ParCFD'2020
32^{nd} International Conference on Parallel Computational Fluid Dynamics
May-11-13 2020, Nice FRANCE

GPU-BASED CELL-CENTERED AMR FOR TWO-PHASE FLOW SIMULATION USING AN ITERATIVE PRESSURE-EVOLUTION PROJECTION METHOD

KAI YANG*, **TAKAYUKI AOKI**†

* Tokyo Institute of Technology
Department of Mechanical Engineering
2-12-1 i7-3 O-okayama, Meguro-ku, Tokyo 152-8550, Japan
e-mail: yang@sim.gsic.titech.ac.jp

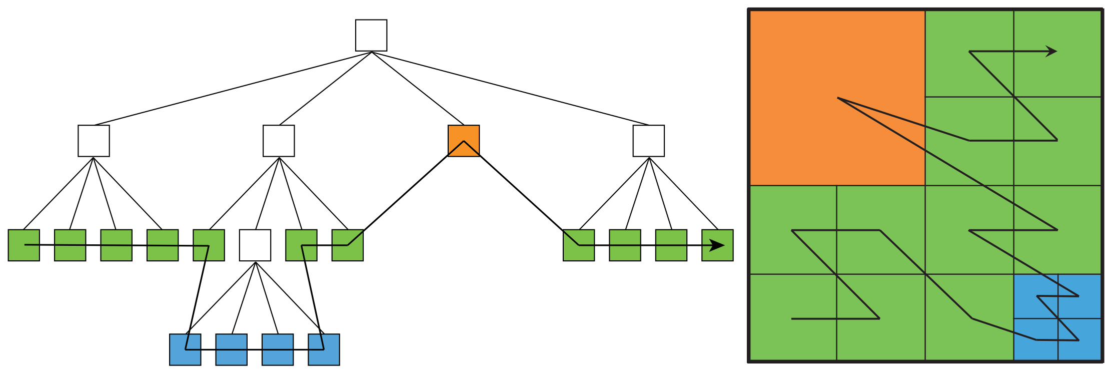
† Tokyo Institute of Technology
Global Scientific Information and Computing Center
2-12-1 i7-3 O-okayama, Meguro-ku, Tokyo 152-8550, Japan
e-mail: taoki@gsic.titech.ac.jp

Key words: Two-phase flow, Pressure evolution, Collocated grid, Adaptive mesh refinement, GPU

Summary. We propose an iterative pressure-evolution projection method to solve incompressible Navier-Stokes equations, taking the advantage of hyperbolic-parabolic system of equations on explicit time integration. Volume of fluid (VOF) method is coupled to capture the interface of two-phase flow. By employing a cell-centered collocated grid for discretization, the transports of mass and momentum are ensured to be consistent in finite volume formulation. Moreover, a parallel algorithm of tree-based block-structured adaptive mesh refinement (AMR) is developed for GPU computing, with special considerations on the memory management and conservation of interpolation for cell-centered data. Droplet splashing on thin liquid film and dam breaking on a wet bed are simulated to demonstrate the capability of our solver.

1 INTRODUCTION

In recent years, the use of GPU as computational accelerator has attracted wide attention. Along with the progress of computer technology like GPU, the study on implementation of complicated algorithms on advanced architecture of hardware is demanded. In the simulation of fluid flow, wide range of spatial scales are often encountered such as the interfaces between immiscible fluids. A straightforward solution is to adjust the mesh resolution to follow the evolution of flow structure, which is the so-called adaptive mesh refinement [1]. On the other hand, the scalability of numerical methods is essential to make the best use of the increasing computing performance of modern computers. Due to the elliptic-parabolic characteristic of incompressible Navier-Stokes equations, there is a considerable complexity for numerical computation caused by fractional-step semi-implicit methods [2]. Moreover, this kind of algorithms has a limited scalability for distributed parallel computing.



(a) Quadtrees (b) Space filling curve

Figure 1: Tree-based block-structured AMR.

2 NUMERICAL METHODS

In this work, a Navier-Stokes solver based on pressure evolution equation is presented. This hyperbolic equation for pressure is derived under the low-Mach number and isothermal conditions [3], and it can be explicitly integrated in time by a local spatial stencil. With the purpose to damp the acoustic wave and suppress oscillating solutions, we iteratively compute the pressure evolution equation coupled with a projection step to correct velocities. Due to the simplicity and locality of this iteration procedure, it does not increase the amount of computation too much.

Volume of fluid (VOF) method is combined with Navier-Stokes equations to capture the interface between two fluids. The system of equations in conservative form is discretized by finite volume method. A consistent transportation scheme for mass and momentum is carefully implemented, where the flux of volume fraction is evaluated by THINC/WLIC scheme [4] and the MUSCL scheme is employed for reconstruction of velocity. Time derivative is integrated by third-order strong-stability-preserving Runge-Kutta scheme. To avoid pressure-velocity decoupling when the pressure projection is performed on collocated grid, face-centered velocities along face normal are first linearly interpolated from neighboring cells, then corrected to divergence-free state by the pressure projection.

3 ADAPTIVE MESH REFINEMENT

The computational domain is decomposed into blocks of different sizes in AMR, where each block is a cube composed of grid cells. A data structure based on the octrees (quadtrees in 2D) is implemented, as shown in Fig. 1(a). The effective blocks are actually the leaves in the forest of octrees. They are stored in the order following a space filling curve manner. The computational grid corresponding to the octrees is illustrated in 1(b). It is reasonable to assign one CUDA block of threads to process one AMR block.

When a block is to be refined, every cell in this block will be subdivided into 4 or 8 smaller cells. Data in these fresh cells should be set artificially. In this work, volumetric formulation for the cell-centered data is adopted, hence a linear interpolation scheme that

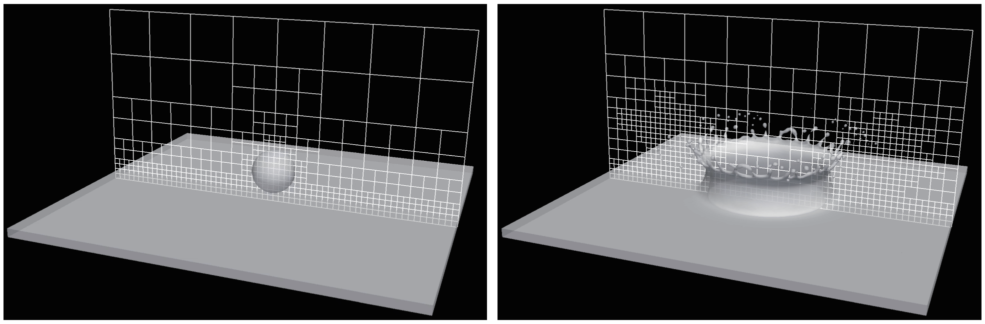


Figure 2: Time evolution of liquid surface in droplet splashing on liquid film.

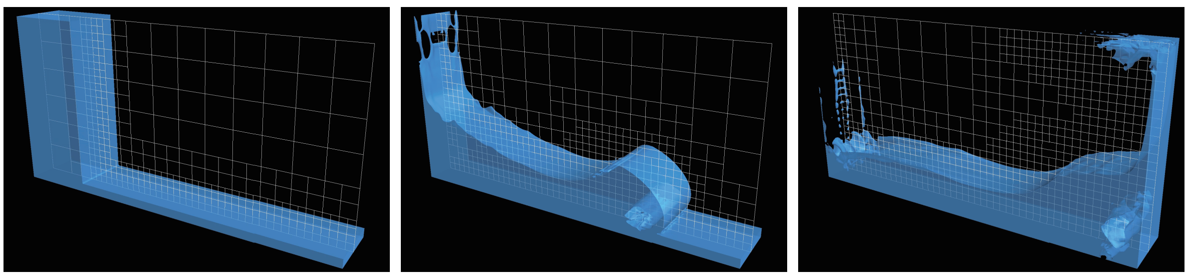


Figure 3: Time evolution of water surface in 3D dam break on a wet bed.

satisfies the conservation law can be constructed based on the local gradient of data.

4 RESULTS AND DISCUSSIONS

Various benchmarking problems including both single-phase and two-phase flows have been calculated to validate our methods. Hereafter, the stability of present method as a practical two-phase flow solver is demonstrated by several violent two-phase flows. Firstly we study the droplet splashing on a thin liquid film, which is also known as milk crown. The calculation conditions are as follows: a thin liquid film of 0.9 mm is set in a computational domain of 4 cm $\times$ 4 cm $\times$ 2 cm, and droplet of 4.8 mm diameter is collided at a speed of 2.8 m/s. The results on the time evolution of liquid surface are depicted in Fig. 2. The mesh is dynamically adapted to follow the motion of interface.

Then a three-dimensional dam break on a wet bed is considered. The water tank studied here has a size of 0.72 m $\times$ 0.12 m $\times$ 0.36 m. A water column is initially enclosed by a virtual baffle plate located at $x = 0.15$ m. The depth of water on the flat bed is 0.018 m. The water column begins to collapse driven by gravity, and crashes against the downstream wall, as shown in Fig. 3. Overall computation for the two problems has succeeded with a high stability.

5 CONCLUSIONS

Based on the cell-centered collocated data layout, a parallel two-phase flow solver with adaptive mesh refinement on GPU is successfully developed. The consistent mass and momentum transport scheme is implemented during the evaluation of flux of finite volume method. It enables our solver for robust computation of two-phase flow with extremely large density ratio. In addition, the iterative pressure-evolution projection method proposed by us has been proved to be effective to solve incompressible Navier-Stokes equations. Further studies on computational efficiency and extension of present solver to multi-GPU cluster are expected.

ACKNOWLEDGMENTS

This research was partly supported by KAKENHI, Grant-in-Aid for Scientific Research (S) 26220002 and 19H05613 from Japan Society for the Promotion Science (JSPS), and Joint Usage/Research Center for Interdisciplinary Large-scale Information Infrastructures (JHPCN) jh180035, jh190054 and High Performance Computing Infrastructure (HPCI) hp190130 in Japan. The authors thank the Global Scientific Information and Computing Center, Tokyo Institute of Technology for use of the computer resources of the TSUBAME 3.0 supercomputer and Kyushu University for use of the computer resources of the ITO supercomputer.

REFERENCES

[1] Stéphane Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of Computational Physics*, 190(2):572–600, 2003.

[2] Sanghyun Ha, Junshin Park, and Donghyun You. A gpu-accelerated semi-implicit fractional-step method for numerical solutions of incompressible navier–stokes equations. *Journal of Computational Physics*, 352:246–264, 2018.

[3] Adrien Toutant. General and exact pressure evolution equation. *Physics Letters A*, 381(44):3739–3742, 2017.

[4] Kensuke Yokoi. Efficient implementation of thinc scheme: a simple and practical smoothed vof algorithm. *Journal of Computational Physics*, 226(2):1985–2002, 2007.