Water Entry Simulation by a Lattice Boltzmann Method

Xuhui Li[1,2], Changhong Hu[2*], David Le Touze[1]

[1]Ecole Centrale de Nantes, LHEEA lab, UMR CNRS 6598, Nantes, France

[2]Research Institute for Applied Mechanics, Kyushu University, Fukuoka, Japan

*hu@riam.kyushu-u.ac.jp

1. INTRODUCTION

With the rapid development of the performance of modern computer hardware, computational fluid dynamics (CFD) has become one of the important numerical tools for nonlinear hydrodynamics problems. However, numerical simulation of strongly nonlinear wave-body interactions is still a big challenge for CFD. Unconventional CFD approaches are expected to tackle such problems. The lattice Boltzmann method (LBM) has been increasing popular in simulating free surface flows for its excellent execution efficiency, implementation simplicity, and intrinsic parallelism[1)]. Particularly, LBM can achieve extremely high computation efficiency when it is accelerated by using GPU (General Purpose Graphics Processor Units). The authors have been devoting to apply LBM to deal with the nonlinear hydrodynamics problems. An efficient GPU accelerated LBM solver has been developed and applied to simulation of a water entry problem, which is a benchmark problem for validation of nonlinear free surface solvers. Present simulation results demonstrate that the LBM solver is as accurate as the other Navier-Stokes solvers but much more efficient.

2. NUMERICAL METHOD

2. 1 LBM Model

The standard D2Q9 MRT model[2)] is adopted to conduct the collision process in the lattice Boltzmann evolution, which can be written as

$$|f_i(\mathbf{x}+\mathbf{e}_i\delta t, t+\delta t)\rangle - |f_i(\mathbf{x},t)\rangle = -M^{-1}S[|m_i(\mathbf{x},t)\rangle - |m_i^{eq}(\mathbf{x},t)\rangle] \tag{1}$$

The moment space $\mathfrak{M}$ and the discrete velocity space $\mathbb{F}$ is linked by the linear transformation M which maps a vector $|f\rangle$ in $\mathbb{F}$ to a vector $|m\rangle$ in $\mathfrak{M}$, which is as follows[)].

$$|m\rangle = M|f\rangle \quad \text{and} \quad |f\rangle = M^{-1}|m\rangle \tag{2}$$

The vector of moment of particle distribution function can be expressed as:

$$|m_i(\mathbf{x},t)\rangle = [\rho, e, \varepsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy}]^T \tag{3}$$

The relaxation matrix S is diagonal in the moment space $\mathfrak{M}$:

$$S = diag(0, s_e, s_\varepsilon, 0, s_q, 0, s_q, s_\nu, s_\nu) \tag{4}$$

where $1/s_\nu = 3\nu + 0.5$.

2. 2 Multi-Phase Treatment

In this study, the algorithm proposed in the literature[3)] is applied to capture the free surface flow. The algorithm uses the volume of fluid (VOF) to track the interface motion. Two values, the mass *m* and the fluid fraction ε are introduced and their relationship is as follows:

$$\varepsilon = m / \rho \tag{5}$$

The mass evolution is more direct than the treatments in the traditional finite differential method or finite volume method. The change of mass is directly computed from the particle distribution functions which are streamed between the neighborhood cells for every direction, and have the following relation:

$$\Delta m_i(\mathbf{x}, t+\Delta t) = \frac{1}{2}[\varepsilon(\mathbf{x}+\mathbf{e}_i\Delta t, t) + \varepsilon(\mathbf{x}, t)] \times [f_{\tilde{i}}(\mathbf{x}+\mathbf{e}_i\Delta t, t) - f_i(\mathbf{x}, t)] \tag{6}$$

where $\tilde{i}$ is the inverse direction of *i*. The mass at the new time step at the cell center can then be obtained by adding up the mass change for all directions.

$$m(\mathbf{x}, t+\Delta t) = m(\mathbf{x}, t) + \sum_{i=1}^{9} \Delta m_i(\mathbf{x}, t+\Delta t) \tag{7}$$

The distribution functions which will stream from the gas to the fluid domain are reconstructed according to the dynamical free surface boundary condition:

$$f_{\tilde{i}}^{'}(\mathbf{x}, t+\Delta t) = f_i^{eq}(\rho_A, \mathbf{u}) + f_{\tilde{i}}^{eq}(\rho_A, \mathbf{u}) - f_i(\mathbf{x}, t) \tag{8}$$

After all the cells are updated, the flag that identifies the cell type should be updated timely. The interface cell which is filled or emptied at previous step should be converted into the fluid or gas cell, meanwhile the gas cell neighboring to a previous filled cell should be changed into the interface cell and the fluid cell neighboring to a previous emptied cell should be converted into the interface cell. The mass in each cell should be adjusted to satisfy $0 \le \varepsilon \le 1$, in which the excess mass needs to be distributed to the neighboring interface cells. In terms of free surface penetrating by the solid body, the non-slip boundary condition should be applied in the interface between solid and fluid. The flags for fluid cell, interface cell, gas cell and solid cell should be updated when the obstacle moves to a new position.

2. 3 Force Calculation

In the present solver, the stress integration method proposed by Inamuro et al.[4)] is applied to evaluate the force and torque exerted on the solid body. The stress tensor can be evaluated through the distribution function on the boundary directly instead of calculating the velocity gradient:

$$\sigma_{ij} = -\frac{1}{6\tau}\rho\delta_{ij} - (1-\frac{1}{2\tau})\sum_{\alpha}(\mathbf{e}_{\alpha i} - u_i)(\mathbf{e}_{\alpha j} - u_j) f_\alpha \tag{9}$$

Then the force and torque exerted on the boundary can be integrated as

$$\mathbf{F} = \int_{\partial\Omega} \{\boldsymbol{\sigma}\cdot\mathbf{n} - \rho\mathbf{u}[(\mathbf{u}-\mathbf{U})\cdot\mathbf{n}]\} ds \tag{10}$$

$$\mathbf{T} = \int_{\partial\Omega} \mathbf{r} \times \{\boldsymbol{\sigma}\cdot\mathbf{n} - \rho\mathbf{u}[(\mathbf{u}-\mathbf{U})\cdot\mathbf{n}]\} ds \tag{11}$$

where $\mathbf{u}$ is the velocity of the Lagrange force point on the body surface, which can be evaluated by extrapolation of f_α. The time marching of the body motion is performed by an Euler explicit scheme.

3. GPU IMPLEMMENTATION

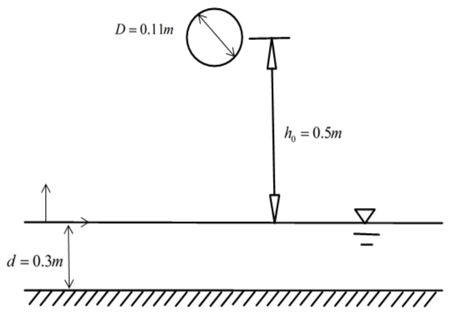
Graphics Processing Unit (GPU) provides possibilities of significant cost savings for CFD simulations. Since LBM is an explicit algorithm that needs only nearest neighbor information, the method allows a highly efficient parallel implementation using GPU architecture. The present GPU solver has been modified to be able to run on multiple GPUs. The details of the GPU computing technique is not elaborated here for the length of paper. Generally, for the grid mapping in the CUDA programming environment, one fluid node is distributed to one thread. The memories on the device are allocated in one dimensional way and the index of the array is given as *j*xDim+i* (*xDim* is the number of nodes in x dimension). The indexes of the fluid nodes in x and y directions can be expressed by the indexes of the threads and blocks explicitly as:

$$j=blockIdx.y*blockDim.y+threadIdx.y$$
$$i=blockIdx.x*blockDim.x+threadIdx.x \qquad (13)$$

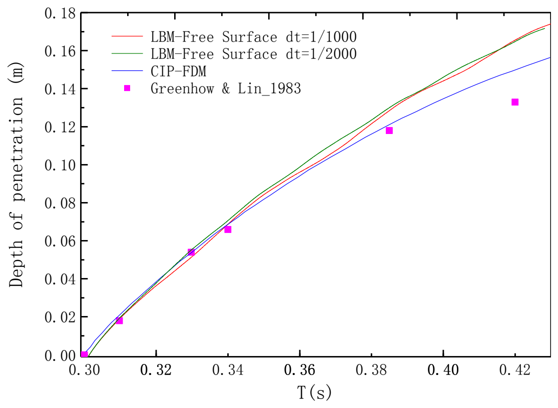
The present GPU accelerated LBM solver has been demonstrated to be much more efficient by comparing to other conventional N-S solver.

4. RESULTS

Verification of the present GPU accelerated LBM free surface solver is conducted on a benchmark experiment of water entry with a circular cylinder as shown in Fig.1 a). This problem is taken from an experimental study[5)]. The density ratio of the cylinder and the water is 0.5. The cylinder diameter is 0.11 m. In the experiment, the cylinder falls freely with an initial height h_0=0.5 m, which is the distance between the initial position of the cylinder center and the still free surface.



a) Initial state



b) Penetration depth

Fig.1 2-D water entry simulation of a circular cylinder

Fig.1 b) shows the results for the penetration depth of the cylinder. Fairly good agreements are obtained among the present LBM solver, the CIP-based FDM solver and experimental data. A pressure oscillation suppression method, which is developed in our previous work[6)], has been applied in the computation.

Fig.2 is the result for free surface deformation during water entry of a half-buoyant circular cylinder, in which the present LBM result, the CIP-based FDM result and the experiment are compared. When the cylinder hits the free surface, two sprays form. It can be found that both the LBM result and the FDM result agree well with the experiment. The CPU time is 20 seconds for the GPU accelerated LBM simulation, and 5 hours for the FDM simulation.

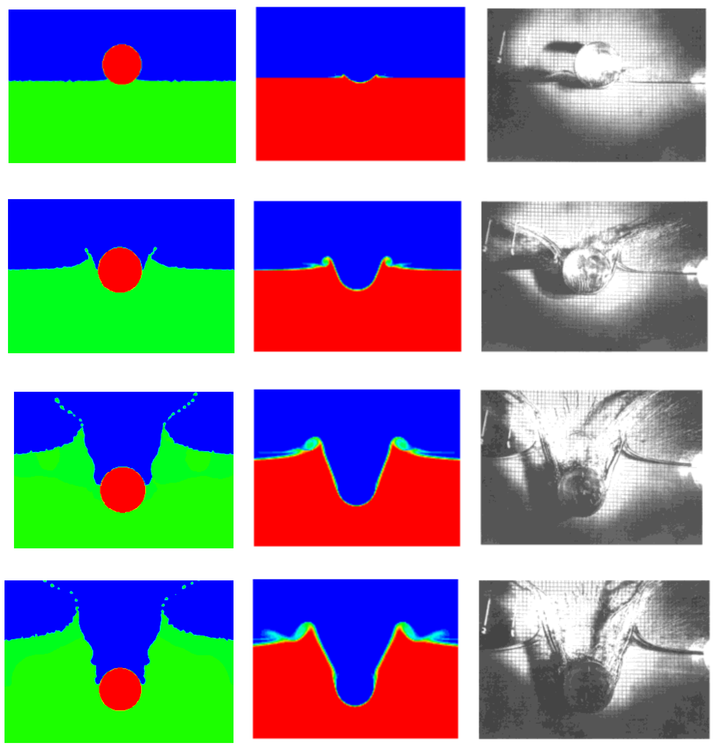


Fig.2 Free surface deformation at *t=0.305s, 0.330s, 0.385s, 0.420s* obtained by LBM (left), FDM (center) and experiment (right).

5. CONCLUSION

A GPU accelerated LBM solver has been developed for numerical simulation of strongly nonlinear wave-body interactions. The performance of the present LBM code is demonstrated by numerical simulation of a water-entry problem with a circular cylinder. The present LBM result is compared to the CIP based FDM result and the experiment. The simulation accuracy of the present LBM solver is as good as the CIP based FDM solver while the computational time can be significantly reduced.

REFERENCES

1) C. K. Aidun *et.al.*: Lattice-Boltzmann method for complex flows, Annual Review of Fluid Mechanics, Vol 42, pp.439-472, 2010.

2) P. Lallemand, L.S. Luo: Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability, Physical Review E, Vol. 61, No. 6, pp.6546-6562, 2000.

3) S. Bogner, U. Rude: Simulation of floating bodies with the lattice Boltzmann method, Computers and Mathematics with Applications, Vol. 65, pp. 901-913, 2013.

4) T. Inamuro et al.: Flow between parallel walls containing the lines of neutrally buoyant circular cylinders, International Journal of Multiphase Flow, Vol. 26, pp.1981-2004, 2000.

5) M. Greenhow and W.M. Lin: Nonlinear free surface effects: Experiments and theory, Report No. 83-19 Department of Ocean Engineering, MIT, 1983.

6) X. Li, F. Jiang, C. Hu: Analysis of the accuracy and pressure oscillation of the lattice Boltzmann method for fluid-solid interactions, Computers and Fluids, Vol. 129, pp.33-52, 2016.