Progress in NUCLEAR SCIENCE and TECHNOLOGY, Vol. 2, pp.491-497 (2011)

ARTICLE

Multiphase Fluid Simulations on a Multiple GPGPU PC Using Unsplit Time Integration VSIAM3

Akio IKEBATA[1,*], Yoshihiko MURAOKA[1] and Feng XIAO[2]

[1] *TOTO LTD., 2-1-1, Nakashima, Kokurakita-ku, Kitakyushu 802-8601, Japan*
[2] *Tokyo Institute of Technology, 4259 Nagatsuta, Midori-ku, Yokohama, 226-8502, Japan*

This talk presents the implementation of simulations on multiphase fluid dynamics on hardware of multiple GPGPU architecture by using robust and efficient numerical methods. An unsplit formulation for the advection computation is proposed to take the place of the original split formulation in the so-called VSIAM3 method. The new formulation improves dimensional symmetry of numerical results without losing the advantages of the original formulation. We also devised a numerical scheme named STAA to capture the moving interface between different fluids. The volume of fluid function is transported by the above advection scheme, and then modified by an anti-diffusion step based on a well designed numerical flux that rigorously conserves the mass and effectively eliminate the numerical smearing around the interface. Our experiments show that it is able to maintain the VOF function in a well-regulated shape with compact transition layer between different fluids. We have developed a 3D numerical model for multi-fluid simulations with above methods incorporated, and implemented it on a multiple GPGPU computer. It is shown that the present model is able to adequately make use of the advantage of the GPGPU architecture. Through this research, we have built a numerical tool with adequate reliability for sanitary ware product design. Computational efficiency which is of particular importance for real-case design process is achieved by implementing the code on a multiple GPGPU hardware.

KEYWORDS: multiphase flow, numerical method, finite volume method, CIP, VSIAM3, GPGPU, OpenCL

I. Introduction

Tremendous efforts for developing numerical simulation technology for gas and liquid multiphase fluid have been dedicated for a long time. In the so-called one-fluid model, the interfaces between different fluids are explicitly solved by the interface capturing (or tracking) schemes. Among the representative ones of this category, VOF (volume of fluid) and level set methods have been suggested to identify different phases.[1–4] These methods transport the identification functions through the advection equations based on the velocity field computed from the Navier-Stokes equations. In practical applications, interface capturing schemes of both numerical accuracy and computational efficiency are of much interests. Following the recent trend in the expanding use of the new type hardware architectures, such as the GPGPU, numerical schemes that are well suited for multi-thread processing are particularly demanded. Level set method is suitable for GPGPU because the calculation of the scheme can be constructed as the assembly of simple vector processing. However, the mass conservation is not inherently guaranteed in the level set method. As a consequence, small liquid drops and bubbles tend to be lost. At this point, the VOF method that exactly conserves the fluid volume is more attractive. However, the geometrical reconstruction procedure in the conventional VOF method has algorithmic complication which prevents it from being well oriented for the GPU processing.

In this work, we propose a new advection scheme using multi-moment concept which is very suitable for GPU computing. Different from the conventional VOF method, an interface capturing scheme that doesn't require geometrical reconstruction has been also suggested. An improved VSIAM3 model for multiphase simulation has been developed and implemented on a multiple GPU computer system. An integrated simulation tool which is able to use the advantage of GPU acceleration has been established for practical applications in product design process.

II. The Advection Transport Scheme

We consider the following two-dimensional advection equation

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = 0. \qquad (1)$$

It can be alternatively written as

$$\frac{\partial \phi}{\partial t} + \frac{\partial u\phi}{\partial x} + \frac{\partial v\phi}{\partial y} = \phi\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right). \qquad (2)$$

For incompressible flow, it is obvious that the conservation of scalar value ϕ is guaranteed if Eq. (2) is solved by the finite volume method (FVM).

The CIP-CSL (Constrained Interpolation Profile-Conservative Semi-Lagrangian) schemes are a class of

[*]Corresponding author, E-mail:akio.ikebata@jp.toto.com

conservative advection schemes of high numerical accuracy and computational efficiency.[5–9)] The most remarkable feature of a CIP-CSL scheme which distinguishes it from the existing methods is using multi-moments and treating them simultaneously as the computational variables. When implementing the CIP-CSL schemes in multi-dimensions, dimensional splitting is usually used to get around the complexities of integrating the piece-wisely constructed interpolation functions over distorted volume. An economical splitting is introduced in the study to simplify the algorithm that uses all moments of three dimensions.[7–9)] The simplified scheme makes use of only two types of moments, i.e. the volume integrated average (VIA) and surface integrated average (SIA) for discretizations in three dimensions. The resultant numerical formulations are called VSIAM3 (Volume/Surface Integrated Average based Multi-Moment Method). In two dimensional context, we define two kinds of moments, VIA and SIA as follows,

$$\overline{^{V}\phi}_{ij} \equiv \frac{1}{\Delta x \Delta y}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}}\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \phi(x,y)dxdy \tag{3}$$

and

$$\overline{^{S^x}\phi}_{i+\frac{1}{2}j} \equiv \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \phi(x_{i+\frac{1}{2}},y)dy, \tag{4}$$

$$\overline{^{S^y}\phi}_{ij+\frac{1}{2}} \equiv \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \phi(x,y_{j+\frac{1}{2}})dy. \tag{5}$$

Instead of the splitting in the existing VSIAM3, we propose here an alternative to update the SIA moment and numerical fluxes in an unsplit manner.

In the unsplit method, SIA is calculated by semi-Lagrangian scheme based on Eq. (1), while VIA is updated with Eq. (2) by a finite volume formulation over the control volume (mesh element).

As shown in **Fig. 1** we approximate the numerical fluxes by the Gaussian quadrature. The flux on right face of the control volume, for example, is calculated by

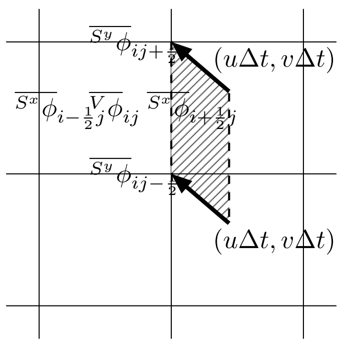
$$\begin{aligned}\mathcal{F}_{x_{i+\frac{1}{2}j}} =& \frac{u\Delta t}{2}\left\{\overline{^{S^x}\phi}^{n}(x_{i+\frac{1}{2}}-g_1u\Delta t, y_j - g_1 v\Delta t)\right. \\ &\left. +\overline{^{S^x}\phi}^{n}(x_{i+\frac{1}{2}}-g_2u\Delta t, y_j - g_2 v\Delta t)\right\},\end{aligned} \tag{6}$$

where $g_1 \equiv (3-\sqrt{3})/6$ and $g_2 \equiv (3+\sqrt{3})/6$. In the same way, the flux on bottom face of the control volume can be calculated by

$$\begin{aligned}\mathcal{F}_{y_{ij+\frac{1}{2}}} =& \frac{v\Delta t}{2}\left\{\overline{^{S^y}\phi}^{n}(x_i - g_1u\Delta t, y_{j+\frac{1}{2}} - g_1 v\Delta t)\right. \\ &\left. +\overline{^{S^y}\phi}^{n}(x_i - g_2u\Delta t, y_{j+\frac{1}{2}} - g_2 v\Delta t)\right\}.\end{aligned} \tag{7}$$

The VIA is then updated by

$$\begin{aligned}\overline{^{V}\phi}^{n+1}_{ij} =& \overline{^{V}\phi}^{n}_{ij} - \left(\frac{\mathcal{F}_{x_{i+\frac{1}{2}j}} - \mathcal{F}_{x_{i-\frac{1}{2}j}}}{\Delta x} + \frac{\mathcal{F}_{y_{ij+\frac{1}{2}}} - \mathcal{F}_{y_{ij-\frac{1}{2}}}}{\Delta y}\right) \\ &+ \Delta t \overline{^{V}\phi}^{n}_{ij}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)_{ij}.\end{aligned} \tag{8}$$



$\overline{^{S^y}\phi}_{ij+\frac{1}{2}}$
$(u\Delta t, v\Delta t)$
$\overline{^{S^x}\phi}_{i-\frac{1}{2}j}$
$\overline{^{V}\phi}_{ij}$
$\overline{^{S^x}\phi}_{i+\frac{1}{2}j}$
$\overline{^{S^y}\phi}_{ij-\frac{1}{2}}$
$(u\Delta t, v\Delta t)$

Fig. 1 X axis direction flux

On the other hand, SIA is updated by

$$\overline{^{S^x}\phi}^{n+1}_{i+\frac{1}{2}j} = \overline{^{S^x}\phi}^{n}(x_{i+\frac{1}{2}} - u\Delta t, y_j - v\Delta t) \tag{9}$$

$$\overline{^{S^y}\phi}^{n+1}_{ij+\frac{1}{2}} = \overline{^{S^y}\phi}^{n}(x_i - u\Delta t, y_{j+\frac{1}{2}} - v\Delta t)\,. \tag{10}$$

The key issue to compute Eqs. (6), (7), (9) and (10) is to properly update the numerical solutions for SIA variables. Without losing the generality, we assume a locally frozen velocity and denote the effective velocity components by u' and v'. The solutions of $\overline{^{S^x}\phi}^{n}(x_{i+\frac{1}{2}} - u'\Delta t, y_j - v'\Delta t)$ and $\overline{^{S^y}\phi}^{n}(x_i - u'\Delta t, y_{j+\frac{1}{2}} - v'\Delta t)$ can be computed by the following two-step procedure as shown in **Fig. 2**

Step 1 In x direction, the intermediate values for SIA $\overline{^{S^x}\phi}^{*}(x_{i+\frac{1}{2}} - u'\Delta t, y_j)$ and flux $\mathcal{F}^{*}_{x}(x_{i+\frac{1}{2}} - u'\Delta t, y_j)$ are calculated by one dimensional CIP-CSL scheme. In the same manner, those in y direction, $\overline{^{S^y}\phi}^{*}(x_i, y_{j+\frac{1}{2}} - v'\Delta t)$ and $\mathcal{F}^{*}_{y}(x_i, y_{j+\frac{1}{2}} - v'\Delta t)$, are calculated.

Step 2 From the intermediate results obtained at step 1, we update the SIA value by,

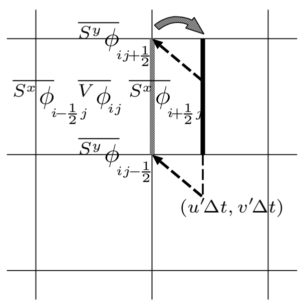
$$\begin{aligned}&\overline{^{S^x}\phi}^{n}(x_{i+\frac{1}{2}} - u'\Delta t, y_j - v'\Delta t) \\ &= \overline{^{S^x}\phi}^{*}(x_{i+\frac{1}{2}} - u'\Delta t, y_j) - \Delta t\left(v'\frac{\partial\phi}{\partial y}\right) \\ &= \overline{^{S^x}\phi}^{*}(x_{i+\frac{1}{2}} - u'\Delta t, y_j) - \Delta t\left\{\frac{\partial v'\phi}{\partial y} - \phi\left(\frac{\partial v'}{\partial y}\right)\right\}\end{aligned} \tag{11}$$

The flux gradient in y direction on the right hand side of (11) can be approximated by a linear interpolation in terms of the intermediate flux function at step 1 as,

$$\frac{\partial v'\phi}{\partial y} = \frac{F^{+} - F^{-}}{\Delta y} \tag{12}$$

with

$$\begin{aligned}F^{+} \equiv& \left(\frac{1}{2} + \xi\right)\mathcal{F}^{*}_{y}(x_i, y_{j+\frac{1}{2}} - v'\Delta t) \\ &+ \left(\frac{1}{2} - \xi\right)\mathcal{F}^{*}_{y}(x_{i+1}, y_{j+\frac{1}{2}} - v'\Delta t)\end{aligned} \tag{13}$$



$\overline{^{S^y}\phi}_{ij+\frac{1}{2}}$
$\overline{^{S^x}\phi}_{i-\frac{1}{2}j}$
$\overline{^{V}\phi}_{ij}$
$\overline{^{S^x}\phi}_{i+\frac{1}{2}j}$
$\overline{^{S^y}\phi}_{ij-\frac{1}{2}}$
$(u'\Delta t, v'\Delta t)$

Fig. 2 Two step SIA calculation method, STEP1(left) and STEP2(right).

$$F^- \equiv \left(\frac{1}{2}+\xi\right)\mathcal{F}^*_y(x_i, y_{j-\frac{1}{2}} - v'\Delta t) + \left(\frac{1}{2}-\xi\right)\mathcal{F}^*_y(x_{i+1}, y_{j-\frac{1}{2}} - v'\Delta t) \tag{14}$$

where $\xi = u'\Delta t/\Delta x$. The velocity gradient in y direction on the right hand side of Eq. (11) is approximated by central difference method. In the same manner, SIA $\overline{^{S^y}\phi}^n(x_i - u'\Delta t, y_{j+\frac{1}{2}} - v'\Delta t)$ can be updated.

By repeating above procedure, fluxes can be calculated by Eqs. (6) and (7) with the effective velocity ($u' = g_{1,2}u$ and $v' = g_{1,2}v$), and thus the VIA is updated by Eq. (8). The SIA is updated by letting $u' = u$ and $v' = v$.

This scheme can be straightforwardly extended to three dimensions. To distinguish the present method from the existing VSIAM3 algorithm, we call this scheme "UTI-VSIAM3" (Unsplit Time Integration VSIAM3). Because of unsplit time integration, the dimensional symmetry is much improved by the present scheme. We use UTI-VSIAM3 as the transport scheme for both moving interface between different fluids and the fluid dynamic solver.

III. Numerical Tests

We tested the present scheme by solving 2D advection problem. The advection velocity u and v are specified to give a solid rotation around the center of computational domain with a mesh of 75×75. We specified the initial advected profile being centered over the computational domain and having a shape of letter "T" (for TOTO as the name of our company) with discontinuous jumps between 0 and 1.

The result of UTI-VSIAM3 under cfl=0.3 (775 cycle per rotation) is shown in **Fig. 3**. It is obvious that the present scheme is more accurate than the conventional MUSCL scheme, and is comparable to the original VSIAM3. Moreover, without the directional sweep, the present scheme guarantees the dimensional symmetry, which provides a great convenience in optimizing the partitioning for parallel and GPU processing.

It should be noted that the UTI-VSIAM3 also suffers from numerical diffusion as other Eulerian type schemes. Thus, extra numerical manipulation will be needed to reduce the smearing of the transition layer of the moving interface. In

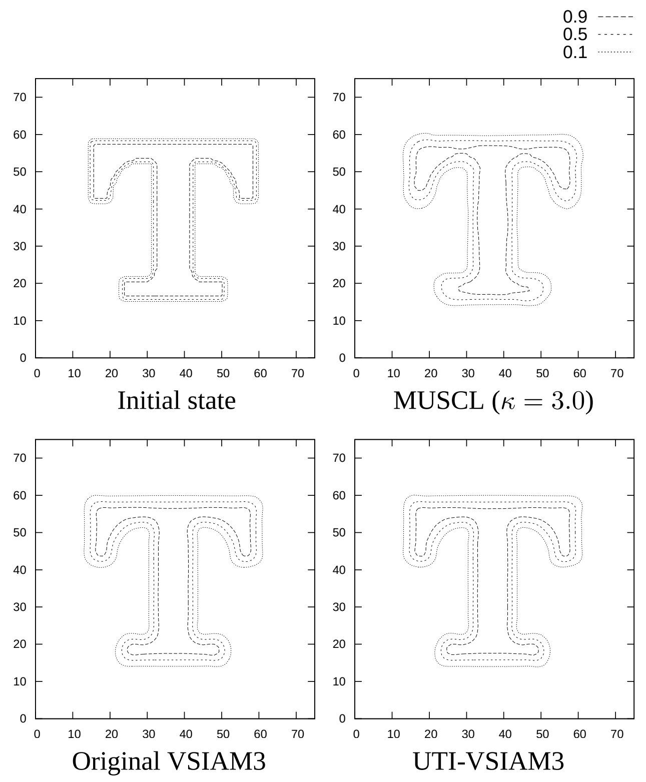


Fig. 3 Results of 2D advection test with various schemes (after 1 rotation).

the present model, we make use of an efficient anti-diffusion method which is to be presented in the next section.[11)]

IV. Correction of Numerical Diffusion

After the calculation of $\partial F/\partial t + (\mathbf{u}\cdot\nabla)F = 0$, where F denotes VOF function, we make modifications to reduce the numerical diffusion at interface between liquid ($F = 1$) and gas ($F = 0$).

Our approach consists of two steps. At first step, we construct a level set function ϕ, that is a signed distance function from the interface based on the VOF function F. As the second step, the cells where $|\phi| > \epsilon$ are corrected to $F = 0$ or $F = 1$ by an effective anti-diffusion formulation of flux form so that the conservation of F is exactly guaranteed.

In level set function calculation, initial value of ϕ must be set from F. We use the following expression,

$$\phi_0 = 2\alpha\,(F - 0.5)\,. \tag{15}$$

The level set function ϕ is then computed iteratively by

$$\phi^*_{n+1}(x,y) = \phi_n(x - C\delta_x, y - C\delta_y) \tag{16}$$

$$\phi_{n+1}(x,y) = \phi^*_{n+1}(x,y) + C\mathrm{sgn}(\phi)|\boldsymbol{\delta}| \tag{17}$$

$$\boldsymbol{\delta} \equiv \frac{\mathrm{sgn}(\phi)}{|\nabla\phi|}\nabla\phi \tag{18}$$

where $C \equiv$ CFL$\times$mesh spacing. If $\phi^*_{n+1}(x,y)\cdot\phi_n(x,y) < 0$, CFL must be reduced locally until $\phi^*_{n+1}(x,y)\cdot\phi_n(x,y) \geq 0$ is met. $\phi^*_{n+1}(x,y)$ can be interpolated by bi-linear function.

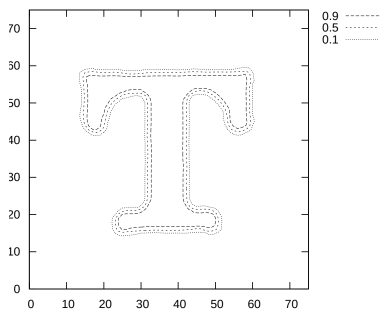


Fig. 4 Result of 2D advection test with UTI-VSIAM3 + STAA (after 1 rotation)

In the anti-diffusion flux calculation, we evaluate the amount of VOF to be redistributed by $\Delta F \equiv F - \overline{F}$ where $\overline{F}$=0 for ϕ<0 and $\overline{F}$=1 for ϕ>0. ΔF is then transported toward the interface along the opposite direction of the signed normal vector given by Eq. (18). In practice, we distribute $\Delta F_{i,j}$ to its neighboring cells $(iup, j), (i, jup), (iup, jup)$ according to the orientation of the interface, i.e. $\Delta F_{i,j}$ is tranported to $(iup, j), (iup, jup)$ if $|\delta_x|>|\delta_y|$ and to $(i, jup), (iup, jup)$ if $|\delta_x|<|\delta_y|$. The corresponding ratio for distributing $\Delta F_{i,j}$ between (iup, j) and (iup, jup) is calulated by $1 - |\delta_y|/|\delta_x|$ in the former case. In the latter case, the ratio between (i, jup) and (iup, jup) is calulated by $1 - |\delta_x|/|\delta_y|$.

The result of 2D advection test is shown in **Fig. 4**. The sharpness of interface between $F = 1$ and $F = 0$ are well maintained while the transported profile of shape "T" is faithfully reproduced. As shown in the latter numerical tests, the present scheme works well as an interface capturing scheme that has adequate numerical accuracy and robustness. This scheme is called STAA (Surface Tracking by Artificial Anti-diffusion) scheme.

As discussed above, because both UTI-VSIAM3 and STAA schemes exactly guarantee numerical conservation, the VOF function is conserved. Meanwhile the level set function ϕ which is generated from the VOF function F provides other conveniences to get the geometric quantities of the interface, like normal vector and curvature, needed in surface tension calculation. Different from the conventional VOF method, there is no geometrical reconstruction involved in the STAA method, so it is computational efficient and can be extended to 3D straightforwardly.

V. Navier Stokes Solver with Multiple GPGPU

For solving liquid and gas multiphase flows, the following Navier Stokes equation and VOF advection equation must be computed.

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u} = -\frac{1}{\rho}\nabla P + \frac{1}{\rho}\nabla : \mathbf{T} + \mathbf{g} + \mathbf{F} \quad (19)$$

$$\frac{\partial F}{\partial t} + (\mathbf{u} \cdot \nabla)F = 0 \quad (20)$$

We will calculate above equations with CIP-CUP method.[10] CIP-CUP method is based on a fractional step solution procedure. First, the left hand side of the above equations, i.e. the advection part, is solved by UTI-VSIAM3, and the resulted F is then corrected by STAA. Second, the non-advection terms except for pressure term is calculated by a method equivalent to the standard central difference. **F** includes surface tension force. It is calculated by the CSF (Continuum Surface Force) formulation.[14] In this formulation, the curvature can be easily calculated by central difference of a level set function ϕ calculated in the STAA scheme. Finally, a pressure projection step is calculated after the following Poisson equation for pressure is solved

$$\nabla \cdot \left(\frac{1}{\rho}\nabla P\right) = \frac{1}{\Delta t}\left(\frac{1}{\rho C_s^2}\frac{\partial P}{\partial t} + \nabla \cdot \mathbf{u}^*\right) \quad (21)$$

where $\mathbf{u}^*$ denotes the intermediate velocity calculated from the previous sub-steps.

We have implemented the code on a multiple GPGPU system in order to save computational time in practical applications. The multiple GPGPU system used in the present study is constructed by installing multiple GPU boards (Geforce GTX 295 2 GPUs$\times$2) into a host PC (Intel Xeon 3.2 GHz CPU).

To retain the numerical accuracy of the original code, which appears to be essential in real applications, we use double-precision data even in GPU hardware environment. It should also be notified that to make the code easy to be read and accessed by users, as well as to be maintained and continuously developed by a community, sophisticated techniques that lead to the full use of the GPU advantage, such as the optimization of the local memory is got around in this study. All these consideration result in a strategy which may compromise to some extent in computational efficiency.

Each GPGPU calculates one of the partitions divided by a domain decomposition technique. At each step, data exchange is conducted among GPUs which shares the partition boundaries of neighboring domains. Based on the current state of data transfer in the GPU architecture, the data to be exchanged among GPUs are firstly sent to the CPU memory buffers, and then transfered to the targeted GPUs. In these data transfer procedures, bottle neck due to the limited bandwidth of data pass between CPU memory and GPU memory may cause degradation of the scalability performance against the number of GPUs.

In GPGPU computations, each thread of GPU calculates its assigned element locally. The numerical algorithms used in the present model is overall suited for GPU computation. In advection phase of $\mathbf{u}$ and F, UTI-VSIAM3 can be implemented by assembly of simple vector calculations because each element value of vector array can be calculated independently. In the STAA method, level set function reconstruction can be implemented in the same way. In anti-diffusion flux calculation, however, the redistribution of $\overline{F}$ to its neighborhood may cause memory access conflicts. We solve this problem by decomposing the loop into $3 \times 3 \times 3 = 27$ sub-loops in which the redistribution of $\overline{F}$ at the cells having same number as shown in **Fig. 5** can be calculated independently by GPU threads.

Other parts except for the pressure-projection are all based on explicit algorithms, and thus can also be implemented by

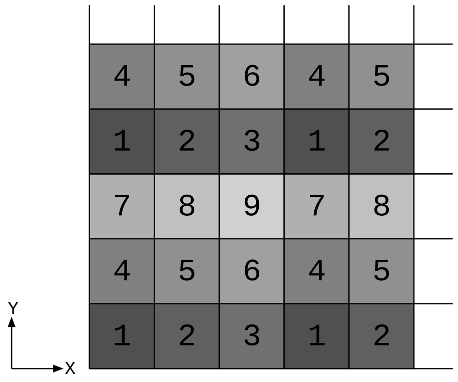


Fig. 5 Calculation order of anti-diffusion flux on STAA

Table 1 Comparison of PCG elapse time between CPU and GPU with OpenCL (cells=1,117,866, cycles=677)

Devices		Elapse	Mem. BW	Ideal BW
Xeon	1 Core	20.3 s	9.4 GB/s	32.0 GB/s
W5580	2 Core	12.4 s	15.5 GB/s	32.0 GB/s
3.2 GHz	3 Core	9.7 s	19.7 GB/s	32.0 GB/s
	4 Core	9.0 s	21.3 GB/s	32.0 GB/s
	1 GPU	2.59 s	74.1 GB/s	111.9 GB/s
Geforce	2 GPU	1.48 s	129.6 GB/s	223.8 GB/s
GTX 295	3 GPU	1.12 s	170.8 GB/s	335.7 GB/s
	4 GPU	0.95 s	201.7 GB/s	447.6 GB/s

simple vector calculations. The solution of Poisson equation of pressure, on the other hand, requires extra manipulations. Among the popular solvers for Poisson equation, preconditioned ICCG or Bi-CGSTAB are widely used due to their robustness and efficiency. However, recursive dependencies are usually involved in the preconditioner calculation. In the present study, we used PCG solver with the diagonal scaling preconditioner, and found that owing to the reduction of total floating point operations per cycle its computational time is comparable to that of the ICCG. The number of cycles in our test with ICCG was roughly 1/4 of PCG while ICCG calculation time per cycle using single CPU core was two times of PCG because of the heavy precondition calculation. Since the PCG is more parallelism oriented, we have not explored further the implementation of the ICCG on the GPU.

We implemented present code with OpenCL in order to retain compatibility among various hardwares. It is known that a code using OpenCL appears to be more complicated compared to those using CUDA Runtime API. We simplified our code by using macro definition or our own basic API package. For example, although each arguments of calling kernel function must be set with "clSetKernelArg" one by one and called with "clEnqueueNDRangeKernel", we wrapped these instructions with our API "CL_Exec", which was coded using variable argument system calls of C, "va_arg", "va_start", "va_end" etc. In the same way, we defined "CL_Put" and "CL_Get" with only four arguments which substitute the OpenCL APIs, "clEnqueueWriteBuffer" and "clEnqueueReadBuffer" that have nine arguments.

We measured the elapse time of PCG calculation as shown in **Table 1**. Generally, solution of the Poisson equation consumes the major part of computational time of the whole code. The calculation speed of 4 GPUs was almost 10 times faster than that of 4 Core CPUs. The scalability performance of 4 GPUs is almost $2/3$. The reason of the performance degradation is partly due to the communication overhead among GPUs and the vector length. Because we do not use the "overlapped communication", the performance degradation due to communication seems not negligible. Nevertheless, the present program uses multi-threads parallel computation, so the major part of total communication time among GPUs is host-device communication time. The device-to-host and host-to-device communication on the PCIExpress(x16) bandwidth in our tests is about 2GB/s although we use the "pinned memory" objects. Nevertheless, we still consider it a big

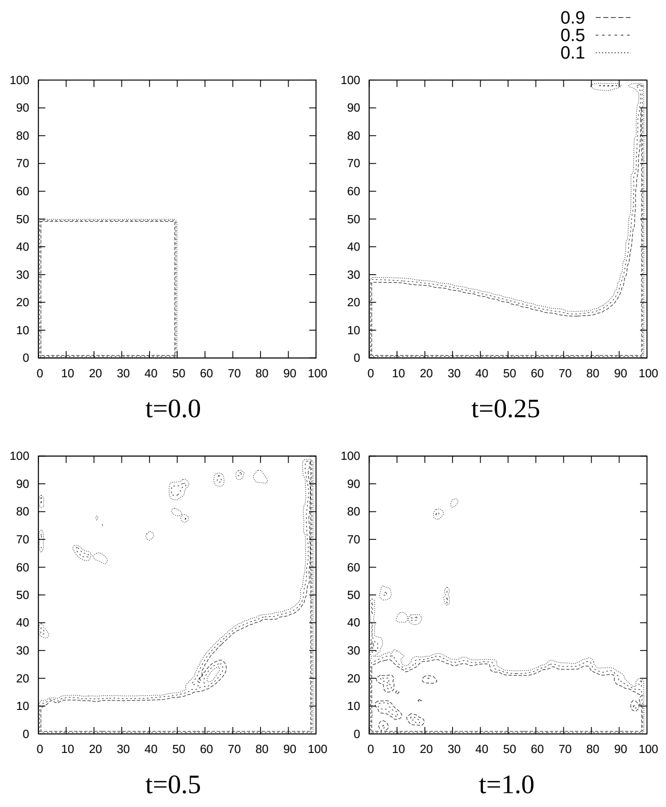


Fig. 6 Result of dam break test (X-Z plane at Y=75 mm)

gain of practical significance in terms of cost-performance improvement since a 10 time leap of the GPGPU code largely shortens the elapse time for each real-case application which usually involves number of runs in product design and optimization.

We computed the dam break test problem and show the result in **Fig. 6**. The number of cells is $100 \times 50 \times 100$ with $\Delta x = \Delta y = \Delta z = 3.0$ mm. The gravity in the opposite direction of Z axis is 9.8 m/s^2. At $t = 0.0$ second (initial condition), F of the region on $0.0 \leq x \leq 150.0$ mm and $0.0 \leq z \leq 150.0$ mm is set to 1.0 to identify the water region. The densities and viscosities of water and air are specified cor-

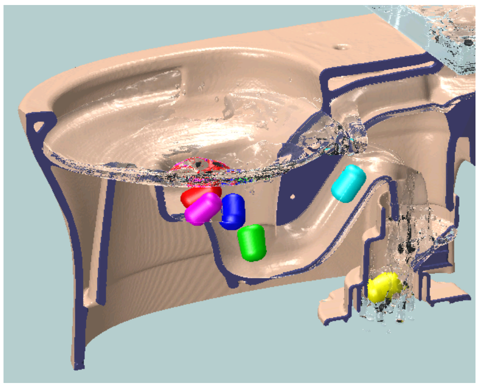


Fig. 7 Result of simulation of a sanitary ware (mesh number = $181 \times 135 \times 267$, CFL=0.48)

respondingly.

It is obvious that the transition layer between water and air has been kept with a compact thickness in the simulation, and small droplets and bubbles are well resolved. Moreover, we would like to emphasize that the calculation was quite stable from the beginning to the end.

VI. Application as a Design Tool for Products

In our company, various products like sanitary wares, kitchens and bath rooms among many others, are produced. Because many products involve water flows, simulations of water/gas/body (deformable or un-deformable) multiphase flows are important for design and improvement of our products. A few examples of applications in our product designs are shown in **Figs. 7** and **8** where the free surface and moving solid bodies are reasonably reproduced.

Our codes can simulate real-case problems which involve multiphases including arbitrarily deformable or undeformable bodies. The calculation of moving bodies is also accelerated with GPGPU computing. The moving bodies shown in Fig. 7 are calculated by method of a finite difference calculation procedure in which bodies represented by color functions are accelerated by integral of forces in each cells.[12,13] In the simulation shown in Fig. 8, the bucket motion is specified by users. The motion of the solid bodies are calculated by coordinate transformation.

As a matter of fact, the elapse times of these simulations are almost less than one or two seconds per time step with only one GPGPU system described above. Our simulation codes are very practical in respect of both the reliability of numerical outputs and the computational efficiency.

With the results of multi-fluid simulations for product design, we are able to evaluate and optimize the performances of those products before manufacturing. Using this powerful tool, we can test different designs visually, which effectively reduces the cost and speeds up the product design procedure.

VII. Conclusion

Some recent efforts are made toward the establishment of an integrated simulation tool for designing products that involve multiphase flows. We have proposed a more accurate

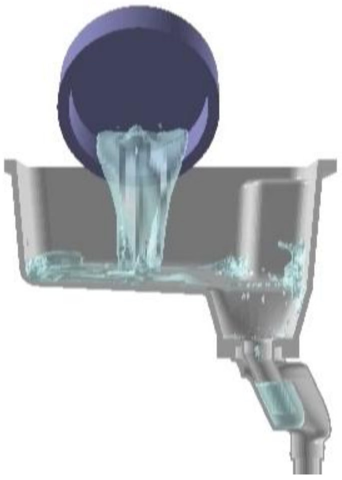


Fig. 8 Result of simulation of a kitchen (mesh number = $181 \times 223 \times 160$, CFL=0.48)

and efficient formulation for the advection computation by using the unsplit solution procedure, so-called UTI-VSIAM3, to improve the original VSIAM3. In order to compute the moving interface in multiphase flows, a numerical correction method named STAA has been incorporated with UTI-VSIAM3, which results in a conservative and accurate interface capturing scheme well suited for GPU architecture.

Owing to the simplicity of the proposed schemes, we were able to implement the multiphase flow solver on a multiple GPGPU hardware with practical computing performance and scalability. We expect the code will be used to the various applications in real-case design processes.

References

1) D. L. Youngs, "Time-dependent multi–material flow with large fluid distortion," *Numerical Methods for Fluid Dynamics*, K. W. Morton, M. J. Baines (eds.), Academic Press, New York, **24**, 273 (1982).

2) W. J. Rider, D. B. Kothe, "Reconstructing volume tracking," *J. Comput. Phys.*, **141**, 112 (1998).

3) J. A. Sethian, "Level Set Methods and Fast Marching Methods," *Cambridge University Press* (1999).

4) M. Sussman, P. Smereka, S. Osher, "A level set approach for computing solutions to incompressible two-phase flow," *J. Comput. Phys.*, **114**, 146 (1994).

5) T. Yabe, R. Tanaka, T. Nakamura, F. Xiao, "Exactly conservative semi-Lagrangian scheme (CIP-CSL) in one dimension," *Mon. Weather Rev.*, **129**, 332 (2001).

6) F. Xiao, T. Yabe, "Completely conservative and oscillationless semi-Lagrangian schemes for advection transportation," *J. Comput. Phys.*, **170**, 498 (2001).

7) F. Xiao, T. Yabe, X. Peng, H. Kobayashi, "Conservative and oscillation-less atmospheric transport schemes based on rational functions," *J. Geophys. Res.*, **107**[D22], 4609 (2002).

8) T. Nakamura, R. Tanaka, T. Yabe, K. Takizawa, "Exactly conservative semi-Lagrangian scheme for multi-dimensional hyperbolic equations with directional splitting technique," *J. Comput. Phys.*, **174**, 171 (2001).

9) K. Takizawa, T. Yabe, T. Nakamura, "Multi-dimensional semi-Lagrangian scheme that guarantees exact conservation," *Comput. Phys. Commun.*, **148**, 137 (2002).

10) T. Yabe, P. Y. Wang, "Unified numerical procedure for compressible and incompressible fluid," *J. Phys. Soc. Japan*, **60**, 2105 (1991).

11) A. Ikebata, F. Xiao, "Development of conservative front capturing scheme and applications to multi-fluid simulations," *Proc. Annual Conference of Japan Society of Mechanical Engineering*, **3**, 301 (2002), [in Japanese].

12) F. Xiao, T. Yabe, T. Ito, M. Tajima, "An algorithm for simulating solid objects suspended in stratified flow," *Comput. Phys. Commun.*, **102**, 147 (1997).

13) F. Xiao, "A computational model for suspended large rigid body in 3D unsteady viscous flow," *J. Comput. Phys.*, **155**, 348 (1999).

14) J. U. Brackbill, D. B. Kothe, C. Zemach, "A continuum method for modeling surface tension," *J. Comput. Phys.*, **100**, 335 (1992).