Computers & Fluids 99 (2014) 156–171

Contents lists available at ScienceDirect

Computers & Fluids

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

Solving seven-equation model for compressible two-phase flow using multiple GPUs

Shan Liang [a,b,c], Wei Liu [a], Li Yuan [a,*]

[a] LSEC and NCMIS, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing 100190, China
[b] Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China
[c] State Key Laboratory of Mathematical Engineering and Advanced Computing, Wuxi 214000, China

ARTICLE INFO

Article history:
Received 17 October 2013
Received in revised form 17 April 2014
Accepted 20 April 2014
Available online 28 April 2014

Keywords:
Compressible multiphase flow
Seven-equation model
HLLC
TVD Runge–Kutta
GPU computing

ABSTRACT

In this paper, the application of an HLLC-type approximate Riemann solver in conjunction with the third-order TVD Runge–Kutta method to the seven-equation compressible two-phase model on multiple Graphics Processing Units (GPUs) is presented. Based on the idea proposed by Abgrall et al. 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 HLLC solver used for the conservative terms. To attain high temporal accuracy, the third-order TVD Runge–Kutta method is implemented in conjunction with operator splitting technique, in which the sequence of operators is recorded in order to compute free surface problems robustly. For large scale simulations, the numerical method is implemented using MPI/Pthread-CUDA parallelization paradigm for multiple GPUs. Domain decomposition method is used to distribute data to different GPUs, parallel computation inside a GPU is accomplished using CUDA, and communication between GPUs is performed via MPI or Pthread. Efficient data structure and GPU memory usage are employed to maintain high memory bandwidth of the device, while a special procedure is designed to synchronize thread blocks so as to reduce frequencies of kernel launching. Numerical tests against several one- and two-dimensional compressible two-phase flow problems with high density and high pressure ratios demonstrate that the present method is accurate and robust. The timing tests show that the overall speedup of one NVIDIA Tesla C2075 GPU is 31× compared with one Intel Xeon Westmere 5675 CPU core, and nearly 70% parallel efficiency can be obtained when using 8 GPUs.

1. Introduction

Compressible two-phase flows exist broadly in nature and industry (like bubbles in ocean, cavitations in hydraulic machinery, flows in chemical reactors and cooling circuits of power plants). Numerical simulations of compressible two-phase flows are important research topics. As this type of flows are complex and diverse, a variety of two-phase models with various levels of complexity were proposed in literature, like the complete seven-equation model [1–3], the reduced six-equation model [4], and the more reduced five-equation model [5], to mention just a few. Most two-phase models are derived from integrating individual balance equations weighted by a characteristic function for each phase. This volume average procedure removes the interfacial detail while introducing additional non-conservative terms for describing interactions between phases. The resultant two-phase 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 seven-equation two-phase model [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 exchange. 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

* Corresponding author. Tel.: +86 10 62625219.
E-mail address: lyuan@lsec.cc.ac.cn (L. Yuan).

http://dx.doi.org/10.1016/j.compfluid.2014.04.021
0045-7930/

[2,3,6], 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 bulk interface is tracked without considering the details even when the distortion is complicated (cavitation, breakdown and coalescence of bubbles, etc.). Of course, the computational cost is larger than a free-surface oriented method, e.g., it is about three times as that of the ghost fluid method according to our experience.

Although the seven-equation model is unconditional hyperbolic, the numerical solution has particular difficulties because it is hard to solve the 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 at the same time 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 et al. [2,3] used operator splitting approach to treat the hyperbolic part and the relaxation terms of their seven-equation model, but the adopted HLL approximate Riemann solver led to excessive numerical diffusion of contact discontinuities due to the use of only two waves in lieu of full waves. Li et al. [7] developed a simple HLLC scheme for the seven-equation model, but they only considered the subsonic case, and used Roe average for the unknown intermediate state of the volume fraction. Zein et al. [8] also presented a simple HLLC-type scheme for the seven-equation model that took into account the heat and mass transfer through relaxation effects. Combining the thin layer theory with special choice for interfacial variables in liquid–solid problems, Tokareva and Toro [9] proposed a HLLC-type approximate Riemann solver which took into account full waves for the Baer-Nunzatio model. Tian et al. [10] implemented the path-conservative method and a simple HLLC solver for the reduced five-equation model. Yeom and Chang [4] presented a modified HLLC-type scheme for a six-equation model which restores the characteristic fields that have been neglected in the Zein's simple HLLC-type scheme [8]. A more thorough effort to construct approximate Riemann solver for the Saurel-Abgrall model was made recently by Ambroso et al. [11]. 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 drag force source is often separately treated as velocity relaxation process), while pressure relaxation process was split from them alone. In all the work mentioned above, the multiphase flow equations were approximated by numerical methods, but a strategy proceeded in the opposite way was proposed by Abgrall [12], which dealt with mixtures and interfaces under a unique formulation. They started from the pure phase Euler equations at the microscopic level, and gave 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, numerical scheme for the averaged multiphase flow equations was derived.

In this study, the first objective is to develop a robust high resolution numerical method for the Saurel-Abgrall's seven-equation compressible two-phase model, which has the simple form of a conventional HLLC scheme and the high temporal accuracy of the third-order TVD Rung-Kutta method. We advance the solution with the third-order TVD Runge–Kutta method, inserting the splitting strategy [3] for the hyperbolic operator and the relaxation operators into every sub-step of the Runge–Kutta method. We reorder the sequence of the split operators so that the resulting Runge–Kutta method can work robustly for extreme compressible gas–liquid two-fluid flow problems with high density and high pressure ratios. To obtain a simple scheme, we apply the conventional HLLC flux to the conservative part of the two-phase model in a way similar to Li [7] and Zein [8], and then utilize the homogeneity idea for a multi-phase system [6] to derive discrete formulas for the non-conservative terms and the non-conservative evolution equation corresponding to the HLLC flux used. Our derivation takes into account both subsonic and supersonic cases of the HLLC scheme rather than only subsonic case as did in Ref. [7].

The second objective of this study is to efficiently reduce the simulation time posed by numerical solution of the seven-equation two-phase model. To this end, we implement our numerical method using CUDA-GPU parallel computing technology. CUDA (Compute Unified Device Architecture) [13] is a programming model for realizing general purpose GPU (Graphics Processing Unit) computing. Recently there is a surge in hybrid CPU/GPU computations using rapidly evolving GPU architectures and CUDA programming paradigm [14]. In single GPU computing, Ref. [15] accelerated a solver for the Euler equations, and observed 29× and 16× speedups for 2D and 3D problems respectively. Ref. [16] pioneered GPU acceleration of problems on non-uniform and irregular grids for the 3D compressible Euler equations, and obtained 15× to 40× speedups using NVIDIA 8800GTX GPU compared with a single Intel Core 2 Duo E6600(2.4 GHz) CPU. When conducting GPU computing on multiple GPUs, three parallel modes are available: single-thread multi-stream mode (CUDA 4.0 and above), in which every involved device is bounded to a CUDA stream; multi-thread multi-GPU mode (Pthread- or OpenMP-CUDA), in which more than one threads are invoked and each thread controls one GPU; multi-process multi-GPU mode (MPI-CUDA). The former two modes are applicable only to a shared memory machine with several GPUs, while the third one is also applicable to a cluster with many GPUs. Ref. [17] implemented 3D incompressible Navier–Stokes equations in CUDA with the help of Pthread. Using four Tesla C870 GPUs, the computation time was reduced to 1/100 of single AMD Opteron 2.4 GHz CPU, and 1/3 of a single GPU. Ref. [18] obtained linear speedup for fewer than four GPUs when solving a 3D equation using a high order finite difference method. Ref. [19] optimized a finite difference code for direct numerical simulations of turbulence on a GPU accelerated cluster, and obtained a speedup of 20× for 192 M2070 Fermi GPUs vs. 192 Xeon Westmere 2.93 GHz CPU cores. In their implementation, all computations were packed in kernel functions for running in the devices (GPUs), and the communication between devices was done through MPI. The boundary cells of each subdomain were dealt with first, and data copy as well as message passing process were synchronized with inner cell computation, through which the time latency due to data copy and message passing was hidden efficiently.

In our implementation, we use both hybrid MPI-CUDA and Pthread-CUDA parallelization on a shared memory AMAX machine with 2× Intel Xeon Westmere 5675 3.06 GHz six-core CPUs, connected via PCIE2 slot to 8× NVIDIA Tesla C2075 Fermi GPUs. Each computational grid point is mapped to a GPU thread, and appropriate data structure is adopted to exploit the high memory bandwidth of GPU. We design a special procedure including atom operator to synchronize thread blocks. This make reduction operations like *min* or *max* execute completely inside GPU without exiting a kernel, thus eliminating numerous kernel-launching overheads. Besides, we use domain decomposition method for multi-GPU computing. The computation of every subdomain is assigned to a device, while the communication between devices is performed through MPI or Pthread. Both modes are compared in numerical tests.