}} \hat{\mathbf{v}}_{k(c)}^{***} + \sum_n^N a_{c,n}^{(k)} \hat{\mathbf{v}}_{k(n)}^{***} = \underbrace{b_c^{(k)} + \sum_{m \neq k}^K \left[ c_c^{(k,m)} \frac{b_c^{(m)} + \sum_{j \neq m,k}^K c_c^{(m,j)} \tilde{\mathbf{v}}_{j(c)}^\diamond - \sum_n^N a_{c,n}^{(m)} \tilde{\mathbf{v}}_{m(n)}^\diamond}{a_{c,c}^{(m)} + \sum_{j \neq m}^K c_c^{(m,j)}} \right]}_{\tilde{b}_c^{(k)}} \quad (5.9)$$

### Step B: Pressure-correction Equation

At this step, “predictor” phasic velocity fields  $\hat{\mathbf{v}}_k^{***}$  are available from Step A. These velocity fields generally satisfy momentum balance equations, but not mass conservation. Mass conservation is restored by forming and solving pressure-correction equation, using the following discrete form of the *global mass conservation equation*:

$$\sum_k \left[ \frac{\alpha_k^* \bar{\rho}_k^{***} - \alpha_k^n \bar{\rho}_k^n}{\rho_k^{(\text{ref})} \Delta t} + \nabla_{\rho_k} \cdot \left( \frac{\alpha_k^*}{\rho_k^{(\text{ref})}} \bar{\rho}_k^{***} \tilde{\mathbf{v}}_k^{***} \right) - \frac{\mathcal{S}_{\text{mass},k}(\mathbf{U}^*)}{\rho_k^{(\text{ref})}} \right] = 0 \quad (5.10)$$

## 5.1. SIMPLE-BASED ALGORITHMS

91

where  $(\nabla_{\rho_k} \cdot)$  is discrete divergence operator, which is space-discretization-scheme-dependent.

The new (corrected) pressure, velocity and density fields can be written as

$$\begin{aligned}\bar{p}^{**} &= \bar{p}^* + \bar{p}' \\ \bar{\rho}_k^{**} &= \bar{\rho}_k^* + \bar{\rho}'_k \\ \tilde{\mathbf{v}}_k^{**} &= \hat{\tilde{\mathbf{v}}}_k^{**} + \tilde{\mathbf{v}}'_k\end{aligned}\quad (5.11)$$

To cook pressure-correction Helmholtz equation, let's first consider momentum conservation equations, re-writing eq.(5.2) as

$$\bar{a}_{c,c}^{(k)} \hat{\tilde{\mathbf{v}}}_{k(c)}^{**} + \sum_n^N \bar{a}_{c,n}^{(k)} \hat{\tilde{\mathbf{v}}}_{k(n)}^{**} = \bar{b}_c^{(k)} + \alpha_k^* \nabla_{\mathbf{v}_k} \bar{p}^* \quad (5.12)$$

where we absorbed interfacial momentum exchange terms into  $\bar{a}_{c,c}^{(k)}$  and  $\bar{b}_c^{(k)}$ , extracted pressure gradient from  $b_c^{(k)}$ . Similar to all previous discussions, we denote by  $(\nabla_{\mathbf{v}_k})$  discrete gradient operator, which is space-discretization-scheme-dependent.

The corrected-velocity/pressure momentum equation is

$$\bar{a}_{c,c}^{(k)} \tilde{\mathbf{v}}_{k(c)}^{**} + \sum_n^N \bar{a}_{c,n}^{(k)} \tilde{\mathbf{v}}_{k(n)}^{**} = \bar{b}_c^{(k)} + \alpha_k^* \nabla_{\mathbf{v}_k} \bar{p}^{**} \quad (5.13)$$

where we kept frozen all coefficients  $\bar{a}_{c,c}^{(k)}$ ,  $\bar{a}_{c,n}^{(k)}$  and  $\bar{b}_c^{(k)}$  – the same as in eq.(5.12)<sup>2</sup>.

Subtracting eq.(5.12) from eq.(5.13) and re-arranging using eq.(5.11), yields the following velocity-correction equations:

$$\tilde{\mathbf{v}}'_{k(c)} = \frac{\alpha_k^*}{\bar{a}_{c,c}^{(k)}} \nabla_{\mathbf{v}_k} \bar{p}' - \sum_n^N \frac{\bar{a}_{c,n}^{(k)}}{\bar{a}_{c,c}^{(k)}} \tilde{\mathbf{v}}'_{k(n)} \quad (5.14)$$

The next step will be plugging eq.(5.11) into eq.(5.10) and linearizing non-linear advection term as:

$$\bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**} = (\bar{\rho}_k^* + \bar{\rho}'_k) (\hat{\tilde{\mathbf{v}}}_k^{**} + \tilde{\mathbf{v}}'_k) \rightarrow \bar{\rho}_k^* (\hat{\tilde{\mathbf{v}}}_k^{**} + \tilde{\mathbf{v}}'_k) + \bar{\rho}'_k \hat{\tilde{\mathbf{v}}}_k^{**} + \bar{\rho}'_k \tilde{\mathbf{v}}'_k \rightarrow^0 \quad (5.15)$$

<sup>2</sup>Upon convergence of Picard iterations, this assumption will not be matter.

The resulting equation is

$$\sum_k \left[ \frac{\alpha_k^* (\bar{\rho}_k^* + \bar{p}') - \alpha_k^n \bar{\rho}_k^n}{\rho_k^{(\text{ref})} \Delta t} + \nabla_{\rho_k} \cdot \left( \frac{\alpha_k^*}{\rho_k^{(\text{ref})}} \left[ \bar{\rho}_k^* (\hat{\mathbf{v}}_k^{**} + \tilde{\mathbf{v}}_k') + \bar{p}' \hat{\mathbf{v}}_k^{**} \right] \right) \right] = \sum_k \frac{S_{\text{mass},k}(\mathbf{U}^*)}{\rho_k^{(\text{ref})}} \quad (5.16)$$

Using the following *á-la* ICE linearization of equation of state<sup>3</sup>:

$$\bar{\rho}'_k = \bar{p}' \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \Big|_{\tilde{u}_k = \text{const}}^* + \cancel{\tilde{u}'_k \frac{\partial \bar{\rho}_k}{\partial \tilde{u}_k} \Big|_{\bar{p} = \text{const}}^*} \quad (5.17)$$

and eq.(5.14) for  $\tilde{\mathbf{v}}'_{k(c)}$ , the following *pressure-correction* variable-coefficient Helmholtz equation is formulated:

$$\sum_k \left[ \frac{\alpha_k^* \left( \bar{\rho}_k^* + \bar{p}' \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \Big|_{\tilde{u}_k = \text{const}}^* \right) - \alpha_k^n \bar{\rho}_k^n}{\rho_k^{(\text{ref})} \Delta t} + \nabla_{\rho_k} \cdot \left( \frac{\alpha_k^*}{\rho_k^{(\text{ref})}} \left[ \bar{\rho}_k^* \left( \hat{\mathbf{v}}_k^{**} + \left( \frac{\alpha_k^*}{\bar{a}_{c,c}^{(k)}} \nabla_{\mathbf{v}_k} \bar{p}' - \sum_n^N \bar{d}_{c,n}^{(k)} \tilde{\mathbf{v}}'_{k(n)} \right) \right) + \hat{\mathbf{v}}_k^{**} \bar{p}' \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \Big|_{\tilde{u}_k = \text{const}}^* \right] \right) \right] = \sum_k \frac{S_{\text{mass},k}(\mathbf{U}^*)}{\rho_k^{(\text{ref})}} \quad (5.18)$$

This parabolic equation must be solved for  $\bar{p}'$  with some direct or iterative algorithm.

A number of variations of SIMPLE algorithm are available (SIMPLER, SIMPLEST, PISO, SIMPLEX), all mainly different in how spatial variation of velocity correction  $\sum_n^N \bar{d}_{c,n}^{(k)} \tilde{\mathbf{v}}'_{k(n)}$  is accounted for [DMS01]. These variations are mainly important on how fast Picard iterations converge, but this term is unimportant for fully-converged state (velocity corrections are zero). In the original SIMPLE [PS72, Pat80], these spatial variations are neglected,

$$\sum_n^N \bar{d}_{c,n}^{(k)} \tilde{\mathbf{v}}'_{k(n)} = 0 \quad (5.19)$$

<sup>3</sup>Specific internal energy correction is neglected because the energy equation is decoupled. This is generally not a good idea for reactor applications, where heat transfer is very important for boiling flows.

## 5.1. SIMPLE-BASED ALGORITHMS

93

turning eq.(5.18) into

$$\sum_k \left[ \frac{\beta_k}{\rho_k^{(\text{ref})} \Delta t} \bar{p}' + \nabla_{\rho_k} \cdot \left( \hat{\mathbf{v}}_k^{**} \frac{\beta_k}{\rho_k^{(\text{ref})}} \bar{p}' + \frac{\gamma_k}{\rho_k^{(\text{ref})}} \nabla_{\mathbf{v}_k} \bar{p}' \right) \right] = \sum_k \frac{\sigma_k}{\rho_k^{(\text{ref})}} \quad (5.20)$$

where

$$\begin{aligned} \beta_k &= \alpha_k^* \left. \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \right|_{\tilde{u}_k = \text{const}}^* \\ \gamma_k &= \bar{\rho}_k^* \frac{(\alpha_k^*)^2}{a_{c,c}^{(k)}} \\ \sigma_k &= \mathcal{S}_{\text{mass},k}(\mathbf{U}^*) - \frac{\alpha_k^* \bar{\rho}_k^* - \alpha_k^n \bar{\rho}_k^n}{\Delta t} - \nabla_{\rho_k} \cdot \left( \alpha_k^* \bar{\rho}_k^* \hat{\mathbf{v}}_k^{**} \right) \end{aligned} \quad (5.21)$$

We will refer for specific details of SIMPLE algorithm variations to [DMS01]. Some of them (like PISO) embed additional predictor-corrector steps within the discussed here Steps A and B.

### Step C: Pressure, velocity and density correction

After  $\bar{p}'$  field is determined, velocity correction is computed with eq.(5.14), density correction is computed by eq.(5.17), and new values  $\bar{p}^{**}$ ,  $\bar{\rho}_k^{**}$  and  $\tilde{\mathbf{v}}_k^{**}$  are obtained using eq.(5.11).

### Step D: Void fraction update

Through steps A to C, phasic void fractions are kept at previous Picard iteration value  $\alpha_k^*$ . New values  $\alpha_k^{**}$  are obtained at step D, using combination of the compatibility equation and implicit updates of discretized phasic mass conservation equations:

$$\frac{\alpha_k^{**} \bar{\rho}_k^{**} - \alpha_k^n \bar{\rho}_k^n}{\Delta t} + \nabla_{\rho_k} \cdot \left( \alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**} \right) - \mathcal{S}_{\text{mass},k}(\mathbf{U}^{**}) = 0 \quad (5.22)$$

Note that new values of  $\bar{\rho}_k^{**}$  are  $\tilde{\mathbf{v}}_k^{**}$  are available from Step C. By  $\mathcal{S}_{\text{mass},k}(\mathbf{U}^{**})$  we indicate that mass source terms are computed using a combination of new update values ( $\bar{\rho}_k^{**}$ ,  $\tilde{\mathbf{v}}_k^{**}$  and  $\bar{p}^{**}$  are already known) and previous Picard iteration values (specific energies are not updated yet,  $\tilde{u}_k^*$ ). If mass source terms contain void fraction, they will use  $\alpha_k^{**}$ , thereby being incorporated in coefficients  $a_{c,c}^{(k)}$  of implicit linear equation for void fraction:

$$a_{c,c}^{(k)} \alpha_{k(c)}^{**} + \sum_n^N a_{c,n}^{(k)} \alpha_{k(n)}^{**} = b_c^{(k)} \quad (5.23)$$

The coefficients/source  $a_{c,c}^{(k)}$ ,  $a_{c,n}^{(k)}$  and  $b_c^{(k)}$  are obviously different from those for linearized momentum discrete equations (5.2).

There are three variations for update in Step D.

**I. (N-1)+compatibility.** Out of  $N$  eqs.(5.23),  $(N - 1)$  are solved implicitly for  $(N - 1)$  void fractions  $\alpha_k^{**}$ . The remaining void fraction  $\alpha_z^{**}$  for a singled-out phase ( $z$ ) is computed from the compatibility equation,

$$\alpha_z^{**} = 1 - \sum_{k \neq z}^K \alpha_k^{**} \quad (5.24)$$

The choice of ( $z$ ) will depend on the problem. Typically, one would chose the stiffest/heaviest phase, to reduce overall mass conservation errors.

The fact that void fractions are updated de-coupled from each other (except for phase ( $z$ )) adversely affect convergence of Picard iterations.

**II. Compatibility-based re-balancing.** Slightly better option is to update all  $N$  phasic void fractions solving eqs.(5.23) for  $\tilde{\alpha}_k^{**}$ , and then enforce compatibility by re-balancing as:

$$\alpha_k^{**} = \frac{\tilde{\alpha}_k^{**}}{\sum_m^K \tilde{\alpha}_m^{**}} \quad (5.25)$$

**III. Implicit compatibility-based re-balancing.** As discussed in [DMS01], void fraction rebalancing can be incorporated into implicit solves, by modifying eqs.(5.23) as

$$a_{c,c}^{(k)} \left( 1 - \sum_m^K \frac{res_\alpha^{(m)}}{a_{c,c}^{(k)}} \right) \alpha_{k(c)}^{**} + \sum_n^N a_{c,n}^{(k)} \alpha_{k(n)}^{**} = b_c^{(k)} \quad (5.26)$$

where residuals are defined as

$$res_\alpha^{(m)} = a_{c,c}^{(m)} \alpha_{m(c)}^{**} + \sum_n^N a_{c,n}^{(m)} \alpha_{m(n)}^{**} - b_c^{(m)} \quad (5.27)$$

In this case, the equations are coupled through the diagonal terms. It is however not clear how exactly the iterative process for this coupled system of  $N$  equations

could be efficiently organized.

**IV. Bounding with under-relaxation.** The last variation we would like to discuss addresses the bounding of phasic void fraction in the physically meaningful range  $[0, 1]$ .  $\alpha_k^{**}$  can go out of bounds during iterative process. This is acceptable if the final converged solution is within a bound, unless the out-of-bound values crash the iterative process. To keep  $\alpha_k^{**}$  bounded, Carver [Car82] introduced procedure with under-relaxations, modifying eq.(5.22) as:

$$\frac{a_{c,c}^{(k)}}{\beta_{(c)}} \alpha_{k(c)}^{**} + \sum_n^N a_{c,n}^{(k)} \alpha_{k(n)}^{**} = b_c^{(k)} + \frac{1 - \beta_{(c)}}{\beta_{(c)}} a_{c,c}^{(k)} \alpha_{k(c)}^* \quad (5.28)$$

where under-relaxation parameter  $\beta_{(c)} \in [0, 1]$ . Carver introduced a procedure to assign relaxation parameter to each computational cell [Car82]. This deviates from the commonly used practice to have a single constant relaxation for the whole computational domain, [Pat80]. Also, it is instructive to note that any under-relaxation tends to negatively impact convergence of Picard iterations.

### Steps E,F: Energy and the rest

The final steps of outer Picard iteration are rather straightforward. Each remaining equation is discretized and the following linear equations are solved implicitly:

$$a_{c,c}^{(k)} (\mathbf{U}^{*,**}) \Phi_{k(c)}^{**} + \sum_n^N a_{c,n}^{(k)} (\mathbf{U}^{*,**}) \Phi_{k(n)}^{**} = b_c^{(k)} (\mathbf{U}^{*,**}) \quad (5.29)$$

where  $\Phi = \tilde{u}_k, \dots$ , while coefficients  $a_{i,j}^{(k)}$  and sources  $b_c^{(k)}$  are computed using currently available Picard iteration values of the solution vector.

### **Remarks**

1. Step A of the MCBA is somewhat similar to the “stabilizer” step for SETS and “nearly-implicit” algorithms, Sections 4.2.1 and 4.2.2.
2. It is instructive to note that there is no guarantee that Picard iterations would always converge to a physical solution. In fact, non-linear problem might have multiple solutions, and “saddle” regions (with local extrema). Multiple solutions are possible when the problem is ill-posed, or when hyperbolic

part of the governing equations (i.e., the one which contain first derivatives on time and space, thereby defining trajectory of the system and wave structure) is *non-hyperbolic* (i.e., the eigenvalues are complex). It is not unconceivable that numerical solution might tend to be attracted to unphysical solutions, like those with out-of-bound void fractions.

3. As noted in the discussions above, the fact that energy equation is completely de-coupled from mass and momentum in all SIMPLE-based algorithms is very problematic for reactor applications. There was a reason why all legacy reactor thermalhydraulics codes pay attention to this coupling. However, one can easily incorporate coupling with energy, modifying iterations along the lines of “semi-implicit” algorithm, Section 4.1. We are not aware however of any such development.

### 5.1.2 Geometric Conservation-Based Algorithms (GCBA)

The GCBA's utilize compatibility equation to derive pressure equation. The IPSA algorithm developed by Spalding [Spa76, Spa80, Spa81, Spa83] is an example of GCBA. As discussed by Darwish et al. [DMS01], The GCBA's introduce a stronger coupling between the pressure and the volume fractions, than the discussed above MCBAs.

The outer iteration loop of GCBA's is organized as following:

---

- A. Solve (implicitly) phasic mass conservation equations for volume fractions. No compatibility constraints are applied.
- B. Solve implicitly phasic momentum equations. No mass mass conservation constraints are applied.
- C. Form and solve (implicit) pressure-correction equation, based on compatibility equation.
- D. Correct phasic velocities, densities, void fractions and pressure.
- E. Solve energy equations.
- F. Solve turbulent transport equations (if any).

G. Check convergence.

### Step A: Void fraction update

Phasic mass conservation equations are discretized as

$$\frac{\hat{\alpha}_k^{**} \bar{\rho}_k^* - \alpha_k^n \bar{\rho}_k^n}{\Delta t} + \nabla_{\rho_k} \cdot (\hat{\alpha}_k^{**} \bar{\rho}_k^* \tilde{\mathbf{v}}_k^*) - \mathcal{S}_{\text{mass},k}(\mathbf{U}^*) = 0 \quad (5.30)$$

rendering  $N$  linear equations of type:

$$a_{c,c}^{(k)}(\mathbf{U}^*) \hat{\alpha}_{k(c)}^{**} + \sum_n^N a_{c,n}^{(k)}(\mathbf{U}^*) \hat{\alpha}_{k(n)}^{**} = b_c^{(k)}(\mathbf{U}^*) \quad (5.31)$$

which are solved implicitly for  $\hat{\alpha}_{k(c)}^{**}$ . Until Picard iteration convergence is achieved, these void fractions do not generally satisfy compatibility equation, i.e.

$$res_\alpha = 1 - \sum_k \hat{\alpha}_{k(c)}^{**} \neq 0 \quad (5.32)$$

and they will be corrected at the Step D.

### Step B: Velocity predictor

This step is essentially the same as Step A in MCBA, except that all linear algebra coefficients are evaluated with known from Step A values of  $\hat{\alpha}_{k(c)}^{**}$ .

### Step C: Pressure-correction Equation

The new (corrected) pressure, velocity, void fraction and density fields can be written as

$$\begin{aligned} \bar{p}^{**} &= \bar{p}^* + \bar{p}' \\ \bar{\rho}_k^{**} &= \bar{\rho}_k^* + \bar{\rho}'_k \\ \alpha_k^{**} &= \hat{\alpha}_k^{**} + \alpha'_k \\ \tilde{\mathbf{v}}_k^{**} &= \tilde{\mathbf{v}}_k^* + \tilde{\mathbf{v}}'_k \end{aligned} \quad (5.33)$$

The phasic mass conservation equations at this step can now be discretized as

$$\frac{\alpha_k^{**} \bar{\rho}_k^{**} - \alpha_k^n \bar{\rho}_k^n}{\Delta t} + \nabla_{\rho_k} \cdot (\alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**}) = \mathcal{S}_{\text{mass},k}(\mathbf{U}^{*,**}) \quad (5.34)$$

Next, plugging eq.(5.33) into eq.(5.34) and linearizing non-linear terms as

$$\begin{aligned}
 \alpha_k^{**} \bar{\rho}_k^{**} &= (\bar{\rho}_k^* + \bar{\rho}'_k) (\hat{\alpha}_k^{**} + \alpha'_k) \rightarrow \bar{\rho}_k^* \hat{\alpha}_k^{**} + \bar{\rho}_k^* \alpha'_k + \bar{\rho}'_k \hat{\alpha}_k^{**} + \cancel{\bar{\rho}'_k \alpha'_k} \\
 \alpha_k^{**} \bar{\rho}_k^{**} \tilde{\mathbf{v}}_k^{**} &= (\bar{\rho}_k^* + \bar{\rho}'_k) (\hat{\alpha}_k^{**} + \alpha'_k) (\hat{\mathbf{v}}_k^{**} + \tilde{\mathbf{v}}'_k) \rightarrow \\
 &\rightarrow \bar{\rho}_k^* \hat{\alpha}_k^{**} \hat{\mathbf{v}}_k^{**} + \bar{\rho}_k^* \alpha'_k \hat{\mathbf{v}}_k^{**} + \bar{\rho}'_k \hat{\alpha}_k^{**} \hat{\mathbf{v}}_k^{**} + \cancel{\bar{\rho}'_k \alpha'_k \hat{\mathbf{v}}_k^{**}} + \\
 &\quad + \bar{\rho}_k^* \hat{\alpha}_k^{**} \tilde{\mathbf{v}}'_k + \cancel{\bar{\rho}_k^* \alpha'_k \tilde{\mathbf{v}}'_k} + \cancel{\bar{\rho}'_k \hat{\alpha}_k^{**} \tilde{\mathbf{v}}'_k} + \cancel{\bar{\rho}'_k \alpha'_k \tilde{\mathbf{v}}'_k}
 \end{aligned} \tag{5.35}$$

yields:

$$\begin{aligned}
 &\frac{\bar{\rho}_k^* \hat{\alpha}_k^{**} + \bar{\rho}_k^* \alpha'_k + \bar{\rho}'_k \hat{\alpha}_k^{**} - \alpha_k^n \bar{\rho}_k^n}{\Delta t} + \\
 &\quad + \nabla_{\rho_k} \cdot \left( \bar{\rho}_k^* \hat{\alpha}_k^{**} \hat{\mathbf{v}}_k^{**} + \bar{\rho}_k^* \alpha'_k \hat{\mathbf{v}}_k^{**} + \bar{\rho}'_k \hat{\alpha}_k^{**} \hat{\mathbf{v}}_k^{**} + \bar{\rho}_k^* \hat{\alpha}_k^{**} \tilde{\mathbf{v}}'_k \right) = \\
 &\quad = \mathcal{S}_{\text{mass},k} \left( \mathbf{U}^{*,**} \right)
 \end{aligned} \tag{5.36}$$

Finally, plugging velocity-correction eq.(5.14) for  $\tilde{\mathbf{v}}'_{k(c)}$  and linearization of equations of state eq.(5.17) for  $\bar{\rho}'_k$  into eq.(5.36) results in  $N$  coupled equations for  $(N + 1)$  unknowns ( $\alpha'_{k=0, \dots, N-1}$  and  $\bar{p}'$ ):

$$\begin{aligned}
 &\frac{\bar{\rho}_k^* \hat{\alpha}_k^{**} + \bar{\rho}_k^* \alpha'_k + \bar{p}' \left. \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \right|_{\tilde{u}_k = \text{const}}^*}{\Delta t} + \frac{\hat{\alpha}_k^{**} - \alpha_k^n \bar{\rho}_k^n}{\Delta t} + \\
 &\quad + \nabla_{\rho_k} \cdot \left( \bar{\rho}_k^* \hat{\alpha}_k^{**} \hat{\mathbf{v}}_k^{**} + \bar{\rho}_k^* \alpha'_k \hat{\mathbf{v}}_k^{**} + \bar{p}' \left. \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \right|_{\tilde{u}_k = \text{const}}^* \hat{\alpha}_k^{**} \hat{\mathbf{v}}_k^{**} + \right. \\
 &\quad \left. + \bar{\rho}_k^* \hat{\alpha}_k^{**} \frac{\alpha_k^*}{\bar{a}_{c,c}^{(k)}} \nabla_{\mathbf{v}_k} \bar{p}' - \sum_n^N \bar{d}_{c,n}^{(k)} \tilde{\mathbf{v}}'_{k(n)} \right) = \mathcal{S}_{\text{mass},k} \left( \mathbf{U}^{*,**} \right)
 \end{aligned} \tag{5.37}$$

To simplify further presentation, we collect terms and re-group eq.(5.37) as:

$$\begin{aligned}
 &\frac{\bar{\rho}_k^* \alpha'_k + \beta_k \bar{p}'}{\Delta t} + \nabla_{\rho_k} \cdot \left( \hat{\mathbf{v}}_k^{**} (\bar{\rho}_k^* \alpha'_k + \beta_k \bar{p}') + \gamma_k \nabla_{\mathbf{v}_k} \bar{p}' \right) = \\
 &\quad = \sigma_k + \nabla_{\rho_k} \cdot \left( \sum_n^N \bar{d}_{c,n}^{(k)} \tilde{\mathbf{v}}'_{k(n)} \right)
 \end{aligned} \tag{5.38}$$

where

$$\begin{aligned}
 \beta_k &= \hat{\alpha}_k^{**} \left. \frac{\partial \bar{\rho}_k}{\partial \bar{p}} \right|_{\tilde{u}_k = \text{const}}^* \\
 \gamma_k &= \bar{\rho}_k^* \hat{\alpha}_k^{**} \frac{\alpha_k^*}{\bar{a}_{c,c}^{(k)}} \\
 \sigma_k &= \mathcal{S}_{\text{mass},k} \left( \mathbf{U}^{*,**} \right) - \frac{\hat{\alpha}_k^{**} \bar{\rho}_k^* - \alpha_k^n \bar{\rho}_k^n}{\Delta t} - \nabla_{\rho_k} \cdot \left( \hat{\alpha}_k^{**} \bar{\rho}_k^* \hat{\mathbf{v}}_k^{**} \right)
 \end{aligned} \tag{5.39}$$

## 5.1. SIMPLE-BASED ALGORITHMS

99

As our next step, we discretize  $\alpha'_k$ -part of eq.(5.38), i.e. replacing

$$\frac{\bar{\rho}_k^* \alpha'_k}{\Delta t} + \nabla_{\rho_k} \cdot \left( \hat{\mathbf{v}}_k^{**} \bar{\rho}_k^* \alpha'_k \right) \rightarrow \hat{a}_{c,c}^{(k)} \alpha'_{k(c)} + \sum_n^N \hat{a}_{c,n}^{(k)} \alpha'_{k(n)} \quad (5.40)$$

to get

$$\begin{aligned} \hat{a}_{c,c}^{(k)} \alpha'_{k(c)} + \frac{\beta_k}{\Delta t} \bar{p}' + \nabla_{\rho_k} \cdot \left( \hat{\mathbf{v}}_k^{**} \beta_k \bar{p}' + \gamma_k \nabla_{\mathbf{v}_k} \bar{p}' \right) = \\ = \sigma_k + \nabla_{\rho_k} \cdot \left( \sum_n^N \frac{\hat{d}_{c,n}^{(k)}}{\hat{a}_{c,c}^{(k)}} \tilde{\mathbf{v}}'_{k(n)} \right) - \sum_n^N \frac{\hat{a}_{c,n}^{(k)}}{\hat{a}_{c,c}^{(k)}} \alpha'_{k(n)} \end{aligned} \quad (5.41)$$

Solving for  $\alpha'_{k(c)}$  :

$$\begin{aligned} \alpha'_{k(c)} = \frac{\sigma_k}{\hat{a}_{c,c}^{(k)}} + \nabla_{\rho_k} \cdot \left( \sum_n^N \frac{\hat{d}_{c,n}^{(k)}}{\hat{a}_{c,c}^{(k)}} \tilde{\mathbf{v}}'_{k(n)} \right) - \sum_n^N \frac{\hat{a}_{c,n}^{(k)}}{\hat{a}_{c,c}^{(k)}} \alpha'_{k(n)} - \\ - \frac{\beta_k}{\hat{a}_{c,c}^{(k)} \Delta t} \bar{p}' - \nabla_{\rho_k} \cdot \left( \hat{\mathbf{v}}_k^{**} \frac{\beta_k}{\hat{a}_{c,c}^{(k)}} \bar{p}' + \frac{\gamma_k}{\hat{a}_{c,c}^{(k)}} \nabla_{\mathbf{v}_k} \bar{p}' \right) \end{aligned} \quad (5.42)$$

These are  $K$  equations. To eliminate  $\alpha'_{k(c)}$ , we can use compatibility equation:

$$\sum_k^K \underbrace{\left( \alpha_{k(c)}^* + \alpha'_{k(c)} \right)}_{\alpha_{k(c)}^{**}} = 1 \quad (5.43)$$

Thus, adding  $\alpha_{k(c)}^*$  to the left and right hand sides of eqs.(5.42), summing over all  $k$ , and equating to 1, the following *Pressure-correction* Helmholtz equation is obtained:

$$\begin{aligned} \sum_k^K \left[ \alpha_{k(c)}^* + \frac{\sigma_k}{\hat{a}_{c,c}^{(k)}} + \nabla_{\rho_k} \cdot \left( \sum_n^N \frac{\hat{d}_{c,n}^{(k)}}{\hat{a}_{c,c}^{(k)}} \tilde{\mathbf{v}}'_{k(n)} \right) - \sum_n^N \frac{\hat{a}_{c,n}^{(k)}}{\hat{a}_{c,c}^{(k)}} \alpha'_{k(n)} - \right. \\ \left. - \frac{\beta_k}{\hat{a}_{c,c}^{(k)} \Delta t} \bar{p}' - \nabla_{\rho_k} \cdot \left( \hat{\mathbf{v}}_k^{**} \frac{\beta_k}{\hat{a}_{c,c}^{(k)}} \bar{p}' + \frac{\gamma_k}{\hat{a}_{c,c}^{(k)}} \nabla_{\mathbf{v}_k} \bar{p}' \right) \right] = 1 \end{aligned} \quad (5.44)$$

Spatial variations of void fraction correction are neglected. This assumption does not affect the final (Picard-converged) solution, as  $\alpha'_k$  must  $\rightarrow 0$ . Different

SIMPLE algorithm implementations allow to account for spatial variations of velocity corrections (see [DMS01] for SIMPLEC, PRIME, SIMPLER, SIMPLEM algorithms). In the original SIMPLE,  $\sum_n \frac{\tilde{d}_{c,n}^{(k)}}{\hat{a}_{c,c}^{(k)}} \tilde{\mathbf{v}}_{k(n)}'$  is neglected, yielding

$$\begin{aligned} \sum_k^K \left[ \frac{\beta_k}{\hat{a}_{c,c}^{(k)} \Delta t} \bar{p}' + \nabla_{\rho_k} \cdot \left( \hat{\mathbf{v}}_k^{**} \frac{\beta_k}{\hat{a}_{c,c}^{(k)}} \bar{p}' + \frac{\gamma_k}{\hat{a}_{c,c}^{(k)}} \nabla_{\mathbf{v}_k} \bar{p}' \right) \right] = \\ = \sum_k^K \left[ \alpha_{k(c)}^* + \frac{\sigma_k}{\hat{a}_{c,c}^{(k)}} \right] - 1 \end{aligned} \quad (5.45)$$

#### **Step D: Pressure, velocity, void fraction and density correction**

Once  $\bar{p}'$  field is computed, velocity correction is computed with eq.(5.14), density correction is computed by eq.(5.17), void fraction correction is computed by eq.(5.42) and new values  $\bar{p}^{**}$ ,  $\bar{\rho}_k^{**}$ ,  $\alpha_k^{**}$  and  $\tilde{\mathbf{v}}_k^{**}$  are obtained using eq.(5.33).

#### **Steps E,F: Energy and the rest**

The same as steps E,F of MCBA.

#### **Remarks**

1. Careful one-to-one inspection of the “semi-implicit” (Section 4.1) and the GCBA algorithms reveals many similarities. In fact, it is reasonable to state that the “semi-implicit” algorithm corresponds to the first Picard iteration of the SIMPLE-GCBA, except that the “semi-implicit” algorithm misses implicit void fraction update, but couples phasic energy equations together with phasic mass, momentum and compatibility equations. This is in contrast to the SIMPLE-GCBA, which segregates the solution of energy equations. On the other hand, implicit void fraction updates of MCBA are acting like “stabilizer” steps of SETS, Section 4.2.1.
2. Nice feature of the GCBA is that the *ad hoc* algorithms of MCBA to restore compatibility are completely avoided. Rather, phasic compatibility is implicitly embedded into the pressure-Helmholtz correction equation (5.44), providing more tight coupling of void fractions and pressure. Thus, the GCBA appears to be a better (more robust) algorithm.

3. It is easy to see that in the limiting case of single-phase, both pressure-correction equations – eq.(5.20) of MCBA and eq.(5.45) of GCBA collapse into the same *single-fluid pressure-correction Helmholtz equation*:

$$\frac{\beta}{\Delta t} \bar{p}' + \nabla_{\rho} \cdot \left( \hat{\mathbf{v}}_k^{**} \beta \bar{p}' + \gamma \nabla_{\mathbf{v}} \bar{p}' \right) = \sigma \quad (5.46)$$

where

$$\begin{aligned} \beta &= \left. \frac{\partial \bar{p}}{\partial \bar{p}} \right|_{\bar{u}=\text{const}}^* \\ \gamma &= \frac{\bar{\rho}^*}{\bar{a}_{c,c}} \\ \sigma &= \mathcal{S}_{\text{mass}} \left( \mathbf{U}^{*,**} \right) - \frac{\bar{\rho}^* - \bar{\rho}^n}{\Delta t} - \nabla_{\rho} \cdot \left( \bar{\rho}^* \hat{\mathbf{v}}^{**} \right) \end{aligned} \quad (5.47)$$

4. It is possible to convert presentation of “semi-implicit” based algorithms into the pressure-/velocity-/energy-/density-/void-fraction-correction form, making comparison with SIMPLE-based algorithms more transparent.

## 5.2 “Semi-implicit”-based algorithm

As discussed in Chapter 4, it is rather straightforward to extend ICE-based operator-splitting “*semi-implicit*” (Section 4.1), *SETS* (Section 4.2.1) and “*Nearly-Implicit*” algorithms to be fully-implicit of “segregated” type. All what is needed is to replace “old-time” variable values in the “mixed-time” discretization terms by some “iterative” values, and to organize outer iteration loop. As an initial guess of these iterative values, one can use “old-time” values, which makes the first iteration be exactly the same as in the ancestor operator-split algorithm.

This extension make lots of sense when viscous operators (and turbulence models) are added, since parabolic viscous and heat conduction operators better be treated implicitly. Fractional step extensions of “semi-implicit” algorithm are not designed to deal with Fourier-number stability restrictions<sup>4</sup>.

Even though this extension is rather simple, we are not aware of anybody using it. This is perhaps Newton-based fully-implicit algorithms (Chapter 6) are much

<sup>4</sup>In fact, RELAP5-3D [cdt09] treats viscous diffusion operator explicitly, which is extremely restrictive for turbulent flows, when Fourier number restrictions (scaled as  $\sim \Delta h^2$  in contrast to  $\sim \Delta h$  of CFL) become dominant.

more efficient and elegant, and recent advances in linear algebra solvers and CPU memory availability make these algorithms feasible.

### 5.3 NPHASE Algorithm

NPHASE code was developed in RPI. The code is based on segregated pressure-based algorithm [AEKP00, Ant11], which appears to be of MCBA type, with PEA for implicit velocity update Step A. It is stated that the code possesses an enhanced robustness for the modeling of multi- $(N)$ -fluid configurations. However, since not a single comprehensive description of the underlying algorithm is available, with any clear demonstration of the advancement, we would avoid any further speculation and discussion on the code capabilities.

*This Page is Intentionally Left Blank*

## Chapter 6

# Fully-Implicit, Newton-Based Algorithms

THE best known algorithms for solving non-linear problems are based on Newton iterations. Depending on the chosen implicit time discretization, non-linear problems must be solved once or several times (in the case of multi-stage implicit Runge-Kutta algorithms) per time step. Discussion of Newton-based algorithms is organized as following. First, we briefly discuss implicit time discretizations in Section 6.1, followed by highlights of Newton algorithm in Section 6.2. CATHARE's Newton-based solver will be outlined in Section 6.3. JFNK-based algorithm will be presented in Section 6.4, followed by the discussion of preconditioning techniques.

### 6.1 Implicit time discretizations

An implicit time discretization of eqs.(2.46)-(2.48) can be written as

$$\begin{aligned}
 \mathbf{U}^{[k]} &= \mathbf{U}^{[n]} + \Delta t \sum_{r=1}^k \mathbf{a}_{kr} \mathfrak{S} \left( \mathbf{U}^{[r]} \right), \quad k = 1, \dots, s-1 \\
 \mathbf{U}^{[n+1]} &= \alpha \mathbf{U}^{[n-1]} + \beta \mathbf{U}^{[n]} + \Delta t \left( \mathbf{b}_0 \mathfrak{S} \left( \mathbf{U}^{[n]} \right) + \sum_{r=1}^s \mathbf{b}_r \mathfrak{S} \left( \mathbf{U}^{[r]} \right) \right)
 \end{aligned} \tag{6.1}$$

where  $s$  is the total number of implicit Runge-Kutta (IRK) stages, while  $\mathbf{a}_{kr}$  and  $\mathbf{b}_r$  are the stage and the main scheme weights, respectively. Superscripts “ $[\times]$ ” denote the stages of IRK iteration.

## 6.1. IMPLICIT TIME DISCRETIZATIONS

105

As in previous discussions, we avoid any specific reference to the details of space discretization, denoting spatial/source discretization operator by  $\mathfrak{S}$ :

$$\mathfrak{S} = \begin{bmatrix} -\nabla_{\rho_k} \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k) + \mathcal{S}_{\text{mass},k} \\ -\nabla_{\mathbf{v}_k} \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k) + \alpha_k \nabla_{\mathbf{v}_k} \bar{p} + \vec{\mathcal{S}}_{\text{mom},k} \\ -\nabla_{e_k} \cdot (\alpha_k \tilde{\mathbf{v}}_k (\bar{\rho}_k \tilde{e}_k + \bar{p})) + (\tilde{\mathbf{v}}_k \bar{p}) \nabla_{e_k} \alpha_k + \mathcal{S}_{\text{ene},k} \end{bmatrix} \quad (6.2)$$

### 6.1.1 Backward Euler

In the case of  $\alpha = 0$ ,  $\beta = 1$ ,  $s = 1$ ,  $\mathbf{b}_0 = 0$  and  $\mathbf{b}_1 = 1$ , eq.(6.1) reduces to the first-order *Backward Euler* (BE<sub>1</sub>) discretization.

### 6.1.2 BDF

In the case of  $\alpha = -\frac{\Delta t^2}{\Delta t_{n-1}(2\Delta t + \Delta t_{n-1})}$ ,  $\beta = \frac{\Delta t}{\Delta t_{n-1}} \frac{\Delta t + \Delta t_{n-1}}{2\Delta t + \Delta t_{n-1}}$ ,  $s = 1$ ,  $\mathbf{b}_0 = 0$  and  $\mathbf{b}_1 = 1$ , eq.(6.1) reduces to the second-order *Backward Difference* (BDF<sub>2</sub>) discretization.

### 6.1.3 Crank-Nicholson

In the case of  $\alpha = 0$ ,  $\beta = 1$ ,  $s = 1$ ,  $\mathbf{b}_0 = \frac{1}{2}$  and  $\mathbf{b}_1 = \frac{1}{2}$ , eq.(6.1) is the second-order *Crank-Nicholson* (CN<sub>2</sub>) scheme.

### 6.1.4 ESDIRK

A family of high-order IRK schemes recently developed by Carpenter et al. [BCVK02], [CKB<sup>+</sup>05] is particularly useful, since these schemes not only do not amplify any left-half-plane-(LHP)-scaled eigenvalues (*A-stability*), but also provide a complete damping of all eigenvalues including those at the limit  $\|z \rightarrow \infty\|$  (*L-stability*). These IRK schemes are prescribed by  $\mathbf{b}_0 = 0$  and the Butcher tableau of the fol-

lowing form:

$$\begin{array}{c|cccccc}
 0 & 0 & 0 & 0 & 0 & \dots & 0 \\
 c_2 & a_{21} & \gamma & 0 & 0 & \dots & 0 \\
 c_3 & a_{31} & a_{32} & \gamma & 0 & \dots & 0 \\
 \dots & & & & & & \\
 c_{s-1} & a_{(s-1)1} & a_{(s-1)2} & \dots & \dots & \gamma & 0 \\
 1 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \dots & \mathbf{b}_{(s-1)} & \gamma \\
 \hline
 & \mathbf{b}_1 & \mathbf{b}_2 & \mathbf{b}_3 & \dots & \mathbf{b}_{(s-1)} & \gamma \\
 \hline
 & \hat{\mathbf{b}}_1 & \hat{\mathbf{b}}_2 & \hat{\mathbf{b}}_3 & \dots & \hat{\mathbf{b}}_{(s-1)} & \hat{\mathbf{b}}_{(s)}
 \end{array} \quad (6.3)$$

where  $c_r$  denotes the point in time of the  $r^{\text{th}}$ -stage,  $t^{[n]} + c_r \Delta t$ . Note that the first stage is explicit, and the diagonal elements for all stages  $r > 1$  are the same,  $a_{rr} = \gamma$ , which is why this family is called “*Explicit, Singly Diagonal Implicit Runge-Kutta*” (*ESDIRK*) in the literature. Note that the  $p^{\text{th}}$ -order *ESDIRK* <sub>$p$</sub>  schemes allow to compute  $(p - 1)^{\text{th}}$ -order solution, as

$$\mathbf{U}^{[n+1]} = \mathbf{U}^{[n]} + \Delta t \sum_{r=1}^s \hat{\mathbf{b}}_r \mathfrak{S} \left( \mathbf{U}^{[r]} \right) \quad (6.4)$$

## 6.2 Newton method

Each stage of IRK requires solution of non-linear system in the form

$$r \vec{e}_{s_U} \left( \vec{\mathcal{X}}_U \right) = 0 \quad (6.5)$$

or, alternatively

$$r \vec{e}_{s_U} \left( \vec{\mathcal{X}}_V \right) = 0 \quad (6.6)$$

where

$$\vec{\mathcal{X}}_U = \left( \mathbf{U}_1^T, \mathbf{U}_2^T, \dots, \mathbf{U}_{N_{\text{cells}}}^T \right)^T \quad (6.7)$$

and

$$\vec{\mathcal{X}}_V = \left( \mathbf{V}_1^T, \mathbf{V}_2^T, \dots, \mathbf{V}_{N_{\text{cells}}}^T \right)^T \quad (6.8)$$

are solution vectors of *conservative* ( $\mathbf{U}$ ) or *primitive* ( $\mathbf{V}$ ) variables, respectively. These include all variables in all  $N_{\text{cells}}$  computational cells.

The residual vector  $r\vec{e}s_{\mathbf{U}}$  for each variable at the cell  $(c)$  takes the form:

$$r\vec{e}s_{\mathbf{U}(c)}\left(\vec{\mathcal{X}}_{\mathbf{V}}\right) = \mathbf{U}_{(c)}^{[rk]} - \mathbf{U}_c^{[n]} - \Delta t \sum_{r=1}^{rk} a_{rk,r} \mathfrak{S}_c\left(\vec{\mathcal{X}}_{\mathbf{V}}^{[r]}\right) \quad (6.9)$$

It is important to note that even though we define residuals  $r\vec{e}s_{\mathbf{U}}$  in terms of *conservative* variables<sup>1</sup>  $\mathbf{U}$ , one can chose another set of variables for *solution (or primitive) vector*  $\mathbf{V}$ . We already discussed this briefly in Section 2.6 (eq.(2.49) vs. eqs.(2.50) and (2.51)).

Non-linear system of eqs.(6.5) can be solved with Newton's method, iteratively, as a sequence of linear problems defined by

$$\mathbb{J}_{\mathbf{U}}^a \delta \vec{\mathcal{X}}_{\mathbf{U}}^a = -r\vec{e}s_{\mathbf{U}}\left(\vec{\mathcal{X}}_{\mathbf{U}}^a\right) \quad (6.10)$$

The matrix  $\mathbb{J}_{\mathbf{U}}^a$  is the Jacobian of the  $a^{\text{th}}$  Newton's iteration and  $\delta \vec{\mathcal{X}}_{\mathbf{U}}^a$  is the update vector for conservative variables. Each  $(i,j)^{\text{th}}$  element of the Jacobian matrix is a partial derivative of the  $i^{\text{th}}$  equation with respect to the  $j^{\text{th}}$  variable:

$$\mathbb{J}_{\mathbf{U}i,j}^a \equiv \frac{\partial (r\vec{e}s_{\mathbf{U}})_i}{\partial (\vec{\mathcal{X}}_{\mathbf{U}})_j} \quad (6.11)$$

The linear system eq.(6.10) is solved for  $\delta \vec{\mathcal{X}}_{\mathbf{U}}^a$ , and the new Newton's iteration value for  $\vec{\mathcal{X}}_{\mathbf{U}}$  is then computed as

$$\vec{\mathcal{X}}_{\mathbf{U}}^{a+1} = \vec{\mathcal{X}}_{\mathbf{U}}^a + \delta \vec{\mathcal{X}}_{\mathbf{U}}^a \quad (6.12)$$

Newton's iterations on  $\vec{\mathcal{X}}_{\mathbf{U}}$  are continued until the convergence criterion

$$\left\| r\vec{e}s\left(\vec{\mathcal{X}}_{\mathbf{U}}^a\right) \right\|_2 < \text{tol}_N \left\| r\vec{e}s\left(\vec{\mathcal{X}}_{\mathbf{U}}^0\right) \right\|_2 \quad (6.13)$$

<sup>1</sup>In fact, this is preferable, as by doing this one can write conservative-to-roundoff discrete forms.

is satisfied, where the nonlinear tolerance  $\text{tol}_N$  is a given input parameter (typically, in the range  $10^{-6} \div 10^{-8}$ ).

Conservative variables are not always a good choice for solution vector. For example, in compressible low-Mach-number or stiff-fluid applications, density is nearly constant. Thus, small error in density would cause huge error in pressure, which leads to numerical instabilities. Similarly, total energy is dominated by internal energy, and kinetic energy is a only small fraction of total energy,

$$\frac{\tilde{v}^2}{2} \ll \tilde{u}$$

Therefore, Newton iterations would have hard time to distinct these two and to converge. In terms of linear algebra, this implies that Jacobian matrix  $\mathbb{J}_U$  is stiff. In this case, it is better to chose another set of unknowns, transforming the Newton method into the following algorithm:

A. **Start Newton iteration.** Initial guess for  $\vec{X}_V^a$ .

B. Solve linear problem:

$$\underbrace{\mathbb{J}_U^a \frac{\partial \mathbf{U}}{\partial \mathbf{V}}}_{\mathbb{J}_V^a(\vec{X}_V^a)} \delta \vec{X}_V^a = -r\vec{e}_{s_U}(\vec{X}_V^a) \quad (6.14)$$

using direct or iterative method,

$$\delta \vec{X}_V^a = -(\mathbb{J}_V^a)^{-1} r\vec{e}_{s_U}$$

C. Update:

$$\vec{X}_V^{a+1} = \vec{X}_V^a + \delta \vec{X}_V^a \quad (6.15)$$

D. Check convergence:

a. If

$$\left\| r\vec{e}_{s_U}(\vec{X}_V^a) \right\|_2 < \text{tol}_N \left\| r\vec{e}_{s_U}(\vec{X}_V^0) \right\|_2 \quad (6.16)$$

Finished, go to Step (E).

---

**6.2. NEWTON METHOD**
**109**

b. Else, reset  $\vec{\mathcal{X}}_v^a = \vec{\mathcal{X}}_v^{a+1}$  and repeat starting from Step (B).

E. **DONE.**

---

**Remarks**

1. The choice of primitive (solution) variables should lead to well conditioned Jacobian matrix,  $\mathbb{J}_v^a$ . For two-fluid model, there are different options. RELAP5-3D [cdt09] uses:

$$\mathbf{V} = \begin{bmatrix} \bar{p} \\ \alpha \\ \tilde{\mathbf{v}}_0 \\ \tilde{\mathbf{v}}_1 \\ \tilde{u}_0 \\ \tilde{u}_1 \end{bmatrix} \quad (6.17)$$

Another choice was used in the JFNK-based algorithm [NBDY11, NDY10, YNK<sup>+</sup>11] (see Section 6.4):

$$\mathbf{V} = \begin{bmatrix} \bar{p} \\ \alpha \\ \mathbf{v}_{\text{mix}} \equiv \frac{\alpha \bar{\rho}_0 \tilde{\mathbf{v}}_0 + (1-\alpha) \bar{\rho}_1 \tilde{\mathbf{v}}_1}{\alpha \bar{\rho}_0 + (1-\alpha) \bar{\rho}_1} \\ \tilde{\mathbf{v}}_0 \\ \tilde{u}_0 \\ \tilde{u}_1 \end{bmatrix} \quad (6.18)$$

There are other possible choices, but the general guidelines are to choose pressure and void fraction instead of phasic densities, and specific internal energies (or enthalpies) instead of total energies.

2. One of the nice features of using  $\bar{p}$  and  $\alpha$  in the solution vector is that non-linear algorithm defined by eq.(3.8) is avoided.
3. Non-linear iterations typically converge rapidly, within 3-10 iterations, which is in contrast to Picard-based iterations of segregated algorithms in Chapter 5 – it might take hundreds of outer iterations to reach an acceptable tolerance.

4. Care must be exercised in discretization of residuals eq.(6.9), to prevent “clicking” of Newton iterations, by removing/smoothing any possible discontinuities in evaluation of spatial discontinuities (limiters), fluxes (Riemann solvers) and closure laws (flow regime maps and transitions).
5. Linear problem eq.(6.14) is the most difficult and expensive part of the algorithm. In general, Jacobian matrix is non-symmetric, leading to rather limited choice for iterative solvers. They are all typically of Krylov-iteration based algorithm family, Section 6.4.1.
6. One of the nice features of the Jacobian-free version of linear solver (Section 6.4.2) is simplicity in code modification. This is particularly important when new closure laws are implemented, or when different options for space discretization and flux evaluation are employed. Also, no special effort is necessary when solution vector is based on  $\mathbf{V}$  instead of  $\mathbf{U}$  (no need for evaluation of  $\frac{\partial \mathbf{U}}{\partial \mathbf{V}}$ ), as the Jacobian matrix  $\mathbb{J}_{\mathbf{V}}^a$  is computed directly by perturbing  $\mathbf{V}$  (see Section 6.4.2).
7. Preconditioning of linear solver is the major factor in defining efficiency of Newton-based fully-implicit algorithms. CATHARE-1D utilizes ILU (math-based) preconditioners, which are known to have poor scalability properties, especially when applied in 3D. This is why only 1D module of CATHARE is Newton-based. Utilizing physics-based preconditioning [KK04], based on operator-splitting or segregated algorithms (Chapters 4 and 5) offers a potentially powerful and scalable solution for three-dimensional effective-field solvers.

### 6.3 CATHARE Algorithm

CATHARE’s one-dimensional module utilizes the *1st-order backward Euler* fully-implicit Newton-based algorithm<sup>2</sup> [BPB93]. All flux and source terms are com-

---

<sup>2</sup>In three-dimensional module, fully-implicit Newton-based method was considered too much time consuming, due to large matrix inversions needed. We believe this is mainly due to non-JFNK (computations of Jacobian are expensive) and poor ILU-(math)-based preconditioning. JFNK with physics-based preconditioning is expected to perform much better.

puted at  $(n + 1)$  time level:

**Mass**,  $k=0,1$ :

$$\text{res}_{\text{mass},k} = (\alpha_k \bar{\rho}_k)^{n+1} - (\alpha_k \bar{\rho}_k)^n - \Delta t \left( \nabla_{\rho_k} \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^{n+1} - \vec{\mathcal{S}}_{\text{mass},k}^{n+1} \right) \quad (6.19)$$

**Momentum**,  $k=0,1$ :

$$\begin{aligned} \text{res}_{\text{mom},k} &= (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^{n+1} - (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k)^n - \\ &\quad - \Delta t \left[ \nabla_{\mathbf{v}_k} \cdot (\alpha_k \bar{\rho}_k \tilde{\mathbf{v}}_k \otimes \tilde{\mathbf{v}}_k)^{n+1} - (\alpha_k \nabla_{\mathbf{v}_k} \bar{p})^{n+1} - \mathbb{S}_{\text{mom},k}^{n+1} \right] \end{aligned} \quad (6.20)$$

**Total energy**,  $k=0,1$ :

$$\begin{aligned} \text{res}_{\text{ene},k} &= (\alpha_k \bar{\rho}_k \tilde{e}_k)^{n+1} - (\alpha_k \bar{\rho}_k \tilde{e}_k)^n - \\ &\quad - \Delta t \left( \nabla_{e_k} \cdot (\tilde{\mathbf{v}}_k \alpha_k (\bar{\rho}_k \tilde{e}_k + \bar{p}))^{n+1} - (\bar{p} \tilde{\mathbf{v}}_k \cdot \nabla_{e_k} \alpha_k)^{n+1} - \vec{\mathcal{S}}_{\text{ene},k}^{n+1} \right) \end{aligned} \quad (6.21)$$

The Jacobian matrix

$$\mathbb{J} = \frac{\partial \vec{\text{res}}}{\partial \vec{\mathcal{X}}}$$

is computed analytically (no numerical derivatives are used), at each Newton iteration. Linear problems are solved using *conjugate gradient* method, with ILU-based “mathematics-based” preconditioning.

## 6.4 Fully-compressible, Fully-implicit, JFNK-based

*Jacobian-Free Newton-Krylov* (JFNK) based algorithms offer a better platform for solving effective-field models. The major ingredients of JFNK are

- (Inexact) Newton method (Sections 6.2 and 6.4.3) for eq.(6.8), combined with
- Jacobian-free versions of Krylov solvers for linear problem eq.(6.14) (e.g., GMRES, Section 6.4.1), and
- Physics-based preconditioning [KK04, KR00, KCMM03, KMCR05] (Section 6.5.2).

### 6.4.1 Krylov subspace iterations (GMRES)

One of the most popular linear solver used in JFNK is the Arnoldi-based *Generalized Minimal RESidual method (GMRES)* [SS86]. It belongs to the general class of Krylov subspace iteration methods. These projection (Galerkin) or generalized projection (Petrov-Galerkin) methods [Saa03] are suitable for solving non-symmetric linear systems of the form eq.(6.10) or eq.(6.14), using Krylov subspace,  $\mathbb{K}_j$ ,

$$\mathbb{K}_j = \text{span} \left( \vec{r}_0, \mathbb{J}\vec{r}_0, \mathbb{J}^2\vec{r}_0, \dots, \mathbb{J}^{j-1}\vec{r}_0 \right) \quad (6.22)$$

where  $\vec{r}_0 = \mathbb{J}^a \delta \vec{\mathcal{X}}_0^a + r \vec{e}_s \left( \vec{\mathcal{X}}^a \right)$ . In GMRES, the Arnoldi basis vectors form a trial subspace out of which the  $m^{\text{th}}$ -iteration solution is constructed:

$$\delta \vec{\mathcal{X}}_m^a = \delta \vec{\mathcal{X}}_0^a + \xi_0 \vec{r}_0 + \xi_1 \mathbb{J}\vec{r}_0 + \xi_2 \mathbb{J}^2\vec{r}_0 + \dots + \xi_m \mathbb{J}^m \vec{r}_0 \quad (6.23)$$

where  $(\xi_0, \xi_1, \dots, \xi_m)$  are “coordinates” of the  $m^{\text{th}}$  trial solution in the Krylov subspace. As one can see, only matrix-vector products are required to create new trial vectors. The iterations are terminated based on a by-product (free) estimate of the residual that does not require explicit construction of intermediate residual vectors. This is a major advantage of GMRES over other Krylov methods. GMRES has a residual minimization property in the Euclidean norm. The major drawback of GMRES is that it requires the storage of all previous Arnoldi/(Krylov) basis vectors. This problem can be alleviated with efficient preconditioning (see section 6.5).

### 6.4.2 Jacobian-free implementation

Since GMRES does not require individual elements of the Jacobian matrix  $\mathbb{J}$ , it never needs to be constructed. Instead only matrix-vector multiplications  $\mathbb{J}\vec{k}$  are needed, where  $\vec{k} \in (\vec{r}_0, \mathbb{J}\vec{r}_0, \mathbb{J}^2\vec{r}_0, \dots)$  are Krylov vectors. Thus, *Jacobian-free implementations* are possible. The action of the Jacobian matrix can be approximated by Fréchet derivatives

$$\mathbb{J}\vec{k} \approx \frac{r \vec{e}_s \left( \vec{\mathcal{X}} + \varepsilon \vec{k} \right) - r \vec{e}_s \left( \vec{\mathcal{X}} \right)}{\varepsilon} \quad (6.24)$$

There are two approaches for choosing  $\varepsilon$ .

**Brown & Saad.** The first approach is taken from [BS90]:

$$\varepsilon = \begin{cases} \epsilon_{\text{rel}} \frac{\vec{\mathcal{X}}^T \vec{\kappa}}{\|\vec{\kappa}\|_2^2} & \text{if } \left| \vec{\mathcal{X}}^T \vec{\kappa} \right| > \mathcal{X}_{\text{min}} \|\vec{\kappa}\|_1 \\ \epsilon_{\text{rel}} \mathcal{X}_{\text{min}} \text{Sign} \left( \vec{\mathcal{X}}^T \vec{\kappa} \right) \frac{\|\vec{\kappa}\|_1}{\|\vec{\kappa}\|_2} & \text{otherwise} \end{cases} \quad (6.25)$$

There are two control parameters:  $\epsilon_{\text{rel}}$  and  $\mathcal{X}_{\text{min}}$ .

**Pernice & Walker.** The second approach is taken from [PW98]:

$$\varepsilon = \frac{\epsilon_{\text{rel}} \sqrt{1 + \|\vec{\mathcal{X}}\|}}{\|\vec{\kappa}\|} \quad (6.26)$$

The only control parameter is  $\epsilon_{\text{rel}}$ . Note that for the entire linear iterative process  $\vec{\mathcal{X}}$  does not change. Therefore,  $\sqrt{1 + \|\vec{\mathcal{X}}\|}$  need be computed only once.

With the Jacobian-free formulation, the work associated with forming the Jacobian matrix and its storage can be eliminated, which is a significant saving of both CPU time and storage, provided that the number of Krylov vectors is kept small (see section 6.5). Moreover, in many non-linear applications, the Jacobian matrix is not available due to size and complexity.

### 6.4.3 Inexact Newton

One important modification to Newton's method employed in JFNK is called an *inexact Newton's method* [KK04]. The term “inexact” refers to the accuracy of the iterative linear solver. The basic idea is that the linear system must be solved to a tight tolerance only when the added accuracy matters – i.e., when it affects the convergence of the Newton's iterations. This is accomplished by making the convergence of the linear residual proportional to the non-linear residual:

$$\left\| \mathbb{J}^a \delta \vec{\mathcal{X}}_m^a + r \vec{e}_s \left( \vec{\mathcal{X}}^a \right) \right\| \leq \nu_a \left\| r \vec{e}_s \left( \vec{\mathcal{X}}^a \right) \right\| \quad (6.27)$$

By default,  $\nu_a$  is a constant. Alternatively, one can invoke the algorithm by Eisenstat and Walker [EW96], which computes  $\nu_a$  at each step of the nonlinear solver.

## 6.5 Preconditioning

Let us consider a solution of the linear problem eq.(6.14), using Krylov subspace iterations. If the problem is stiff, the Jacobian matrix  $\mathbb{J}_v^a$  is ill-conditioned, which implies that a significant number Krylov subspace dimensions must be introduced to reach a converged solution. This is the major drawback of Krylov method: forming additional dimensions not only CPU-consuming, but also extremely demanding in memory, as each added  $n^{\text{th}}$  Krylov vector  $\mathbf{k}_n$  is of the size of the solution vector  $\vec{\mathcal{X}}$ .

In practice, unpreconditioned Krylov methods are seldomly used. To improve performance, Krylov methods **must be preconditioned** [SS86] to effectively cluster eigenvalues of the iteration matrix. Here, we will consider *right* preconditioning techniques<sup>3</sup>, as these are best suited for physics-based preconditioning discussed in Section 6.5.2.

Consider the following modification of eq.(6.14):

$$\underbrace{\mathbb{J}_v^a \mathbb{P}^{-1}}_{\mathbb{J}_p^a} \underbrace{\mathbb{P} \delta \vec{\mathcal{X}}_v^a}_{\delta \vec{\mathcal{Y}}} = \underbrace{-r \vec{e}_{s_U}}_{\vec{b}} \left( \vec{\mathcal{X}}_v^a \right) \quad (6.28)$$

where  $\mathbb{P}$  symbolically represents the preconditioning matrix (or process), and  $\mathbb{P}^{-1}$  is its inverse. Thus, the solution procedure is splitted into two processes:

1. Solving for

$$\mathbb{J}_p^a \delta \vec{\mathcal{Y}} = \vec{b} \quad (6.29)$$

(this is what actually crunched by GMRES), and

2. Preconditioning:

$$\delta \vec{\mathcal{X}}_v^a = \mathbb{P}^{-1} \delta \vec{\mathcal{Y}} \quad (6.30)$$

While one refers to the matrix/process  $\mathbb{P}$ , operationally the algorithm only requires the action of  $\mathbb{P}^{-1}$  on a vector. The main requirement is that  $\mathbb{P}$  designed properly, to enable clustering eigenvalues of the  $\mathbb{J}_p^a$ , making the solution of eq.(6.29) to converge faster.

<sup>3</sup>Left preconditioning changes the norm of the residual by which convergence to a linear iterative method is generally measured, [KK04, SS86].

### 6.5.1 Matrix-based preconditioning

If the Jacobian matrix  $\mathbb{J}_v^a$  is given (say, computed analytically as in the CATHARE-1D, or being evaluated by applying Fréchet derivatives eq.(6.24) on a sequence of unit vectors<sup>4</sup>), then a number of well-established *matrix-based preconditioning* techniques are available. These include different flavors of SOR, SSOR, ILU, MILU, ILUT, ILUTP, ILUS, ILUC, etc. preconditioners, see [SS86] for review. In these cases, the preconditioning matrix  $\mathbb{P}$  is a suitable approximation for  $\mathbb{J}_v^a$ .

Computation of the Jacobian matrix however is extremely expensive (both CPU- and memory-wise). Moreover, most of the matrix-based preconditioning techniques tend to scale poorly with increase of the number of unknowns, especially for 3D applications. Therefore, it is generally agreed that the use of the Newton-Krylov (NK) method prohibitively expensive for 3D reactor simulations [BPB93].

### 6.5.2 Physics-(process)-based preconditioning (PBP)

The main premise for the *Jacobian-free Newton Krylov (JFNK)* method is that the computation of the Jacobian matrix is completely avoided. In this case, the solution of the equation (6.29) is obtained by the Jacobian-free version of GMRES (Section 6.4.2), while the action of preconditioning  $(\mathbb{P}^{-1}\delta\vec{y})$  is a *process*, possibly formed as a linear combination of approximate inverses of submatrices [KK04]. We refer to this approach as *physics-based* preconditioning, because typically one would like to use available and well-tested (as a solver) operator-splitting algorithms to represent  $(\mathbb{P}^{-1}\delta\vec{y})$ . In the context of the present discussion, these could any of the described algorithms from Chapters 4 and 5.

Advantages of the PBP are discussed and demonstrated in [KK04, KR00, KCM03, KMK96, KMCR05], for a range of applications, starting from time-dependent diffusion equations, MHD equations, steady-state Navier-Stokes equations, shallow-water equations, etc. If the preconditioning process is designed well, one could solve linear problem faster than computation time needed for

<sup>4</sup>PETSc [BBE<sup>+</sup>04] provides very nice and efficient MatFDColoring routines for finite-difference Jacobian approximations.

Jacobian evaluations<sup>5</sup>. Moreover, when combined with efficient multigrid algorithms utilized in the physics-based preconditioning process (designed to extract dominantly elliptic component of the problem, when appropriate<sup>6</sup>), one can achieve very scalable algorithms, which would make 3D reactor simulations cost-effective.

There are a number of available tools for implementing JFNK with matrix- and process-based precondition. An example of pseudo code for PETSc's SNES package is given in Appendix B.

---

<sup>5</sup>Special care must be exercised to ensure that discrete forms of equations in the preconditioning process are consistent with discrete forms of the original linear problem.

<sup>6</sup>For example, when solving relatively low-speed or slow two-phase flow problems.

*This Page is Intentionally Left Blank*

## Chapter 7

# Phase appearance and disappearance

**P**HASE appearance and disappearance presents very serious numerical challenge for all effective field models. To our knowledge, no fully satisfactory solutions exist. There are several *ad hoc* numerical treatments used in different codes. We will discuss these fixes next.

### 7.1 CATHARE strategy

CATHARE strategy for dealing with phase appearance/disappearance described by Bestion in [Bes00]. The main idea is keep void fraction bounded in the range  $[\alpha_{\min}, \alpha_{\max}]$  by manipulating wall and interfacial exchange terms for phasic mass, momentum and energy equations, combined with time step control. Another driving consideration is to force the disappearing phase to stay thermally at/near the saturation state, and, mechanically, at equilibrium with carrying/dominant phase. More specifically, for two-fluid formulation,

$$\begin{aligned} \tilde{T}_{(d)} &\rightarrow \tilde{T}_{\text{sat}} \\ \tilde{\mathbf{v}}_{(d)} &\rightarrow \tilde{\mathbf{v}}_{(c)} \end{aligned} \quad (7.1)$$

where  $(d)$  denotes disappearing phase, while  $(c)$  stands for “carrying” phase. Thus, the phasic governing conservation equations are solved even when the phase essentially non-existent,  $\alpha_d \approx \alpha_{\min}$ , avoiding any void fraction cut-off procedures, which are undesirable not only from the point of view conservation, but also be-

cause they have adverse effects on convergence of iterative procedures.

Interfacial exchange terms appear in source terms of eqs.(2.46), (2.47) and (2.48), as

$$\begin{aligned}
 \mathcal{S}_{\text{mass},k} &\rightarrow \Gamma \\
 \mathcal{S}_{\text{mom},k} &\rightarrow \Gamma \mathbf{v}_{ki}^m \\
 \mathcal{S}_{\text{ene},k} &\rightarrow \Gamma \left( u_{ki} + \frac{(v_{ki}^e)^2}{2} \right)
 \end{aligned} \tag{7.2}$$

(see also eqs.(2.3), (2.4) and (2.5) of Chapter 2), where  $\Gamma$ ,  $\mathbf{v}_{ki}^m$ ,  $u_{ki}$  and  $v_{ki}^e$  are modeling (closure) parameters. From the current discussion perspective, interfacial mass exchange terms are of importance. They are modeled as

$$\Gamma = \frac{q_{w,i} - q_{i,v} - q_{i,l}}{H_{vl}} \tag{7.3}$$

where  $H_{vl}$  is the latent heat of evaporation/condensation phase change, while the interfacial and wall heat fluxes are given by

$$\begin{aligned}
 q_{w,i} &= h_{w,i} (T_w - T_{\text{sat}}) \\
 q_{i,v} &= a_i h_{i,v} (T_{\text{sat}} - \tilde{T}_v) \\
 q_{i,l} &= a_i h_{i,l} (T_{\text{sat}} - \tilde{T}_l)
 \end{aligned} \tag{7.4}$$

$T_w$ ,  $T_{\text{sat}}$ ,  $\tilde{T}_v$  and  $\tilde{T}_l$  are wall, saturation, vapor and liquid averaged temperatures, correspondingly. Wall-to-, vapor-to- and liquid-to-interface heat transfer coefficients are defined by  $h_{w,i}$ ,  $h_{i,v}$  and  $h_{i,l}$ , respectively. Interfacial area density is given by  $a_i$ .

### 7.1.1 Void fraction bounding and thermal conditioning

The necessary (but not sufficient) conditions to keep  $\alpha_k \in [0, 1]$  are:

1. No condensation when there is no vapour:  $\alpha_v = 0 \Rightarrow \Gamma \geq 0$ . From eq.(7.3), the necessary limiting conditions on interfacial heat transfer coefficients are:

$$\left\{ \begin{array}{l} q_{i,l} \leq 0 \Rightarrow \left| \begin{array}{l} \text{if } (\tilde{T}_l < T_{\text{sat}}) \Rightarrow a_i h_{i,l} = 0 \\ \text{if } (\tilde{T}_v < T_{\text{sat}}) \Rightarrow a_i h_{i,v} = 0 \\ \text{if } (T_w < T_{\text{sat}}) \Rightarrow h_{w,i} = 0 \end{array} \right. \\ q_{i,v} \leq 0 \Rightarrow \\ q_{w,i} \geq 0 \Rightarrow \end{array} \right. \tag{7.5}$$

2. No evaporation when there is no liquid:  $\alpha_l = 0 \Rightarrow \Gamma \leq 0$ . From eq.(7.3), the necessary limiting conditions on interfacial heat transfer coefficients are:

$$\left\{ \begin{array}{l} q_{i,l} \geq 0 \Rightarrow \left| \begin{array}{l} \text{if } (\tilde{T}_l > T_{\text{sat}}) \Rightarrow a_i h_{i,l} = 0 \\ \text{if } (\tilde{T}_v > T_{\text{sat}}) \Rightarrow a_i h_{i,v} = 0 \\ \text{if } (T_w > T_{\text{sat}}) \Rightarrow h_{w,i} = 0 \end{array} \right. \\ q_{i,v} \geq 0 \Rightarrow \\ q_{w,i} \leq 0 \Rightarrow \end{array} \right\} \quad (7.6)$$

These limiting conditions are necessary, but not sufficient. To enforce  $\alpha_k \in [\alpha_{\min}, \alpha_{\max}]$ <sup>1</sup>, additional “*residual phase treatments*” are introduced as follows.

### Bounding at $\alpha_v \approx 0$ (single-phase liquid)

When  $\alpha_v \rightarrow 0$ , the following conditioning on interfacial heat fluxes is enforced:

- $q_{i,v}$  must force  $\tilde{T}_v$  to be close to  $T_{\text{sat}}$  (this is **thermal conditioning**). Thus,

$$\boxed{[0 < \alpha_v \leq \alpha_{\min}] \text{ or } [\alpha_{\min}^+ \leftarrow \alpha_v]} \Rightarrow a_i h_{i,v} \gg 0 \quad (7.7)$$

- $q_{w,v}$  must not heat residual vapor when  $T_w > \tilde{T}_v$ . Thus,

$$\begin{aligned} \boxed{[0 < \alpha_v \leq \alpha_{\min}]} &\Rightarrow q_{w,v} = 0 \\ \boxed{[\alpha_{\min}^+ \leftarrow \alpha_v]} &\Rightarrow q_{w,v} \rightarrow 0 \end{aligned} \quad (7.8)$$

- $q_{w,i}$  and  $q_{i,l}$  must not condense residual vapour:

$$\begin{aligned} \boxed{[0 < \alpha_v \leq \alpha_{\min}]} &\Rightarrow \begin{cases} q_{i,l} \leq 0 \\ q_{w,i} \geq 0 \end{cases} \\ \boxed{\begin{array}{l} [\alpha_{\min}^+ \leftarrow \alpha_v] \\ T_l < T_{\text{sat}} \end{array}} &\Rightarrow q_{i,l} \rightarrow 0^- \end{aligned} \quad (7.9)$$

<sup>1</sup>In CATHARE,  $\alpha_{\min} = 10^{-5}$  and  $\alpha_{\max} = (1 - 10^{-6}) = 0.99999$ .

## 7.1. CATHARE STRATEGY

121

These conditions should render interfacial mass transfer  $\Gamma$  positive or small-negative<sup>2</sup>. When small-negative, some small condensation of the residual vapour occur. The following numerical residual mass transfer is added to attract  $\alpha_v \approx \alpha_{\min}$ :

$$\Gamma = \frac{q_{w,i} - q_{i,v} - q_{i,l}}{H_{vl}} + \underbrace{\bar{\rho}_v \frac{\alpha_{\min} - \alpha_v}{\tau_v}}_{\Gamma_{res}} \quad (7.10)$$

where  $\tau_v = 10^{-5}$ .

### Bounding at $\alpha_l \approx 0$ (single-phase vapour)

When  $\alpha_l \rightarrow 0$ , the following conditioning on interfacial heat fluxes is enforced:

- $q_{i,l}$  must force  $\tilde{T}_l$  to be close to  $T_{\text{sat}}$  (this is **thermal conditioning**). Thus,

$$\boxed{[0 < \alpha_l \leq (1 - \alpha_{\max})] \text{ or } [(1 - \alpha_{\max})^+ \leftarrow \alpha_l]} \Rightarrow a_i h_{i,l} \gg 0 \quad (7.11)$$

- $q_{w,l}$  must not heat residual liquid when  $T_w > \tilde{T}_l$ . Thus,

$$\begin{aligned} \boxed{[0 < \alpha_l \leq (1 - \alpha_{\max})]} &\Rightarrow q_{w,l} = 0 \\ \boxed{[(1 - \alpha_{\max})^+ \leftarrow \alpha_l]} &\Rightarrow q_{w,l} \rightarrow 0 \end{aligned} \quad (7.12)$$

- $q_{w,i}$  and  $q_{i,l}$  must not condense residual liquid:

$$\begin{aligned} \boxed{[0 < \alpha_l \leq (1 - \alpha_{\max})]} &\Rightarrow \begin{cases} q_{i,l} \geq 0 \\ q_{w,i} \leq 0 \end{cases} \\ \boxed{\begin{matrix} [(1 - \alpha_{\max})^+ \leftarrow \alpha_l] \\ \tilde{T}_l > T_{\text{sat}} \end{matrix}} &\Rightarrow q_{i,l} \rightarrow 0^+ \end{aligned} \quad (7.13)$$

<sup>2</sup>It can be negative when  $q_{i,l} = 0$  and  $q_{i,v} > 0$ .

These conditions should render interfacial mass transfer  $\Gamma$  negative or small-positive<sup>3</sup>. When small-positive, small evaporation of the residual liquid occur. To prevent this, the following numerical residual mass transfer is added to attract  $\alpha_l \approx (1 - \alpha_{\max})$ :

$$\Gamma = \frac{q_{w,i} - q_{i,v} - q_{i,l}}{H_{vl}} + \underbrace{\bar{\rho}_l \frac{1 - \alpha_{\max} - \alpha_l}{\tau_l}}_{\Gamma_{res}} \quad (7.14)$$

where  $\tau_l = 10^{-5}$ .

In addition to the above-conditioning, CATHARE adds *time step control*, preventing large time steps which can overshoot and make out-of-bounds solutions.

### 7.1.2 Velocity conditioning

To enforce velocity-equilibrium at the limit of vanishing phase, the following numerical conditioning is applied on **interfacial and wall friction**:

- When  $\alpha_v \rightarrow \alpha_{\min}^+$  and  $0 < \alpha_v \leq \alpha_{\min}$ , or when  $\alpha_l \rightarrow (1 - \alpha_{\max})^+$  and  $0 < \alpha_l \leq (1 - \alpha_{\max})$ , interfacial friction is set to very high value.
- When  $\alpha_v \rightarrow \alpha_{\min}^+$  and  $0 < \alpha_v \leq \alpha_{\min}$ , wall friction is set to very small value.

### 7.1.3 Discussion

1. The physical arguments behind the above-discussed conditionings are meaningful only for *boiling/condensing dispersed two-phase flows* in reactor thermalhydraulics:
  - (a) When vapour bubbles appear/disappear in bulk flow during direct contact condensation, their radius is small and interfacial drag is very high, making their velocity to be very close to the velocity of the carrying liquid. Also, it is reasonable to assume that bubbles formed at

<sup>3</sup>It can be negative when  $q_{i,v} = 0$  and  $q_{i,l} < 0$ .

*saturation temperature*<sup>4</sup>.

- (b) Similarly, when liquid droplets appear/disappear under bulk condensation during depressurization, or bulk evaporation during reflooding, the interfacial drag is large due to small drop radii, making the equilibrated phasic velocities assumption meaningful. Also, it is reasonable to assume that droplets generated/collapsed at *saturation temperature*<sup>5</sup>.
2. The physical arguments for CATHARE velocity conditioning are **breaking down** for [Bes00]:
    - (a) Forming the first vapour bubbles *at the wall* (the bubbles tend to stay or slide at the wall).
    - (b) Forming the first vapour bubbles during *flashing process with heterogeneous nucleation along the walls*, which is more likely to occur than bulk homogeneous nucleation.
    - (c) Liquid film dryout by wall-heating.
    - (d) Film condensation by wall cooling.
  3. Because of the Newton-based iterative procedures used in the CATHARE, the above-discussed conditionings must be implemented smoothly, to prevent “clicking” of Newton iterations.

## 7.2 Paillère at al. treatment

In [PCGC03], Paillère et al. utilized a variation of the AUSM scheme [Lio06, CL07] to solve 4- and 6-equation two-fluid models. Recognizing importance of phase appearance and disappearance, they introduced very simple treatment of the disappearing phase. Similar to CATHARE, it is assumed that the “residual” (or

---

<sup>4</sup>In practice, the pressure inside bubbles is higher due to surface tension, so that they can appear only with *certain liquid overheating*. This effect can be accounted for by either including *bulk pressure difference* term (see eq.(2.47)), or by *modeling a flashing delay* in the evaporation law.

<sup>5</sup>Accounting for surface tension-caused droplet over-pressure by including *bulk pressure difference* term of eq.(2.47) should incorporate the realistic physical effects of having the necessary overcooled-vapour conditions.

“vanishing”) phase have the same velocity as the “carrying” phase. However, no attempt is made to enforce this condition through the interfacial exchange terms. Instead, when the velocity of the “residual” phase is required (e.g., when computing AUSM-based fluxes), it is computed using the following simple formula:

$$\tilde{v}_{(d)} = \mathfrak{G}(\alpha_{(d)}) \frac{(\alpha_{(d)} \bar{\rho}_{(d)} \tilde{v}_{(d)})}{(\alpha_{(d)} \bar{\rho}_{(d)})} + (1 - \mathfrak{G}(\alpha_{(d)})) \frac{(\alpha_{(c)} \bar{\rho}_{(c)} \tilde{v}_{(c)})}{(\alpha_{(c)} \bar{\rho}_{(c)})} \quad (7.15)$$

where  $(\alpha_{(k)} \bar{\rho}_{(k)})$  and  $(\alpha_{(k)} \bar{\rho}_{(k)} \tilde{v}_{(k)})$  are the solutions of the phasic mass and momentum equations. Positive function  $\mathfrak{G}(\alpha_{(d)})$  is introduced to provide a smooth transition (it becomes 1 when  $\alpha_{(d)} \geq \alpha_{\min}$ ). The value  $\alpha_{\min} = 10^{-4}$  was used in [PCGC03].

Similar consideration is applied for temperature of the “residual” phase – it is assumed to be equal to the one of the “carrying” fluid, at the limit  $\alpha_{(d)} \rightarrow 0$ , by applying an equation similar to eq.(7.15).

Importantly, *no cut-off* is actually applied to the void fraction itself.

Essentially the same treatment of the vanishing phase is used by Chang and Liou [CL07]:

$$\Psi_{\text{adjust}} = \mathfrak{G}(\xi) \Psi_{(d)} + (1 - \mathfrak{G}(\xi)) \Psi_{(c)}, \quad \Psi = \tilde{\mathbf{v}}, \tilde{T} \quad (7.16)$$

where

$$\mathfrak{G}(\xi) = -\xi^2 (2\xi - 3), \quad \xi = \frac{\alpha_{(d)} - \epsilon_{\min}}{\epsilon_{\max} - \epsilon_{\min}} \quad (7.17)$$

where  $\epsilon_{\min}$  is set to  $10^{-7}$ , while  $\epsilon_{\max}$  is in the range from  $10^{-3}$  to  $10^{-6}$ .

### Discussion

1. The above treatment is reasonable for no-phase-change (boiling flows) configurations.
2. This treatment is applied when explicit algorithms for solving two-fluid equations, Chapter 3.

*This Page is Intentionally Left Blank*

## Chapter 8

# Concluding Remarks

A number of algorithms developed in the past 40 years for numerical solution of averaged multi-phase flow equations have been reviewed. While many of these (operator-splitting-based) algorithms have proven in time to be suitable for application in reactor thermalhydraulics codes, the fully-implicit algorithms are the future, as the strong non-linearity and complexity of governing equations and constitutive physics imposes serious challenge, which can be overcome with tight coupling and removing stability constraints. The major challenge in implementing the fully-implicit algorithms is the efficiency and robustness. While segregated SIMPLE-based algorithms are the base for all current commercial CFD codes, these are known to have robustness and numerical convergence issues due to Picard iterations employed to solve non-linearly coupled equations. On the other hand, while Newton-based algorithms are attractive from the point of view of their robustness and mathematical properties, they are quite expensive from the point of view of memory requirement and difficulties to precondition. We believe that the future is in the combination of operator-splitting, segregated and Newton-based algorithms, where some form of inexact Newton method should be used for guiding non-linear iterations, with matrix-free form of Krylov (GMRES) subspace iterations used for linear steps, and operator-splitting (“semi-implicit”, SETS, Nearly-Implicit) and simplified versions of segregated algorithms utilized as physics-based precondition techniques. This approach is chosen to be implemented in Hydra-TH.

*This Page is Intentionally Left Blank*

# APPENDICES

*This Page is Intentionally Left Blank*

## Appendix A

### Cartesian Vector Calculus

Given two Cartesian vectors  $\mathbf{a} = \{a_x, a_y, a_z\}^T$  and  $\mathbf{b} = \{b_x, b_y, b_z\}^T$ , the *dot product* is defined as

$$\mathbf{a} \cdot \mathbf{b} = a_x b_x + a_y b_y + a_z b_z = a_k b_k \quad (\text{A.1})$$

The *dyadic product* is denoted as [Ari]

$$\mathbf{ab} = \mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_x b_x & a_x b_y & a_x b_z \\ a_y b_x & a_y b_y & a_y b_z \\ a_z b_x & a_z b_y & a_z b_z \end{bmatrix} = a_k b_l \quad (\text{A.2})$$

*Spatial derivatives* are denoted as

$$\nabla = \left\{ \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right\}^T = \{\partial_x, \partial_y, \partial_z\}^T = \partial_j \quad (\text{A.3})$$

Thus, the *gradient of an arbitrary scalar*  $\varphi$  is defined as

$$\nabla \varphi = \left\{ \frac{\partial \varphi}{\partial x}, \frac{\partial \varphi}{\partial y}, \frac{\partial \varphi}{\partial z} \right\}^T = \{\partial_x \varphi, \partial_y \varphi, \partial_z \varphi\}^T = \partial_j \varphi \quad (\text{A.4})$$

and, accordingly, dot product of a vector and a gradient of a scalar is

$$\mathbf{a} \cdot \nabla \varphi = a_x \frac{\partial \varphi}{\partial x} + a_y \frac{\partial \varphi}{\partial y} + a_z \frac{\partial \varphi}{\partial z} \quad (\text{A.5})$$

*Laplacian of an arbitrary scalar* is defined as

$$\nabla \cdot \nabla \varphi = \nabla^2 \varphi = \Delta \varphi = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} + \frac{\partial^2 \varphi}{\partial z^2} = \partial_k^2 \varphi \quad (\text{A.6})$$

*Divergence of a vector* is defined as

$$\nabla \cdot \mathbf{a} = \frac{\partial a_x}{\partial x} + \frac{\partial a_y}{\partial y} + \frac{\partial a_z}{\partial z} \quad (\text{A.7})$$

*Gradient of an arbitrary vector* is a tensor, defined as

$$\nabla \mathbf{a} = \begin{bmatrix} \frac{\partial a_x}{\partial x} & \frac{\partial a_x}{\partial y} & \frac{\partial a_x}{\partial z} \\ \frac{\partial a_y}{\partial x} & \frac{\partial a_y}{\partial y} & \frac{\partial a_y}{\partial z} \\ \frac{\partial a_z}{\partial x} & \frac{\partial a_z}{\partial y} & \frac{\partial a_z}{\partial z} \end{bmatrix} \quad (\text{A.8})$$

and its *transpose*:

$$\nabla \mathbf{a}^T = \begin{bmatrix} \frac{\partial a_x}{\partial x} & \frac{\partial a_y}{\partial x} & \frac{\partial a_z}{\partial x} \\ \frac{\partial a_x}{\partial y} & \frac{\partial a_y}{\partial y} & \frac{\partial a_z}{\partial y} \\ \frac{\partial a_x}{\partial z} & \frac{\partial a_y}{\partial z} & \frac{\partial a_z}{\partial z} \end{bmatrix} \quad (\text{A.9})$$

Scalar product of a vector and divergence of a vector is a vector:

$$\mathbf{a} \cdot \nabla \mathbf{b} = \begin{Bmatrix} a_x \cdot \nabla b_x \\ a_y \cdot \nabla b_y \\ a_z \cdot \nabla b_z \end{Bmatrix} = \begin{Bmatrix} a_x \frac{\partial b_x}{\partial x} + a_y \frac{\partial b_x}{\partial y} + a_z \frac{\partial b_x}{\partial z} \\ a_x \frac{\partial b_y}{\partial x} + a_y \frac{\partial b_y}{\partial y} + a_z \frac{\partial b_y}{\partial z} \\ a_x \frac{\partial b_z}{\partial x} + a_y \frac{\partial b_z}{\partial y} + a_z \frac{\partial b_z}{\partial z} \end{Bmatrix} \quad (\text{A.10})$$

Divergence of a dyadic product of two vectors is defined as

$$\nabla \cdot (\mathbf{ab}) = \nabla \cdot (\mathbf{a} \otimes \mathbf{b}) = \begin{Bmatrix} \frac{\partial}{\partial x} (a_x b_x) + \frac{\partial}{\partial y} (a_x b_y) + \frac{\partial}{\partial z} (a_x b_z) \\ \frac{\partial}{\partial x} (a_y b_x) + \frac{\partial}{\partial y} (a_y b_y) + \frac{\partial}{\partial z} (a_y b_z) \\ \frac{\partial}{\partial x} (a_z b_x) + \frac{\partial}{\partial y} (a_z b_y) + \frac{\partial}{\partial z} (a_z b_z) \end{Bmatrix} \quad (\text{A.11})$$

## Appendix B

# SNES JFNK-PBP pseudo-code example

IN the present chapter, we provide a blue-print/recipe for C/C++ implementation of the JFNK with physics-based preconditioning, using ANL's `PETSC` `SNES` and `KSP` package. We presume that some preliminary knowledge of PETSc programming, such as vector/matrix initiation and assembly, distributing parallel data, etc. More detail information and comprehensive introduction to PETSc can be found in [BBE<sup>+</sup>04, BBG<sup>+</sup>01]. We base our discussion on Release Version 3.3. The actual pseudo-code lines are given in boxes, with comments and remarks in between.

At the code initiation stage, the `SNES` library of `PETSC` must be instantiated as:

```
SNES _snes;
CHKERRQ (SNEScreate (PETSC_COMM_WORLD, &_snes) );
```

The pointer to `_snes` must be made available for all relevant solution routines.

There are five steps needed for implementation of the `SNES`-based JFNK-PBP:

1. Configure `SNES` and `KSP`: set all needed options, connect user-supplied `Form_Function()`, `Precond_SetUp()` and `PBP_precondition()` routines. These are discussed in Sections [B.1.1](#) and [B.1.2](#).
2. Execution step, Section [B.2](#). Called within time update loop, each time solution to non-linear problem is needed.

3. Implement residual evaluation `Form_Function()` routine, Section **B.3**. It is called by **PETSC** each time residual evaluation is required.
4. Implement preconditioning setup `Precond_SetUp()` routine, Section **B.4**. It is called by **PETSC** at the beginning of each Newton iteration.
5. Implement process/physics-based preconditioning `PBP_precondition()` routine, Section **B.5**. Called by **PETSC** at each Krylov iteration.

## B.1 Configuration

Define and connect residual evaluation function `Form_Function()` as

```
CHKERRQ(SNESSetFunction(_snes, ResV, Form_Function, (void *)ctx_class));
```

where:

`(Vec ResV)` is an appropriately initiated/allocated residual vector (see [BBE<sup>+</sup>04] for PETSc data structure definition),

`(PetscErrorCode Form_Function(SNES, Vec, Vec, void*))` is residual function evaluation routine (see Section **B.3**), and

`(ctx_class)` is a user-provided class to enable connection with C++ classes (if necessary).

### B.1.1 Configure SNES

First, the line search Newton method is defined by

```
CHKERRQ(PetscOptionsSetValue("-snes_type", "ls"));
CHKERRQ(PetscOptionsSetValue("-snes_ls", "basic"));
```

The other line search options are "cubic", "quadratic" and "basicnorm" (see [BBE<sup>+</sup>04] for details).

Non-linear iteration tolerances are set as

```
CHKERRQ(SNESSetTolerances(_snes, _atol, _rtol, _stol, _MAXIT_NEWTON, _max_funcs));
```

where `_MAXIT_NEWTON` is the maximum allowable number of nonlinear iterations and `_max_funcs` is the maximum allowable number of function evaluations. Non-linear iterations are declared converged when:

- The norm of the change in the solution between successive iterations is less than `_stol` (by default, set to  $10^{-50}$ ).
- Absolute size of the norm is less than `_atol` (say,  $10^{-13}$ ).
- Relative (to initial guess) size of the norm is less than `_rtol` (say,  $10^{-8}$ ).

Next, define the matrix-free method to be used for (Jacobian  $\times$  vector) product evaluation,

```
CHKERRQ(PetscOptionsSetValue("-snes_mf",""));
CHKERRQ(PetscOptionsSetValue("-snes_mf_type","default"));
```

This will make `PETSC` to use Fréchet derivative evaluation eq.(6.24) with perturbation computed with eq.(6.25). To use eq.(6.26), set option `"wp"` in place of `"default"`. The following option will make use of the Eisenstat and Walker [EW96] algorithm to compute  $\nu_a$  for eq.(6.27):

```
CHKERRQ(PetscOptionsSetValue("-snes_ksp_ew_conv",""));
```

Two control parameters of matrix-free perturbation by eq.(6.25) –  $\epsilon_{\text{rel}}$  and  $\mathcal{X}_{\text{min}}$ , are set by

```
CHKERRQ(PetscOptionsSetValue("-snes_mf_err","1.E-8"));
CHKERRQ(PetscOptionsSetValue("-snes_mf_umin","1.E-6"));
```

The next stage is to set and configure linear solver (KSP). This is described in Section B.1.2.

## B.1.2 Configure KSP

After SNES options are specified, one must define KSP linear solver and its preconditioner.

## B.1. CONFIGURATION

135

```
KSP ksp;
PC pc;
CHKERRQ (SNESGetKSP (_snes, &ksp) );
CHKERRQ (KSPSetPCSide (ksp, PC_RIGHT) );
CHKERRQ (KSPGetPC (&ksp, pc) );
```

This will extract pointers to linear solver and preconditioner, and sets `KSP` to be right-preconditioned (see eq.(6.28)). Next, specify restarted modified-Gram-Schmidt GMRES as a type of linear solver:

```
CHKERRQ (KSPSetType (ksp, KSPGMRES) );
CHKERRQ (KSPGMRESRestart (ksp, _maxGMRES_itr) );
CHKERRQ (PetscOptionsSetValue ("-ksp_gmres_modifiedgramschmidt", ""));
CHKERRQ (KSPGMRESSetCGSRefinementType (&ksp, KSP_GMRES_CGS_REFINE_NEVER) );
```

where `_maxGMRES_itr` is the maximum GMRES iterations before restarting. Tolerance of `KSP` is specified as

```
CHKERRQ (KSPSetTolerances (ksp, _ksp_rtol, _ksp_abstol, _ksp_dtol, _ksp_max_its) );
```

Linear solver convergence (or divergence) is based on  $\mathcal{L}_2$ -norm of the linear residual vector,  $\vec{r} = (\vec{b} - \mathbb{J}_P^a \delta \vec{Y})$  (see also eq.(6.29)), and it is defined by three quantities:

- The decrease of the linear residual norm relative to the norm of the non-linear residual,  $\vec{b}$ , and
- the absolute size of the linear residual norm. Thus, convergence is declared at iteration  $k$  if

$$\|\vec{r}_k\|_2 < \max \left( \_ksp\_rtol \cdot \|\vec{b}\|_2, \_ksp\_atol \right) \quad (\text{B.1})$$

- Divergence is detected if

$$\|\vec{r}_k\|_2 > \_ksp\_dtol \cdot \|\vec{b}\|_2 \quad (\text{B.2})$$

Default values are  $\_ksp\_atol=10^{-50}$ ,  $\_ksp\_rtol=10^{-5}$ ,  $\_ksp\_dtol=10^5$ .

As a final KSP configuration step, set the routines for physics/process-based preconditioning:

```
CHKERRQ(PCSetType(pc, PCSHELL));
CHKERRQ(PCShellSetContext(pc, (void *)ctx_class));
CHKERRQ(PCShellSetSetup(pc, Precond_SetUp)); CHKERRQ(PCShellSetApply(pc, PBP_
precondition));
```

where:

(`ctx_class`) is a user-provided class to enable connection with C++ classes (if necessary).

(`PetscErrorCode Precond_SetUp(PC)`) is a preconditioning setup routine (see Section B.4), and

(`PetscErrorCode PBP_precondition(PC, Vec, Vec)`) is a physics-based preconditioning routine (see Section B.5).

Finally, activate all above-defined options with

```
CHKERRQ(SNESSetFromOptions(.snes));
```

## B.2 Execution stage

SNES execution is invoked in the driver for time update loop. Typically, it is called at each Runge-Kutta step, when a solution of the non-linear set of equations is required.

First, one must set convergence history for both linear and non-linear solvers, as

```
KSP ksp; CHKERRQ(SNESGetKSP(.snes, &ksp));
PetscReal *Khist=NULL, *Shist=NULL;
PetscInt *Shistit=NULL, Shistl=_HisLx_size, Khistl=_ksp_his_store;
CHKERRQ(PetscMalloc(_ksp_his_store*sizeof(PetscReal), &Khist));
CHKERRQ(PetscMalloc(Shistl*sizeof(PetscReal), &Shist));
CHKERRQ(PetscMalloc(Shistl*sizeof(PetscInt), &Shistit));
CHKERRQ(KSPSetResidualHistory(ksp, Khist, Khistl, PETSC_FALSE));
CHKERRQ(SNESSetConvergenceHistory(.snes, Shist, Shistit, Shistl, PETSC_FALSE));
```

### B.3. RESIDUAL EVALUATION ROUTINE: `Form_Function`

137

Next, SNES is executed by calling:

```
CHKERRQ(SNESSolve(_snes, PETSC_NULL, X));
```

where (`Vec X`) is a properly initiated solution vector. On the input, it must be set to hold an initial guess. On the output, it will hold a solution to non-linear problem (provided the solution procedure is properly converged).

To get convergence history, execute:

```
int NofLinearIt; CHKERRQ(SNESGetLinearSolveIterations(_snes, &NofLinearIt));
CHKERRQ(KSPGetResidualHistory(ksp, PETSC_NULL, &Khist1));
CHKERRQ(SNESGetConvergenceHistory(_snes, PETSC_NULL, PETSC_NULL, &Shist1));
```

(see [BBE<sup>+</sup>04, BBG<sup>+</sup>01] on how to process/interpret convergence history).

### B.3 Residual evaluation routine: `Form_Function`

```
PetscErrorCode Form_Function(SNES snes, Vec x, Vec f, void *ctx)
{PetscFunctionBegin;

// Extract pointer to your class for computation of residual:
Your_RES_Class *ctx_class = reinterpret_cast<Your_RES_Class*>(ctx);

// Extract a pointer to residual vector:
PetscScalar *ff; CHKERRQ(VecGetArray(f, &ff));

// Given a pointer to residual vector ff, and solution vector x, compute and return residuals:
CHKERRQ(ctx_class->Form_Function(x, ff));

CHKERRQ(VecRestoreArray(f, &ff));
PetscFunctionReturn(0);}
```

## B.4 Preconditioning setup routine: Precond\_SetUp

```
PetscErrorCode Preconditioning_SetUp(PC pc) {PetscFunctionBegin;

// Extract pointer to your class for PBP preconditioning:
Your_PBP_Class *ctx_class; PCShellGetContext(pc, (void**)&ctx_class);

// Call your preconditioning setup routines (if and whatever is needed)...

PetscFunctionReturn(0);}
```

## B.5 Preconditioning routine: PBP\_precondition

```
PetscErrorCode PBP_precondition(PC pc, Vec x, Vec y) {PetscFunctionBegin;

// Extract pointer to your class for PBP preconditioning:
Your_PBP_Class *ctx_class; PCShellGetContext(pc, (void**)&ctx_class);

// Call your preconditioning:
// Basically, given  $x \rightarrow \delta\vec{y}$  apply preconditioning on it and return the result as  $y \rightarrow \delta\vec{x}_V = \mathbb{P}^{-1}\delta\vec{y}$ ...

// For no-preconditioning:
VecCopy(x, y);

PetscFunctionReturn(0);}
```

*This Page is Intentionally Left Blank*

## Bibliography

- [AEKP00] S.P. Antal, S.M. Ettore, R.F. Kunz, and M.Z. Podowski. Development of a next generation computer code for the prediction of multi-component multiphase flows. In *International Meeting on Trends in Numerical and Physical Modeling for Industrial Multiphase Flow*, 2000.
- [Ant11] S.P. Antal. NPHASE-CMFD coupled mass/momentum solver and solution algorithm, December 16 2011. CASL Meeting.
- [Ari] R. Aris. *Vectors, Tensors, and the Basic Equations of Fluid Mechanics*. Dover Publications, Inc., New York.
- [B<sup>+</sup>92] J.A. Borkowski et al. TRAC-BF1: An advanced best-estimate computer program for BWR accident analysis. Technical Report Report EGG-2626, US NRC Report NUREG/CR-4356, Idaho National Laboratories, 1992.
- [BBE<sup>+</sup>04] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.5, Argonne National Laboratory, 2004.
- [BBG<sup>+</sup>01] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2001. <http://www.mcs.anl.gov/petsc>.
- [BCVK02] H. Bijl, M.H. Carpenter, V.N. Vatsa, and C.A. Kennedy. Implicit time integration schemes for the unsteady compressible Navier-

**BIBLIOGRAPHY****141**

- Stokes equations: Laminar flow. *Journal of Computational Physics*, 179:313–329, 2002.
- [Bes90] D. Bestion. The physical closure laws in the CATHARE code. *Nuclear Engineering and Design*, 124:229–245, 1990.
- [Bes00] D. Bestion. The phase appearance and disappearance in the CATHARE code. In *Trends in Numerical and Physical Modeling for Industrial Multiphase Flows*, Cargese, (Corse), France, September 27-29 2000.
- [BPB93] F. Barre, M. Parent, and B. Brun. Advanced numerical methods for thermalhydraulics. *Nuclear Engineering and Design*, 145:147–158, 1993.
- [BS90] P.N. Brown and Y. Saad. Hybrid Krylov methods for nonlinear systems of equations. *SIAM Journal of Science and Statistical Computing*, 11:450–480, 1990.
- [Car82] M.B. Carver. A method of limiting intermediate values of volume fractions in iterative two-fluid computations. *J. Mech. Eng. Sci.*, 24:221–224, 1982.
- [cdt] TRACE code development team. TRACE v5.0 Theory Manual: Field equations, Solution methods, and Physical models. [http://spot.infosyslabs.com:8080/nrccodes/how\\_to\\_obtain.html](http://spot.infosyslabs.com:8080/nrccodes/how_to_obtain.html).
- [cdt09] RELAP5-3D code development team. RELAP5-3D code manual Volume I: Code structure, System models and Solution methods. Technical Report Technical Report INEEL-EXT-98-00834, Revision 3.0, Idaho National Laboratories, Idaho Falls, Idaho, September 2009.
- [CKB<sup>+</sup>05] M.H. Carpenter, C.A. Kennedy, H. Bijl, S.A. Wilken, and V.N. Vatsa. Fourth-order Runge-Kutta schemes for fluid mechanics applications. *SIAM Journal of Scientific Computing*, 25:157–194, 2005.
- [CL07] C.H. Chang and M.S. Liou. A robust and accurate approach to computing compressible multiphase flow: Stratified flow model and AUSM<sup>+</sup>-up scheme. *Journal of Computational Physics*, 225:840–873, 2007.

- [Del68] J.M. Delhay. Equations of Fondamentales des écoulements Diphasiques, Part 1 and 2. Technical Report CEA-R-3429, CEA, 1968.
- [DMS01] M Darwish, F. Moukalled, and B. Sekar. A unified formulation of the segregated class of algorithms for multifluid flow at all speeds. *Numerical Heat Transfer, Part B*, 40:99–137, 2001.
- [DP99] D.A. Drew and S.L. Passman. *Theory of Multicomponent Fluids*, volume 135 of *Applied Mathematical Sciences*. Springer, New York/Berlin/Heidelberg, 1999.
- [EW96] S.C. Eisenstat and H.F. Walker. Choosing the forcing terms in an inexact Newton method. *SIAM Journal of Science and Statistical Computing*, 17:16–32, 1996.
- [Got05] S. Gottlieb. On high order strong stability preserving Runge-Kutta and multi step discretizations. *SIAM Journal of Scientific Computing*, 25(1/2):105–128, 2005.
- [HA68] F. H. Harlow and A. A. Amsden. Numerical calculation of almost incompressible flow. *Journal of Computational Physics*, 3:80–93, 1968.
- [HA71] F. H. Harlow and A. A. Amsden. A numerical fluid dynamics calculation method for all flow speeds. *Journal of Computational Physics*, 8:197–213, 1971.
- [HA75a] F. H. Harlow and A. A. Amsden. Flow of interpenetrating material phases. *Journal of Computational Physics*, 18:440–464, 1975.
- [HA75b] F. H. Harlow and A. A. Amsden. Numerical calculation of multi-phase fluid flow. *Journal of Computational Physics*, 17:19–52, 1975.
- [HB77] W.T. Hancox and S. Banerjee. Numerical standards for flow boiling analysis. *Nuclear Sci. Eng.*, 64:106, 1977.
- [IH06] M. Ishii and T. Hibiki. *Thermo-fluid Dynamics of Two-Phase Flow*. Springer, 2006.

**BIBLIOGRAPHY****143**

- [KCMM03] D.A. Knoll, L. Chacon, L.G. Margolin, and V.A. Mousseau. On balanced approximations for time integration of multiple time scales systems. *Journal of Computational Physics*, 185:583–611, 2003.
- [KK04] D. A. Knoll and D. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *Journal of Computational Physics*, 193:357–397, 2004.
- [KMCR05] D.A. Knoll, V.A. Mousseau, L. Chacon, and J. M. Reisner. Jacobian-free Newton-Krylov methods for the accurate time integration of stiff wave systems. *SIAM Journal of Scientific Computing*, 25:213–230, 2005.
- [KMK96] D.A. Knoll, P. R. McHugh, and D. E. Keyes. Newton-Krylov methods for low-Mach-number compressible combustion. *AIAA Journal*, 34(5):961, 1996.
- [KR00] D. A. Knoll and W. J. Rider. A multigrid preconditioned Newton-Krylov method. *SIAM Journal of Scientific Computing*, 21:691–710, 2000.
- [Lio06] M. Liou. A sequel to AUSM, Part II: AUSM+-up for all speeds. *Journal of Computational Physics*, 214:137–170, 2006.
- [Lo90] S.M. Lo. Multiphase flow model in Harwell-FLOW#D computer code. Technical Report AEA-InTec-0062, AEA Report, 1990.
- [LW78] D.R. Liles and Reed Wm.H. A Semi-Implicit method for two-phase fluid dynamics. *Journal of Computational Physics*, 26:390–407, 1978.
- [Mah82] J.H. Mahaffy. A Stability-Enhancing Two-Step method for fluid flow calculations. *Journal of Computational Physics*, 46:329–341, 1982.
- [Mah93] J.H. Mahaffy. Numerics of codes: stability, diffusion, and convergence. *Nuclear Engineering and Design*, 145:131–145, 1993.
- [McF81] J.H. McFadden. RETRAN-02: User’s manual. Technical report, Electric Power Institute, 1981.

- [MD04] F. Moukalled and M. Darwish. Pressure-based algorithms for multifluid flow at all speeds - part i: Mass conservation formulation. *Numerical Heat Transfer, Part B*, 45:495–522, 2004.
- [MDS03] F. Moukalled, M. Darwish, and B. Sekar. Pressure-based algorithm for multi-phase flow at all speeds. *Journal of Computational Physics*, 190:550–571, 2003.
- [Mes98] G.L. Mesina. Border-Profile LU solver for RELAP5-3D. In *1998 RELAP5 International Users Seminar*, College Station, Texas, May 17-21 1998.
- [NBDY11] R.R. Nourgaliev, V.A. Bui, N.D. Dinh, and R. Youngblood. Summary report of NGSAC (Next Generation Safety Analysis Code) development and testing. Technical report, Idaho National Laboratory, Idaho Falls, USA, September 28 2011.
- [NDY10] R.R. Nourgaliev, N.D. Dinh, and R. Youngblood. Development, selection, implementation and testing of architectural features and solution techniques for Next Generation of System Simulation Codes to support safety case of the LWR life extension. Technical Report INL/EXT-10-19984, Idaho National Laboratory, Idaho Falls, USA, 2010.
- [Nig90] R.I. Nigmatulin. *Dynamics of Multiphase Media*, volume 1. Taylor & Francis, 1990.
- [Pat80] S. Patankar. *Numerical Heat Transfer and Fluid Flow*. Taylor & Francis, 1980.
- [PCGC03] H. Paillère, C. Corre, and J.R. García Cascales. On the extension of the AUSM+ scheme to compressible two-fluid models. *Computers & Fluids*, 32:891–916, 2003.
- [PS72] S.V. Patankar and D.B. Spalding. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. *International Journal of Heat Mass Transfer*, 15:1787–1806, 1972.
- [PW98] M. Pernice and H.F. Walker. NITSOL: A Newton iterative solver for nonlinear systems. *SIAM Journal of Science and Statistical Computing*, 19:302–318, 1998.

**BIBLIOGRAPHY****145**

- [Ran87] V. Ransom. Numerical benchmark tests. *Multiphase Science and Technology*, 3(1-4):465–467, 1987.
- [Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2nd edition, 2003.
- [SO89] C.-W. Shu and S. Osher. Efficient implementation of essentially non-oscillatory shock-capturing schemes. *Journal of Computational Physics*, 77:439–471, 1989.
- [Spa76] D.B. Spalding. The calculation of free-convection phenomena in gas-liquid mixtures. Technical Report Rep. HTS/76/11, Imperial College, London, U.K., 1976.
- [Spa80] D.B.. Spalding. Numerical computation of multi-phase fluid flow and heat transfer. In C. Taylor and K. Morgan, editors, *Recent Advances in Numerical Methods in Fluid*, volume 1, pages 139–167, 1980.
- [Spa81] D.B. Spalding. A general purpose computer program for multi-dimensional one- and two-phase flow. *Mathematics and Computers in Simulation XXIII*, pages 267–276, 1981.
- [Spa83] D.B.. Spalding. Development in the IPSA procedure for numerical computation of multiphase-flow phenomena with interfacial slip, unequal temperatures. In T.M. Shih, editor, *Numerical Methodologies in Heat Transfer, Proc. Second National Symposium*, pages 421–436. Hemisphere, 1983.
- [SS86] Y. Saad and M.H. Schultz. GMRES: a Generalized Minimal Residual algorithm for solving linear systems. *SIAM Journal of Science and Statistical Computing*, 7:856, 1986.
- [Sta01] N. Stadtke. Advanced modeling and numerical strategies in nuclear thermal-hydraulics. In *International Conference Nuclear Energy in Central Europe 2001*, September 10-13 2001.
- [Sta06] H. Stadtke. *Gasdynamic Aspects of two-phase flow. Hyperbolicity, Wave Propagation Phenomena and Related Numerical Methods*. Wiley-VCH, 2006.

- [Ste81] H.B. Stewart. Fractional step methods for thermalhydraulics calculation. *Journal of Computational Physics*, 40:77–90, 1981.
- [Stu77] J.H. Stuhmiller. The influence of interfacial pressure forces on the character of two-phase flow model equations. *International Journal of Multiphase Flow*, 3:551–560, 1977.
- [Tor99] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics. A practical Introduction*. Springer, Berlin/Heidelberg, 2nd edition, 1999.
- [TR86] 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.
- [Yan71] N.N. Yanenko. *The Method of Fractional Steps*. Springer-Verlag, New York-Heidelberg-Berlin, 1971.
- [YNK<sup>+</sup>11] R. Youngblood, R. Nourgaliev, D. Kelly, C. Smith, and T.-N. Dinh. Framework for applying a Next-Generation Safety Analysis Code to plant life extension decision-making. In *ANS Proceedings of the 2011 ANS Summer Meeting*, June 2011.

*This Page is Intentionally Left Blank*

# Index

- A-stability, 105
- acoustic pressure wave, 25
- acoustic pressure waves, 22
- acoustic time scale, 20
- acoustically-filtered formulation, 83
- Added mass, 13
- added mass coefficient, 16
- added mass forces, 16
- All-speed, 24
- Arnoldi basis vectors, 112
- AUSM, 123
- Averaging, 6
  
- Backward Difference, 105
- Backward Euler, 105
- balance equations, 6
- BDF, 105
- boiling, 119
- BPLU, 30
- Bulk pressure difference, 13, 15
- Butcher tableau, 105
  
- CASL, 2
- CATHARE, 16, 24, 26, 34, 110, 118
- CFL, 24
- closure, 2, 6
- Closures, 9
- collocated mesh, 24
- compatibility condition, 29
- compatibility conditions, 11
- compatibility equation, 8, 14, 60
- compressibility effects, 24
- condensation, 119
- conservation variables, 14
- conservative flux, 16
- constitutive physics, 2, 6, 7, 9
- continuum phase, 16
- Crank-Nicholson, 105
- cut-off, 124
  
- degree of freedom, 49
- Diagonal stabilizer, 51
- Diagonalization, 48
- Discontinuous Galerkin, 24, 49
- dispersed phase, 16
- donor-cell, 24
- drift-flux, 24, 30
- dynamic interfacial pressure, 42
- dynamic pressure coefficient, 16
- dynamic time scale, 20
  
- Ensemble-Averaging, 6
- Entropy inequality, 12
- equation of state, 8, 16, 45
- Equation of state, 7
- ESDIRK, 105, 106
- evaporization, 119
- expanded form, 17
- explicit discretization, 20
- Explicit, Singly Diagonal Implicit Runge-Kutta, 106
- Fevré averaged, 8

- finite-element, 49  
 Fréchet derivatives, 112  
 fractional step method, 24, 34  
 friction coefficient matrix, 50  
 fully-implicit algorithm, 16  
  
 Galerkin, 112  
 Gauss-Seidel, 84, 87  
 GCBA, 96  
 Generalized Minimal RESidual method, 112  
 global mass conservation equation, 86  
 GMRES, 111, 112  
  
 Helmholtz equation, 30  
 Hydra, 2  
 hyperbolic conservation laws, 20  
 hyperbolicity, 16  
  
 ICE, 24, 28, 86, 92  
 ICE linearization, 45  
 ILU, 33, 110  
 Implicit Continuous-fluid Eulerian, 24  
 implicit Runge-Kutta, 104  
 Implicit stabilizer, 51  
 Incremental form, 41  
 inexact Newton method, 111  
 inter-penetrating continua, 7  
 Inter-Phase Slip Algorithm, 84  
 Interfacial dynamic pressure, 13  
 interfacial dynamic pressure, 15  
 interfacial exchange terms, 22  
 interfacial forces, 15  
 IPSA, 84  
 IRK, 104  
  
 Jacobian, 33  
 Jacobian-free, 111, 112  
 Jacobian-Free Newton-Krylov, 111  
 JFNK, 16, 33, 110, 111  
 jump conditions, 11  
  
 Krylov, 111  
 Krylov subspace, 112  
 Krylov subspace iterations, 112  
  
 L-stability, 105  
 Laplacian, 29  
  
 mass-energy matrix, 29  
 mass-energy wavenumber matrix, 53, 57  
 mass-weighted, 8  
 material CFL, 24  
 material CFL stability limit, 25  
 mathematical-based preconditioners, 33  
 MCBA, 86  
  
 NBGS, 30  
 Nearly-Implicit, 24, 34, 37, 87  
 Newton Block Gauss Seidel, 30  
 Newton iterations, 42, 104  
 Newton method, 106  
 Newton's method, 22  
 non-conservative flux, 16  
 non-symmetric linear systems, 112  
 noncondensable gases, 33  
 NPHASE, 102  
  
 operator-splitting, 42, 50  
  
 P'HE, 51, 60  
 Partial Elimination Algorithm, 88  
 PEA, 88  
 phase appearance and disappearance, 17, 22, 33, 118  
 phase appearance and disappearance, 4  
 phase change, 119  
 phasic bulk pressure differences, 42  
 Phasic density, 8

- Phasic fluctuation (Reynolds) kinetic energy, 8  
 Phasic pressure, 8  
 Phasic specific internal energy, 8  
 Phasic total energy, 8  
 Phasic velocity, 8  
 PHE, 28  
 physics-based preconditioning, 33, 41, 111  
 Picard, 87, 88  
 Picard iteration, 84  
 Picard iterations, 42  
 PISO, 92  
 Pressure correction P'HE, 60  
 pressure-correction Helmholtz equation, 51  
 Pressure-Helmholtz equation, 28  
 pressure-Helmholtz equation, 38  
 PRIME, 99  
 Primitive variables, 21  
 primitive variables, 14  
  
 RELAP, 17, 24, 34  
 RELAP-3D, 16, 17  
 RELAP5, 26, 37, 38  
 RELAP5-3D, 24, 32  
 relative velocity, 16  
 residual vector, 107  
 RETRAN, 24  
 Riemann solvers, 24  
 Runge-Kutta, 20  
  
 segregated algorithms, 84  
 semi-implicit, 24, 25, 32, 34, 41  
 semi-implicit algorithm, 45  
 SETS, 24, 34, 37, 39, 45, 51  
 SIMPLE, 41, 45, 50, 84, 92, 99  
 SIMPLEC, 99  
 SIMPLEM, 99  
 SIMPLER, 92, 99  
 SIMPLEST, 92  
 SIMPLEX, 92  
 SINCE, 89  
 single-pressure 6-equation two-fluid model, 16  
 stability limit, 20  
 Stabilizer, 39  
 stabilizer, 34  
 staggered mesh, 24  
 Strong-Stability-Preserving, 20  
  
 thermal conditioning, 119  
 Total Variation Diminishing, 20  
 TRAC, 16, 24, 26, 34, 39  
 TRAC-BF1, 16  
 TRACE, 16, 24, 26, 34, 39  
 Trial subspace, 112  
 Two-fluid equations, 13  
  
 unexpanded form, 17  
  
 Velocity stabilizer, 51  
 velocity-Helmholtz equation, 38  
 virtual mass, 15  
 virtual mass matrix, 16  
 Void fraction bounding, 119  
 Volume fraction, 7  
  
 weakly-compressible, 24

*This Page is Intentionally Left Blank*