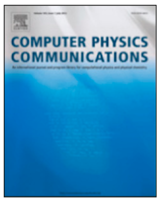
Contents lists available at ScienceDirect

Computer Physics Communications

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



Parallel finite-element codes for the simulation of two-dimensional and three-dimensional solid–liquid phase-change systems with natural convection[☆,☆☆]

Georges Sadaka[a], Aina Rakotondrandisa[a], Pierre-Henri Tournier[b], Francky Luddens[a], Corentin Lothodé[a], Ionut Danaila[a,*]

[a] *Laboratoire de Mathématiques Raphaël Salem, Université de Rouen Normandie, CNRS UMR 6085, Avenue de l'Université, BP 12, F-76801 Saint-Étienne-du-Rouvray, France*
[b] *Sorbonne Université, CNRS UMR 7598, Laboratoire Jacques-Louis Lions, F-75005, Paris, France*

ARTICLE INFO

Article history:
Received 20 February 2020
Received in revised form 7 June 2020
Accepted 28 June 2020
Available online 12 July 2020

Keywords:
Phase change
PCM
Parallel computing
Finite element
3D melting
Mesh adaptivity
Navier–Stokes–Boussinesq
FreeFem

ABSTRACT

We present and distribute a FreeFem++ Toolbox for the parallel computing of two- or three-dimensional liquid–solid phase-change systems involving natural convection. FreeFem++ (www.freefem.org) is a free finite-element software available for all existing operating systems. We use the recent library `ffddm` that makes available in FreeFem++ state-of-the-art scalable Schwarz domain decomposition methods (DDM). The single domain approach used in our previous contribution (Rakotondrandisa et al., 2020) is adapted for the use of the DDM method. As a result, the computational time is considerably reduced for 2D configurations and furthermore 3D problems become affordable. The numerical method is based on an enthalpy-porosity model. The same set of equations is solved in both liquid and solid phases: the incompressible Navier–Stokes equations with Boussinesq approximation for thermal effects. A Carman–Kozeny-type penalty term is added to the momentum equations to bring progressively the velocity to zero into the solid. Model equations are discretized using Galerkin triangular or tetrahedral finite elements. The coupled system of equations is integrated in time using a second-order Gear implicit scheme. The resulting discrete equations are solved using a Newton algorithm. The DDM approach is based on an overlapping Schwarz method. The mesh is first split in subdomains using `Scotch` or `Metis` libraries. The final linear system is then solved in parallel using a GMRES Krylov method, with a Restricted Additive Schwarz (RAS) preconditioner. The mesh is adapted during the computation using metrics control. The 3D-mesh adaptivity uses the `mmg` (www.mmgtools.org) open source library. Parallel 2D and 3D computations of benchmark cases of increasing difficulty are presented: natural convection of air, natural convection of water, melting or solidification of a phase-change material, and, finally, a water freezing case. For each case, careful validations are provided and the performance of the code is assessed. The robustness of the Toolbox in 3D is also demonstrated by adapting the number of processors to the number of tetrahedra, which can considerably vary after the mesh adaptation.

Program summary
Program Title: PCM_Toolbox_DDM_2D and PCM_Toolbox_DDM_3D
CPC Library link to program files: http://dx.doi.org/10.17632/dk49rfrz9y.1
Licensing provisions: Apache License, 2.0
Programming language: FreeFem++(www.freefem.org), mmg (www.mmgtools.org)
Nature of problem: The software is scoped to parallel computations of 2D or 3D configurations of liquid–solid phase-change problems with convection in the liquid phase. Natural convection, melting and solidification processes are illustrated in the paper. The software can be easily modified to take into account different related physical models.
Solution method: We use a single domain approach, solving the incompressible Navier–Stokes equations with Boussinesq approximation in both liquid and solid phases. A Carman–Kozeny-type penalty term is

☆ The review of this paper was arranged by Prof. Hazel Andrew.
☆☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.

E-mail addresses: georges.sadaka@univ-rouen.fr (G. Sadaka), aina.rakotondrandisa@etu.univ-rouen.fr (A. Rakotondrandisa), tournier@ljll.math.upmc.fr (P.-H. Tournier), francky.luddens@univ-rouen.fr (F. Luddens), corentin.lothode@univ-rouen.fr (C. Lothodé), ionut.danaila@univ-rouen.fr (I. Danaila).

https://doi.org/10.1016/j.cpc.2020.107492

added to the momentum equations to bring the velocity to zero into the solid phase. An enthalpy model is used in the energy equation to take into account the phase change. Discontinuous variables (latent heat, material properties) are regularized through an intermediate (mushy) region. Space discretization is based on Galerkin triangular/tetrahedral finite elements. A second order Gear implicit scheme is used for the time integration of the coupled system of equations. The resulting discrete equations are solved using a Newton algorithm. Piecewise quadratic (P2) finite-elements are used for the velocity and piecewise linear (P1) for the pressure. For the temperature both P2 or P1 discretization are possible. The mesh is first split in subdomains using `Scotch` or `Metis` libraries, which are interfaced with FreeFem++. Then, a Schwarz domain decomposition method is used through the FreeFem++library `ffddm`. The final linear system is solved in parallel using a GMRES Krylov method, with a Restricted Additive Schwarz (RAS) preconditioner. Mesh adaptivity using metrics control makes possible the optimization of the distribution of mesh elements. For 3D case, the `mmg` open source library is used to adapt the mesh.

1. Introduction

Accurate and efficient numerical simulations of solid–liquid phase-change problems are needed in many practical applications. Metal casting, Earth's mantle formation and food freezing are well explored topics in this area. Recently, a great deal of attention was paid to the simulation of latent thermal energy storage (LTES) systems based on phase-change materials (PCM). Such devices are used for thermal energy storage (*e.g.* for solar power generation) or passive temperature control (*e.g.* for modern portable electronics) devices. For a review of various applications of PCMs, see recent reviews by Agyenim et al. [1] and Kalnæs and Jelle [2].

In all these problems, melting and solidification are fundamental processes that are difficult to simulate if accurate physical models are used. A sketch of the melting problem is shown in Fig. 1a. Buoyancy forces in the liquid (melted) phase generate a significant convective flow and thus deform the liquid–solid interface. Solidification is a similar process, but with a slower evolution and the possibility to generate several solid–liquid interfaces propagating simultaneously. Resolving all the scales in the liquid region using the Navier–Stokes–Boussinesq equations and accurately capturing the solid–liquid interfaces are the main challenges for a numerical system addressing these problems. In our recent contribution [3] we presented an in-depth review of physical and numerical models dealing with solid–liquid phase-change problems with convection. The approach retained in [3] was based on the widely used enthalpy-porosity single-domain model, also called the fixed-domain model [4]. In the present contribution, we extend this model to parallel computations using domain-decomposition (DD) methods, as illustrated in Fig. 1b.

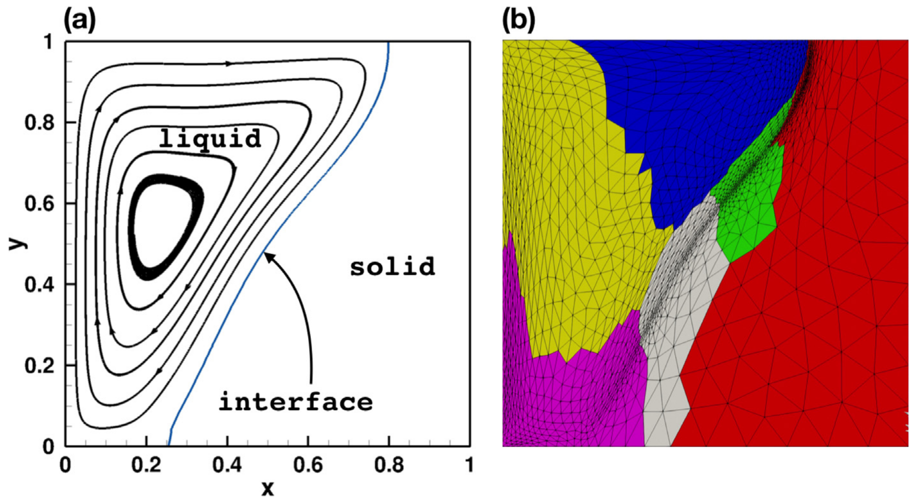


Fig. 1. Sketch of the liquid–solid phase change problem considered in this paper. (a) Illustration of the melting, with convection in the liquid phase (streamlines of the velocity field) and a bent interface separating the two phases. (b) Illustration of the domain-decomposition used in present numerical simulations. Colored patches represent subdomains created by automatic graph partitioning libraries. Note the adapted mesh, especially around the liquid–solid interface. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The main advantage of the enthalpy-porosity single-domain model is that the solid–liquid interface is not explicitly tracked. The position of the interface is computed a posteriori by post-processing the obtained temperature field T (in Fig. 1b, the interface is represented by the iso-contour $T = 0$). This makes the model appropriate for the use of domain decomposition models and parallel computing. Front tracking or front fixing deforming-grid methods (*e.g.* [5,6]) would obviously introduce an algorithmic complexity penalizing parallel computing performances. Note that a different category of models was recently suggested in the literature, based on the Lattice Boltzmann Method [7,8]. Such methods based on non-deterministic models are also well adapted for parallel computations. For a comprehensive review of models for phase-change problems with convection, see [9].

Another advantage of the enthalpy-porosity single-domain model is that the same Navier–Stokes–Boussinesq (NSB) system of equations is solved in both liquid and solid phases. Keeping in mind that the NSB equations are pertinent only in the liquid phase, their extension into the solid phase has to preserve the single-domain formulation. This is generally obtained by introducing in the momentum equations a penalization source term that brings the velocity to zero in the solid and do not affect the liquid. One of the most used expression of the penalization source term follows the Carman–Kozeny model for the permeability of a porous medium [10–12], but other mathematically equivalent expressions were suggested [13,14]. The energy equation is also modified to be valid in both phases using an enthalpy (temperature-transform) model introducing a regularized latent-heat term. Different formulations and implementations of the enthalpy-porosity model are presented in [5,15,16].

The enthalpy-porosity single-domain model was implemented in [3] using an adaptive finite-element method. The corresponding Toolbox for FreeFem++(a free software under LGPL license)[1] was distributed with that paper. Extensive validation tests proved the ability of the numerical system to deal with melting and/or solidification problems of increasing difficulty. Problems with complex shapes of the computational domain or with multiple solid–liquid interfaces were successfully computed. The mesh adaptivity capabilities of the method ensured reasonable computational costs, since the mesh was refined only in the zones of large gradients of variables (artificial mushy region, boundary layers) and coarsened in the solid zones with low gradients. However, some cases, such as the water freezing or the melting-solidification cycle of a PCM, demanded hours or days on a personal computer to simulate physically pertinent time evolution.

Consequently, the main purpose of the present contribution is to extend the single-domain model of Rakotondrandisa et al. [3] to parallel computing using domain-decomposition methods. For two-dimensional (2D) configurations, the new Toolbox can be used to reduce the computational time on personal computers with multi-core processors. Using high-performance computing (HPC) facilities, the Toolbox is well adapted to simulate 3D configuration of phase-change problems. It is important to note that very few 3D simulations with accurate capturing (mesh adaptivity) of the solid–liquid interface were reported in this research area. Adaptive FE methods were suggested for classical two-phase Stefan problem in 2D and 3D [17,18]. For phase-change systems with convection, adaptivity strategies were suggested and tested only for 2D problems [12,19,20]. An attempt to adapt the mesh in 3D simulations for melting phenomena was undertaken by Zimmerman and Kowalski [21] using an AMR (Adaptive Mesh Refinement) technique based on a dual-weighted residual method. The method displayed a major drawback, since the mesh was refined during the advancement of the liquid–solid front, but never coarsened behind. As a consequence, only a very preliminary coarse grid simulation of the 3D melting was possible using the AMR method. We show in this contribution that our new method using domain decomposition and mesh adaptivity is perfectly suited to the accurately simulate 3D melting or solidification, in simple or complex-shape geometries.

The new parallel Toolbox is based on a single-domain enthalpy-porosity model for solid–liquid phase change problems with convection. For the energy conservation equation, a temperature-based formulation takes into account the latent heat by introducing a discontinuous source term. For the mass and momentum conservation equations, we solve in the entire domain the incompressible Navier–Stokes equations with Boussinesq approximation for buoyancy effects. To bring the velocity to zero in the solid phase, we introduce in the momentum equation a penalty term following the Carman–Kozeny model. The coupled system of momentum and energy equations is integrated in time using a second-order implicit Gear scheme. The resulting discretized equations are solved using a Newton method [19]. For the space discretization we use Taylor–Hood triangular finite elements, *i.e.* P_2 for the velocity and temperature and P_1 for the pressure. Temperature is discretized using P_2 or P_1 finite elements. Discontinuous variables (latent heat, thermal diffusivity, etc.) at the solid–liquid interface are regularized through an intermediate artificial mushy region.

To enable parallel computing, we use the `ffddm` framework [22], which is a set of FreeFem++ scripts implementing Schwarz domain-decomposition methods for the efficient solution of linear systems. The mesh is first split into subdomains using an automatic graph partitioning library, such as `Scotch` [23] or `Metis` [24]. Each subdomain is assigned to a MPI process. The linear systems are assembled and solved in parallel using a GMRES Krylov method with a Restricted Additive Schwarz (RAS) preconditioner [25]. Mesh adaptivity using metrics control makes possible the optimization of the distribution of mesh elements. For 3D cases, the mesh adaptivity is more involved: the metric is first computed using `mshmet`, which is a module inside FreeFem++, and then used within the `mmg`.[2] remeshing tool [26] to generate the adapted new mesh.

The paper is organized as follows. Section 2 introduces the enthalpy-porosity single domain model based on the Navier–Stokes–Boussinesq equations. Section 3 presents the adaptive finite-element numerical method using a Newton algorithm for the non-linear discretized equations. We also discuss in this section the Schwarz domain decomposition method used for parallel computations and the algorithm for mesh adaptivity (with emphasis on the 3D adaptivity technique). A description of the programs contained in the provided 2D and 3D toolboxes is given in Section 4. The next two sections are devoted to extensive numerical validations of the method for 2D benchmarks (Section 5) and corresponding 3D configurations (Section 6). The robustness of the algorithm is demonstrated by comparing our results with reference data available in the literature. The capabilities of the Toolbox to deal with complex geometries are also illustrated. The main features of the software and possible extensions are summarized in Section 7.

2. Navier–Stokes-Boussinesq equations and enthalpy-porosity model

We consider a solid–liquid system placed in a three-dimensional domain Ω. The dimensionless system of equations to be solved in both liquid and solid regions is based on the incompressible Navier–Stokes equations, with Boussinesq approximation for buoyancy effects, and a temperature transforming model for the energy equation [27,28]:

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\boldsymbol{u} + \nabla p - \frac{1}{Re}\nabla^2 \boldsymbol{u} - f_B(\theta)\, \boldsymbol{e}_z - A_{mushy}(\theta)\boldsymbol{u} = 0, \tag{2}$$

[1] FreeFem++for different OS can be downloaded from http://www.freefem.org/.

[2] http://www.mmgtools.org/

$$\frac{\partial (C\theta)}{\partial t} + \nabla \cdot (C\theta \boldsymbol{u}) - \nabla \cdot \left(\frac{K}{RePr} \nabla \theta \right) + \frac{\partial (CS)}{\partial t} = 0. \tag{3}$$

Non-dimensional space, velocity, temperature and time variables in (1)–(3) were obtained from physical ones after applying the following scaling:

$$\boldsymbol{x} \to \frac{\boldsymbol{x}}{H}, \ \boldsymbol{u} \to \frac{\boldsymbol{u}}{V_{ref}}, \ \theta = \frac{T - T_{ref}}{\delta T}, \ t \to \frac{V_{ref}}{H} t, \tag{4}$$

where H is the reference length (usually the height of the cavity when a rectangular domain is considered) and V_{ref} a reference velocity that will be defined differently for melting and solidification problems. T_{ref} is the reference temperature and in most cases $T_{ref} = T_f$ (the temperature of fusion), unless otherwise specified. Consequently, the non-dimensional temperature of fusion is set to $\theta_f = 0$. Temperature difference δT defines a temperature scale, that will be set differently for melting and solidification cases.

The linearized Boussinesq buoyancy force (f_B), the Reynolds (Re) and Prandtl (Pr) numbers are defined as (subscripts s and l refer to the solid and the liquid phases, respectively):

$$f_B(\theta) = \frac{Ra}{PrRe^2} \theta, \quad Re = \frac{\rho V_{ref} H}{\mu_l} = \frac{V_{ref} H}{\nu_l}, \quad Pr = \frac{\nu_l}{\alpha_l}, \tag{5}$$

with ν the kinematic viscosity and $\alpha = k/(\rho c)$ the thermal diffusivity. In the expression of f_B, the Rayleigh number of the flow is defined as:

$$Ra = \frac{g \beta H^3 \delta T}{\nu_l \alpha_l}, \tag{6}$$

with β the thermal expansion coefficient and g the gravitational acceleration.

If previous non-dimensional numbers are pertinent only in the liquid phase, the non-dimensional conductivity K and specific heat C are defined in both phases:

$$K(\theta) = \frac{k}{k_l} = \begin{cases} 1, & \theta \geq \theta_f, \\ k_s/k_l, & \theta < \theta_f. \end{cases}, \quad C(\theta) = \frac{c}{c_l} = \begin{cases} 1, & \theta \geq \theta_f, \\ c_s/c_l, & \theta < \theta_f. \end{cases} \tag{7}$$

The non-dimensional function $S = s/s_l$ in the energy equation (3) takes a similar non-dimensional form:

$$S(\theta) = \frac{s}{s_l} = \begin{cases} \dfrac{h_{sl}/c_l}{\delta T} = \dfrac{1}{Ste}, & \theta \geq \theta_f, \\ 0, & \theta < \theta_f, \end{cases} \tag{8}$$

with h_{sl} the latent heat of fusion and Ste the Stefan number.

Discontinuous step-functions defined in (7) and (8) are replaced by continuous and differentiable hyperbolic-tangent functions, generically defined for all θ by the formula [19]:

$$F(\theta; a_s, \theta_s, R_s) = f_l + \frac{f_s - f_l}{2} \left\{ 1 + \tanh \left(a_s \left(\frac{\theta_s - \theta}{R_s} \right) \right) \right\}, \tag{9}$$

where f_l, f_s are the imposed values in the liquid and solid phases, a_s a smoothing parameter, θ_s the central value (around which we regularize) and R_s the smoothing radius. For example, we use for the non-dimensional source term in (3) the following regularization over the artificial mushy region $\theta \in [-\varepsilon, \varepsilon]$:

$$S(\theta) = \frac{1}{Ste} - \frac{1}{2\,Ste} \left\{ 1 + \tanh \left(\frac{\theta_r - \theta}{R_s} \right) \right\}, \tag{10}$$

where θ_r is the central value around which we regularize (typically $\theta_r = \theta_f = 0$) and R_s the smoothing radius (typically $R_s = \varepsilon$).

Finally, the penalty term $A_{mushy}(\theta)\boldsymbol{u}$ in momentum equation (2) follows from the Carman–Kozeny model [12,27,29]:

$$A_{mushy}(\theta) = -\frac{C_{CK}(1 - L_f(\theta))^2}{L_f(\theta)^3 + b}, \tag{11}$$

where $L_f(\theta)$ is the local liquid fraction, which is 1 in the fluid region and 0 in the solid. L_f is regularized inside the artificial mushy-region using a hyperbolic-tangent similar to (10). The Carman–Kozeny constant C_{CK} is set to a large value (10^6) and the constant $b = 10^{-6}$ is introduced to avoid divisions by zero.

3. Numerical method

3.1. Finite-element formulation

Finite-element methods for solving Navier–Stokes type systems of equations like (1)–(3) are generally based on a separate discretization of the temporal derivative (using finite differences, splitting or characteristics methods) and the generalization of the Stokes problem for the resulting system [30–32]. To simplify the presentation, we consider in the following that $C = 1$. For the phase-change problems considered in this paper, this is a physically valid assumption.

For the time integration, we use a second-order Gear (BDF2) finite-difference scheme (see also [12]):

$$\frac{d\phi}{dt} \simeq \frac{3\phi^{n+1} - 4\phi^n + \phi^{n-1}}{2\delta t}, \tag{12}$$

computing the solution ϕ^{n+1} at time $t_{n+1} = (n+1)\delta t$ by using two previous states (ϕ^n, ϕ^{n-1}). We use this scheme to advance in time both velocity ($\phi = \boldsymbol{u}$) and temperature fields ($\phi = \theta$). The other terms in Eqs. (1)–(3) are treated implicitly. We obtain the following implicit semi-discretization in time of the single-domain model (1)–(3):

$$\nabla \cdot \boldsymbol{u}^{n+1} = 0, \tag{13}$$

$$\begin{aligned}\frac{3}{2}\frac{\boldsymbol{u}^{n+1}}{\delta t} + (\boldsymbol{u}^{n+1}\cdot\nabla)\boldsymbol{u}^{n+1} + \nabla p^{n+1} - \frac{1}{Re}\nabla^2\boldsymbol{u}^{n+1} \\ -A_{mushy}(\theta^{n+1})\boldsymbol{u}^{n+1} - f_B(\theta^{n+1})\boldsymbol{e}_z = \\ 2\frac{\boldsymbol{u}^n}{\delta t} - \frac{\boldsymbol{u}^{n-1}}{2\delta t},\end{aligned} \tag{14}$$

$$\begin{aligned}\frac{3}{2}\frac{\theta^{n+1} + S(\theta^{n+1})}{\delta t} + \nabla\cdot\left(\boldsymbol{u}^{n+1}\theta^{n+1}\right) - \nabla\cdot\left(\frac{K(\theta^{n+1})}{RePr}\nabla\theta^{n+1}\right) = \\ 2\frac{\theta^n + S(\theta^n)}{\delta t} - \frac{\theta^{n-1} + S(\theta^{n-1})}{2\delta t} \quad .\end{aligned} \tag{15}$$

To solve the system of Eqs. (13)–(15) we use a classical Galerkin finite-element method. We consider homogeneous Dirichlet boundary conditions for the velocity, *i.e.* $\boldsymbol{u} = 0$ on $\partial\Omega$, and set the classical Hilbert spaces for the velocity and pressure:

$$\boldsymbol{V} = V \times V \times V,\ V = H_0^1(\Omega), \quad Q = \left\{ q \in L^2(\Omega) \middle| \int_\Omega q = 0 \right\} \tag{16}$$

Following the generalization of the Stokes problem [30–32], the weak formulation of the system (13)–(15) can be written as: find $(\boldsymbol{u}^{n+1}, p^{n+1}, \theta^{n+1}) \in \boldsymbol{V} \times Q \times V$, such that:

$$b\left(\boldsymbol{u}^{n+1}, q\right) - \gamma(p^{n+1}, q) = 0,\ \ \forall q \in Q \tag{17}$$

$$\begin{aligned}\frac{3}{2\delta t}\left(\boldsymbol{u}^{n+1}, \boldsymbol{v}\right) + c\left(\boldsymbol{u}^{n+1}; \boldsymbol{u}^{n+1}, \boldsymbol{v}\right) + \frac{1}{Re}a\left(\boldsymbol{u}^{n+1}, \boldsymbol{v}\right) \\ -(A_{mushy}(\theta^{n+1})\boldsymbol{u}^{n+1}, \boldsymbol{v}) + b\left(\boldsymbol{v}, p^{n+1}\right) - \left(f_B(\theta^{n+1})\boldsymbol{e}_z, \boldsymbol{v}\right) \\ = \frac{2}{\delta t}\left(\boldsymbol{u}^n, \boldsymbol{v}\right) - \frac{1}{2\delta t}\left(\boldsymbol{u}^{n-1}, \boldsymbol{v}\right),\ \ \forall \boldsymbol{v} \in \boldsymbol{V}\end{aligned} \tag{18}$$

$$\begin{aligned}\frac{3}{2\delta t}\left(\theta^{n+1} + S(\theta^{n+1}), \phi\right) + \left(\boldsymbol{u}^{n+1}\cdot\nabla\theta^{n+1}, \phi\right) + \left(\frac{K(\theta^{n+1})}{RePr}\nabla\theta^{n+1}, \nabla\phi\right) \\ = \frac{2}{\delta t}\left(\theta^n + S(\theta^n), \phi\right) - \frac{1}{2\delta t}\left(\theta^{n-1} + S(\theta^{n-1}), \phi\right),\ \forall \phi \in V,\end{aligned} \tag{19}$$

where $(u, v) = \int_\Omega u \cdot v$ denotes the scalar product in $L^2(\Omega)$ or $\left(L^2(\Omega)\right)^2$; the bilinear forms a, b and trilinear form c are defined as [31,32]:

$$a : \boldsymbol{V} \times \boldsymbol{V} \to \mathbb{R},\ a(\boldsymbol{u}, \boldsymbol{v}) = \int_\Omega \nabla^t\boldsymbol{u} : \nabla\boldsymbol{v} = \sum_{i,j=1}^{3}\int_\Omega \partial_j u_i \cdot \partial_j v_i,$$

$$b : \boldsymbol{V} \times Q \to \mathbb{R},\ b(\boldsymbol{u}, q) = -\int_\Omega \nabla\cdot\boldsymbol{u}\, q = -\sum_{i=1}^{3}\int_\Omega \partial_i u_i \cdot q,$$

$$c : \boldsymbol{V} \times \boldsymbol{V} \times \boldsymbol{V} \to \mathbb{R},\ c(\boldsymbol{w}; \boldsymbol{z}, \boldsymbol{v}) = \int_\Omega \left[(\boldsymbol{w}\cdot\nabla)\boldsymbol{z}\right]\cdot\boldsymbol{v} = \sum_{i,j=1}^{3}\int_\Omega w_j(\partial_j z_i)v_i.$$

Note that we introduced in Eq. (17) a penalty term on the pressure. An extensive discussion of the role of this parameter in reinforcing the incompressibility constraint and in the stabilization of the method is provided in [3]. We recall the main lines of this discussion. Following the incompressibility constraint (13), the pressure is defined up to an additive constant and imposing that $p^{n+1} \in Q$ removes this uncertainty. Equation (17) numerically ensures that the pressure field is of zero average in Ω. The penalty term in (17) acts at an algebraic level, by modifying the matrix of the final system (a zero lower diagonal block is avoided). Since in the present toolbox we use only iterative solvers (GMRES), this modification is not essential. It is in exchange important when LU-type direct solvers without pivoting are used, essentially for 2D calculations (UMFPACK solver in [3,33], SuperLU in [34]). Finally, the penalty constant γ has no role in stabilizing the method. Since the Taylor–Hood finite elements used in our simulations satisfy the inf–sup condition, this technique is assimilated to a *stable penalty*, or, in other words, just a computational trick to obtain a good solution [35,36]. Note that, compared to classical penalty methods, we use in our calculations very low values of the penalty parameter ($\gamma = 10^{-7}$). This ensures very low values of the average on Ω for the pressure and also for the divergence of the velocity field.

The system of non-linear equations (17)–(19) is solved using a Newton method. To advance the solution from time t_n to t_{n+1}, we start from an initial guess $\boldsymbol{w}_0 = (\boldsymbol{u}^n, p^n, \theta^n)$ (which is the solution at t_n), and construct the Newton sequence $\boldsymbol{w}_k = (\boldsymbol{u}_k, p_k, \theta_k)$ by solving for each inner iteration k:

$$b\left(\boldsymbol{u}_{k+1}, q\right) - \gamma(p_{k+1}, q) = 0, \tag{20}$$

$$\frac{3}{2\delta t}\left(\boldsymbol{u}_{k+1}, \boldsymbol{v}\right) + c\left(\boldsymbol{u}_{k+1}; \boldsymbol{u}_k, \boldsymbol{v}\right) + c\left(\boldsymbol{u}_k; \boldsymbol{u}_{k+1}, \boldsymbol{v}\right)$$

$$\begin{aligned}
&+\frac{1}{Re}a\left(\boldsymbol{u}_{k+1},\boldsymbol{v}\right)-\left(\frac{dA_{mushy}}{d\theta}(\theta_k)\,\theta_{k+1}\,\boldsymbol{u}_k,\boldsymbol{v}\right)-\left(A_{mushy}(\theta_k)\,\boldsymbol{u}_{k+1},\boldsymbol{v}\right)+b\left(\boldsymbol{v},p_{k+1}\right)\\
&-\left(\frac{df_B}{d\theta}(\theta_k)\,\theta_{k+1}\,\boldsymbol{e}_z,\boldsymbol{v}\right)=\frac{1}{\delta t}\left(2\boldsymbol{u}^n-\frac{1}{2}\boldsymbol{u}^{n-1},\boldsymbol{v}\right)\\
&+c\left(\boldsymbol{u}_k;\boldsymbol{u}_k,\boldsymbol{v}\right)-\left(\frac{dA_{mushy}}{d\theta}(\theta_k)\,\theta_k\,\boldsymbol{u}_k,\boldsymbol{v}\right)-\left(\left(\frac{df_B}{d\theta}(\theta_k)\,\theta_k-f_B(\theta_k)\right)\boldsymbol{e}_z,\boldsymbol{v}\right),
\end{aligned}\tag{21}$$

$$\begin{aligned}
&\frac{3}{2\delta t}\left(\theta_{k+1}+\frac{dS}{d\theta}(\theta_k)\,\theta_{k+1},\phi\right)+\left(\boldsymbol{u}_k\cdot\nabla\theta_{k+1},\phi\right)+\left(\boldsymbol{u}_{k+1}\cdot\nabla\theta_k,\phi\right)+\left(\frac{K(\theta_k)}{RePr}\nabla\theta_{k+1},\nabla\phi\right)\\
&+\left(\frac{dK}{d\theta}(\theta_k)\frac{\theta_{k+1}}{RePr}\nabla\theta_k,\nabla\phi\right)=\frac{2}{\delta t}\left(\theta^n+S(\theta^n),\phi\right)+\frac{3}{2\delta t}\left(\frac{dS}{d\theta}(\theta_k)\,\theta_k-S(\theta_k),\,\phi\right)\\
&\left(\boldsymbol{u}_k\cdot\nabla\theta_k,\phi\right)-\frac{1}{2\delta t}\left(\theta^{n-1}+S(\theta^{n-1}),\,\phi\right)+\left(\frac{dK}{d\theta}(\theta_k)\frac{\theta_k}{RePr}\nabla\theta_k,\nabla\phi\right).
\end{aligned}\tag{22}$$

Note that the last term of Eq. (21) cancels in the case of a linear Boussinesq force f_B (see Eq. (5)); this is not the case when non-linear variations of the density of the liquid are considered (convection or solidification of water). Note also that the previous system of Eqs. (20)–(22) depends only on $\boldsymbol{u}^n$, $\boldsymbol{u}^{n-1}$, θ^n and θ^{n-1} and is independent of p^n, the pressure being in this approach a Lagrange multiplier for the divergence free constraint.

The Newton loop (following k) has to be iterated until convergence for each time step δt following the algorithm:

```
|| Navier-Stokes time loop following $n$
|| set $\boldsymbol{w}_0=(\boldsymbol{u}^n,p^n,\theta^n)$
||     || Newton iterations following $k$
||     ||     solve (20) to get $\boldsymbol{w}_{k+1}$
||     || stop when $\|\boldsymbol{w}_{k+1}-\boldsymbol{w}_k\|<\xi_N$
|| actualize $(\boldsymbol{u}^{n+1},p^{n+1},\theta^{n+1})=\boldsymbol{w}_{k+1}$.                (23)
```

For the space discretization of the system (20)–(22) we use standard Taylor–Hood finite elements [37], approximating the velocity with P$_2$ (piecewise quadratic) finite elements ($\boldsymbol{V}_h$ space), and the pressure with the P$_1$ (piecewise linear) finite elements (Q_h space):

$$\boldsymbol{V}_h=\left\{\boldsymbol{v}\in C^0(\bar{\Omega})^2\,\middle|\,\forall K\in\mathcal{T}_h,\quad \boldsymbol{v}_{|K}\in P_2\right\},\tag{24}$$

$$Q_h=\left\{v\in C^0(\bar{\Omega})\,\middle|\,\forall K\in\mathcal{T}_h,\quad v_{|K}\in P_1\right\},\tag{25}$$

where K is an element of the triangulation $\mathcal{T}_h$, with characteristic mesh size h. Temperature and enthalpy variables are discretized using either P$_1$ or P$_2$ finite elements.

This algorithm was implemented using the open-source software FreeFem++ [38,39]. The FreeFem++programming framework offers the advantage to hide all technical issues related to the finite element method. The high level programming language with syntax close to mathematical formulations, makes the implementation of the present numerical algorithm very easy. Similar algorithms based on FreeFem++were successfully used for solving different systems of equations with locally sharp variation of the solution, such as Gross–Pitaevskii equation [40,41] or Laplace equations with nonlinear source terms [42].

The FreeFem++syntax to implement the Newton algorithm is very close to the mathematical formulation given above. We present below the main elements of syntax for the 3D formulation. We start by creating a new type `Wh` that will be used to define variables gathering in a vector all the unknowns of the problem: the three components of the velocity, the pressure and the temperature. `Wh` thus corresponds to $V\times V\times V\times Q\times V$ (see Eq. (16)). Each component of a vector of type `Wh` is independently assigned with a finite-element discretization (P$_2$, P$_1$ or other available in FreeFem++). In our case, we use Taylor–Hood finite elements for the fluid part and P$_2$ for the temperature and thus we define the vector finite-element space as `fespace Wh(Th,[P2,P2,P2,P1,P2])`. The unknowns of the problem are then defined by declaring: `Wh [u1,u2,u3,p,T];`. Note that `Wh` is associated to the mesh `Th`. If the mesh `Th` changes (following mesh adaptation), the definition of `Wh` is automatically associated to the new mesh. Corresponding test functions are defined similarly: `Wh [v1,v2,v3,q,TT];`. The next step in building the FreeFem++program is to define the weak (variational) formulation. The built-in function `varf` facilitates the implementation of the algorithm (20)–(22) with a syntax close to the mathematical formulation. The use of the `macro` environment (which is a pre-processor command for a simple syntax replacement in the script) makes the reading of the programs very intuitive, when comparing each term to its mathematical expression. For example, the following `macros` define the necessary operators for the variational formulation (20)–(22):

```
macro grad(u) [dx(u),dy(u),dz(u)]//
macro Grad(u) [grad(u#1),grad(u#2),grad(u#3)]//
macro div(u)(dx(u#1)+dy(u#2)+dz(u#3))//
macro ugrad(u,v)([u#1,u#2,u#3]'*grad(v))//
macro UgradV(u,v) [ugrad(u,v#1),ugrad(u,v#2),ugrad(u,v#3)]//
macro a(Mu,u,v)(Mu*(Grad(u):Grad(v)))//
macro b(u,q)(-div(u)*q)//
macro c(w,Z,v)(UgradV(w,Z)'*[v#1,v#2,v#3])//
```

Note that in some `macros` we used the concatenation operator `#`. This means that when we use in a script, for example, `Grad(uw)`, the preprocessor will automatically replace this part with `[grad(uw1),grad(uw2),grad(uw3)]` in the final script (that will be executed). Vector operators (transposition `'` and contraction `:`) are naturally implemented in FreeFem++. The `varf` syntax is used to build the matrix and the rhs-vector of the final linear system. Consequently, it follows closely the mathematical formulation (20)–(22).

To simplify the presentation, we start by giving below the `varf` formulation for the simple case of the stationary convection of air (the terms coming from the discretization of the time derivative are not present):

```
macro VarfStatNATCONV(varfName, meshName, VhName)
varf varfName([uw1,uw2,uw3,pw,Tw],[v1,v2,v3,q,TT])
= int3d(meshName,qforder=ord)( b(uw,q) - gamma*pw*q
+ c(uw,u,v) + c(u,uw,v) + IRe*a(uw,v) + b(v,pw)
- dfB(T)*Tw*v3+ ugrad(u,Tw)*TT + ugrad(uw,T)*TT
+ grad(Tw)'*grad(TT)*IPr )
+ bcu1 + bcu2 + bcu3 + bcT;
// EOM

macro VarfrhsStatNATCONV(varfName, meshName, VhName)
varf varfName([uw1,uw2,uw3,pw,Tw],[v1,v2,v3,q,TT])
= int3d(meshName,qforder=ord)( c(u,u,v) + ugrad(u,T)*TT
- dfB(T)*T*v3 + fB(T)*v3 )
+ bcu1 + bcu2 + bcu3 + bcT;
// EOM
```

The syntax `+bcu1+ ...` is used to include the `macros` that implement boundary conditions. The correspondence with variables in Eqs. (20)–(22) is the following: `[uw1,uw2,uw3,pw,Tw]` corresponds to unknowns $\boldsymbol{w}_{k+1} = (u1_{k+1}, u2_{k+1}, u3_{k+1}, p_{k+1}, \theta_{k+1})$, `[v1,v2,v3,q,TT]` to test functions and `[u1,u2,u3,p,T]` to previous Newton step $\boldsymbol{w}_k = (u1_k, u2_k, u3_k, p_k, \theta_k)$. For the time evolution formulations, we use `[u1p,u2p,u3p,pp,Tp]` and `[u1pp,u2pp,u3pp,ppp,Tpp]` for variables at time t_n and t_{n-1}, respectively. Consequently, the full formulation for the Newton algorithm (20)–(22) for the case of the melting of a PCM becomes:

```
macro VarfPCM(varfName, meshName, VhName)
varf varfName([uw1,uw2,uw3,pw,Tw],[v1,v2,v3,q,TT])
= int3d(meshName,qforder=ord)( b(uw,q) - gamma*pw*q
+ c1*[uw1,uw2,uw3,Tw]'*[v1,v2,v3,TT]
+ c(uw,u,v) + c(u,uw,v) + IRe*a(uw,v) + b(v,pw)
- dfB(T)*Tw*v3 - Amushy(T)*[uw1,uw2,uw3]'*[v1,v2,v3]
- dAmushy(T)*Tw*[u1,u2,u3]'*[v1,v2,v3]
+ ugrad(u,Tw)*TT + ugrad(uw,T)*TT + grad(Tw)'*grad(TT)*IPr
+ c1*dS(T)*Tw*TT )
+ bcu1 + bcu2 + bcu3 + bcT;
// EOM

macro VarfrhsPCM(varfName, meshName, VhName)
varf varfName([uw1,uw2,uw3,pw,Tw],[v1,v2,v3,q,TT])
= int3d(meshName,qforder=ord)( c(u,u,v) + ugrad(u,T)*TT
- [u1,u2,u3]'*[v1,v2,v3]*dAmushy(T)*T- dfB(T)*T*v3
+ fB(T)*v3 - c2*[u1p,u2p,u3p,Tp]'*[v1,v2,v3,TT]
- c3*[u1pp,u2pp,u3pp,Tpp]'*[v1,v2,v3,TT]
+ c1*(dS(T)*T*TT - S(T)*TT)
- c2*S(Tp)*TT - c3*S(Tpp)*TT )
+ bcu1 + bcu2 + bcu3 + bcT;
// EOM
```

New variables and coefficients can be easily identified. For example, `c1, c2, c3` are the coefficients (depending on δt) corresponding to the Gear scheme, `Amushy` is the Carman–Kozeny penalty term, `S` the enthalpy source term, etc.

New terms can be added to the variational formulation expressed in the `varf` structure, without affecting other parts of the program. The implementation of new models or numerical methods for this problem is greatly facilitated by this modular structure of programs.

3.2. *Domain decomposition method*

The main time consuming part in the algorithm (23) lies in the solution of the sequence of linear systems of the form $Ax = b$. To reduce the computational time, we implement an overlapping Schwarz domain decomposition method to solve these linear systems in parallel. This task becomes an easy job by using the parallel framework `ffddm`, which was recently made available in FreeFem++. `ffddm` is a set of high-level FreeFem++ `macros` that the user can call in the script to perform different steps needed for the parallel solution of a linear system using a domain decomposition preconditioner. We recall these steps below and give the corresponding `ffddm` macro calls that are used in the implementation of our toolbox. The detailed description of these macros can be found in the online `ffddm` documentation [22].

The first step in using `ffddm` is the partition of the global mesh $\mathcal{T} := \mathcal{T}_h$ into N_S non-overlapping meshes $\{\mathcal{T}_i\}_{1\le i\le N_S}$ (see Fig. 1b). Standard graph partitioners available in FreeFem++, such as `Scotch` [23] or `Metis` [24], can be used for this task. If δ is a positive integer, the *overlapping* decomposition $\{\mathcal{T}_i^\delta\}_{1\le i\le N_S}$ is defined recursively as follows: $\mathcal{T}_i^\delta$ is obtained by including all mesh elements of $\mathcal{T}_i^{\delta-1}$ and adding recursively one layer of elements. For $\delta = 0$, $\mathcal{T}_i^\delta = \mathcal{T}_i$ and a non-overlapping decomposition is obtained. In practice we use $\delta = 1$ which corresponds to an overlap of width 2. The mesh partitioning step is performed in macro `ffddmbuildDmeshAug`.

`Metis` is the default partitioner. `Scotch` can be selected by simply declaring `ffddmpartitioner` = 2. Each MPI process i then holds the local mesh $\mathcal{T}_i^\delta$ corresponding to subdomain i.

The second important step is to build an appropriate preconditioner for the linear system. Several types of preconditioners are available in `ffddm`. For the mathematical background of domain decomposition methods and associated preconditioners, we refer to [25]. In the present toolbox, we use the Restricted Additive Schwarz (RAS) preconditioner that proved very efficient. We describe below the main ideas of the corresponding algorithm. Let $W_h = \boldsymbol{V}_h \times Q_h \times V_h$ be the global finite element space for velocity, pressure and temperature variables. The mesh decomposition induces a natural decomposition of the global space W_h on $\mathcal{T}$ into N_S local finite element spaces $\{W_i^\delta\}_{1\le i\le N_S}$, each of them defined on the corresponding local mesh $\mathcal{T}_i^\delta$. Consider the restrictions $\{R_i\}_{1\le i\le N_S}$ from W_h to $\{W_i^\delta\}_{1\le i\le N_S}$, and a local partition of unity $\{D_i\}_{1\le i\le N_S}$ such that

$$\sum_{i=1}^{N_S} R_i^T D_i R_i = I_{n\times n}. \tag{26}$$

At the algebraic level, if n is the global number of unknowns and $\{n_i\}_{1\le i\le N_S}$ are the numbers of unknowns for each local finite element space, then R_i is a Boolean matrix of size $n_i \times n$, and D_i is a diagonal matrix of size $n_i \times n_i$, for all $1 \le i \le N_S$. Note that R_i^T, the transpose of R_i, is a $n \times n_i$ matrix that gives the extension by 0 from W_i^δ to W_h. The construction of the local finite element spaces $\{W_i^\delta\}_{1\le i\le N_S}$ and partition of unity matrices $\{D_i\}_{1\le i\le N_S}$ are implemented in the macro `ffddmbuildDfespaceAug`.

Using these matrices, we define the RAS preconditioner as:

$$M_{\text{RAS}}^{-1} = \sum_{i=1}^{N_S} R_i^T D_i A_i^{-1} R_i, \tag{27}$$

with local subdomain matrices $\{A_i\}_{1\le i\le N_S} = \{R_i A R_i^T\}_{1\le i\le N_S}$. The preconditioner (27) is naturally parallel since its assembly requires the concurrent factorization of each $\{A_i\}_{1\le i\le N_S}$. In practice, this operation is performed locally on different processes in a distributed computing context, as one subdomain is assigned to each MPI process. Likewise, applying (27) to a distributed vector only requires peer-to-peer communications between neighboring subdomains, and a local forward elimination and backward substitution (see Chapter 8 of Dolean et al. [25], for a more detailed description). Local matrices $\{A_i\}_{1\le i\le N_S}$ are also used to perform the parallel matrix–vector product $A * v$ that are main operations in a parallel GMRES algorithm with preconditioner M_{RAS}^{-1}. The parallel assembly and factorization of local matrices $\{A_i\}_{1\le i\le N_S}$ are performed in the macro `ffddmsetup`, that also defines the parallel matrix–vector product and preconditioner operators. Additionally, the macro `ffddmbuildrhs` computes the distributed right-hand side $\{b_i\}_{1\le i\le N_S} = \{R_i b\}_{1\le i\le N_S}$. Macros `ffddmsetup` and `ffddmbuildrhs` use the corresponding variational `varf` formulations of the problem defining the bilinear and linear parts, respectively. Finally, the linear system $Ax = b$ is solved in parallel with a GMRES algorithm, called in the function `fGMRES`. We apply in our algorithms a left preconditioning by M_{RAS}^{-1}. The error tolerance ε_G in GMRES is variable and adapted to the convergence of the Newton algorithm by monitoring $\varepsilon_N = \|\boldsymbol{w}_{k+1} - \boldsymbol{w}_k\|$. We set $\varepsilon_G = 10^{-10}$ if $\varepsilon_N < 10^{-4}$, $\varepsilon_G = 10^{-9}$ if $10^{-4} < \varepsilon_N < 10^{-2}$ and $\varepsilon_G = 10^{-2}$ if $\varepsilon_N > 10^{-2}$. The output of the `fGMRES` function is the distributed solution $\{x_i\}_{1\le i\le N_S} = \{R_i x\}_{1\le i\le N_S}$. The macro `fromVhi` is then used to recover the global solution x from the distributed solution. The global solution is finally used for visualization and also in the mesh adaptation procedure.

A detailed description of the `ffddm` calls used for the different steps of the domain decomposition method can be found in the User's manual of `ffddm` [22]. In the entire simulation process, the parallel algorithm described above to assemble and solve the linear system is performed at each time step and at each Newton iteration. However, the first two steps concerning the construction of the overlapping mesh and the finite element space decomposition are performed only once at the beginning of each time step. These steps are reiterated only if a mesh change occurs, after calling the mesh adaptivity procedure (see next section).

3.3. Mesh adaptivity

For 2D simulations, we use the standard mesh adaptivity function (`adaptmesh`) offered by FreeFem++ [39]. The key idea implemented in this function (see also [43–48]) is to use the Delaunay algorithm to generate a *new* triangular mesh with edges close to the unit length in the metric $\mathcal{M} = \frac{|\mathcal{H}|}{\mathcal{E}}$, where $|\mathcal{H}(x)|$ is the Hessian of the variable χ at point x (after being made positive definite) and $\mathcal{E}$ the interpolation error for χ. This implies to modify the scalar product used in the automatic mesh generator to evaluate distance and volume by defining a scalar product based on the evaluation of the Hessian $\mathcal{H}$ of the variables of the problem. Equilateral elements are thus constructed, with an equally distributed interpolation error $\mathcal{E}$ over the edges of the mesh. The previous approach could be generalized for a vector variable $\chi = [\chi_1, \chi_2]$. After computing the metrics $\mathcal{M}_1$ and $\mathcal{M}_2$ for each variable, the retained metric is the intersection $\mathcal{M} = \mathcal{M}_1 \cap \mathcal{M}_2$, defined such that the unit ball of $\mathcal{M}$ is included in the intersection of the two unit balls of metrics $\mathcal{M}_2$ and $\mathcal{M}_1$ (for details, see the procedure defined in [47]).

The use of the mesh adaptivity algorithm for 2D phase-change systems is described in detail in [3]. In 2D problems considered below, we took into account several metrics computed from different variables monitoring the evolution of the phase-change system. For the natural convection system, the mesh was adapted using the values of the two velocity components and the temperature. For phase-change systems, to accurately track the solid–liquid interface we added the variation of the enthalpy source term in the adaptivity criterion. For water systems (convection or freezing), we also added an extra function tracking the anomalous change of density around 4^oC. To reduce the impact of the interpolation on the global accuracy for time-depending problems, we considered, for each variable used for adaptivity, the metrics computed at actual (t_{n+1}) and previous (t_n) time instants (see also [17]). The capabilities of the mesh adaptivity algorithm in 2D are illustrated in Section 5.

For 3D simulations, the metric was first computed using `mshmet`, which is a FreeFem++ module, following the code:

```
load "mshmet"
real[int] hmetric(6*ThBackup.nv);
hmetric=mshmet(ThBackup,[u1Backup,u2Backup,u3Backup],normalization=1,aniso=1,nbregul=1,
   hmax=hmax,hmin=hmin,err=errh);
```

Several parameters of the mesh can be thus controlled (h_{min}, h_{max}, anisotropy, etc.). Then, the metric was regularized in order to avoid flat tetrahedra and saved in m11[] :

```
load "aniso"
boundaniso(6,hmetric,40);
fespace Wh6Backup(ThBackup,[P1,P1,P1,P1,P1,P1]);
Wh6Backup [m11,m21,m22,m31,m32,m33];
m11[]=hmetric;
```

The external software mmg [26] was finally loaded to read the saved metric and the old mesh to finally generate the adapted new mesh:

```
load "mmg"
ThBackup = mmg3d(ThBackup,metric=m11[],verbose=0,hmin=hmin,hmax=hmax,hgrad=adaptratio,mem
    =10000);
```

4. Description of the programs

In this section, we first describe the architecture of the programs and the organization of the provided files. Then we focus on the list of input parameters and the structure of output files.

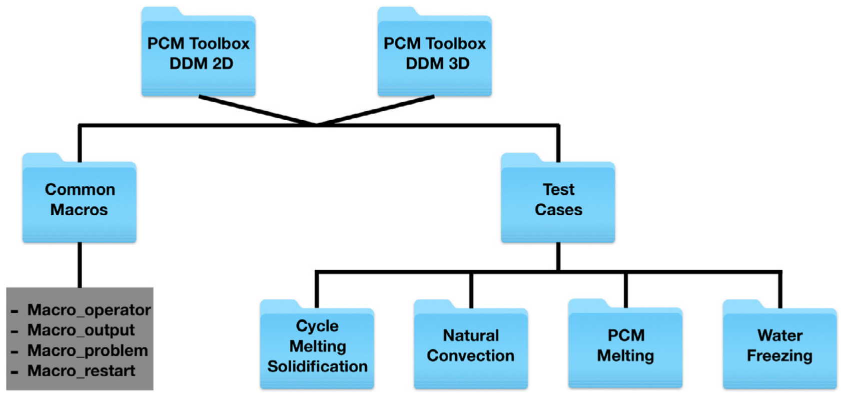


Fig. 2. Folder tree structure of the PCM_Toolbox_DDM_2D and PCM_Toolbox_DDM_3D to solve solid–liquid phase-change problems.

4.1. Program architecture

Figure 2 gives a schematic overview of the content of the Toolbox. To facilitate the reading and the assimilation of the programs, we separated the 2D and 3D Toolboxes in directories PCM_Toolbox_DDM_2D and PCM_Toolbox_DDM_3D. Many detailed comments were included in programs, with direct reference to the mathematical expressions used in this paper. The used FreeFem syntax was intentionally kept at a low level of technicality and supplemented with detailed comments when specific more technical syntax was used.

This directory is organized as follows:

1. The directory Common_Macros contains five files:
   - *Macro_operator.idp* includes macros and functions defining mathematical operators,
   - *Macro_problem.idp*: macros defining the variational formulation of the problem,
   - *Macro_restart.idp*: macros used to start a new simulation from a saved field,
   - *Macro_output.idp*: macros used to save the solution with different formats.
2. The directory Test_Cases contains four subdirectories, each of them defining one of the following applications:
   - natural convection of air or water in a differentially heated square/cubic cavity,
   - melting of a PCM stored in containers of different shapes,
   - melting followed by solidification of a rectangular/cubic PCM,
   - freezing of pure water in a square/cubic cavity.

   Each subdirectory contains three files: NEWTON_$case.edp* is the main FreeFem++script file, *param_phys.inc* defines the physical parameters and *param_num.inc* the numerical parameters. The 3D Toolbox is supplemented with the script *Read*_3D_data.edp, which reads data from the RST folder and generates the files for visualization with Medit, Paraview or Tecplot. For example, to run the natural convection case of air in a square cavity, the user can use the following command in a terminal window: mpirun -np 6 FreeFem++-mpi NEWTON_stat_natconv_ddm.edp -wg -v 0 -ns.
   The folder structure of each test case is illustrated in Fig. 3. The obtained solutions are saved in the folder OUTPUT/Data. Depending on the output format selected by the user, data files are generated in specific folders for being visualized with Tecplot, Paraview, Gnuplot or Medit. We also provide in the folder Figures ready-made layouts for each visualization software. The user can thus obtain the figures from this paper using newly generated data. More details about the output structure are given below.

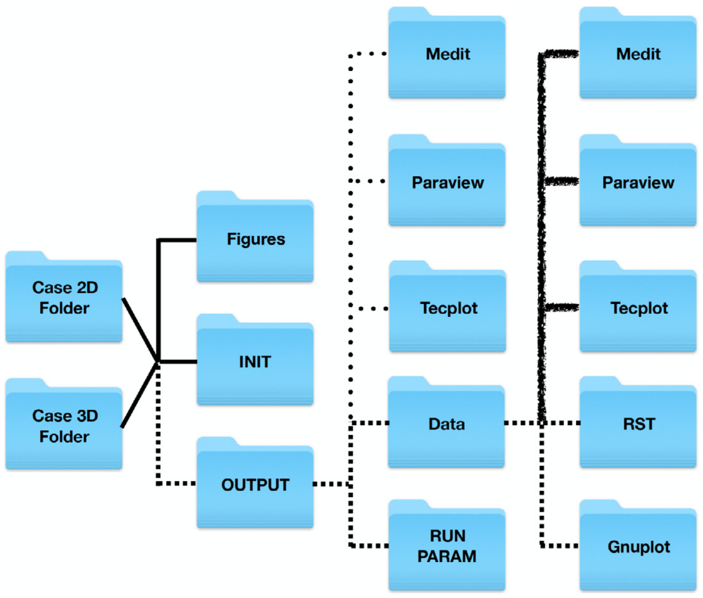


Fig. 3. Structure of each Test-case folder. Solid lines in the diagram correspond to folders which come with the Toolbox, the dashed lines to folders generated after running the script *NEWTON_case.edp*, the hashed lines to folders generated after running the script *NEWTON_case.edp* in the 2D case. Finally, the dotted lines indicate folders generated after running the script *Read_3D_data.edp*, which reads data from the RST folder and generates the Medit, Paraview and Tecplot files for visualization.

4.2. Input parameters

Physical parameters and parameters related to the run are separated into two files.
(1) The file *param_phys.inc* contains the physical descriptions of the problem:

- **typeT**: is the finite-element type for the temperature, with possible values P2 or P1,
- **Torder**: is the order of accuracy of the time integration scheme, with possible values 1 (Euler scheme) or 2 (Gear scheme),
- **scalAdim**: defines the characteristic scales of the problem, see (4). Possible values 1, 2 or 3 correspond to the following choice of the characteristic scales [19]:

$$(1):\ V_{ref}^{(1)}=\frac{\nu_l}{H}\Longrightarrow t_{ref}^{(1)}=\frac{H^2}{\nu_l}\Longrightarrow \mathcal{R}e=1, \tag{28}$$

$$(2):\ V_{ref}^{(2)}=\frac{\alpha}{H}\Longrightarrow t_{ref}^{(2)}=t_{ref}^{(1)}\mathcal{P}r\Longrightarrow \mathcal{R}e=1/\mathcal{P}r, \tag{29}$$

$$(3):\ V_{ref}^{(3)}=\frac{\nu_l}{H}\sqrt{\frac{\mathcal{R}a}{\mathcal{P}r}}\Longrightarrow t_{ref}^{(3)}=t_{ref}^{(1)}\sqrt{\frac{\mathcal{P}r}{\mathcal{R}a}}\Longrightarrow \mathcal{R}e=\sqrt{\frac{\mathcal{R}a}{\mathcal{P}r}}, \tag{30}$$

- $\mathbf{x}_l, \mathbf{x}_r, \mathbf{y}_l, \mathbf{y}_r$: are the values defining the dimensions of the cavity $[x_l, x_r]\times[y_l, y_r]$,
- **Pr, Ra, Ste**: are the Prandtl, Rayleigh and Stefan numbers, see (6) and (5),
- $\mathbf{T}_{hot}, \mathbf{T}_{cold}$: are dimensionless temperatures according to (4),
- $\mathbf{bcu}_1$, $\mathbf{bcu}_2$, **bcT**: are macros defining the velocity (u) and the temperature (T) boundary conditions,
- **epsi**: is the half width ε of the mushy region. *Default value* $= 0.01$,
- **dt**: is the dimensionless time step,
- $\mathbf{t}_{max}$: is the dimensionless final time,
- **Parameters for regularization functions**:
The parameters of the hyperbolic-tangent function (9) used to regularize discontinuous functions are set by default as follows:

	f_s	f_l	a_s	θ_s	R_s	C_{CK}	b
Enthalpy	0	1/Ste	1	0.01	0.01	–	–
Carman–Kozeny	0	1	1	0.01	0.01	10^6	10^{-7}
Conductivity (water)	1	2.26/0.578	1	θ_f	0.015	–	–

- **rho(T) and Drho(T)**: (water cases only) define the density and its derivative as functions of the temperature, following the model [49]:

$\rho(T)=\rho_m(1-\omega\lvert T-T_m\rvert^q)$,			
ρ_m [kg/m^3]	ω [°C^{-q}]	q	T_m [°C]
999.972	$9.2793\cdot 10^{-6}$	1.894816	4.0293

- $\mathbf{f}_B$**(T)**, $\mathbf{df}_B$ **(T)**: define the buoyancy force and its derivative.

(2) The file *param_num*`.inc` contains the parameters controlling the run.
Restart parameters:

- **Nsave**: the solution is saved every *Nsave* time steps in the `Data` folder (see Fig. 3). The temperature and velocity fields are saved in `Tecplot` and `Medit` folders, while the liquid fraction, the Nusselt number, and the accumulated heat input are saved in the `Gnuplot` folder.
- **Nrestart**: restart files (mesh and solution) are saved every *Nrestart* time steps. Solutions at current and previous iterations, the CPU time, the accumulated heat input Q_0, and the time step dt are saved in the folder `RST`.
- **Ncondt**: allows the user to stop the run and save the solution properly. The file `OUTPUT/zz.condt` is read every *Ncondt* time steps: if the user replaces the value "0" in this file by "1" the run is stopped. This is a simple solution for a clean stop of the job by the user. *Default value* = 20.
- **Nremesh**: the mesh is adapted every *Nremesh* iterations. If this parameter is set to "1" the mesh is adapted every time step.
- **IFrestart**: is a Boolean controlling the set up of the initial field.
  IFrestart = 0, the initial condition is built in the code for each test case. For the PCM melting cases, the PCM is initially motionless at isothermal temperature. To set-up a smooth initial field, a few time steps (with very small δt) are computed by increasing progressively the boundary temperature at the hot wall and the Rayleigh number (by continuation). Outputs are saved in `OUTPUT/Data-RST-0`.
  IFrestart > 0, (positive integer values) the solution field previously computed at iteration *IFrestart* is loaded from the folder `OUTPUT/Data-RST-filenameRST/RST`, with `filenameRST` a variable selecting the restart folder. Note that for 3D cases, the user has to copy manually the solution used as restart field to the folder `INIT`.
  IFrestart < 0, (negative integer values), the same principle for loading a solution is used, but from the folder `INIT` (see Fig. 3). The solution fields stored in this folder could come from different previous calculations (*e.g.* a steady state solution or, for the water, the natural convection field before freezing).

Newton parameters:

- **epsconv**: is the value of the stopping criterion for steady cases,
- **gamma**: is the penalty parameter in (13). *Default value* = 10^{-7},
- **tolNewton**: is the Newton tolerance ξ_N (see (23)). *Default value* = 10^{-6},
- **newtonMax**: limits the maximum number of iterations in the Newton algorithm (23). *Default value* = 50.

Mesh parameters:

- **nbseg**: is the number of segments for the discretization along the x and y directions,
- **errh**: is the interpolation error level. *Default value* = 0.02,
- **hmin, hmax**: are the minimum and the maximum edge size, respectively,
- **adaptratio**: is the ratio for a prescribed smoothing of the metric. For a value less than 1.1 no smoothing is done. *Default value* = 1.5,
- **nbvx**: is the maximum number of vertices allowed in the mesh generator. *Default value* = 50 000.

Output parameters:

- **dircase**: is the name of the output folder,
- **fcase**: is the prefix-name for output files.
- **Paraview, Tecplot, Medit, Gnu**: correspond to the name of the visualization software to be used; the format of the outputs written in `OUTPUT/Data` (see Fig. 3) is set accordingly.

4.3. Outputs

When a computation starts, the `OUTPUT` directory is created (see Fig. 3). It contains two folders storing the output data and the echo of the run parameters. The folder `Data` contains four subdirectories with different output format files of the computed solution. File names are created using the prefix defined by the parameter **fcase**, the current iteration and the current dimensionless time t. Solution files can be visualized using either Tecplot or any other CFD Visualization tools (Paraview, Visit, etc.). Moreover, *.gmsh* (mesh) and *.rst* (fields) files are generated in the folder `RST` to enable restarts of the computation. Note that the folder `FFglut` contains FreeFem++scripts that read and visualize the RST-files to facilitate the selection of a restart field. An *.echo* file with a summary of the main parameters, information on the run and the names of the output files is saved in the folder `RUNPARAM`. This directory additionally contains a copy of the *.inc* parameter files, allowing an easy identification of each case and preparing an eventual rerun of the same case.

5. 2D parallel simulations

In this section we consider well defined 2D benchmarks used to validate numerical codes for natural convection and phase-change problems. All these cases were described in detail in [3] and computed using sequential codes. We present below the results obtained for the same cases, but using parallel computations with the new Toolbox: (i) the natural convection of air (Section 5.1), (ii) the melting or solidification of a phase-change material (PCM) (Section 5.2) and (iii) the convection and the freezing of pure water (Section 5.3). This approach allows us to test the programs by adding progressively non-linearities in the Newton algorithm. In the following, we validate each case with respect to the physical reference data available in the literature. To have a quick overview of the benefits of using parallel computing, we compare in Table 1 computational (CPU) times necessary to run sequential and parallel computations of

Table 1
Comparison of CPU times for 2D simulations performed with one or 6 MPI processes. Speed-up of the parallel computation, maximum number of triangles of the adapted mesh, total number of time steps and ratio between the CPU time for adapting the mesh and the CPU time for a complete time step. All computations were performed on a Macbook pro 2.2 GHz Intel Core i7, 16 GB of DDR4 2400 MHz RAM.

Case	CPU time 1 MPI proc.	CPU time 6 MPI procs.	Speed-up	Max number of triangles	Number of time steps	Ratio CPU adapt/CPU time step
Unsteady NC of air, $Ra = 10^6$	00:01:39	00:00:50	1.98	2902	53	18%
PCM-Case #1	00:06:39	00:04:35	1.45	2590	81	5%
PCM-Case #2	01:31:11	00:40:44	2.23	5928	501	8%
PCM-Case #3: 1 tube	00:25:09	00:19:18	1.3	5052	401	6%
PCM-Case #3: 4 tubes	00:28:57	00:16:13	1.78	13,507	151	6%
PCM-Case #3: 9 tubes	00:33:28	00:17:44	1.88	24,866	100	5%
PCM-Case #4	06:31:31	01:38:35	3.97	5371	2001	15%
PCM-Case #5	00:07:52	00:03:17	2.39	2898	117	12%
Cycle of a PCM	00:13:58	00:08:49	1.58	3160	383	17%
NC of water	00:00:36	00:00:21	1.71	2686	20	14%
Unsteady NC of water	00:01:49	00:00:54	2.02	3368	33	10%
Water freezing	00:22:49	00:13:38	1.67	3743	296	10%

each case (with default parameters) on a personal computer. We also display the maximum number of triangles of adapted mesh and the ratio between the CPU time required to adapt the mesh to the CPU time for a complete time step. Mesh adaptation is performed every time step. To make the simulations accessible and testable by using a personal computer with a multi-thread processor, we use in this section only 6 MPI processes.

5.1. Natural convection of air

We start by testing the Newton algorithm (20)–(22) for the case of natural convection, *i.e.* $C = K = 1$, $A_{mushy}(\theta) = S(\theta) = 0$. We consider the classical problem of the thermally driven square cavity $[0, 1]\times[0, 1]$, filled with air. The Boussinesq term $f_B(\theta)$ is then linear and takes the form (5). Top and bottom walls are adiabatic, while the temperature is fixed on the left (hot) wall and the right (cold) wall. Natural convection flows are computed for three values of the Rayleigh number: $Ra = 10^4, 10^5, 10^6$. The Prandtl number is set to $Pr = 0.71$. For these values of the Ra number, the flow is known to be steady [50] and therefore, we also solve the time-independent version of Eqs. (20)–(22).

We provide programs for both steady (time-independent) and time-dependent cases. The steady case is performed using a continuation following the Rayleigh number: a smaller value for Ra is set initially and is then smoothly increased until reaching the wanted value. The time-dependent case is computed until a steady state with a single convection cell is reached.

The temperature is imposed at the right (cold) wall as $\theta_c = -0.5$ and at the left (hot) wall as $\theta_h = 0.5$. Top and bottom walls are adiabatic. The initial condition models a cavity filled with motionless air ($\boldsymbol{u} = 0$), with a linear distribution of the temperature. Both steady and time-dependent codes converge to the same flow state, with a single convection cell. For this final state, horizontal $u(y)$ and vertical $v(x)$ velocity profiles were extracted at mid-domain ($y = 0.5$ and $x = 0.5$, respectively) and plotted in Fig. 4. Our results are in very good agreement with reference numerical results obtained by Le Quéré [50] with a spectral code.

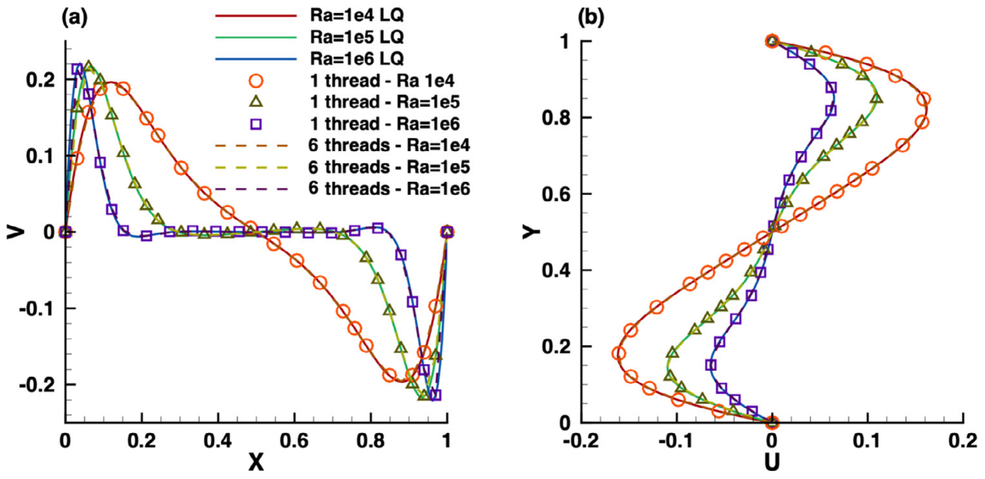


Fig. 4. Natural convection of air in a differentially heated cavity for values of Ra ranging from 10^4 to 10^6. (a) Vertical velocity profile $v(x)$ along the horizontal symmetry line ($y = 0.5$). (b) Longitudinal velocity profile $u(y)$ along the vertical symmetry line ($x = 0.5$). Results obtained using the present Toolbox, with a mesh resolution $nbseg = 80$. Comparison with the spectral simulations by Le Quéré [50] (solid lines).

Table 2 offers a quantitative assessment of the accuracy of the present Newton method. The values of u_{max} and the location Y of this maximum are compared to reference values from [50]. The Newton method gives results identical to reference values, with a difference less than 0.1%.

Table 2
Natural convection of air in a differentially heated cavity. Maximum value u_{max} of the horizontal velocity profile at mid-domain ($x = 0.5$) and location Y of this maximum. Comparison to reference values reported by Le Quéré [50].

Run		u_{max} at x = 0.5 (error)	Y (error)
Reference values	spectral	0.0648344	0.850
Newton (Steady)	$nbseg = 80$	0.0648126 (0.03%)	0.850394 (0.05%)
Newton (Unsteady)	$nbseg = 80$	0.0647805 (0.08%)	0.850394 (0.05%)

5.2. *Melting or solidification of a phase-change material (PCM)*

We continue our validation tests by considering the full system (20)–(22) for the case of the melting or solidification of a phase-change material. Two new non-linearities are now present in the system: the Carman–Kozeny penalty term $A_{mushy}(\theta)$ and the enthalpy source term $S(\theta)$. The function S is regularized using (10). We also consider that the material properties in the liquid and solid are the same, *i.e.* $C = K = 1$. This is a frequent assumption [51,52]. Five cases were computed (the exact values of the defining parameters are summarized in Table 3):

- PCM-Case #1 simulates the experimental study of Okada [53]. It consists of a differentially heated square cavity, filled with octadecane paraffin.
- PCM-Case #2 is extracted from the collective publication by Bertrand et al. [54], compiling the results of different numerical approaches used for the simulation of the melting of a PCM at high Rayleigh numbers.
- PCM-Case #3 simulates the melting of a cylindrical PCM with heated inner tubes, as in [7].
- PCM-Case #4 simulates the melting of Gallium in a rectangular cavity heated by the side-wall, as in [10].
- PCM-Case #5 simulates the solid crust formation in a highly distorted PCM domain, as in [55].

We summarize in Table 3 the values of physical parameters and the scales used to simulate these cases. The values of the CPU time necessary to run each case (with default parameters) are given in Table 1.

Table 3
Parameters for the cases simulating the melting of a phase-change material.

	Case #1	Case #2	Case #3	Case #4	Case #5
Ra	$3.27 \cdot 10^5$	10^8	$5 \cdot 10^4$	$7 \cdot 10^5$	10^6
Pr	56.2	50	0.2	0.0216	0.1
Ste	0.045	0.1	0.02	0.046	4.854
δt	1	10^{-2}	10^{-2}	10^{-5}	1
V_{ref}	$\frac{\nu_l}{H}$	$\frac{\nu_l}{H}$	$\frac{\nu_l}{H}$	$\frac{\nu_l}{H}$	$\frac{\nu_l}{H}\sqrt{\frac{Ra}{Pr}}$

5.2.1. *PCM-case #1: Melting of an octadecane PCM in a square cavity*

Okada [53] studied experimentally the melting of an octadecane PCM in a square cavity of height $H = 1.5$ cm. His results were often used to validate numerical methods [19,51–53]. The material is initially solid ($\theta_0 = -0.01$) and melts progressively starting from the left boundary, maintained at a hot temperature $\theta_h = 1$. The right boundary is also isothermal, with cold temperature $\theta_c = -0.01$. Horizontal boundaries are adiabatic. The other parameters of this case are reported in Table 3. The computation starts from a refined mesh near the hot boundary. Mesh adaptivity is applied every time step. Fig. 5 shows in the left panel the adapted mesh at $t = 80.1$ and the 6 subdomains of the domain decomposition used for the parallel computation. The velocity vectors in the liquid phase, represented in the middle panel, show the strong convection cell affecting the shape of the liquid–solid interface. To offer a quantitative validation of this computation, we compare in Fig. 5 (right panel) the position of the solid–liquid interface at two time instants. Since $\delta t = 1$ in the simulation, we consider $t = 40.1$ and $t = 79.1$ and compare with experimental data of Okada [53] taken at slightly different time instants, $t = 39.9$ and $t = 78.7$. The obtained shape and position of the liquid–solid interface are very close to experimental results. We show in the same figure that the parallel computation (6 MPI processes) gives the same results as the sequential simulation (1 MPI process).

5.2.2. *PCM-Case #2: Melting of an octadecane PCM with high Rayleigh number*

This case considers the same problem of the melting of a PCM, but with a very high value of the Rayleigh number, $Ra = 10^8$ (see Table 3). This case is very challenging since the natural convection becomes important in the fluid flow, and enhances considerably the heat transfer. Bertrand et al. [54] compiled results provided by five different authors (Lacroix, Le Quéré, Gobin-Vieira, Delannoy and Binet-Lacroix) who used different numerical methods to compute the basic configuration presented in this section. This benchmark is a very demanding numerical test. The high velocity, inducing a very narrow thermal boundary layer, can lead to unrealistic results if under-resolved. This explains why two investigators among the five failed to correctly predict the process.

We show in Fig. 6 an illustration of the flow configuration at dimensionless time $t = 5$. The adapted mesh and the domain decomposition for the parallel computation with 6 MPI processes is presented in the left panel. The convection flow is depicted in the middle panel. Note the very narrow boundary layers, which impose a very refined mesh in these regions. This flow particularity explains why this case is rather costly in terms of computational time (see Table 1), when compared to the previous one. The right panel of Fig. 6 offers a quantitative validation of this computation. The position of melting front at $t = 5$ is in good agreement with that reported by Gobin and Le Quéré. Details of their numerical method are presented in [56]. Gobin used a front-tracking method based on a coordinate transformation with a finite volume method and a 62×42 grid, while Le Quéré solved a single domain model using a second-order finite volume method with a 192×192 grid. The interest of the mesh adaptation is clearly demonstrated for this case, since we initially used a coarse 40×40 grid.

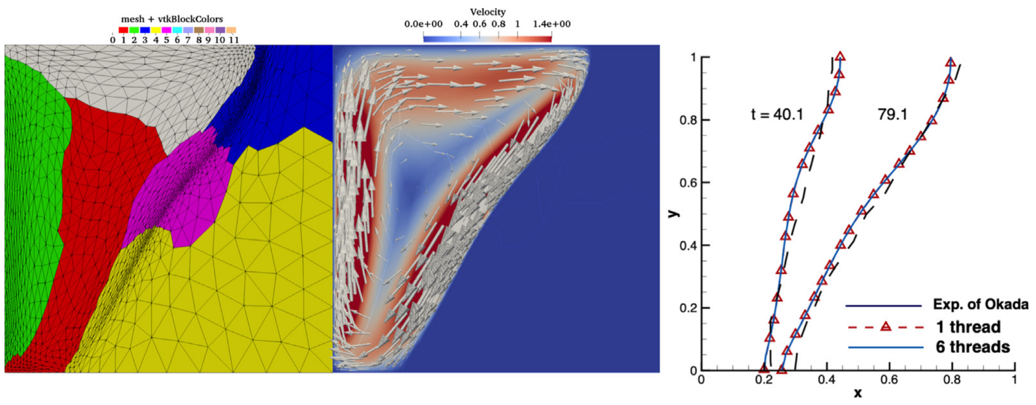


Fig. 5. PCM-Case #1. Melted PCM at time instant $t = 80.1$. Adapted finite-element mesh and subdomains used for the domain decomposition and parallel computing with 6 MPI processes (left panel). Velocity vectors in the liquid phase (central panel). Position of the solid–liquid interface and comparison with experimental data of Okada [53] for two time instants (right panel).

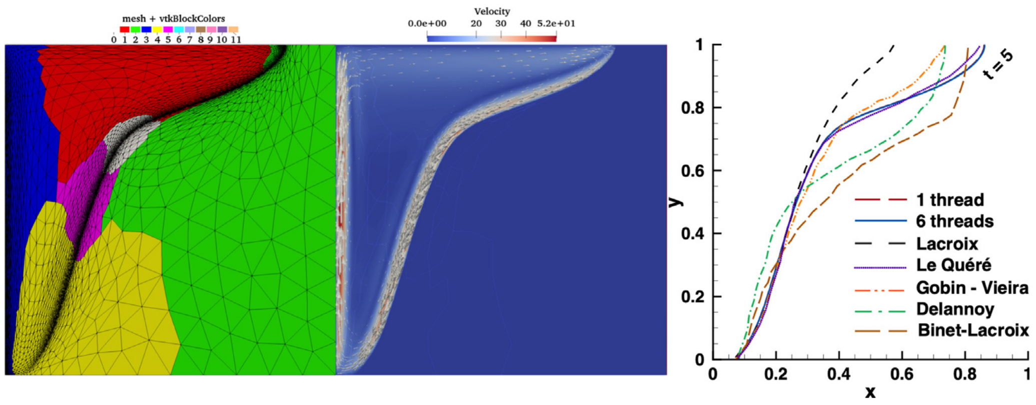


Fig. 6. PCM-Case #2. Melting of the PCM with high Rayleigh number ($Ra = 10^8$). Simulated configuration at time instant $t = 5$. Adapted finite-element mesh and subdomains used for the domain decomposition and parallel computing with 6 MPI processes (left panel). Velocity vectors in the liquid phase (central panel). Position of the solid–liquid interface and comparison with five sets of results presented in [54].

5.2.3. *PCM-Case #3: Melting of cylindrical PCM with inner heated tubes*

A more complex geometry, suggested by Luo et al. [7], is simulated in this section. It consists of a cylindrical PCM of radius $R = 1$ with heated tube inclusions of different arrangements. This configuration is also interesting from a practical point of view. Agyenim et al. [1] pointed out that more than 70% of the PCM containers used for heat storage are using shell-tube systems. We simulate three configurations, with one, four and nine heated tubes (see Fig. 7). The size of the tubes is adjusted to have the same total tube area for all configurations. The radius R_i of the inner tube is $R_i = R/4$ for the case with one tube, $R_i = R/8$ for the four heated tubes case and $R_i = R/12$ for the case with nine tubes. The inner tubes are heated at constant $\theta = \theta_h$ (Dirichlet boundary condition). A Neumann boundary condition ($\partial\theta/\partial n = 0$) is used for the outer boundary. For the velocity, all boundaries are considered as non-slip walls ($\boldsymbol{u} = 0$).

For the three configurations, Fig. 7 shows the adapted mesh and the domain decomposition with 6 subdomains (left column), the velocity vectors (central column) and the temperature field (right column). The time instants are chosen to have the same liquid fraction in the system, $L_f = 80\%$. Only half of the domain is simulated, since all configurations are symmetric with respect to the vertical axis. The mesh is refined initially around the inner tubes, and is dynamically adapted at each time step around the melting front and in the thermal boundary layer area.

To estimate the heat-storage efficiency of each configuration, we plot in Fig. 8 the time evolution of the liquid fraction L_f. Including more heated tubes results in an enhanced heat transfer and a faster melting process. The nine-tube configuration melts 5 times faster than the reference configuration with one tube. Note also from Fig. 8 the good agreement between our results and those reported by Luo et al. [7], obtained using a completely different model based on the Lattice Boltzmann Method. We checked again that the results obtained with 6 MPI processes are identical to those computed with the sequential code.

5.2.4. *PCM-Case #4: Melting of gallium in a rectangular cavity*

The melting of the Gallium in a rectangular cavity is a challenging case that generated an animated debate in the literature, since [57]. Le Quéré and Gobin [58] showed that the flow has to display a multi-cellular structure, resulting from the hydrodynamic instabilities during the conduction regime before the onset of convection. This multi-cellular flow configuration was also found numerically by Hannoun et al. [10]. These authors pointed out that coarse grids or inconsistencies in the mathematical model could generate unphysical flows, with a mono-cellular convection cell. Therefore, this case is a relevant exercise to test the accuracy of our method. The parameters of this case are reported in Table 3. To capture the very small convection cells during the first step of the melting, Hannoun et al. [10] used a 800×1120 fixed grid. With our adaptive method, a maximum of 4820 triangles are necessary to reproduce the

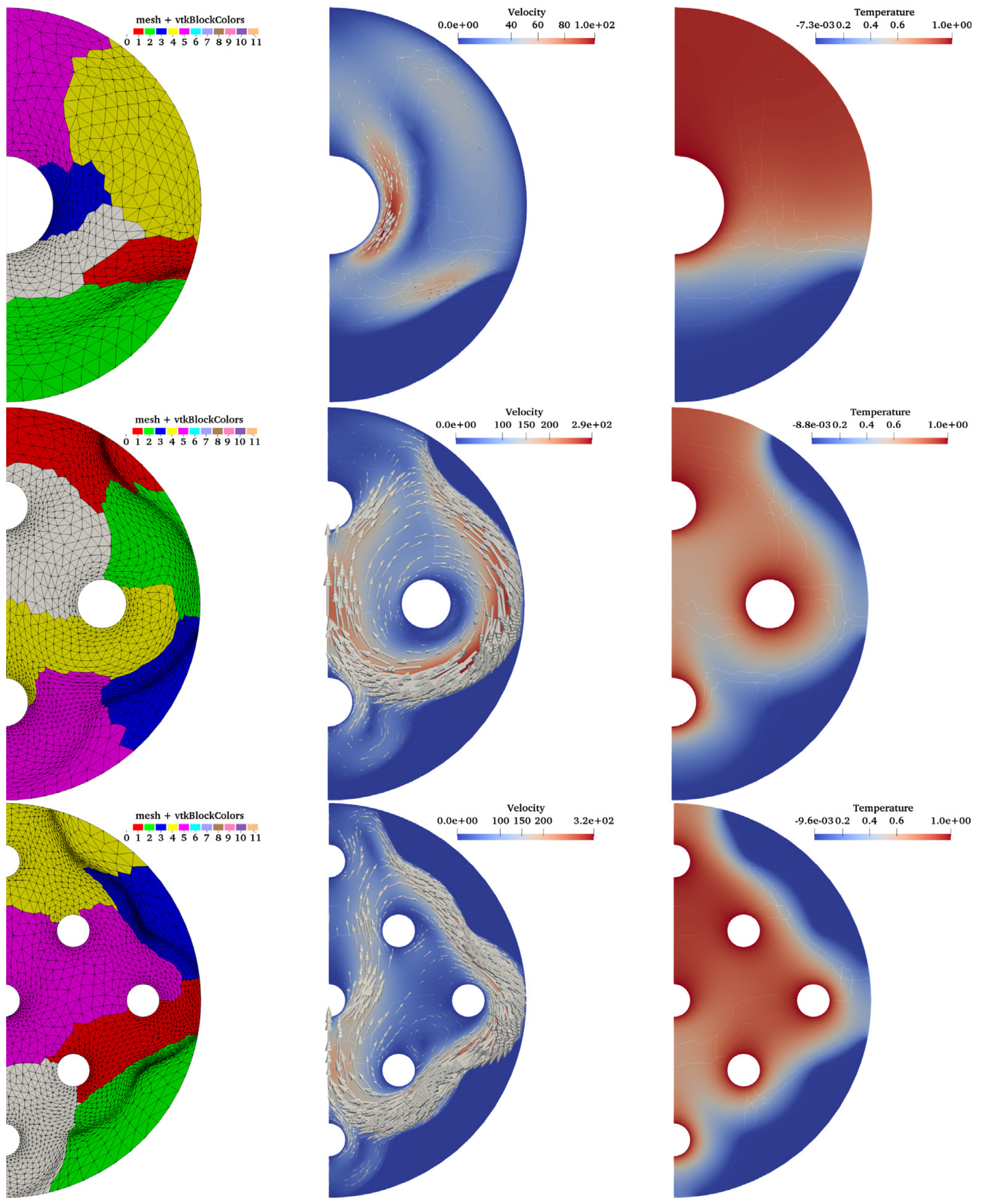


Fig. 7. PCM-Case #3. Melting of a cylindrical PCM with heated inner tubes. Flow configuration for one tube at $t = 2.491$ (first raw), four tubes at $t = 0.981$ (middle row) and nine tubes at $t = 0.411$ (last row). Time instants correspond to the same liquid fraction in the system, $L_f = 80\%$. Adapted mesh and domain decomposition with 6 subdomains (left column), velocity vectors with background color representing the velocity intensity (central column) and temperature field (right column).

numerical result of Hannoun et al. [10]. The grid size is thus reduced with our method by a factor of 100. We illustrate in Fig. 9 the configuration of the phase-change system at dimensionless time $t = 0.02$. Note in the left panel the refined mesh at the liquid–solid interface and also at the borders of each convection cell. The multi-cellular structure of the flow is illustrated by the velocity field presented in the middle panel. The structure of the temperature field in the right panel shows that the rolls of the convection cells are well resolved. The number of cells decreases later through a process of roll merging, as it was also reported by Hannoun et al. [10]. Our numerical results are in good agreement with the observations of Hannoun et al. [10], Cerimele et al. [59] and Giangi and Stella [60].

5.2.5. *PCM-Case #5: Solid crust formation in a highly distorted domain*

This final PCM case considers the solidification of a PCM in a domain with a more complicated shape (Fig. 10). This case was simulated by Nourgaliev et al. [55], using a discontinuous Galerkin finite element method. The fluid is initially motionless, with an initial dimensionless temperature $\theta_0 = 2$. The temperature of fusion is set to $\theta_f = 1.4$, according to [55] (see Table 3 for the values of all parameters). The left side boundary is maintained at a cold temperature $\theta_c = 1.39$ in the initial stage. The right wall is isothermal, with hot temperature $\theta_h = 2$. A nearly steady-state natural circulation is induced in the early time evolution of the flow. Then, the cold temperature at the left wall is decreased to $\theta_c = 1$, below the temperature of solidification. At this point, the formation of a solid

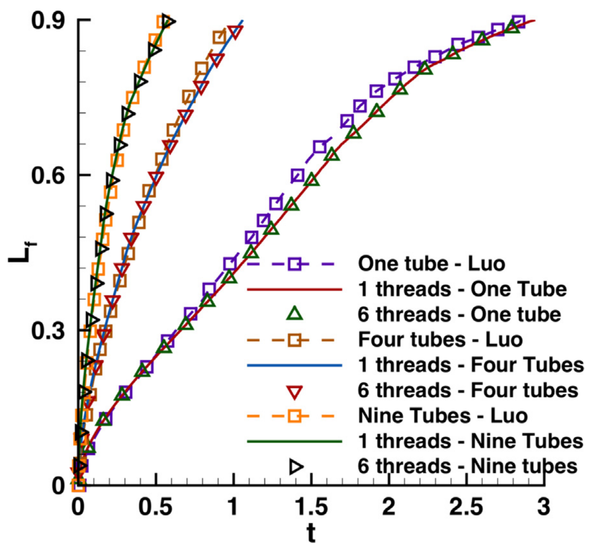


Fig. 8. PCM-Case #3. Melting of a cylindrical PCM with heated inner tubes. Time evolution of the liquid fraction for configurations with one, four, and nine heated tubes. Comparison with numerical results of Luo et al. [7].

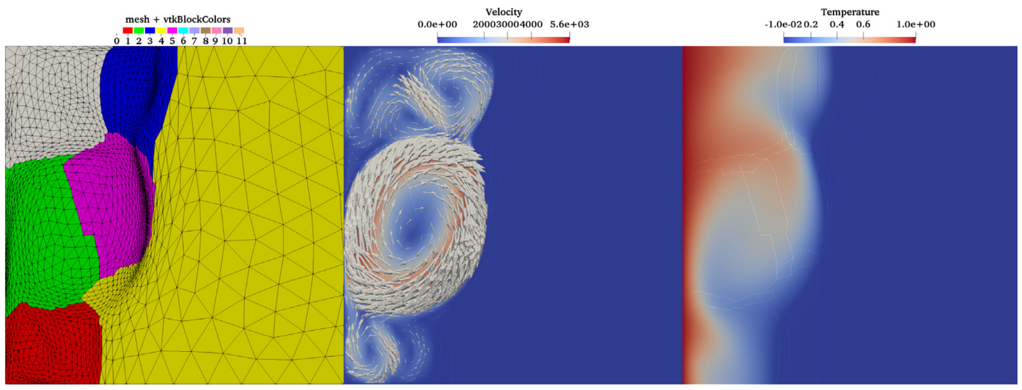


Fig. 9. PCM-Case #4. Melting of Gallium. Simulated configuration at $t = 0.02$. Adapted finite-element mesh and subdomains used for the domain decomposition and parallel computing with 6 MPI processes (left panel). Velocity vectors in the liquid phase (central panel). Temperature field (right panel).

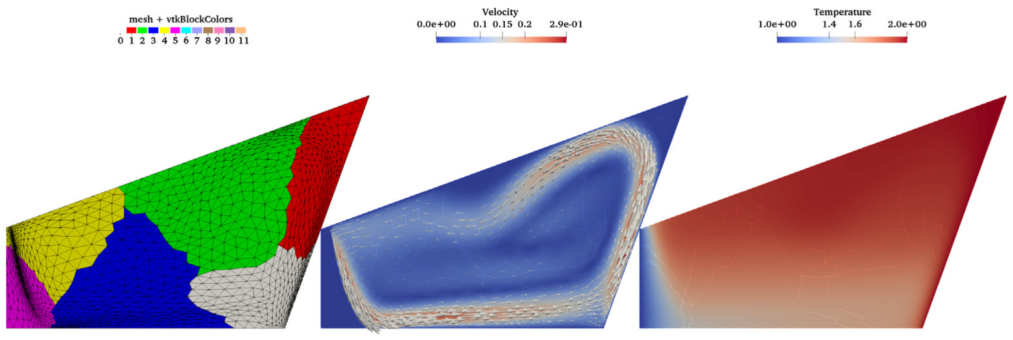


Fig. 10. PCM-Case #5. Solid crust formation in a distorted domain. Simulated configuration at $t = 29.33$. Adapted finite-element mesh and subdomains used for the domain decomposition and parallel computing with 6 MPI processes (left panel). Velocity vectors in the liquid phase (central panel). Temperature field (right panel).

crust layer starts at the left boundary. Figure 10 depicts the configuration of the solid–liquid system at dimensionless time $t = 30$. Mesh adaptivity used metrics computed from the two components of the velocity and from the phase-change variables (enthalpy and temperature). Consequently, it is shown in the left panel that the mesh is well adapted near the solid–liquid front and following the distorted shape of the convection cell in the liquid phase. In the middle panel, we notice the high velocity values defining the convection cell, waving in the liquid phase. The final panel of Fig. 10 shows the temperature field, with the solid phase well identified in blue. This structure of the flow is in a very good agreement with that reported by Nourgaliev et al. [55].

5.3. Natural convection of water and water freezing

In this section we consider phase-change systems using water. Since pure water exhibits non-linear density variation for $T < 10.2\,°\mathrm{C}$, with a maximum at $T_m = 4.0293\,°\mathrm{C}$, the Boussinesq force becomes non-linear. Convection and freezing of water are therefore interesting cases to test the flexibility of our Toolbox to deal with additional non-linear terms. We used the following density-temperature formula suggested by Gebhart and Mollendorf [49]:

$$\rho(T) = \rho_m \left(1 - w\,|T - T_m|^q\right), \tag{31}$$

with $\rho_m = 999.972$ [kg/m^3], $w = 9.2793 \cdot 10^{-6}$ [$(°C)^{-q}$], and $q = 1.894816$.

Hence, the buoyancy term $f_B = g(\rho_{ref} - \rho)/\rho_{ref}$ in (1) is not any more linear and becomes after scaling:

$$f_B(\theta) = \frac{Ra}{Pr\,Re^2}\,\frac{1}{\beta\delta T}\,\frac{\rho(\theta_f) - \rho(\theta)}{\rho(\theta_f)}, \tag{32}$$

where $\beta = (1/\rho_m)(d\rho/dT)$ is the thermal expansion coefficient taking the value $\beta = 6.91 \cdot 10^{-5}$ [$(\mathrm{K})^{-1}$] [61].

We simulate a differentially heated square cavity filled with liquid pure distilled water. This problem was investigated experimentally and numerically by Giangi et al. [16], Kowalewski and Rebow [15], Michalek and Kowalewski [62]. The non-dimensional parameters describing the problem are (see [62] for physical details): $\mathcal{R}a = 2.518084 \cdot 10^6$, $\mathcal{P}r = 6.99$ and $\mathcal{S}te = 0.125$.

5.3.1. Natural convection of water

The initial temperature is linearly distributed in the square cavity, with a hot temperature $T_h = 10\,°\mathrm{C}$ at the left wall and a cold temperature $T_c = T_f = 0\,°\mathrm{C}$ at the right wall. The temperature field and the streamlines of the steady state are presented in Fig. 11a. The isoline $\theta = \theta_m$, corresponding to the line of maximum density is represented by a dashed line. Due to the anomalous thermal variation of water density, two recirculating zones are formed in the flow: a lower (abnormal) recirculation in the vicinity of the cold wall where $\theta < \theta_m$ and an upper (normal) one where the density decreases with temperature ($\theta > \theta_m$).

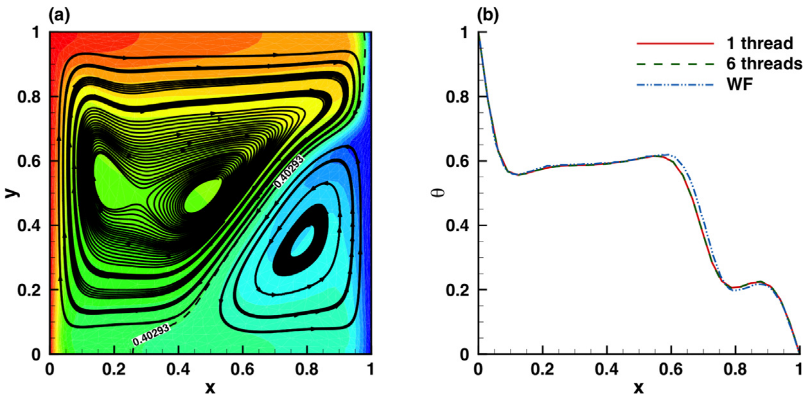


Fig. 11. Natural convection of water in a differentially heated cavity. Non-dimensional temperature θ at steady state. (a) Two-dimensional temperature field and streamlines showing the two recirculating zones. (b) Temperature profile along the horizontal central line. Comparison with the numerical results of Michalek and Kowalewski [62].

A more precise comparison with previously published results is shown in Fig. 11b. The obtained temperature profile $\theta(x)$ along the horizontal central line of the cavity ($y = 0.5$) is in good agreement with the numerical results of Michalek and Kowalewski [62]. Their results were obtained with finite-volume and finite-difference codes (FLUENT and FRECONV3V).

5.3.2. Water freezing

We finally consider the difficult case of water freezing in a square cavity. The initial state for this computation is the convection steady pattern in the cavity presented in Fig. 11. The freezing starts by dropping smoothly the temperature of the cold right wall from $T_c = 0\,°\mathrm{C}$ to $T_c = -10\,°\mathrm{C}$. The new boundary condition on the right cold wall is imposed by setting a very thin layer of $\delta x = 0.01$, with constant temperature $T = T_c$ and zero velocity.

Figure 12 shows the flow configuration at dimensionless time $t = 1.61$. The left panel depicts the domain composition and the adapted mesh following two lines of interest: the solid–liquid interface $T = 0$ and the line $T = T_m$ separating the two recirculating zones (see the middle panel of the same figure). The metrics used for adaptivity were computed from the two components of the velocity, the temperature and a P_1 tanh-function $\phi(T)$ "tracking" the value T_m. To reduce the impact of the interpolation on the global accuracy we used both $\phi(T^n)$ and $\phi(T^{n+1})$ in the adaptivity procedure (see also [17]). This adaptivity strategy allowed us to accurately capture the structure and the extent of the two recirculating zones, features that are difficult to obtain with fixed meshes (discrepancies in numerical results are described in [15,16,62]). In the right panel of Fig. 12 we superimpose the experimental image from [15] with our numerical results for the same physical time $t_\varphi = 2340$ [s]. The position of the solid–liquid interface and the flow pattern in the liquid phase correspond very well qualitatively to the experimental image. The simulation was performed with a small time step ($\delta t = 10^{-2} \approx 15$ [s]). Note from Table 1 that the computational time for this case is more than reasonable (23 min using 1 thread and only 13 min with 6 MPI processes).

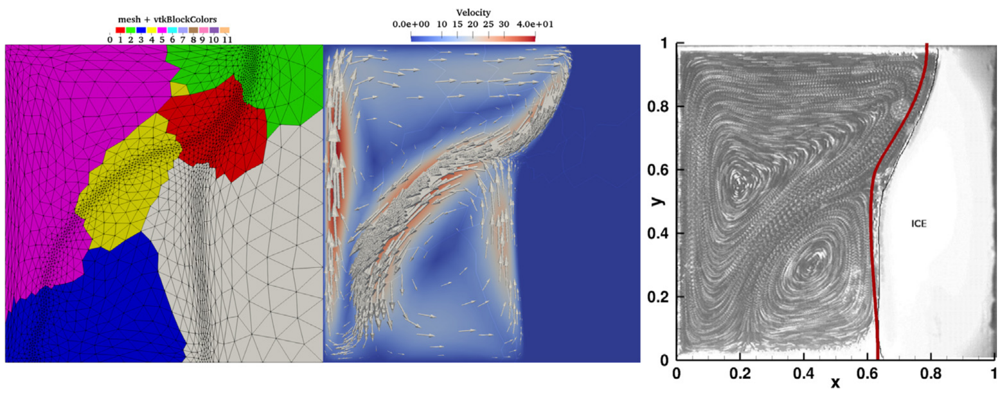


Fig. 12. Freezing of pure water. Simulated configuration at $t = 1.61$. Adapted finite-element mesh and subdomains used for the domain decomposition and parallel computing with 6 MPI processes (left panel). Velocity vectors in the liquid phase (central panel). Comparison with the experimental image from [15]; the thick red line represents the solid–liquid interface computed with the present method (right panel).

6. 3D parallel numerical results

All the 2D benchmarks presented in the previous section could be simulated in 3D using the second provided Toolbox `PCM_Toolbox_DDM_3D`. To avoid an excessively long paper, we shortly present in this section only the following selection of 3D cases: (i) the steady natural convection of air in a cubic cavity with $Ra = 10^6$ (Section 6.1), (ii) the melting of a phase-change material (Section 6.2), within a cubic (Section 6.2.1) or cylindrical domain (Section 6.2.2), and (iv) the convection and the freezing of pure water in a cube (Section 6.3). Running parameters for these cases are summarized in Tables 4 and 5, for computations using P_1 and P_2 finite elements, respectively. Again, to make these simulations accessible with fast-running jobs in a batch-queuing system, we gave priority to jobs with a reasonable number of MPI processes (less than 400). We used 4GB memory for each CPU-Core node for all simulations, excepting for the case PCM-Case #3 with 9 heated tubes, when 9GB of memory were necessary. In the subsequent sections, we present the results obtained with P_2 finite elements for the temperature, since these simulations are more challenging. The results using P_1 elements for the temperature are provided in the Supplemental Material http://lmrs-num.math.cnrs.fr/2020CPCP2.html containing images and animations of the cases presented in this section.

Table 4
Running parameters for 3D simulations: total CPU time, maximum number of degrees of freedom (dof) after mesh adaptivity, number of MPI processes, total number of time steps, maximum CPU time necessary for `mmg` library to adapt the 3D mesh and ratio between the CPU time for adapting the mesh and the CPU time for a complete time step. All computations were performed using P_1 finite elements for the temperature. All computations were performed using a parallel computer (CRIANN Computing Center and MATRICS platform) based on Intel Broadwell E5-2680 v4 @ 2.40 GHz (14 cores per socket) architecture with two sockets per node and 128 GB of DDR4 2400 MHz RAM. An Intel Omnipath 100 Gb/s low latency network was used for communications.

Case	CPU time	Max of dof	MPI procs.	Number of time steps	Max `mmg` CPU time	Ratio CPU adapt/CPU time step
Unsteady NC of air, $Ra = 10^4$	00:55:21	249 625	6	23	9 s	6%
Unsteady NC of air, $Ra = 10^6$	01:14:49	725 097	56	50	20 s	19%
PCM-Case #1	07:42:32	3,299 661	224	80	53 s	16%
PCM-Case #3: 1 tube	16:02:31	3,809 760	224	401	65 s	17%
PCM-Case #3: 4 tubes	13:29:23	8,813 510	280	151	150 s	20%
PCM-Case #3: 9 tubes	10:37:44	14,860 117	364	71	262 s	24%
Cycle of a PCM	13:06:26	2,150 052	224	382	36 s	32%
NC of water	00:59:26	3,701 916	112	20	62 s	20%
Unsteady NC of water	06:39:40	5,492 302	112	37	76 s	14%
Water freezing	1–10:11:55	4,448 031	224	296	53 s	21%

6.1. Natural convection of air in a cubic cavity

We go back to the benchmark of natural convection of air described in Section 5.1 and add the third dimension in space. We thus simulate the thermally driven cubic cavity $[0, 1]^3$, filled with air. The temperature is now fixed on the left (cold) wall surface and the right (hot) wall surface. All the other lateral surfaces are adiabatic ($\partial T/\partial n = 0$). No-slip walls are applied for the velocity on all boundary surfaces. We show in Fig. 13 the final (steady) flow configuration for $Ra = 10^6$ obtained using the time-dependent solver with P_2 finite-elements for the temperature. The mesh is mainly adapted following the boundary layers of the ascending and descending flows (Fig. 13a), which are clearly visible when representing the 3D velocity vectors (Fig. 13b). A quantitative assessment of the quality of the simulation is offered in Fig. 13c and d, comparing our results with those reported by Wakashima and Saitoh [63]. These authors solved the vorticity-stream function formulation of the Navier–Stokes equations using a fourth–order finite difference scheme and a uniform mesh of 120^3 grid nodes.

Table 5
Same caption as for Table 4, but with all computations performed using P_2 finite elements for the temperature.

Case	CPU time	Max of dof	MPI procs.	Number of time steps	Max mmg CPU time	Ratio CPU adapt/CPU time step
Unsteady NC of air, $Ra = 10^4$	01:12:38	313 769	6	23	9 s	5%
Unsteady NC of air, $Ra = 10^6$	01:22:33	913 057	56	50	20 s	18%
PCM-Case #1	08:05:24	4,871 640	224	80	66 s	13%
PCM-Case #3: 1 tube	21:49:27	3,919 350	224	401	49 s	12%
PCM-Case #3: 4 tubes	15:49:16	9,224 708	280	151	128 s	17%
PCM-Case #3: 9 tubes	13:23:43	14,981 519	364	71	210 s	17%
Cycle of a PCM	11:28:47	2,709 371	224	382	32 s	27%
NC of water	01:10:08	4,700 233	112	20	63 s	17%
Unsteady NC of water	07:14:19	6,049 784	112	35	82 s	12%
Water freezing	1–16:14:44	5,107 229	224	296	58 s	17%

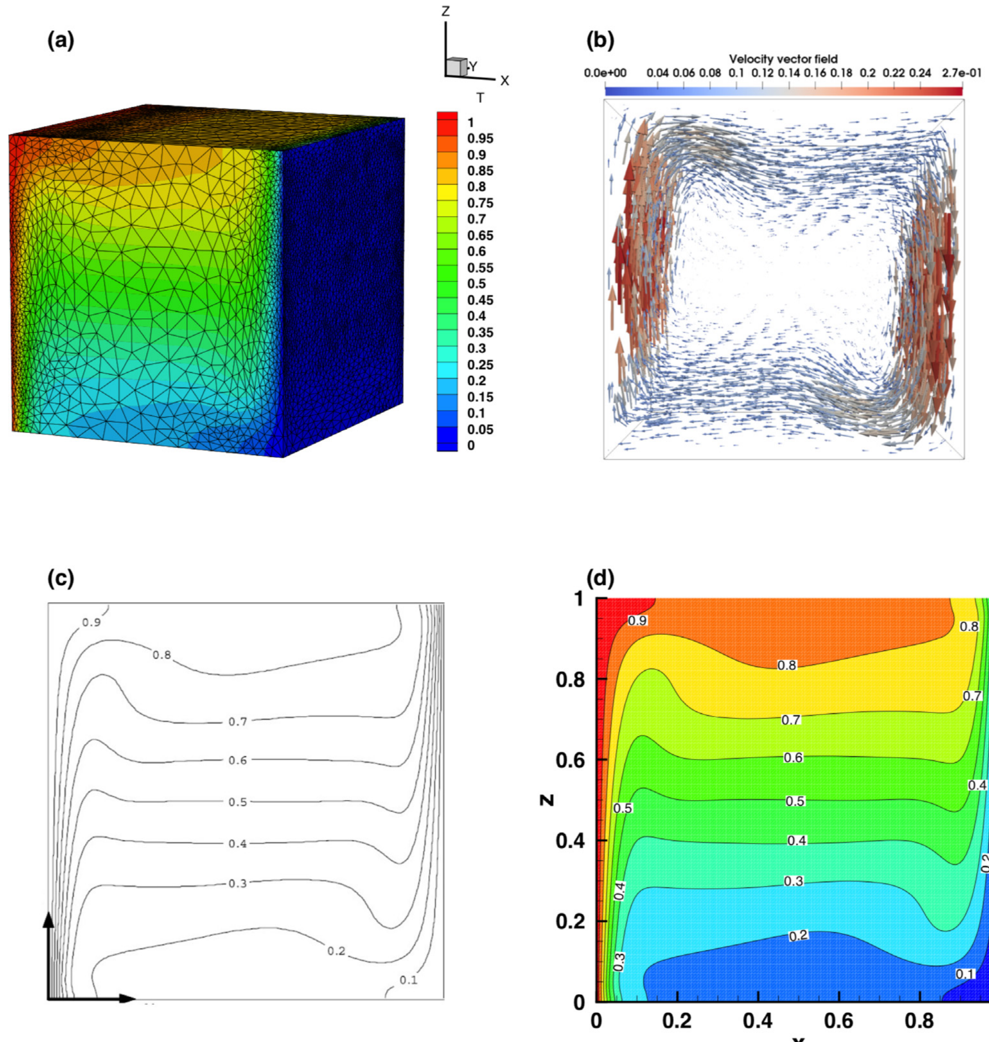


Fig. 13. Natural convection of air in a cubic differentially heated cavity ($Ra = 10^6$) computed with P_2 finite elements for the temperature. (a) Adapted mesh with a maximum number of degrees of freedom of 913 057. (b) 3D velocity vector field in the final (steady) state. Temperature contour-lines in the central section of the cube ($y = 0.5$): results of Wakashima and Saitoh [63] (c) and present results (d). Parallel computation with 56 MPI processes.

6.2. Melting of 3D phase-change materials

We simulate in this section 3D configurations of PCMs, in cubic or cylindrical geometries.

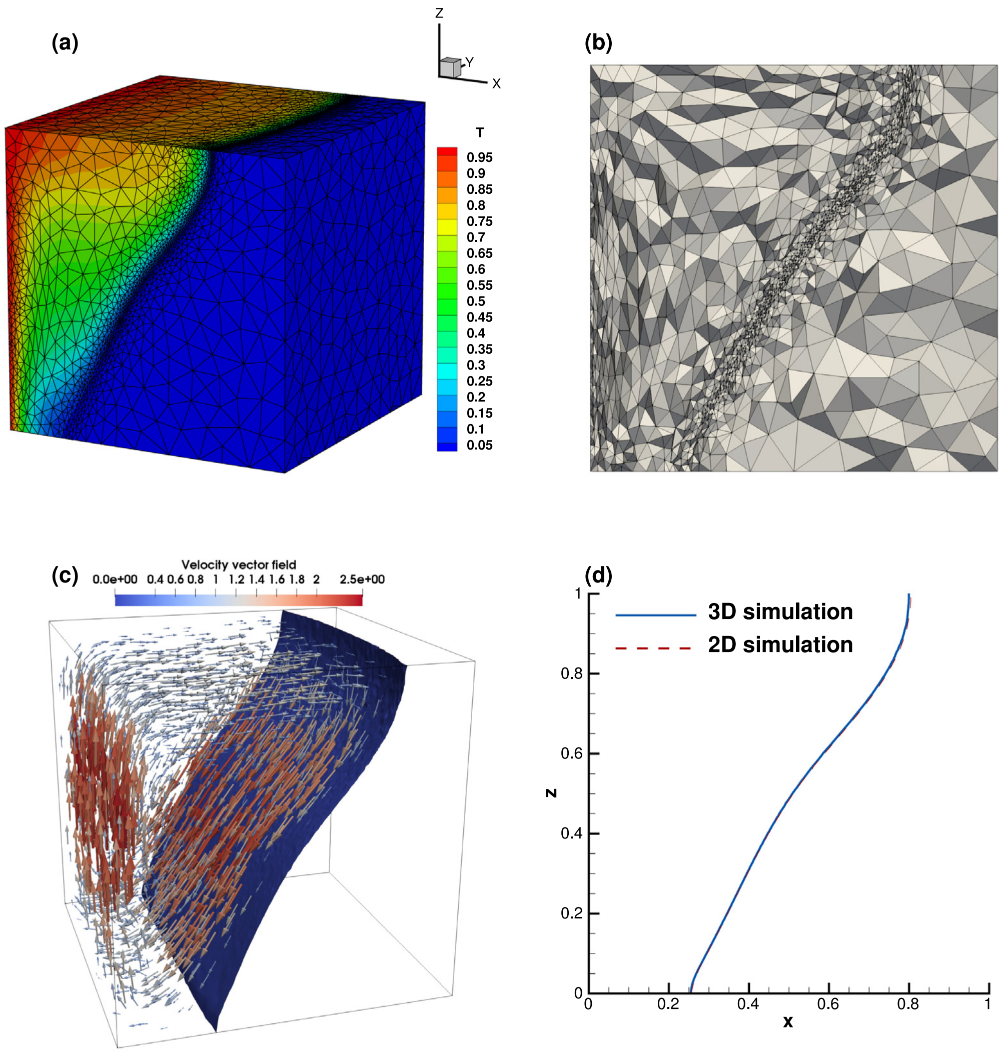


Fig. 14. Melting of a cubic PCM [53] computed with P_2 finite elements for the temperature. Configuration at $t = 80.1$, computed using 224 MPI processes with P_2 finite elements for the temperature. (a) Adapted finite-element mesh with a maximum number of degrees of freedom ndof $= 4,871\,640$. (b) Illustration of the adapted mesh in the center of the domain. (c) 3D velocity vectors and isosurface $\theta = 0$ representing the liquid–solid interface. (d) Position of the interface in the central plane ($y = 0.5$): comparison between 2D and 3D simulations (both using P_2 finite elements for the temperature).

6.2.1. Melting of a cubic octadecane PCM

We extend the 2D simulations presented in Section 5.2.1 to a 3D cubic domain $\Omega = [0, 1]^3$. We impose cold dimensionless temperature $\theta_c = -0.01$ at $x = 1$ (right wall), hot temperature $\theta_h = 1$ at $x = 0$ (left wall), and a homogeneous Neumann boundary condition at the remaining walls. A Dirichlet boundary condition $\boldsymbol{u} = 0$ is prescribed for all surfaces defining $\partial\Omega$. The temperature distribution and the corresponding adapted mesh at $t = 80.1$ is shown in Fig. 14a. The blue region denotes the solid phase. A zoom of the mesh in the mid-plane, at $t = 80.1$, refined along the iso-surface $\theta = 0$, is shown in Fig. 14b. As the heating progresses, the natural convection intensifies enough to have a pronounced influence on the shape of the interface. The temperature difference between the hot wall and the solid PCM induces a clockwise recirculation of the flow in the melted PCM (Fig. 14c). The shape of the liquid–solid interface at $t = 80.1$ is also displayed in Fig. 14c, showing a non-uniform melting front receding from the top to the bottom of the domain. The position of the solid–liquid interface at the mid-plane for 2D and 3D configurations is plotted in Fig. 14d. Differences between 2D and 3D results are not visible in this mid-plane. Three-dimensional effects are more important near the lateral walls, as explained in [64].

6.2.2. Melting of a cylindrical PCM with inner heated tubes

We simulate the melting of a cylindrical PCM with inner heated tubes, as in Section 5.2.3. The 3D domain is now a horizontal cylinder of radius $R = 1$ and length $L = 0.25$. In addition to the 2D configuration, we impose the following boundary conditions. For the temperature, a Dirichlet boundary condition ($\theta = \theta_h = 1$) is applied at the inner tubes and a Neumann boundary condition ($\frac{\partial\theta}{\partial n} = 0$) at the outer sections of the cylinder. For the velocity, a homogeneous Dirichlet boundary condition ($\boldsymbol{u} = 0$) is prescribed on the lateral surfaces of external and inner cylinders and a symmetry boundary condition is imposed on the ending sections of the cylinder. The PCM is initially solid and when the melting process starts, a liquid layer grows around the heated tubes and expands toward the lateral boundary. A slice in the adapted mesh, the velocity vector field, and temperature iso-surfaces are illustrated in Fig. 15 for configurations with one, four, and nine heated tubes. The mesh is adapted along the melting front and in the boundary layers around the heated tubes.

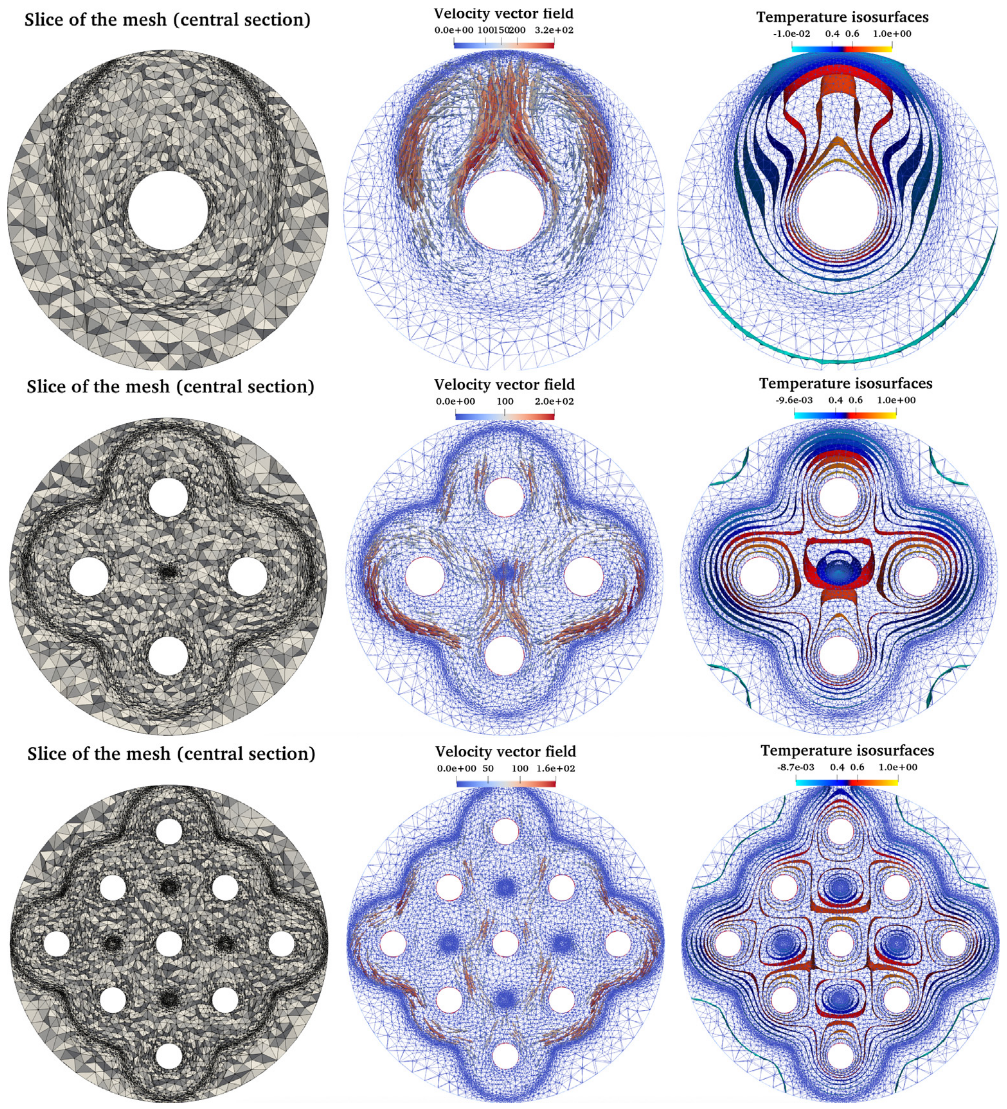


Fig. 15. Melting of cylindrical PCM computed with P_2 finite elements for the temperature. Adapted finite-element mesh (left panels), 3D velocity vector field (middle) and temperature iso-surfaces (right). Configurations with one inner heated tube (time instant $t = 1.591$, simulation with 224 MPI processes and a maximum number of degrees of freedom ndof $= 3,919\,350$), four tubes ($t = 0.621$, 280 processes and max ndof $= 9,224\,708$) and nine tubes ($t = 0.331$, 364 processes and max ndof $= 14,981\,519$).

Adaptivity is efficient even when the solid–liquid interface touches the outer wall (see left panels). Counter rotating flows are observed at the left and right parts of the cylindrical domain (middle panels). Iso-surfaces of temperature shown in the right panels also confirm that the evolution of the liquid flows keeps the symmetry of the configuration.

6.3. *Natural convection of water and water freezing*

We simulate in this final section the challenging case of the natural convection of water and the ice formation inside a differentially heated cubic cavity. The dimensionless parameters are the same that those used in Section 5.3.2. For the natural convection of water, the vertical walls at $x = 0$ and $x = 1$ are isothermal and have different temperatures $\theta_h = 1$ and $\theta_c = 0$, respectively. The remaining walls are adiabatic. A homogeneous Dirichlet boundary condition $\boldsymbol{u} = 0$ is applied on all walls. The fluid is initially at rest and the temperature is linearly distributed from the cold to the hot wall (see Fig. 16).

Given the anomalous thermal variation of water density, we also adapted the 3D mesh along $\theta = 0.4$ to capture correctly the flow structure. The temperature distribution and the corresponding adapted mesh for the steady (time-independent) state computation are shown in Fig. 16a. The blue region identifies the cold water trapped by the abnormal fluid recirculation and the red region the hot fluid driven by the upper clockwise circulation. A cut in the mesh following the mid-plane (Fig. 16b) shows the internal structure of the mesh. Smaller tetrahedra are clearly observed in the vicinity of the walls and between the two counter-rotating circulation patterns. Figure 16c shows that the three-dimensional evolution of the flow (spiral movement of the fluid along the walls) affects the topology of the iso-surface $\theta = 0.4$.

Giangi et al. [16] analyzed three-dimensional effects in both convection and freezing of water and they noted that only the flow in the symmetry plane is similar to the two-dimensional flow. They pointed out that no-slip velocity and adiabatic thermal boundary

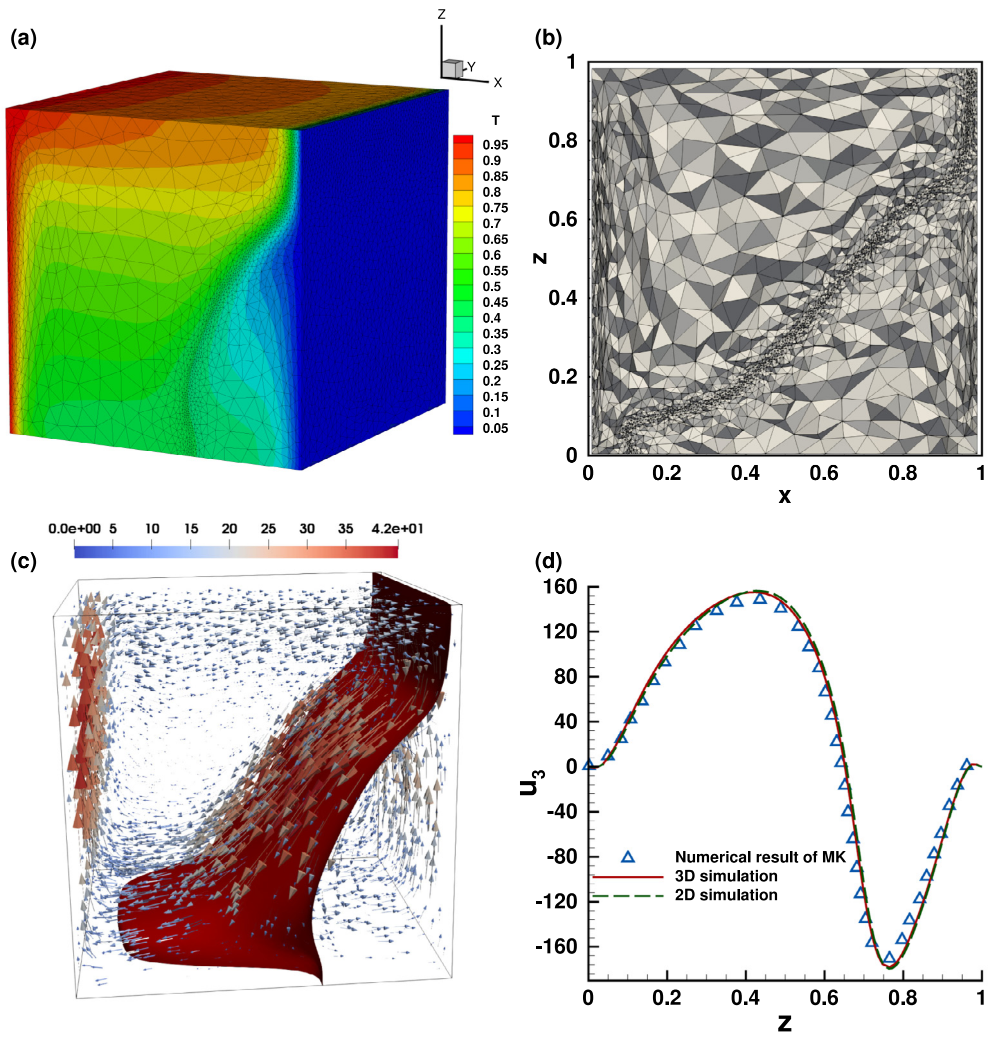


Fig. 16. Natural convection of water in a differentially heated cubic cavity computed with P_2 finite elements for the temperature. (a) Temperature distribution and 3D adapted finite element mesh at the steady state. (b) Illustration of the adapted mesh in the center of the domain. (c) Velocity vector field and temperature iso-surface $\theta = 0.4$. (d) Profile of the vertical velocity along the z−direction at the mid-plane ($y = 0.5$) and $x = 0.93$. Comparison with the numerical benchmark suggested by Kowalewski and Rebow [15] (symbols). Simulations using 112 MPI processes with P2 for the temperature and a maximum number of degrees of freedom ndof = 4, 700 233.

conditions at the side walls enhance three-dimensional effects near the walls. To validate this observation, we plot in Fig. 16d the profile of the vertical velocity along the z direction, at $x = 0.93$ and passing through the velocity saddle point, where normal and abnormal convection streams collide in the vicinity of the cold wall. We compare both 2D and 3D simulations with 3D results of Michalek and Kowalewski [62] (symbols), obtained with a finite-difference code. 3D (red solid line) profile agrees well with the benchmark solution with a maximum difference of 3%.

For the water freezing case, the simulation starts from the steady solution shown in Fig. 16. The cold temperature at the right wall is suddenly dropped from $\theta_c = 0$ to $\theta_c = -1$ (corresponding to physical temperature $T_c = -10\,^{\circ}\mathrm{C}$). Ice starts to form at the cold wall and expands toward the left wall. The temperature distribution and the adapted mesh at the final time $t_\varphi = 2340$ [s] ($t = 1.61$) are displayed in Figs. 17a and 17b.

When compared to the natural convection case (Fig. 16), the mesh is also adapted along a second moving front represented by the solid–liquid interface (iso-surface $\theta = 0$ shown in Fig. 17c). For the identification of flow structures during the ice formation, Fig. 17c shows the 3D velocity vectors, the iso-surface $\theta = 0.4$ along the anomalous density variation, and the phase-change front at time instant $t = 1.61$. The effect of the secondary flow (see [64]) is visible from the curved shape of the iso-surface $\theta = \theta_m$ in the transverse y direction. We note, however, that the shape of the solidification front is almost 2D. The buoyancy-induced fluid motion in the abnormal recirculation region is too weak to influence the solid front. Finally, the superimposition of the experimental image of Kowalewski and Rebow [15] and the current simulation give good agreement for the location of the solid–liquid interface. Differences come mainly from the fact that the undercooling phenomenon during the solidification stage is not taken into account in our physical model.

6.4. Scalability test for 3D parallel computations

We present the results for both strong and weak scalability tests. We simulate the 3D unsteady natural convection of air with $Ra = 10^6$ presented in Section 6.1. Mesh adaptation is used only for the strong scalability test.

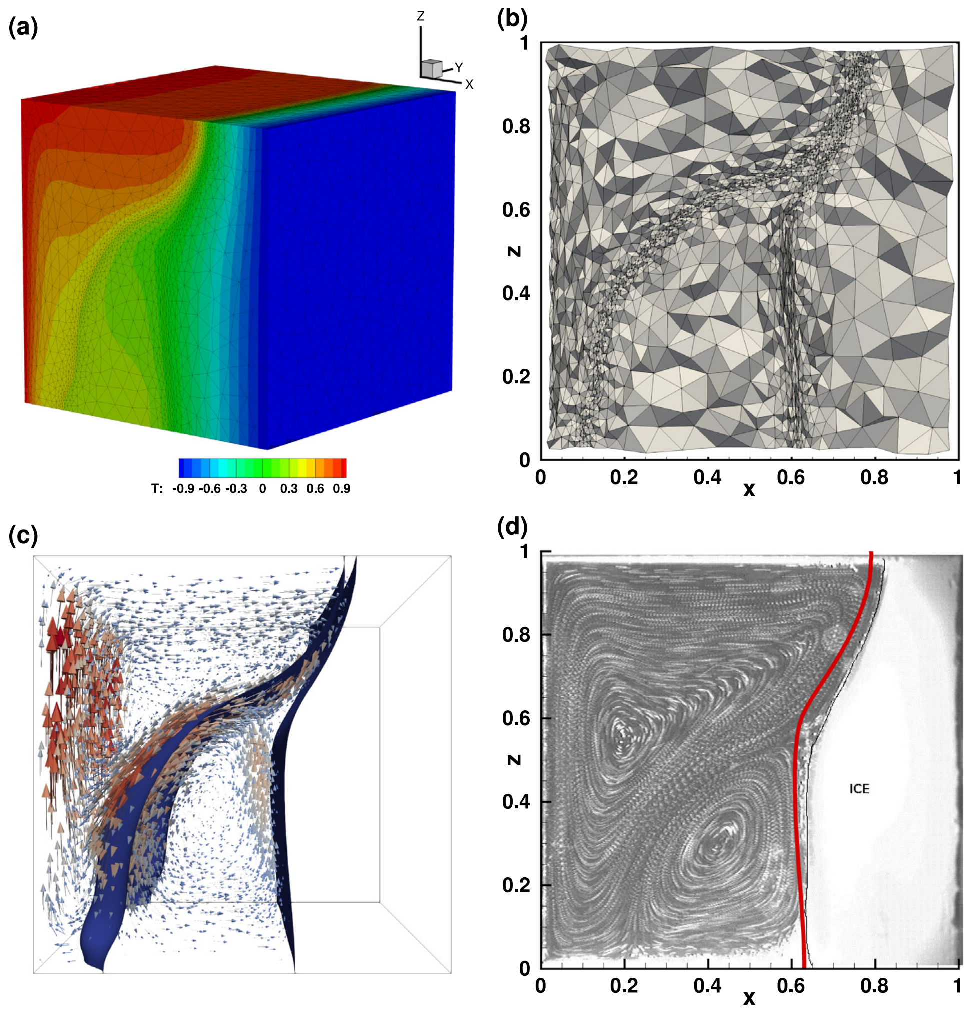


Fig. 17. Freezing of pure water in a 3D cubical cavity computed with P_2 finite elements for the temperature. (a) Temperature distribution and adapted mesh at time instant $t = 1.61$. (b) Cut through the adapted mesh at mid-plane. (c) 3D velocity vectors in the liquid phase and temperature iso-surfaces $\theta = 0.4$ and $\theta = 0$. (d) Superimposition of the interface obtained in the present simulation (red thick line) on the experimental image of Kowalewski and Rebow [15]. Simulations using 224 MPI processes with P2 for the temperature and a max of degrees of freedom 5, 107 229. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

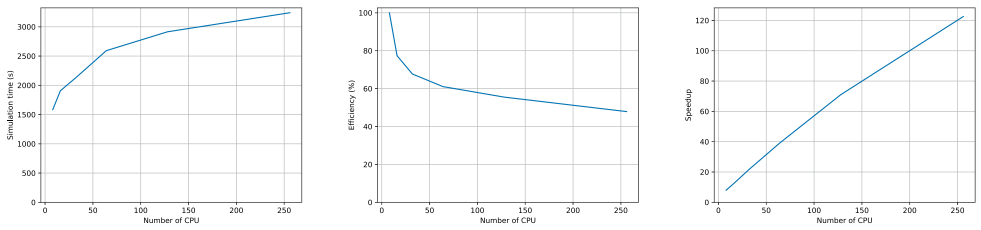


Fig. 18. Weak scalability test for 3D simulations: simulation time (left), efficiency (center) and speed-up (right) for a number of processes ranging from 8 to 256.

The CPU is Intel Broadwell E5-2680 v4 @ 2.40 GHz (14 cores per socket). Each node has two sockets and 128 GB of DDR4 2400 MHz RAM. They communicate through Intel Omnipath 100Gb/s low latency network. First, we report the simulation time, the efficiency and speed-up for the weak (Fig. 18) and strong scalability tests (Fig. 19). Assuming a perfect speed-up for 8 cores, we obtained for 256 cores a speed-up of approximately 122.56 (weak) and 70.6 (strong), resulting in an efficiency of 47.9%(weak) and 27.6%(strong).

In Figs. 20 (weak) and 21 (strong), we report the computational time for the main steps of the computational algorithm, namely the mesh adaptation, the matrix construction and the resolution of the linear system. For 8 cores, the mesh adaptation takes 221.9s (strong), the construction of the matrix 83.5s (strong) and 0.96s (weak) and the resolution of the linear system 4866.9s (strong) and 552.1s (weak). For 256 cores, the mesh adaptation takes 223.0s (strong), the construction of the matrix 76.4s (strong) and 6.91s (weak) and the resolution of the linear system 422.2s (strong) and 893.5s (weak). It results from this analysis that the only step that benefits from the parallelization is the resolution of the linear system. The speed-up of the resolution of the linear system is 78.9 (strong) and 158.2 (weak), resulting in an efficiency of 36.0% (strong) and 61,8% (weak).

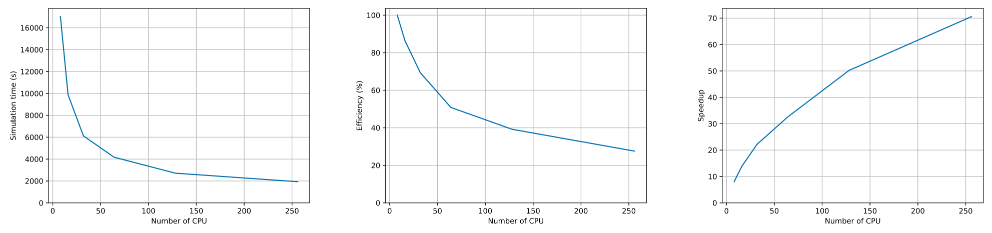


Fig. 19. Strong scalability test for 3D simulations: simulation time (left), efficiency (center) and speed-up (right) for a number of processes ranging from 8 to 256.

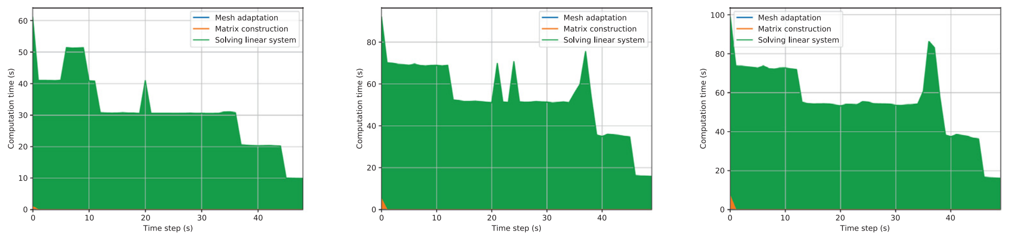


Fig. 20. Weak scalability test: stacked timings of the main steps of the computation using 8 (left), 24 (center) and 112 (right) processes.

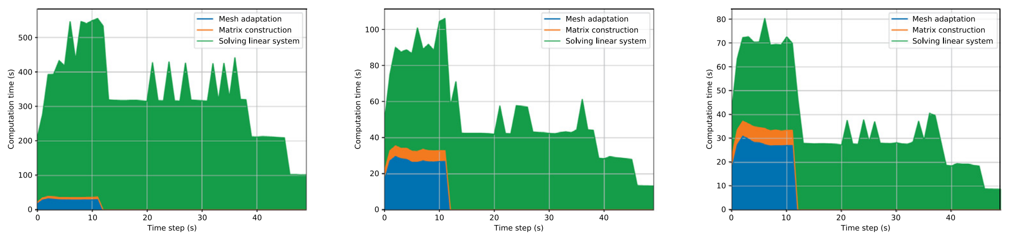


Fig. 21. Strong scalability test: stacked timings of the main steps of the computation using 8 (left), 24 (center) and 112 (right) processes.

7. Summary and conclusions

The main advantage of the new toolbox distributed with this paper is to enable parallel computing of liquid–solid phase-change systems in 2D or 3D complex geometries. The physical model implemented in the software is one of the most accurate existing for such systems: the enthalpy-porosity model, based on the incompressible Navier–Stokes equations with Boussinesq approximation for thermal effects. This model captures all the characteristics of the flow developing in the liquid phase, dominated by convection. Using a Carman–Kozeny-type penalty term, added to the momentum equations to bring progressively the velocity to zero into the solid, results in a single-domain formulation of equations for both liquid and solid phases. We proved that this single-domain enthalpy-porosity model is well adapted to the use of Schwarz domain decomposition methods (DDM), and, consequently, to parallel computing.

The numerical method was implemented using FreeFem++, a free software offering multiple advantages to easily implement a finite-element algorithm. The key ingredients of the implemented method are: (i) a second order accuracy in space (P_2 finite elements for velocity and temperature, P_1 for the pressure) and time (Gear scheme); (ii) the use of an adaptive finite element method with regularized functions representing the variation of thermodynamic properties at the solid–liquid interface, (iii) a fully implicit discretization with a Newton algorithm for solving the non-linear system of equations, and (iv) a parallel final algorithm using automatic partitioners (`Scotch` or `Metis`) and the recent library `ffddm` that makes available in FreeFem++state-of-the-art scalable Schwarz domain decomposition methods. Mesh adaptivity is an essential ingredient of the toolbox and a new algorithm was implemented using external remeshing tools (`mmg`) for 3D configurations.

We presented careful validations of the toolbox by considering parallel 2D and 3D computations of well-known benchmark cases of increasing difficulty: natural convection of air, natural convection of water, melting or solidification of a phase-change material, and, finally, a water freezing case. For 2D configurations, parallel computations with the present toolbox using modest resources (6 MPI processes) can bring a CPU gain factor between 2 and 3, when compared to previous sequential algorithms [3]. But the most significant progress brought by the new toolbox is to make affordable well-resolved 3D computations with mesh adaptivity, which are hardly accessible with sequential algorithms. The efficiency of the toolbox in 3D was demonstrated by simulating difficult cases (melting of a PCM, water freezing) in simple (cubic) or complex geometries (cylindrical with inner tubes). To facilitate its use, the toolbox is provided with separate folders containing all the necessary files (parameters, restart files) necessary to run all the cases described in the paper. Ready-made scripts and layouts allow the user to generate the figures presented in this paper with newly generated data after running the programs. Validation data sets from experiments or previous publications are included in these layouts.

The present parallel approach can be easily tested and adapted to address other computational challenges related to different physical or mathematical models in this field. For instance, other possible choices for the finite-element discretization could be tested for the implementation of the algorithm presented in this paper. To prove the versatility of the toolbox, we illustrate in Fig. 22 the results obtained using the mini-element introduced by Arnold, Brezzi and Fortin [see35,65]. This is the simplest element for Stokes-type problems, offering inf–sup stability and global linear convergence. Compared to the Taylor–Hood element, in the mini-element

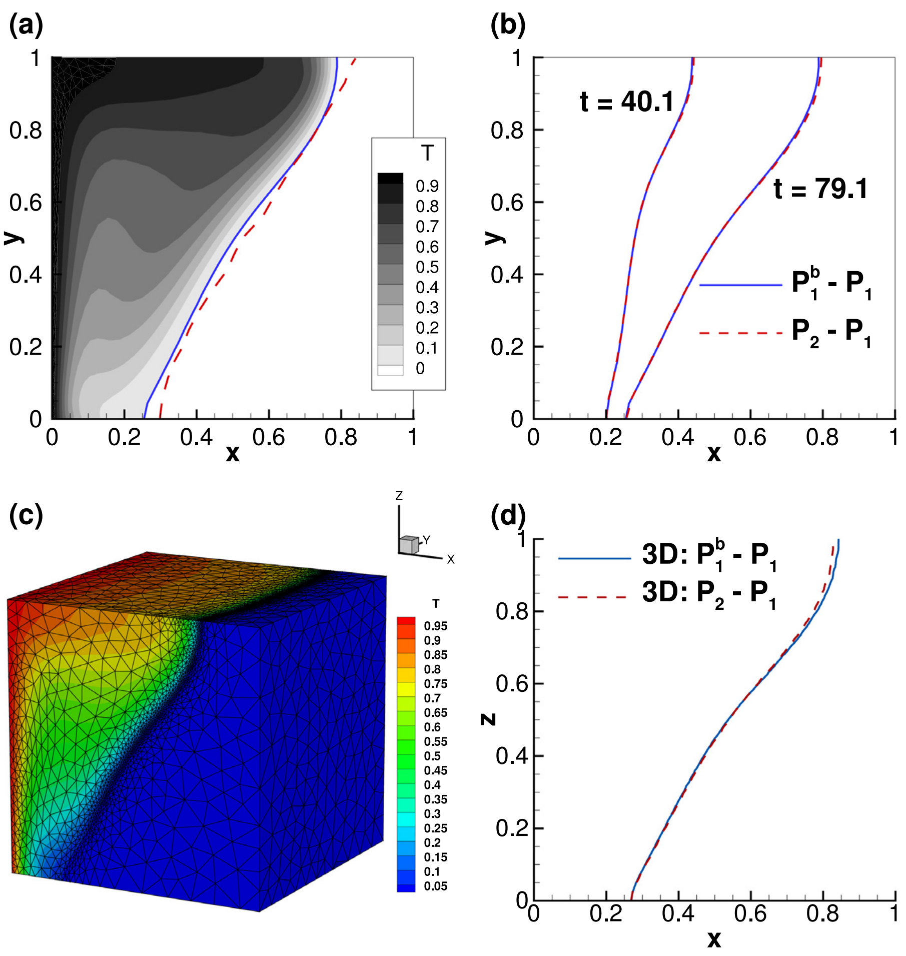


Fig. 22. Melting of a PCM. Results obtained using the mini-element (P_1^b for the velocity and P_1 for the pressure). 2D simulation: (a) Temperature field and position of the interface at $t = 80.1$; the red dashed line is the experimental result of Okada [53]. (b) Comparison with the Taylor–Hood discretization (red dashed lines) for the position of the interface at $t = 40.1$ and $t = 79.1$. 3D simulation: (c) Temperature field and adapted mesh at $t = 80.1$. (d) Comparison with the Taylor–Hood discretization (red dashed line) for the position of the interface at $t = 79.1$ in the central plane ($y = 0.5$). To be compared with Figs. 5 and 14.

discretization velocities are piecewise P_1^b (P_1-bubble), while pressure is still piecewise P_1. Since the mini-element already exists in FreeFem++ the only necessary change in the distributed scripts is to declare the unknowns of the problem accordingly to the syntax discussed in Section 3.1: `fespace Wh(Th,[P1b,P1b,P1b,P1,P1])`.

Figure 22 shows that, for the test case of the melting of a PCM presented in Section 5.2.1 (2D case) and 6.2.1 (3D case), the results are remarkably accurate when compared to those obtained using the Taylor–Hood discretization. The CPU time is reduced by 20% for the 2D computation and 42% for the 3D simulation with the same number of MPI processes (6 for 2D and 224 for 3D). However, knowing that the mini-element offers a minimal accuracy for the velocity field, a careful assessment of this type of discretization in computing more complicated cases (with several convection rolls and complex dynamics in the fluid part) is necessary. Also, the efficiency of the mesh adaptivity in this case has to be carefully investigated to obtain a robust algorithm (as in the present toolbox using the Taylor–Hood finite element). This topic will be addressed in a future contribution, together with the possibility to enrich the physical model for the solidification process.

Supplementary images and movies depicting the dynamics of some cases simulated in this paper are provided as Supplemental Material at http://lmrs-num.math.cnrs.fr/2020CPCP2.html.

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 project was co-financed by the European Union with the European regional development funds and by the Normandy Regional Council, France via the M2NUM (ERDF, HN0002081) and M2SiNUM (ERDF, 18P03390/18E01750/18P02733) projects. Part of this work was performed using computing resources of CRIANN (Centre Régional Informatique et d'Applications Numériques de Normandie, France) and MATRICS plateform (Amiens, France).

References

[1] F. Agyenim, N. Hewitt, P. Eames, M. Smyth, Renew. Sustain. Energy Rev. 14 (2) (2010) 615–628.
[2] S. Kalnæs, B. Jelle, Energy Build. 94 (2015) 150–176.
[3] A. Rakotondrandisa, G. Sadaka, I. Danaila, Comput. Phys. Comm. 253 (2020) 107188.
[4] A.D. Brent, V.R. Voller, K.J. Reid, Numer. Heat Transfer 13 (1988) 297–318.
[5] F. Stella, M. Giangi, in: A. Kowalewski, D. Gobin (Eds.), Phase Change with Convection: Modelling and Validation, Vol. 21, Springer, 2004, pp. 9–272.
[6] R.T. Tenchev, J.A. Mackenzie, T.J. Scanlon, .M.T. Stickland, Int. J. Heat Fluid Flow 26 (4) (2005) 597–612.
[7] K. Luo, F.-J. Yao, H.-L. Yi, H.-P. Tan, Appl. Therm. Eng. 86 (2015) 238–250.
[8] W. Gong, K. Johannes, F. Kuznik, Commun. Comput. Phys. 17 (5) (2015) 1201–1224.
[9] A. Kowalewski, D. Gobin, Phase Change with Convection: Modelling and Validation, Springer, 2004.
[10] N. Hannoun, V. Alexiades, T.Z. Mai, Numer. Heat Transfer B 44 (3) (2003) 253–276.
[11] N. Hannoun, V. Alexiades, T.Z. Mai, Internat. J. Numer. Methods Fluids 48 (11) (2005) 1283–1308.
[12] Y. Belhamadia, A.S. Kane, A. Fortin, Int. J. Numer. Anal. Model. 3 (2) (2012) 192–206.
[13] P. Angot, C.-H. Bruneau, P. Fabrie, Numer. Math. 81 (4) (1999) 497–520.
[14] B. Favier, J. Purseed, L. Duchemin, J. Fluid Mech. 858 (2019) 437–473.
[15] T.A. Kowalewski, M. Rebow, Int. J. Comput. Fluid Dyn. 11 (1999) 193–210.
[16] M. Giangi, T.A. Kowalewski, F. Stella, E. Leonardi, Comput. Assist. Mech. Eng. Sci. 7 (2000) 321–342.
[17] Y. Belhamadia, A. Fortin, E. Chamberland, J. Comput. Phys. 194 (1) (2004) 233–255.
[18] Y. Belhamadia, A. Fortin, E. Chamberland, J. Comput. Phys. 201 (2004) 753–770.
[19] I. Danaila, R. Moglan, F. Hecht, S. Le Masson, J. Comput. Phys. 274 (2014) 826–840.
[20] Y. Belhamadia, A. Fortin, T. Briffard, Numer. Heat Transfer A 76 (4) (2019) 179–197.
[21] A. Zimmerman, J. Kowalski, in: M. Schäfer, M. Behr, B. Wohlmuth (Eds.), Recent Advances in Computational Engineering, ICCE 2017, in: Lecture Notes in Computational Science and Engineering, vol. 124, Springer, 2018, pp. 177–197.
[22] P.-H. Tournier, F. Nataf, P. Jolivet, Ffddm online documentation, 2019, http://doc.freefem.org/documentation/ffddm.
[23] F. Pellegrini, J. Roman, High-Performance Computing and Networking, Vol. 49, Springer, 1996, pp. 3–498.
[24] G. Karypis, V. Kumar, SIAM J. Sci. Comput. 20 (1) (1998) 359–392.
[25] V. Dolean, P. Jolivet, F. Nataf, An Introduction to Domain Decomposition Methods: Algorithms, Theory and Parallel Implementation, SIAM, 2015.
[26] C. Dapogny, C. Dobrzynski, P. Frey, J. Comput. Phys. 262 (2014) 358–378.
[27] V.R. Voller, M. Cross, N.C. Markatos, Int. J. Numer. Methods Eng. 24 (1987) 271–284.
[28] Y. Cao, A. Faghri, W.S. Chang, Int. J. Heat Mass Transfer 32 (7) (1989) 1289–1298.
[29] A.C. Kheirabadi, D. Groulx, Proceedings of CHT-15, ICHMT International Symposium on Advances in Computational Heat Transfer, Ichmt Digital Library Online. Begel House Inc, 2015.
[30] R. Temam, Navier–Stokes Equations and Nonlinear Functional Analysis, SIAM, Philadelphia, 1983.
[31] V. Girault, P.-A. Raviart, Finite Element Methods for Navier-Stokes Equations, Springer Verlag, Berlin, 1986.
[32] A. Quarteroni, A. Valli, Numerical Approximation of Partial Differential Equations, Springer-Verlag, Berlin and Heidelberg, 1994.
[33] R. Aldbaissy, F. Hecht, G. Mansour, T. Sayah, Calcolo 55 (4) (2018) 44.
[34] J. Woodfield, M. Alvarez, B. Gamez-Vargas, R. Ruiz-Baier, J. Comput. Appl. Math. 360 (2019) 117–137.
[35] F. Brezzi, M. Fortin, Mixed and Hybrid Finite Element Methods, Springer Verlag, 1991.
[36] D. Boffi, F. Brezzi, M. Fortin, Mixed Finite Element Methods and Applications, Springer Verlag, 2013.
[37] C. Taylor, P. Hood, Comput. & Fluids 1 (1973) 73–100.
[38] F. Hecht, O. Pironneau, A.L. Hyaric, K. Ohtsuke, FreeFem++ (manual), 2007, www.freefem.org.
[39] F. Hecht, J. Numer. Math. 20 (2012) 251–266.
[40] I. Danaila, F. Hecht, J. Comput. Phys. 229 (2010) 6946–6960.
[41] G. Vergez, I. Danaila, S. Auliac, F. Hecht, Comput. Phys. Commun. 209 (2016) 144–162.
[42] Y. Zhang, I. Danaila, Appl. Math. Model. 37 (2013) 4809–4824.
[43] H. Borouchaki, M.J. Castro-Diaz, P.L. George, F. Hecht, B. Mohammadi, 5th Inter. Conf. on Numerical Grid Generation in Computational Field Simulations, Mississipi State Univ, 1996.
[44] M. Castro-Diaz, F. Hecht, B. Mohammadi, Internat. J. Numer. Methods Fluids 25 (2000) 475–491.
[45] F. Hecht, B. Mohammadi, AIAA Paper, Vol. 97, 1997, pp. 0859.
[46] P.L. George, H. Borouchaki, Delaunay Triangulation and Meshing, Hermès, Paris, 1998.
[47] P.J. Frey, P.L. George, Maillages, Hermès, Paris, 1999.
[48] B. Mohammadi, O. Pironneau, Applied Shape Design for Fluids, Oxford Univ. Press, 2000.
[49] B. Gebhart, J. Mollendorf, Deep Sea Res. 24 (1977) 831–848.
[50] P Le Quéré, Comput. Fluids 20 (1991) 24–41.
[51] S. Wang, A. Faghri, T.L. Bergman, Int. J. Heat Mass Transfer 53 (2010) 1986–2000.
[52] Z. Ma, Y. Zhang, Int. J. Numer. Methods Heat Fluid Flow 16 (11) (2006) 204–225.
[53] M. Okada, Int. J. Heat Mass Transfer 27 (1984) 2057–2066.
[54] O. Bertrand, B. Binet, H. Combeau, S. Couturier, Y. Delannoy, D. Gobin, M. Lacroix, M. Le Quéré, J. Mencinger, et al., Int. J. Therm. Sci. 38 (1) (1999) 5–26.
[55] R. Nourgaliev, H. Luo, B. Weston, A. Anderson, S. Schofield, T. Dunn, J.-P. Delplanque, J. Comput. Phys. 305 (2016) 964–996.
[56] D. Gobin, P. Le Quéré, Comput. Assist. Mech. Eng. Sci. 7 (3) (2000) 289–306.
[57] J.A. Dantzig, Internat. J. Numer. Methods Engrg. 28 (8) (1989) 1769–1785.
[58] P. Le Quéré, D. Gobin, Int. J. Therm. Sci. 38 (7) (1999) 595–600.
[59] M.M. Cerimele, D. Mansutti, F. Pistella, Comput. & Fluids 31 (4) (2002) 437–451.
[60] F. Giangi, M. Stella, Numer. Heat Transfer A 38 (2) (2000) 193–208.
[61] T.J. Scanlon, M.T. Stickland, Int. J. Heat Mass Transfer 47 (2004) 429–436.
[62] T. Michalek, T.A. Kowalewski, Task Q. 7 (3) (2003) 389–408.
[63] S. Wakashima, T. Saitoh, Int. J. Heat Mass Transfer 47 (4) (2004) 853–864.
[64] K.W.P.A. Nikrityuk, IOP Conference Series: Materials Science and Engineering, Vol. 27, IOP Publishing, 2012, 012054.
[65] D.N. Arnold, F. Brezzi, M. Fortin, Calcolo 21 (1984) 337–344.