A Schur Complement Method for Compressible Two-Phase Flow Models

Thu-Huyen Dao, Michael Ndjinga, and Frédéric Magoulès

1 Introduction

Computations of complex two-phase flows are required for the safety analysis of nuclear reactors. These computations keep causing problems for the development of best estimate computer codes dedicated to design and safety studies of nuclear reactors. Moreover, we often need to find the long-term behavior of the system. In these cases, implicit schemes are proven very efficient. Unfortunately, for implicit schemes, after the discretization, we need to solve a nonlinear system $\mathscr{A}U = b$. This task is computationally expensive in particular since the matrix $\mathscr{A}$ is usually non-symmetric and very ill-conditioned. It is therefore necessary to find an efficient preconditioner.

When the size of the system is large, the parallel resolution on multiple processors is essential to obtain reasonable computation times. Currently in the thermal hydraulic code, FLICA-OVAP (see [7]), the matrix $\mathscr{A}$ and the right hand side b are stored on multiple processors and the system is solved in parallel with a Krylov solver with a classical incomplete factorization preconditioner. Unfortunately, the parallel preconditioners of FLICA-OVAP only perform well on a few processors. In contrast, if we want to increase the number of processors these parallel preconditioners perform poorly. Tests were run on different test cases and

T.-H. Dao
CEA-Saclay, DEN, DM2S, STMF, LMEC, 91191 Gif-sur-Yvette, France

Appl. Mat. and Syst. Lab., Ecole Centrale Paris, 92295 Châtenay-Malabry, France

M. Ndjinga
CEA-Saclay, DEN, DM2S, STMF, LMEC, 91191 Gif-sur-Yvette, France

F. Magoulès (✉)
Appl. Mat. and Syst. Lab., Ecole Centrale Paris, 92295 Châtenay-Malabry, France
e-mail: frederic.magoules@hotmail.com

DOI 10.1007/978-3-319-05789-7_73,

led us to conclude that it is often better not to use these parallel preconditioners, especially for $3D$ problems [2]. This strategy does not make an optimal use of the available computational power. Hence, we seek for more efficient methods to distribute the computations. We study and use a domain decomposition method as an alternative to the classical distribution.

2 Mathematical Model

For the modeling of two-phase flows, several sets of equations have been worked out. They range in complexity from the homogeneous equilibrium model to two-fluid models involving unequal pressure for each phase. In this paper, we consider the well-known two-fluid model. This model is obtained by averaging the balance equations for each separated phase, using space, time or ensemble averaged quantities (see [8] and [6]). The unknown physical quantities are the volume fraction $\alpha_k \in [0, 1]$, the density $\rho_k \geq 0$, and the velocity $\mathbf{u}_k$ of each phase. The subscript k stands for l if it is the liquid phase and g for the gas phase. The common averaged pressure of the two phases is denoted by p. In our model, pressure equilibrium between the two phases is postulated. For the sake of simplicity, we study the isentropic two-fluid model. This model can be written as follows:

$$\begin{cases} \frac{\partial(\alpha_g\rho_g)}{\partial t} + \nabla\cdot(\alpha_g\rho_g\boldsymbol{u_g}) = 0, \\ \frac{\partial(\alpha_l\rho_l)}{\partial t} + \nabla\cdot(\alpha_l\rho_l\boldsymbol{u_l}) = 0, \\ \frac{\partial(\alpha_g\rho_g\boldsymbol{u_g})}{\partial t} + \nabla\cdot(\alpha_g\rho_g\boldsymbol{u_g}\otimes\boldsymbol{u_g}) + \alpha_g\nabla p + \Delta p\nabla\alpha_g - \nabla\cdot(\alpha_g\nu_g\nabla\boldsymbol{u_g}) = 0, \\ \frac{\partial(\alpha_l\rho_l\boldsymbol{u_l})}{\partial t} + \nabla\cdot(\alpha_l\rho_l\boldsymbol{u_l}\otimes\boldsymbol{u_l}) + \alpha_l\nabla p + \Delta p\nabla\alpha_l - \nabla\cdot(\alpha_l\nu_l\nabla\boldsymbol{u_l}) = 0, \end{cases} \tag{1}$$

with $\alpha_g + \alpha_l = 1$, and the two equations of state(EOS) $\rho_g = \rho_g(p)$ and $\rho_l = \rho_l(p)$. In our problem, we use the stiffened equation of state. Here ν_k is the viscosity of phase k, and Δp denotes the pressure default $p - p_k$ between the bulk average pressure and the interfacial average pressure.

By denoting $m_k = \alpha_k\rho_k$, $\mathbf{q}_k = \alpha_k\rho_k\mathbf{u}_k$ and $\mathbf{U} = \left(m_g, \mathbf{q}_g, m_l, \mathbf{q}_l\right)^t$, we can write the system (1) as follows:

$$\frac{\partial\mathbf{U}}{\partial t} + F^{conv}(\mathbf{U}) + F^{diff}(\mathbf{U}) = 0, \quad \text{where} \tag{2}$$

$$F^{conv}(\mathbf{U}) = \begin{pmatrix} \nabla\cdot\mathbf{q}_g \\ \nabla\cdot\mathbf{q}_l \\ \nabla\cdot(\mathbf{q}_g\otimes\frac{\mathbf{q}_g}{m_g}) + \alpha_g\nabla p + \Delta p\nabla\alpha_g \\ \nabla\cdot(\mathbf{q}_l\otimes\frac{\mathbf{q}_l}{m_l}) + \alpha_l\nabla p + \Delta p\nabla\alpha_l \end{pmatrix}, F^{diff}(\mathbf{U}) = \begin{pmatrix} 0 \\ 0 \\ -\nabla\cdot(\alpha_g\nu_g\nabla\frac{\mathbf{q}_g}{m_g}) \\ -\nabla\cdot(\alpha_l\nu_l\nabla\frac{\mathbf{q}_l}{m_l}) \end{pmatrix}.$$

3 Numerical Method

Most of the numerical methods used in two-phase flow computer codes are based upon semi-implicit finite difference schemes with staggered grids and donor-cell differencing. The main features of these schemes are their efficiency and their robustness. However, these methods have a large amount of numerical dissipation, giving poor accuracy in smooth regions of the flow. Moreover, discontinuities are heavily smeared on coarse grids and oscillations appear when the grid is refined. Here, we propose to use an approximate Riemann solver to discretize and solve the system (2). We decompose the computational domain into N disjoint cells C_i with volume v_i. Two neighboring cells C_i and C_j have a common boundary ∂C_{ij} with area s_{ij}. We denote $N(i)$ the set of neighbors of a given cell C_i and $\boldsymbol{n}_{ij}$ the exterior unit normal vector of ∂C_{ij}. Integrating the system (2) over C_i and setting $\mathbf{U}_i(t) = \frac{1}{v_i}\int_{C_i} U(x,t)dx$ and $U_i^n = \mathbf{U}_i(n\Delta t)$, the discretized equations can be written:

$$\int_{C_i} \frac{\partial \mathbf{U}}{\partial t}\,\mathrm{d}\mathbf{x} + \sum_{j\in N(i)} \Phi_{ij}^{conv} + \sum_{j\in N(i)} \Phi_{ij}^{diff} = 0 \tag{3}$$

with Φ_{ij}^{conv}, Φ_{ij}^{diff} denote the numerical flux of convection and diffusion on the cell C_i in direction of the neighbor cell C_j.

The diffusion numerical flux Φ_{ij}^{diff} is approximated on structured meshes using the formula:

$$\Phi_{ij}^{diff} = D(\frac{\mathbf{U}_i + \mathbf{U}_j}{2})(\mathbf{U}_j - U_i). \tag{4}$$

Full details of the evaluation of diffusive flux terms are given in [15].

Due to the $\alpha_k \nabla p$ and $\Delta p \nabla \alpha_k$ terms, the inviscid part of the two-phase flow cannot be written in a conservative form. But this system can be written in the quasi-linear form:

$$\frac{\partial \mathbf{U}}{\partial t} + A(\mathbf{U})\frac{\partial \mathbf{U}}{\partial x} = 0. \tag{5}$$

Under some simplifying assumptions, the authors of [16] were able to obtain a conservative form that allowed them to give a sense to discontinous solutions. It was also under those assumptions that they have been able to develop an approximate Riemann solver of Roe-type for the system (5) providing a local linearization of the non-conservative term $\alpha_k \nabla p$. We can also construct other linearizations than that of [16]. Here, we will not propose a specific linearization but a general method for the construction of the Roe matrix once we have chosen a linearization. We then define a local inviscid flux function $F^{loc}$and a local Roe matrix A_{Roe} for this linearization. The inviscid flux in the normal direction to the cell interface $\partial C_{i,j}$ is given by:

$$\begin{aligned}\Phi_{ij}^{conv} =& \frac{F^{loc}(\mathbf{U}_i) + F^{loc}(\mathbf{U}_j)}{2}.\mathbf{n}_{ij} + \mathscr{D}\frac{\mathbf{U}_i - \mathbf{U}_j}{2} \\ =& F^{loc}(\mathbf{U}_i).\mathbf{n}_{ij} + A^-(\mathbf{U}_j - \mathbf{U}_i),\end{aligned} \tag{6}$$

where $\mathscr{D}$ is an upwinding matrix, A_{Roe} the Roe matrix and $A^{\pm} = \frac{1}{2}(A_{Roe} \pm \mathscr{D})$.

The choice $\mathscr{D} = 0$ gives the centered scheme, whereas $\mathscr{D} = |A_{Roe}|$ gives the upwind scheme.

3.1 Newton Scheme

Finally, since $\sum_{j\in N(i)} F^{loc}(\mathbf{U}_i).\boldsymbol{n}_{ij} = 0$, using (6) and (4) Eq. (3) of the numerical scheme becomes:

$$\frac{\mathbf{U}_i^{n+1} - \mathbf{U}_i^n}{\Delta t} + \sum_{j\in N(i)} \frac{s_{ij}}{v_i}\{(A^- + D)(\mathbf{U}_i^{n+1}, \mathbf{U}_j^{n+1})\}(\mathbf{U}_j^{n+1} - \mathbf{U}_i^{n+1}) = 0. \tag{7}$$

The system (7) is nonlinear, hence we use the following Newton iterative method to obtain the required solutions:

$$\begin{aligned}\frac{\delta\mathbf{U}_i^{k+1}}{\Delta t} + \sum_{j\in N(i)} \frac{s_{ij}}{v_i}\left[(A^- + D)(\mathbf{U}_i^k, \mathbf{U}_j^k)\right]\left(\delta\mathbf{U}_j^{k+1} - \delta\mathbf{U}_i^{k+1}\right) \\ = -\frac{\mathbf{U}_i^k - \mathbf{U}_i^n}{\Delta t} - \sum_{j\in N(i)} \frac{s_{ij}}{v_i}\left[(A^- + D)(\mathbf{U}_i^k, \mathbf{U}_j^k)\right](\mathbf{U}_j^k - \mathbf{U}_i^k),\end{aligned} \tag{8}$$

where $\delta\mathbf{U}_i^{k+1} = \mathbf{U}_i^{k+1} - \mathbf{U}_i^k$ is the variation of the k-th iterate that approximates the solution at time $n+1$. Defining the unknown vector $\mathscr{U} = (\mathbf{U}_1, \ldots, \mathbf{U}_N)^t$, each Newton iteration for the computation of $\mathscr{U}$ at time step $n+1$ requires the numerical solution of the following linear system:

$$\mathscr{A}(\mathscr{U}^k)\delta\mathscr{U}^{k+1} = b(\mathscr{U}^n, \mathscr{U}^k). \tag{9}$$

3.2 Scaling Strategy

The larger the time step, the worse the condition number of the matrix $\mathscr{A}$ in (9). As a consequence, it is important to apply a preconditioner before solving the linear system. The most popular choice is the Incomplete LU factorisation (later named ILU, see [1] for more details). The error made by the approximate factorisation using an ILU preconditioner depends on the size of the off diagonal coefficients of the matrix. For a better performance of the preconditioner, it is desirable that off diagonal entries of the matrix have small magnitudes.

Here, we use the Scaling strategy (see details in [3]) to improve the condition number of the matrix. This strategy is a similarity transformation. Combined with the classical ILU preconditioner this strategy has reduced significantly the GMRES iterations for local systems and the computational time.

4 Domain Decomposition Method

The object of the present work is to solve the compressible fluids by a nonoverlapping domain decomposition methods [9, 11, 12, 14], and more precisely by a Schur complement method. A simple attempt is to adapt the principle of the domain decomposition method for elliptic problems [10, 13] to our problems. As in the case of elliptic problems, the principle is that we decompose the global problem into independent subproblems which are solved by each processor. However, the implementation of these ideas in hyperbolic problems raise some technical difficulties such as:

- The scheme must be conservative.
- In the finite volume formulation, there is no unknown defined at the interface.
- The boundary condition of hyperbolic systems must depend on the characteristics of the problem.

Those difficulties are solved in [5] for the Euler equations by replacing the interface variables in the context of elliptic problems by the interface fluxes in the context of hyperbolic problems. In this paper, we introduce a new interface variable which make the Schur complement method easy to build and allows us to treat diffusion terms.

4.1 *Implicit Coupling*

We recall the linear system at each Newton iteration of the implicit scheme (8):

$$\begin{aligned}\frac{\delta \mathbf{U}_i^{k+1}}{\Delta t} + & \sum_{j \in N(i)} \frac{s_{ij}}{v_i} \left[(A^- + D)(\mathbf{U}_i^k, \mathbf{U}_j^k) \right] \left(\delta \mathbf{U}_j^{k+1} - \delta \mathbf{U}_i^{k+1} \right) \\ = & \quad -\frac{\mathbf{U}_i^k - \mathbf{U}_i^n}{\Delta t} - \sum_{j \in N(i)} \frac{s_{ij}}{v_i} \left[(A^- + D)(\mathbf{U}_i^k, \mathbf{U}_j^k) \right] (\mathbf{U}_j^k - \mathbf{U}_i^k).\end{aligned}$$

We would like to solve (8) on N processors and each processor work on one subdomain. We see that it lacks $\delta \mathbf{U}_j^{k+1}$ to the computational unit of the subdomain I if the cell j belongs to another subdomain, and it is not calculable by the system since $\delta \mathbf{U}_j^{k+1}$ is to be calculated. Then the processor I needs from the processor J the value $\delta \mathbf{U}_j^{k+1}$ which is not yet available. Conversely, the processor J needs $\delta \mathbf{U}_i^{k+1}$ from the processor I.

4.2 A New Interface Variable

In order to include diffusion terms in the model and to use various schemes and various systems, we introduce a new interface flux variable $\delta\phi_{ij}$ (see [4]) at the domain interface between two neighboring cells C_i and C_j which belong to different subdomains:

$$\delta\phi_{ij} = \delta \mathbf{U}_j - \delta \mathbf{U}_i \tag{10}$$

In the case where the cell i of the subdomain I is at the boundary and has to communicate with the neighboring subdomains, we can rewrite the system (8) as:

$$\begin{aligned}
&\frac{\delta \mathbf{U}_i^{k+1}}{\Delta t} + \sum_{j\in I, j\in N(i)} \frac{s_{ij}}{v_i}\left[(A^- + D)(\mathbf{U}_i^k, \mathbf{U}_j^k)\right]\left(\delta \mathbf{U}_j^{k+1} - \delta \mathbf{U}_i^{k+1}\right)\\
&\quad = -\frac{\mathbf{U}_i^k - \mathbf{U}_i^n}{\Delta t} - \sum_{j\in N(i)} \frac{s_{ij}}{v_i}\left[(A^- + D)(\mathbf{U}_i^k, \mathbf{U}_j^k)\right](\mathbf{U}_j^k - \mathbf{U}_i^k)\\
&\qquad - \sum_{j\notin I, j\in N(i)} \frac{s_{ij}}{v_i}\left[(A^- + D)(\mathbf{U}_i^k, \mathbf{U}_j^k)\right]\delta\phi_{ij}^{k+1}
\end{aligned}$$

We define $\mathbf{U}_I = (\mathbf{U}_1, \dots, \mathbf{U}_m)^t$ the unknown vector of the subdomain I,

$$\delta\phi_I = (\delta\phi_{ij})_{i\in I, j\in J, j\in N(i)}, \tag{11}$$

$\mathscr{A}_I$ the local Neumann matrix of the subdomain I, and $P_I = \sum_{j\notin I, j\in N(i)} \frac{s_{ij}}{v_i}\left[A^-(\mathbf{U}_{Roe}^k) + D(\mathbf{U}_{diff}^k)\right]$, we can write the linear system as:

$$\mathscr{A}_I(\mathbf{U}^k)\delta \mathbf{U}_I^{k+1} = b_I(\mathbf{U}^n, \mathbf{U}^k) - P_I\delta\phi_I \tag{12}$$

By taking into account Eqs. (10)–(12), and denoting $\delta\Phi = (\delta\phi_I)$, $I = 1\dots N$ we can build an extended system that distinguishes the internal unknowns from the interface ones:

$$\left(\begin{array}{cccc|c} \mathscr{A}_1 & 0 & \dots & \dots & P_1 \\ 0 & \mathscr{A}_2 & 0 & \dots & P_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & \mathscr{A}_N & P_N \\ \hline M_1 & \dots & \dots & M_N & \mathbb{I} \end{array}\right) \begin{pmatrix} \delta\mathbf{U}_1 \\ \delta\mathbf{U}_2 \\ \dots \\ \delta\mathbf{U}_N \\ \delta\Phi \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_N \\ b_\Phi \end{pmatrix} \tag{13}$$

where $\mathscr{A}_I$ is the matrix that couples the unknowns associated with internal cells of Ω_I whereas M_I links $\delta\mathbf{U}_I$ to $\delta\Phi$ through (10). Then, in our method, M_I comprises only 0 or ±1.

The internal unknowns in (13) can be eliminated in favor of the interface ones to yield the following interface system:

$$S\delta\Phi = b_\Phi, \tag{14}$$

with $(S\delta\Phi) = \delta\Phi + \sum_{I=1}^{N} M_I \mathscr{A}_I{}^{-1} P_I \delta\phi_I$ and $(b_\Phi) = \sum_{I=1}^{N} M_I A_I{}^{-1} b_I$.

The computation of the matrix S is so costly as we have to inverse the local matrix $\mathscr{A}_I$. Fortunately, we do not have to compute explicitly the coefficients of S. All we need is to design the operator $\delta\Phi \to S\delta\Phi$. Then Eq. (14) can be solved by, e.g., GMRES, BICGStab, or the Richardson methods. Once we solved the interface system, we know $\delta\Phi$ and then we can solve the internal unknowns on each processor using Eq. (12).

5 Numerical Results

We have implemented our method for the compressible Navier-Stokes equations and the isentropic two-fluid model and compared the results obtained using single and multiple domains. After this validation, we compare the computation time of the ILU preconditioner, our method and our method with strategy Scaling [3].

Figure 1 presents the computational time required to perform a time step of a fixed global problem of one million cells using upwind scheme. We compare the computational time required using the classical distributed method (red curve), the domain decomposition method (blue curve) and the domain decomposition method with scaling (green curve). We vary the number of processors up to 128. One can see that the domain decomposition method is comparable with classical distributed method and using scaling [3] is better.

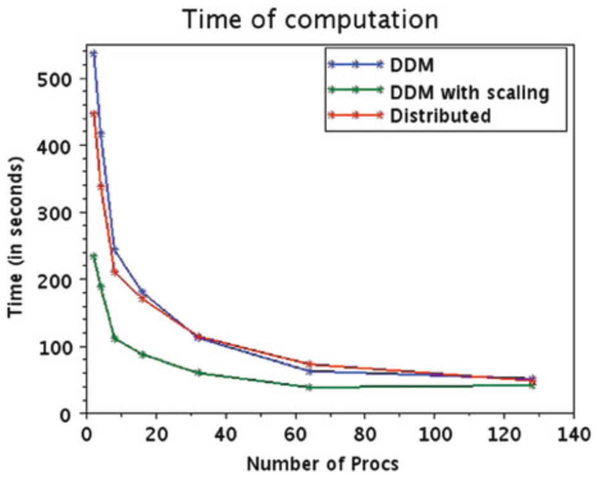

Fig. 1 Upwind scheme, single-phase flow, global mesh = 96 × 96 × 96, CFL 20

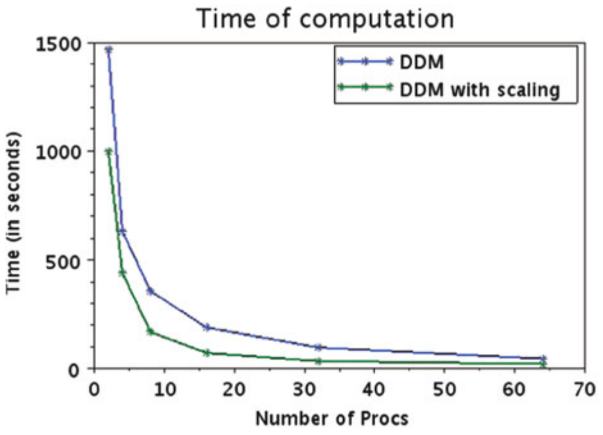

Fig. 2 Centered scheme, single-phase flow, global mesh = 96 × 96 × 96, CFL 10

Figure 2 shows the computational time required to perform the previous test but using centered scheme. We can see only two curves. This is because, in this case the classical distributed method does not converge like we use the centered scheme. Domain decomposition is the only one method that converges.

Similarly, Figs. 3 and 4 show the computational time required to perform a time step in the case of the two-phase flow for the upwind and centered schemes.

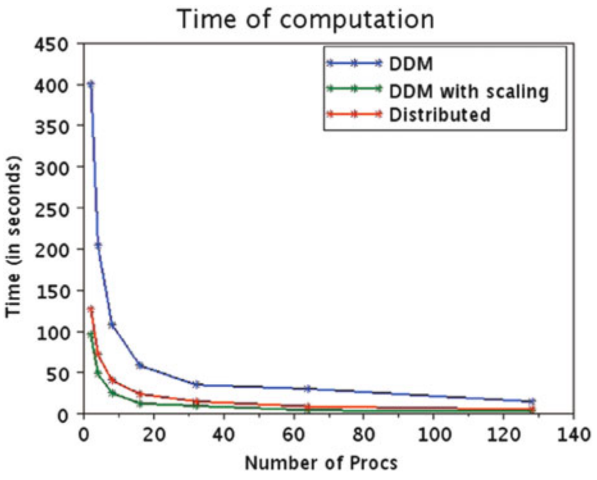

Fig. 3 Upwind scheme, two-phase flow, global mesh $= 96 \times 96 \times 96$, CFL 20

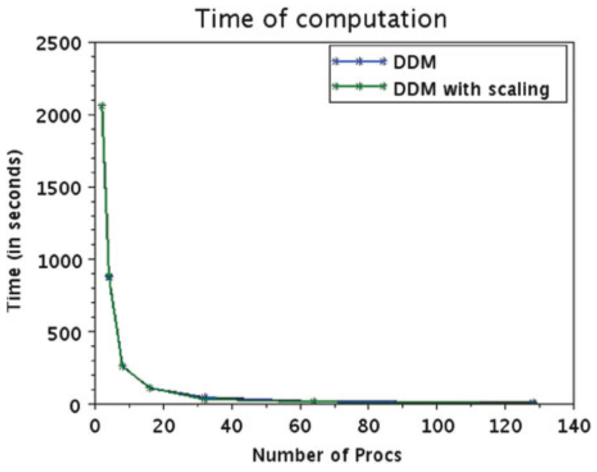

Fig. 4 Centered scheme, two-phase flow, global mesh $= 96 \times 96 \times 96$, CFL 20

6 Conclusion

We have presented a new interface variable which allows for the treatment of diffusion terms and the use of various numerical schemes for two-phase flows. We also introduced the Scaling strategy to improve the conditioner number of the matrix and reduce the computational time. We compared the scalability of our method with the classical distributed computations. Numerical results showed that our method is more robust and efficient.

References

1. Benzi, M.: Preconditioning techniques for large linear systems: a survey. J. Comput. Phys. **182**, 418–477 (2002)
2. Bergeaud, V., Fillion, P., Dérouillat, J.: Etude bibliographique sur l'inversion parallèle des matrices ovap. Tech. rep., Rapport CS/311-1/AB06A002-010/RAP/07-065 version 1.0 (2007)
3. Dao, T., Ndjinga, M., Magoulès, F.: Comparison of upwind and centered schemes for low Mach number flows. In: Proceedings of the International Symposium Finite Volumes for Complex Application VI. Springer Proceedings in Mathematics, vol. 4, pp. 303–311 (2011)
4. Dao, T., Ndjinga, M., Magoulès, F.: A Schur complement method for compressible Navier-Stokes equations. In: Proceedings of the 20th International Conference on Domain Decomposition Methods (2011)
5. Dolean, V., Lanteri, S.: A domain decomposition approach to finite volume solution of the Euler equations on unstructured triangular meshes. Int. J. Numer. Methods Fluids **37**(6), 625–656 (2001)
6. Drew, D.A., Passman, S.L.: Theory of Multicomponents Fluids. Springer, New York (1999)
7. Fillion, P., Chanoine, A., Dellacherie, S., Kumbaro, A.: Flica-ovap: a new platform for core thermal-hydraulic studies. In: NURETH-13 (2009)
8. Ishii, M.: Thermo-Fluid Dynamic Theory of Two-Phase Flow. Eyrolles, Paris (1975)
9. Kruis, J.: Domain Decomposition Methods for Distributed Computing. Saxe-Coburg Publications, Stirling (2006)
10. Maday, Y., Magoulès, F.: Absorbing interface conditions for domain decomposition methods: a general presentation. Comput. Methods Appl. Mech. Eng. **195**(29–32), 3880–3900 (2006)
11. Magoulès, F., Roux, F.X.: Lagrangian formulation of domain decomposition methods: a unified theory. Appl. Math. Model. **30**(7), 593–615 (2006)
12. Quarteroni, A., Valli, A.: Domain Decomposition Methods for Partial Differential Equations. Oxford University Press, Oxford (1999)
13. Smith, B., Bjorstad, P., Gropp, W.: Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations. Cambridge University Press, Cambridge (1996)
14. Toselli, A., Widlund, O.B.: Domain Decomposition Methods—Algorithms and Theory. Springer, Berlin (2005)
15. Toumi, I., Caruge, D.: An implicit second order method for 3d two-phase flow calculations. Nucl. Sci. Eng. **130**, 213–225 (1998)
16. Toumi, I., Kumbaro, A.: An approximate linearized Riemann solver for a two-fluid model. J. Comput. Phys. **124**, 286–300 (1996)