CUDA と MPI による自由界面圧縮性多相流の数値シミュレーション

Large scale simulation of interfacial compressible multiphase flow with CUDA and MPI

○ 宮川　直，東工大, 東京都目黒区大岡山 2-12-1 石川台 6 号館, E-mail：miyakawa.t.ab@m.titech.ac.jp
ジン　ジョンフン，東工大, 東京都目黒区大岡山 2-12-1 石川台 6 号館, E-mail：jin.j.ab@m.titech.ac.jp
肖　鋒，東工大, 東京都目黒区大岡山 2-12-1 石川台 6 号館, E-mail：xiao.f.aa@m.titech.ac.jp
Tadashi Miyakawa, Tokyo Institute of Technology, 2-12-1 Ishikawadai 6-Building Ōokayama, Meguro, Tokyo
Jonghoon Jin, Tokyo Institute of Technology, 2-12-1 Ishikawadai 6-Building Ōokayama, Meguro, Tokyo
Feng Xiao, Tokyo Institute of Technology, 2-12-1 Ishikawadai 6-Building Ōokayama, Meguro, Tokyo

Numerical simulation of compressible multiphase flows involving moving interfaces is a big challenge to CFD, which requires reliable and accurate numerical methods as well as large scale computations. We have developed a novel numerical method, so-called BVD (Boundary Variation Diminishing), to resolve both smooth and discontinuous solutions in compressible interfacial multiphase flows with very high solution quality. In order to facilitate the large scale high performance computations, we have developed and validated a computational code in this work to implement simulations on hardwares of distributed-memory multi-CPUs with GPGPU accelerators. The code using CUDA and MPI libraries realizes the expected parallel speedup and GPU-acceleration performance, and shows great potential in real-case applications of compressible multiphase flow simulations.

1. 緒言

界面の移動を伴う圧縮性多相流などの複雑流れを高精度に解くことは，数値流体力学の大きな課題となっている．それを解決するために，次の各方面からのアプローチが必要である．一つ目のアプローチは高精度な数値計算モデルの開発，二つ目のアプローチに大規模なシミュレーションに適した計算コードの整備が挙げられる．一つ目のアプローチとして，我々の研究室では圧縮性自由界面多相流の連続領域と不連続領域を高精度に解くことが可能な新しい数値モデル，BVD (Boundary Variation Diminishing)[1][2] を開発した．BVD を用いることで連続領域と不連続領域を精度よく再構築できることが考えられる．そこで本稿では圧縮性多相流の空間再構築に BVD を用いて計算を行った．

BVD を実装したことにより，流体問題を高精度に解くことが可能になった．そこで今回はこの数値モデルを用いて実問題を解くために，二つ目のアプローチである大規模シミュレーションに適した計算コードの整備に焦点を当てる．有限体積法の数値計算では計算領域を格子で区切り，格子内での物理量の変化を計算している．そのため，細かい構造や物理量の変化などを見たい場合は格子を細かく区切る必要がある．計算格子を細かくすることにより大きなメモリが必要になること，計算時間が長くなることが問題となる．この問題に対処するためには，演算領域を分けることで計算時に使用するメモリを分散させること，大規模な並列計算を行うことで計算時間を短くすることが必要である．大規模な計算を行うためには MPI による分散メモリ型のマルチ CPU と GPGPU（General-Purpose Graphics Processing Unit）アクセラレータ [3][4] による高速計算を併用することが挙げられる．GPGPU は CPU より多くの演算コアを保有していることから並列計算を用いることで，高い演算性能を引き出すことができる [3]．そこで本稿では CUDA による GPU の並列スレッド計算とメモリ分散型の並列計算のコードの開発と改良を行った．

本稿では衝撃波と気泡の干渉を含む 3 次元の Shock bubble interaction[5] を例にして並列計算による計算速度の向上および計算結果の妥当性を検証した．プログラムに GPU を実装することで実行時間は CPU の 20 倍程度，並列計算の実装による実行時間は 4GPU で 3 倍程度になり，計算の大規模化を行うことで計算時間の削減ができた．

2. 数値モデルの概要

数値モデルの概要について述べる．

2.1 支配方程式

本稿では直交格子における有限体積法の圧縮性二相流の計算を行う．今回は多相流モデルの Allaire の Five-Equation model[6] を用いる．

$$\frac{\partial \alpha_1 \rho_1}{\partial t} + \nabla \cdot (\alpha_1 \rho_1 \boldsymbol{u}) = 0 \tag{1}$$

$$\frac{\partial \alpha_2 \rho_2}{\partial t} + \nabla \cdot (\alpha_2 \rho_2 \boldsymbol{u}) = 0 \tag{2}$$

$$\frac{\partial \rho \boldsymbol{u}}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) + \nabla p = 0 \tag{3}$$

$$\frac{\partial E}{\partial t} + \nabla \cdot (E\boldsymbol{u} + p\boldsymbol{u}) = 0 \tag{4}$$

$$\frac{\partial \alpha_1}{\partial t} + \boldsymbol{u} \cdot \nabla \alpha_1 = 0 \tag{5}$$

ここで ρ_1，ρ_2 は各相の密度を表し，α_1，α_2 は体積率であり $\alpha_1 + \alpha_2 = 1$ の関係である．状態方程式は，理想気体を用いる．

2.2 数値解法

時間積分法は，2 段 2 次 SSPRK 法 [7] を，近似リーマン解法には HLLC[8] を用いた．全セルにおいて二種類の関数で空間再構築を行い，セル境界値の差が小さくなるように各セルごとに再構築関数を決定する BVD[1][2] を用いて再構築関数の決定を行った．再構築関数には MUSCL と THINC[2][9] を用いた．MUSCL の勾配制限関数には minmod を使用した．

3. CUDA を用いた自由界面圧縮性多相流プログラムの GPU 実装

本研究では圧縮性多相流のコードに GPU アクセラレータを用い，高速計算を実装した．コードのうち，初期条件の設定，CPU と GPU のメモリ転送，GPU コードの実行指示と結果の出力は C 言語で記述した．流体計算はすべて CUDA[10] を用いて GPU の並列計算に対応するコードを実装した．

3.1 GPU による計算高速化の検証

ここでは後述する 3 次元 Shock bubble interaction を用いて検証を行った．検証に使用するコードは，流体計

算部分も C 言語で記述した CPU コードと先述した流体計算部に GPU を実装した GPU コードである．両方のコードを同じ初期条件で 10 Time step 計算した際の計測時間を比較する．時間の計測には timeradd(sys/time.h) を用いた．計算に使用したメッシュ数と計測時間を Tab.1 に示す．計算に使用した CPU は Intel Xeon E5-2680(2.4 GHz)，GPU は Tesla P100 を用いた．コンパイルには CUDA 8.0.61，OpenMPI 2.1.2 を用いた．

Tab. 1: Comparison of the performance between CPU and GPU.

	CPU [s]	GPU [s]	Speed up ratio
$34 \times 64 \times 244$	82.7	4.4	18.9
$44 \times 84 \times 324$	93.4	6.3	14.9
$54 \times 104 \times 404$	318.1	9.9	32.2
$64 \times 124 \times 484$	555.3	14.5	38.4

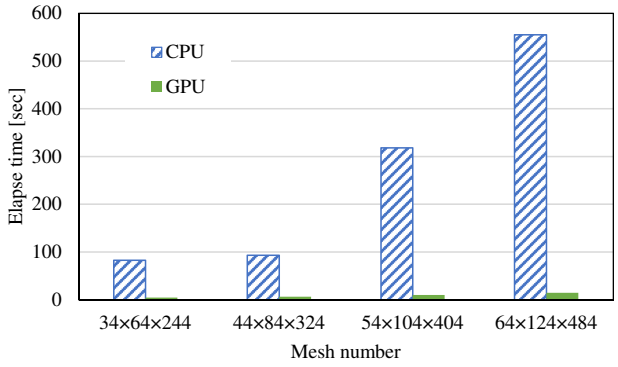

Fig. 1: Elapse time of CPU and GPU(Blue hatching bar is CPU, Green bar is GPU result).

Fig.1 がメッシュ数と計算時間を示すグラフである．縦軸に計測時間，横軸にメッシュ数を示す．青色斜め線の縦棒が CPU の結果を，緑色の縦棒が GPU の結果を示す．

メッシュ数により高速化率が異なるがすべて GPU の計算時間の方が CPU の計算時間に比べ短くなっていることがわかる．Tab.1 から GPU は CPU と比べると本稿で用いたプログラムの場合，20 倍程度の計算時間の高速化できるとわかった．

4. MPI を用いた GPU 実装プログラムの計算並列化

ここでは GPU を実装したプログラムに，分散メモリ型の並列処理である MPI の実装について述べる．

4.1 MPI による GPU 間の通信方法

Fig.2 は今回計算を行った際の計算領域の分割を表す．計算領域を z 方向に k_{max} 個分け，それぞれの領域を 1 つの GPU が担当する．N_x, N_y, N_z は x, y, z 方向の各セル数である．$N_{z,k}$ は k 番目 GPU の z 方向のセルの数である．

境界条件を設定するためにゴーストセルを設定した．ゴーストセルは境界条件のほかに空間再構築する際にも用いる．今回 MPI を用いて通信を行うセルはゴーストセルである．ゴーストセルの値は計算領域に含まれておらず値の更新が行われない．そのため，時間積分や空間再構築の際に両隣の GPU の計算領域から自身のゴーストセルの値を更新するため，MPI による通信が必要である．今回は Fig.2 のように各 GPU が計算する領域を決め，隣り合った領域の境界領域で MPI が行われるように

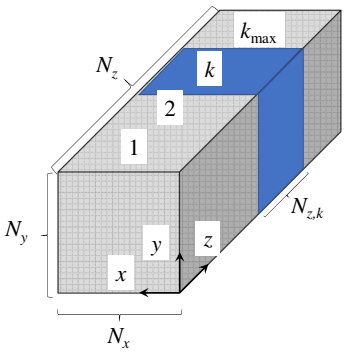

Fig. 2: Mesh and Partition of 3D calculation.

した．今回行った MPI はすべて同期通信で行った．本稿での GPU 間におけるデータの転送は以下の手順で行った．通信を行うセル数は受信側のゴーストセル分である z 方向に 2 セル分の xy 断面である．

(1) GPU1 の計算結果のうち，通信が必要なセルのみ通信用の配列に値をコピー

(2) 通信用の配列を PCIe を通して CPU1 のメモリに転送（cudaMemcpyDeviceToHost）

(3) CPU1 と CPU2 間による MPI

(4) CPU2 から PCIe（cudaMemcpyHostToDevice）を通して GPU2 へデータを転送

(5) 転送された値を計算に使用する配列へコピー

本稿で用いた MPI による通信方法を Fig.3 に示す．Fig.3 は GPU1 から GPU2 にデータを転送する際の図である．立方体は GPU が担当する計算領域，青色の直方体は通信が必要な領域である．CPU mem は CPU のメモリ，GPU mem は GPU のメモリを表す．GPU1 の計算領域のうち通信が必要なデータのみを GPU2 に転送する．

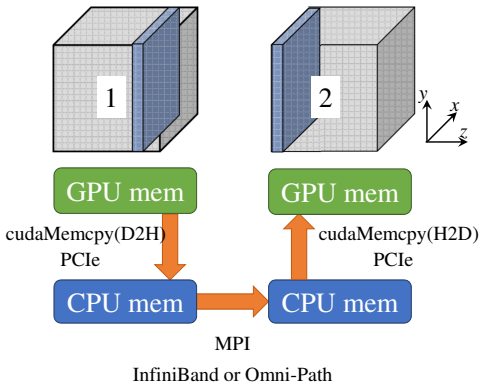

Fig. 3: data transfer for GPU.

時間を進めるための dt は CFL 条件から計算する．今回は等間隔直交格子を用いたため，全計算領域内の最高速度を全ノードで共有する必要がある．最高速度の通信には MPI の集団通信である MPI_Allreduce を用い，各ノードの最高速度を集計しながら全ノードで共有した．

4.2 MPI による計算高速化の検証

上記のような領域分割および通信の方法を用いて計算高速化の検証を行う．流体計算部に GPU を実装したプログラムを用いて上記の MPI による通信を実装した．検証は同じ初期条件，メッシュ数で 100 Time step の計算を行った．GPU の並列数と加速率のグラフを Fig.4 に示す．計算格子数は $104 \times 204 \times 804 = 17,057,664$ である．通信を行うセル数は 42,432 である．計算機の諸元は 3.1 と同様である．

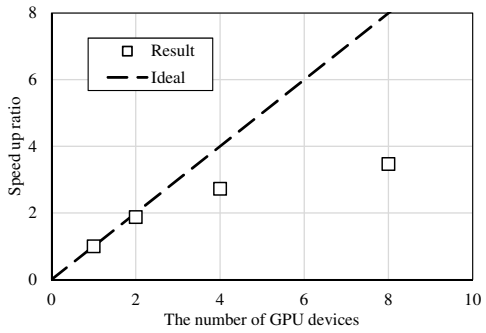

Fig. 4: The result of performance measurement.

加速率は 1GPU の計算時間を複数 GPU を用いたときの計算時間で割ったものである．Fig.4 の破線は理想的な加速率，□は実測値である．本稿で用いたプログラムの場合，2GPU での加速率は 2 倍程度で理想値に近かった．4GPU での加速率は 3 倍程度，8GPU は 3.5 倍程度であった．今回は同期通信を行ったため，通信時間を隠蔽できなかったこと，通信を行うごとにノード間で同期が行われたため，加速率が理想値と離れた値になったと考えられる．

5. 3 次元 shock bubble interraction による数値計算結果について

Hass らや Layes ら [11] は衝撃波管内を大気圧の空気で満たし，球形の空気とは密度の異なる気泡を作成し，それに衝撃波を作用させる実験を行った．本稿ではこの実験を例にして数値シミュレーションを行い，結果の妥当性を検証した．

5.1 初期条件

初期条件は以下の通りである．

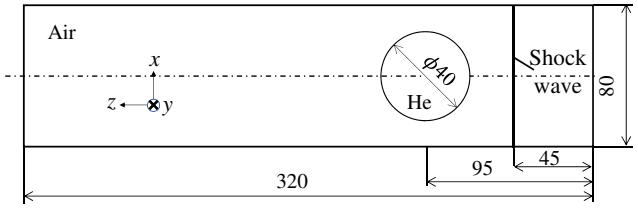

Fig. 5: Initial condition for 3D SBI(unit [mm]).

各辺の長さは $(x[\mathrm{mm}], y[\mathrm{mm}], z[\mathrm{mm}])$=(80 ,80 ,320) である．$\phi$ 40 mm の He の球体を (x, y, z)=(40,40,95) の位置に，衝撃波 (Mach=1.5) を z = 45 mm の xy 平面に配置し．周りの気体を空気とした．計算領域は軸対称と仮定して x 軸方向を半分にして（半球の状態で）計算を行った．計算領域の長さは (40 ,80 ,320) である．今回は Fig.2 のように z 方向に 4 分割して並列計算を行った．計算で使用した各要素の長さは $(dx[\mathrm{mm}], dy[\mathrm{mm}], dz[\mathrm{mm}])$=(0.286,0.286,0.286) である．全体のメッシュ数は $144 \times 284 \times 1124 = 45,967,104$，1GPU あたりの z 方向のメッシュ数 $N_{z,k} = 284$ であり，1GPU あたりのメッシュ数は 11,614,464 である．計算時間は 580 μs まで，計算で使用した変数および物理量は倍精度浮動小数点を用いて計算を実行した．Fig.6 に計算結果を示す．

計算時に使用した各物理量は以下の表のようにした．

Tab. 2: The material properties of the tested jobs.

	γ	$\rho[\mathrm{kg/m^3}]$
Air	1.4	1.29
He	1.67	0.167

5.2 計算結果

Fig.6 の上下方向は x 軸，左右方向は y 軸，奥から手前側は z 軸である．画像は密度の差（左側）と y 軸の渦度（右側）を表した結果である．Fig.6 の上部は，衝撃波が気泡を通過した後（$t = 385\mu$s），下部は衝撃波が気泡を通過し時間が経過した後（$t = 580\mu$s）の図である．Fig.6 の上部から気泡の形は球体から下流側と上流側に 2 つのトーラス形状を持つ物体に変形しており，気泡は下流側と上流側に別れ，渦が発生していることがわかる．Fig.6 の下部を見ると，気泡下流側の渦が発達しているのがわかる．また，$t = 385\mu$s と $t = 580\mu$s を比べると上流側と下流側の 2 つのトーラス形状が離れており，2 つのトーラスは速度が異なっていることがわかる．これらは文献 (5) および (11) と同様の特徴を示している．Fig.6 から，渦は気泡の下流側に集中し渦輪を形成しており，上流側は下流側に比べ渦度が小さくなっていることがわかる．これは文献 (5) と同様の特徴を示している．

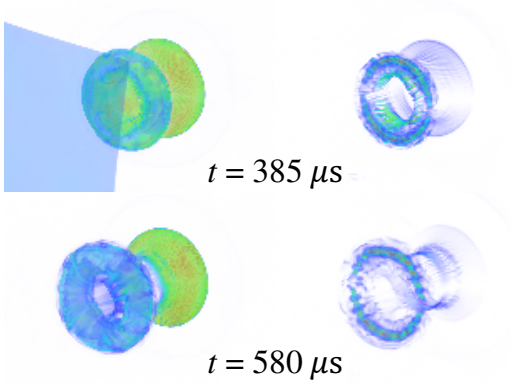

Fig. 6: Results of shock bubble interaction at t = 385 μs (top) t = 580 μs (bottom). Left column shows the difference of density (red and blue colors indicate high and low values respectively), while right column shows the y-direction vorticity

Shock bubble interaction を評価する方法として循環を計算する方法がある [12]．Fig.7 は衝撃波と気泡が衝突しているときの xz 断面の図である．衝撃波は右から左へ移動する．Fig.7 のように，$y = 40$mm の xz 断面のうち気泡の半円部の速度場の閉曲線を P とし，循環を計算した．速度ベクトルを $\mathbf{V}$ と，Fig.7 のように閉曲線 P の循環 Γ は

$$\Gamma = \oint_P \mathbf{V} \cdot ds \tag{6}$$

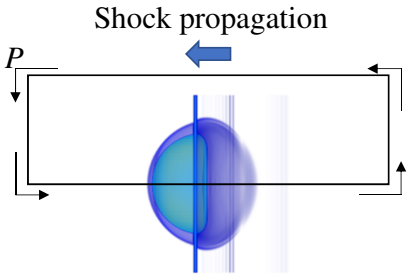

Fig. 7: schematic diagram of shock bubble interaction t = 43 μs.

であり，渦度を ω とおくと，ストークスの定理から，

$$\Gamma = \int_S \omega \cdot dA \tag{7}$$

と書ける．循環は $y = 40$ mm の xz 断面において式 (7) を計算することで正味循環 Γ_0 が得られる．正の循環 Γ_+ は $\omega > 0$ の値を，負の循環 Γ_- は $\omega < 0$ 値を用いて計算することで得られる．本稿のように空気とヘリウムにおける shock bubble interaction では，初期の衝撃波により正方向の回転 ($\omega > 0$) が発生する [12]．これを一次循環といい，今回の場合は正の循環 Γ_+ がこれに該当する [12]．

Fig.8 に循環（$\Gamma_0, \Gamma_+, \Gamma_-$）の結果を示す．

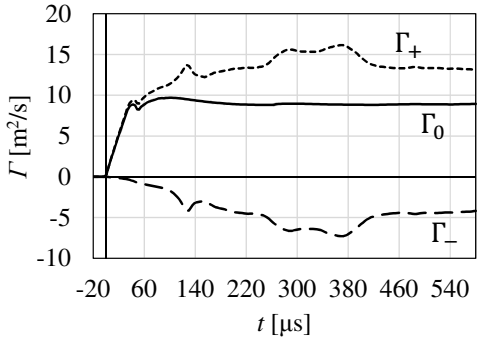

Fig. 8: Decomposed circulation Γ_* versus time, Air-He M=1.5.

Fig.8 は縦軸に循環 $\Gamma[\mathrm{m}^2/\mathrm{s}]$，横軸に時間 $t[\mu\mathrm{s}]$ を示す．Fig.8 を見ると衝撃波は $t = 0$ で気泡に衝突しており，衝突後大きく循環の値が変化している．また，時間が経過した後，Γ_0 の変化は横ばいになる．Γ_+ と Γ_0 の値がともに正の値を取っており，これらは文献 (12) と同様の特徴を示している．

6. 結言

今回は圧縮性自由界面多相流の計算において，BVD を用いた空間再構築手法，CUDA を用いた GPU の実装，MPI を用いた GPU 実装プログラムの並列化を行った．

圧縮性流体の数値計算は並列化が容易であり，今回はプログラムの流体計算部分に CUDA を用いて GPU を実装したことにより計算高速化を行った．本稿で用いたプログラムの場合，20 倍程度の計算高速が図れた．

さらに大規模な計算を行うため，GPU 実装プログラムに MPI を用いて通信を行うい，複数の GPU を用いた計算が可能になった．本稿で用いたプログラムの場合，2GPU での加速率は 2 倍程度．4GPU での加速率は 3 倍程度，8GPU は 3.5 倍程度の計算高速化ができた．

今後の課題として，GPU のさらなる高速化がある．本稿では配列の確保にはすべてグローバルメモリを用いて計算を行った．更なる高速化のためにシェアードメモリなどを利用したメモリの最適化が考えられる．また，MPI について非同期通信や GPU メモリ間での直接通信などを行い通信時間の削減が必要である．また，改良されたコードを用いて他の圧縮性自由界面多相流の問題に取り組むことで本稿の計算より大規模かつ高精度な計算を行うことが考えられる．

参考文献

(1) Ziyao Sun, Satoshi Inaba, Feng Xiao, “Boundary Variation Diminishing (BVD) reconstruction: A new approach to improve Godunov schemes ”, Journal of Computational Physics, Vol. 322, pp. 309-325, 2016.

(2) 稲場　智, 肖　鋒, “THINC 法による圧縮性流体における不連続面の高解像度スキーム”, 第 29 回数値流体力学シンポジウム, 講演番号 C04-4, 2015.

(3) 青木 尊之, “GPU コンピューティングによる CFD の超高速計算,” 第 17 回数値流体力学講演論文集, 1 (2003), pp. 1-1.

(4) 青木 尊之, “GPU コンピューティングによる大規模シミュレーション”, J.Plasma Fusion Res. Vol90, No.12, pp. 755-763, 2014.

(5) G. Layes and O. Le Mtayer, “Quantitative numerical and experimental studies of the shock accelerated heterogeneous bubbles motion”, Physics of Fluids, Vol. 19, p. 042-105, 2007.

(6) Grégoire Allaire, Sébastien Clerc, and Samuel Kokh, “A Five-Equation Model for the Simulation of Interface between Compressible Fluids”, Journal of Computational Physics, Vol. 181, pp. 577-616, 2002.

(7) S. Gottlieb, C.W. Shu, and E. Tadmor, “Strong Stability-Preserving High- Order Time Discretization Methods”, SIAM REVIEW, Vol. 43, pp. 89-112, 2001.

(8) A. Harten, P.D. Lax, and B. van Leer, “On upstream Differencing and Godunov-Type Scheme for Hyperbolic Conservation Laws”, SIAM Review, Vol. 25, pp. 35-61, 1983.

(9) F. Xiao, Y. Honma, and T. Kono, “A simple algebraic interface capturing scheme using hyperbolic tangent function”,International Journal for Numerical Methods in Fluids , Vol. 48, pp. 1023-1040,2005.

(10) NVIDIA:CUDAZone, https://developer.nvidia.com/cuda-zone .2018/10/21 accessed

(11) Babak Hejazialhosseini , Diego Rossinelli , Petros Koumoutsakos, “Vortex dynamics in 3D shock-bubble interaction”,J. Fluid Mech. vol.594, pp. 85-124, 2008

(12) John H. J. Niederhaus , J. A. Greenough, J. G. Oakley, D. Ranjan, M. H. Anderson, R. Bonazza, “A computational parameter study for the three-dimensional shock-bubble interaction”,J. Fluid Mech. vol.594, pp. 85-124, 2008