View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Institute Of Mechanics,Chinese Academy of Sciences

International Journal of Computational Fluid Dynamics
Vol. 22, No. 6, July 2008, 405–410

Large scale numerical simulations for multi-phase fluid dynamics with moving interfaces

S. Yamashita[a], C.G. Chen[a], K. Takahashi[b] and F. Xiao[a,c,*]

[a]Department of Energy Sciences, Tokyo Institute of Technology, Yokohama, Japan; [b]Earth Simulator Center, JAMSTEC, Yokohama, Japan; [c]DES/LHD, Institute of Mechanics, Chinese Academy of Sciences, Beijing, People's Republic of China

(*Received 14 August 2007; final version received 28 March 2008*)

This short communication presents our recent studies to implement numerical simulations for multi-phase flows on top-ranked supercomputer systems with distributed memory architecture. The numerical model is designed so as to make full use of the capacity of the hardware. Satisfactory scalability in terms of both the parallel speed-up rate and the size of the problem has been obtained on two high rank systems with massively parallel processors, the Earth Simulator (Earth simulator research center, Yokohama Kanagawa, Japan) and the TSUBAME (Tokyo Institute of Technology, Tokyo, Japan) supercomputers.

Keywords: large-scale computing; direct numerical simulation; multi-phase flow; moving interfaces; volume of fluid

Downloaded by [Institute of Mechanics, CAS] at 21:07 16 October 2011

Multi-phase flows, such as air/water surface, raindrops, bubbles and sprays, are commonly observed in nature. Including at least two fluids of different physical properties that interact with each other, the multi-phase flows exhibit more complex behaviour compared to the homogeneous fluid.

Owing to the progress in both the numerics and the computer hardware, direct numerical simulation (DNS) is getting widely accepted as a useful and increasingly mature tool to investigate the dynamics of multi-phase flows where the interface separating different fluid, like air and water, needs to be explicitly resolved. To this end, a practical code for the DNS of multi-phase fluid dynamics should be able to effectively utilise the parallel computers with distributed memory architectures. Some practices in large scale parallel simulations of multi-phase fluid dynamics with moving interfaces are found in Guler *et al.* (1999), Xiao (2001), Aliabadi *et al.* (2002) and Sussman (2005).

We have recently developed a numerical model for multi-fluid dynamics, based on a so-called one-fluid continuum model (Tryggvason *et al.* 2006). The numerical model has been built so as to efficiently make use of modern supercomputers with massively parallel architectures of distributed memory. Satisfactory convergence rate and scalability for large scale simulations have been achieved on the Earth simulator (JAMSTEC 2006) and TSUBAME (Endo *et al.* 2006), two of the top ranked supercomputers in Japan.

For incompressible fluid, the governing equations of the one-fluid numerical model for two phase flows involving free interface are written as

$$\nabla \cdot \mathbf{U} = 0, \tag{1}$$

$$\partial_t u_\alpha + \nabla \cdot (u_\alpha \mathbf{U}) = -\frac{1}{\rho}\partial_\alpha p + \frac{1}{\rho}\sum_{\beta=x,y,z}\partial_\beta \tau_{\alpha\beta} + f_\alpha, \tag{2}$$

$$\partial_t c + \mathbf{U}\cdot\nabla c = 0, \tag{3}$$

where $\mathbf{U} = (u_x,\ u_y,\ u_z)$ is the velocity vector in three dimensions, ρ is the density and p is the pressure. $\tau_{\alpha\beta} = \mu\, \partial_\beta u_\alpha$ denotes the components of the viscous shear stress tensor and μ the viscosity coefficient. f_α is the body force in α-direction and $\alpha = x, y, z$ denotes the direction.

Equation (3) is the transport equation of the volume of fluid (VOF) function c which represents the volume fraction of a specified fluid, say the water, in each control volume element. In a one-fluid incompressible model, with the VOF function explicitly computed at each step, the free interface can be uniquely identified, and the physical field of density and other material properties can be straightforwardly specified.

As discussed in Tryggvason *et al.* (2006), the one-fluid model, in which one uses only a single set of governing equations for both the dispersed phase and the carrier (or the continuum phase), is an efficient, and adequate in most cases, physical framework for the DNS of multi-phase flows.

The fluid dynamics is solved by combining the volume/surface integrated average based multi-moment method (VSIAM3) (Xiao *et al.* 2005, 2006) and a pressure-based projection method. Being an efficient and

*Corresponding author. Email: xiao@es.titech.ac.jp

ISSN 1061-8562 print/ISSN 1029-0257 online
© 2008 Taylor & Francis
DOI: 10.1080/10618560802199691
http://www.informaworld.com

Downloaded by [Institute of Mechanics, CAS] at 21:07 16 October 2011

practical constrained interpolation profile (Yabe *et al.* 2001) type finite volume formulation, the VSIAM3 uses two integrated moments, namely volume integrated average (VIA) and surface integrated average (SIA), and is a general framework developed for computational fluid dynamics. Details of the method are found in Xiao *et al.* (2005, 2006).

A fractional solution procedure with a pressure projection is adopted. In the multi-moment context, the coupling between pressure and velocity in the projection step is computed through the VIA of the pressure and the SIA of the normal components of velocity. Considering the VIA and the SIA are staggeringly located, the pressure and the velocity are well coupled like those in the staggered grid. Meanwhile, we should also remark that the VIA of the velocity collocates with the VIA of pressure, which makes difference from the conventional staggered or collocated grid.

The discretisation of the pressure Poisson yields a standard seven-point sparse symmetrical matrix system. As discussed later, we solve the discretised Poisson equations by a multigrid preconditioned conjugate gradient (CG) method in present study.

The surface tension force is computed by the continuum surface force formulation (Brackbill *et al.* 1992). A level set function is generated at each step from the VOF function to obtain the geometrical information, such as normal vector and curvature, of the interface.

In a one-fluid model for multi-phase flows, different fluids are separated by moving the interface which is explicitly computed at every time step. We have recently developed an algebraic interface capturing method, a tangent of hyperbola for interface capturing (THINC) scheme (Xiao *et al.* 2005; Yokoi 2007). This method exactly conserves the mass of the transported quantity, moreover, it is computationally efficient and parallel oriented because the geometrical reconstruction such as that involved in conventional VOF method is not required.

As discussed above, the numerical methods used in this study are quite computationally efficient and well-suited for parallel implementations. All the computations, except that for the pressure Poisson equation, are computed in an explicit manner.

Our target machines for parallelisation are TSUBAME (Endo *et al.* 2006) and Earth simulator (JAMSTEC 2006) supercomputers, separately installed in Tokyo Institute of Technology, Tokyo, Japan and the Earth simulator research center, Yokohama Kanagawa, Japan. Both of the supercomputers consist of a large number of processor elements (or nodes) with distributed memory.

TSUBAME is a ‘supercomputing grid’ system installed in Tokyo Institute of Technology in April 2006. It clusters 655 Sun Fire™ X4600 servers with 10,480 AMD Opteron™ processor cores connected by Voltaire’s InfiniBand network. TSUBAME system has 85 Teraflops peak speed and 21 Terabytes of memory. Each Sun Fire node server possesses eight dual-core 2.4 GHz Opteron CPU processors, and 16 CPU core share 32 GB memory. The inter-node networks are featured by 10 Gbps InfiniBand host channel adapter, 288-port Voltaire ISR9288 switches.

Compared to the TSUBAME supercomputer, which is a massively parallel grid computing system with scalar processors, the Earth simulator is a highly parallel vector supercomputer system of distributed memory. The Earth simulator consists of 640 processor nodes (PNs) connected by 640 × 640 single-stage crossbar switches. Each PN is a system with a shared memory, consisting of eight vector-type arithmetic processors (APs), one 16 GB main memory system, one remote access control unit (RCU), and one I/O processor. The peak performance of each AP is 8 Gflops. The Earth simulator as a whole thus consists of 5120 APs with 10 TB of main memory and has the theoretical peak performance of 40 Tflops. The RCU is directly connected to the crossbar switches and controls inter-node data communications at 12.3 GB/s bi-directional transfer rate for both sending and receiving data. Thus the total bandwidth of inter-node network is about 8 TB/s. Several data-transfer modes, including access to three-dimensional (3D) sub-arrays and indirect access modes, are realised in hardware. In an operation that involves access to the data of a sub-array, the data is moved from one PN to another in a single hardware operation, which consumes relatively less processing time.

The code has been parallelised using domain decomposition. The computational domain can be flexibly divided by one to three dimensional partitions to get well-balanced computational load for each processor and minimise the data communications among the processors of distributed memory. Based on the numbers of grid points in x, y and z directions and the available processor number, the user can specify the partitioning numbers in each direction and determine the sub-domains. Each sub-domain is then assigned to one processor. The computation is carried out in a single instruction multi data manner, i.e. the data is allocated separately in different processor elements and the sequence of operations is identical for all processors.

All physical variables are defined as allocatable arrays for each processor. We use the message passing interface (MPI) package for data communications. So, the parallel code is flexible and portable for different simulation configurations on any hardware with distributed memory.

As discussed before, most of the computations are carried out with explicit algorithms. For these explicit computations, we just need to transfer the data from the partitioning boundary cells to neighbouring processors and do the operations separately within each processor. In the present model, only one layer of holo cells between two

Downloaded by [Institute of Mechanics, CAS] at 21:07 16 October 2011

neighbouring processors is required for the inter-processor communications because of the compactness of the numerical schemes.

In the whole code, the only part that involves implicit computation is the solution of the pressure Poisson equation cast into the following form,

$$\mathbf{A}x = \mathbf{b}, \tag{4}$$

where **A** and **b** stand for the coefficient matrix (or matrix operator) and the source term derived from the multi-moment finite volume formulation for pressure Poisson equation, and x is the unknown pressure fields.

The iteration procedure of the Poisson equation takes a significant portion of the total CPU time. In Xiao (2001), a parallelisable iterative solver, namely Tridiagonal Factorisation BiCGSTAB (TF-BiCGSTAB) method is used. A fully parallelisable pre-conditioner TF_J is devised by incorporating a single step Jacobi splitting into the tridiagonal factorisation algorithm. Our numerical experiments as well as the practical simulations show that (TF_J)Bi-CGSTAB converges well even for large jumps in the coefficients or with singular source terms. However, for large scale problems, other iterative methods with multi-level smoothing are more demanding for the uniform convergence rate. For example, an application of a V-cycle multigrid method was reported in Blosch and Shyy (1996) for steady incompressible Navier–Stokes equations, where satisfactory convergence rate and scalability were achieved.

The multigrid method is a recursive smoothing procedure on different grid levels. A V-cycle multigrid iteration, for example, includes the following operations: (i) smoothing iterations on fine grid; (ii) restriction to transfer residuals to coarse grid; (iii) solving the coarse-grid problem to get the corrections on coarse grid; (iv) prolongation to interpolate the coarse-grid corrections to the fine grid and (v) corrections to the fine-grid solution. Due to the high efficiency in damping the errors on coarser grids, the multigrid method keeps a high convergence rate even for large scale problems.

We adopted in this paper the algebraic multigrid (AMG) method (Stüben 2001) as a preconditioner of a CG method to solve the pressure Poisson equation. Compared to the geometrical multigrid, AMG method does not use any information of the physical model nor the computational grid. The grids of different levels for the recursive computation and the corresponding operators are solely constructed from the information contained in the given matrix **A**. In an AMG, the constructions of the coarser levels and the interpolations are adapted locally according to the relaxation schemes, which makes the method flexible and robust for complex geometry and large scale simulations.

We make use of the AMG algorithm detailed in (Stüben 2001) with the following specific features:

- AMG is used as a pre-conditioner of the CG method.
- We specify the re-scaling factor $\alpha = 1.8$ in the computations presented in this paper, i.e.

  $$x^h_{\text{new}} = x^h_{\text{old}} + \alpha I^H_h e^H, \tag{5}$$

  where x_{old} and x_{new} denote the solutions before and after corrections on the fine level. I^H_h stands for the interpolation operator that transfers information from the coarse level to the fine level, and e^H is the converged solution of the residual equation on the coarse level.
- The incomplete Cholesky (IC) scheme is used as the smoother in the AMG method. On the parallel architecture with distributed memory, the IC algorithm is conducted locally on each individual processing unit.
- A hyperplane ordering is separately used in each processor to make the vectorisation of the computation on the Earth Simulator.

We evaluated the computational efficiencies of both TF-BiCGSTAB and AMG-CG as the iterative solvers to Poisson equation

$$\partial_{xx} f + \partial_{yy} f + \partial_{zz} f = \sin(\pi x)\sin(\pi y)\sin(\pi z), \tag{6}$$

with Dirichlet boundary conditions in 3D. We ran the sample code with a $150 \times 150 \times 150$ grid on a Linux PC which has Pentium4 3 GHz CPU, 2 GB memory and 80 GB hard disk drive.

The evolution of the residuals norm is plotted in Figure 1. It is obvious that AMG-CG has a much superior convergence rate. We have conducted numerical experiments with different grid size of the computational domain. Table 1 shows the iteration numbers and the CPU time in seconds required for convergence tolerance $|\mathbf{r}|/|\mathbf{b}| \leq 10^{-10}$ of the iteration solvers on grids of different sizes. The TF-BiCGSTAB method takes less computational cost because of less arithmetic operations per iteration. However, when the problem size increases, the iteration number and the execution time required for the converged solution of the TF-BiCGSTAB method grows significantly. The AMG-CG method, on the other hand, takes slightly more time on the coarse grid, but appears much more efficient for large size problems. More importantly, the iteration number of the AMG-CG method remains quite stable, whereas the TF-BiCGSTAB method needs much more iterations to get the converged solution. Since we are aiming at large scale simulations, the AMG-CG method is much more suitable due to the scalability in terms of problem sizes.

Considering the density ratio between air and water and the variable grid spacing in fluid simulations, the discontinuous jump in the coefficients of matrix **A** may have a magnitude larger than 10^3. Shown as follows, the

Downloaded by [Institute of Mechanics, CAS] at 21:07 16 October 2011

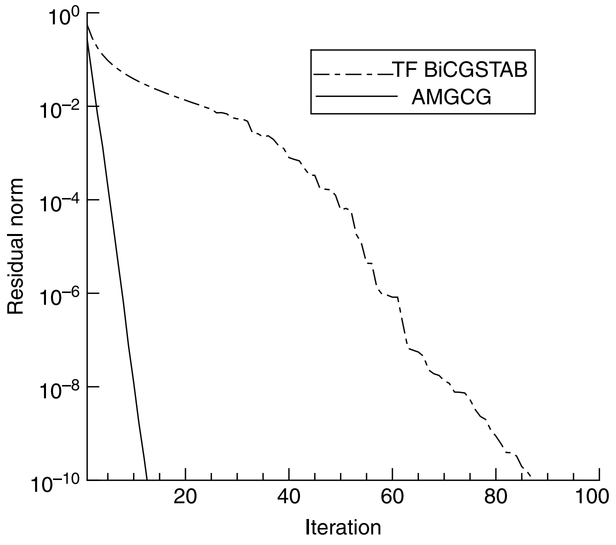

Figure 1. The history of the residual norms of the TF-BiCGSTAB and AMG-CG iterative methods. Solid line and dot dashed line represent the results of the AMG-CG and TF-BiCGSTAB, respectively.

much superior performance of the AMG-CG method is observed in the simulations of real multi-phase flows as well. It reveals that the AMG-CG method can be a robust pressure Poisson solver for multi-phase flows with good scalability.

For evaluating the parallel performance of the code, we define some measures of parallel acceleration. We denote the execution time with n processor by T_n, so the speed-up due to parallel computation is

$$S_{n1/n2} = \frac{T_{n1}}{T_{n2}}, \tag{7}$$

where we assume $n1 < n2$. Correspondingly, the parallel efficiency on $n2$ processors in respect to $n1$ processors is

$$E_{n1/n2} = S_{n1/n2} \times \frac{n1}{n2}. \tag{8}$$

Table 1. Numbers of iterations and run times required by TF-BiCGSTAB method and AMG-CG to reduce the normalised residual norm below 10^{-10} for the 3D poisson equation with Dirichlet boundary conditions on a cubic domain $X \times Y \times Z = 1 \times 1 \times 1$.

	TF-BiCGSTAB		AMG	
Grid	Iteration	Time [s]	Iteration	Time [s]
16^3	11	7.686e-3	8	0.012
32^3	20	0.148	10	0.137
64^3	42	8.118	11	2.655
128^3	78	131.5	11	31.81
150^3	88	212.2	13	61.13

The effective parallelisation ratio (or the effective parallelisation portion) can be then estimated by

$$\alpha = \frac{T_{n1} - T_{n2}}{(n2-1)/(n2)T_{n1} - (n1-1)/(n1)T_{n2}}. \tag{9}$$

As aforementioned, the TSUBAME supercomputer is a massively parallel computer with scalar processing elements of distributed memories. TSUBAME uses MPI as the library for data communication among distributed memory. Since our parallel code has been written in the standard MPI, it is straightforwardly portable to the TSUBAME supercomputer.

We evaluated the parallel performance of the code on TSUBAME supercomputer. Considering the current routine availability of the machine, we computed a test problem with a size of 300^3 grid. We used a 3D parallel partition, and ran the computations for 10 steps on 16 nodes (256 CPU) and 24 nodes (384 CPU), respectively. Table 2 shows the CPU time and other measures for parallel performance. It is observed that our parallelised code can be effectively accelerated with several hundred processors. A scalability evaluation in terms of problem size was also carried out. Based on the domain decomposition, computations of a fixed size 60^3 were assigned to each node. We increased the number of the processors as the total size of the problem increased. The CPU times with different number of processors and different sizes of problems are given in Table 3. It is observed that the larger size problem takes more computational time even with proportionally increased processors because of the increase in the communication overhead for the global data transfer and the growth in cost of the Poisson solver. Nevertheless, a parallel sustained speed-up can still be expected for large scale simulations.

As an example of the DNS of multi-phase flows we computed a bubbly flow with 35 spherical bubbles initially disposed in a vertical tube. Figure 2 shows a snapshot of the rising bubbles. The bubbles rise up under the floating force and interact with each other through the surrounding liquid. The fully developed rising bubbly flow exhibits a very complex behaviour.

The program was originally coded for scalar parallel machine, and the ratio of the vector operation is about 92%

Table 2. Parallel performance of the code on TSUBAME supercomputer with a 300 × 300 × 300 grid size.

Node	CPU	Time [s]	S_{perfect}	$S_{n1/n2}$	$E_{n1/n2}$	α
16	256 ($n1$)	70.17				
24	384 ($n2$)	56.13	1.5	1.25	0.833	0.997

S_{perfect} is the perfect parallel speed-up simply computed by $S_{\text{perfect}} = n2/n1$.

Downloaded by [Institute of Mechanics, CAS] at 21:07 16 October 2011

Table 3. Computational times of parallel tests of different problem sizes with proportionally different numbers of processors.

Problem size	CPU	Time [s]
27×60^3	216	5.37
125×60^3	1000	6.81

The subdomain size assigned to each node is fixed to be $60 \times 60 \times 60$. Ideal computational times for different cases should remain the same.

on the Earth Simulator before vectorisation tuning. In order to take the advantage of the vector processing capability of the Earth simulator, we have optimised the code. Our major effort is to replace the *if* logic operation by mask index vectors. With the automatic vectorisation function of the compiler, the current code has been vectorised up to 95%.

We evaluated the parallel performance of the code for two relatively large scale simulations with the grid points being of 2.048×10^9 and 4.096×10^9, respectively. For each problem, we ran with different numbers of CPU and extracted the measures for parallel performance. Shown in Tables 4 and 5, adequate parallel performance has been achieved. It should be notified that 512 is the largest number of nodes currently available for the whole system of the Earth Simulator. It is found that the code is able to make use of the full capacity of the Earth Simulator with high computational efficiency.

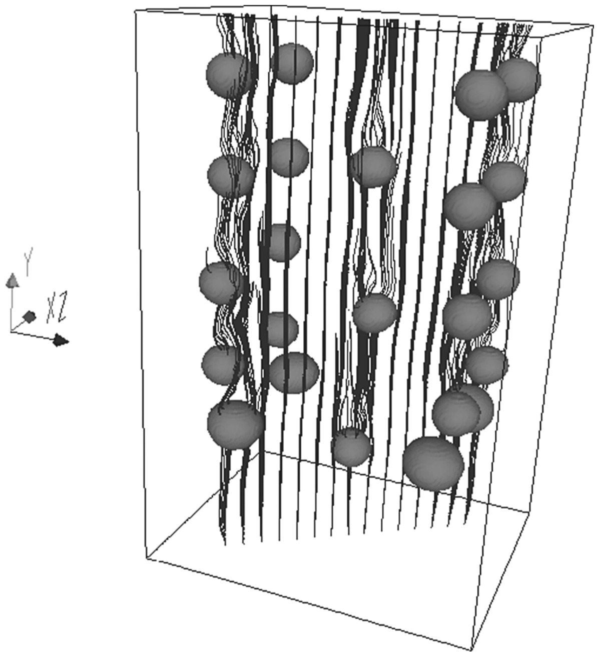

Figure 2. A bubbly flow with 35 spherical bubbles initially disposed in a vertical tube. Lines in the figure represent the streamline.

Table 4. Same as Table 2, but on earth simulator supercomputer with a $800 \times 6400 \times 400$ grid size.

Node	CPU	Time [s]	S_{perfect}	$S_{n1/n2}$	$E_{n1/n2}$	α (%)
156	1248 ($n1$)	425.086				
256	2048 ($n2$)	265.677	1.641	1.6	0.975	99.997

One of our major goals for implementing large scale simulations on the Earth Simulator is to directly compute the interactions of air and water, as well as the turbulent structures and the transport of mass and energy across the air/water interface. Being the preliminary numerical experiment, a wind-driven water tank has been simulated. We have carried out a large scale wind-driven interfacial flow simulation with a size of $800 \times 2000 \times 250$ grid points. To our knowledge, it is the largest simulation of its kind ever reported. Figure 3 displays the one-fifth portion of the free surface of the whole computational domain. With extremely high resolution of the computational grid, the fine structures of the surface were captured. Further studies on the turbulent structures and mass and energy transfer will be conducted via the large scale DNS with the computational tools reported in this communication.

In summary, we have implemented large scale numerical simulations for multi-phase interfacial flows on parallel supercomputers with distributed memory. The model is based on continuum dynamics and all the phases are computed via a unified approach. The interface between different fluids is solved by an algebraic interface capturing scheme, and an AMG-preconditioned CG method is used to solve the pressure Poisson equation. The resulting hydrodynamic code is well suited for parallel implementation.

The parallel performance of the code has been evaluated on two top supercomputers in Japan, i.e. the TSUBAME grid system in Tokyo Institute of Technology and the Earth Simulator. Adequate parallel acceleration has been achieved. Particularly, the satisfactory parallel speed-up is obtained on the Earth Simulator using 4096 CPUs. Our preliminary numerical experiments have made the large scale simulation of multi-phase interfacial flows with several billion grid points possible.

Table 5. Same as Table 4, but with a $1600 \times 6400 \times 400$ grid size.

Node	CPU	Time [s]	S_{perfect}	$S_{n1/n2}$	$E_{n1/n2}$	α (%)
480	3840 ($n1$)	153.486				
512	4096 ($n2$)	144.976	1.0667	1.0587	0.9925	99.996

Downloaded by [Institute of Mechanics, CAS] at 21:07 16 October 2011

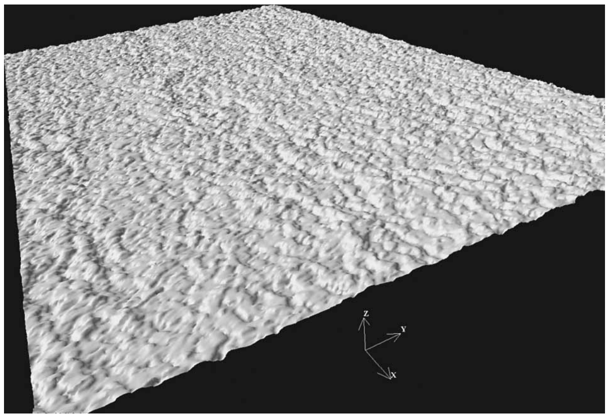

Figure 3. The air/water surface of a large scale wind-driven wave simulation. The total number of the computational grid points is 800 × 2000 × 250. Only one-fifth of the air/water surface is shown.

Acknowledgements

We gratefully acknowledge Dr Goto, Dr Fuchigami and Dr Nagano for their help in porting and parallelising the code and installation of the AMG-CG parallel solver in the Earth Simulator supercomputer.

References

Aliabadi, S., Johnson, A., Zellar, B., Abatan, A. and Berger, C., 2002. Parallel simulation of the flows in open channels. *Future Generation Computer System*, 18, 627–637.

Blosch, E.L. and Shyy, W., 1996. Scalability and performance of data-parallel pressure-based multigrid methods for viscous flows. *Journal of Computational Physics*, 125, 338–353.

Brackbill, J.U., Kothe, D.B. and Zemach, C., 1992. A continuum method for modeling surface tension. *Journal of Computational Physics*, 100 (2), 335–354.

Endo, T., Matsuoka, S., Hashizume, N., Nagasaka, M. and Goto, K., Performance evalution of TSUBAME heterogeneous supercomputer with Linpack. IPSJ SIG Notes No.87(20060731), 2006

Guler, I., Behr, M. and Tezduyar, T., 1999. Parallel finite element computation of free-surface flows. *Computational Mechanics*, 23, 117–123.

JAMSTEC, 2006. The Earth simulator, http://www.jamstec.go.jp/es/en/index.html

Stüben, K., 2001. A review of algebraic multigrid. *Journal of Computational and Applied Mathematics*, 128, 281–309.

Sussman, M., 2005. A parallelised adaptive algorithm for multiphase flows in general geometries. *Computers and Structures*, 83, 435–444.

Tryggvason, G., Esmaeeli, A., Lu, J. and Biswas, S., 2006. Direct numerical simulations of gas/liquid multi-phase flows. *Fluid Dynamics Research*, 38, 660–681.

Xiao, F., 2001. Implementation of multi-fluid hadridynamic simulations on distributed memory computer with a fully parallellisable preconditioned Bi-CGSTAB method. *Computer Physics Communications*, 137, 274–285.

Xiao, F., Ikebata, A. and Hasegawa, T., 2005. Numerical simulation of free-surface fluids by a multi-integrated moment method. *Computers and Structures*, 83, 409–423.

Xiao, F., Honma, Y. and Kono, T., 2005. A simple algebraic interface capturing scheme using hyperbolic tangent function. *International Journal for Numerical Methods in Fluids*, 48, 1023–1040.

Xiao, F., Akoh, R. and Ii, S., 2006. Unified formulation for compressible and incompressible flows by using multi-integrated moments II: multidimensional version for compressible and incompressible flows. *Journal of Computational Physics*, 213, 31–56.

Yabe, T., Xiao, F. and Utsumi, T., 2001. The constrained interpolation profile method for multiphase analysis. *Journal of Computational Physics*, 169, 556–593.

Yokoi, K., 2007. Efficient implementation of THINC scheme: a simple and practical smoothed VOF algorithm. *Journal of Computational Physics*, 226, 1985–2002.