Nuclear Engineering and Design 379 (2021) 111178

Contents lists available at ScienceDirect

Nuclear Engineering and Design

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

Development of a novel two-phase flow solver for nuclear reactor analysis: algorithms, verification and implementation in OpenFOAM

Stefan Radman [a,*], Carlo Fiorina [a], Andreas Pautz [a,b]

[a] École Polytechnique Fédérale de Lausanne, Laboratory for Reactor Physics and Systems Behaviour, 1015 Lausanne, Switzerland
[b] Paul Scherrer Institut, Nuclear Energy and Safety Department, 5232 Villigen, Switzerland

A B S T R A C T

A porous medium-based representation of nuclear reactors and complex engineering systems more in general can significantly reduce simulation and modelling costs, while preserving a reasonable degree of accuracy via regime map-based correlations for modelling physical interactions. This paper presents a segregated algorithm for the simulation of dispersed two phase flows in such systems treated as porous media in an Eulerian framework. The global algorithm pertaining to the coupling between the mass, momentum and energy conservation equations solution is discussed and implemented via the finite volume OpenFOAM programming library. In the context of pressure–velocity coupling, a novel implementation of the Partial Elimination Algorithm for the treatment of the inter-phase momentum transfer term is developed. It is found to perform better than existing implementations for a number of cases with important momentum coupling between phases. A conclusive verification of the overall solution algorithm is performed with the Method of Manufactured Solutions and order-of-accuracy testing. From an implementation perspective, the performance of the algorithm in parallel strong scaling up to 4096 cores is assessed and proves to be in line with OpenFOAM-based code standards.

1. Introduction

The simulation of multi-phase fluid and heat dynamics in complex engineering systems often results in a trade-off between complexity in the physics and in the geometric modelling. With an increasingly detailed modelling of the system geometry, the resolution of boundary layers and phase interfaces allows the estimation of momentum, heat and mass transfer via first principles. However, this comes at the cost of significant computational burdens. Conversely, any major simplification to the geometric representation needs to be compensated by an adequate choice of models for the aforementioned momentum, heat and mass transfer, generally in the form of system or fluid-specific experimental correlations. Notwithstanding the steady increase of available computational resources to the scientific community, the possibility of exploiting geometrical simplifications by taking advantage of established experimental correlations remains of interest for complex engineering systems modelling. In this context, the porous medium approximation based on volume averaging techniques (Whitaker, 1985) has enjoyed a wealth of applications for Navier–Stokes flows both in single-phase and multi-phase scenarios, thus outside its traditional domains of Darcy and Brinkman-Forchheimer flows in microscopically porous media (Kozelkov et al., 2018). These applications range from flows through internal combustion engine silencers (Montenegro et al., 2013), fibrous materials and rod arrays (Penha et al., 2011), media obstructed by macroscopic debris (Colas et al., 2019), gas turbines (Ford et al., 2013), boilers (Hovi et al., 2016), condensers and heat exchangers in general (Kim et al., 2019).

In the nuclear field, several codes take advantage of a porous-medium approach, such as BACCHUS-3D (Bottoni et al., 1987), TWO-POROFLOW (Chavez et al., 2018) and, more recently, various OpenFOAM-based (Weller et al., 1998; OpenCFD, 2021) single-phase solvers such as GeN-Foam (Fiorina et al., 2015) and other reactor-specific solvers (Clifford, 2013), with applications to fuel elements (Radman et al., 2019; Yu et al., 2015; Sarkar et al., 2018) as well as whole nuclear reactor cores (Vanderhaegen et al., 2011; Sipaun and Usman, 2015; Abbassi et al., 2016). One may notice also that the widely employed sub-channel codes can be considered as a non-isotropic declination of the porous-medium approach. Porous-medium (and sub-channel) approaches are normally employed to achieve higher accuracy with respect to legacy 1-D system codes. Furthermore, the porous-medium approach presents some advantages that fit well the growing interest towards high-fidelity multi-physics applications. A first advantage is that the porous-medium equations revert back to standard Navier–Stokes RANS equations in clear-fluid (i.e. non-porous) regions. This allows for a seamless integration of porous-medium and fine-mesh RANS approaches, which in turns allows for implicitly coupled simulations e.g. of core and plena in Light Water Reactors (LWRs), or core and pools in Sodium Fast Reactors (SFRs). In addition to this, the porous medium approach can be developed based on common unstructured

* Corresponding author.
E-mail address: stefan.radman@epfl.ch (S. Radman).

https://doi.org/10.1016/j.nucengdes.2021.111178
Received 19 August 2020; Received in revised form 13 December 2020; Accepted 5 March 2021
Available online 12 April 2021

Nomenclature

Nomenclature

α	Volumetric phase fraction $(-)$
δ	Characteristic mesh cell dimension (m)
ϵ	Numerical error $(\cdot)$
ε	Specific turbulent kinetic energy dissipation rate $(\frac{m^2}{s^3})$
Γ	Volumetric mass source/sink $(\frac{kg}{m^3 s})$
κ	Thermal conductivity $(\frac{W}{m\,K})$
μ	Molecular viscosity $(Pa\,s)$
ν	Kinematic diffusivity $(\frac{m^2}{s})$
ρ	Density $(\frac{kg}{m^3})$
$\boldsymbol{\tau}$	Structure tortuosity tensor when modelled by porous medium $(-)$
ϕ	Superficial flux $(\frac{m^3}{s})$
ϕ_α	Volumetric flux $(\frac{m^3}{s})$
ξ	Continuity error $(\frac{kg}{m^3 s})$
Ω	Computational domain
a	Thermal diffusivity $(\frac{m^2}{s})$
D_h	Hydraulic diameter or phase characteristic dimension (m)
e_p	Parallel efficiency $(-)$
g	Gravitational acceleration $(\frac{m}{s^2})$
h	Enthalpy (J)
H	Heat transfer coefficient $(\frac{W}{m^2 K})$
i,j	Phase indices, either 1 or 2
k	Specific turbulent kinetic energy $(\frac{m^2}{s^2})$
K, **K**	Dimensional drag factor, scalar for fluid–fluid drag, tensor for fluid–structure drag $(\frac{kg}{m^3 s})$
p	Pressure (Pa)
Q	Volumetric heat flux $(\frac{W}{m3})$
T	Temperature (K)
u	Velocity $(\frac{m}{s})$
V	Volume (m^3)
V_m	Virtual mass coefficient $(-)$

mesh formats. This allows for a straightforward integration in multi-physics applications via the use of standard algorithms for mesh-to-mesh projection, without the need of developing code-specific and error-prone message-passaing interfaces.

This paper presents a novel segregated algorithm for the treatment of dispersed two-phase flows in nuclear reactors or other complex engineering systems that can be modelled via a porous-medium approach. The algorithm allows modelling all essential components of porous-medium flows, including: 1) modelling of structure anisotropy via tortuosity tensors; 2) tensorial approach to modelling of fluid–structure drag; 3) fluid-coupled models for the treatment of structure heat sources; 4) flexible flow regime map representation of transfer models (i.e. drag, heat, mass transfer). The algorithm has been implemented based on the OpenFOAM C++ numerical library (Jasak, 2009; Weller et al., 1998; OpenCFD, 2021) in the context of the activities carried out at the EPFL/LRS and aimed at the development of an OpenFOAM-based platform for nuclear reactor analysis (Fiorina et al., 2016; Fiorina et al., 2017; Scolaro et al., 2020; Fiorina, 2019). It will be used to complement the GeN-Foam solver with two-phase flow capabilities, aiming at the multi-physics analysis of BWRs, accidental transients in SFRs, and helium bubbling in Molten Salt Reactors.

In particular, the focus of the present paper is on: 1) description of the algorithms for the solution of the continuity, momentum and energy conservation equations, with special regard to a novel implementation of a Partial Elimination Algorithm (PEA) (Spalding, 1980) for the solution of the pressure–velocity coupling; 2) verification of the numerics via the Method of Manufactured Solutions (MMS) (Salari and Knupp, 2000); 3) assessment of the performance of the PEA against alternative solution approaches.

The paper is organized as follows. Section 2 provides a description of the code algorithms and numerics, inclusive of their derivation. Section 3 deals with code verification by the Method of Manufactured Solutions. In Section 4, the performance of the novel Partial Elimination Algorithm is assessed. This is performed for a variety of cases, ranging from a 1-D sodium boiling case to a standard two-phase OpenFOAM test case, namely a 2-D problem concerning bubble injection in a water column. In Section 5, the strong scaling of the solver up to 4096 processor cores is assessed. In Section 6, conclusions are drawn and future work perspectives are provided.

2. Modelling, algorithms and numerics

We employ a dispersed Eulerian formalism for the segregated mathematical description of two-phase flows with shared pressure in porous media. By Euler-Euler we mean that both phases[1] are described in an Eulerian frame of reference, which translates into a mathematical description of phase fields as inter-penetrating continua. These phases are tracked via their volume fractions, with each phase being governed by its own set of equations for mass, momentum and energy conservation. By dispersed we mean that the physical interface between the two fluids is not explicitly tracked nor resolved by the mesh. The choice of a dispersed framework for the interface treatment is consistent with the adoption of a porous medium approach for the fluid–structure interface. By segregated we mean that each set of equations is solved independently via the construction of individual linear systems, whose coupling is resolved via iterations. The choice of a segregated algorithm was taken due to the ease of implementation, smaller memory requirements, easier matrix preconditioning, the possibility to selectively resolve non-converged physics, and the possibility of sub-cycling, namely the use of different time steps for the different physics, according to their numerical requirements. The implemented conservation equations are discussed in subSection 2.1 while their solution algorithms, inclusive of the novel implementation of the Partial Elimination Algorithm, are presented in subSection 2.2.

2.1. Main equations

2.1.1. Continuity equation

The general form of the compressible continuity equation for the i-th phase is:

$$\frac{\partial}{\partial t}(\alpha_i \rho_i) + \nabla\cdot(\alpha_i \rho_i \mathbf{u}_i) = \Gamma_i \tag{2.1}$$

Compressibility effects due to density changes can be isolated on the left-hand-side by expanding the derivative terms:

$$\frac{\partial}{\partial t}\alpha_i + \nabla\cdot(\alpha_i \mathbf{u}_i) = \frac{\Gamma_i}{\rho_i} + C_i(\rho_i(p)) \tag{2.2}$$

$$C_i(\rho(p)) = -\frac{1}{\rho_i}\left(\left(\frac{\partial}{\partial t}\rho_i\right)\alpha_i + \nabla\rho_i\cdot(\alpha_i \mathbf{u}_i)\right) \tag{2.3}$$

The compressible term $C_i(\rho(p))$ is evaluated explicitly and the phase fraction equations is solved via the Multidimensional Universal Limiter

[1] As a remark, the terms "fluid" and "phase" will be used interchangeably.

with Explicit Solution (MULES) algorithm (Weller, 2008), which is a particular instance of a Flux Corrected Transport (FCT) technique (Boris and Book, 1997). The overall idea behind FCT and MULES is outlined in subSection 2.2.1.

2.1.2. Momentum conservation equations

The general form of the momentum conservation equation for the i-th phase is:

$$\frac{\partial}{\partial t}(\alpha_i\rho_i\mathbf{u}_i)+\nabla\cdot(\alpha_i\rho_i\mathbf{u}_i\otimes\mathbf{u}_i)=-\alpha_i\nabla p+\nabla\cdot(\alpha_i\rho_i\nu_i\boldsymbol{\tau}_i\cdot(\nabla\mathbf{u}_i+(\nabla\mathbf{u}_i)^T-\frac{2}{3}(\nabla\cdot\mathbf{u}_i)\mathbf{I}))+\mathbf{S}_{\mathbf{u},i} \tag{2.4}$$

in which the second term on the left hand side is the divergence of the deviatoric stress tensor treated with the Stokes hypothesis. The porous structure tortuosity tensor figures as $\boldsymbol{\tau}$, and, for ease of visualization, one can think of the whole effective kinematic viscosity term being modelled no longer as scalar but as a tensor in the form $\nu_i\boldsymbol{\tau}$. Other source terms have been grouped in $\mathbf{S}_{\mathbf{u},i}$:

$$\mathbf{S}_{\mathbf{u},i}=K_{ij}(\mathbf{u}_j-\mathbf{u}_i)-\mathbf{K}_{is}\cdot\mathbf{u}_i+V_m\rho_c(\frac{\partial}{\partial t}\mathbf{u}_i+\mathbf{u}_i\cdot\nabla\mathbf{u}_\mathbf{i}-\frac{\partial}{\partial t}\mathbf{u}_j)+\Gamma_{j\to i}\mathbf{u}_j-\Gamma_{i\to j}\mathbf{u}_i+\alpha_i\rho_i\mathbf{g}+\xi_i\mathbf{u}_i \tag{2.5}$$

which are, respectively, the fluid–fluid drag, fluid–structure drag, virtual mass force, momentum source due to mass transfer from phase j to i and sink due to mass transfer from phase i to j, the gravitational term, and a purely numerical correction that accounts for continuity errors. The fluid–fluid drag factor is scalar by nature while the fluid–structure drag factor is a tensor that accounts for the potential anisotropy of the structure. The fluid–fluid drag and its numerical treatment via the Partial Elimination Algorithm in the pressure–velocity coupling procedure will be discussed more in detail in sub-Section 2.2.2. The fluid–structure drag is treated semi-implicitly as:

$$\mathbf{K}_{is}\cdot\mathbf{u}_i=\frac{1}{3}tr(\mathbf{K}_{is})\mathbf{u}_i+\left(\mathbf{K}_{is}-\frac{1}{3}tr(\mathbf{K}_{is})\mathbf{I}\right)\cdot\mathbf{u}_i$$

where $tr(\cdot)$ denotes the matrix trace operator. The fluid–structure drag coefficients $\mathbf{K}_{is}$ are computed starting from a single total structure drag coefficient $\mathbf{K}_s$. In turn, this is calculated starting from the single phase drag coefficient of the continuous phase and the well know two-phase drag multiplier method. The actual multiplier and drag coefficient models are selectable at run time and will not be discussed for the remainder of this work. Lastly, by the continuity error ξ we mean the numerical evaluation of Eq. (2.1) so that:

$$\xi_i=\frac{\partial}{\partial t}(\alpha_i\rho_i)+\nabla\cdot(\alpha_i\rho_i\mathbf{u}_i)-\Gamma_i$$

which should physically be null at all times. These continuity errors are a consequence of the segregated nature of our algorithm. Once evaluated, they can be used explicitly to correct Eqs. (2.1), (2.4), (2.6) which was also observed to help stabilizing the solution.

2.1.3. Energy conservation equation

The energy conservation equation for each phase is formulated and solved in terms of its enthalpy h_i with an explicit contribution from specific kinetic energy $e_{K,i}=\frac{1}{2}\mathbf{u}_i\cdot\mathbf{u}_i$ as:

$$\frac{\partial}{\partial t}(\alpha_i\rho_i(h_i+e_{K,i}))+\nabla\cdot(\alpha_i\rho_i\mathbf{u}_i(h_i+e_{K,i}))-\nabla\cdot(\alpha_i a_i\boldsymbol{\tau}\cdot\nabla h_i)=-\alpha_i\frac{\partial}{\partial t}p+\alpha_i\rho_i\mathbf{u}_i\cdot\mathbf{g}+S_{i\delta}+S_s+S_{i,\Gamma}+\xi_i(h_i+e_{K,i}) \tag{2.6}$$

The source terms on the left-hand side consist of the pressure-work term, gravitational work, interfacial heat transfer, source due to structure heat sources, mass transfer and a numerical correction term to account for numerical deviations from mass conservation, as seen when discussing the momentum equation.

The interfacial fluid heat transfer $S_{i\delta}$ is treated via a two-resistance model, which considers that the bulk of each phase does not exchange heat directly with the bulk of the other phase, but with the fluid–fluid interface. This source contribution can be modelled as $S_{i\delta}=H_{i\delta}A_\delta'''(T_\delta-T_i)$ with $H_{i\delta}$ being the bulk-interface heat transfer coefficient for phase i, A_δ''' the interfacial area density and T_δ the interfacial temperature. It is clear that an interfacial source term expressed in terms of a temperature difference cannot be treated implicitly when solving for enthalpy. However, by linearizing enthalpy around the latest available phase enthalpy h_i^* corresponding to T_i^*, the source can be re-written as:

$$S_{i\delta}=H_{i\delta}A_\delta'''\left(\frac{1}{C_{p,i}}(h_i^*-h_i)+T_\delta-T_i^*\right) \tag{2.7}$$

In general, all heat sources are treated semi-implicitly via enthalpy linearization as it increases the diagonal dominance of the resulting discretized energy conservation matrix. The interfacial temperature T_δ is modelled in different ways depending of whether a phase change model is used or not. If no phase change model is used, no mass transfer can occur between the phases and the temperature is set so to ensure heat conservation across the interface:

$$T_\delta=\frac{H_{1\delta}T_1+H_{2\delta}T_2}{H_{1\delta}+H_{2\delta}} \tag{2.8}$$

If a phase change model is used and mass transfer can occur, the interfacial temperature is set to the saturation temperature by a saturation model. The assumption that the interface is always at saturation is relatively common in dispersed multiphase flows (Saurel et al., 2016). The fluid–structure heat source is treated in a similar way:

$$S_{is}=H_{is}f_iA_s'''\left(\frac{1}{C_{p,i}}(h_i^*-h_i)+T_s-T_i^*\right) \tag{2.9}$$

which is formally identical to (2.7), except that A_s''' is the volumetric area density of the structure and f_i is a regime-dependent factor that quantifies how the fluid–structure contact area is split between the phases, with $f_1+f_2=1$. This factor is introduced to enable the description of transitions in flow geometry e.g. from annular, in which only one fluid contacts the structure and thus exchanges heat with it, to inverted-annular, in which the fluids switch roles. The structure surface temperature T_s is updated via dedicated models before the solution of the energy equation. The last source term $S_{i,\Gamma}$ is the enthalpy transfer associated with mass transfer between the two phases. Currently it is computed via a heat conduction limited model (U.S. Nuclear Regulatory Commission, 2008), namely:

$$\Gamma_i=-\Gamma_j=\frac{Q_{j\to\delta}+Q_{i\to\delta}}{h_{sat,j}-h_{sat,i}} \tag{2.10}$$

with $Q_{i\to\delta}$ being the volumetric heat flux from phase i to the interface and $h_{s,i}$ the saturation enthalpy of phase i. In essence, this formulation computes mass transfer so to ensure energy conservation at the interface. Defining $\Gamma_{i,+}(\mathbf{r}) = max\left(\Gamma_i(\mathbf{r}), 0\right) \forall \mathbf{r}$ as the mass source contribution, and $\Gamma_{i,-}(\mathbf{r}) = min\left(\Gamma_i(\mathbf{r}), 0\right) \forall \mathbf{r}$ as the mass sink contribution, the enthalpy source due to mass change is modelled as:

$$S_{i,\Gamma} = \Gamma_{i,+}\, h_{sat,i} + \Gamma_{i,-}\, h_i \tag{2.11}$$

with $h_{sat,i}$ being the enthalpy evaluated at the saturation temperature. In qualitative terms, wherever mass transfer acts as a mass source for phase i, enthalpy is added at saturation. However, wherever mass transfer acts as a sink, enthalpy is removed at the current fluid enthalpy, rather than saturation enthalpy. The reason for this is that phase transfer models generally depend on super-heat or under-cooling of the phase to drive mass transfer. Thus, if mass is being removed from one phase at an enthalpy (e.g. the saturation one) different than its current enthalpy, the remaining fluid might be left a temperature different than the saturation one, and a thermal run-away is possible. These phenomena were highlighted during the development of some computer codes for nuclear reactor analysis, e.g. the TRACE code of the U.S. Nuclear Regulatory Commission (U.S. Nuclear Regulatory Commission, 2008).

2.2. Solution algorithm

The solution strategy we have adopted is presented in Fig. 1. It is based on a merged PISO-SIMPLE (Issa et al., 1986; Patankar and Spalding, 1972) algorithm (PIMPLE). Within each time step, a certain number of outer iterations (logically equivalent to SIMPLE iterations, even though they are more generally referred to by the global algorithm name, i.e. PIMPLE iterations) can be performed to resolve the coupling between velocity, pressure and energy. In particular, each outer iteration consists of the following steps:

1. update the flow regime map. In this context, each flow regime acts as a place-holder for different models used to compute the inter-phase transfer coefficients. At this step, the spatial extent of every regime is assessed based on the flow regime map;
2. update inter-phase transfer coefficient. These consist of: fluid–fluid drag factor K_{ij}, fluid–structure drag factors $\mathbf{K_{is}}$, fluid bulk-interface heat transfer coefficients $H_{i\delta}$, fluid–structure heat transfer coefficients H_{is}, fluid–fluid interfacial area density $A^{'''}_{\delta}$, fluid–structure contact fractions f_i and characteristic fluid dimensions $D_{h,i}$;. This update is performed based on the updated spatial extent of the regimes;
3. the MULES algorithm is used to compute the new phase fractions α_i and volumetric fluxes $\phi_{\alpha,i}$ starting from the existing face-centered superficial fluxes ϕ_i and the latest available evaluation of compressibility effects quantified by C_i (as defined by (2.3)). The algorithm can be iterated multiple times within each outer iteration to allow for sub-cycling. Sub-cycling consists in solving the phase fraction equation a number N of times over a time step N times smaller than the global time-step, which allows to ease the time step

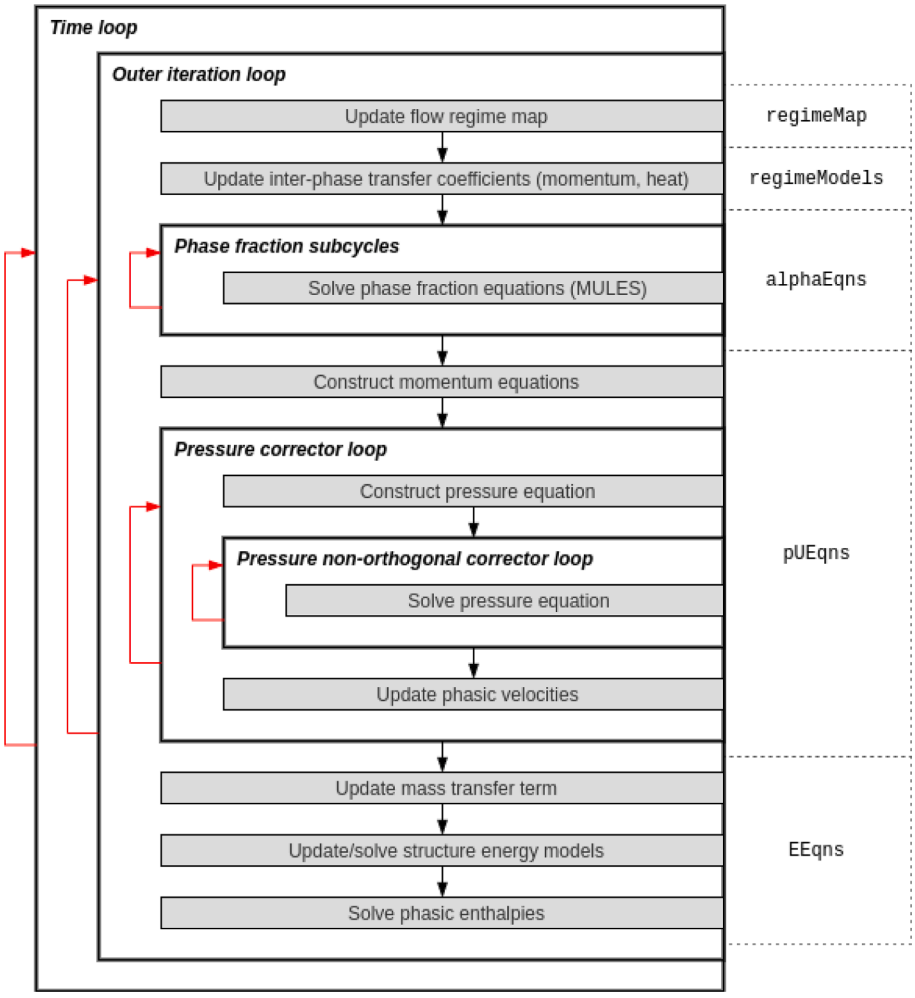

Fig. 1. Flowchart of the solution algorithm. Red arrows indicate that a section (outlined in bold black) consists of a loop (governed by certain conditions). On the rightmost side, the different flowchart steps are logically grouped into what we refer to as "components".

limitations imposed by the CFL condition. The MULES algorithm is discussed in subSection 2.2.1;
4. construct the momentum equations, as described in greater detail in subSection 2.2.2;
5. pressure corrector loop. The solution of the pressure equation is iterated to resolve the pressure–velocity coupling non-linearity and other momentum source terms. This is logically equivalent to the PISO loop. Potential pressure gradient non-orthogonality is treated via an OpenFOAM-standard non-orthogonal nested corrector loop. Velocities $\mathbf{u}_i$ and superficial fluxes ϕ_i are updated at the end of each pressure correction iteration, as well as the compressibility contributions C_i;
6. update mass transfer term $\Gamma_i = -\Gamma_j$. This is inclusive of the saturation model update and fluid–fluid interfacial temperature $T_{i\delta}$ update;
7. update/solve structure energy models. Structure surface temperatures T_s are updated according to the specified structure modelling options;
8. solve for phasic enthalpies h_i and update fluid temperatures T_i.

2.2.1. Phase fraction equations and Flux Corrected Transport

Flux Correct Transport (FCT) was originally devised as a method to solve the continuity equations using a hybrid discretization of the advective term: less diffusive than a low-order scheme (e.g. upwind) yet not resulting in a potentially unbounded phase fraction that a higher-order scheme might cause. This is particularly relevant close to steep phase fraction gradients, thus to porous structure-clear fluid interfaces in a porous medium context.

Let us consider a finite-volume discretization of the phase fraction Eq. (2.2). Subsequent to volume integration and the application of the Gauss–Green theorem, it can be re-arranged as:

$$\frac{\partial}{\partial t}\alpha_i + \frac{1}{V}\sum_{cf}\phi_{\alpha,i,cf} = \frac{\Gamma_i}{\rho_i} + C_i \tag{2.12}$$

where the second term on the left hand side is the advection contribution, which can be explicitly evaluated as a sum over cell faces *cf* of the volumetric flux $\phi_{\alpha,i}$. The core idea of FCT is to construct $\phi_{\alpha,i}$ as a combination of a flux obtained via a low-order interpolation scheme $\phi_{\alpha,i,LO} = (\alpha_i\mathbf{u}_i)|_{f,LO}\cdot\mathbf{S}_f$ and a flux obtained via a high-order interpolation scheme $\phi_{\alpha,i,HI} = (\alpha_i\mathbf{u}_i)|_{f,HI}\cdot\mathbf{S}_f$:

$$\phi_{\alpha,i} = \phi_{\alpha,i,LO} + \lambda\,(\phi_{\alpha,i,HI} - \phi_{\alpha,i,LO}) \tag{2.13}$$

in which λ are limiting factors that quantify the maximum possible contribution from a high-order flux that guarantees boundedness. All the different implementations of a FCT technique consist in variations on how λ is computed. The MULES algorithm is already implemented in several OpenFOAM solvers and was adopted in this work, beside few minor modifications. The general idea behind MULES is to compute λ from mass balance and allowable phase fraction considerations on a cell-by-cell basis. The detailed procedure at the heart of the MULES algorithm is described in (Weller, 2008; Santiago Marquez, 2013) and is conceptually similar to the limiters devised by Zalesak (Zalesak, 1979).

2.2.2. Pressure–velocity coupling and the Partial Elimination Algorithm

For a generic system of two phases that share the same pressure, one needs to solve a system of three variables, namely two velocities and pressure. The equation for pressure is normally obtained from an adequate manipulation of the momentum and continuity equations. The solution algorithm that couples these equations traditionally consists of three steps:

1. construct and solve the momentum Eq. (2.4) based on an explicit evaluation of the pressure gradient. This yields new phase velocities that do not strictly ensure mass conservation due to the explicit pressure gradient. This step is referred to as velocity predictor step;
2. construct and solve a pressure equation from the momentum and continuity equations. This and the next step are referred to as pressure correction step;
3. correct the velocities with the newly estimated pressure gradient to ensure mass conservation;

The specifics of how these steps are iterated are what distinguishes the various established solution algorithms (e.g. SIMPLE (Patankar and Spalding, 1972), PISO (Issa et al., 1986), PIMPLE). As previously stated, the framework of the pressure–velocity coupling employed in this work is the PIMPLE algorithm. In particular, the present subsection deals with introducing the particular way in which the pressure–velocity coupling is dealt with in our case, and the role of the Partial Elimination Algorithm. Assume we can linearize and discretize the momentum equations according to a chosen set of discretization schemes, so that Eq. (2.4) can be expressed as:

$$\widehat{M}_1\mathbf{u}_1 = \mathbf{b}_1(p) + K(\mathbf{u}_2 - \mathbf{u}_1)\quad \widehat{M}_2\mathbf{u}_2 = \mathbf{b}_2(p) + K(\mathbf{u}_1 - \mathbf{u}_2) \tag{2.14}$$

in which $\widehat{M}_i$ is the discretized momentum transport operator, $\mathbf{b}(p)_i$ is the source term resulting from an explicit evaluation for the pressure gradient, and where the fluid–fluid drag term is treated separately. The drag coefficient K_{12} pedix will be omitted for the sake of conciseness.

The first option to treat the coupling term is an explicit representation where the latest available values for the phasic velocities are employed. This treatment is simple but potentially unstable for large values of the coupling coefficient. The second option is to treat it semi-implicitly by modifying the operator diagonal as follows

$$\widehat{M}_i^{*} = \widehat{M}_i + K\widehat{I} \tag{2.15}$$

so to increase the diagonal dominance of the system. Nonetheless, an explicit evaluation of the remaining $K\mathbf{u}_j$ on the left hand side is still required. This approach is generally referred to as the Partially Implicit treatment (Karema and Lo, 1999). The third option, know as Partial Elimination (Spalding, 1980) and adopted in our algorithm, removes the coupling terms via an adequate manipulation of the original equations:

$$(\widehat{M}_1 + K(\widehat{M}_2^{-1}\widehat{M}_1 + \widehat{I}))\mathbf{u}_1 = \mathbf{b}_1(p) + K\widehat{M}_2^{-1}(\mathbf{b}_1(p) + \mathbf{b}_2(p))\quad (\widehat{M}_2 + K(\widehat{M}_1^{-1}\widehat{M}_2 + \widehat{I}))\mathbf{u}_2 = \mathbf{b}_2(p) + K\widehat{M}_1^{-1}(\mathbf{b}_1(p) + \mathbf{b}_2(p)) \tag{2.16}$$

It is clear from the structure of system (2.16) that the coupling between the phases is now treated in a fully implicit manner. However, the numerical assembly of the system is computationally more intensive due to the additional matrix inversion operations. The idea is then to not solve the momentum equations, but to take advantage of the discretized momentum matrices to assemble the pressure equation. The velocities can then be algebraically reconstructed from the pressure gradient as it is generally done in a traditional pressure correction step.

Various possibilities exist to assemble the pressure equation. In our case, due to the shared phase pressure framework, it is obtained from the sum of the continuity Eq. (2.1) for the two phases:

$$\nabla\cdot(\alpha_1\mathbf{u}_1 + \alpha_1\mathbf{u}_1) = S(p) \tag{2.17}$$

in which $S(p)$ denotes the sum of all terms related to mass-transfer and compressibility effects of both phases as seen on the right-hand side of Eq. (2.1), the latter possibly being pressure-dependent in some way. The time derivative of the sum of phase fractions is null assuming that the volume fraction of the structure is constant in time. In order to obtain an equation for pressure from (2.17), an algebraic relationship between phase velocities and pressure is required.

The momentum transport operator $\widehat{M}_i$ introduced earlier can be decomposed in a diagonal part $\widehat{M}_{D,i}$ and an off-diagonal part $\widehat{M}_{OD,i}$. Since the diagonal coefficients are the same for all velocity components of a phase, these will be represented with a scalar field $A_i = \widehat{M}_{D,i}$. By recalling that the source term $\mathbf{b}_i(p) = \mathbf{b}_i^* + \alpha_i \nabla p$ is inclusive of the pressure gradient and by defining $\mathbf{H}_i = -\widehat{M}_{OD,i}\mathbf{u}_i + \mathbf{b}_i^*$ as the explicit evaluation of the off-diagonal velocity contributions inclusive of other source terms, an algebraic relationship between each velocity field and the pressure gradient can be obtained from either (2.14) or (2.16) (Issa et al., 1991; Jasak., 1996). In our algorithm, we employ (2.16) by decomposing the momentum transport operator $\widehat{M}_i$ in a diagonal part A_i and an explicit off-diagonal contribution $\mathbf{H}_i$ as described before. By defining $A_i^{**} = A_i + K(1 + \frac{A_i}{A_j})$, the following relationship can be obtained:

$$\begin{aligned}\mathbf{u}_1 &= \frac{1}{A_1^{**}}\left(\left(1+\frac{K}{A_2}\right)\mathbf{H}_1 + \frac{K}{A_2}\mathbf{H}_2 - \left(\left(1+\frac{K}{A_2}\right)\alpha_1 + \frac{K}{A_2}\alpha_2\right)\nabla p\right)\\ \mathbf{u}_2 &= \frac{1}{A_2^{**}}\left(\left(1+\frac{K}{A_1}\right)\mathbf{H}_2 + \frac{K}{A_1}\mathbf{H}_1 - \left(\left(1+\frac{K}{A_1}\right)\alpha_2 + \frac{K}{A_1}\alpha_1\right)\nabla p\right)\end{aligned} \tag{2.18}$$

The relationship established by (2.18) thus accounts for Partial Elimination (as there is no coupling between the velocities) and is used to construct and solve a pressure equation from (2.17) as:

$$\begin{aligned}&\nabla\cdot\left(\left(\frac{\alpha_1}{A_1^{**}}\left(1+\frac{K}{A_2}\right)+\frac{\alpha_2 K}{A_2^{**}A_1}\right)\mathbf{H}_1 + \left(\frac{\alpha_2}{A_2^{**}}\left(1+\frac{K}{A_1}\right)+\frac{\alpha_1 K}{A_1^{**}A_2}\right)\mathbf{H}_2\right) +\\ -\ &\nabla\cdot\left(\left(\frac{\alpha_1}{A_1^{**}}\left(\alpha_1+\frac{K}{A_2}(\alpha_1+\alpha_2)\right)+\frac{\alpha_2}{A_2^{**}}\left(\alpha_2+\frac{K}{A_1}(\alpha_1+\alpha_2)\right)\right)\nabla p\right) = S(p)\end{aligned} \tag{2.19}$$

Finally, the relationship established by (2.18) is used to compute the velocities from the newly computed pressure gradient.

For the sake of completeness and better comparison to readily available literature and implementations, it is worth noting that, in existing OpenFOAM implementations, the pressure–velocity relationship is instead derived from (2.14) with an implicit treatment of $-K\mathbf{u}_i$ by decomposing the momentum transport operator $\widehat{M}_i$ in a diagonal part $A_i^* = A_i + K$ and an explicit off-diagonal contribution $\mathbf{H}_i$ as described before. The following relationship is obtained:

$$\begin{aligned}\mathbf{u}_1 &= \frac{1}{A_1^*}(-\alpha_1\nabla p + \mathbf{H}_1 + K\mathbf{u}_2)\\ \mathbf{u}_2 &= \frac{1}{A_2^*}(-\alpha_2\nabla p + \mathbf{H}_2 + K\mathbf{u}_1)\end{aligned} \tag{2.20}$$

The relationship established by (2.20) is not based on Partial Elimination. Nonetheless, it is used to construct and solve a pressure equation from (2.17), which consequently also evaluates the coupling contribution explicitly:

$$\nabla\cdot\left(\frac{\alpha_1}{A_1^*}(\mathbf{H}_1 + K\mathbf{u}_2) + \frac{\alpha_2}{A_2^*}(\mathbf{H}_2 + K\mathbf{u}_1)\right) - \nabla\cdot\left(\left(\frac{\alpha_1^2}{A_1^*} + \frac{\alpha_2^2}{A_2^*}\right)\nabla p\right) = S(p) \tag{2.21}$$

Partial Elimination is considered in the velocity correction step. Unlike the approach we adopted, velocities are not computed from the same pressure–velocity relationship used to construct the pressure equation. Instead, Partial Elimination is used to eliminate the coupling variables from (2.20) to obtain:

$$\begin{aligned}\mathbf{u}_1 &= \frac{1}{A_1^* - \frac{K^2}{A_2^*}}\left(-\alpha_1\nabla p + \mathbf{H}_1 + \frac{K}{A_2^*}(\mathbf{H}_2 - \alpha_2\nabla p)\right)\\ \mathbf{u}_2 &= \frac{1}{A_2^* - \frac{K^2}{A_1^*}}\left(-\alpha_2\nabla p + \mathbf{H}_2 + \frac{K}{A_1^*}(\mathbf{H}_1 - \alpha_1\nabla p)\right)\end{aligned} \tag{2.22}$$

One may notice how the algorithm we adopted eliminates an explicit evaluation of the coupling term in the pressure equation. The performance of this approach versus the one that is currently implemented in OpenFOAM is compared in Section 4.

For completeness, we should point out at this point that face-centered superficial fluxes ϕ_i, instrumental to the solution of the phase fraction equations via the MULES algorithm, must also be updated after solving the pressure equation. In principle these fluxes could be evaluated as $\phi_i = \mathbf{u}|_f \cdot \mathbf{S}_f$ with $\cdot|_f$ denoting face interpolation via a prescribed scheme and $\mathbf{S}_f$ being the face surface area vectors. In practice, velocity itself is not interpolated to the faces, rather, the right-hand-side of the employed pressure–velocity relationship (e.g. (2.18), (2.22)) is interpolated and dotted with $\mathbf{S}_f$. As the solution of the pressure equation relies on a standard OpenFOAM variant of the Rhie-Chow interpolation technique (Rhie and Chow, 1983), these interpolated quantities are already available after the pressure equation solution, so that there is no need for additional velocity interpolation operations.

2.2.3. Remarks on cell-centered and face-centered momentum treatment

Our algorithm employs a standard co-located variable treatment, meaning that all variables are stored at cell centers. In particular, during the update of the velocity at the pressure-correction step (e.g. (2.18), (2.22)), pressure gradients are evaluated at cell-centers. However, an alternative option consists in reconstructing the velocity at the cell centers directly from the superficial fluxes ϕ_i, evaluated as described in the previous section. In particular, the face-interpolated velocity field $\mathbf{u}|_f$ (or any vector field defined at cell faces for that matter) can be reconstructed at cell centers via:

$$\mathbf{u} = \left(\sum_f (\mathbf{n}_f \otimes \mathbf{S}_f)\right)^{-1} \cdot \sum_f (\mathbf{n}_f (\mathbf{u}|_f \cdot \mathbf{S}_f)) \tag{2.23}$$

with $\mathbf{n}_f$ being the face normal vector. By considering that $\phi_i = \mathbf{u}|_f \cdot \mathbf{S}_f$, it follows that (2.23) can be used to reconstruct cell-center velocities from the superficial flux. This option is implemented in some multi-phase OpenFOAM solvers and is referred to as face-momentum, as it mimics the logic behind a staggered grid approach. The face-momentum approach was found to reduce phase fraction checkerboarding effects in some circumstances and enhance stability (Weller, 2015). For this reason, a face-centered variant of our algorithm has also been developed. It should however be noted that, as the superficial fluxes are obtained from a face-interpolated version of the pressure–velocity relationship, a certain degree of loss of accuracy is expected, with particular regard to the transport part of the momentum equation that figures in both the explicit off-diagonal contributions $\mathbf{H}_i$ and the diagonal coefficients A_i.

2.2.4. Remarks on phase appearance-disappearance

As it is clear from the governing equation, the presence of single-phase regions in the computational domain results in null matrix diagonal coefficients. While different approaches exist with dealing with the so called phase appearance-disappearance issue (Zou et al., 2016), the approach employed in the present algorithm consists in limiting the phase fraction value to a residual value of $1{\cdot}10^{-9}$ before velocity and enthalpy matrix assembly and discretization. The actual choice of the residual value was found to be inconsequential on calculation result as long as it was reasonably small (i.e. $\leqslant 1{\cdot}10^{-3}$). On a different note instead is the issue of phase-appearance disappearance due to thermally-

driven phase change. If the effective volumetric enthalpy of one phase (i.e. the enthalpy per unit volume times the phase fraction $\alpha\rho h$) is small in a given portion of the domain (e.g. of the vapour phase at boiling inception), said phase will be considerably more prone to large temperature oscillations: for a given quantity of volumetric heat transferred to or from one phase in a computational cell, the phase temperature change will be inversely proportional to its mass in said cell. In turn, the large temperature oscillations will affect the thermally-driven mass transfer, and severe instabilities can occur. This issue can be entirely solved by setting the enthalpy of the low-volume-fraction phase to the saturation enthalpy in those cells where the phase fraction is below a user-specified threshold, and solving for the phasic enthalpy in the rest of the domain. This proves necessary only for the vapour phase in systems in which a large density difference exists between vapour and liquid (e.g. sodium in SFRs). A threshold phase fraction value of the order of magnitude of 10^{-2} proved to be effective in the vast majority of sodium boiling cases.

3. Verification

While verification should in principle cover both fluid-dynamics and thermo-dynamics, the thermal part of the solver relies on well established OpenFOAM matrix assembly and solution operations. Thus, the verification covers only the fluid-dynamics aspect of the solver, with particular regard to the Partial Elimination Algorithm. The chosen verification approach consists in the Method of Manufactured Solution, which has enjoyed some utilization in the field of multi-phase solvers (Brady et al., 2012) and is considered to be the most flexible verification approach. A detailed explanation of the Method of Manufactured Solutions is out of scope of the present work, as it is a standard, albeit complex verification approach that has been extensively covered in other works (Salari and Knupp, 2000). Nonetheless, the general idea behind such approach is introduced in subSection 3.1 while the results of the verification effort are summarized in (3.2) for a variety of cases.

3.1. The Method of Manufactured Solutions

Let us consider a computer code implementation that solves a problem for an unknown x governed by an equation represented in an operator form as $\widehat{M}x = b$. The Method of Manufactured Solutions can be summarized as follows:

1. select/construct an analytical target solution x_0 that the code is supposed to approximate;
2. derive the analytical source term s that satisfies $\widehat{M}x_0 - b = s$, which is possible since the analytical form of $\widehat{M}$ is known;
3. implement the analytical source term s in the code;
4. select a set of boundary conditions that reflect the behaviour of x_0 at the domain boundaries;
5. compare the solution x provided by the code with the target solution x_0 and draw conclusions based on a set of verification acceptance or rejection criteria.

The choice of x_0 is arbitrary and not bound by the physical properties that the system equation is supposed to model. Nonetheless, following the purpose of a verification effort, any x_0 should satisfy a certain number of properties, most importantly: 1) be composed of smooth analytic functions to ensure that theoretical order-of-accuracy can be attained (which ties into verification criteria discussed later); 2) be general enough and have a non trivial number of derivatives so to exercise all terms of the governing equation; 3) do not compromise code robustness by predicting values outside the intended solution range (e.g. unbounded phase fraction values). Furthermore, since boundary conditions are an integral part of the solution procedure, the verification should be repeated for different sets of boundary conditions.

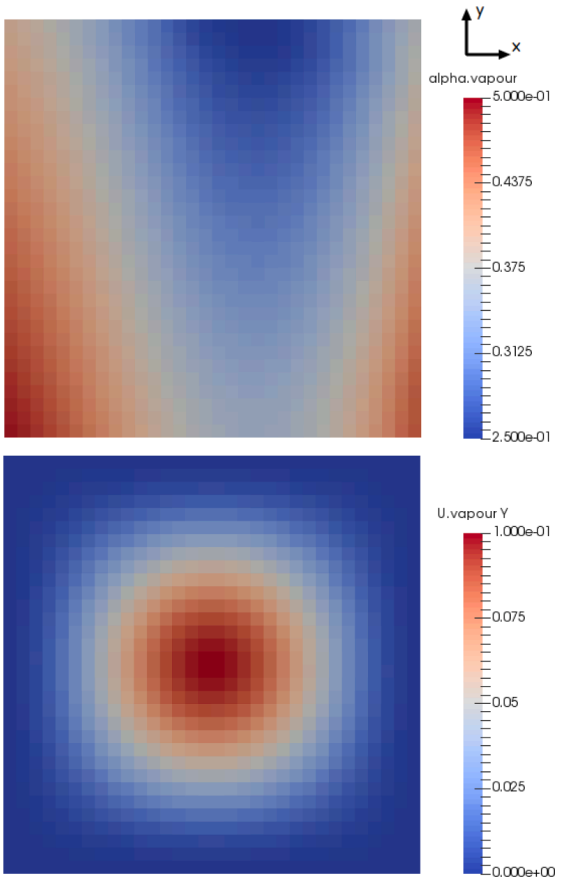

Fig. 2. Solution fields obtained at $t = 1.5\ s$ for α_2 (top) and $u_{2,y}$ (in $\frac{m}{s}$, bottom) on a 32 by 32 mesh.

With regards to verification acceptance criteria, the one employed in this work consists in the assessment of the order-of-accuracy of the numerical solution. This translates into evaluating the solution with increasingly refined meshes, and to assess the order-of-accuracy q, which quantifies how numerical errors ε scale as a function of a characteristic mesh dimension h, so that $\varepsilon = O(h^q)$. The verification criterion thus consists in assessing whether or not the observed order-of-accuracy reflects the theoretical order-of-accuracy, which depends on the employed set of equation discretization and interpolation schemes. In particular, the errors considered in this work consist of the L^2 norm error:

$$\varepsilon_{L^2} = \sqrt{\int_\Omega (x - x_0)^2 d\Omega} \tag{3.1}$$

and the L^∞ norm error:

$$\varepsilon_{L^\infty} = sup\{|x - x_0|\} \tag{3.2}$$

3.2. Results

The investigated domain consists in a 2-D square domain of side $L = 0.1\ m$ with a uniform square meshing. The following expressions have been selected for phase fractions α_i, phase velocities $\mathbf{u}_i$ and pressure p:

$$\begin{aligned}
\alpha_s &= x + y; \\
\alpha_1 &= 0.5 + 2.5x\ cos(\pi(10x - 4t)) \\
\alpha_2 &= 1.0 - \alpha_1 - \alpha_s \\
u_{1,x} &= 0.25 sin^2(10\pi y) \\
u_{1,y} &= 0.1 sin^2(10\pi y) sin^2(10\pi x) \\
u_{2,x} &= u_{1,x} \\
u_{2,y} &= -u_{1,y} \\
p &= 1{\cdot}10^5 + (x - 0.1){\cdot}10^4
\end{aligned} \tag{3.3}$$

This choice exercises all the terms of the governing equations, including all derivative terms. With regards to boundary conditions, a number of combinations were investigated. The results presented in this section were obtained via Neumann boundary conditions for pressure at $y = 0, y = L, x = 0$ and for $\alpha_{1,2}, \mathbf{u}_{1,2}$ at $y = 0, y = L, x = L$, while Dirichlet boundary conditions were imposed for pressure at $x = L$ and for $\alpha_{1,2}, \mathbf{u}_{1,2}$ at $x = 0$. Results comparable to the ones that are presented were obtained for other combinations of boundary conditions. Fluid thermo-physical properties and inter-phase coupling coefficients were set so that the contributions of all terms in the momentum equations are comparable in magnitude. The resulting analytical source terms for the phase fraction equations, momentum equations and the pressure equations are not reported for conciseness, given the non-trivial amount of derivative operators to be evaluated.

Time-dependent simulations were run from 0 s to 6 s and the L^2 and L^∞ norms for all the involved fields were averaged over said time duration. Progressively refined meshes from 16 to 128 cells per side were investigated. In particular, the performance of the cell-centered variant versus the face-centered variant of the algorithms were compared. In both cases, the newly implemented Partial Elimination Algorithm was used. As an example, the fields for some of the variables of interest are provided in Fig. 2. The results of the verification process are summarized in Fig. 3 for the cell-centered algorithm and in Fig. 4 for the face-centered algorithm, respectively.

Since a linear scheme was employed for the discretization of derivatives and for variable interpolation in the momentum and pressure equations, the theoretical order of accuracy for pressure and velocity should be 2. However, this is expected to be affected by the coupled nature of the equations with other equations discretized with lower schemes. In particular, the theoretical order of accuracy for the phase fraction equations cannot be exactly determined due to the iterative nature of the flux correction technique employed by the MULES algorithm. Nonetheless, the order of accuracy should lie between 1 due to the upwind first order scheme used for the bounded fluxes estimation, and 2 due to the scheme by Van Leer (van Leer, 1974) used for the high-order correction fluxes. Based on these considerations, the verification of the cell-centered version of the algorithm is deemed satisfactory, also in the light of the small magnitudes of the errors. This holds true for both the L^2 and L^∞ norms. However, the same does not appear to hold for the face-momentum algorithm, as the order-of-accuracy deteriorates for fine meshes. This is consistent with a larger diffusive contribution of the interpolation of velocity coefficients and off-diagonal velocity source terms at cell faces.

As a matter of fact, the diffusivity of the face-momentum algorithm is exacerbated by having set thermo-physical fluid properties so to result in comparable contributions from the different terms of the momentum equation. This results in a rather large choice for the fluids viscosity of $\mu_{1,2} \simeq 0.1 Pa\ s$, in the range of e.g. motor oils. A new set of less diffusive simulations was then performed with $\mu_{1,2} \simeq 1{\cdot}10^{-3} Pa\ s$, more compatible with the applications envisioned for the algorithm presented here. The results are presented in Fig. 5 and show that the order of accuracy significantly improves for the face-momentum algorithm. It should also be noticed that in all cases, degradation of the order of accuracy happens for error magnitudes below practical relevance.

4. Performance of the Partial Elimination Algorithm

To assess the performance of the novel implementation of a Partial Elimination Algorithm, the existing implementation in the available OpenFOAM solvers has been used as reference case. As a first test case, the 1-D motion of a dispersed fluid via advection by a carrier fluid through a porous structure was investigated. The initial and boundary conditions for the variables of interest in the domain of length $L = 2\ m$ are:

$$\begin{aligned}
\alpha_s(x,t) &= \begin{cases} 0 & x < 0.5 \vee x > 0.5 \\ 0.5 & 0.5 \leqslant x \leq 0.5 \end{cases} \\
\alpha_1(x,0) &= \begin{cases} 1 & x < 0.5 \vee x > 0.5 \\ 0.25 & 0.5 \leqslant x \leq 0.5 \end{cases} \\
\alpha_2(x,0) &= \begin{cases} 0 & x < 0.5 \vee x > 0.5 \\ 0.25 & 0.5 \leqslant x \leq 0.5 \end{cases}
\end{aligned} \tag{4.1}$$

$$\mathbf{u}_1(x,0) = 0.5\ \frac{m}{s}$$

$$\mathbf{u}_2(x,0) = 0\ \frac{m}{s} \tag{4.2}$$

$$\mathbf{u}_1(0,t) = 0.5\ \frac{m}{s} \tag{4.3}$$

$$p(2,t) = 1{\cdot}10^5\ Pa \tag{4.4}$$

with all other variables having a null gradient at the boundaries. This

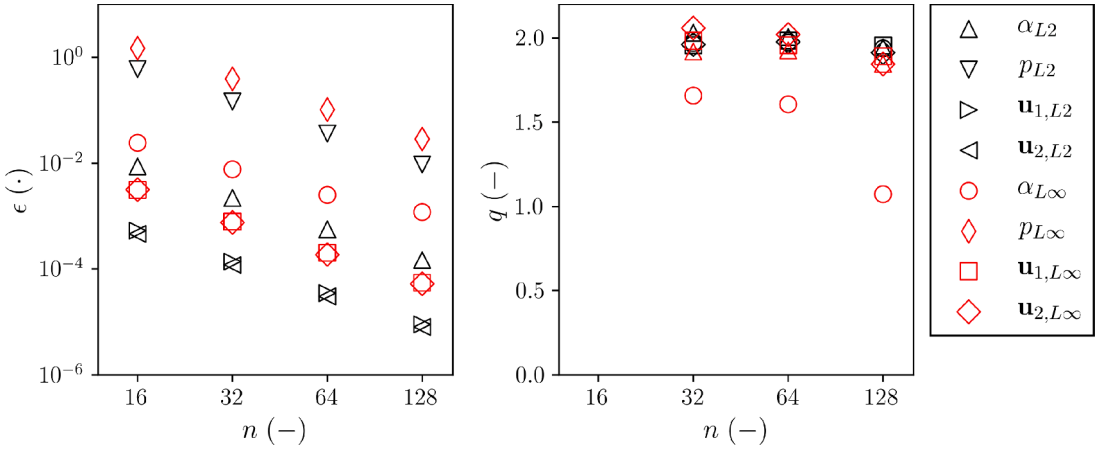

Fig. 3. Errors ϵ (left) and associated order-of-accuracy q (right) computed by the L_2 (black) and L_∞ (red) norms for the various quantities of interest with the cell-centered algorithm. Errors are absolute and dimensional, namely: $\alpha\ (-), p\ (Pa), \mathbf{u}_1, \mathbf{u}_2\ (\frac{m}{s})$.

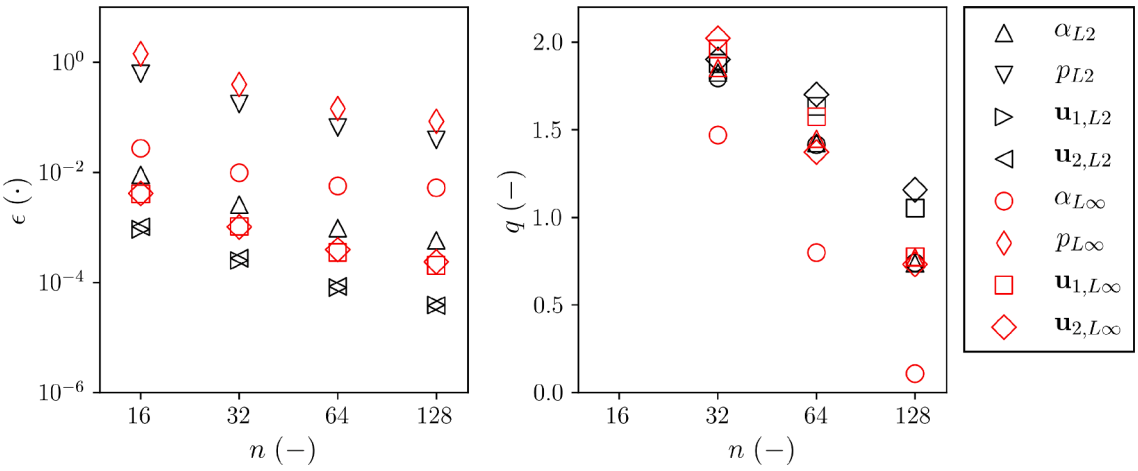

Fig. 4. Errors ϵ (left) and associated order-of-accuracy q (right) computed by the L_2 (black) and L_∞ (red) norms for the various quantities of interest with the face-centered algorithm. Errors are absolute and dimensional, namely: $\alpha\ (-), p\ (Pa), \mathbf{u}_1, \mathbf{u}_2\ (\frac{m}{s})$.

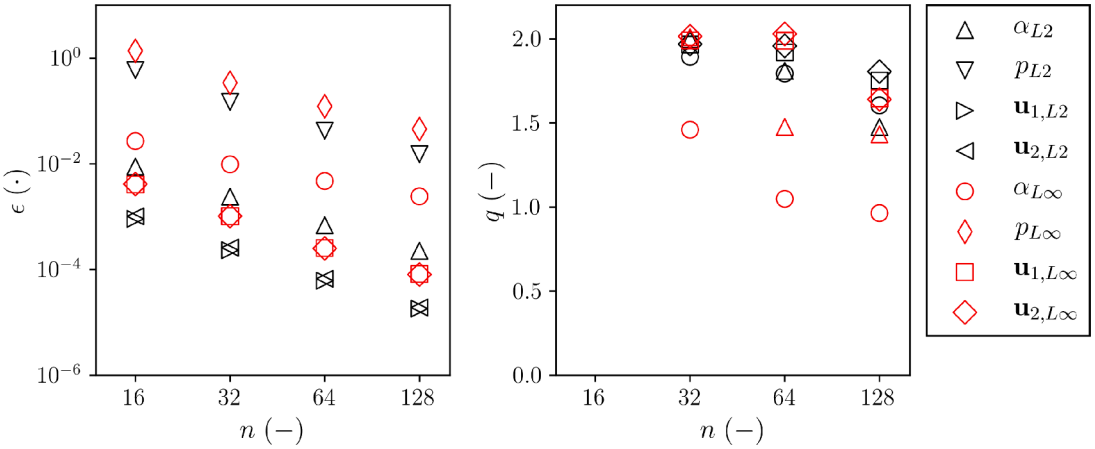

Fig. 5. Errors ϵ (left) and associated order-of-accuracy q (right) computed by the L_2 (black) and L_∞ norms for the various quantities of interest with the face-centered algorithm for a lower fluid viscosity. Errors are absolute and dimensional, namely: $\alpha\ (-), p\ (Pa), \mathbf{u}_1, \mathbf{u}_2\ (\frac{m}{s})$.

simplified test allows to establish the general trends to be expected as certain system properties are varied. Results are reported in Fig. 6. The left plots reports the last initial pressure residual at the last outer iteration for each time step, which is used as a measure of how well the pressure–velocity coupling is resolved. The right plots represents the total number of pressure linear solver iterations performed within the time-step. It is recalled that, as velocity is entirely reconstructed from pressure, the pressure equation residual is the only indicator of convergence of the pressure–velocity coupling. All simulations where performed with 3 outer iterations per time step and 3 pressure correctors per outer iteration. The Generalized Algebraic Multi-Grid (GAMG) linear solver was used for the cases reported here, yet the same trends were observed with the Preconditioned Bi-conjugate Gradient Stabilized (PBiCGStab) linear solver. The same linear solver convergence criteria were used in both cases.

As Partial Elimination is designed to improve performance in scenarios involving large inter-phase drag coefficients, the following cases were investigated. The only force acting on the fluids (excluding pressure gradients) is an inter-phase drag modelled as $K_{12} = 1{\cdot}10^4 \frac{kg}{m^3 s}$ and $K_{12} = 1{\cdot}10^5 \frac{kg}{m^3 s}$ for the first and second row in Fig. 6, respectively. Since the novel Partial Elimination implementation alters both the pressure equation coefficients and source terms, the last row investigates instead the effect of larger source terms due to e.g. virtual mass effects and fluid–structure drag. In particular, a virtual mass coefficient of 0.5 was chosen and a diagonal fluid–structure drag tensor with magnitude $|\mathbf{K}_{1s}| = 5{\cdot}10^3 \frac{kg}{m^3 s}$ to be comparable in magnitude with the fluid–fluid drag. Overall, the novel implementation appears to perform consistently better in these scenarios in terms of pressure equation residuals and of number of necessary iterations.

The performance of the algorithm has then been assessed for two more realistic scenarios. The first consists in a 1-D liquid sodium boiling transient in a bundle of electrically heated pins. The modelling details and correlation choices are specific to liquid sodium and are covered in previous work (Radman et al., 2019). The key aspect is that in such a transient, a high slip between the phases is observed, which results in large values for the inter-phase drag factor. Results are presented in Fig. 7 for the existing and novel Partial Elimination implementation in the same format as before. The start and end of the boiling can be inferred from the increase in pressure equation residuals and total number of pressure equation linear solver iterations in time. The overall results observed before still apply, with a significant reduction in both residuals and linear solver iterations. For this particular case, a $\sim 5\%$ reduction in simulation time was observed as well as a consequence of the reduced number of pressure equation iterations.

The second realistic scenario consists in Red a modified version of the standard 2-D OpenFOAM test case `bubbleColumn`, consisting of air bubbles injected at the bottom of a column of stationary water. Results for the performance of the existing and novel implementation of the

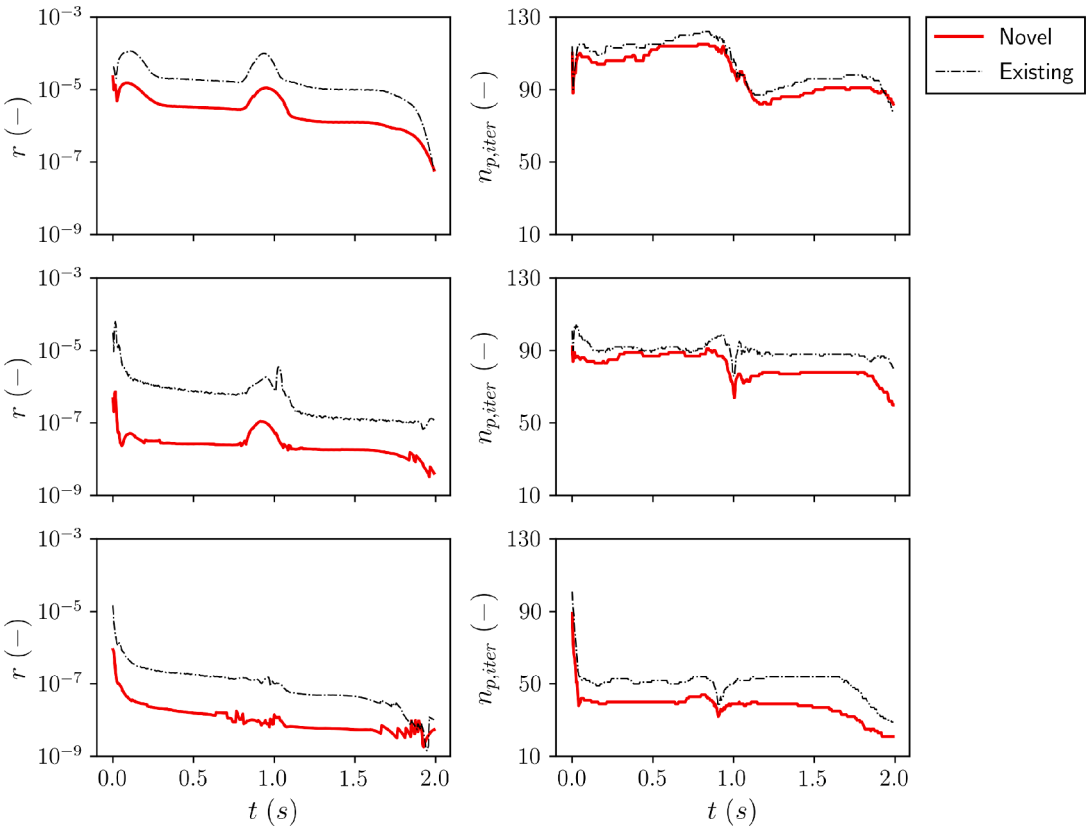

Fig. 6. Evolution of pressure initial residuals r at the last outer iteration and last pressure corrector (left column) and number of pressure iterations $n_{p,iter}$ (right column) within each simulation time step. These are compared between the novel and existing PEA implementations for three different cases: small drag factor $K_{12} = 1{\cdot}10^4 \frac{kg}{m^3 s}$ with no other forces acting on the fluids (top row); large drag factor $K_{12} = 1{\cdot}10^5 \frac{kg}{m^3 s}$ with no other forces acting on the fluids (middle row); large drag factor with the addition of explicit virtual mass forces due to a virtual mass coefficient of 0.5 and large fluid–structure drag $K_{1s} = 5{\cdot}10^3 \frac{kg}{m^3 s}$ (bottom row).

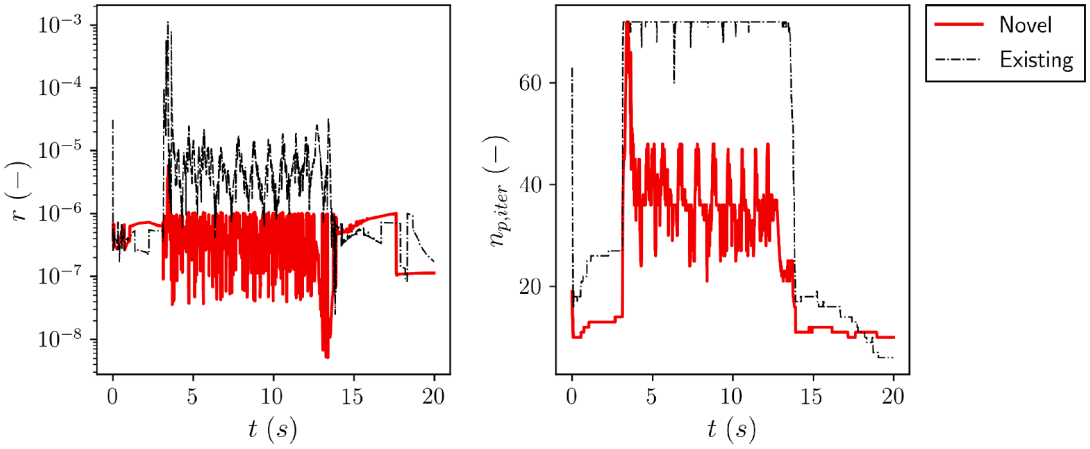

Fig. 7. Evolution of pressure initial residuals r at the last outer iteration and last pressure corrector (left) and number of pressure iterations $n_{p,iter}$ (right) within each simulation time step for a 1-D sodium boiling transient.

Partial Elimination Algorithm are presented in Fig. 8 for the first 2s of simulation. Due to the considerably smaller coupling between the phases, partly due to the lower slip velocity, there are no noticeable differences between the two algorithms. This case serves as an example to show that in those scenarios where inter-phase drag does not play a significant role between the phases, the novel Partial Elimination implementation performs comparably to the existing one.

5. Parallel performance

While it is true that one of the primary goals of a porous medium approach in this context is to reduce computational burdens, the modelling of large engineering systems (e.g. nuclear reactors) can still result, depending on the required degree of detail, in several tens of millions of mesh cells. For parallelization, our algorithm employs the standard MPI-based (Padua, 2011) domain-decomposition capabilities of the OpenFOAM framework. The scaling performance provided by this framework depends on the equations that are solved, and on the steps

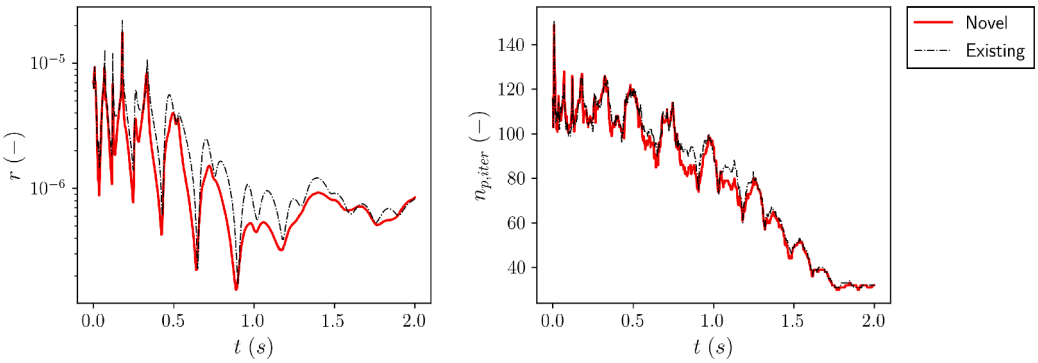

Fig. 8. Evolution of pressure initial residuals r at the last outer iteration and last pressure corrector (left) and number of pressure iterations $n_{p,iter}$ (right) within each simulation time step for a modified 2-D `bubbleColumn` OpenFOAM test case.

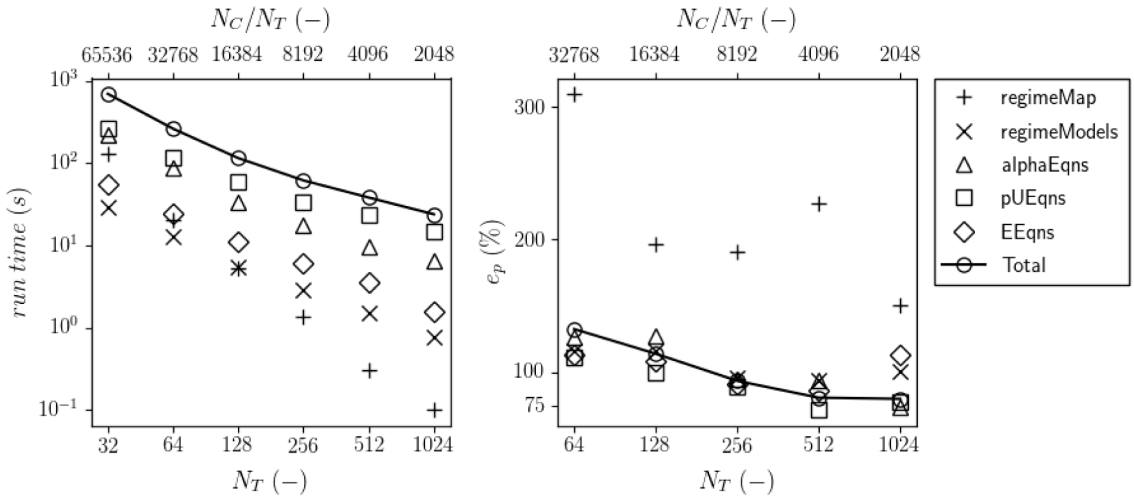

Fig. 9. strong scaling results for a mesh of 128^3 cells. These consist of the component run times (right) and parallel efficiencies (left) in relationship to the amount of threads N_T used for the simulation. Additional axes in terms of cells per thread $\frac{N_C}{N_T}$ are provided for ease of discussion. By "component" we mean a specific block of code with reference to what is defined in Fig. 1.

that are taken to solve them. This is why scaling performances have been preliminary investigated for our algorithm. In particular, the strong scaling characteristics of the solver and of the different algorithm components (with reference to Fig. 1) were assessed.

A test case that could stress all of the different components of the solver was devised. It consists of a 3-D cubic domain with one face acting as an inlet and the opposite face as an outlet. Two structures modelled as isotropic porous media are defined in the two cube halves contacting the inlet and outlet. These are referred to as "upwind" and "downwind" structures respectively. The surface temperatures of both structures are set to constant values so that the upwind half is above a prescribed fluid saturation temperature and the downwind one below said temperature. The thermo-physical properties and correlations that are chosen for modelling drag and heat transfer were that of liquid sodium (Radman et al., 2019). A fictitious regime map consisting of five identical regimes (i.e. prescribing the same correlations for drag and heat transfer) depending on the vapour phase volume fraction is provided. In essence, this test case enables the simulation of a transients that leads to a numerically steady state boiling scenario.

Strong scaling performance was assessed with a mesh of $128^3 \simeq 2 \cdot 10^6$ cells decomposed on a varying number of cores, ranging from 32 to 1024. A further set of simulations on a mesh of $256^3 \simeq 16 \cdot 10^6$ up to 4096 cores was also performed. However, the scalability results were in line with those established on the smaller mesh, and only these are presented in the following. Simulations were performed on a Cray XC-50 supercomputer operated by the Swiss National Supercomputing Centre (CSCS). Results are summarized in Fig. 9. The results report the total run time as well as the run time of each code section, referred to as "component" as reported in Fig. 1, as the number of threads is varied. For a better understanding of the trends, the resulting parallel efficiency of the algorithm is also reported. In this context, the parallel efficiency for a given case i is a measure of the simulation speed-up with respect to a case j performed on a different number of threads:

$$e_{P,i} = \frac{T_i}{T_j} \frac{N_{T,i}}{N_{T,j}} \tag{5.1}$$

with T being the simulation run time and N_T is the number of threads used for the simulation. In essence, the parallel efficiency should ideally be of the order of 100% or greater, meaning that e.g. a doubling of the number of threads results in halving of the total run time.

It can be observed that the algorithm as a whole as well as its individual components scale well up to 8192 cells per core, after which the parallel efficiency degrades, especially for the pressure–velocity coupling and MULES algorithms. This is expected for OpenFOAM-based solvers, which are known to scale poorly below $\simeq 10000$ cells per thread. When it comes to the individual impact of the different solver components, a clear trend emerges. The most computationally demanding components are, respectively, the pressure–velocity coupling algorithm, the MULES algorithm, the energy equation construction and solution, and the update of momentum and heat transfer coefficients based on the regime map. However, the update of the

regime map itself scales significantly more than linearly, as observed by the large values of the parallel efficiency above unity. This is attributed to the fact that the regime map update relies on a considerable amount of operations to be performed on a cell-by-cell basis to establish which particular flow regime exists in each mesh cell.This is suspected to cause a relatively large number of cache misses, which decrease as the amount of cells per thread (and thus cache requirements) is reduced. Nonetheless, even for more substantial counts of cells per thread, the overall cost associated with the regime map update is acceptable. What should be furthermore noted is that currently the regime map is updated at every outer corrector within a time-step. However, depending on the particular transient under investigation and the variety of the correlations prescribed for each flow regime, this update frequency of the regime map (i.e. at every outer iteration) could prove to be excessive. For very small time steps, the same might hold at a temporal scale. With these considerations in mind, there is further room for significant reduction of the regime map update cost at lower cells per thread counts.

6. Conclusions

A segregated algorithm for modelling dispersed two-phase flows in nuclear reactors and other complex engineering systems was developed and implemented based on the OpenFOAM finite volume library. The algorithm was verified via the Method of Manufactured Solutions by showing that the theoretical order of accuracy of the discretization schemes could be attained. This was performed on a variety of domains ranging from 1-D to 2-D, as 3-D is not necessary to exercise all of the terms that govern the conservation equations. Moreover, both a traditional cell-centered approach and a face-centered approach for velocity reconstruction from the pressure gradient have been verified via the same approach. A face-centered approach is a feature found in some multi-phase solvers within the OpenFOAM environment. Its inclusion (as an optional feature) and verification were of interest due to the advantages it provides in terms of face force balance and elimination of phase fraction staggering in certain scenarios. The verification process included a novel implementation of the Partial Elimination Algorithm (PEA), which was devised to further increase the computational performance in scenarios in which inter-phase momentum coupling is important. Subsequent to verification, its performance was also compared against the standard PEA implementation present in other OpenFOAM-based solvers for a variety of cases of interest. The novel implementation was found advantageous in terms of pressure equation residuals and number of pressure iterations for a number of applications of interest, most notably sodium boiling simulations. The advantages grow smaller as the importance of the inter-phase momentum coupling is decreased, yet this is in line with the range of applications in which PEA can prove beneficial.

The strong scaling performance of the algorithm and of its individual components was assessed up to 4096 threads and was found to be compatible with the performance of OpenFOAM-based solvers, meaning linear scalability up to $\simeq$ 10000 cells per thread.

Future work directions consist in expanding the available selection of correlations for momentum and heat transfer for additional working fluids (e.g. water), more targeted wall boiling models (e.g. the RPI wall boiling model (Kurul et al., 1991) for water) as well as their validation against experimental data. In particular, validation efforts are currently ongoing based on data from past sodium boiling experiments, namely those performed at experimental facilities at the JRC, Italy (Kottowski and Savatteri, 1984) and KfK, Germany (Bottoni et al., 1990). Moreover, the code will be integrated in the GeN-Foam platform, with the new multi-phase and multi-physics capabilities that will be subject to testing and validation.

CRediT authorship contribution statement

Stefan Radman: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualization. **Carlo Fiorina:** Conceptualization, Methodology, Resources, Data curation, Writing - review & editing, Supervision, Project administration, Funding acquisition. **Andreas Pautz:** Resources, Writing - review & editing, Supervision, Project administration, Funding acquisition.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by a grant from the Swiss National Supercomputing Centre (CSCS) under project ID sm34.

References

Abbassi, Y., Asgarian, S., Ghahremani, E., Abbasi, M., 2016. Investigation of natural convection in miniature neutron source reactor of isfahan by applying the porous media approach. Nuclear Engineering and Design 309, 213–223. https://doi.org/10.1016/j.nucengdes.2016.09.019. URL:http://www.sciencedirect.com/science/article/pii/S0029549316303375.

Boris, J.P., Book, D.L., 1997. Flux-corrected transport. Journal of Computational Physics 135 (2), 172–186. https://doi.org/10.1006/jcph.1997.5700. URL:http://www.sciencedirect.com/science/article/pii/S0021999197957004.

Bottoni, M., Dorr, B., Homann, C., Struwe, D., 1987. State of development of the computer programme BACCHUS-3D/TP for the description of transient two-phase flow conditions in LMFBR fuel pin bundles. Nuclear Engineering and Design 100 (3), 321–349. https://doi.org/10.1016/0029-5493(87)90084-7.

Bottoni, M., Dorr, B., Homann, C., Huber, F., Mattes, K., Peppler, F.W., Struwe, D., 1990. Experimental and numerical investigations of sodium boiling experiments in pin bundle geometry. Nuclear Technology 89 (1), 56–82. arXiv:https://doi.org/10.13182/NT90-A34359, doi:10.13182/NT90-A34359. URL:https://doi.org/10.13182/NT90-A34359.

Brady, P., Herrmann, M., Lopez, J., 2012. Code verification for finite volume multiphase scalar equations using the method of manufactured solutions. Journal of Computational Physics 231 (7), 2924–2944. https://doi.org/10.1016/j.jcp.2011.12.040.

Chavez, V.J., Imke, U., Sanchez-Espinoza, V., 2018. TWOPORFLOW: A two-phase flow porous media code, main features and validation with BWR-relevant bundle experiments. Nuclear Engineering and Design 338, 181–188. https://doi.org/10.1016/j.nucengdes.2018.08.009.

Clifford, I., 2013. A hybrid coarse and fine mesh solution method for prismatic high temperature gas-coold reactor thermal-fluid analysis, Ph.D. thesis, Pennsylvania State University, Pennsylvania, USA. URL:https://pdfs.semanticscholar.org/7096/4dc2d72b27aeab25fbd83746b564622f4623.pdf.

Colas, C., Ferrand, M., Hérard, J.-M., Latché, J.-C., Coupanec, E.L., 2019. An implicit integral formulation to model inviscid fluid flows in obstructed media. Computers & Fluids 188, 136–163. https://doi.org/10.1016/j.compfluid.2019.05.014. URL:http://www.sciencedirect.com/science/article/pii/S0045793019301598.

Fiorina, C., 2019. Impact of the volume heat source on the rans-based cfd analysis of molten salt reactors. Annals of Nuclear Energy 134, 376–382. https://doi.org/10.1016/j.anucene.2019.06.024. URL:http://www.sciencedirect.com/science/article/pii/S0306454919303421.

Fiorina, C., Clifford, I., Aufiero, M., Mikityuk, K., 2015. GeN-Foam: a novel OpenFOAM based multi-physics solver for 2D/3D transient analysis of nuclear reactors. Nuclear Engineering and Design 294, 24–37. https://doi.org/10.1016/j.nucengdes.2015.05.035.

Fiorina, C., Kerkar, N., Mikityuk, K., Rubiolo, P., Pautz, A., 2016. Development and verification of the neutron diffusion solver for the GeN-Foam multi-physics platform. Annals of Nuclear Energy 96, 212–222. https://doi.org/10.1016/j.anucene.2016.05.023.

Fiorina, C., Hursin, M., Pautz, A., 2017. Extension of the GeN-Foam neutronic solver to SP3 analysis and application to the CROCUS experimental reactor. Annals of Nuclear Energy 101, 419–428. https://doi.org/10.1016/j.anucene.2016.11.042.

Ford, C., Carrotte, J., Walker, A., 2013. The application of porous media to simulate the upstream effects of gas turbine injector swirl vanes. Computers & Fluids 77, 143–151. https://doi.org/10.1016/j.compfluid.2013.03.001. URL:http://www.sciencedirect.com/science/article/pii/S0045793013000844.

Hovi, V., Pättikangas, T., Riikonen, V., 2016. Coupled one-dimensional and cfd models for the simulation of steam generators. Nuclear Engineering and Design 310, 93–111. https://doi.org/10.1016/j.nucengdes.2016.09.023. URL:http://www.sciencedirect.com/science/article/pii/S002954931630351X.

Issa, R., Gosman, A., Watkins, A., 1986. The computation of compressible and incompressible recirculating flows by a non-iterative implicit scheme. Journal of Computational Physics 62 (1), 66–82. https://doi.org/10.1016/0021-9991(86)

90100-2. URL:http://www.sciencedirect.com/science/article/pii/0021999186901002.
Issa, R., Ahmadi-Befrui, B., Beshay, K., Gosman, A., 1991. Solution of the implicitly discretised reacting flow equations by operator-splitting. Journal of Computational Physics 93 (2), 388–410. https://doi.org/10.1016/0021-9991(91)90191-M. URL: http://www.sciencedirect.com/science/article/pii/002199919190191M.
Jasak, H., 1996. Error analysis and estimation for the finite volume method with applications to fluid flows, Ph.D. thesis, Impreial College, University of London.
Jasak, H., 2009. OpenFOAM: Open source CFD in research and industry. International Journal of Naval Architecture and Ocean Engineering 1 (2), 89–94. https://doi.org/10.2478/IJNAOE-2013-0011.
Karema, H., Lo, S., 1999. Efficiency of interphase coupling algorithms in fluidized bed conditions. Computers & Fluids 28 (3), 323–360. https://doi.org/10.1016/S0045-7930(98)00028-0. URL:http://www.sciencedirect.com/science/article/pii/S0045793098000280.
Kim, J., Sibilli, T., Ha, M.Y., Kim, K., Yoon, S.Y., 2019. Compound porous media model for simulation of flat top U-tube compact heat exchanger. International Journal of Heat and Mass Transfer 138, 1029–1041. https://doi.org/10.1016/j.ijheatmasstransfer.2019.04.116. URL:http://www.sciencedirect.com/science/article/pii/S0017931018333003.
Kottowski, H., Savatteri, C., 1984. Fundamentals of liquid metal boiling thermohydraulics. Nuclear Engineering and Design 82 (2), 281–304. https://doi.org/10.1016/0029-5493(84)90216-4.
Kozelkov, A., Lashkin, S., Efremov, V., Volkov, K., Tsibereva, Y., Tarasova, N., 2018. An implicit algorithm of solving Navier-Stokes equations to simulate flows in anisotropic porous media. Computers & Fluids 160, 164–174. https://doi.org/10.1016/j.compfluid.2017.10.029. URL:http://www.sciencedirect.com/science/article/pii/S0045793017303948.
Kurul, N., Podowski, M.. 1991. On the modeling of multidimensional effects in boiling channels. In: 27th National Heat Transfer Conference, American Nuclear Society, pp. 28–31.
Montenegro, G., Onorati, A., Torre, A.D., 2013. The prediction of silencer acoustical performances by 1d, 1d–3d and quasi-3d non-linear approaches. Computers & Fluids 71, 208–223. https://doi.org/10.1016/j.compfluid.2012.10.016. URL:http://www.sciencedirect.com/science/article/pii/S0045793012004057.
OpenCFD Ltd., 2021. URL:http://www.openfoam.com.
D. Padua (Ed.), 2011. Message Passing Interface (MPI), Springer US, Boston, MA, pp. 1116–1116. doi:10.1007/978-0-387-09766-4_2085.
Patankar, S., Spalding, D., 1972. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. International Journal of Heat Mass Transfer 15, 1787–1806.
Penha, D.L., Geurts, B., Stolz, S., Nordlund, M., 2011. Computing the apparent permeability of an array of staggered square rods using volume-penalization. Computers & Fluids 51 (1), 157–173. doi: 10.1016/j.compfluid.2011.08.011. URL: http://www.sciencedirect.com/science/article/pii/S0045793011002544.
Radman, S., Fiorina, C., Mikityuk, K., Pautz, A., 2019. A coarse-mesh methodology for modelling of single-phase thermal-hydraulics of esfr innovative assembly design. Nuclear Engineering and Design 355, 110291. https://doi.org/10.1016/j.nucengdes.2019.110291. URL:http://www.sciencedirect.com/science/article/pii/S0029549319303085.
Radman, S., Fiorina, C., Pautz, A., 2019. Preliminary development of a coarse-mesh sodium boiling model for openfoam-based multi-physics solvers. In: 8th International Topical Meeting on Nuclear Reactor Thermal Hydraulics. American Nuclear Society.
Rhie, C.M., Chow, W.L., 1983. Numerical study of the turbulent flow past an airfoil with trailing edge separation. AIAA Journal 21 (11) (1983) 1525–1532. arXiv:https://doi.org/10.2514/3.8284, doi:10.2514/3.8284. URL:https://doi.org/10.2514/3.8284.
Salari, K., Knupp, P., 2000. Code verification by the method of manufactured solutions. URL:https://prod-ng.sandia.gov/techlib-noauth/access-control.cgi/2000/001444.pdf.
Santiago Marquez, D., 2013. An extended mixture model for the simultaneous treatment of short and long scale interfaces, Ph.D. thesis. Universidad Nacional del Litoral.
Sarkar, M., Velusamy, K., Munshi, P., Singh, O.P., 2018. Investigation of heat transfer from a totally blocked fuel subassembly of fast breeder reactor with 7 and 19 pin bundles. Nuclear Engineering and Design 338, 74–91. https://doi.org/10.1016/j.nucengdes.2018.08.001. URL:http://www.sciencedirect.com/science/article/pii/S0029549318308604.
Saurel, R., Boivin, P., Métayer, O.L., 2016. A general formulation for cavitating, boiling and evaporating flows. Computers & Fluids 128, 53–64. https://doi.org/10.1016/j.compfluid.2016.01.004. URL:http://www.sciencedirect.com/science/article/pii/S0045793016000153.
Scolaro, A., Clifford, I., Fiorina, C., Pautz, A., 2020. The offbeat multi-dimensional fuel behavior solver. Nuclear Engineering and Design 358, 110416. https://doi.org/10.1016/j.nucengdes.2019.110416. URL:http://www.sciencedirect.com/science/article/pii/S0029549319304479.
Sipaun, S., Usman, S., 2015. Prediction of missouri s&t reactor's natural convection with porous media approximation. Nuclear Engineering and Design 285, 241–248. https://doi.org/10.1016/j.nucengdes.2015.01.001. URL:http://www.sciencedirect.com/science/article/pii/S0029549315000114.
Spalding, D.B., 1980. Numerical computation of multi-phase fluid flow and heat transfer 1, 139–167.
Vanderhaegen, M., Vierendeels, J., Arien, B., 2011. Cfd analysis of the myrrha primary cooling system. Nuclear Engineering and Design 241 (3), 775–784, the International Conference on Structural Mechanics in Reactor Technology (SMiRT19) Special Section. doi: 10.1016/j.nucengdes.2010.12.009. URL:http://www.sciencedirect.com/science/article/pii/S0029549311000070.
van Leer, B., 1974. Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second-order scheme. Journal of Computational Physics 14 (4), 361 – 370. doi:https://doi.org/10.1016/0021-9991(74)90019-9. URL:http://www.sciencedirect.com/science/article/pii/0021999174900199.
Weller, H.G., 2008. A new approach to vof-based interface capturing methods for incompressible and compressible flow, Tech. Rep. Report. OpenCFD Ltd., London.
Weller, H.G., twoPhaseEulerEoam: Added experimental face-based momentum equation formulation, URL:https://github.com/OpenFOAM/OpenFOAM-dev/commit/16f03f8a393bd4b6e70b548812777bb10a3e1dda (2015).
Weller, H.G., Tabor, G., Jasak, H., Fureby, C., 1998. A tensorial approach to computational continuum mechanics using object-oriented techniques. Computers in Physics 12 (6), 620–631. https://doi.org/10.1063/1.168744 arXiv:https://aip.scitation.org/doi/pdf/10.1063/1.168744 URL:https://aip.scitation.org/doi/abs/10.1063/1.168744.
Yu, Y., Merzari, E., Obabko, A., Thomas, J., 2015. A porous medium model for predicting the duct wall temperature of sodium fast reactor fuel assembly. Nuclear Engineering and Design 295, 48–58. https://doi.org/10.1016/j.nucengdes.2015.09.020. URL: http://www.sciencedirect.com/science/article/pii/S0029549315004276.
Zalesak, S.T., 1979. Fully multidimensional flux-corrected transport algorithms for fluids. Journal of Computational Physics 31 (3), 335–362. https://doi.org/10.1016/0021-9991(79)90051-2. URL:http://www.sciencedirect.com/science/article/pii/0021999179900512.
Zou, L., Zhao, H., Zhang, H., 2016. Solving phase appearance/disappearance two-phase flow problems with high resolution staggered grid and fully implicit schemes by the jacobian-free newton-krylov method. Computers & Fluids 129, 179–188. https://doi.org/10.1016/j.compfluid.2016.02.008. URL:http://www.sciencedirect.com/science/article/pii/S0045793016300263.
U.S. Nuclear Regulatory Commission, 2008. TRACE theory manual V5.0, URL:https://www.nrc.gov/docs/ML1200/ML120060218.pdf.
Whitaker, S., A simple geometrical derivation of the spatial averaging theorem. Chemical Engineering Education, pp. 18-21, 50, Winter 1985.