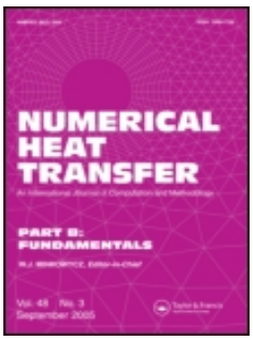


Taylor & Francis
Taylor & Francis Group

Numerical Heat Transfer, Part B: Fundamentals

ISSN: 1040-7790 (Print) 1521-0626 (Online) Journal homepage: http://www.tandfonline.com/loi/unhb20

A COMPARATIVE ASSESSMENT OF THE PERFORMANCE OF MASS CONSERVATION-BASED ALGORITHMS FOR INCOMPRESSIBLE MULTIPHASE FLOWS

F. Moukalled & M. Darwish

To cite this article: F. Moukalled & M. Darwish (2002) A COMPARATIVE ASSESSMENT OF THE PERFORMANCE OF MASS CONSERVATION-BASED ALGORITHMS FOR INCOMPRESSIBLE MULTIPHASE FLOWS, Numerical Heat Transfer, Part B: Fundamentals, 42:3, 259-283, DOI: 10.1080/10407790260233565

To link to this article: http://dx.doi.org/10.1080/10407790260233565

Numerical Heat Transfer, Part B, 42: 259–283, 2002

1040-7790/02 $12.00 + .00
DOI: 10.1080/10407790190053941

A COMPARATIVE ASSESSMENT OF THE PERFORMANCE OF MASS CONSERVATION-BASED ALGORITHMS FOR INCOMPRESSIBLE MULTIPHASE FLOWS

F. Moukalled and M. Darwish

American University of Beirut, Faculty of Engineering & Architecture, Mechanical Engineering Department, Riad El Solh, Beirut, Lebanon

This work is concerned with the implementation and testing, within a structured collocated finite-volume framework, of seven incompressible-segregated multiphase flow algorithms that belong to the mass conservation-based algorithms (MCBA) group in which the pressure-correction equation is derived from overall mass conservation. The pressure-correction schemes in these algorithms are based on SIMPLE, SIMPLEC, SIMPLEX, SIMPLEM, SIMPLEST, PISO, and PRIME. The performance and accuracy of the multiphase algorithms are assessed by solving eight one-dimensional two-phase flow problems spanning the spectrum from dilute bubbly to dense gas–solid flows. The main outcome of this study is a clear demonstration of the capability of all MCBA algorithms to deal with multiphase flow situations. Moreover, results displayed in terms of convergence history plots and CPU times indicate that the performance of the MCBA versions of SIMPLE, SIMPLEC, and SIMPLEX are very close. In general, the performance of SIMPLEST approaches that of SIMPLE for diffusion-dominated flows. As expected, the PRIME algorithm is found to be the most expensive, due to its explicit treatment of the phasic momentum equations. The PISO algorithm is generally more expensive than SIMPLE, and its performance depends on the type of flow and solution method used. The behavior of SIMPLEM is consistent, and in terms of CPU effort it stands between PRIME and SIMPLE.

INTRODUCTION

The extensive developments that have taken place in computational fluid dynamics (CFD) over the last three decades have established this still-evolving technology as a reliable and essential tool for the simulation and optimization of a wide variety of engineering fluid flow processes (mixing, solidification, turbulence, etc.). Several issues that were hindering its progress have been addressed and remedies suggested. Concerns related to accuracy were assuaged through the development of high-resolution (HR) schemes [1–3]. Moreover, better solution algorithms [4–8], solvers [9, 10], and multigrid techniques [11, 12] have greatly reduced the computational cost and made it feasible to solve real-life problems.

Received 15 January 2002; accepted 26 March 2002.

The financial support provided by the University Research Board of the American University of Beirut through Grant 14886073129 is gratefully acknowledged.

Address correspondence to Dr. F. Moukalled, American University of Beirut, Mechanical Engineering Department, P.O. Box 11-0236, Riad El Solh, Beirut 1107 2020, Lebanon. E-mail: memouk@aub.edu.lb

NOMENCLATURE

$A_p^{(k)}, \ldots$	coefficients in the discretized equation for $\phi^{(k)}$	$Q^{(k)}$	general source term of fluid/phase k
		$r^{(k)}$	volume fraction of fluid/phase k
$B_p^{(k)}$	source term in the discretized equation for $\phi^{(k)}$	$\mathbf{S}_f$	surface vector
		t	time
$\mathbf{B}^{(k)}$	body force per unit volume of fluid/phase k	$U_f^{(k)}$	interface flux velocity $(\mathbf{v}_f^{(k)} \cdot \mathbf{S}_f)$ of fluid/phase k
$\mathbf{D}_p^{(k)}[\phi^{(k)}]$	vector form of the D operator	$\mathbf{u}^{(k)}$	velocity vector of fluid/phase k
$H_p[\phi^{(k)}]$	H operator	$\Gamma^{(k)}$	diffusion coefficient of fluid/phase k
$\mathbf{HP}_p[\mathbf{u}^{(k)}]$	vector form of the HP operator	δt	time step
$\mathbf{I}^{(k)}$	interphase momentum transfer	$\Delta_p[\phi^{(k)}]$	Δ operator
$\mathbf{J}_f^{(k)D}$	diffusion flux of $\phi^{(k)}$ across cell face f	$\nu^{(k)}, \mu^{(k)}$	kinematic and dynamic viscosity of fluid/phase k
$\mathbf{J}_f^{(k)C}$	convection flux of $\phi^{(k)}$ across cell face f	$\rho^{(k)}$	density of fluid/phase k
		$\phi^{(k)}$	general scalar quantity associated with fluid/phase k
$\dot{M}^{(k)}$	mass source per unit volume		
P	pressure	Ω	cell volume

While high-resolution schemes, solvers, multigrid techniques, etc., can be applied to simulate both single- and multiphase flows, nearly all developments in solution algorithms have been directed toward the simulation of single-fluid flow. In particular, many segregated single-fluid solution algorithms have been developed, such as the well-known SIMPLE [4], SIMPLEST [13], SIMPLEC [6], SIMPLEM [14], PISO [5], PRIME [15], and SIMPLEX [7] algorithms, to cite a few. Additionally, several techniques have been devised to improve the performance, facilitate the implementation, and extend the capability of these algorithms. On the other hand, developments in solution algorithms for simulating multiphase flow phenomena have lagged behind that of single-phase flow algorithms due to the much higher computational cost involved, the numerical difficulties that first had to be addressed in the simulation of single-phase flow, and the increase in algorithmic complexity. While the major difficulty in the simulation of single-phase flow stems from the coupling between the momentum and continuity equations, in the simulation of multiphase flow phenomena this problem is further complicated by the fact that there are as many sets of continuity and momentum equations as there are fluids, they are all coupled together in various ways (interchange momentum by interphase mass and momentum transfer, etc.) and the fluids share space (the volume fractions sum to unity, but are not known in advance).

Despite these complexities, successful segregated pressure-based solution algorithms have been devised. The IPSA variants devised by the Spalding group at Imperial College [16–18] and the set of algorithms devised by the Los Alamos Scientific Laboratory (LASL) group [19–21] are examples of multiphase algorithms. However, in contrast with the widespread information available on single-fluid solution algorithms, little information is available on multiphase solution algorithms, and even less on their relative performance.

Recently, Darwish et al. [22] extended the large number of segregated single-fluid flow algorithms reviewed in [8] to predict multiphase flow phenomena and showed that the pressure-correction equation can be derived either by using the

geometric conservation equation or the overall mass conservation equation. Depending on the chosen equation, the segregated pressure-based multiphase flow algorithms were classified as either the geometric conservation-based family of algorithms (GCBA) or the mass conservation-based family of algorithms (MCBA). Many of these algorithms have neither been tested nor implemented in CFD codes.

The objective of the present work is to implement and test seven multiphase algorithms from the MCBA group and to assess their relative performance by solving a total of eight one-dimensional incompressible two-phase flow problems encompassing dilute and dense gas–solid flows in addition to bubbly flows on several grid sizes.

In what follows, the equations governing incompressible multiphase flow phenomena are first introduced, followed by a brief description of the discretization procedure. Then the capability of MCBA to predict incompressible multiphase flow phenomena is demonstrated, and their performance characteristics (in terms of convergence history and speed) assessed.

THE GOVERNING EQUATIONS

In incompressible multiphase flow the various fluids/phases coexist with different concentrations at different locations in the flow domain and move with unequal velocities. Thus, the equations governing multiphase flows are the following conservation laws of mass and momentum for each individual fluid:

$$\frac{\partial\left(r^{(k)}\rho^{(k)}\right)}{\partial t}+\nabla\cdot\left(r^{(k)}\rho^{(k)}\mathbf{u}^{(k)}\right)=r^{(k)}\dot{M}^{(k)} \tag{1}$$

$$\begin{aligned}\frac{\partial\left(r^{(k)}\rho^{(k)}\mathbf{u}^{(k)}\right)}{\partial t}&+\nabla\cdot\left(r^{(k)}\rho^{(k)}\mathbf{u}^{(k)}\mathbf{u}^{(k)}\right)\\&=\nabla\cdot\left[r^{(k)}\mu^{(k)}\nabla\mathbf{u}^{(k)}\right]+r^{(k)}\left(-\nabla P+\mathbf{B}^{(k)}\right)+\mathbf{I}_M^{(k)}\end{aligned} \tag{2}$$

where the superscript (k) refers to the kth phase, $r^{(k)}$ the volume fraction $(\Omega^{(k)}/\Omega)$, $\rho^{(k)}$ the phasic density, $\mathbf{u}^{(k)}$ the velocity vector, P the pressure (assumed to be shared by all fluids/phases), $\mathbf{B}^{(k)}$ the body force per unit volume, $\mu^{(k)}$ the laminar viscosity, and $\mathbf{I}_M^{(k)}$ represents the interfacial forces per unit volume due to drag, virtual mass effects, lift, etc.

If a typical representative variable associated with phase (k) is denoted by $\phi^{(k)}$, the above two equations can be written using a general phasic equation as

$$\frac{\partial\left(r^{(k)}\rho^{(k)}\phi^{(k)}\right)}{\partial t}+\nabla\cdot\left(r^{(k)}\rho^{(k)}\mathbf{u}^{(k)}\phi^{(k)}\right)=\nabla\cdot\left(r^{(k)}\Gamma^{(k)}\nabla\phi^{(k)}\right)+r^{(k)}Q^{(k)} \tag{3}$$

where the expressions for $\Gamma^{(k)}$ and $Q^{(k)}$ can be deduced from the parent equations.

The above set of differential equations has to be solved in conjunction with constraints on certain variables represented by algebraic relations. For incompressible laminar multiphase flow, these auxiliary relations include the geometric conservation equation $\left(\sum_k r^{(k)}=1\right)$, and the interfacial mass and momentum transfers. Several models have been developed for the interfacial mass and

momentum transfers terms. In this work, only interfacial momentum transfer is of interest, and its closure will be detailed later. Moreover, in order to present a closed mathematical model, initial and boundary conditions should supplement the above equations.

DISCRETIZATION PROCEDURE

The general conservation equation (3) is integrated over a finite volume (Figure 1*a*) to yield

$$\int\int_{\Omega} \frac{\partial(r^{(k)}\rho^{(k)}\phi^{(k)})}{\partial t}\, d\Omega + \int\int_{\Omega} \nabla\cdot(r^{(k)}\rho^{(k)}\mathbf{u}^{(k)}\phi^{(k)})d\Omega = \int\int_{\Omega} \nabla\cdot(r^{(k)}\Gamma^{(k)}\nabla\phi^{(k)})\, d\Omega + \int\int_{\Omega} r^{(k)}Q^{(k)}\, d\Omega \tag{4}$$

where Ω is the volume of the control cell. Using the divergence theorem to transform the volume integral into a surface integral and then replacing the surface integral by a summation of the fluxes over the sides of the control volume, Eq. (4) is transformed to

$$\frac{\partial(r^{(k)}\rho^{(k)}\phi^{(k)})}{\partial t}\Omega + \sum_{nb}(\mathbf{J}_{nb}^{(k)D} + \mathbf{J}_{nb}^{(k)C}) = r^{(k)}Q^{(k)}\Omega \tag{5}$$

where $\mathbf{J}_{nb}^{(k)D}$ and $\mathbf{J}_{nb}^{(k)C}$ are the diffusive and convective fluxes, respectively. The discretization of the diffusion term is second-order-accurate and follows the derivations presented in [23]. For the convective terms, the high-resolution SMART [1] scheme is employed and applied within the context of the NVSF methodology [3].

After substituting the face values by their functional relationship relating to the node values of ϕ, Eq. (5) is transformed after some algebraic manipulations into the following discretized equation:

$$A_P^{(k)}\phi_P^{(k)} = \sum_{\mathrm{NB}} A_{\mathrm{NB}}^{(k)}\phi_{\mathrm{NB}}^{(k)} + B_P^{(k)} \tag{6}$$

where the coefficients $A_p^{(k)}$ and $A_{\mathrm{NB}}^{(k)}$ depend on the selected scheme and $B_p^{(k)}$ is the source term of the discretized equation . In compact form, the above equation can be written as

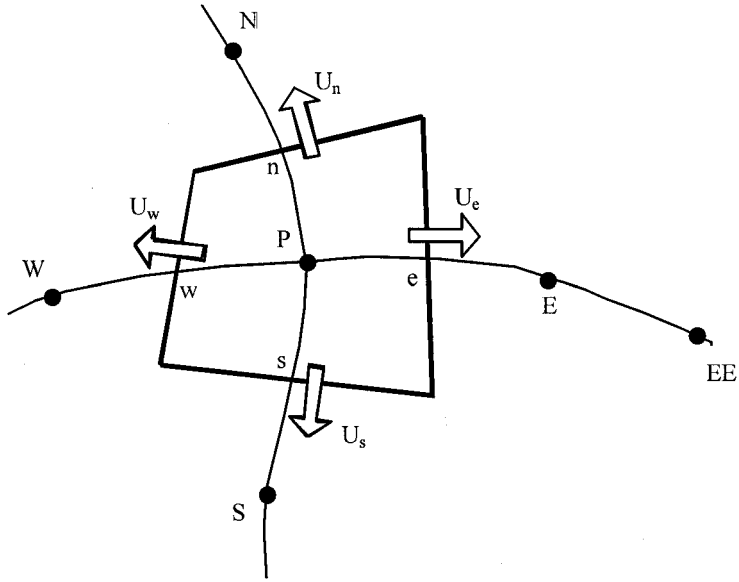
$$\phi^{(k)} = H_P[\phi^{(k)}] = \frac{\sum_{\mathrm{NB}} A_{\mathrm{NB}}^{(k)}\phi_{\mathrm{NB}}^{(k)} + B_P^{(k)}}{A_P^{(k)}} \tag{7}$$

The discretization procedure for the momentum equation yields an algebraic equation of the form:

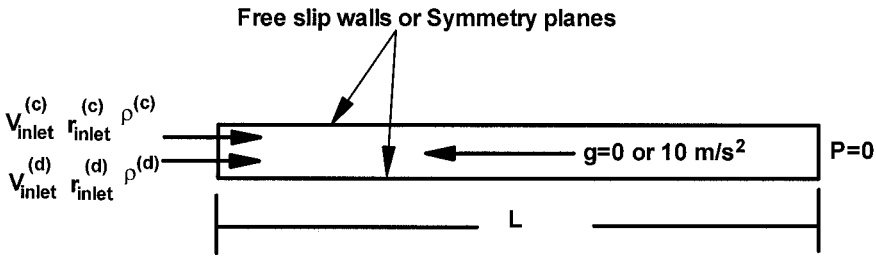
$$\mathbf{u}_P^{(k)} = \mathbf{HP}_P[\mathbf{u}^{(k)}] - r^{(k)}\boldsymbol{D}_P^{(k)}\nabla_P(P) \tag{8}$$

Moreover, the phasic mass conservation equation [Eq. (1)] can be viewed as a phasic volume fraction equation, which can be written as

$$r_P^{(k)} = H_P[r^{(k)}] \tag{9}$$



(a)



(b)

Figure 1. (*a*) Control volume. (*b*) Physical domain for the gas–particle transport problem.

or as a phasic continuity equation to be used in deriving the pressure-correction equation:

$$\frac{(r_P^{(k)}\rho_P^{(k)}) - (r_P^{(k)}\rho_P^{(k)})^{\text{Old}}}{\delta t}\Omega + \Delta_P[r^{(k)}\rho^{(k)}\mathbf{u}^{(k)} \cdot \mathbf{S}] = r^{(k)}\dot{M}^{(k)} \tag{10}$$

where the Δ operator represents the following operation:

$$\Delta_P[\Theta] = \sum_{f=\text{nb}(P)} \Theta_f \tag{11}$$

THE MASS CONSERVATION-BASED ALGORITHMS

The numbers of equations describing an n-fluid/phase flow situation are: n phasic momentum equations, n phasic volume fraction (or mass conservation) equations, and a geometric conservation equation. Moreover, the variables involved are the n phasic velocity vectors, the n phasic volume fractions, and the pressure field. In all MCBA algorithms, the n momentum equations are used to calculate the n velocity fields; $n-1$ volume fraction (mass conservation) equations are used to calculate $n-1$ volume fraction fields, and the last volume fraction field is calculated using the geometric conservation equation

$$r^{(n)} = 1 - \sum_{k \neq n} r^{(k)} \tag{12}$$

The remaining volume fraction equation can be used to calculate the pressure field. However, instead of using this last volume fraction equation, the global conservation equation is employed, i.e., the sum of the individual mass conservation equations, to derive a pressure-correction equation. The sequence of events in the MCBA is as follows:

1. Solve the phasic momentum equations for velocities.
2. Solve the pressure-correction equation based on global mass conservation.
3. Correct velocities and pressure.
4. Solve the phasic mass conservation equations for volume fractions.
5. Return to the first step and repeat until convergence.

THE MCBA PRESSURE-CORRECTION EQUATION

To derive the pressure-correction equation, the mass conservation equations of the various fluids are added to yield the global mass conservation equation given by

$$\sum_k \left\{ \frac{(r_P^{(k)} \rho_P^{(k)}) - (r_P^{(k)} \rho_P^{(k)})^{\mathrm{Old}}}{\delta t} \Omega + \Delta_P (r^{(k)} \rho_P^{(k)} \mathbf{u}^{(k)} \cdot \mathbf{S}) \right\} = 0 \tag{13}$$

In the predictor stage a guessed or an estimated pressure field from the previous iteration, denoted by P°, is substituted into the momentum equations. The resulting velocity fields denoted by $\mathbf{u}^{(k)*}$, which now satisfy the momentum equations will not, in general, satisfy the mass conservation equations. Thus, corrections are needed in order to yield velocity and pressure fields that satisfy both equations. Denoting the corrections for pressure and velocity by P' and $\mathbf{u}^{(k)\prime}$, respectively, the corrected fields are written as

$$P = P^{\circ} + P' \qquad \mathbf{u}^{(k)} = \mathbf{u}^{(k)*} + \mathbf{u}^{(k)\prime} \tag{14}$$

Hence the equations solved in the predictor stage are

$$\mathbf{u}_P^{(k)*} = \mathbf{H}\mathbf{P}_P[\mathbf{u}^{(k)*}] - r^{(k)} \mathbf{D}_P^{(k)} \nabla_P P^{\circ} \tag{15}$$

while the final solutions satisfy

$$\mathbf{u}_P^{(k)} = \mathbf{HP}_P[\mathbf{u}^{(k)}] - r^{(k)}\mathbf{D}_P^{(k)}\nabla_P P \tag{16}$$

Subtracting the two equation sets [(15) and (16)] from each other yields the following equation involving the correction terms:

$$\mathbf{u}_P^{(k)'} = \mathbf{HP}_P[\mathbf{u}^{(k)'}] - r^{(k)}\mathbf{D}_P^{(k)}\nabla_P P' \tag{17}$$

Moreover, substituting Eqs. (14) and (17) into Eq. (13) and rearranging, the final form of the pressure-correction equation is written as

$$\sum_k \{\Delta_P[r^{(k)}\rho^{(k)^*}(r^{(k)}\mathbf{D}^{(k)}\nabla P')\cdot\mathbf{S}]\} = \sum_k \left\{ \begin{array}{l} \frac{r_P^{(k)}\rho_P^{(k)^*} - \left(r_P^{(k)}\rho_P^{(k)}\right)^{\mathrm{Old}}}{\delta t}\Omega + \Delta_P[r^{(k)}\rho^{(k)^*}U^{(k)^*}] \\ +\Delta_P[r^{(k)}\rho^{(k)^*}(\mathbf{HP}[\mathbf{u}^{(k)'}])\cdot\mathbf{S}] \end{array} \right\} \tag{18}$$

If the $\mathbf{HP}[\mathbf{u}^{(k)\prime}]$ term in the above equation is retained, there will result a pressure-correction equation relating the pressure-correction value at a point to all values in the domain. To facilitate implementation and reduce cost, simplifying assumptions related to this term have been introduced. Depending on these assumptions, different algorithms are obtained. These algorithms were accorded a full-length article [22] of discussion, to which interested readers are referred. The corrections are then applied to the velocity and pressure fields using the following equations:

$$\mathbf{u}_P^{(k)^*} = \mathbf{u}_P^{(k)^*} - r^{(k)}\mathbf{D}_P^{(k)}\nabla_P P' \qquad P^* = P^\circ + P' \tag{19}$$

Numerical experiments using the above approach to simulate two-fluid flow with large difference in densities have shown poor conservation of the lighter fluid. This problem can be considerably alleviated by normalizing the individual continuity equations (see [22] for details) by means of a weighting factor such as a reference density $\rho^{(k)}$ (which is fluid dependent). This approach has been adopted in solving all problems presented in this work.

RESULTS AND DISCUSSION

Due to the large number of parameters affecting the performance of the various multiphase mass conservation-based algorithms and to allow a thorough testing of these algorithms, eight one-dimensional two-phase problems are considered. These problems can be broadly classified as: (1) horizontal particle transport, and (2) vertical particle transport. Results are presented in terms of the convergence history and the CPU time needed to converge the solution to a set level. Predictions are compared against available numerical/theoretical values. The residual of a variable ϕ at the end of an outer iteration is defined as

$$\mathrm{RES}_\phi^{(k)} = \sum_{c.v} \left| A_p\phi_p^{(k)} - \sum_{\text{all } p \text{ neighbors}} A_{\mathrm{nb}}\phi_{\mathrm{nb}}^{(k)} - B_p^{(k)} \right| \tag{20}$$

For global mass conservation, the imbalance in mass is defined as

$$\mathrm{RES}_C = \sum_k \sum_{c.v.} \left| \frac{(r_P^{(k)} \rho_P^{(k)}) - (r_P^{(k)} \rho_P^{(k)})^{\mathrm{Old}}}{\delta t} \Omega - \Delta_P [r^{(k)} \rho^{(k)} \mathbf{u}^{(k)} \cdot \mathbf{S}] - r^{(k)} \dot{M}^{(k)} \right| \tag{21}$$

All residuals are normalized by their respective inlet fluxes. Computations are terminated when the maximum normalized residual of all variables drops below a very small number ε_s. For a given problem, the same value of ε_s is used with all algorithms. In general, it is found that requiring the overall mass residuals to be satisfied to within ε_s is a very stringent and sufficient requirement. This is why these residuals are the ones presented here and used to compare the performance of the various algorithms. In all problems, the first phase represents the continuous phase [denoted by a superscript (c)], which must be fluid, and the second phase is the disperse phase [denoted by a superscript (d)], which may be solid or fluid. Unless otherwise specified, the HR SMART scheme is used in all computations reported in this study. For a given problem, all solutions are obtained starting from the same initial guess. Moreover, it should be stated that in iterative techniques, different initial guesses might require different computational efforts.

Despite its geometric simplicity, the one-dimensional particle transport problem can represent a wide range of physical conditions. The effects of grid refinement on accuracy and convergence are studied by solving the problems on four grid systems of sizes 20, 40, 80, and 160 control volumes, with ε_s assigned the value of 10^{-8}.

Many runs were performed so as to set the control parameters of each algorithm near optimum values. To allow a comparative assessment of performance, the CPU times are reported in the form of graphs. Moreover, all CPU times are normalized by the time needed by MCBA-SIMPLE to reach the set residuals on the coarsest grid.

Horizontal Particle Transport

The physical situation is depicted in Figure 1*b*. Depending on the set densities, it represents either the steady flow of solid particles suspended in a free stream of air or the steady flow of air bubbles in a stream of water. The slip between the phases determines the drag, which is the sole driving force for the particle-bubble/air-water motion ($g = 0$). In the suspension, the interparticle/bubble forces are neglected. Diffusion within both phases is set to zero while the interphase drag force is calculated as

$$I_M^{(c)} = \frac{3}{8} \frac{C_D}{r_p} r^{(d)} \rho^{(c)} V_{\mathrm{slip}} (u^{(d)} - u^{(c)}) \tag{22}$$

$$I_M^{(d)} = -\frac{3}{8} \frac{C_D}{r_p} r^{(d)} \rho^{(c)} V_{\mathrm{slip}} (u^{(d)} - u^{(c)}) \tag{23}$$

$$V_{\mathrm{slip}} = \left\| \mathbf{u}^{(d)} - \mathbf{u}^{(c)} \right\| \tag{24}$$

The drag coefficient, C_D, is set to 0.44. Since phasic diffusion is neglected, the MCBA-SIMPLEST and MCBA-PRIME become identical and reference will be made to MCBA-SIMPLEST only. The task is to calculate the particle/bubble-

velocity distribution as a function of position. If the flow field is extended far enough (here computations are performed over a length of $L = 2$ m), the particle/bubble and fluid phases are expected to approach an equilibrium velocity given by

$$U_{\text{equilibrium}} = r_{\text{inlet}}^{(c)} V_{\text{inlet}}^{(c)} + r_{\text{inlet}}^{(d)} V_{\text{inlet}}^{(d)} \tag{25}$$

Problem 1: Dilute gas–solid flow. The steady flow of dilute particles suspended in a free stream of air is studied first. At the inlet, the air and particle velocities are 5 and 1 m/s, respectively. The physical properties of the two phases are $\rho^{(d)}/\rho^{(c)} = 2,000$, $r_p = 1$ mm, $r_{\text{inlet}}^{(d)} = 10^{-5}$. Due to the dilute concentration of the particles, the free-stream velocity is more or less unaffected by their presence and the equilibrium velocity is nearly equal to the inlet free-stream velocity. Based on this observation, Morsi and Alexander [24] obtained the following analytical solution for the particle velocity $u^{(d)}$ as a function of the position x and the properties of the two phases:

$$\ln[V_{\text{inlet}}^{(c)} - u^{(d)}] + \frac{V_{\text{inlet}}^{(c)}}{V_{\text{inlet}}^{(c)} - u^{(d)}} = \frac{3}{8}\frac{\rho^{(c)}}{\rho^{(d)}}\frac{C_D}{r_p}x + \ln[V_{\text{inlet}}^{(c)} - V_{\text{inlet}}^{(d)}] + \frac{V_{\text{inlet}}^{(c)}}{V_{\text{inlet}}^{(c)} - V_{\text{inlet}}^{(d)}} \tag{26}$$

This case is of particular importance since the flow situation has an exact solution. As shown in Figure 2*a* the predicted particle velocity distribution falls on top of the analytical solution given by Eq. (26), which is an indication of the accuracy of the numerical procedure. The convergence histories of the various MCBA over the four grid networks used are displayed in Figures 2*b*–2*h*. For all algorithms, the required number of iterations increases as the grid size increases, with PISO (Figure 2*b*) requiring the minimum and SIMPLEST/PRIME (Figure 2*f*) the maximum number of iterations on all grids. The convergence histories of SIMPLE, SIMPLEC, and SIMPLEX (Figures 2*c*, 2*d*, and 2*g*, respectively) are very similar, requiring nearly the same number of iterations on all grids. The convergence path of SIMPLEM (Figure 2*e*) is not as smooth as that of SIMPLE, due to the fact that at the end of an outer iteration, the velocity field is momentum satisfying rather than continuity satisfying. The convergence paths of the various algorithms over a grid of size 80 control volumes (C.V.) are compared in Figure 2*h*, and the above observations are easily inferred from the figure.

Problem 2: Dense gas–solid flow. The only difference between this case and the previous one is in the concentration of particles, which is set to $r_{\text{inlet}}^{(d)} = 10^{-2}$. Despite the low value of the inlet disperse-phase volume fraction, the ratio of disperse-phase and continuous-phase mass loadings is large: $r^{(d)}\rho^{(d)}/r^{(c)}\rho^{(c)} = 20$. Thus the disperse phase carries most of the inertia of the mixture. The equilibrium velocity in this case, as obtained from Eq. (25), is 4.96 m/s, as compared to 4.99996 m/s in the previous case. Due to this slight difference between the inlet air velocity and the final equilibrium velocity, the free-stream velocity may be assumed to be nearly constant and the variation in particle velocity can be obtained again from Eq. (26). The predicted air and particle velocity distributions are displayed in Figure 3*a*. The numerical and analytical particle velocity profiles are indistinguishable and fall on top of each other (denoted solid in the figure). Moreover, the slight decrease in the air velocity can be easily depicted. The

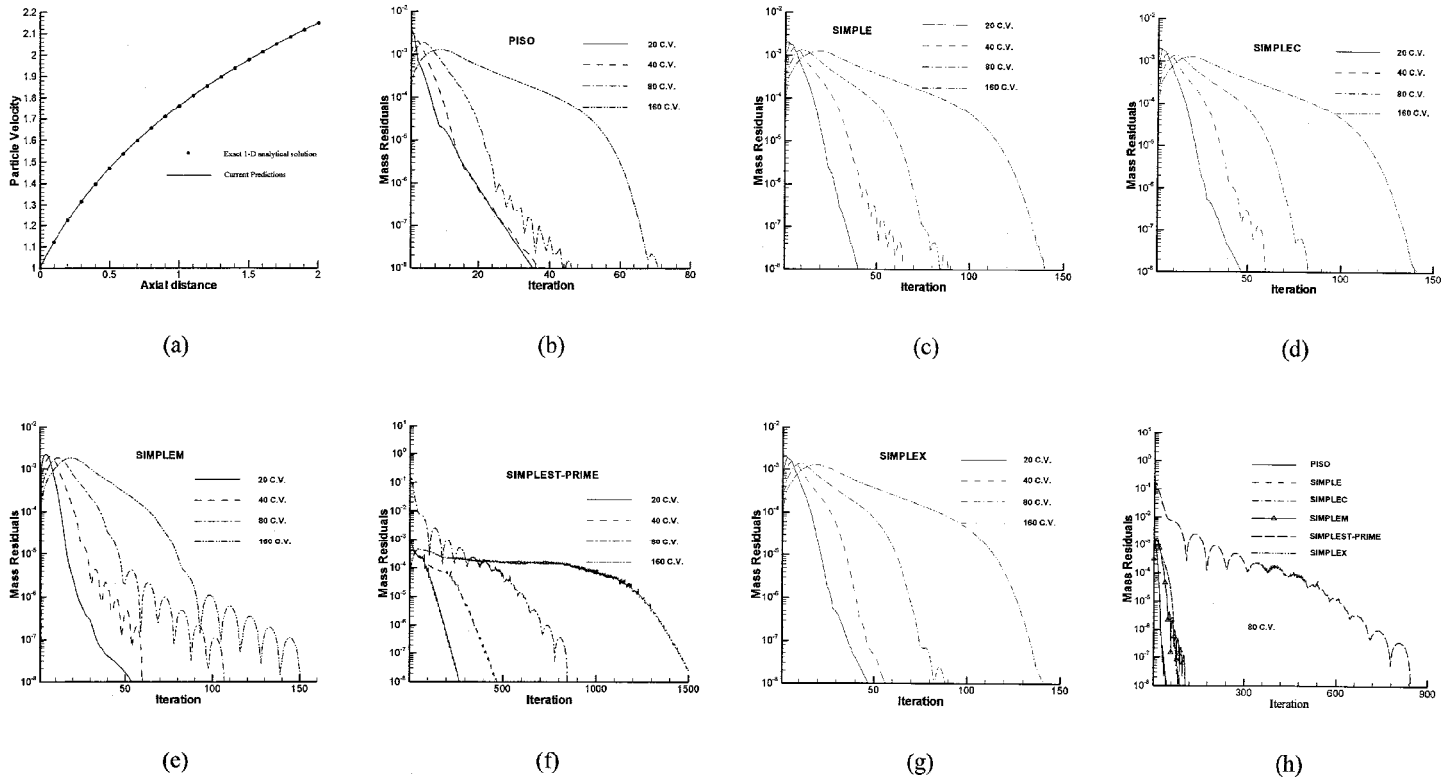


Figure 2. (*a*) Comparison between analytical and numerical particle velocity distributions. (*b*)–(*g*) Convergence histories on the different grid systems. (*h*) Convergence histories on the 80-C.V. grid for the horizontal dilute gas–solid flow problem.

convergence paths for all algorithms and over all grid systems used are displayed in Figures 3*b*–3*h*. In general, a larger number of iterations is required to reach the desired level of convergence on a given grid as compared to the dilute case, due to the increased importance of the interphase term. The general convergence trend is similar to that of the dilute problem, with PISO requiring the minimum and SIMPLEST the maximum number of iterations. The SIMPLEX algorithm (Figure 3*g*) is seen to require a slightly lower number of iterations on the finest grid as compared to SIMPLE (Figure 3*c*), SIMPLEC (Figure 3*d*), and SIMPLEM (Figure 3*e*). The smoothest convergence paths are for SIMPLEX and SIMPLEC. As depicted in Figures 3*f* and 3*h*, the performance of SIMPLEST/PRIME is poor as compared to other algorithms for the same reasons stated above.

Problem 3: Dilute bubbly flow. For the same configuration displayed in Figure 1*b*, the continuous phase is considered to be water and the disperse phase to be air. The resulting flow is denoted in the literature by bubbly flow. With the exception of $\rho^{(d)}/\rho^{(c)} = 10^{-3}$ and at inlet $r^{(d)}_{\text{inlet}} = 0.1$, other physical properties and inlet conditions are the same as those considered earlier. This is a strongly coupled problem and represents a good test for the numerical procedure and performance of the algorithms. The correct physical solution is that the bubble and continuous-phase velocities both reach the equilibrium velocity of 4.6 m/s [Eq. (25)] in a distance too small to be correctly resolved by any of the grid networks used. Results for this case are presented in Figure 4. Axial velocity distribution for both water and air are displayed in Figure 4*a*. As expected, both phases reach the equilibrium velocity of 4.6 m/s over a very short distance from the inlet section and remain constant afterward. The relative convergence characteristics of the various algorithms remain the same. However, all algorithms require a larger number of iterations as compared to the dilute gas–solid flow case, due to the stronger coupling between the phases. Consistently, the PISO (Figure 4*b*) and SIMPLEST/PRIME (Figure 4*f*) algorithms need the lowest and highest number of iterations, respectively. As in the previous two cases, the convergence attributes of SIMPLE (Figure 4*c*), SIMPLEC (Figure 4*d*), and SIMPLEX (Figure 4*g*) are very similar, and those of SIMPLEM (Figure 4*e*) are close to them. The large difference in performance between SIMPLEST/PRIME and the remaining algorithms is clearly demonstrated in Figure 4*h*.

Problem 4: Dense bubbly flow. The only difference between this case and the previous one is in the concentration of bubbles, which is set to $r^{(d)}_{\text{inlet}} = 0.5$. With such a high value of void fraction, bubble coalescence may occur. However, this is not accounted for here. The analytical solution is the same as in the previous case, with the equilibrium velocity, as computed from Eq. (25), being 3 m/s. As depicted in Figure 5*a*, the equilibrium velocity obtained numerically is exact. Moreover, the performance of the various algorithms (Figures 5*b*–5*h*) vary relatively in a manner similar to what was previously discussed, and it is deemed redundant to be repeated.

CPU time: Horizontal particle transport. The normalized CPU efforts required by the various algorithms over all grids are depicted in Figure 6. The charts clearly show that the CPU time increases with increasing grid density. For

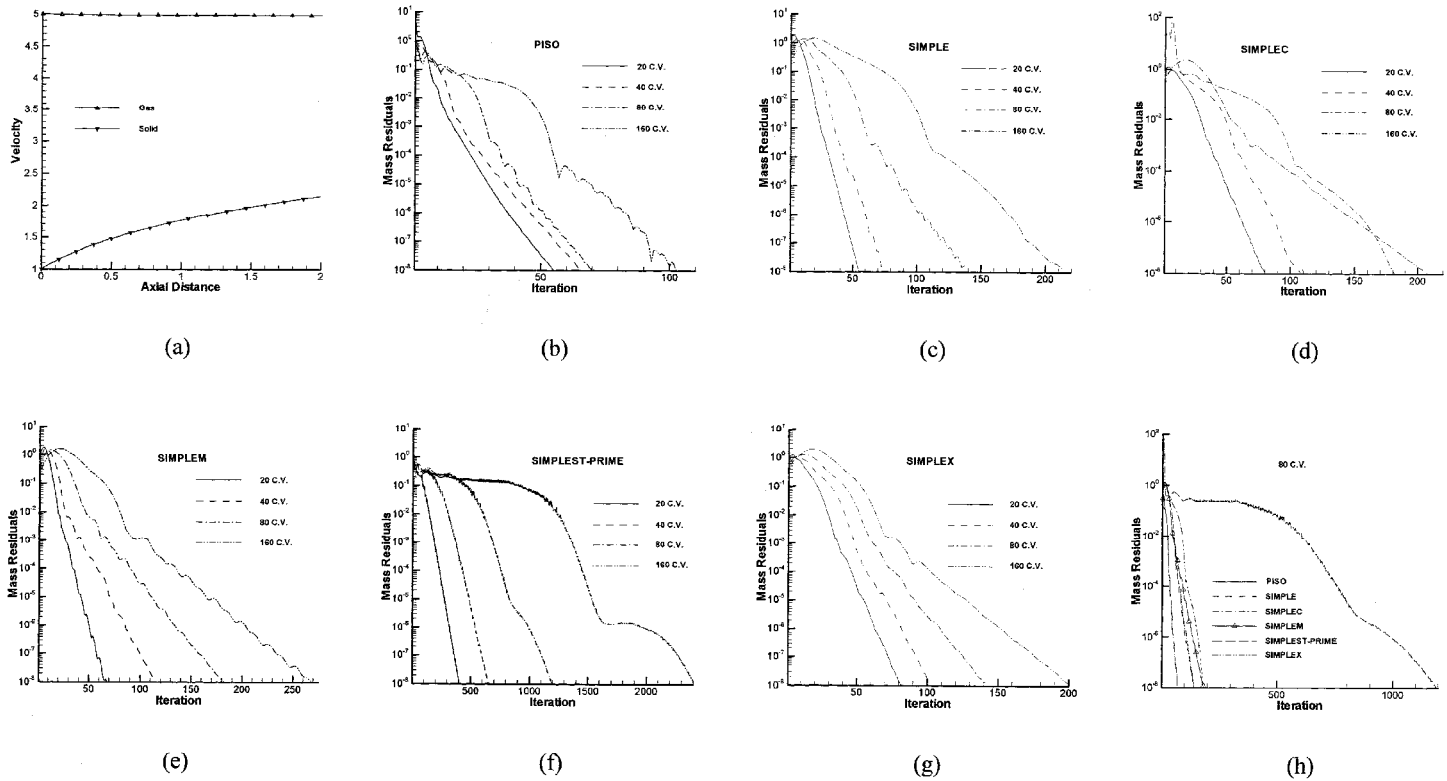


Figure 3. (*a*) Gas and particle velocity distributions. (*b*)–(*g*) Convergence histories on the different grid systems. (*h*) Convergence histories of the various algorithms on the 80-C.V. grid for the horizontal dense gas–solid flow problem.

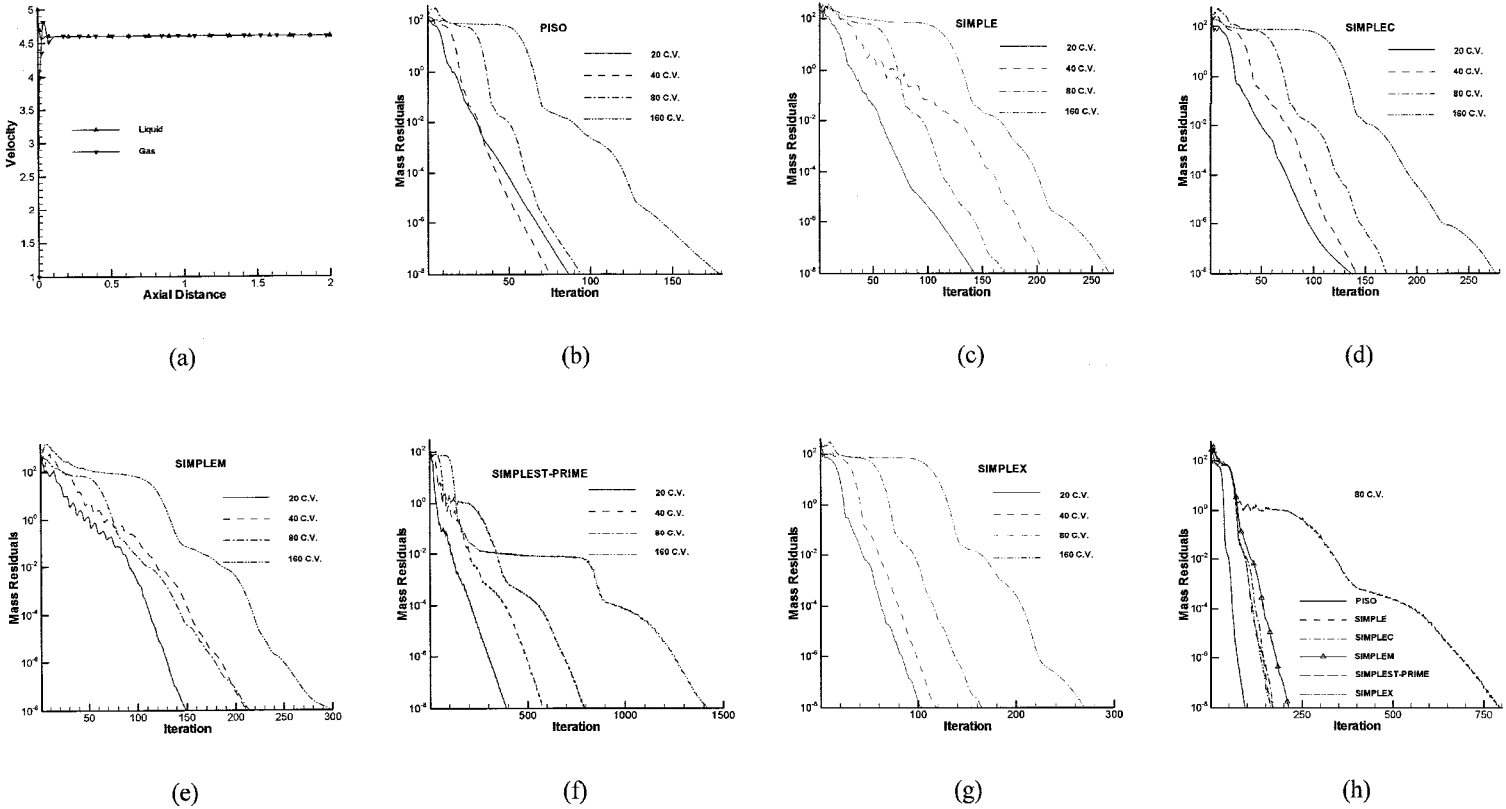


Figure 4. (*a*) Liquid and gas velocity distributions. (*b*)–(*g*) Convergence histories on the different grid systems. (*h*) Convergence histories of the various algorithms on the 80-C.V. grid for the horizontal dilute bubbly flow problem.

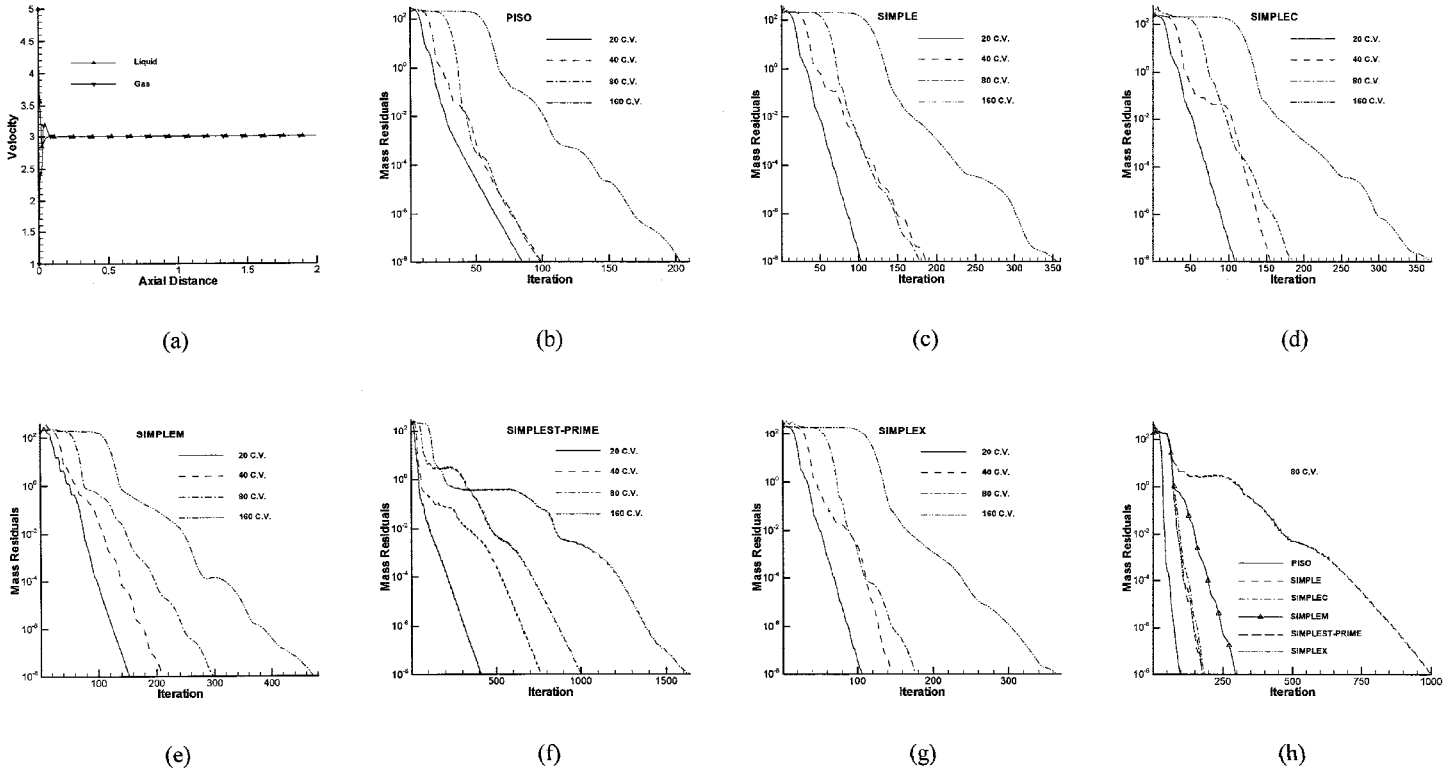


Figure 5. (*a*) Liquid and gas velocity distributions. (*b*)–(*g*) Convergence histories on the different grid systems. (*h*) Convergence histories on the 80-C.V. grid for the horizontal dense bubbly flow problem.

the dilute gas–solid problem (Figure 6*a*), it is hard to see any noticeable difference in the CPU times for SIMPLE, SIMPLEC, SIMPLEX, and PISO. The SIMPLEM algorithm requires slightly higher computational effort as compared to SIMPLE. The worst performance is for SIMPLEST, which degenerates to PRIME in the absence of diffusion and results in a fully explicit solution scheme. For the dense gas–solid flow (Figure 6*b*), PISO requires the lowest computational effort (10% less than SIMPLE on the finest mesh). Moreover, the performance of SIMPLE, SIMPLEC, and SIMPLEX is more or less identical, while that of SIMPLEM is of lower quality, necessitating increasingly higher computational effort with denser meshes, and requiring about 40% more effort than SIMPLE on the fine grids (80 and 160 control volumes). The computational effort needed by SIMPLEST/PRIME is, however, the most extensive, and is nearly 623% the one needed by SIMPLE on the finest mesh.

The normalized CPU time of SIMPLEST/PRIME for the bubbly flow problems (Figures 6*c* and 6*d*) is lower than in the previous two problems due to a higher rate of increase in the time needed by other algorithms (the computational time of all algorithms has increased). The relative performance of the various algorithms is nearly as described earlier, with the time required by PISO, SIMPLE, SIMPLEC, and SIMPLEX being on average the same. The SIMPLEST/PRIME algorithm, however, requires nearly threefold the time needed by SIMPLE, which represents a noticeable improvement.

Vertical Particle Transport

Here, the flow is in the vertical direction (Figure 1*b*), the gravitational acceleration is assigned the constant value of $g = 10\,\mathrm{m/s^2}$, and the flow field is extended over a length of $L = 20\,\mathrm{m}$. For this situation, the velocities of the two phases do not reach an equilibrium value. Rather, the disperse phase equilibrates to a finite settling velocity relative to the continuous phase, at which the gravitational force balances the drag force [24]. As for the horizontal transport problems, the interparticle/bubble forces are neglected. However, unlike the previous situation, diffusion in the continuous phase is retained. Moreover, the interphase drag force is calculated using Eqs. (22)–(24) and the drag coefficient, C_D, is considered to be particle Reynolds number dependent and is calculated as

$$C_D = \frac{24}{\mathrm{Re}_p} + 0.44 \qquad \mathrm{Re}_p = \frac{2 r_p V_{\mathrm{slip}}}{\vartheta^{(c)}} \tag{27}$$

Since diffusion in the continuous phase is not neglected, MCBA-SIMPLEST and MCBA-PRIME are expected to behave differently.

Problem 5: Dilute gas–solid flow. The material properties and boundary conditions considered for this case are given by

$$\frac{\rho^{(d)}}{\rho^{(c)}} = 1,000 \qquad \vartheta^{(c)} = 10^{-5} \qquad r_p = 1\,\mathrm{mm} \tag{28}$$

$$V^{(c)}_{\mathrm{inlet}} = 100\,\mathrm{m/s} \qquad V^{(d)}_{\mathrm{inlet}} = 10\,\mathrm{m/s} \qquad r^{(d)}_{\mathrm{inlet}} = 10^{-6} \tag{29}$$

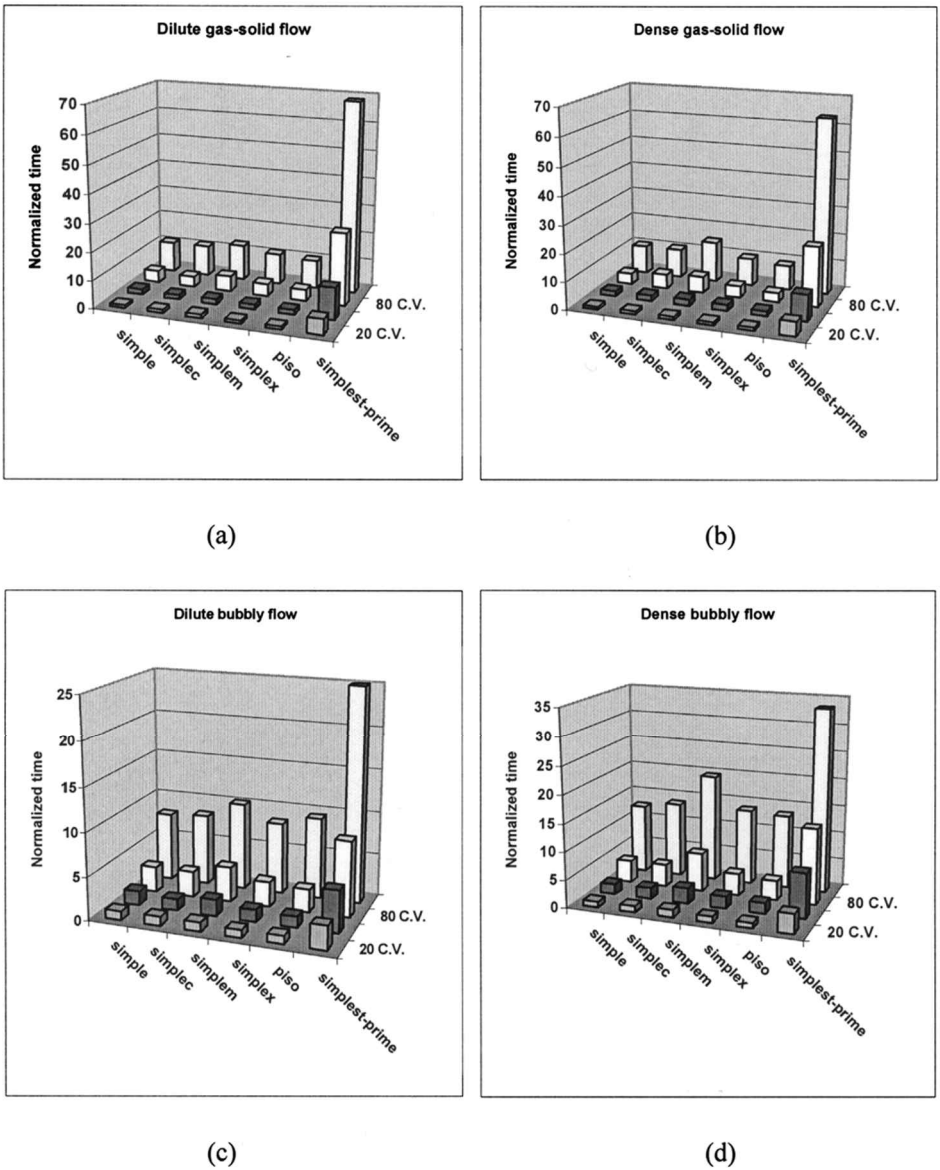


Figure 6. Normalized CPU times for the horizontal (*a*) dilute gas–solid, (*b*) dense gas–solid, (*c*) dilute bubbly, and (*d*) dense bubbly flow problem.

The large velocity boundary condition is used to ensure that the solid phase does not exit the inlet. The predicted air and particle velocity distributions depicted in Figure 7*a* are in excellent agreement with similar predictions reported in [25]. As shown in Figures 7*b*–7*h*, the decrease in the mass residuals is highly nonmonotonic, showing a cyclic decaying behavior. The PISO algorithm (Figure 7*b*) seems to be the least affected, its convergence path showing little cycling as compared to other algorithms, and requiring the least number of iterations over all grids. It should be

mentioned here that decreasing the underrelaxation factors could have smoothed this cycling behavior out. This, however, would have been accomplished at the expense of increasing the computational time. Even though the convergence is not monotonic, the number of iterations needed to converge the solution to the desired level is very close to that needed in the similar horizontal transport case. The convergence of the SIMPLEC algorithm over the dense grid system (Figure 7*d*) shows cycles of large amplitudes due to the fact that pressure is not underrelaxed. The performance of SIMPLEST (Figure 7*f*) and PRIME (Figure 7*g*) is very close, with SIMPLEST requiring slightly fewer iterations due to the small implicitness introduced by the diffusion of the continuous phase. However, both require on the finest mesh almost 1,300% the number of iterations needed by SIMPLE. The number of outer iterations needed by SIMPLEX, SIMPLEC, and SIMPLEM is very close to that of SIMPLE, with SIMPLEM requiring the highest number of iterations.

Problem 6: Dense gas–solid flow. The material properties and boundary conditions are similar to the previous case with the exception of the particles' volume fraction, which is set to $r_{\text{inlet}}^{(d)} = 10^{-2}$. Predicted air and particle velocity profiles are displayed in Figure 8*a*, while mass residuals are presented in Figures 8*b*–8*h*. The convergence paths are smoother in comparison with the dilute case due to the higher volume fraction of the disperse phase, which makes the computations less sensitive to the intermediate level of convergence. However, more iterations are needed in comparison with the dilute case due to the higher mass-loading ratio. Otherwise, the convergence behavior is similar to the previous cases, with SIMPLEST (Figure 8*f*) and PRIME (Figure 8*g*) requiring the highest number of iterations (SIMPLEST needs slightly fewer iterations, for the reason stated above). The number of iterations needed by PISO, SIMPLEC, and SIMPLEX (compare Figures 8*b*, 8*d*, and 8*h*) is very close. The SIMPLEM algorithm (Figure 8*e*) requires a slightly higher number of iterations than SIMPLE (Figure 8*c*).

Problem 7: Dilute bubbly flow. In this problem, the continuous phase is water and the disperse phase is air. With the exception of $\rho^{(d)}/\rho^{(c)}$ set to 10^{-3}, $V_{\text{inlet}}^{(c)}$ and $V_{\text{inlet}}^{(d)}$ to 1, and $r_{\text{inlet}}^{(d)}$ to 0.1 at the inlet, other physical properties and inlet conditions are the same as those considered earlier. This is a very difficult problem to get convergence to unless the proper underrelaxation is used. It was possible to get feasible solutions (i.e., with reasonable computational time) when underrelaxing by inertia (i.e., through the use of false time steps). For the results presented in Figure 9, a time step (Δt) of value 10^{-4} s is used for the velocity field of the dispersed gas phase, $\Delta t = 1$ s for the volume fractions, and $\Delta t = 0.01$ s for the velocity field of the liquid phase (this last value is employed with all algorithms except PISO and PRIME, for which a value of 0.1 s is used). Also, to accelerate convergence, it is found advantageous not to underrelax the pressure field. As expected, the correct velocity fields are predicted (Figure 9*a*). All algorithms require more iterations compared to the previous problems. Moreover, convergence flattens after a certain level because of the large underrelaxation used for the bubble-phase momentum equations. The performance of PISO (Figure 9*b*) is totally unexpected, requiring more iterations than SIMPLE (Figure 9*c*), SIMPLEC (Figure 9*d*), SIMPLEM (Figure 9*e*), and SIMPLEX (Figure 9*h*). The

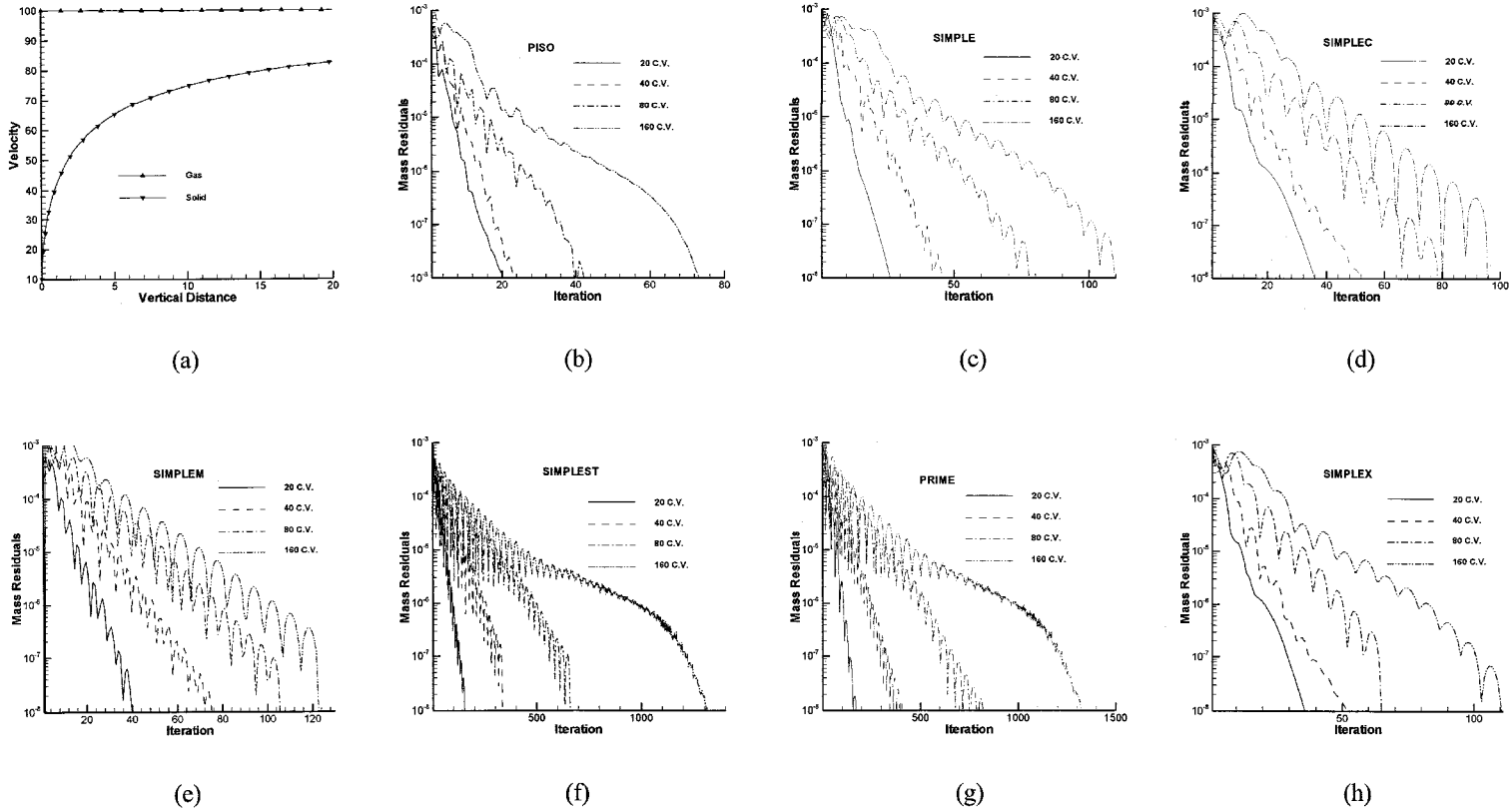


Figure 7. (*a*) Gas and particle velocity distributions. (*b*)–(*h*) Convergence histories on the different grid systems for the vertical dilute gas–solid flow problem.

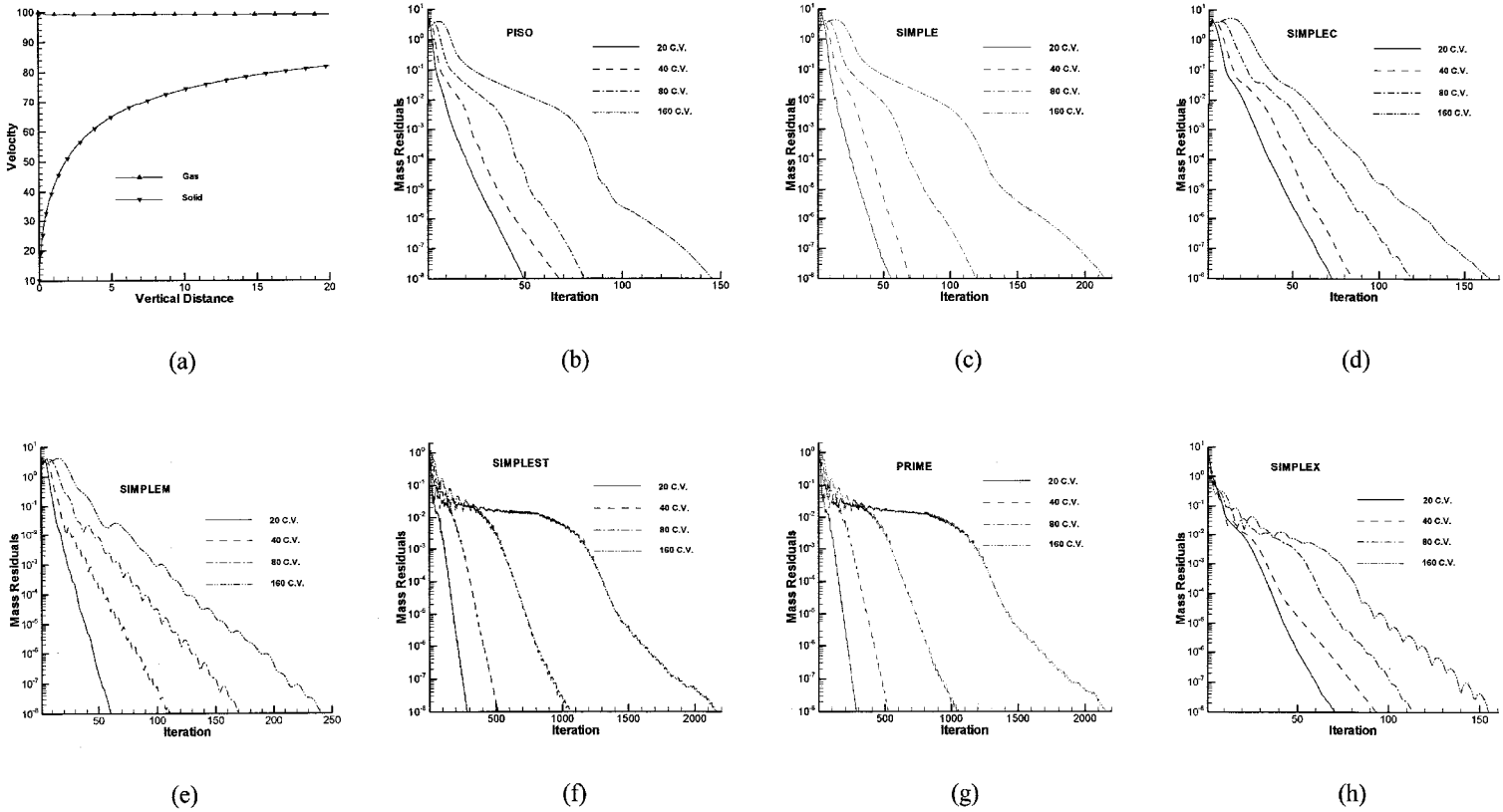


Figure 8. (*a*) Gas and particle velocity distributions. (*b*)–(*h*) Convergence histories on the different grid systems for the vertical dense gas–solid flow problem.

decrease in the rate of convergence as the residuals are decreased is very clear, with SIMPLE, SIMPLEC, and SIMPLEX, which show a sudden change in their convergence slopes. The convergence histories of these algorithms are nearly identical. The explicitness introduced in PISO (Figure 9*b*), SIMPLEST (Figure 9*f*), and PRIME (Figure 9*g*) causes an increase in the number of iterations and a highly nonmonotonic convergence behavior and seems to be undesirable in laminar bubbly flows. The rate of convergence of the SIMPLEM algorithm (Figure 9*e*) is nearly constant and does not show any change in slope as the computations progress.

Problem 8: Dense bubbly flow. With the exception of setting $r_{\text{inlet}}^{(d)}$ to 0.5, the physical situation, material properties, and boundary conditions are the same as in the previous problem. Results for the problem are presented in Figure 10. The predicted liquid and gas velocity distributions, which are in excellent accord with published data, are depicted in Figure 10*a*. The trend of convergence (Figures 10*b*–10*h*) is very similar to the dilute case with the following differences. With PISO and PRIME, in order to drive residuals to the desired level of convergence, the SMART scheme had to be blended with the UPWIND scheme. The percentage increased from 15% on the coarsest grid to 50% on the finest grid. Moreover, mass residuals reached the desired convergence level long before the momentum residuals. Thus, the numbers of iterations needed by PISO and PRIME are much greater than the ones presented in Figures 10*b* and 10*g*. In addition, to stabilize SIMPLEST on the finest grid, the SMART scheme had to be blended with 20% of the upwind value. The difficulties faced by PISO, PRIME, and SIMPLEST are attributed to the additional explicitness introduced as a result of using a HR scheme implemented via a deferred-correction strategy [26]. This deferred-correction technique has less influence on the performance of the remaining algorithms, due to their higher implicitness.

CPU time: Vertical particle transport. The normalized CPU times for the vertical particle transport problems are displayed in Figure 11. As in the horizontal case, the CPU time increases with increasing grid density. For the gas–solid flow problems (Figures 11*a* and 11*b*), the relative performance of the various algorithms is similar for both dilute and dense concentration of particles. For the dilute case (Figure 11*a*), the efficiency of SIMPLEST is slightly better than PRIME; both, however, are about five times more expensive than all other algorithms, whose performance is very comparable. For the dense case (Figure 11*b*), SIMPLEX requires the least computational time, followed by SIMPLEC and SIMPLE. Again, the computational effort needed by SIMPLEST and PRIME is much higher (about 700% the time needed by SIMPLEX on the finest grid). For the vertical bubbly flows, a noticeable change in the normalized time chart (Figures 11*c* and 11*d*) is depicted, with the performance of SIMPLEST showing a good improvement and that of PISO deteriorating. For the dilute bubbly case (Figure 11*c*), the CPU times needed by SIMPLE, SIMPLEC, and SIMPLEX are consistently very close. For a given grid, however, the largest CPU time consumed is only double the lowest CPU time needed. For the dense bubbly case (Figure 11*d*), even with the blending strategy, PISO and PRIME still need high computational time to decrease the residuals to the desired level. The

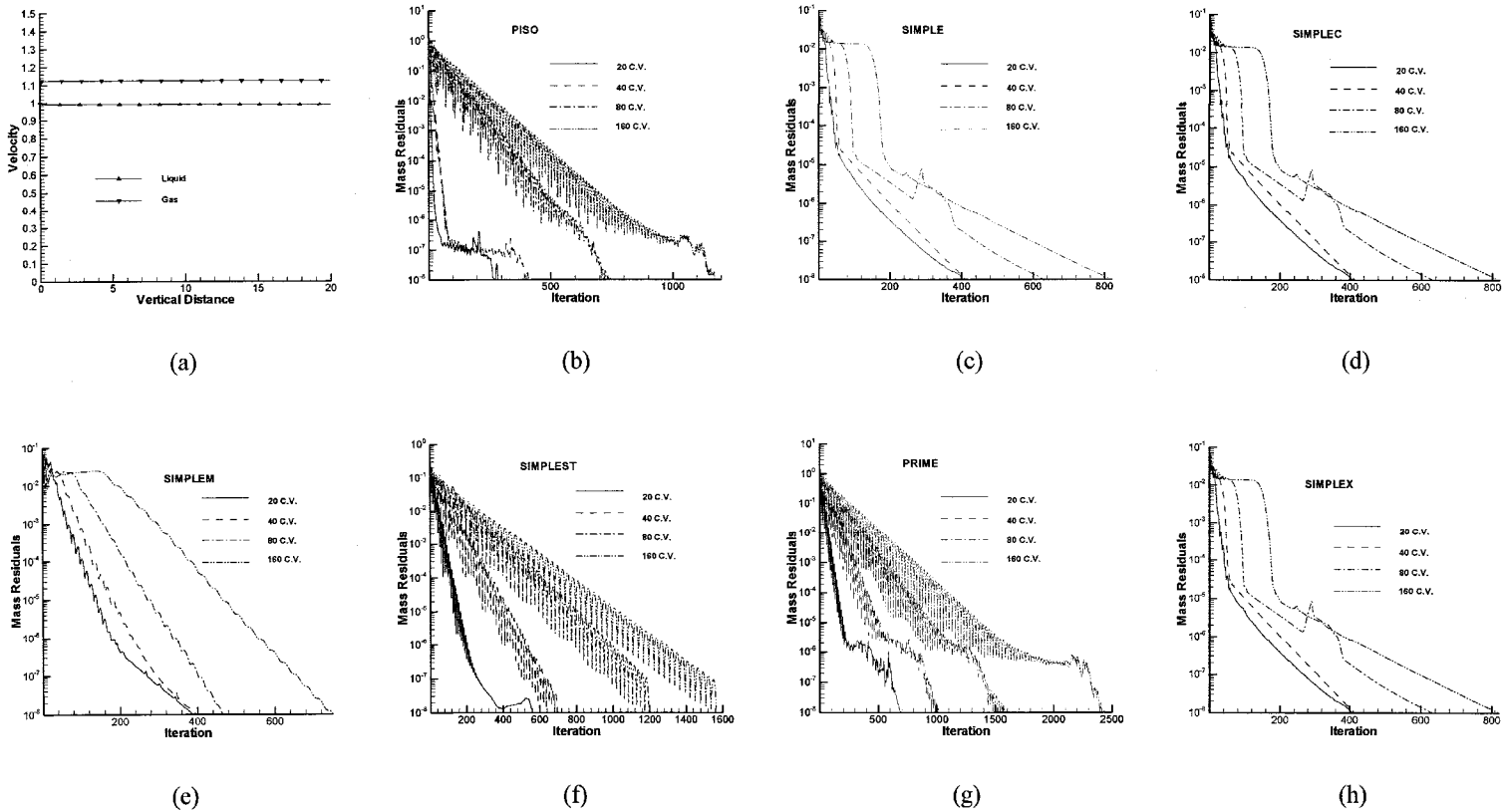


Figure 9. (*a*) Gas and particle velocity distributions. (*b*)–(*h*) Convergence histories on the different grid systems for the vertical dilute bubbly flow problem.

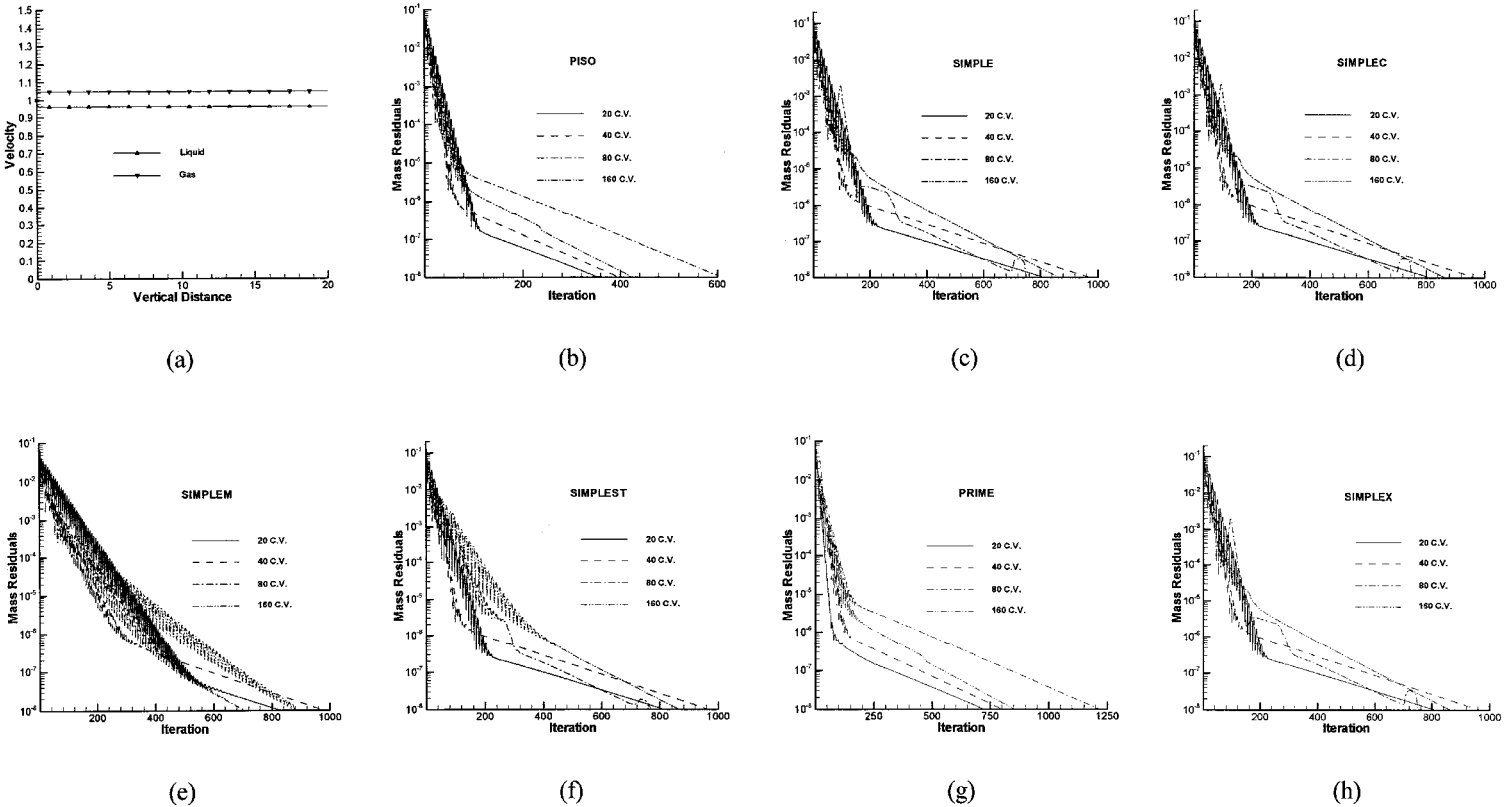


Figure 10. (*a*) Gas and particle velocity distributions. (*b*)–(*h*) Convergence histories on the different grid systems for the vertical dense bubbly flow problem.

SIMPLEST algorithm requires the least computational effort among the three algorithms on the finest grid when blended with the upwind scheme. The CPU times consumed by SIMPLE, SIMPLEC, and SIMPLEX are consistently very close to each other.

By comparing the behavior of the various algorithms in all problems, it is clear that the performance of SIMPLE, SIMPLEC, and SIMPLEX is consistent and requires, on average, the least computational effort. Even though in some cases PISO

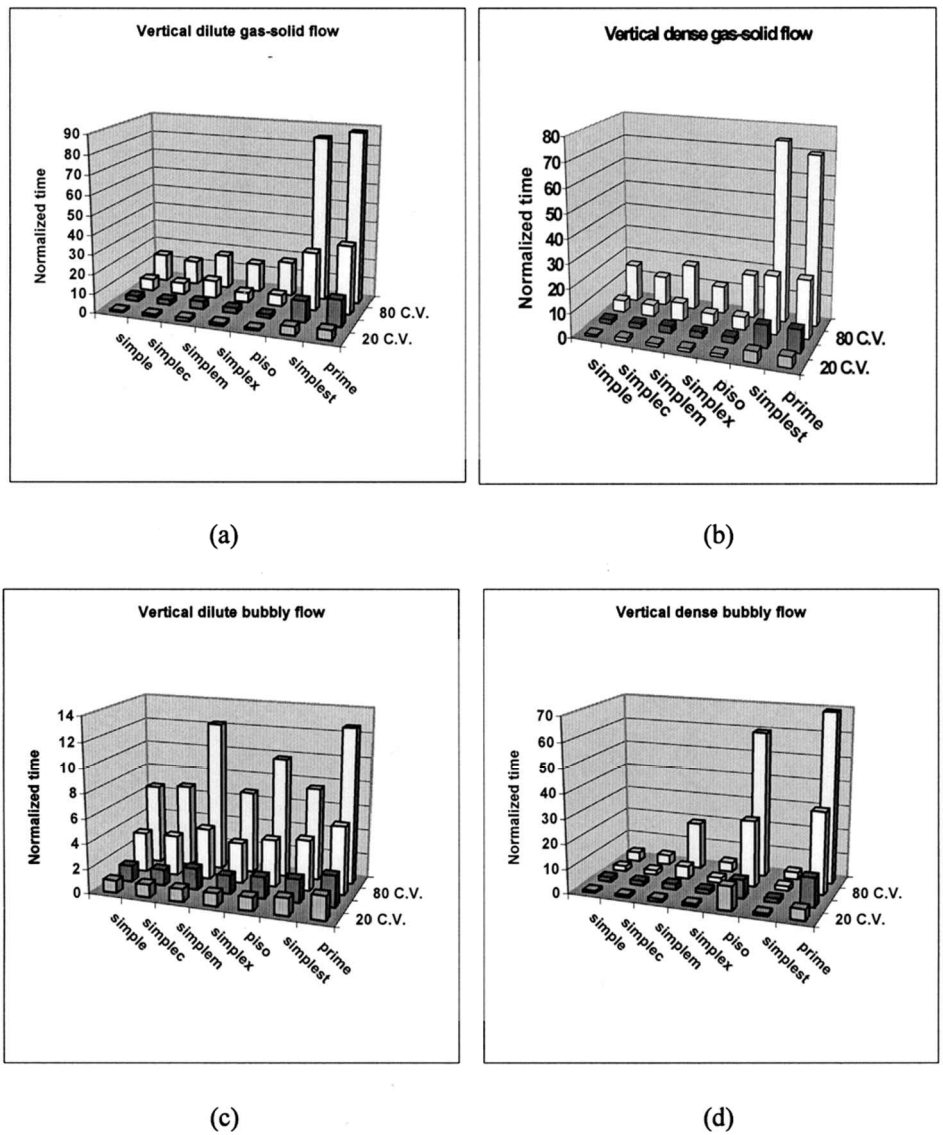


Figure 11. Normalized CPU times for the vertical (*a*) dilute gas–solid, (*b*) dense gas–solid, (*c*) dilute bubbly, and (*d*) dense bubbly flow problem.

consumes less CPU time, its performance for upward bubbly flows is not satisfactory. The performance of SIMPLEM is consistent, but demands more CPU time than SIMPLE, SIMPLEC, and SIMPLEX. The performance of the SIMPLEST algorithm was comparable to SIMPLE for upward bubbly flows only. The PRIME algorithm is the most expensive to use on all grids and for all physical situations presented here. Most important, however, is the fact that all these algorithms can be used to predict multiphase (in this case two-phase) flows.

CLOSING REMARKS

Seven MCBA algorithms for the simulation of incompressible multiphase flows were implemented, tested, and their relative performance assessed by solving a variety of one-dimensional two-phase flow problems. For each test problem, solutions were generated on a number of grid systems. With all algorithms, the normalization procedure was essential to improve the convergence behavior. Results obtained demonstrated that all MCBA multiphase algorithms are capable of dealing with a wide variety of incompressible multiphase flow problems. The convergence history plots and CPU times presented indicated similar performances for SIMPLE, SIMPLEC, and SIMPLEX. The PISO, SIMPLEM, and SIMPLEST algorithms were in general more expensive than SIMPLE. In general, the PRIME algorithm was the most expensive to use.

REFERENCES

1. P. H. Gaskell and A. K. C. Lau, Curvature Compensated Convective Transport: SMART, A New Boundedness Preserving Transport Algorithm, *Int. J. Numer. Meth. Fluids*, vol. 8, pp. 617–641, 1988.
2. B. P. Leonard, Locally Modified Quick Scheme for Highly Convective 2-D and 3-D Flows, in C. Taylor and K. Morgan (eds.), *Numerical Methods in Laminar and Turbulent Flows*, vol. 15, pp. 35–47, Pineridge Press, Swansea, UK, 1987.
3. M. S. Darwish and F. Moukalled, Normalized Variable and Space Formulation Methodology for High-Resolution Schemes, *Numer. Heat Transfer B*, vol. 26, pp. 79–96, 1994.
4. S. V. Patankar and D. B. Spalding, A Calculation Procedure for Heat, Mass and Momentum Transfer in Three Dimensional Parabolic Flows, *Int. J. Heat Mass Transfer*, vol. 15, pp. 1787–1806, 1972.
5. R. I. Issa, Solution of the Implicit Discretized Fluid Flow Equations by Operator Splitting, Mechanical Engineering Department Rep. FS/82/15, Imperial College, London, UK, 1982.
6. J. P. Van Doormaal and G. D. Raithby, Enhancement of the SIMPLE Method for Predicting Incompressible Fluid Flows, *Numer. Heat Transfer*, vol. 7, pp. 147–163, 1984.
7. J. P. Van Doormaal and G. D. Raithby, An Evaluation of the Segregated Approach for Predicting Incompressible Fluid Flows, ASME Paper 85-HT-9, National Heat Transfer Conf., Denver, CO, 4–7 August 1985.
8. F. Moukalled and M. Darwish, A Unified Formulation of the Segregated Class of Algorithms for Fluid Flow at All Speeds, *Numer. Heat Transfer B*, vol. 40, pp. 99–137, 2001.
9. D. Kershaw, The Incomplete Cholesky-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations, *J. Comput. Phys.*, vol. 26, pp. 43–65, 1978.

10. H. L. Stone, Iterative Solution of Implicit Approximations of Multidimensional Partial Differential Equations, *SIAM J. Numer. Anal.*, vol. 5, no. 3, pp. 530–558, 1968.
11. A. Brandt, Multi-Level Adaptive Solutions to Boundary-Value Problems, *Math. Comput.*, vol. 31, pp. 333–390, 1977.
12. W. Shyy and M. H. Chen, Pressure-Based Multigrid Algorithm for Flow at All Speeds, *AIAA J.*, vol. 30, no. 11, pp. 2660–2669, 1992.
13. D. B. Spalding, Mathematical Modelling of Fluid Mechanics, Heat Transfer and Mass Transfer Processes, Mechanical Engineering Department Rep. HTS/80/1, Imperial College of Science, Technology and Medicine, London, UK, 1980.
14. S. Acharya and F. Moukalled, Improvements to Incompressible Flow Calculation on a Non-Staggered Curvilinear Grid, *Numer. Heat Transfer B*, vol. 15, pp. 131–152, 1989.
15. C. R. Maliska and G. D. Raithby, Calculating 3-D Fluid Flows Using Non-Orthogonal Grid, *Proc. Third Int. Conf. on Numerical Methods in Laminar and Turbulent Flows*, Seattle, WA, pp. 656–666, 1983.
16. D. B. Spalding, The Calculation of Free-Convection Phenomena in Gas-Liquid Mixtures, Mechanical Engineering Department Rep. HTS/76/11, Imperial College, London, UK, 1976.
17. D. B. Spalding, Numerical Computation of Multi-Phase Fluid Flow and Heat Transfer, in C. Taylor and K. Morgan (eds.), *Recent Advances in Numerical Methods in Fluid*, vol. 1, pp. 139–167, Pineridge Press, NY, 1980.
18. D. B. Spalding, A General Purpose Computer Program for Multi-Dimensional, One and Two Phase Flow, Mechanical Engineering Department Rep. HTS/81/1, Imperial College, London, UK, 1981.
19. W. W. Rivard and M. D. Torrey, KFIX: A Program for Transient Two Dimensional Two Fluid Flow, Rep. LA-NUREG-6623, Los Alamos Nuclear Regulatory Commission, 1978.
20. A. A. Amsden and F. H. Harlow, KACHINA: An Eulerian Computer Program for Multifield Flows, Rep. LA-NUREG-5680, Los Alamos Nuclear Regulatory Commission, 1975.
21. A. A. Amsden and F. H. Harlow, KTIFA Two-Fluid Computer Program for Down Comer Flow Dynamics, Rep. LA-NUREG-6994, Los Alamos Nuclear Regulatory Commission, 1977.
22. M. Darwish, F. Moukalled, and B. Sekar, A Unified Formulation of the Segregated Class of Algorithms for Multi-Phase Flow at All Speeds, *Numer. Heat Transfer B*, vol. 40, no. 2, pp. 99–137, 2001.
23. P. J. Zwart, G. D. Raithby, and M. J. Raw, An Integrated Space-Time Finite-Volume Method for Moving-Boundary Problems, *Numer. Heat Transfer B*, vol. 34, pp. 257–270, 1998.
24. S. A. Morsi and A. J. Alexander, An Investigation of Particle Trajectories in Two-Phase Flow System, J. Fluid Mech., vol. 55, part 2, pp. 193–208, 1972.
25. A. H. A. Baghdadi, Numerical Modelling of Two-Phase Flow with Inter-Phase Slip, Ph.D. thesis, Imperial College, University of London, UK, 1979.
26. S. G. Rubin and P. K. Khosla, Polynomial Interpolation Method for Viscous Flow Calculations, *J. Comput. Phys.*, vol. 27, pp. 153–168, 1982.