# Solving Seven-Equation Model of Compressible Two-Phase Flow Using CUDA-GPU

Shan Liang[1,⋆], Wei Liu[2], and Li Yuan[3]

[1] Computer Network Information Center,
Chinese Academy of Sciences, Beijing 100190, China
liangshan@sccas.cn
[2] Academy of Mathematics and Systems Science,
Chinese Academy of Sciences, Beijing 100190, China

**Abstract.** In this paper, a numerical method which combines a HLLC-type approximate Riemann solver with the third-order TVD Runge-Kutta method is presented for the two-pressure and two-velocity seven-equation model of compressible two-phase flow of Saurel and Abgrall. Based on the idea proposed by Abgrall that "a multiphase flow, uniform in pressure and velocity at $t = 0$, will remain uniform on the same variables during time evolution", discretization schemes for the non-conservative terms and for the volume fraction evolution equation are derived in accordance with the adopted HLLC solver for the conservative terms. To attain high temporal accuracy, the third-order TVD Runge-Kutta method is implemented in conjunction with the operator splitting technique in a robust way by virtue of reordering the sequence of operators. Numerical tests against several one- and two-dimensional compressible two-fluid flow problems with high density and high pressure ratios demonstrate that the proposed method is accurate and robust. Besides, the above numerical algorithm is implemented on multi graphics processing units using CUDA. Appropriate data structure is adopted to maintain high memory bandwidth; skills like atom operator, counter and so on are used to synchronize thread blocks; overlapping domain decomposition method is applied for mission assignment. Using a single-GPU, we observe 31× speedup relative to a single-core CPU computation; linear speedup can be achieved by multi-GPU parallel computing although there might be a little decrease in single-GPU performance. *abstract* environment.

**Keywords:** compressible multiphase flow, seven-equation model, HLLC, TVD Runge-Kutta, GPU.

## 1 Introduction

Compressible multiphase flows exist broadly in nature and industry (like bubbles in ocean, cavitation in hydraulic machinery, flows in reactors and cooling

⋆ Corresponding author.

K. Li et al. (Eds.): ParCFD 2013, CCIS 405, pp. 25–36, 2014.

circuits of power plants). Numerical simulation of such flows is an important research topic. As the flows are complex and diverse, many multiphase models with various levels of complexity were proposed in literature, like the seven-equation model of Baer and Nunziato [1], that of Saurel *et al.* [2, 3], and the reduced five-equation model of Allaire [4], to mention just a few. Most multiphase models are derived from integrating individual balance equations weighted by a characteristic function for each phase. This volume average process removes the interfacial details while introducing additional non-conservative terms. The resultant multiphase models pose challenge to numerical solutions mainly due to the complicated characteristics of the equation system and the troublesome non-conservative terms.

In this paper, we are interested in numerical solution of the compressible multiphase model formulated by Saurel and Abgrall [2, 3]. In this model, each phase is assumed to have its own velocity, pressure and density, which satisfy respective balance equations. The evolution equation of volume fraction is introduced from integrating the characteristic function to describe how fluid compositions change with time. Due to non-equilibrium of velocity and pressure, drag forces appear between phases causing momentum and energy change. In the case of one space dimension, the model has seven equations (two sets of mass, momentum, and energy equations, one volume fraction evolution equation). The advantage of this model is that it is unconditionally hyperbolic, and can treat a wide range of applications including non-equilibrium dispersive multiphase flows as well as free-surface multi-fluid flows [3]. For the latter case, the velocity and pressure of all phases on each side of the interface must be in equilibrium from a physical point of view [2, 3, 5], which can be realized by infinite pressure and velocity relaxation process in the model. As the volume fraction only stands for the constitutive fluid distribution, the material surface is indirectly represented by location where large gradient occurs. The interface is tracked without considering the details even when the distortion is complicate (cavitation, breakdown and coalescence of bubbles, etc). Of course, the computational cost is larger than a free-surface oriented method, e.g., about three times as large as a ghost fluid method from our experience.

Although this model is unconditional hyperbolic, the numerical solution has particular difficulties because it is hard to solve associated Riemann problem with a large system of equations, and careless approximations to the non-conservative terms in the momentum and energy equations and the non-conservative evolution equation (the volume fraction equation) will often lead to failure in computation. Therefore, the key in numerical solution is to construct an accurate and efficient approximate Riemann solver and in the meantime derive corresponding discretization schemes for the non-conservative terms and the non-conservative volume fraction equation.

Many studies have devoted to numerical solution of compressible two-phase models in various variants. Saurel [2, 3] used operator splitting approach to solve their seven-equation model, but the adopted HLL approximate Riemann solver was inaccurate due to the use of two waves instead of full waves and this led

to excessive numerical diffusion of contact discontinuities. In [6], combining thin layer theorem with special choice for interfacial variables in liquid-solid problem, Tokareva and Toro proposed a HLLC-type approximate Riemann solver which took into account full waves for the Baer-Nunzatio model. Li *et al.* [7] developed a general and simple HLLC scheme for the two-phase model but was confined to subsonic case only, and they used Roe average as the unknown intermediate state of the volume fraction. Tian *et al.* [8] also studied numerical solution of the reduced five-equation by using path-conservative method and a simple HLLC solver. A more thorough effort to construct approximate Riemann solver for the Saurel-Abgrall model was made recently by Ambroso *et al.* [9]. Their definition of Riemann problem included not only convective terms and non-conservative terms, but also source terms associated with gravity and drag forces (the latter refers to so-called velocity relaxation process), while pressure relaxation process was split from them alone. In all the above mentioned work, the multiphase flow equations were approximated by a numerical method, but a strategy proceeded in the opposite way was proposed in Abgrall and Saurel's subsequent work [10], which was able to deal with mixtures and interfaces under a unique formulation. They started from the pure phase Euler equations at the microscopic level, and gave the corresponding numerical approximations via the Godunov scheme and the HLLC flux. After randomization, ensemble average procedures and estimation of the various coefficients of these approximations, the numerical scheme for the averaged multiphase flow equations was derived.

In this study, our objective is to develop a high resolution method for the Saurel-Abgrall two-phase model, which has the simplicity of a standard HLLC scheme and the high temporal accuracy of the third-order TVD Rung-Kutta method, and also is robust to simulate extreme compressible gas-liquid two-fluid flow problems. We advance the solution with the third-order TVD Runge-Kutta method, incorporating the splitting strategy [3] between the hyperbolic parts and the relaxation terms into every sub-step of the Runge-Kutta method. The sequence of the split operators is reordered for robust treatment of free surface flows. To obtain a simple scheme for the two-phase model, we apply the conventional HLLC flux to the conservative part of the two-phase model in a way similar to [7], and then utilize the homogeneity idea of a multi-phase system as proposed by Abgrall to deriving discrete formulas for the non-conservative terms and non-conservative equation corresponding to the HLLC solver used. Our derivation takes into account all cases of HLLC scheme rather than only subsonic case as in [7]. The characteristic boundary conditions for the model are also presented.

Driven by the market demand for real-time, high-definition 3-D graphics, general-purpose graphic processing unit (GPGPU) has been developed for parallel computing decades ago. Many CFD simulations have beed done using GPU.

In order to solve large-scale problems efficiently, the second part of our study is to implement the above numerical algorithm on multi graphics processing units using CUDA. Firstly, every computational mesh point is assigned to a GPU thread, and appropriate data structure is adopted to maintain high memory

bandwidth; skills like atom operator, counter and so on are used to synchronize thread blocks. Besides, we use overlapping domain decomposition method for multi- GPU parallel computing. The computation of every subdomain is assigned to a device, the communication between which is done through MPI or Pthread.

## 2 Seven-Equation Model

For the sake of simplicity, we consider 1D case first. Suppose that two kinds of fluid exist in the system, and there is neither mass transfer nor heat exchange between them. The compressible two-phase model of Saurel-Abgrall [2] can be written in the following form:

$$\frac{\partial U}{\partial t} + \frac{\partial F(U)}{\partial x} = H(U)\frac{\partial \alpha_1}{\partial x} + S_v(U) + S_p(U) \tag{1}$$

where

$$U = \begin{pmatrix} \alpha_1 \\ \alpha_1\rho_1 \\ \alpha_1\rho_1 u_1 \\ \alpha_1 E_1 \\ \alpha_2\rho_2 \\ \alpha_2\rho_2 u_2 \\ \alpha_2 E_2 \end{pmatrix} \quad F(U) = \begin{pmatrix} 0 \\ \alpha_1\rho_1 u_1 \\ \alpha_1\rho_1 u_1^2 + \alpha_1 p_1 \\ \alpha_1 u_1(E_1 + p_1) \\ \alpha_2\rho_2 u_2 \\ \alpha_2\rho_2 u_2^2 + \alpha_2 p_2 \\ \alpha_2 u_2(E_2 + p_2) \end{pmatrix} \quad H(U) = \begin{pmatrix} -u_I \\ 0 \\ p_I \\ u_I p_I \\ 0 \\ -p_I \\ -u_I p_I \end{pmatrix}$$

$$S_v(U) = \begin{pmatrix} 0 \\ 0 \\ \lambda(u_2 - u_1) \\ \lambda(u_2 - u_1)u_I \\ 0 \\ -\lambda(u_2 - u_1) \\ -\lambda(u_2 - u_1)u_I \end{pmatrix} \quad S_p(U) = \begin{pmatrix} -\mu(p_2 - p_1) \\ 0 \\ 0 \\ \mu p_I(p_2 - p_1) \\ 0 \\ 0 \\ -\mu p_I(p_2 - p_1) \end{pmatrix}$$

Here $\rho$, $u$, $p$ and $E = \rho e + 1/2\rho u^2$ represent the density, velocity, pressure and total energy per volume respectively, and subscripts $k = 1$ or 2 is related to phase $k$. $\alpha_k$ is the volume fraction ranged from 0 to 1, and satisfies relation $\alpha_1 + \alpha_2 = 1$. When $\alpha_k = 0$ there is no $k$-th fluid. Actually, to avoid infinite density and velocity computed by mass and energy conservation equations, $\alpha_k = 10^{-7}$ is used in lieu of $\alpha_k = 0$ at the initial time. $\mu$ and $\lambda$ are the pressure and velocity relaxation coefficients, and $p_I$ and $u_I$ are the interfacial pressure and velocity respectively. Various choices are possible, for example, in the gas-liquid (or solid) problem, $p_I = p_{\text{gas}}$ and $u_I = u_{\text{liquid}}$. Here we follow Saurel's choice [3]:

$$p_I = \sum \alpha_k p_k, \quad u_I = \sum(\alpha_k \rho_k u_k) / \sum \alpha_k \rho_k \tag{2}$$

## 3 Numerical Method

Based on the operator splitting method given in [3], equation (1) can be decomposed into three parts: the ODE system of pressure relaxation $U_t = S_p(U)$, that of velocity relaxation $U_t = S_v(U)$, and the non-conservative hyperbolic equations $U_t + F(U)_x = H(U)\alpha_{1x}$. In [3], the complete solution of (1) was obtained by the succession of operators:

$$U^{n+1} = L_p^{\Delta t} L_v^{\Delta t} L_H^{\Delta t}(U^n) \tag{3}$$

where $L_p$ and $L_v$ denote the pressure and velocity relaxation operators respectively, and $L_H$ denotes the non-conservative hyperbolic operator. Scheme (3) is only first order accurate in time and higher accuracy can be obtained by adopting Strange splitting or Runge-Kutta method. In this paper, the third-order TVD Runge-Kutta method is used together with operator splitting.

### 3.1 Hyperbolic Operator

The numerical scheme is based on the idea proposed by Abgrall [5] that"a two phase system, uniform in velocity and pressure at $t = 0$ will remain uniform on the same variables during time evolution".

The considered hyperbolic equations read

$$\alpha_t + u_I \alpha_x = 0 \tag{4}$$

$$Q_t + F(Q)_x = H(Q)\alpha_x \tag{5}$$

$Q = (\alpha\rho, \alpha\rho u, \alpha E)^T, F(Q) = (\alpha\rho u, \alpha\rho u^2 + \alpha p, \alpha u(E + p))^T, H(Q) = (0, p_I, p_I u_I)^T$.

We wish to construct :

- upwind scheme for (4) of the form

$$\alpha_j^{n+1} = \alpha_j^n - \frac{\Delta t}{\Delta x}(u_I)_j^n(\phi_{j+1/2} - \phi_{j-1/2}) \tag{6}$$

- Godunov-type finite volume method for (5) of the form

$$Q_j^{n+1} = Q_j^n - \frac{\Delta t}{\Delta x}\left(F_{j+1/2}^n - F_{j-1/2}^n\right) + \Delta t H(Q_j^n)\Theta \tag{7}$$

### 3.2 Third-Order TVD Runge-Kutta Scheme

For Eq. (1), the Godunov splitting (3) can be seen as a forward Euler step $U^{n+1} = U^n + \Delta t L(U^n)$ which is the sum of the last two terms of each substep of standard TVD Runge-Kutta scheme. As a naive implementation, we can apply TVD RK3 method to Eq. (1) directly

$$\begin{cases} U^{(1)} &= U^n + \Delta t L(U^n) \\ U^{(2)} &= \frac{3}{4}U^n + \frac{1}{4}U^{(1)} + \frac{1}{4}\Delta t L(U^{(1)}) \\ U^{(n+1)} &= \frac{1}{3}U^n + \frac{2}{3}U^{(2)} + \frac{2}{3}\Delta t L(U^{(2)}) \end{cases} \rightarrow \begin{cases} U^{(1)} &= L_p^{\Delta t} L_v^{\Delta t} L_H^{\Delta t} U^n \\ U^{(2)} &= \frac{3}{4}U^n + \frac{1}{4}L_p^{\Delta t} L_v^{\Delta t} L_H^{\Delta t} U^{(1)} \\ U^{(n+1)} &= \frac{1}{3}U^n + \frac{2}{3}L_p^{\Delta t} L_v^{\Delta t} L_H^{\Delta t} U^{(2)} \end{cases} \tag{8}$$

However, the problem with (8) is that it does not guarantee the pressure and velocity are in equilibrium at the end of each time step. To solve this problem, we reorder the sequence of operators

$$U^{(1)} = L_H^{\Delta t} L_p^{\Delta t} L_v^{\Delta t} U^n \tag{9}$$

$$U^{(2)} = \frac{3}{4} U^n + \frac{1}{4} L_H^{\Delta t} L_p^{\Delta t} L_v^{\Delta t} U^{(1)} \tag{10}$$

$$U^{(3)} = \frac{1}{3} U^n + \frac{2}{3} L_H^{\Delta t} L_p^{\Delta t} L_v^{\Delta t} U^{(2)} \tag{11}$$

$$U^{n+1} = L_p^{\Delta t} L_v^{\Delta t} U^{(3)} \tag{12}$$

(9)(10)(11)are the three sub-steps of the Runge-Kutta method, and (12) is an additional step, and the total computational count is not increased compared with the naive implementation (8).

## 4 Implementation on GPU

In order to solve large-scale problems efficiently, we present an implementation of the above numerical algorithm on multi graphics processing units using CUDA.

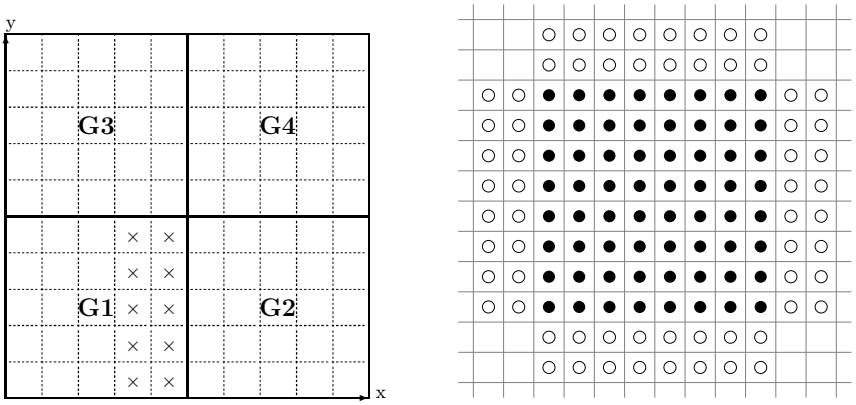
### 4.1 Implementation on Multi-GPUs

Although single GPU has achieved high performance in many applications, the compute capacity that required for simulating large-scale multi-phase flow problems is far beyond what a single GPU can deliver. Consequently, it is worthwhile to develop multi-GPU parallelization techniques.

In our work, domain decomposition method is used. A 2D grid of $N_x \times N_y$ is divided into $m$ and $n$ parts along its two coordinate directions respectively, which makes the total sub-domain number $m \times n$ (Fig. 1(a)), The computation of each sub-domain is assigned to different GPUs. There are two principles to follow: firstly, $m \times n$ equals the total number of GPUs in use; secondly, the shape of each sub-domain should be as square-like as possible to reduce the amount of information exchanged between GPUs. Indeed, flow field variables of a sub-domain reside in the memory of only one GPU from the very beginning to the end of a simulation, and they will be exported to different files when the simulation ends. A five point difference module, including the target cell and the four cells that adjacent to it in 2D, is needed to support the MUSCL reconstruction; and the reconstructed results of two adjoining cells are used to estimate numerical fluxes in the vicinity of their common cell face. Therefore, as sketched in Fig. 1(b), two layers of auxiliary grids is defined at every boundary of a sub-domain to couple computation. Take sub-domain G1 and G2 in Fig. 1(a) for example, the left two layers of auxiliary grids (invisible) of G2 is updated according to the right-most and second from right layers of initial grids (marked by $\times$) in G1.

To facilitate communications, the inner-boundary data is passed between devices by means of MPI or Pthread, which is operated in CPU. Considering that most computation is performed in GPU, to avoid explicit data copy between CPU and GPU frequently, mapped pinned memory is used to store auxiliary boundary data, which has two addresses: one at host and the other in device. When boundary computation is finished, each device uploads the boundary data that others need to mapped pinned memory, and then downloads the auxiliary boundary data for next time step computation. As is seen in Fig. 2, there is a little difference in MPI-CUDA and Pthread-CUDA application: in Pthread-CUDA, CPU threads share a common mapped pinned array as Fig. 2(a), and no data transfer is performed; while in MPI-CUDA, data is stored in different arrays which is passed between MPI processes as Fig. 2(b).

The main process is described in Fig. 3. Notice that the entire computation is performed on the GPUs, leaving the CPU almost idle during all stages of the computation except performing data copy and subsequent communication.

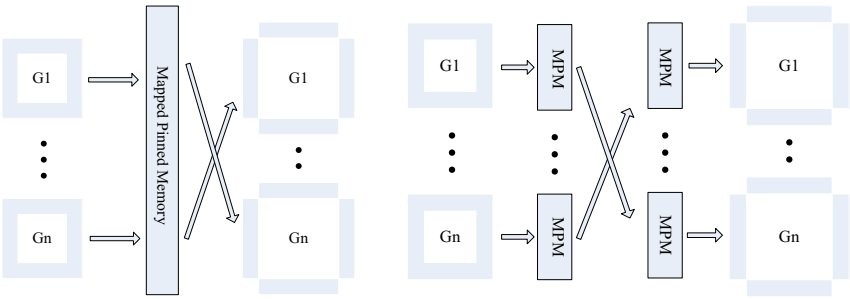



(a) Domain decomposition (b) Auxiliary grids

**Fig. 1.** Grid and mission assignment to four GPUs. (a) Decomposition of a domain into sub-domains. A sub-domain is computed by the corresponding GPU; (b) Computation domain for a single GPU. Solid points denote initial cells allocated to the current GPU while auxiliary cells are marked by hollow points.

## 5 Numerical Results

In this section, numerical tests with several 1D and 2D compressible gas-liquid multi-fluid problems are provided. For 1D cases, we compared our method with the RGFM as well as Saurel's HLL method [2] under the same MUSCL reconstruction. RGFM [11] can capture material interface with little diffusion, but it is not a conservative method. HLL is robust, but too diffusive. We include examples with high density and high pressure ratios to test these numerical methods. 2D problems are simulated with the help of GPU, and speedup of single- / multi-GPU is presented.




(a) Pthread-CUDA

(b) MPI-CUDA (mapped pinned memory is called MPM for short)

**Fig. 2.** Data exchange between devices. G stands for grid, and the blue area stands for boundary data that passed between devices.

### 5.1 Gas-Liquid Shock Tube (High Pressure Ratio)[12]

We consider a shock tube filled on the left side with high pressure gas and on the right side with liquid. The initial data are

$$(\rho, u, p, \gamma, B) = \begin{cases} (1.27,\ 0,\ 8000,\ 1.4,\ \ 0), & x \le 0.4 \\ (1.0,\ \ 0,\ 1.0,\ \ \ 7.15,\ 3309), & x > 0.4 \end{cases}$$

In this problem, the pressure ratio at the gas-liquid interface is up to $8000:1$, while density ratio is $1.27:1$. Numerical simulation was conducted to $t = 0.002$. Pressure and density distributions near the material interface are shown in Fig 4. The solid black line is results obtained by RGFM method. It can be seen that there is a density decrease at the interface which might be caused by the non-conservative error of RGFM. Results of multiphase method (the dashed line) look closer to the analytical solution (the solid grey line).

### 5.2 Underwater Explosion[13]

Material interface is located at $x = 0.5$ initially. The initial data are

$$(\rho, u, p, \gamma, B) = \begin{cases} (0.01,\ 0,\ 1000,\ 2,\ \ \ 0), & x \le 0.5 \\ (1.0,\ \ 0,\ 1.0,\ \ \ 7.15,\ 3309), & x > 0.5 \end{cases}$$

Rarefaction wave with very high speed will appear on the left side. At a very short time $t = 0.000718$, corresponding velocity and density distributions are shown in Fig 5. Results of HLL scheme [2] and present HLLC method are compared. HLL scheme uses only two waves instead of three, and contact discontinuity is heavily smeared out. This might also explain the velocity distribution is abnormal near the interface. Results computed by HLLC scheme are more reasonable and have less numerical diffusion.

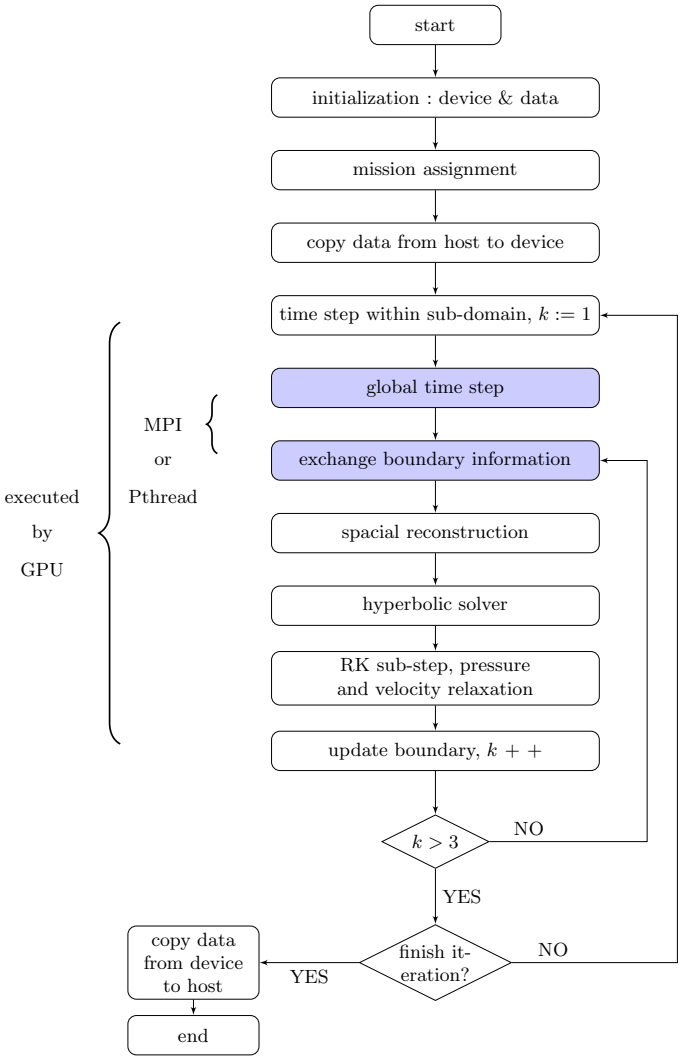



**Fig. 3.** Computational flow chart for multi-GPU implementation

### 5.3 Shock-Bubble Interaction

This is a classical problem [14–16]. A 2D bubble filled with helium initially in equilibrium is surrounded by air. Initially the bubble is located at $x = 175$ and $y = 44.5$; its radius is equal to 25. A shock coming from the right of the bubble is characterized by Mach number $M_s = 1.22$. The data are

$$(\rho, u, v, p, \gamma) = \begin{cases} (1.3764, -0.394, 0, 1.5698, 1.4), & \text{postshock} \\ (1, \quad 0, \quad 0, 1, \quad 1.4), & \text{preshock} \\ (0.138, \ 0, \quad 0, 1.0, \quad 1.67), & \text{inside bubble} \end{cases}$$

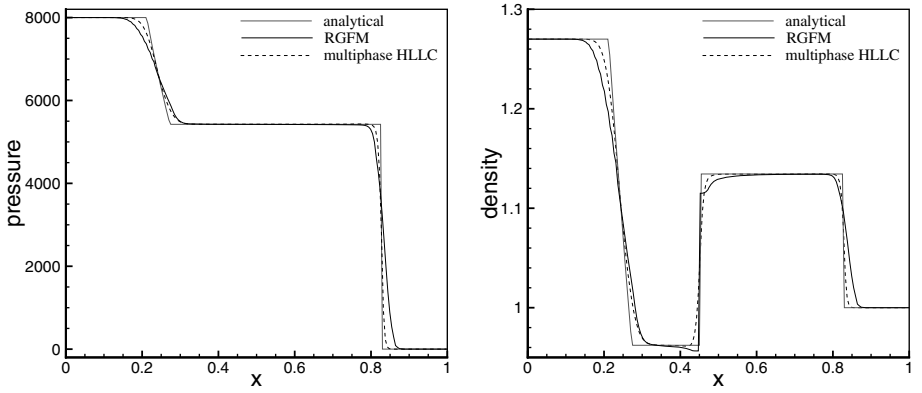



**Fig. 4.** Flow distributions in Gas-Liquid Shock Tube problem

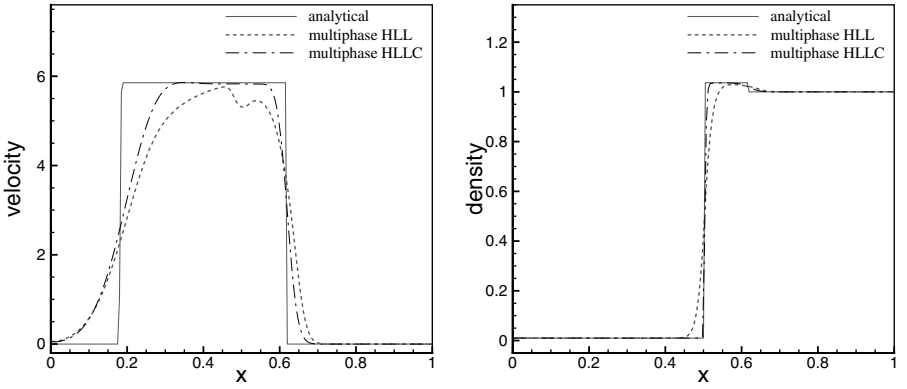



**Fig. 5.** Flow distributions in the problem of underwater explosion

The computational domain is $[0, 325] \times [0, 89]$, and $1024 \times 256$ uniform meshes are used. The simulation is done with the help of CUDA-GPU. Using a single-GPU, we observe $31\times$ speedup relative to a single-core CPU computation from Table 1; linear speedup can be achieved by multi-GPU parallel computing although there might be a little decrease in single-GPU performance, which is shown in Table 2. Our numerical behavior is in accordance with the dynamics of the bubble-shock interaction observed in the physical experiment by Haas [16]. Comparisons are done at different times, one of which is shown in Fig 6. The bubble outline is displayed by volume fraction contour $\alpha_1 = 0.5$. It is seen the results are comparable.

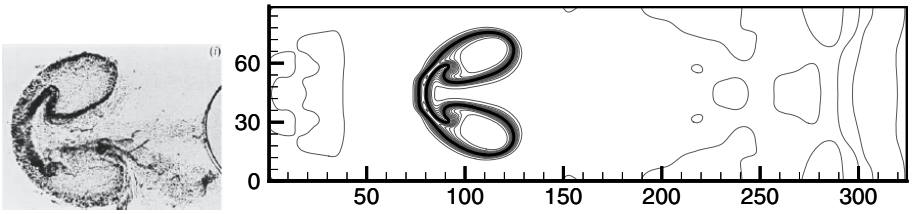



**Fig. 6.** Comparison of experiment (left) and computed density contours (right) at $t = 674\mu s$ in shock-bubble interaction problem

**Table 1.** Speedup (GPU *vs* single-core CPU)

| grid number | CPU time | GPU time | speedup |
|---|---|---|---|
| $128 \times 32$ | 0.097503 | 0.002830 | 34.45 |
| $256 \times 64$ | 0.358785 | 0.011469 | 31.28 |
| $512 \times 128$ | 1.434512 | 0.042854 | 33.47 |
| $1024 \times 256$ | 4.634705 | 0.167221 | 27.71 |

**Table 2.** Multi-GPU speedup

| GPU number | speedup | single-GPU efficiency |
|---|---|---|
| 1 | 1 | 100% |
| 2 | 1.836 | 91.83 % |
| 4 | 3.266 | 81.64% |
| 8 | 5.460 | 68.25% |

**Acknowledgments.** This work is supported by Natural Science Foundation of China (91130019) and 863 programme (2012AA01A304).

## References

1. Baer, M., Nunziato, J.: A two-phase mixture theory for the deflagration-to-detonation transition (ddt) in reactive granular materials. International Journal of Multiphase Flow 12, 861–889 (1986)
2. Saurel, R., Abgrall, R.: A multiphase godunov method for compressible multifluid and multiphase flows. J. Comput. Phys. 150, 425–467 (1999)
3. Saurel, R., Lemetayer, O.: A multiphase model for compressible flows with interfaces, shocks, detonation waves and cavitation. J. Fluid Mech. 431, 239–271 (2001)
4. Allaire, G., Clerc, S., Kokh, S.: A five-equation model for the simulation of interfaces between compressible fluids. J. Comput. Phys 181, 577–616 (2002)
5. Abgrall, R.: How to prevent pressure oscillations in multicomponent flow calculations: a quasi conservative approach. J. Comput. Phys. 125, 150–160 (1996)
6. Tokareva, S.A., Toro, E.F.: Hllc-type riemann solver for the baer-nunziato equations of compressible two-phase flow. J. Comput. Phys. 229, 3573–3604 (2010)

7. Li, Q., Feng, H.J., Cai, T., Hu, C.B.: Difference scheme for two-phase flow. Applied Mathematics and Mechanics 25, 536–545 (2004)
8. Tian, B., Toro, E.F., Castro, C.E.: A path-conservative method for a five-equation model of two-phase flow with an hllc-type riemann solver. Computers and Fluids 46, 122–132 (2011)
9. Ambroso, A., Chalons, C., Raviart, P.-A.: A godunov-type method for the seven-equation model of compressible two-phase flow. Computers and Fluids 54, 67–91 (2012)
10. Abgrall, R., Saurel, R.: Discrete equations for physical and numerical compressible multiphase mixtures. J. Comput. Phys. 186, 361–396 (2003)
11. Wang, C.W., Liu, T.G., Khoo, B.C.: A real ghost fluid method for the simulation of multimedium compressible flow. SIAM J. Sci. Comput. 28, 278–302 (2006)
12. Liu, T.G., Khoo, B.C., Yeo, K.S.: Ghost fluid method for strong shock impacting on material interface. J. Comput. Phys. 190, 651–681 (2003)
13. Tang, H.S., Huang, D.: A second-order accurate capturing scheme for 1d invisid flows of gas and water with vacuum zones. J. Comput. Phys. 128, 301–318 (1996)
14. Fedkiw, R.P., Aslam, T., Merriman, B., Osher, S.: A non-oscillatory eulerian approach to interfaces in multimaterial flows (the ghost fluid method). J. Comput. Phys. 152, 457–492 (1999)
15. Quirk, J.J., Karni, S.: On the dynamics of a shock-bubble interaction, Tech. Rep. 94-75, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA (September 1994)
16. Haas, J.-F., Sturtevant, B.: Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities. Journal of Fluid Mechanics 181, 42–76 (1987)