論文 / 著書情報
Article / Book Information

題目(和文)	AMR法を導入した格子ボルツマン法による混相流の大規模GPUシミュレーション
Title(English)	
著者(和文)	渡辺勢也
Author(English)	Seiya Watanabe
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第11126号, 授与年月日:2019年3月26日, 学位の種別:課程博士, 審査員:青木 尊之,奥野 喜裕,末包 哲也,肖 鋒,長﨑 孝夫
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11126号, Conferred date:2019/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

平成30年度 博士論文

AMR法を導入した格子ボルツマン法による混相流の大規模GPUシミュレーション

東京工業大学 工学院 機械系 機械コース

渡辺勢也

指導教員

青木尊之 教授

2019 年 1 月 29 日

目次

図目次

表目次

第 1 章

緒言

1.1 研究背景

物質は固体，液体，気体の 3 つの状態に区別される．気体が電離した状態をプラズマといい，物資の第 4 の状態である．これらの状態のうち 2 つ以上の状態を含む流れを混相流という．流れに含まれる物質の状態により混相流は大別され，固体と気体の流れは固気二相流，固体と液体の流れは固液二相流，気体と液体の流れは気液二相流，3 つの状態を含む流れは固気液三相流と呼ばれる．

混相流には粉体のような多数の固体粒子を含む流れもあり，粒子系混相流は化学工学や土木工学，防災工学などの様々な分野で現れる．化学プラントでは，流動層による粉体の混合や熱交換，空気による粉粒体のパイプライン輸送などで固気二相流が用いられている．粉粒体の粒子径の測定方法のひとつである沈降法は，溶媒中の粒子の沈降速度から粒子径を計測する技術であり，測定には固液二相流が利用されている．液状化や土石流などの自然災害では，土粒子と水などの流体が相互作用する現象である．これらの粒子を含む流れでは，粒子サイズや粒子数密度，粒子形状が流れ場に大きく影響するが，その流体力学的振る舞いは十分に理解されていない．

このような粒子系混相流の現象を解明するためには，粒子群と流体の振る舞いを理解する必要があり，数値計算によるアプローチが広く行われてきた．固体粒子と流体が相互作用する複雑な流れであるため，これまでは実験や経験則に基づいた粒子と液体の相互作用モデル [1][2] が利用されてきた．固体粒子の運動をラグランジュ的に計算し，格子法による流体計算と連成させる方法であり，Fig.1.1 の左図のように固体粒子より大きい流体計算セルを用いる．流体側には計算セルの空隙率に基づいた体積力を与え，固体粒子側には粒子を真球と仮定した解析的な抗力を与える．この相互作用モデルにより，流動層などでよく一致した結果が得られている [3][4]．しかし，この方法では非球形の粒子を扱うのが困難であり，粒子スケール以下の微視的な流れは解析できない．そのため，複雑な形状の固体粒子を含む場合や高レイノルズ数の乱流などにこの方法を適用することは難しい．

流体と固体粒子の相互作用を直接計算することで，非球形の粒子形状を考慮することや間隙流れの解析が可能となる．そのためには，Fig.1.1 の右図のように，粒子形状を表現できる高解像度の流体計算格子が必要となる．固体粒子の運動を流体計算に移動境界条件として組み込み，粒子の周囲の速度や圧力から粒子に作用する流体力を直接計算する．モデル化による連成計算と比べて膨大な数の流体計算格子

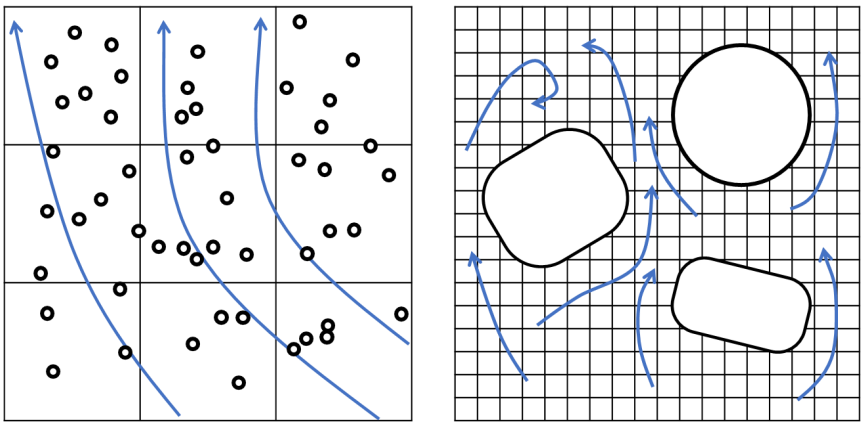


Fig.1.1: Simulation methods for fluid-particle systems.

が必要となり，計算コストが高くなる．そのため，汎用計算機で計算するときは，実際よりもかなり大きい固体粒子を用いて粒子数を削減したり，2 次元計算にすることで格子点数を減らしているのが現状である [5][6].

空気と水などが混ざり合う気液二相流は，原子力発電所など工学的に応用されるだけでなく，津波などの自然災害にも現れる．土石流や湿式のボールミルなどは粉粒体と気液界面を含む固気液三相流であり，気液界面の運動や表面張力などの影響により流れはより複雑になる．固気液三相流の解析には，気液界面の変形や分離，表面張力といた界面現象を精度よく計算するために界面に高解像度が必要である．特に，高レイノルズ数の激しい流れでは，格子を細かくすると新たに液滴や気泡が発生し [7]，十分な解像度での計算は行えていない．混相流の解析では界面での現象が重要であり，それを捉えるためには高解像度を用いた大規模解析が必要となる．

このような混相流は流体の流れが流体の音速に対して十分に遅いため，流体の圧縮性の影響が少なく，音速を無限であると仮定した非圧縮性流体として近似できる．非圧縮性流体解析では，連続の式を満足させるために圧力のポアソン方程式を解き，その圧力で速度場を修正する半陰解法が一般的に用いられる．大規模な連立一次方程式を解く必要があるが，混相流では単相流に比べて係数行列が悪条件となる．大規模計算では，マルチグリッド法などの前処理を用いたとしても，ポアソン方程式の反復計算の収束性が悪化し反復回数が増加してしまう [8]．反復回数が増えるとポアソン方程式を解くためのコストが格子点数に比例しなくなり，計算が大規模になるほど計算全体におけるポアソン方程式のコストが高くなる．混相流の大規模解析の実現には，ポアソン方程式を解く半陰解法を用いず，完全な陽解法による解析が必要である．

格子ボルツマン法 [9] は流体に弱圧縮性を近似することで，ポアソン方程式を解くことなく，完全な陽解法で非圧縮性流体を計算できる．格子ボルツマン法はナビエ・ストークス方程式を解くのではなく，ボルツマン方程式に基づいた計算手法である．流体は仮想粒子の集合と仮定され，粒子の移流と衝突による速度分布関数 $f(\boldsymbol{x}, \boldsymbol{v}, t)$ の時間発展がボルツマン方程式で記述される．格子ボルツマン法は，ボルツマ

ン方程式の速度を有限個の速度方向に離散化し，格子点上での仮想粒子の移流と衝突の計算を行う．例えば，2 次元計算では，1 タイムステップで隣接の 8 個の格子点に移動する分布関数と元の格子点にとどまる分布関数の合計 9 方向の速度の離散化を行う．各格子点にはこの 9 個の速度分布関数を変数として記憶する．格子ボルツマン法の特徴として，完全な陽解法である点とメモリアクセスが局所的である点が挙げられる．大規模計算に向いた計算アルゴリズムであり，スーパーコンピューティングの分野で血管内の流れの解析など多くの大規模計算が実現されている [10][11].

近年のスーパーコンピューターには，演算アクセラレータとして GPU（Graphichs Processing Unit）やメニーコアプロセッサなどが搭載されている．GPU はコンピュータグラフィックスなどの画像処理用のプロセッサとして開発されたが，その演算性能の高さが注目され，CFD などの汎用計算への適用が進められてきた．NVIDIA 社が 2006 年にリリースした CUDA[12] という GPU コンピューティング向けの統合開発環境により，GPU を用いた数値計算の高速化の研究が盛んに行われるようになった．GPU は搭載している数千個の演算コアによる共有メモリ型の並列計算をすることで，1 台の GPU で高い演算性能を出すことができる．例えば，科学技術計算用の最新の GPU である Tesla V100 は 5120 個の演算コアを搭載し，倍精度浮動小数点数の演算で 7 TFLOPS，単精度浮動小数点数で 14 TFLOPS の演算性能である．また，GPU にはアクセス速度が高速である積層メモリが搭載せれており，メモリアクセス時間に計算時間が律速される場合にも GPU は有効である．2018 年 11 月に発表されたスーパーコンピュータの性能ランキング「TOP500」[13] では，NVIDIA の GPU を 27,648 台搭載したオークリッジ国立研究所の Summit が世界 1 位を獲得した．2 位も GPU を搭載したマシンであるローレンス・リバモア国立研究所の Sierra が獲得した．ランキング 10 位以内には他にも，5 位の Piz Daint（スイス），7 位の ABCI（日本），9 位の Taitan（米国）といった GPU を搭載したスパコンがランクインしている．このように，GPU は大規模計算を実現するために，近年注目されていることが確認できる．

格子ボルツマン法の計算はアルゴリズムがシンプルであるため GPU 計算と相性がよく，GPU を用いた格子ボルツマン法の高速化の研究が多く行われている．xiong らは格子ボルツマン法による固液二相流の計算を GPU で高速化し [14]，CPU の 1core の計算に対して最大で約 33 倍の高速化を達成している．しかし，アプリケーションの計算規模は 256 個の球形粒子を含む小規模なものである．格子ボルツマン法はひとつの格子点に全方向の速度分布関数を定義するため，ナビエ・ストークス方程式を解く手法よりもメモリ使用量が多い．GPU のメモリ容量は最新の Tesla V100 でも 16 GB であり，格子ボルツマン法の計算可能な規模はメモリ容量に制限される場合が多い．スーパーコンピューターに搭載された複数台の GPU を用いることで格子ボルツマン法による大規模流体解析が実現されている [15]．小野寺らは 4000 台の GPU を用いることで 500 億格子による都市気流解析を実現した [16]．Wei Ge らは中国のスパコン Mole-8.5 を用いての約 13 万個の微粒子を含む固気二相流の直接計算を実現した [17]．しかし，これらの計算は計算領域全体で同一の解像度を用いた計算である．界面のように時間に伴い大変形する問題では，計算領域全体で高解像度が必要であることは稀であるため，等間隔の直行格子では計算効率が低下してしまう．

格子計算の高速化手法として，Berger らにより提案された AMR 法（Adaptive Mesh Refinement）がある [18]．Fig.1.2 のように物理量に基づき計算領域に解像度の異なる格子を配置する方法であり，界

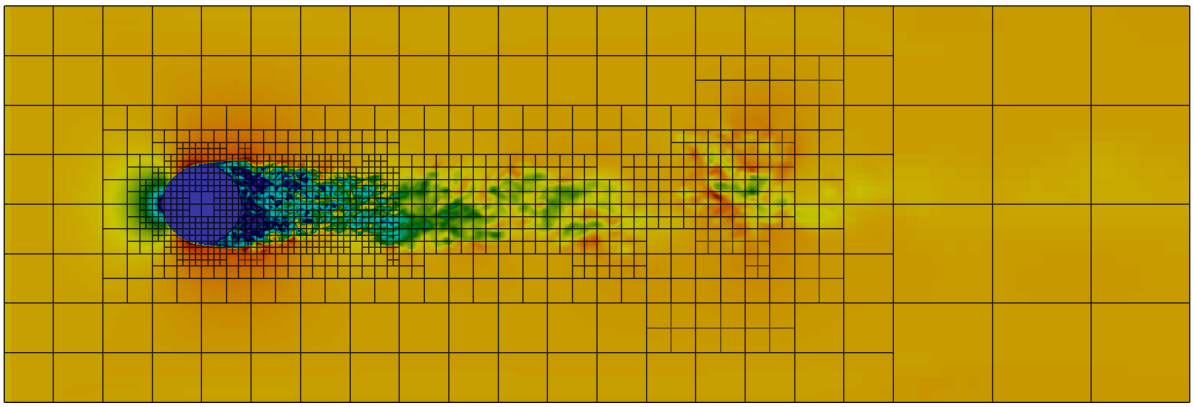


Fig.1.2: An example of an AMR computation. An analysis of a flow around a sphere adapting a fine mesh to the sphere and vortexes is shown as an example.

面や衝撃波，渦などに対して高解像度の格子を配置し，それ以外の領域には低解像度の格子を配置する．これにより，等間隔の格子と同程度の計算精度を維持しつつ，計算格子点数を大幅に削減することが可能である．マルチスケールの問題に対して非常に有効であり，エクサスケールの計算に向けた重要な計算手法として認識されている [19]．米国バークレイ国立研究所は AMR CO-Design Center を立ち上げ，AMReX という超大規模計算向けの AMR フレームワークの開発を進めている [20]．AMR には，解像度の異なる直交格子を任意の位置に配置するパッチ型 [21] と，木データ構造に基づく領域分割を行い，分割の深度によって解像度を変えるツリー型 [22] がある．このうち，ツリー型の AMR は，解像度を制御するアルゴリズムが簡単であるため，ツリー型の AMR が法の研究が多く行われている [23]．特に，木構造で表現された領域に直交格子のパッチ割り当てるブロック構造 AMR[24] は，計算セルのメモリが連続となりメモリアクセス効率が良いため，GPU などのアクセラレータに適している．

自動車や航空機などの空力解析の分野で，AMR 法による格子ボルツマン法の高速化が盛んに行われている．空力解析では，自動車や航空機などの物体表面には境界層を計算するための高解像度格子が必要であるが，同時に計算境界の影響を減らすための広い計算領域が必要である．そのため，AMR 法の導入が必須であり，PowerFLOW[25] や XFlow[26] などの格子ボルツマン法を用いた流体解析の商用ソフトウェアにも AMR 法は用いられている．複数 GPU 計算の研究も行われており，パッチ型の格子を用いた船や自動車周りの乱流解析 [27] や，木構造に基づくブロック構造格子を用いた気流計算 [28] などが行われている．また，ultraFluidX[29] は複数 GPU 計算に対応したブロック構造格子を用いた空力解析ソフトウェアであり，非定常の LES 解析をひとつのサーバーで一晩で完了する．空力解析では，自動車や航空機などの固体表面近傍に高解像度の格子を配置するが，それらの計算では物体は固定されているため格子を動かす必要がない．

混相流のように，高解像度が必要な領域が時間変化する場合，界面や物体の運動に合わせて動的に格子生成を行う必要があり，GPU コードの実装はより複雑になる．格子ボルツマン法に動的な AMR 法を導入した例は少なく，Fakhari らは渦に高解像度格子を適合させる計算 [30] や気液界面に格子を適合した気液二相流計算 [31] などを行っているが，これらの計算は 2 次元計算であり，計算は単一の CPU で

実行されている．他にも円柱周りの流れに AMR を導入した計算 [32] や気泡上昇の 3 次元計算に適用した研究 [33] があるが，GPU 計算や分散メモリ型の並列化は行われていない．ドイツのフリードリヒ・アレクサンダー大学の Rüde らの研究グループは，WaLBerla[34][35] というオープンソースの格子ボルツマン法のコードを開発している．ドイツのスーパーコンピューターである SuperMUC や JUQUEEN などで，動的な AMR 法を用いた大規模な格子ボルツマン法の計算を実現しており [36]，最先端の AMR 法の研究である．WaLBerla は人間の声帯から発生する乱流の解析 [36] や，粉粒体の計算と組み合わせた掃流砂の解析 [37] など様々なアプリケーションに応用されている．しかし，これらの計算は CPU がメインのスーパーコンピューターで実行されており，GPU への対応は十分に行われていない．GPU のメモリ構造は階層的で CPU と大きく異なり，メモリの動的確保や開放を高速に行えないため，メモリ管理が重要となる AMR 法の大規模並列計算はより困難とされている．Top500 より近年のスーパーコンピューターは GPU を搭載したマシンが増加しており，GPU がメインのスーパーコンピューターでの大規模 AMR 計算の技術が必要になると考えられる．

1.2 研究目的

複数の相が混ざり合う混相流の高精度な解析には，固体物体と流体の相互作用や気液界面の運動を捉えるための直接計算が必要であり，シミュレーションは必然的に大規模流体計算になる．本研究では，混相流の直接シミュレーションを GPU スパコンで高効率に実行するための計算手法を開発し，複数 GPU を用いた大規模並列計算により混相流の詳細な解析を実現することを目指す．そのための，格子ボルツマン法による混相流に対する高精度な計算手法の構築，および，界面に高解像度格子を適合する AMR 法を用いた大規模計算を GPU スパコンで高効率に実行するための高速化手法と並列化アルゴリズムの開発を本研究の目的とする．

非圧縮性流体の計算に一般的に用いられる半陰解法では，大規模計算で圧力のポアソン方程式の収束性の悪化により計算コストが大幅に増加するため，大規模な混相流解析を実現するためには半陰解法から脱却し，完全な陽解法による流体解析が必要である．本研究では，完全陽解法である格子ボルツマン法に基づく混相流の解析手法を提案し，固気二相流や自由界面流れに対する大規模シミュレーションを実現する．

混相流シミュレーションでは，物体の表面や気液界面における現象を捉えることが重要であることに着目し，界面と物体表面に高解像度の格子を局所的に集める AMR 法を格子ボルツマン法に導入することで，計算精度を維持しつつ計算コストとメモリ使用量の削減を試みる．AMR の計算では解像度の異なる格子を計算領域の任意の位置に動的に割り当てるため，計算負荷の空間分布が時間変化する．均一格子の並列計算のように計算領域を均等に分割し，分割された小領域の計算を各 GPU に割り振るだけでは，各 GPU が処理する格子点数に偏りが生じる．全体の計算時間は，最も格子点数が多い小領域を担当する GPU の計算時間に律速されるため，大規模計算では並列化効率が著しく低下する．それだけでなく，局所的に高解像度格子が集まる場合，GPU のメモリ容量を上回る格子点数の割り当てが生じ，メモリ不足により計算が破綻してしまう．一台の GPU のメモリ容量が 16 GB に制限されるため，GPU スパコンでの大規模計算の実現には，各 GPU の計算コストの均一化だけでなくメモリ使用量の均一化も重

要となる．AMR 法の複数 GPU 計算で生じるこれらの問題を解決するために，本研究では，計算領域を空間充填曲線に従い動的に分割することで各 GPU が処理する格子点数を均一化し，GPU スパコンでの大規模な AMR 計算を実現する．

GPU スパコンは演算アクセラレータとして GPU を搭載するためノード当たりの演算性能が高いが，ノードをつなぐネットワークの速度は従来の CPU がメインのスパコンと同等であるため，格子計算の大規模計算ではノード間の通信（GPU 間の通信）がオーバーヘッドとなる．そのため，GPU スパコンにおける AMR 法の計算を高効率に実行するためには，計算コストとメモリ使用量を均一にしつつ，GPU 間の通信量を低減できる領域分割法が必要である．しかし，従来の空間充填曲線を用いた領域分割 [38] は通信コストを考慮することが困難であり，グラフ理論に基づく領域分割 [39] は通信コストを最適化できるが分割に時間がかかる．本研究では，多結晶組織のシミュレーション手法であるマルチフェーズフィールド法に基づく AMR 法に対する新しい動的領域分割法を提案し，GPU 間の通信コストの削減による実行性能の向上を試みる．

具体的な研究内容を以下に示す．

第 2 章では，格子ボルツマン法に基づく混相流解析手法に関する研究を示し，格子ボルツマン法による流体解析，フェーズフィールド法による界面捕獲 [40][41]，個別要素法 [42] による粉粒体計算，およびそれらのカップリング手法について述べる．一般的な格子ボルツマン法では，時間積分に一つの緩和時間係数を用いる SRT（Single Relaxation Time）モデルが採用されているが，高レイノルズ数の流れで計算不安定になる．速度に関する速度分布関数のモーメントに変換して計算する MRT（Multiple Relaxation Time）モデル [43] や LES 乱流モデル，統計量であるキュムラントを用いた衝突モデル [44] を導入することで安定性を向上させる．気液二相流に対しては，高レイノルズ数においても計算を安定に行うため，液相のみを解き，界面で境界条件を与える自由界面流れとして扱う．球の沈降計算における沈降速度の実験との比較やダム崩壊計算における水面高さと物体表面圧力の実験との比較などのベンチマーク問題を行い，提案する混相流解析手法の妥当性を評価する．

第 3 章では，2 章で提案した計算手法の複数 GPU 実装，および東京工業大学の TSUBAME を用いた大規模混相流計算について述べる．GPU の階層的なメモリ構造やスレッド並列化の概念について説明し，格子計算と粒子計算の単一 GPU 実装について述べる．粒子法である個別要素法では，周囲の粒子との接触判定でランダムメモリアクセスが頻発し，実行性能が著しく低下してしまう．個別要素法計算で最も計算時間のかかる近傍粒子の探索に関して，既往研究で提案されている様々な手法 [45][46][47][48] の GPU 計算における実行性能を比較し，個別要素法に適している近傍探索手法を検討する．格子ボルツマン法を 2 次元領域分割による複数 GPU 並列化を行い，個別要素法に関しては各 GPU が全ての粒子の計算を冗長に行う方法を提案する．弱スケーリングと強スケーリングによる実行性能と並列化効率を測定し，大規模解析への適用可能性を評価する．複数 GPU を用いて，固気二相流である噴流層や，流木を含む津波流などの実問題に対する大規模シミュレーションを実行し，提案手法の有用性を実証する．

第 4 章では，格子ボルツマン法への AMR 法の実装と AMR 法の複数 GPU 計算について述べる．AMR 法の格子細分化は木構造に基づいて行い，GPU で高い実行性能を出すために木構造で分割された領域に対して均一格子の塊を配置するブロック構造格子を用いる．AMR 法の計算では計算負荷の空間

分布の偏りにより，均一格子の並列計算のように単純な領域分割では並列化効率の悪化やメモリ不足が起こるため，GPU 間の負荷分散を導入する．細分化格子の分割には，METIS[49] などのグラフ理論に基づいた領域分割 [39] や空間充填曲線 [38] による領域分割などが提案されている．グラフ分割は各領域の計算コストを均一にしつつ，領域間の通信コストを最小化することができるが，一回の領域分割にかかるコストが高く，AMR のように計算中に繰り返し領域分割が必要な場合では，分割のコストがオーバーヘッドになる．FEM による応力解析や空力解析などの固定格子を用いる場合は，計算初期に一度だけ領域分割を行うだけでよいのでグラフ分割が適している．一方，空間充填曲線はプロセス間の通信コストを考慮することはできないが，領域分割のコストが小さいため，p4est[23] や DAINO[50]，GAMER[51] といった AMR のフレームワークでも用いられている．本研究では，空間充填曲線のひとつであるモートン曲線による領域分割法を導入し，TSUBAME3.0 で複数 GPU 計算の弱スケーリングと強スケーリングを測定する．球周りの乱流解析や噴流と球の相互作用計算，ダム崩壊計算などの実アプリケーションに取り組み，AMR 法による格子点数の削減効果を評価する．

第 5 章では，マルチフェーズフィールド法 [52][53] を用いた AMR 法に対する新しい領域分割手法の開発について述べる．空間充填曲線による領域分割は低コストであるがプロセス間の通信を考慮することはできず，一方，グラフ分割は通信コストを最適化するが分割のコストが高い．AMR 法で実行性能をより向上させるためには，分割のコストが低く通信量を低減できる分割法が必要だと考えられる．マルチフェーズフィールド法は多結晶組織の成長過程のシミュレーション手法であり，各結晶粒は系の界面エネルギーの減少に基づき成長するため，界面の表面積が最小化された組織が得られる．領域間の通信コストは小領域の表面積に大きく依存するため，マルチフェーズフィールド法で得られた表面積が最小化された組織に基づいて格子を分割することで，通信コストの削減を試みる．マルチフェーズフィールド法の時間発展方程式に各領域の計算コストを均一化するための体積補正項を導入する．固定された格子の領域分割のテストを行うことで，体積補正項のパラメータや格子解像度が領域分割の結果に与える影響を評価する．AMR 法を導入した界面移流計算に提案する領域分割法を導入し，マルチフェーズフィールド法による通信コストの削減を評価する．

以上の研究から，提案する格子ボルツマン法に基づく混相流解析手法が実問題に適用可能であり，GPU コンピューティングと AMR 法により混相流の大規模直接シミュレーションが実現可能になることを示す．

第 2 章

格子ボルツマン法に基づく混相流解析手法の開発

2.1 格子ボルツマン法

一般的な非圧縮性流体の数値計算では，ナビエ・ストークス方程式を差分法などで半陰解法的に解く方法が用いられている．この方法では，圧力のポアソン方程式に対する疎行列の反復計算が必要となる．格子点数が増加すると疎行列の行列サイズが大きくなり，収束性が悪化するため必要な反復計算の回数が増える．大規模計算では，ポアソン方程式の疎行列計算がボトルネックとなり，計算性能が低下する．

格子ボルツマン法 [9] はナビエ・ストークス方程式を解くのではなく，仮想的な流体粒子群の速度分布関数の時間発展方程式を完全な陽解法で計算する非圧縮性流体の解析手法である．ポアソン方程式の疎行列計算がないため，大規模計算でも計算性能が低下しない．局所的なステンシルアクセスで単純な計算を行うため，並列計算を容易に行うことができる．そのため，様々な大規模流体解析に格子ボルツマン法が用いられている [10][11][16]．しかし，格子ボルツマン法の計算は，非圧縮のナビエ・ストークス方程式を 2 次中心差分で解くことに相当するため，数値振動が発生しやすく，解像度が十分で無いと計算が破綻してしまう．また，弱圧縮性流体を仮定しているため，時間刻み幅が小さくなり，所望の時間まで計算するために数十万タイムステップの計算が必要となる．しかし，1 タイムステップにかかる時間が短いため，全体の計算時間としては非圧縮ナビエ・ストークス方程式を解くよりも高速に計算が可能である．

2.1.1 基礎式

格子ボルツマン法は Fig.2.1 のように流体を格子点上で並進・衝突する仮想粒子と仮定し，仮想粒子の速度分布関数の時間発展方程式を解く手法である．仮想粒子の速度分布関数は 1 タイムステップで隣接の格子点に移動する速度を持ち，本研究では，Fig.2.2 に示す速度分布関数を 27 方向の速度で離散化する D3Q27 速度モデルを用いる．流体の拡散過程などに対応する仮想粒子の衝突は，BGK モデル [54] を

用いて速度分布が局所平衡状態へ緩和すると仮定する．速度分布関数 f の時間発展方程式は

$$f_{ijk}\left(\boldsymbol{x}+\boldsymbol{\xi}_{ijk}\Delta t, t+\Delta t\right)=\frac{1}{\tau}f_{ijk}^{\mathrm{eq}}\left(\boldsymbol{x},t\right)+\left(1-\frac{1}{\tau}\right)f_{ijk}\left(\boldsymbol{x},t\right)+G_{ijk}\left(\boldsymbol{x},t\right) \tag{2.1}$$

である．ここで，添字 ijk は速度分布関数の方向で $(i,j,k)\in\{-1,0,1\}^3$，添字 eq は局所平衡状態の速度分布関数を示す．$\boldsymbol{\xi}$ は速度分布関数の速度であり，$c=\Delta x/\Delta t$ を用いて

$$\boldsymbol{\xi}_{ijk}=(ic,jc,kc)^T \tag{2.2}$$

である．τ は速度分布関数の衝突項にかかる緩和係数であり，流体の動粘度を ν として

$$\tau=\frac{1}{2}+\frac{3\nu}{c^2\Delta t} \tag{2.3}$$

と与える．局所平衡状態における速度分布関数 f^{eq} は格子点での流体の速度 $\boldsymbol{u}$ と密度 ρ を用いて

$$f_{ijk}^{\mathrm{eq}}=w_{ijk}\rho\left\{1+\frac{\boldsymbol{\xi}_{ijk}\cdot\boldsymbol{u}}{c_{\mathrm{s}}^2}+\frac{\left(\boldsymbol{\xi}_{ijk}\cdot\boldsymbol{u}\right)^2-c_{\mathrm{s}}^2\left|\boldsymbol{u}\right|^2}{2c_{\mathrm{s}}^4}\right\} \tag{2.4}$$

とする．ここで，w_{ijk} は速度分布関数の方向ごとの重み係数であり

$$w_{ijk}=\begin{cases}8/27, & |\boldsymbol{\xi}_{ijk}|=0\\ 2/27, & |\boldsymbol{\xi}_{ijk}|=1\\ 1/54, & |\boldsymbol{\xi}_{ijk}|=2\\ 1/216, & |\boldsymbol{\xi}_{ijk}|=3\end{cases} \tag{2.5}$$

と設定する．c_{s} は音速であり $c_{\mathrm{s}}^2=c^2/3$ である．重力などの外力は，外力による加速度を $\boldsymbol{a}$ とし，

$$G_{ijk}=-3w_{ijk}\rho\frac{\boldsymbol{\xi}_{ijk}\cdot\boldsymbol{a}_{ijk}}{c^2}\Delta t \tag{2.6}$$

と与える．流体の密度と速度は，速度分布関数から

$$\rho\left(\boldsymbol{x},t\right)=\sum_{i=-1}^{1}\sum_{j=-1}^{1}\sum_{k=-1}^{1}f_{ijk}(\boldsymbol{x},t) \tag{2.7}$$

$$\rho\boldsymbol{u}\left(\boldsymbol{x},t\right)=\sum_{i=-1}^{1}\sum_{j=-1}^{1}\sum_{k=-1}^{1}\boldsymbol{c}_{ijk}f_{ijk}(\boldsymbol{x},t) \tag{2.8}$$

と求める．

2.1.2 MRT（Multiple Relaxation time）モデル

格子ボルツマン法の計算では，時間積分に一つの緩和係数 τ を用いる SRT（Single Relaxation Time）モデルが一般的に用いられるが，その単純さゆえに高レイノルズ数の乱流などの複雑な流れでは計算

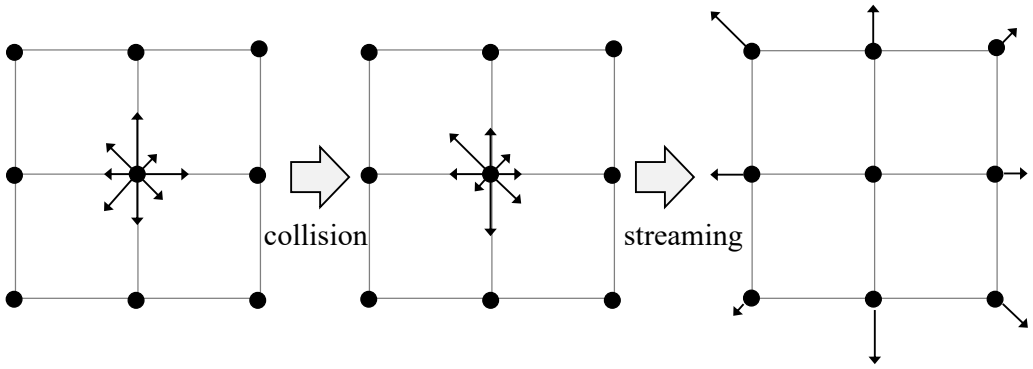


Fig.2.1: Collision and streaming of distribution functions.

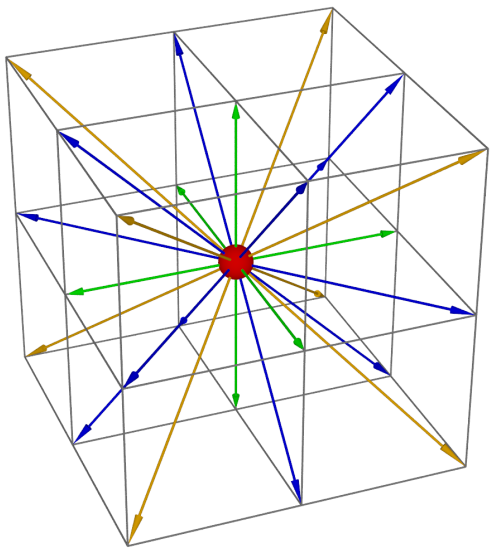


Fig.2.2: D3Q27 velocity model.

が不安定になる．安定性を向上する方法として，複数の緩和係数を用いる MRT（Multiple Relaxation Time）モデルが提案されている [43]．MRT モデルでは速度分布関数 $\boldsymbol{f}$ を密度や運動量，運動エネルギーなどのモーメント $\boldsymbol{m}$ に変換して計算を行う．MRT モデルを用いた格子ボルツマン法の時間発展方程式は

$$\boldsymbol{f}\left(\boldsymbol{x}+\boldsymbol{c}_{ijk}\Delta t, t+\Delta t\right)=\boldsymbol{f}\left(\boldsymbol{x}, t\right)-M^{-1}\hat{S}\left(\boldsymbol{m}-\boldsymbol{m}^{\mathrm{eq}}\right) \tag{2.9}$$

となる．ここで，Q を速度分布関数の方向の数として，M は速度分布関数をモーメントに変換する $Q\times Q$ の行列，$\hat{S}$ は緩和係数の $Q\times Q$ の対角行列を表す．モーメント $\boldsymbol{m}$ と $\boldsymbol{m}^{\mathrm{eq}}$ は，変換行列 M を用

いて

$$\boldsymbol{m} = M\boldsymbol{f} \tag{2.10}$$

$$\boldsymbol{m}^{\mathrm{eq}} = M\boldsymbol{f}^{\mathrm{eq}} \tag{2.11}$$

と計算する．D3Q27 モデルでの変換行列 M は既往研究 [55] で提案されていて，

$$M = \begin{pmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & -1 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 0 \\
0 & 1 & 0 & -1 & 0 & 0 & 1 & 1 & -1 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 \\
-1 & -1 & -1 & -1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -2 \\
2 & -1 & 2 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & -2 & 1 & -2 & 1 & -2 & 1 & -2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & -1 & -1 & 1 & 1 & 1 & 1 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & -1 & 0 & 1 & 0 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 0 \\
-4 & 0 & 4 & 0 & 0 & 0 & -1 & 1 & 1 & -1 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 2 & -2 & -2 & 2 & 2 & -2 & -2 & 2 & 0 \\
0 & -4 & 0 & 4 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 2 & 2 & -2 & -2 & 2 & 2 & -2 & -2 & 0 \\
0 & 0 & 0 & 0 & -4 & 4 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & -2 & -2 & -2 & -2 & 0 \\
4 & 0 & -4 & 0 & 0 & 0 & -2 & 2 & 2 & -2 & -2 & 0 & 2 & 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 & 0 \\
0 & 4 & 0 & -4 & 0 & 0 & -2 & -2 & 2 & 2 & 0 & -2 & 0 & 2 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 & 0 \\
0 & 0 & 0 & 0 & 4 & -4 & 0 & 0 & 0 & 0 & -2 & -2 & -2 & -2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 4 \\
4 & 4 & 4 & 4 & 4 & 4 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & -8 \\
-4 & 2 & -4 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & -2 & 1 & -2 & 1 & -2 & 1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -2 & 0 & -2 & 2 & 2 & 1 & 1 & 1 & 1 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 & -2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 & 2 & 0 & -2 & 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & 2 & 0 & 2 & 0 & -2 & 0 & 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 1 & -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 & 0
\end{pmatrix} \tag{2.12}$$

として与えられる．MRT モデルでは，緩和係数の行列 $\hat{S}$ を適切に設定することで乱流を安定に計算することが可能になる．須賀らにより D3Q27 速度モデルに対する緩和係数の行列 $\hat{S}$ は経験的に提案されており

$$\begin{aligned}
\hat{S} = \mathrm{diag}(&0,\, 0,\, 0,\, 0,\, s_4,\, s_5,\, s_5,\, s_7,\, s_7,\, s_7,\, s_{10},\, s_{10},\, s_{10},\, s_{13},\, s_{13},\, s_{13},\\
&s_{16},\, s_{17},\, s_{18},\, s_{18},\, s_{20},\, s_{20},\, s_{20},\, s_{23},\, s_{23},\, s_{23},\, s_{26}),
\end{aligned} \tag{2.13}$$

$$\begin{aligned}
&s_5 = s_7 = \frac{1}{\tau},\\
&s_4 = 1.54, s_{10} = 1.5, s_{13} = 1.83, s_{16} = 1.4,\\
&s_{17} = 1.61, s_{18} = s_{20} = 1.98, s_{23} = s_{26} = 1.74.
\end{aligned} \tag{2.14}$$

となる．緩和係数を全て $1/\tau$ とした場合は SRT モデルと同等の計算になる．s_5 と s_7 以外の緩和係数を 1 とすれば，1 ステップで多くのモーメントが平衡状態になるため非常に安定した計算が行なえ，このような緩和行列は気液二相流の計算 [56] で用いられている．本研究では，須賀らにより提案された緩和行列を用いる．

2.1.3 キュムラントモデル

MRT モデルよりも安定性を向上した衝突モデルとして，2015 年にキュムラントモデル [44] が提案された．キュムラントモデルは多数の緩和係数を用いる MRT モデル [43] と似ている手法であり，速度分布関数 f を統計的な量であるキュムラント C に変換して衝突過程の計算を行う．MRT モデルでは，各モーメントに異なる緩和係数を設定することで計算を安定化していたが，それにそり SRT モデルでは満たされていたガリレイ不変性が 3 次以上のモーメントで満たされず，ガリレイ不変性を満たしていないことによる誤差が発生してしまう．そのため，MRT モデルでは高レイノルズ数を安定に計算するために，LES などの乱流モデルを導入するのが一般的である．キュムラントモデルでは，Cascaded モデル [57] のように流体の速度場を考慮した中央モーメントを利用することでガリレイ不変性を満たし，MRT モデルよりも更に計算を安定化させる．

キュムラントの次数を $\alpha+\beta+\gamma$ とし，キュムラントは

$$C_{\alpha\beta\gamma} = c^{-\alpha-\beta-\gamma} \left. \frac{\partial^\alpha \partial^\beta \partial^\gamma}{\partial \Xi^\alpha \partial \Upsilon^\beta \partial Z^\gamma} \ln\left(F\left(\boldsymbol{\Xi}\right)\right) \right|_{\boldsymbol{\Xi}=0}, \tag{2.15}$$

$$F\left(\boldsymbol{\Xi}\right) = \mathcal{L}\left[f\left(\boldsymbol{\xi}-\boldsymbol{u}\right)\right] = e^{-\boldsymbol{u}\cdot\boldsymbol{\Xi}} \int_{-\infty}^{\infty} f\left(\boldsymbol{\xi}\right) e^{-\boldsymbol{\Xi}\cdot\boldsymbol{\xi}} d\boldsymbol{\xi} \tag{2.16}$$

と定義される．ここで，$\boldsymbol{\Xi}$ は波数である．キュムラントモデルでの衝突項は

$$C^*_{\alpha\beta\gamma} = \omega_{\alpha\beta\gamma} C^{\mathrm{eq}}_{\alpha\beta\gamma} + \left(1-\omega_{\alpha\beta\gamma}\right) C_{\alpha\beta\gamma} \tag{2.17}$$

となり，$\omega_{\alpha\beta\gamma}$ は各キュムラントに対する緩和係数である．キュムラントモデルを提案した Geier らは 2017 年に緩和係数を適切に設定し [58]，4 次精度の計算を実現している．しかし，計算の安定性に関しては 2015 年に提案された緩和係数が優れているため，本研究では，非対角の 2 次のキュムラントに対しては $\omega_{\alpha\beta\gamma}=1/\tau$ とし，それ以外に対しては $\omega_{\alpha\beta\gamma}=1$ として局所平衡状態へ 1 ステップで緩和するように設定する．衝突過程を計算した後のキュムラント C^* を速度分布関数 f に逆変換する．高次のモーメントでの衝突過程が数値粘性のように作用することで，高レイノルズ数の計算においても計算が安定化し，非粘性流体 $\tau=1/2$ においても計算が破綻しない．

2.1.4 ラージエディシミュレーション

乱流解析で細かい渦まで直接計算する DNS（Direct Numerical Simulation）では，計算に必要な格子点数がレイノルズ数の 9/4 乗に比例するため，高レイノルズ数の流れを解くのは困難である．そのため，高レイノルズ数の計算にはレイノルズ平均モデル（RANS : Raynolds Averaged Navier-Stokes）やラージエディシミュレーション（LES: Large Eddy Simulation）などの乱流モデルを導入する場合が多い．乱流は非定常な現象であるが，工学的には時間平均した流れや効力などの統計量が重要であるため，RANS では時間平均した方程式を用いて解析を行う．計算コストを大幅に削減できるため，空力解析な

どで一般的に利用されている．しかし，非定常な渦を捉える必要がある問題や物体との連成解析などの非定常な問題への RANS の適用は困難である．一方，LES では格子解像度相当までは渦を直接計算し，格子解像度以下の渦をモデル化する．非定常な問題に適用可能であるため，本研究では LES モデルを MRT 型の格子ボルツマン法に導入する．

格子解像度以下の変動が分子粘性と同様に作用するとし，渦粘性 ν_{SGS} を

$$\nu_{\mathrm{SGS}} = C\bar{\Delta}^2|\bar{S}| \tag{2.18}$$

と仮定する．ここで，C はモデル係数，$\bar{\Delta}$ はフィルター幅，$\bar{S}$ はひずみ速度テンソル

$$S_{ij} = \frac{1}{2}\left(\frac{\partial u_j}{\partial x_i} + \frac{\partial u_i}{\partial x_j}\right) \tag{2.19}$$

である．格子解像度以下の変動を考慮した各格子点ごとの動粘度 ν^* を分子粘性 ν_0 と渦粘性 ν_{SGS} から

$$\nu^* = \nu_0 + \nu_{\mathrm{SGS}} \tag{2.20}$$

と設定する．ν^* を用いてボルツマン方程式の緩和係数 τ を各格子点ごとに

$$\tau = \frac{1}{2} + \frac{3\nu^*}{c^2\Delta t} \tag{2.21}$$

と修正する [16]．

一般的に用いられるスマゴリンスキーモデルでは，モデル係数 C を一定とするが，壁境界で適切に渦粘性を扱えない問題がある．動的スマゴリンスキーモデルは流れ場から動的にモデル係数を設定することでスマゴリンスキーモデルの問題を解決している．しかし，モデル係数の設定に流れ場の平均操作が必要であり，大規模計算には不向きである．そこで，本研究では，局所的な速度場からモデル係数を決定でき，壁境界を適切に扱えるコヒーレント構造スマゴリンスキーモデル [59] を用いる．コヒーレント構造関数 F_{CS} は速度勾配テンソルの第二不変量 Q と速度勾配テンソルの大きさ E から計算され，モデル係数 C を

$$C = C_{\mathrm{CSM}}|F_{\mathrm{CS}}|^{3/2} \tag{2.22}$$

$$F_{\mathrm{CS}} = \frac{Q}{E} \tag{2.23}$$

$$Q = -\frac{1}{2}\frac{\partial \bar{u}_j}{\partial x_i}\frac{\partial \bar{u}_i}{\partial x_j} \tag{2.24}$$

$$E = \frac{1}{2}\frac{\partial \bar{u}_j}{\partial x_i}\frac{\partial \bar{u}_j}{\partial x_i} \tag{2.25}$$

と設定する．ここで，C_{CSM} は係数であり，$C_{\mathrm{CSM}} = 1/20$ とする．

2.2 界面捕獲手法

自由界面を含む流れの解析では，流れ場により変形・合体・分裂する界面の表現が必要である．界面形状に合わせた格子を生成する方法 [60] や界面に配置したマーカー粒子を追跡する方法 [61] なども開発されているが，界面が分裂・合体する現象では複雑な処理が必要となる．一方，VOF(Volume Of Fluid) 法 [62] やレベルセット法 [63]，フェーズフィールド法 [40] などは，格子上で界面を陰的に表現する手法であり，複雑に界面が変形する問題にも容易に適用可能である．これらの手法では，界面を陰的に表現する関数を定義し，流れ場に従いプロファイルを移流させることで界面を追跡する．レベルセット法は界面からの距離を表すレベルセット関数を用いて界面を表現する．界面の法線ベクトルや曲率などの幾何的情報を精度良く計算できるが，各相の質量が保存しない問題がある．VOF 法は各計算セル内の流体の割合を用いて界面を表し，流体の割合が 50% の場所を界面とする．体積の保存性に優れているが，移流計算の数値粘性により界面が拡散してしまう．フェーズフィールド法は相を識別する秩序変数を導入し，界面が有限の厚さを持つとして仮定する．移流計算に加え，界面の拡散・逆拡散の計算により界面の厚さを均一に保つことができる．

本研究では，界面捕獲手法としてフェーズフィールド法を用い，この節では，フェーズフィールド法による界面捕獲の計算について述べる．

2.2.1 フェーズフィールド法による界面捕獲

フェーズフィールド法はメソスケールのモデルであるが，界面を捕獲するためにマクロスケールの問題に適用する．界面を陰的に表現する秩序変数 ϕ を定義し，Fig.2.3 のように液相を $\phi = 1$，気相または固相を $\phi = 0$ とし，界面では $0 < \phi < 1$ の値をとる．自由界面で ϕ が急激に変化し，界面を厚さ数メッシュで解像する．界面の時間発展方程式には，激しい流れ場においても界面の厚さを均一に保つことのできる Allen-Cahn 方程式 [41] を用いる．

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\boldsymbol{u}\phi) = \nabla \cdot \left[M \left\{ \nabla \phi - \frac{1 - 4\left(\phi - \phi^{\mathrm{ave}}\right)^2}{W} \boldsymbol{n} \right\} \right] \tag{2.26}$$

ここで，M はモビリティ，W は界面厚さ，$\boldsymbol{n}$ は界面の法線ベクトル，ϕ^{ave} は液相と気相の ϕ の平均値である．Allen-Cahn 方程式の左辺は ϕ の保存方程式，右辺は界面の拡散項と逆拡散項である．拡散項と逆拡散項により界面の厚さを一定に保つことができる．

保存型 Allen-Cahn 方程式を有限体積法などで解く場合，左辺の保存方程式に MUSCLE 法などのスキームを用いると，勾配の計算に 2 点隣りの格子点の値が必要になる．界面を表現する速度分布関数 h を導入することで，隣接格子点のデータのみで保存型 Allen-Cahn 方程式を解くことができる [64]．

速度分布関数 h とフェーズフィールド変数 ϕ は

$$\phi(\boldsymbol{x}, t) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \sum_{k=-1}^{1} h_{ijk}(\boldsymbol{x}, t) \tag{2.27}$$

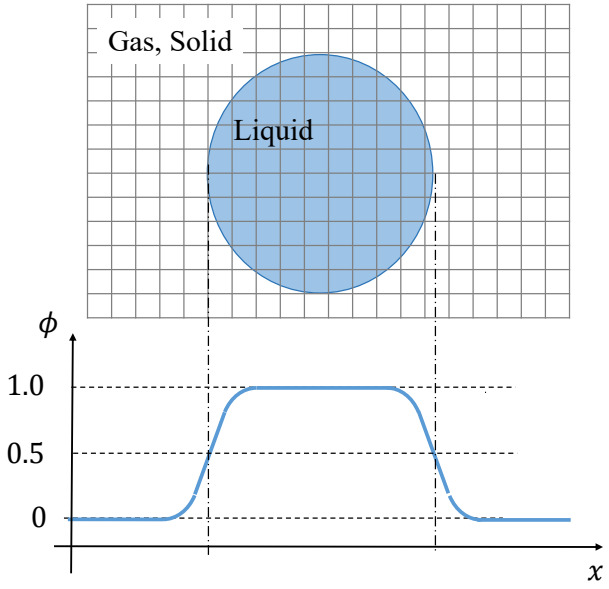


Fig.2.3: Free-surface represented by the phase-field model.

の関係を満たす．速度分布関数 h の時間発展方程式は

$$h_{ijk}\left(\boldsymbol{x}+\boldsymbol{\xi}_{ijk}\Delta t, t+\Delta t\right)=\frac{1}{\tau}h_{ijk}^{\mathrm{eq}}\left(\boldsymbol{x},t\right)+\left(1-\frac{1}{\tau_{\phi}}\right)h_{ijk}\left(\boldsymbol{x},t\right) \tag{2.28}$$

である．ここで τ_{ϕ} は緩和係数であり

$$\tau_{\phi}=\frac{1}{2}+\frac{3M}{c^2\Delta t} \tag{2.29}$$

とする．局所平衡状態の分布関数 h^{eq} は

$$h_{ijk}^{\mathrm{eq}}=\phi\Gamma_{ijk}+\theta w_{ijk}\boldsymbol{\xi}_{ijk}\cdot\boldsymbol{n} \tag{2.30}$$

$$\Gamma_{ijk}=w_{ijk}\left\{1+\frac{\boldsymbol{\xi}_{ijk}\cdot\boldsymbol{u}}{c_{\mathrm{s}}^2}+\frac{\left(\boldsymbol{\xi}_{ijk}\cdot\boldsymbol{u}\right)^2-c_{\mathrm{s}}^2\left|\boldsymbol{u}\right|^2}{2c_{\mathrm{s}}^4}\right\} \tag{2.31}$$

$$\theta=\frac{M}{c_{\mathrm{s}}^2}\left\{\frac{1-4\left(\phi-\phi^{\mathrm{ave}}\right)^2}{W}\right\} \tag{2.32}$$

である．界面の単位法線ベクトル $\boldsymbol{n}$ はレベルセット関数 ψ を用いて計算する．

2.2.2 レベルセット関数

フェーズフィールド法による界面の表現は，体積の保存性に優れ，界面の過度な拡散を防ぐが，界面でフェーズフィールド変数のプロファイルが急激に変化するため，界面の法線ベクトルや曲率などの幾何的情報を高精度に計算することが困難である．そこで，法線と曲率の計算に界面からの距離を表すレベルセット関数 ψ を用いることで，幾何計算の精度を向上させる．

レベルセット関数は

$$|\nabla\psi| = 1 \tag{2.33}$$

の特性を持ち，界面は $\psi = 0$ の等値面，液相領域は $\psi < 0$，気相領域は $\psi > 0$ で表される．フェーズフィールド変数からレベルセット関数 ϕ を

$$\psi = \delta\left(\frac{1}{2} - \phi\right) \tag{2.34}$$

と計算する．拡散界面である $0 < \phi < 1$ の領域には適切なレベルセット関数を計算できるが，$\phi = 0$ と $\phi = 1$ の領域はそれぞれ $\psi = \delta/2$，$\psi = -\delta/2$ の値となり，界面からの距離を適切に計算できない．また，界面から離れた領域では $|\nabla\psi| = 1$ の性質を満たしていない．そこで，レベルセット関数の再初期化処理 [65] を行い，界面から離れた領域のレベルセット関数を修正する．流体計算とは無関係な仮想的な時間刻み t_ψ を設定し，

$$\frac{\partial\psi}{\partial t_\psi} = -S(\psi)\left(|\nabla\phi| - 1\right) \tag{2.35}$$

$$S(\psi) = \frac{\psi}{\sqrt{\psi^2 + |\nabla\psi|^2\,\Delta x}} \tag{2.36}$$

を繰り返し計算する．1 ステップの計算でレベルセット関数は $\Delta t_\psi/\Delta x$ 分修正され，本計算では，$\Delta t_\psi = 1/2\Delta x$ と設定する．本研究では，再初期化の反復回数を 10 回に設定する．また，フェーズフィールド変数からレベルセット関数の作成は流体計算の毎ステップ行う．

界面の法線ベクトル $\boldsymbol{n}$ は，レベルセット関数 ψ から

$$\boldsymbol{n} = \frac{\nabla\psi}{|\nabla\psi|} \tag{2.37}$$

と計算する．

界面の曲率 κ はレベルセット関数を用いて，

$$\begin{aligned}
\kappa &= -\nabla\cdot\boldsymbol{n} \\
&= -\nabla\cdot\frac{\nabla\psi}{|\nabla\psi|} \\
&= -\frac{\psi_x^2\left(\psi_{yy}+\psi_{zz}\right)+\psi_y^2\left(\psi_{zz}+\psi_{xx}\right)+\psi_z^2\left(\psi_{xx}+\psi_{yy}\right)-2\left(\psi_x\psi_y\psi_{xy}+\psi_y\psi_z\psi_{yz}+\psi_z\psi_x\psi_{zx}\right)}{|\nabla\psi|^3}
\end{aligned}$$

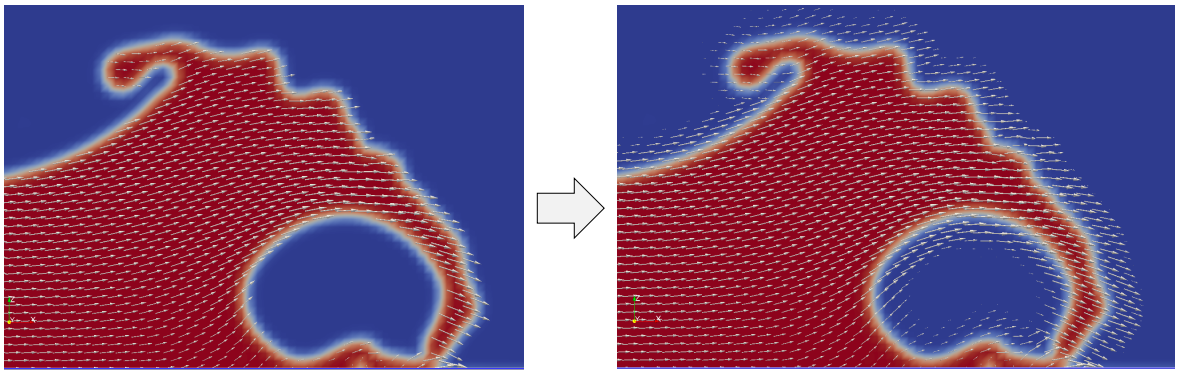


Fig.2.4: Extrapolation of the velocity field using the velocity extension method (left: before, right: after). The liquid is represented by red and the gas by blue.

(2.38)

と求める．ここで，添字は微分を表し，2 次精度中心差分で求める．

2.2.3　Velocity extension 法

Allen-Cahn 方程式を解くためには界面近傍の気相領域に速度場が必要であるが，本研究では，計算の安定性の観点から $\phi > 0.5$ の液相領域の速度場のみを計算する．そこで，Fig.2.4 のように，Velocity extension 法 [66] を用いて液相領域の速度場を気相領域に外挿し，その速度場を用いて気相領域のフェーズフィールド変数の時間発展を計算する．次式を反復計算することで，気相領域に速度成分 u を外挿する．

$$\frac{\partial \boldsymbol{u}}{\partial t_\psi} = -S\left(\psi\right) \frac{\nabla \psi}{|\nabla \psi|} \cdot \boldsymbol{u}, \tag{2.39}$$

$$S\left(\psi\right) = \begin{cases} 0, & \psi < 0 \\ 1, & \psi \geq 0 \end{cases} \tag{2.40}$$

これにより補外された速度成分 u は

$$\nabla u \cdot \boldsymbol{n} = 0 \tag{2.41}$$

を満たし，界面の法線方向に一定の速度となる．ここで，$\boldsymbol{n}$ は界面の単位法線ベクトルである．仮想的な時間刻み幅 t_ψ を $t_\psi = 1/2\Delta x$ と設定し，1 ステップ計算すると気相側の領域に半格子分だけ速度場が補外される．本研究では，反復回数を 10 回に設定し，界面から気相側に 5 メッシュ分の領域に速度を外挿する．空間離散化には 1 次風上差分を用い，時間積分には 1 次オイラー法を用いる．

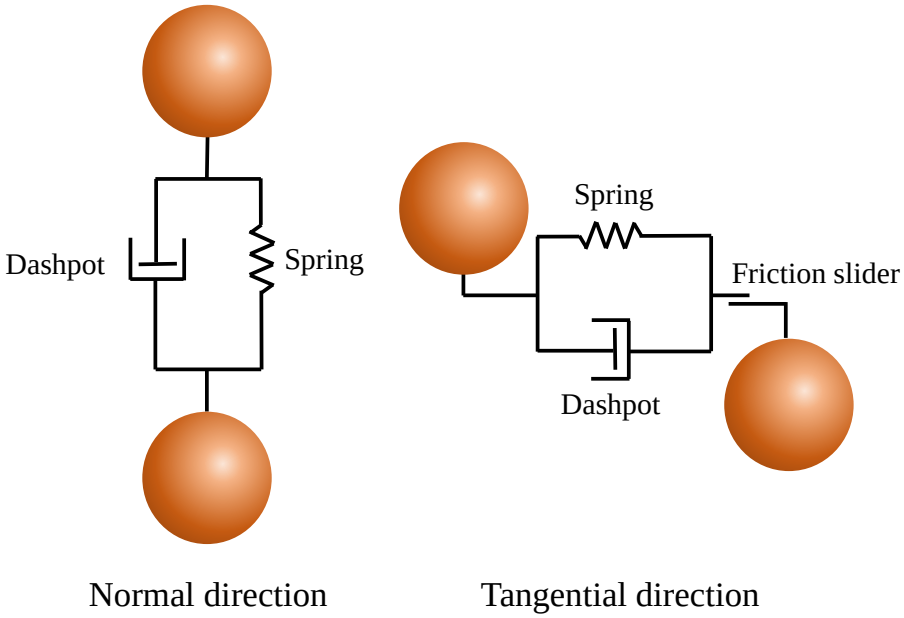


Fig.2.5: Contact force model based on springs and dashpots.

2.3 個別要素法

2.3.1 接触力の計算

粒子群の個々の粒子に着目すると，一つの粒子は周囲の複数の粒子から接触力を受けて運動している．各粒子が接触している粒子から受ける力を計算することが出来れば，ニュートンの第二法則に基づく運動方程式を解くことができ，粒子に作用する加速度が求まる．時間積分により加速度から次の時刻における粒子の速度が計算でき，速度を時間積分することで粒子の位置の更新を行うことができる．系を構成している全ての粒子に対して前述した方法で運動の計算をすれば，それが系全体の挙動を表すことになる．

個別要素法では，Fig.2.5 のように粒子の接触をバネとダッシュポット，摩擦スライダーでモデル化する．接触力はあらゆる方向に作用するが，計算する上で接触する 2 つの粒子における法線方向と接線方向に分離して考える．バネは粒子の食い込み深さに比例した反発力を粒子に与え，ダッシュポットは相対速度に比例した減衰力を与える．粒子の回転や移動により接線方向に相対速度が生じている場合は，粒子間の摩擦が作用するため，バネとダッシュポットに加えて摩擦スライダーが挿入されたモデルで接線方向の力を計算する．摩擦スライダーは，粒子の摩擦係数に応じて摩擦力の上限値を設定する．

球形要素の粒子を用いた場合の接触力の計算について述べる．粒子 i の位置，速度，角速度のベクトルをそれぞれ $\boldsymbol{x}_i$，$\boldsymbol{v}_i$，$\boldsymbol{\omega}_i$ とする．粒子 i と j の半径を r_i と r_j とすると，接触における食い込み深さ δ は

$$\delta = (r_i + r_j) - (\boldsymbol{x}_i - \boldsymbol{x}_j) \cdot \boldsymbol{n} \tag{2.42}$$

となる．ここで，$\boldsymbol{n}$ は 2 つの粒子間における単位法線ベクトルであり，

$$\boldsymbol{n} = \frac{\boldsymbol{x}_i - \boldsymbol{x}_j}{|\boldsymbol{x}_i - \boldsymbol{x}_j|} \tag{2.43}$$

で計算する．$\delta > 0$ のとき粒子は接触している．粒子の法線方向の相対速度 v_n は，

$$v_\mathrm{n} = -(\boldsymbol{v}_i - \boldsymbol{v}_j) \cdot \boldsymbol{n} \tag{2.44}$$

であり，食い込み深さ δ と相対速度 v_n より法線方向の接触力 f_n は

$$f_\mathrm{n} = K_\mathrm{n}\delta + C_\mathrm{n}v_\mathrm{n} \tag{2.45}$$

で計算できる．K_n と C_n は法線方向のバネとダッシュポットの係数である．よって，法線方向の接触力ベクトル $\boldsymbol{f}_\mathrm{n}$ は

$$\boldsymbol{f}_\mathrm{n} = f_\mathrm{n}\boldsymbol{n} \tag{2.46}$$

となる．

次に，接線方向の接触力の計算について述べる．接触点における相対速度ベクトル $\boldsymbol{v}_{ij}$ は，粒子の回転を考慮して

$$\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j + r_i\boldsymbol{n} \times \boldsymbol{\omega}_i + r_j\boldsymbol{n} \times \boldsymbol{\omega}_j \tag{2.47}$$

と計算する．接線方向の相対速度のベクトル $\boldsymbol{v}_\mathrm{t}$ は，$\boldsymbol{v}_{ij}$ より

$$\boldsymbol{v}_\mathrm{t} = \boldsymbol{v}_{ij} - \boldsymbol{n}(\boldsymbol{n} \cdot \boldsymbol{v}_{ij}) \tag{2.48}$$

で求められる．$\boldsymbol{\xi}$ を接線方向のバネの圧縮量ベクトル，K_t と C_t を接線方向のバネとダッシュポットの係数とすると，接線方向の接触力ベクトル $\boldsymbol{f}_\mathrm{t}$ は，

$$\boldsymbol{f}_\mathrm{t} = -K_\mathrm{t}\boldsymbol{\xi} - C_\mathrm{t}\boldsymbol{v}_\mathrm{t} \tag{2.49}$$

で計算できる．接線方向のバネの圧縮量は接触してから離れるまで蓄積されるため，各粒子は接触している粒子との接触履歴を保持する必要がある．着目している時刻における接線方向のバネの圧縮量ベクトル $\boldsymbol{\xi}$ は，

$$\boldsymbol{\xi} = \boldsymbol{\xi}' + \boldsymbol{v}_\mathrm{t}\Delta t \tag{2.50}$$

と前のステップの圧縮量ベクトル $\boldsymbol{\xi}'$ に，接線方向の相対速度のベクトル $\boldsymbol{v}_\mathrm{t}$ から求まる圧縮量の増分を足し合わせることで計算する．前のステップで接触していない場合は，$\boldsymbol{\xi}' = \boldsymbol{0}$ とする．粒子同士が離れた場合は，接線方向のバネの圧縮量を 0 に開放する．接線方向の接触力は，クーロンの摩擦法則から得られる摩擦力の上限値より大きくならないため，摩擦係数を μ として次の条件を与える．

$$\begin{aligned} \boldsymbol{f}_\mathrm{t} &= \boldsymbol{f}_\mathrm{t} && (|\boldsymbol{f}_\mathrm{t}| < \mu|\boldsymbol{f}_\mathrm{n}|) \\ \boldsymbol{f}_\mathrm{t} &= \mu\frac{|\boldsymbol{f}_\mathrm{n}|}{|\boldsymbol{f}_\mathrm{t}|}\boldsymbol{f}_\mathrm{t} && (|\boldsymbol{f}_\mathrm{t}| \geq \mu|\boldsymbol{f}_\mathrm{n}|) \end{aligned} \tag{2.51}$$

これは，Fig.2.5 の個別要素法の接触モデルの摩擦スライダーの効果である．

粒子 i が粒子 j から受ける力 $\boldsymbol{F}_{ij}$ とモーメント $\boldsymbol{M}_{ij}$ は，法線方向と接線方向の接触力ベクトルから

$$\boldsymbol{F}_{ij} = \boldsymbol{f}_{\mathrm{n}} + \boldsymbol{f}_{\mathrm{t}} \tag{2.52}$$

$$\boldsymbol{M}_{ij} = r_i \boldsymbol{n} \times \boldsymbol{f}_{\mathrm{t}} \tag{2.53}$$

と計算する．一つの粒子は複数の粒子から力とモーメントを受けるため，接触しているすべての粒子から受ける力とトルクの総和を計算して，着目粒子に加わる合力を計算する．

$$\boldsymbol{F}_i = \sum_{i \neq j} \boldsymbol{F}_{ij} \tag{2.54}$$

$$\boldsymbol{M}_i = \sum_{i \neq j} \boldsymbol{M}_{ij} \tag{2.55}$$

2.3.2 モデル定数の設定

個別要素法では，接触力モデルのバネとダッシュポットのパラメータ設定の難しさが問題となる．固体粒子のヤング率 E とポアソン比 ν，反発係数 e を用いてパラメータを設定する方法がよく用いられ，バネとダッシュポットの定数は次式で決定する [67][68]．

$$K_{\mathrm{n}} = \frac{2E}{3\left(1-\nu^2\right)}\sqrt{\frac{r}{2}} \tag{2.56}$$

$$C_{\mathrm{n}} = 2\sqrt{\frac{mK_{\mathrm{n}}}{1+\left(\pi/\ln e\right)^2}} \tag{2.57}$$

$$K_{\mathrm{t}} = \frac{K_{\mathrm{n}}}{2\left(1+\nu\right)} \tag{2.58}$$

$$C_{\mathrm{t}} = 2\sqrt{\frac{mK_{\mathrm{t}}}{1+\left(\pi/\ln e\right)^2}} \tag{2.59}$$

$$\tag{2.60}$$

ここで，m は粒子質量，r は粒子半径である．安定して計算するための時間刻み幅 Δt の条件は

$$\Delta t < 2\pi\sqrt{\frac{m}{K_{\mathrm{n}}}} \tag{2.61}$$

であり，多数の粒子が衝突する場合は，これで決定する Δt の十分の一程度にする．しかし，実際のヤング率を用いると，安定に計算するための時間刻み幅が非常に小さくなる．所望の時間まで計算するのに数千万から数十億ステップ必要になり，膨大な計算時間がかかってしまう．そのため，粒子の運動にあまり影響しない程度まで，実際の物性値よりもヤング率を小さくし，時間刻み幅をマイクロ秒程度まで大きく設定する場合が多い [69]．文献 [70] は実際のヤング率を用いて時間刻みをナノ秒のオーダに設定して計算を行っているが，2 次元計算で 6000 個程度の粒子しか扱えていない．

2.3.3　時間積分

個別要素法の計算では，粒子の並進と回転の運動方程式を陽的に時間積分をすることで粒子の位置と速度の更新を行う．粒子の並進と回転の運動方程式はそれぞれ式 (2.62) と式 (2.63) であり，計算された各粒子に加わる力とモーメントを代入する．

$$m_i \frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = \boldsymbol{F}_i + m_i \boldsymbol{g} \tag{2.62}$$

$$I_i \frac{\mathrm{d}\boldsymbol{\omega}_i}{\mathrm{d}t} = \boldsymbol{M}_i \tag{2.63}$$

ここで，m_i と I_i は粒子 i の質量と慣性モーメント，$\boldsymbol{g}$ は重力加速度のベクトルである．式 (2.62) と式 (2.63) を時間積分することで，粒子の速度と角速度の更新を行う．オイラー法の場合は，

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n + \frac{\boldsymbol{F}_i + m_i \boldsymbol{g}}{m_i} \Delta t \tag{2.64}$$

$$\boldsymbol{\omega}_i^{n+1} = \boldsymbol{\omega}_i^n + \frac{\boldsymbol{M}_i}{I_i} \Delta t \tag{2.65}$$

と計算することで，微小時間 Δt 後の速度と角速度を求める．粒子の位置と速度には

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{v}_i \tag{2.66}$$

の関係がある．オイラー法では式 (2.66) を

$$\boldsymbol{x}_i^{n+1} = \boldsymbol{x}_i^n + \boldsymbol{v}_i^n \Delta t \tag{2.67}$$

と時間積分することで，粒子の位置の更新を行う．球形粒子の計算では，式 (2.42) から式 (2.55) の接触力の計算に粒子の回転角 $\boldsymbol{\theta}_i$ の情報は必要ないため，回転角の計算は行わなくても計算を進めていくことができる．

2.3.4　非球形の粒子形状の表現

個別要素法では，実装が容易であり計算コストが低いことから，球形粒子のモデルが一般的に用いられている．転がり摩擦などをモデル化をすることである程度は非球形の効果を取り入れることができるが [71]，複雑な形状の物体同士が噛み合うような相互作用を表現することは非常に困難である．実際の固体粒子の形状が真球であることはほとんどなく，複雑な非球形の形状をしている．そのため，非球形の粒子形状をそのまま個別要素法の計算に用いることで，より正確な粉体の振舞を計算することができる．個別要素法で非球形粒子を扱う方法には，形状をポリゴンで表現する方法 [72][73] や球形粒子を剛体連結して表現する方法 [74] などがある．ポリゴンを用いて非球形の形状を表現する方法では，非球形粒子の面や角を考慮できるため，接触力を精度よく計算できる．しかし，複雑な接触判定や接触力の場合分けの計算コストが高い．一方，球形粒子を連結するモデルでは，非球形粒子の角が丸くなったり表面

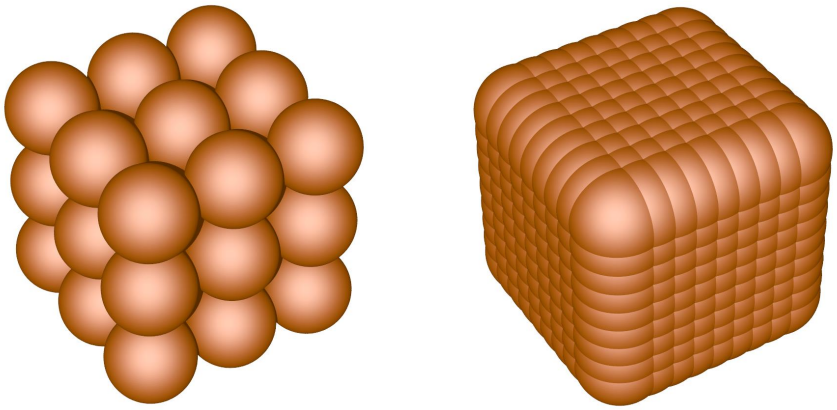


Fig.2.6: Non-spherical particle modeled by multiple spheres.

に凹凸ができたりして，ポリゴンを用いたモデルよりも接触力の計算精度は低くなる．衝突判定や接触力の計算は球形粒子の計算とほとんど同じであり，単純な実装で行えるため，ポリゴンを用いた場合よりも計算コストが大幅に低く，より多くの非球形粒子を用いた計算を行うことができる．そのため，本研究では球形粒子を連結したモデルを用いて非球形粒子の計算を行う．

非球形の形状を球形粒子の集合として表現する方法として，Fig.2.6 に示す 2 つの方法が考えられる．Fig.2.6 は立方体型の非球形粒子をそれぞれの方法でモデル化したものである．1 つ目は左に示すような球形粒子同士をオーバーラップさせないで配置する方法で，粒子数が少なく済む反面，粒子表面に凹凸ができてしまう．2 つ目は右に示すように，粒子を配置するときに粒子のオーバーラップを許容することで表面を滑らかにする事ができるが，球形粒子数の増加と接触粒子数の増加により計算コストは高くなる．本研究では，オーバーラップを許容した 2 つ目のモデルを使用した．このモデルを用いた非球形粒子による個別要素法の計算では，球の集合体を一つの剛体と仮定して，非球形粒子の並進運動と回転運動を計算する．非球形粒子を構成する球形粒子は，他の非球形粒子との接触判定と接触力の計算に用いられる．他の非球形粒子から接触力を受けるのは表面に配置された球形粒子だけで，内部の粒子は力を受けない．そのため，形状を表現するときに内部に粒子を配置してもしなくても計算結果は変わらない．接触判定と接触力の計算に必要である表面の粒子のみ配置し，内部は空洞とすることで非球形粒子を表現するのに必要な粒子数を削減している．

2.3.5 非球形粒子の運動

非球形粒子の運動は，重心の並進と重心周りの回転に関する運動方程式を時間積分していくことで計算する [75]．並進運動の計算は，球形粒子の場合とほぼ同じに計算することができる．非球形粒子に作用する力 $\boldsymbol{F}_{\mathrm{G}}$ は，非球形粒子を構成している各球形粒子 i に作用する接触力 $\boldsymbol{F}_i$ の総和を計算することで

求まる．

$$\boldsymbol{F}_{\mathrm{G}} = \sum^{n_{\mathrm{G}}} \boldsymbol{F}_i \tag{2.68}$$

ここで，n_{G} はその非球形粒子を構成する球形粒子の数である．非球形粒子の質量 m_{G} と並進速度 $\boldsymbol{v}_{\mathrm{G}}$，非球形粒子に作用する力 $\boldsymbol{F}_{\mathrm{G}}$ には以下の関係が成り立つ．

$$m_{\mathrm{G}} \frac{\mathrm{d}\boldsymbol{v}_{\mathrm{G}}}{\mathrm{d}t} = \boldsymbol{F}_{\mathrm{G}} + m_{\mathrm{G}}\boldsymbol{g} \tag{2.69}$$

この式をオイラー法などで時間積分して非球形粒子の重心の位置と速度を更新することで，非球形粒子の並進運動を計算できる．

各球形粒子に接触力が加わることで，非球形粒子には重心周りの回転運動が生じる．非球形粒子に加わるモーメント $\boldsymbol{M}_{\mathrm{G}}$ は，各球形粒子 i に作用する接触力 $\boldsymbol{F}_i$ と球形粒子の位置ベクトル $\boldsymbol{x}_i$ を用いて

$$\boldsymbol{M}_{\mathrm{G}} = \sum^{n_{\mathrm{G}}} \left((\boldsymbol{x}_i - \boldsymbol{x}_{\mathrm{G}}) \times \boldsymbol{F}_i \right) \tag{2.70}$$

と計算する．なお，$\boldsymbol{x}_{\mathrm{G}}$ は非球形粒子の重心の位置である．非球形粒子の角運動量 $\boldsymbol{L}_{\mathrm{G}}$ とモーメント $\boldsymbol{M}_{\mathrm{G}}$ には

$$\frac{\mathrm{d}\boldsymbol{L}_{\mathrm{G}}}{\mathrm{d}t} = \boldsymbol{M}_{\mathrm{G}} \tag{2.71}$$

の関係が成り立つ．角速度 $\boldsymbol{\omega}_{\mathrm{G}}$ は非球形粒子の角運動量 $\boldsymbol{L}_{\mathrm{G}}$ を用いて

$$\boldsymbol{\omega} = \boldsymbol{I}_{\mathrm{G}}(t)^{-1} \boldsymbol{L}_{\mathrm{G}} \tag{2.72}$$

と表すことができる．ここで，$\boldsymbol{I}_{\mathrm{G}}^{-1}(t)$ は時刻 t における非球形粒子の慣性モーメントテンソル $\boldsymbol{I}_{\mathrm{G}}(t)$ の逆行列である．慣性モーメントテンソルは，非球形粒子の回転に伴い変化するため，毎タイムステップ計算する必要がある．慣性モーメントの逆行列 $\boldsymbol{I}_{\mathrm{G}}(t)^{-1}$ は，初期時刻から時刻 t の状態に回転させる回転行列 $\boldsymbol{R}(t)$ を用いて

$$\boldsymbol{I}_{\mathrm{G}}(t)^{-1} = \boldsymbol{R}(t)\, \boldsymbol{I}_{\mathrm{G}}(0)^{-1}\, \boldsymbol{R}(t)^{T} \tag{2.73}$$

と求める．

式 (2.72) と式 (2.73) から求まる非球形粒子の角速度 $\boldsymbol{\omega}_{\mathrm{G}}$ を用いて回転角の更新を行う．3 次元計算の剛体の回転を精度良く扱うためにクォータニオンを導入する．クォータニオンは複素数を拡張したものであり，1 つの実部と 3 つの虚部からなる．3 つの虚数単位を i，j，k とするとクォータニオン $\boldsymbol{q}$ は

$$\boldsymbol{q} = q_{\mathrm{s}} + q_{\mathrm{t}}\mathrm{i} + q_{\mathrm{u}}\mathrm{j} + q_{\mathrm{v}}\mathrm{k} \tag{2.74}$$

と表され，q_{s}，q_{t}，q_{u}，q_{v} はクォータニオンの各成分である．クォータニオン $\boldsymbol{q}$ に共役なクォータニオン $\boldsymbol{q}^*$ は

$$\boldsymbol{q}^* = q_{\mathrm{s}} - q_{\mathrm{t}}\mathrm{i} - q_{\mathrm{u}}\mathrm{j} - q_{\mathrm{v}}\mathrm{k} \tag{2.75}$$

となる．回転操作を行うにはクォータニオンの積を求める計算が必要となる．クォータニオンの虚数単位の乗法は

$$\begin{array}{ll} \mathrm{i}^2 & = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1 \\ \mathrm{ij} & = -\mathrm{ji} = \mathrm{k} \\ \mathrm{jk} & = -\mathrm{kj} = \mathrm{i} \\ \mathrm{ki} & = -\mathrm{ik} = \mathrm{j} \end{array} \tag{2.76}$$

と定義されている．これを用いて，クォータニオン $\boldsymbol{q}$ と $\boldsymbol{p}$ の積は

$$\begin{aligned} \boldsymbol{qp} &= (q_\mathrm{s}p_\mathrm{s} - q_\mathrm{t}p_\mathrm{t} - q_\mathrm{u}p_\mathrm{u} - q_\mathrm{v}p_\mathrm{v}) \\ &+ (q_\mathrm{s}p_\mathrm{t} + q_\mathrm{t}p_\mathrm{s} + q_\mathrm{u}p_\mathrm{v} - q_\mathrm{v}p_\mathrm{u})\,\mathrm{i} \\ &+ (q_\mathrm{s}p_\mathrm{u} - q_\mathrm{t}p_\mathrm{v} + q_\mathrm{u}p_\mathrm{s} + q_\mathrm{v}p_\mathrm{t})\,\mathrm{j} \\ &+ (q_\mathrm{s}p_\mathrm{v} + q_\mathrm{t}p_\mathrm{u} - q_\mathrm{u}p_\mathrm{t} + q_\mathrm{v}p_\mathrm{s})\,\mathrm{k} \end{aligned} \tag{2.77}$$

となる．これらの規則にしたがって共役クォータニオンやクォータニオン積を計算することで，回転操作を行う．説明の都合上，虚部の成分をベクトル $\boldsymbol{V}$ として

$$\boldsymbol{q} = [q_\mathrm{s}, \boldsymbol{V}] \tag{2.78}$$

とクォータニオンを表記する．計算された非球形粒子の角速度 $\boldsymbol{\omega}_\mathrm{G}$ から，微小時間でのクォータニオンの変化 $\Delta\boldsymbol{q}$ を

$$\Delta\boldsymbol{q} = \left[\cos\frac{\theta}{2}, \boldsymbol{a}\sin\frac{\theta}{2}\right] \tag{2.79}$$

とする．ここで，θ は微小時間 Δt での回転角で

$$\theta = |\boldsymbol{\omega}_\mathrm{G}\Delta t| \tag{2.80}$$

と計算し，$\boldsymbol{a}$ は任意の回転軸を表す単位ベクトルであり

$$\boldsymbol{a} = \frac{\boldsymbol{\omega}_\mathrm{G}}{|\boldsymbol{\omega}_\mathrm{G}|} \tag{2.81}$$

と求める．クォータニオンの微小変化 $\Delta\boldsymbol{q}$ と時刻 t におけるクォータニオン $\boldsymbol{q}(t)$ のクォータニオン積を計算することで，クォータニオンの時間発展を行う．

$$\boldsymbol{q}(t + \Delta t) = \Delta\boldsymbol{q}\boldsymbol{q}(t) \tag{2.82}$$

以上により，初期状態での姿勢を基準として，回転の管理を行うことができる．式 (2.73) の時刻 t における回転行列 $\boldsymbol{R}(t)$ はクォータニオン $\boldsymbol{q}(t)$ を用いて

$$\boldsymbol{R}(t) = \begin{pmatrix} 1 - 2q_\mathrm{u}^2 - 2q_\mathrm{v}^2 & 2q_\mathrm{t}q_\mathrm{u} - 2q_\mathrm{s}q_\mathrm{v} & 2q_\mathrm{t}q_\mathrm{v} + 2q_\mathrm{s}q_\mathrm{u} \\ 2q_\mathrm{t}q_\mathrm{u} + 2q_\mathrm{s}q_\mathrm{v} & 1 - 2q_\mathrm{t}^2 - 2q_\mathrm{v}^2 & 2q_\mathrm{u}q_\mathrm{v} - 2q_\mathrm{s}q_\mathrm{t} \\ 2q_\mathrm{t}q_\mathrm{v} - 2q_\mathrm{s}q_\mathrm{u} & 2q_\mathrm{u}q_\mathrm{v} + 2q_\mathrm{s}q_\mathrm{t} & 1 - 2q_\mathrm{t}^2 - 2q_\mathrm{u}^2 \end{pmatrix} \tag{2.83}$$

と与えられる．

2.3.6　非球形粒子を構成している球形粒子の更新

非球形粒子の重心の並進運動と回転運動の方程式を時間積分し，1 タイムステップ後の位置と速度，角速度，クォータニオンが求まったら，構成してる球形粒子の更新を行う．構成している球形粒子の更新は，運動方程式を時間積分して更新していくのではなく，計算された非球形粒子の重心の並進と回転運動に合わせて，位置や速度を与える．初期時刻における粒子 i と構成している非球形粒子の重心の相対座標を $\boldsymbol{r}_i(0)$ とすると，時刻 t の相対座標 $\boldsymbol{r}_i(t)$ は

$$\boldsymbol{r}_i(t) = \boldsymbol{q}(t)\,\boldsymbol{r}_i(0)\,\boldsymbol{q}^*(t) \tag{2.84}$$

のクォータニオンによるベクトルの回転操作を行うことで求める．球形粒子の位置 $\boldsymbol{x}_i$ は，非球形粒子の重心の位置 $\boldsymbol{x}_\mathrm{G}$ と相対座標 r_i を用いて

$$\boldsymbol{x}_i = \boldsymbol{x}_\mathrm{G} + \boldsymbol{r}_i \tag{2.85}$$

と計算する．球形粒子の速度は非球形粒子の速度 $\boldsymbol{v}_\mathrm{G}$ と角速度 $\boldsymbol{\omega}_\mathrm{G}$ より

$$\boldsymbol{v}_i = \boldsymbol{v}_\mathrm{G} + \boldsymbol{\omega}_\mathrm{G} \times \boldsymbol{r}_i \tag{2.86}$$

と求まる．球形粒子による個別要素法では球形粒子は回転の自由度をもっていたが，非球形粒子の場合は構成している球形粒子には回転の自由度がなく，球形粒子の角速度 $\boldsymbol{\omega}_i$ は常に $\boldsymbol{\omega}_i = \boldsymbol{0}$ にしておく．このようにして更新された球形粒子の位置と速度を用いて，各球形粒子に加わる接触力を計算する．

2.4　連成手法

この節では，格子ボルツマン法とフェーズフィールド法による界面捕獲手法，個別要素法を組み合わせる方法について述べる．

2.4.1　自由界面の境界条件

本研究の計算では，フェーズフィールド変数 ϕ が 0.5 以上の領域を液相とし，液相領域の流れ場のみを格子ボルツマン法で解き，界面で境界条件を与えることで自由界面流れを計算する．自由界面の境界では，Fig.2.7 のように気相領域から移流してくる速度分布関数（Fig.2.7 の黒色の矢印）が未知であり，その分布関数に自由界面の条件を与える [76]．気相領域の格子点から移流してくる速度分布関数を

$$f_{ijk}(\boldsymbol{x}, t+\Delta t) = -f_{\overline{ijk}}(\boldsymbol{x}, t) + f^\mathrm{eq}_{ijk}(\rho_\mathrm{b}, \boldsymbol{u}) + f^\mathrm{eq}_{\overline{ijk}}(\rho_\mathrm{b}, \boldsymbol{u}) \tag{2.87}$$

と計算する．添字 $\overline{ijk}$ は反対方向の速度分布関数を示す．ρ_b は自由界面における密度であり，自由界面の曲率を κ，表面張力を σ として

$$\rho_\mathrm{b} = \rho_0 - 6\kappa\sigma \tag{2.88}$$

と与え，自由界面における圧力と大気圧の釣り合いを考慮する．表面張力はラプラス圧に基づいて与えられる．

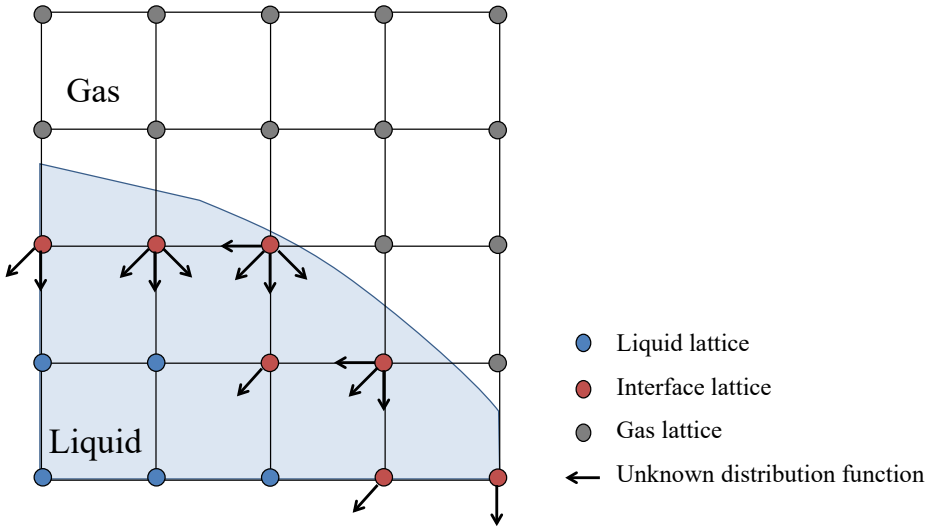


Fig.2.7: Unknown distribution functions from lattice points of the gas phase.

2.4.2　移動境界条件

流体と物体の連成計算を行うためには，物体の運動を移動境界条件として格子ボルツマン法の計算に組み込む必要がある．Fig.2.8 のように格子点の間にある壁の位置を高精度に表現できる Interpolated bounce-back[77] に基づく移動境界条件 [78] を課す．これは，Fig.2.9 のように壁に向かう速度分布関数が壁で跳ね返り，元の格子点に戻ってくると仮定して境界条件を与える方法である．速度分布関数は 1 タイムステップで隣の格子点に移動する速度を持っているため，格子点間の距離と格子点から壁までの距離の比 q に応じて，次式で計算する．

$$f_{ijk}\left(\boldsymbol{x}, t+\Delta t\right) = \begin{cases} 2q f_{\overline{ijk}}\left(\boldsymbol{x}, t\right) + \left(1-2q\right) f_{\overline{ijk}}\left(\boldsymbol{x}+\boldsymbol{c}_{ijk}\Delta t, t\right) + \dfrac{6\omega_{ijk}\rho\left(\boldsymbol{c}_{ijk}\cdot\boldsymbol{u}_{\mathrm{wall}}\right)}{c^2}, & 0 < q \leq 0.5 \\ \dfrac{1}{2q}\left[f_{\overline{ijk}}\left(\boldsymbol{x}, t\right) + \dfrac{6\omega_{ijk}\rho\left(\boldsymbol{c}_{ijk}\cdot\boldsymbol{u}_{\mathrm{wall}}\right)}{c^2}\right] + \dfrac{\left(2q-1\right)}{2q} f_{ijk}\left(\boldsymbol{x}, t\right), & 0.5 < q \leq 1.0 \end{cases} \tag{2.89}$$

ここで，添字は $\overline{ijk}$ 反対向きの速度分布関数の方向，$\boldsymbol{u}_{\mathrm{wall}}$ は壁の速度を表す．移動物体の影響は壁面における運動量交換に基づいて与えられる．

フェーズフィールド変数の分布関数 h に関しては，保存性を向上するために $q = 0.5$ とした Interpolated bounce-back 法を用いて移動境界条件を設定する．

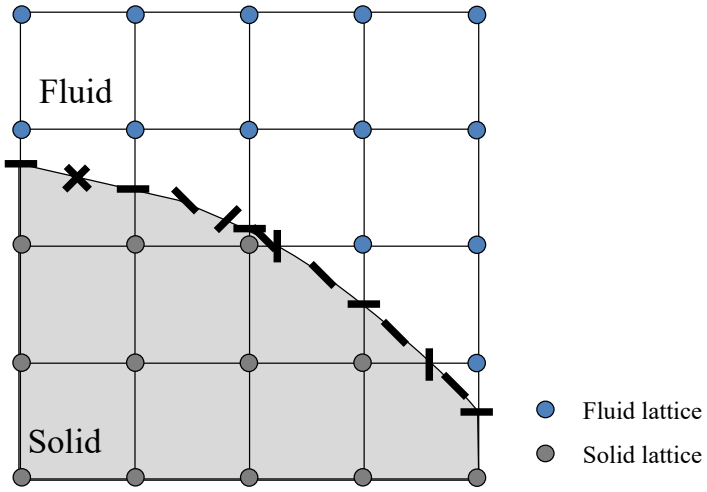


Fig.2.8: Object surface represented by the interpolated bounce-back scheme.

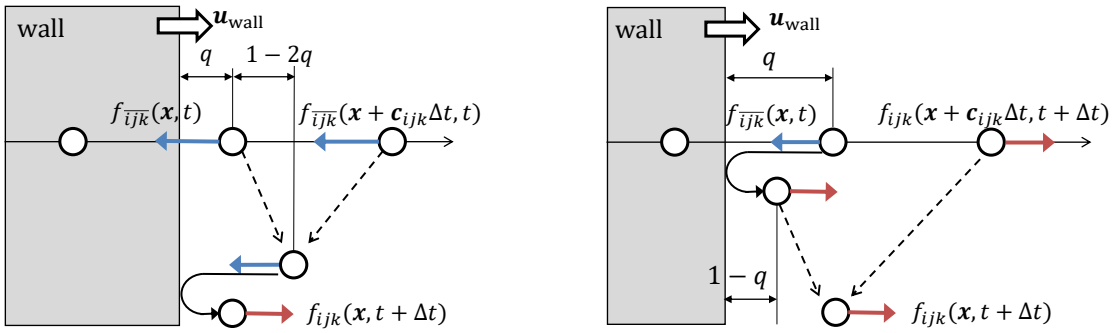


Fig.2.9: Interpolated bounce-back scheme (left: $0 < q \leq 0.5$, right: $0.5 < q \leq 1$).

2.4.3　物体に作用する流体力の評価

壁で反射された速度分布関数が物体に与える流体力は，壁面における分布関数と物体の運動量交換に基づいて

$$\boldsymbol{F}_{ijk}\left(\boldsymbol{x},t\right)=\left(\boldsymbol{c}_{ijk}-\boldsymbol{u}_{\mathrm{wall}}\right)f_{ijk}\left(\boldsymbol{x},t\right)-\left(\boldsymbol{c}_{\overline{ijk}}-\boldsymbol{u}_{\mathrm{wall}}\right)f_{\overline{ijk}}\left(\boldsymbol{x},t+\Delta t\right) \tag{2.90}$$

と計算する [79]．壁面に対して斜め方向に移動する分布関数が与える流体力に，壁面の接線方向のせん断力が考慮されている．

物体 l に作用する流体力 $\boldsymbol{F}_{\text{fluid}}$ とトルク $\boldsymbol{T}_{\text{fluid}}$ は，物体 l の壁面領域 Ω_l の格子点において $\boldsymbol{F}_{ijk}$ の総和を計算することで求める．

$$\boldsymbol{F}_l^{\text{fluid}} = \sum_{\boldsymbol{x}\in\Omega_l} \sum_{i=-1}^{1} \sum_{j=-1}^{1} \sum_{k=-1}^{1} \boldsymbol{F}_{ijk}\left(\boldsymbol{x},t\right) \tag{2.91}$$

$$\boldsymbol{T}_l^{\text{fluid}} = \sum_{\boldsymbol{x}\in\Omega_l} \left[\left(\boldsymbol{x}-\boldsymbol{x}_l\right) \times \sum_{i=-1}^{1} \sum_{j=-1}^{1} \sum_{k=-1}^{1} \boldsymbol{F}_{ijk}\left(\boldsymbol{x},t\right) \right] \tag{2.92}$$

ここで，$\boldsymbol{x}_l$ は物体 l の重心の位置である．流体から物体に作用する力 $\boldsymbol{F}_{\text{fluid}}$ とトルク $\boldsymbol{T}_{\text{fluid}}$ を物体の並進と回転の運動方程式に代入する．

2.4.4 物体・自由界面の移動により発生する新しい流体格子点の扱い

Fig.2.10 のように点線の位置にあった物体が実線の位置に移動した場合，物体内部の格子点が液相の格子点に変わる（Fig.2.10 の赤色の点）．液相領域の格子点にのみ速度分布関数の時間発展を計算しており，気相および固相の格子点には速度分布関数が定義されていない．そのため，新しい流体格子点の速度分布関数を壁の移動速度 $\boldsymbol{u}_{\text{wall}}$，周囲の流体格子点（水色の点）の密度の平均値 ρ_{ave}，壁の法線方向の格子における非平衡成分を用いて

$$f_{ijk} = f_{ijk}^{\text{eq}}\left(\boldsymbol{x};\rho_{\text{ave}},\boldsymbol{u}_{\text{wall}}\right) + \left[f_{ijk}\left(\boldsymbol{x}+\boldsymbol{\xi}_{\text{n}}\Delta t,t\right) - f_{ijk}^{\text{eq}}\left(\boldsymbol{x}+\boldsymbol{\xi}_{\text{n}}\Delta t,t\right) \right] \tag{2.93}$$

と計算する [80]．ここで，$\boldsymbol{\xi}_{\text{n}}$ は物体表面の単位法線ベクトル $\boldsymbol{n}_{\text{wall}}$ の方向にある格子点の方向を表すベクトルであり，$\boldsymbol{n}_{\text{wall}} \cdot \dfrac{\boldsymbol{\xi}_{ijk}}{|\boldsymbol{\xi}_{ijk}|}$ が最大となる $\boldsymbol{\xi}_{ijk}$ を用いる．

自由界面の移動により気相領域の格子点が液相領域の格子点に変わる場合は，隣接の液相格子の分布関数を用いて，新しい液相格子の分布関数を

$$f_{ijk}\left(\boldsymbol{x},t\right) = \frac{1}{N_{\text{liquid}}} \sum_{l=-1}^{1} \sum_{m=-1}^{1} \sum_{n=-1}^{1} \Omega_{lmn}\left(\boldsymbol{x},t\right) f_{ijk}\left(\boldsymbol{x}+\boldsymbol{\xi}_{lmn}\Delta t,t\right) \tag{2.94}$$

と平均値を与える．ここで，N_{liquid} は隣接している液相の格子点数，Ω_{lmn} は隣接の格子点が液相領域かを識別する変数であり

$$\Omega_{lmn}\left(\boldsymbol{x},t\right) = \begin{cases} 1, & \boldsymbol{x}+\boldsymbol{\xi}_{lmn}\Delta t \in \text{liquid} \\ 0, & \boldsymbol{x}+\boldsymbol{\xi}_{lmn}\Delta t \notin \text{liquid} \end{cases} \tag{2.95}$$

である．

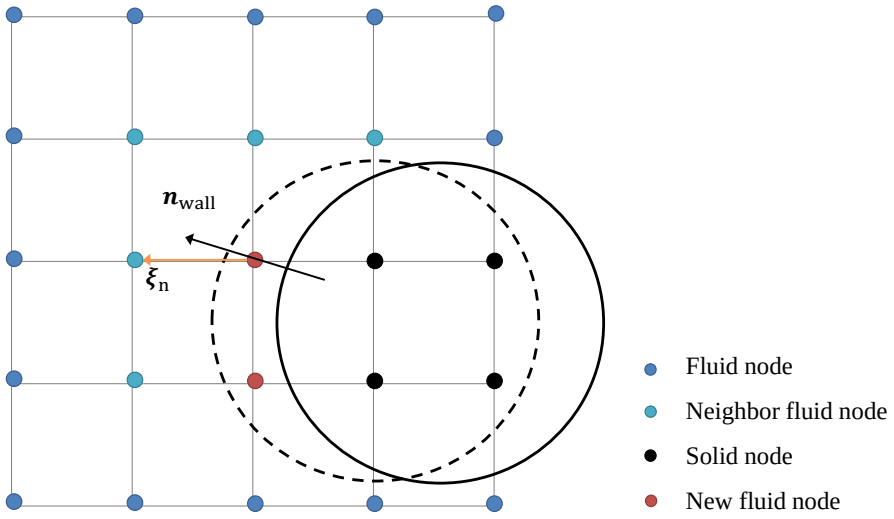


Fig.2.10: Refill of the velocity distribution function at lattice point changed from the solid phase to the liquid phase when the spherical object moves from the dashed line to the solid line.

2.4.5　サブタイムステップ

格子ボルツマン法と個別要素法の連成計算で問題となるのが，時間刻み幅の違いである．格子ボルツマン法の時間刻み Δt_{LBM} は，式 (2.3) の関係から

$$\Delta t_{\mathrm{LBM}} = \left(\tau - \frac{1}{2}\right)\frac{\Delta x^2}{3\nu} \tag{2.96}$$

となる．一方，個別要素法の時間刻み Δt_{DEM} は，粒子質量を m，法線方向のバネ定数 を K_{n} として，

$$\Delta t_{\mathrm{DEM}} < 2\pi\sqrt{\frac{m}{K_{\mathrm{n}}}} \tag{2.97}$$

を満たすように設定する．個別要素法で実際のヤング率に近い硬い材質で計算する場合は，バネ定数が大きくなるため時間刻み幅が非常に小さくなる．粒子の運動に影響が出ないまでバネ定数を小さくし，時間刻み幅を大きく設定しても，格子ボルツマン法の時間刻み幅と比べて小さくなる場合が多い．格子ボルツマン法の時間刻みを小さくして，個別要素法と同じ時間刻み幅で計算することもできるが，格子ボルツマン法の 1 タイムステップあたりの計算時間は，個別要素法と比べて遥かに大きいため，計算コストが大幅に増加する．そこで，格子ボルツマン法の 1 タイムステップ計算に対して，個別要素法の計算を複数ステップ行う方法を用いる [81]．式 (2.97) の条件をみたすように，個別要素法の時間刻みを

$$\Delta t_{\mathrm{DEM}} = \frac{1}{N_{\mathrm{substep}}}\Delta t_{\mathrm{LBM}} \tag{2.98}$$

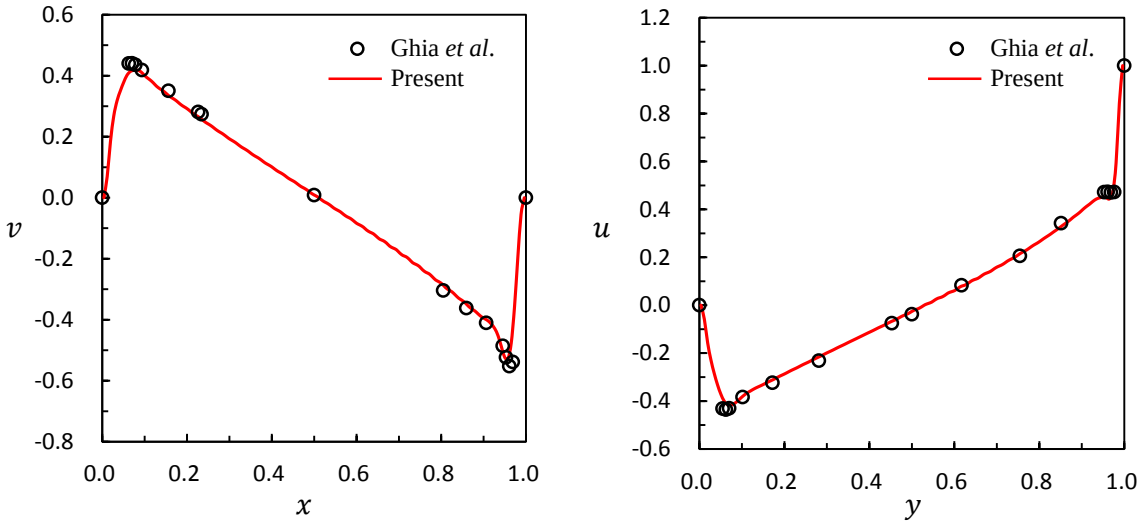


Fig.2.11: Results of the cavity flow problem at $Re = 7500$.

と設定する．ここで，N_{substep} は 1 以上の整数である．格子ボルツマン法の 1 ステップに対して，N_{substep} 回の個別要素法の計算ステップを行う．N_{substep} 回の計算を行う際は，流体から受ける力とトルクは一定と仮定する．これにより，計算コストの増加を抑えつつ，2 つの手法とも安定して計算することが可能になる．

2.5　検証計算

2.5.1　キャビティ流れ

格子ボルツマン法による流体解析の検証として，2 次元のキャビティーフローの計算を行う．流体で満たされた正方形の計算領域を設定し，上面が一定速度で移動している問題である．格子点数は 256×256 として，レイノルズ数 $7,500$ の条件を設定する．$x = 0.5$，$y = 0.5$ での流速を Ghia ら [82] の結果と比較する．

計算結果を Fig.2.11 に示す．左図が $y = 0.5$ での y 方向の流速 v で，右図が $x = 0.5$ での x 方向の流速 u である．本研究の結果は Ghia らの結果をとよく一致しており，単相の流体計算が十分に行えることが確認できた．

2.5.2　3 次元速度場における界面の移流計算

フェーズフィールド法による界面捕獲の検証として，与えられた速度場で界面を引き延ばすベンチマークテストを行う．計算領域 $[0,1]\times[0,1]\times[0,1]$ を設定し，半径 0.15 の球形プロファイルを位置 $(0.35,0.35,0.35)$ に配置する．球形のプロファイルを 3 次元の非圧縮性速度場 [83]

$$u(x,y,z,t) = 2\sin^2(\pi x)\sin(2\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right) \tag{2.99}$$

$$v(x,y,z,t) = -\sin(2\pi x)\sin^2(\pi y)\sin(2\pi z)\cos\left(\frac{\pi t}{T}\right) \tag{2.100}$$

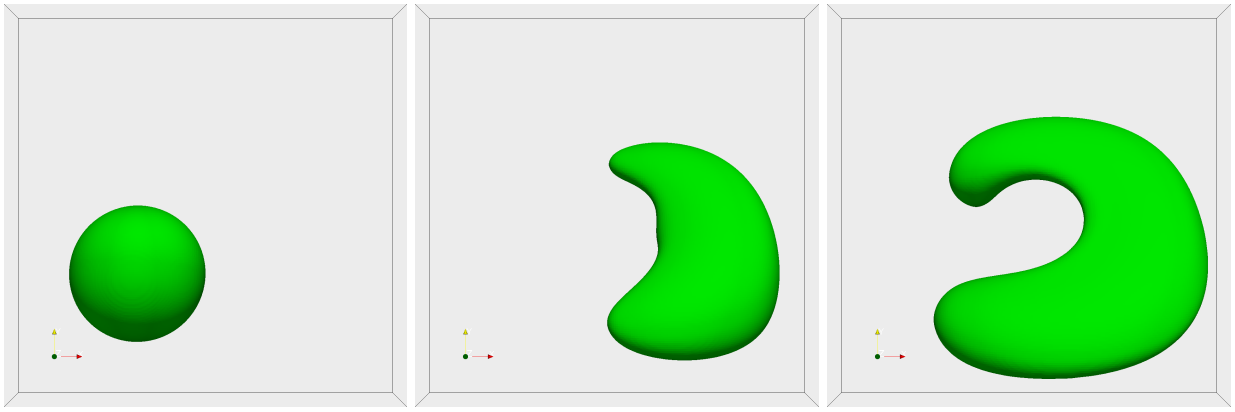
$$w(x,y,z,t) = -\sin^2(2\pi x)\sin(2\pi y)\sin^2(\pi z)\cos\left(\frac{\pi t}{T}\right) \tag{2.101}$$

で移流させる．ここで，t は時刻，T は周期である．この速度場により，球形プロファイルは $t=T/2$ まで引き伸ばされ，その後折返し，$t=T$ で元の球形に戻る．

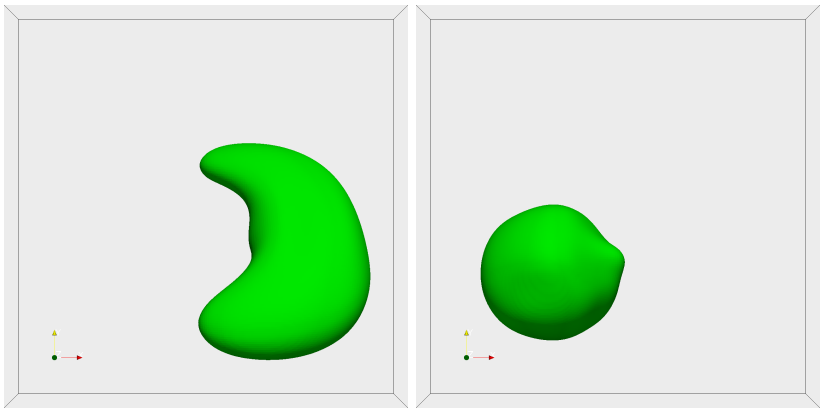
格子点数を $256\times256\times256$, 周期を $T=2$ と設定し，モビリティを $M=0.05$ として 1 周期分の計算を行った結果を Fig.2.12 に，モビリティを $M=0.01$ とした結果を Fig.2.13 に示す．周期の半分である $t=1$ までプロファイルが引き伸ばされ，$t=2$ で元の球形に近いプロファイルに戻っている．フェーズフィールド法を用い，界面の移流計算を適切に行えていることが確認できた．

次に，周期を $T=3$ と設定し，プロファイルをより薄く引き伸ばすテストを行う．周期を $T=3$ と設定し，モビリティを $M=0.05$ とした結果を Fig.2.25 に，モビリティを $M=0.01$ とした結果を Fig.2.15 に示す．プロファイルが最も引き伸ばされる $t=1.5$ において，どちらの条件でも界面がちぎれてしまっている．モビリティが大きいほど界面に空く穴が大きく，これは界面の拡散・逆拡散の影響が強くなり，プロファイルを集める効果が大きくなるためだと考えられる．また，モビリティが小さいほど $t=2$ のプロファイルが球形に近づいている．プロファイルが引き伸ばされ，界面同士が近づく条件では，モビリティの影響が大きいことが分かった．

Velocity extension 法の検証として，プロファイルの内側にのみ速度場を与え，外側の速度を Velocity extension 法で補外して移流計算を行う．格子解像度を $256\times256\times256$，周期を $T=3$，モビリティを $M=0.01$ と設定して 1 周期分の計算を行った結果を Fig.2.16 に示す．同条件で計算領域全体に速度場を与えた Fig.2.15 の結果とほぼ同じであり，Velocity extension 法で補外された速度場を用いて十分な精度で移流計算が可能であることが確認できた．

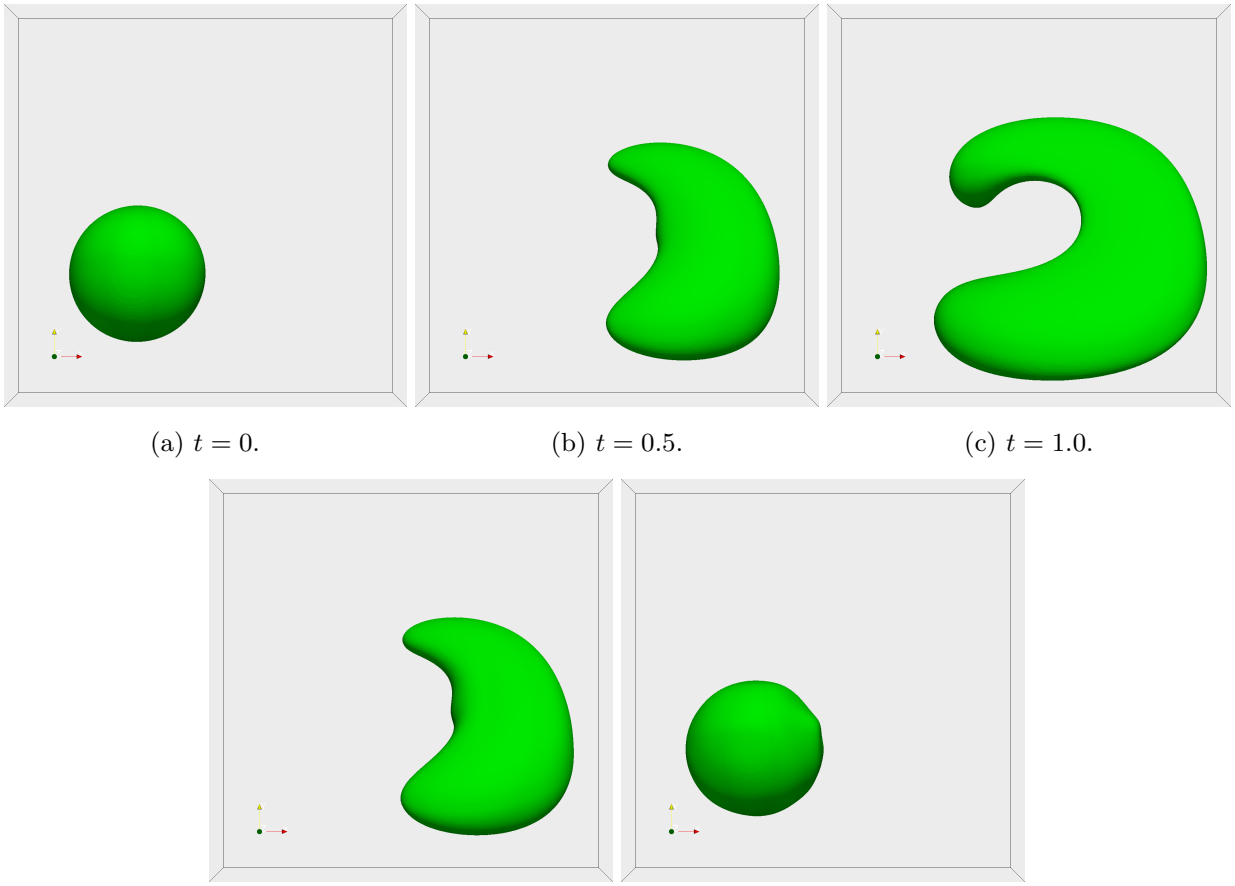


(a) $t = 0$. (b) $t = 0.5$. (c) $t = 1.0$.



(d) $t = 1.5$. (e) $t = 2.0$.

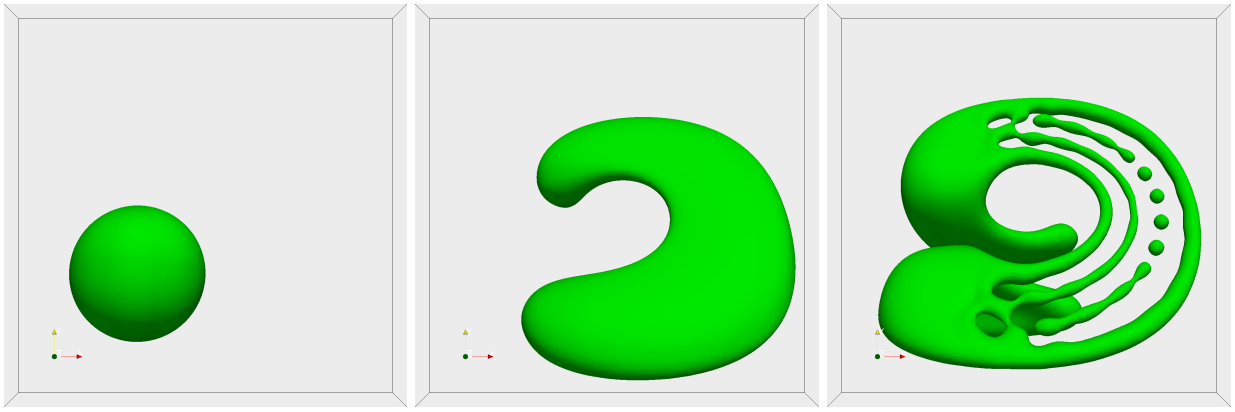
Fig.2.12: Three-dimensional advection problem with $T = 2$ and $M = 0.05$.



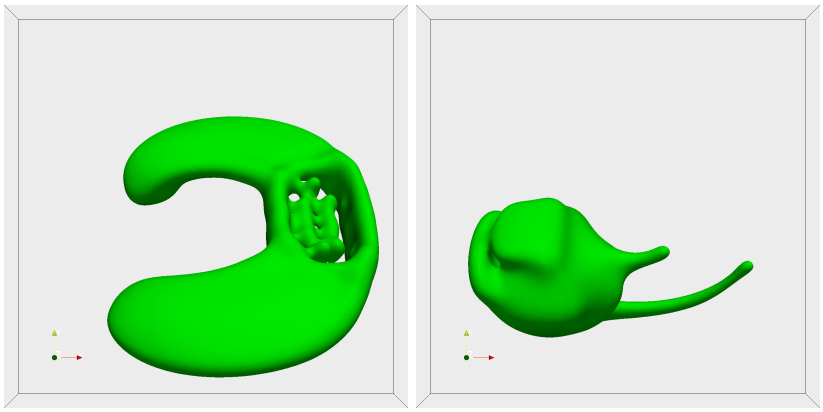
(a) $t = 0$.　(b) $t = 0.5$.　(c) $t = 1.0$.

(d) $t = 1.5$.　(e) $t = 2.0$.

Fig.2.13: Three-dimensional advection problem with $T = 2$ and $M = 0.01$.

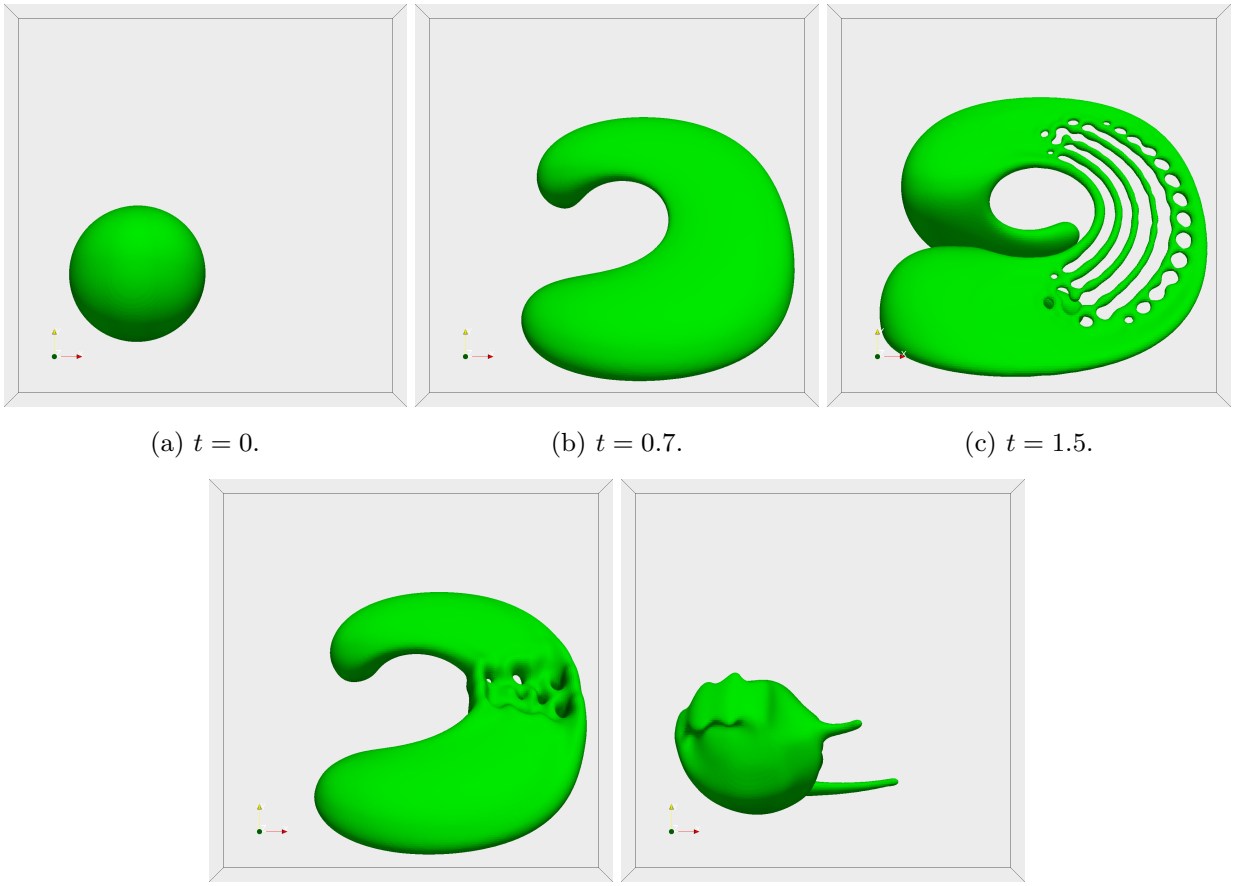


(a) $t = 0$. (b) $t = 0.7$. (c) $t = 1.5$.



(d) $t = 2.3$. (e) $t = 3.0$.

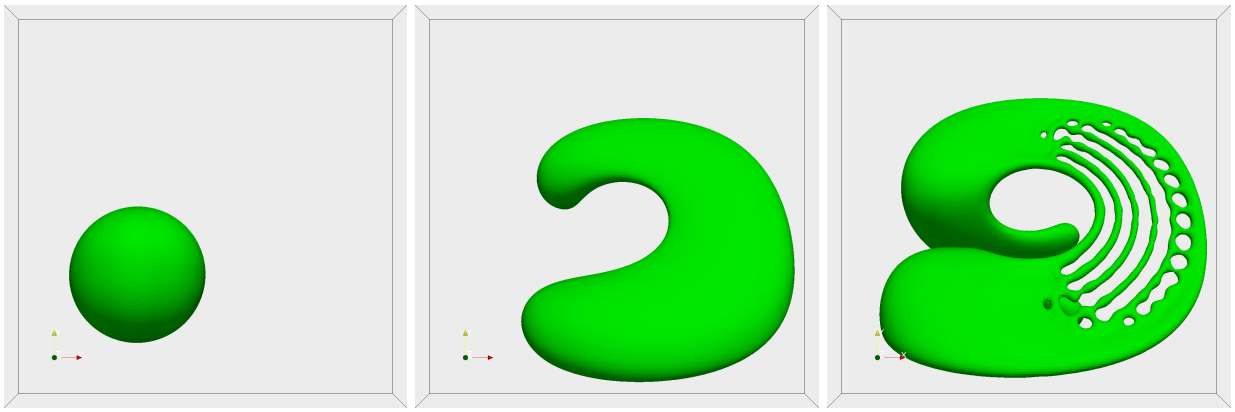
Fig.2.14: Three-dimensional advection problem with $T = 3$ and $M = 0.05$.



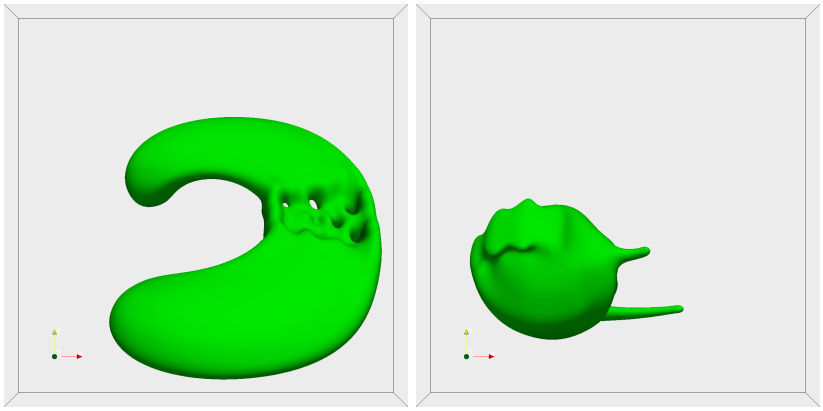
(a) $t = 0$.　(b) $t = 0.7$.　(c) $t = 1.5$.

(d) $t = 2.3$.　(e) $t = 3.0$.

Fig.2.15: Three-dimensional advection problem with $T = 3$ and $M = 0.01$.



(a) $t = 0$. (b) $t = 0.7$. (c) $t = 1.5$.



(d) $t = 2.3$. (e) $t = 3.0$.

Fig.2.16: Three-dimensional advection problem with $T = 3$ and $M = 0.01$ using the velocity extension method.

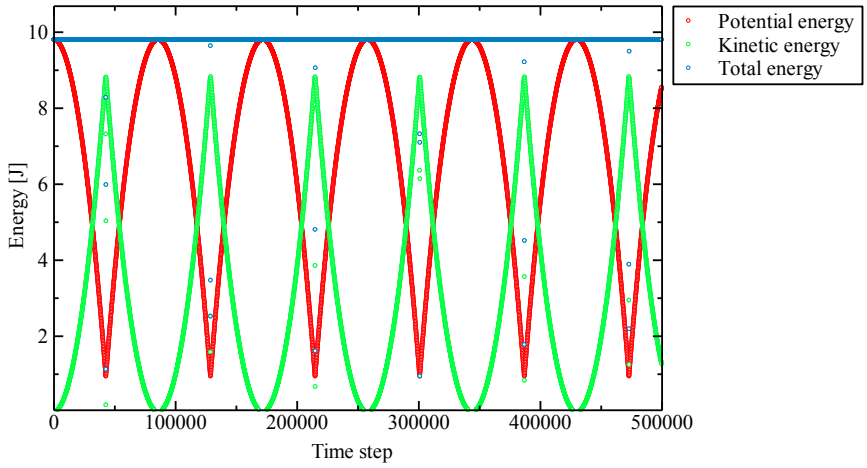


Fig.2.17: Total energy conservation in elastic collision of a free falling particle.

2.5.3　個別要素法のバネ-ダッシュポットモデルの検証

個別要素法のバネ-ダッシュポットモデルの検証として，粒子を自由落下させて床に弾性衝突させ，弾性衝突の前後で粒子が持つエネルギーが保存するかを確認する．ダッシュポットの係数を 0 に設定することで，個別要素法で弾性衝突を仮定することができる．粒子半径を 0.1 m，粒子質量を 1.0 kg，時間刻み幅を 1.0×10^{-5} s，バネ定数を 1.0×10^{6} N/m と設定し，1.0 m の高さから球形粒子を自由落下させたときのエネルギー変化を Fig.2.17 に示す．縦軸はエネルギー，横軸はタイムステップ数である．青色が全エネルギー，赤色が位置エネルギー，緑色が運動エネルギーである．粒子の自由落下では，位置エネルギーは減少，運動エネルギーは増加していて，その合計の全エネルギーは保存していることが確認できる．45,000 ステップ付近で粒子は床と衝突し，今度は位置エネルギーは増加し，運動エネルギーは減少している．その後，床と複数回衝突しても全エネルギーが保存しており，この結果から個別要素法のモデルで弾性衝突を扱えることが確認できる．

次に，非球形粒子モデルを用いた場合に，弾性衝突の前後でのエネルギー保存を確認する．半径 0.05 m，質量 0.2 kg の球形粒子を L 字型に 9 個並べて非球形粒子を作成し，初期状態では 10.0π rad/s の角速度を与えている．床との衝突は弾性衝突を仮定し，ダッシュポットの定数は 0 としている．衝突の前後において，位置エネルギーと並進運動のエネルギー，回転のエネルギーの和が保存するかを確認する．エネルギーの時間変化を Fig.2.18 に示す．位置エネルギーを赤色，並進運動のエネルギーを緑色，回転エネルギーを紫色，全エネルギーを青色で示す．2000 ステップから 2500 ステップの間で床面と接触していて，その前後で全エネルギーが保存していることが確認できる．

次に，バネ-ダッシュポットモデルで計算される力の大きさを評価するために，粒子が衝突する際の力を力積から計算し，バネ-ダッシュポットモデルと比較する．時間刻み幅 Δt に作用する力は物体の運動

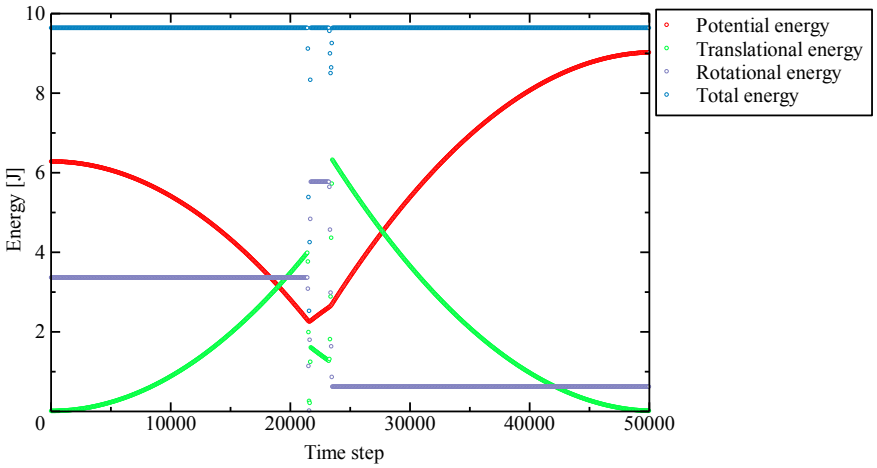


Fig.2.18: Total energy conservation in elastic collision of a free falling non-spherical particle.

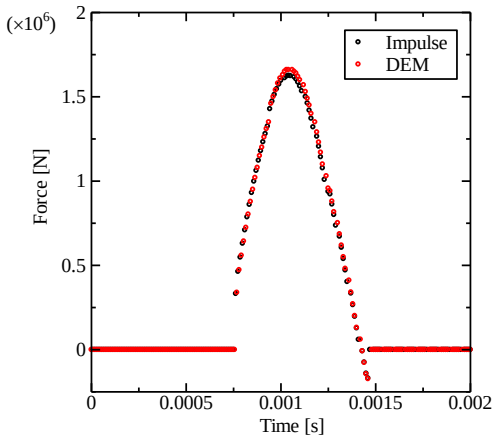


Fig.2.19: Comparison of the DEM interaction model and impulse.

量から

$$\boldsymbol{F}\Delta t = m\boldsymbol{v}\left(t+\Delta t\right) - m\boldsymbol{v}\left(t\right) \tag{2.102}$$

と計算できる．質量 $m = 8$ kg の直方体の物体を x 軸方向に速度 $v = 50$ m/s で移動させ，壁面に衝突したときに物体に作用する x 軸方向の力を評価する．時間刻み幅は $\Delta t = 1.2 \times 10^{-5}$s と設定した．計算結果を Fig.2.19 に示す．黒が力積から計算した力，赤が DEM のモデルから計算した力であり，両者がよく一致していることが確認できる．

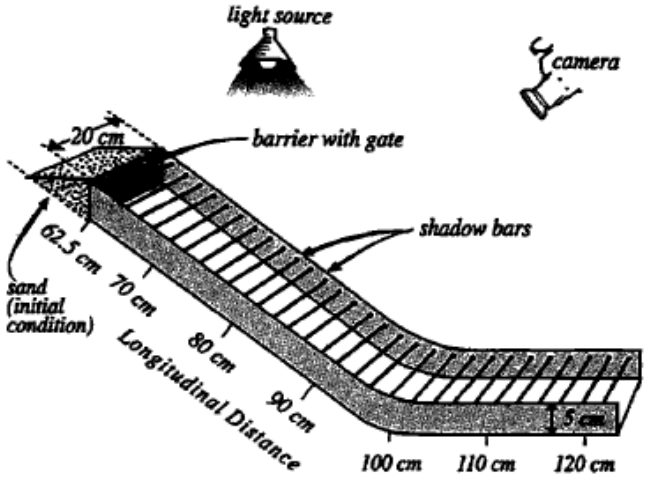


Fig.2.20: The experimental set-up of dry sand flow experiments. (Adapted from [84])

2.5.4　斜面を滑り落ちる砂の計算

非球形粒子を用いた個別要素法で，既往研究 [84] で実験が行われた砂が斜面を滑り落ちる問題の計算を行い，既往研究の実験結果と比較する．実験装置の概要を Fig.2.20 に示す．斜面の傾斜は 31.4 度であり，斜面の幅は 20 cm である．時刻 Time = 0 s においてゲートを開けて斜面に砂を流し，粒子の位置を測定する．実験では直径が 0.5 mm の砂粒が用いられているが，計算コストとメモリ容量の制限から，今回は 3 mm 程度の非球形粒子で計算を行う．Fig.2.21 に示す 3 種類の非球形粒子をそれぞれ 4,000 個ずつ用いて行い，構成する球形粒子の数は左から順に 1,100 個，1,004 個，804 個である．計算に用いた球形粒子は合計で 11,632,000 個である．

時刻 Time = 0.10 s，Time = 0.32 s，Time = 0.53 s，Time = 0.93 s，Time = 1.50 s における既往研究 [84] の実験結果と本研究の計算のスナップショットを Fig.2.22 に示す．実験結果は砂の厚さが等しい部分を結んだ等層厚線を表示している．計算結果の粒子分布は実験とほぼ一致していて，斜面の側面付近の粒子が壁の影響で中央付近よりも遅く流れていく様子や，斜面末端部に粒子が堆積する様子を再現できている．非球形粒子の個別要素法で定性的にではあるが実験と合う結果が得られた．

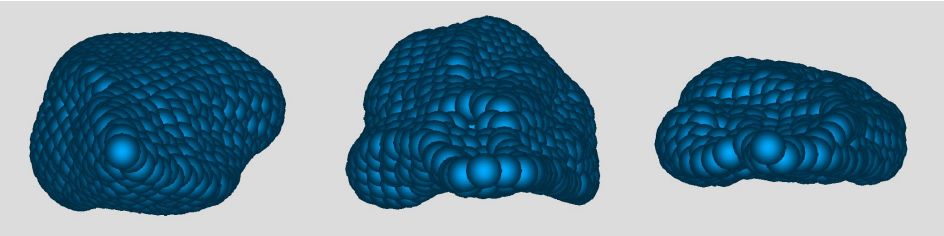


Fig.2.21: Non-spherical particle models used in the dry sand flow simulation.

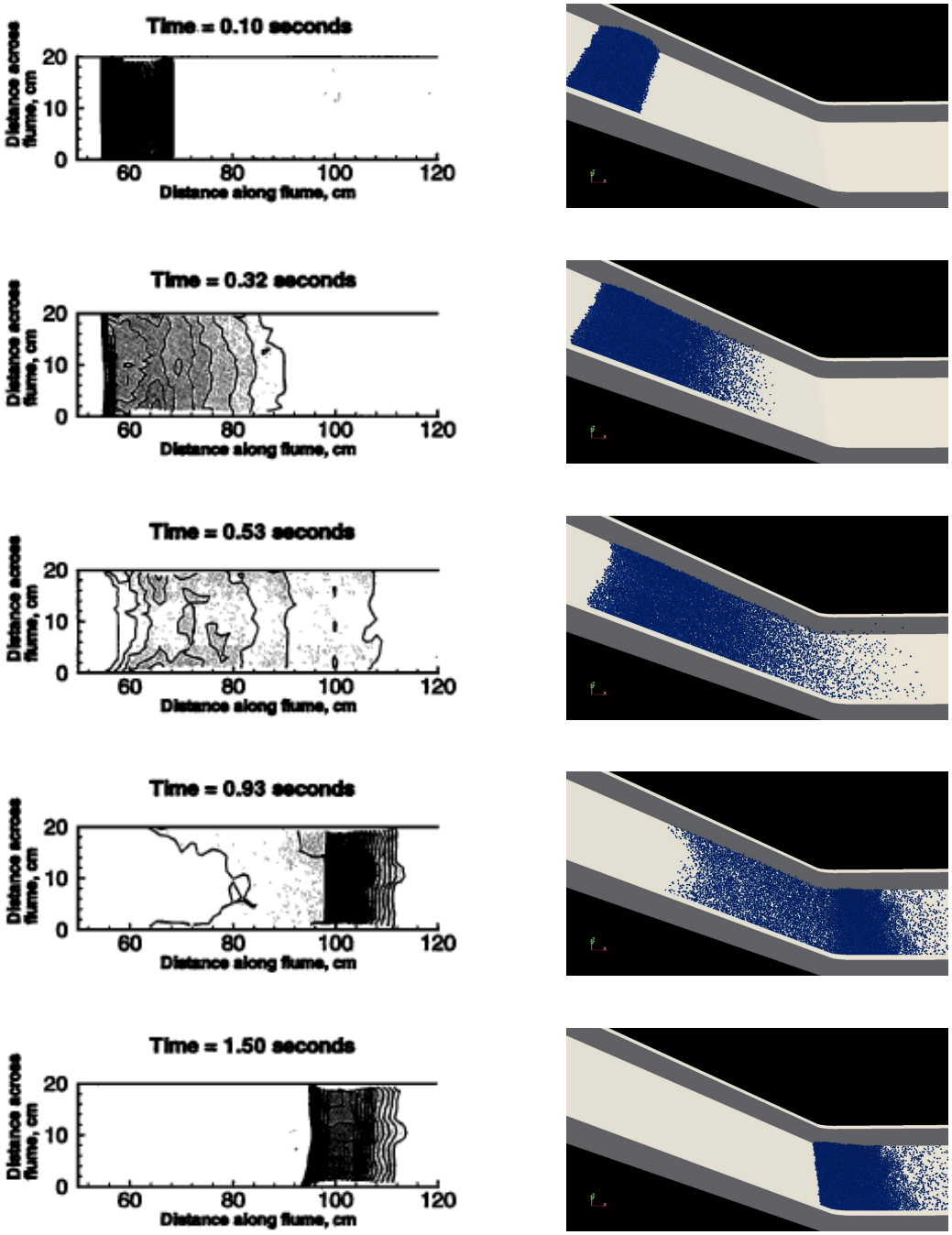


Fig.2.22: Experimental results[84] (left) and simulation results of dry sand flow (left figures adapted from [84]).

2.5.5 一定速度で移動する球の抗力係数

Interpolated bounce-back に基づく移動境界条件を用いた格子ボルツマン法の検証として，球を一定速度で移動させたときの抗力係数を計算し，風洞実験で得られている球の抗力係数の実験値と比較する．球の直径に対して 20 メッシュを割り当て，計算領域の格子点数は $256 \times 4096 \times 256$ と設定する，計算格子点数の多い y 軸方向に球を一定速度で移動させる．境界条件として，球に対しては non-slip 境界条件，計算領域の境界には non-slip 境界条件を課す．格子ボルツマン法の衝突モデルには MRT モデルを用い，LES を導入する．代表長さを粒子直径を D，代表速度を粒子の移動速度 U，動粘度を ν としてレイノルズ数を

$$Re = \frac{UD}{\nu} \tag{2.103}$$

と計算し，レイノルズ数が 1 から 100,000 の範囲で計算を行う．抗力係数 C_D は，球が受ける移動方向の力を F_y として，

$$C_D = \frac{F_y}{\frac{1}{2}\rho U^2 S} \tag{2.104}$$

で計算する．ここで，S は投影面積であり，球の場合は

$$S = \pi \left(\frac{D}{2}\right)^2 \tag{2.105}$$

である．

計算結果と実験結果の抗力係数の比較を Fig.2.23 に示す．赤い丸のプロットは LES モデルなしの計算結果，青い三角形のプロットは LES モデルありの計算結果，黒い実線は実験値のフィッティング曲線 [85] である．LES モデルなしの計算では，レイノルズ数 1 から 1000 まで実験値とよく一致した抗力係数の値となっている．しかし，レイノルズ数 1000 以上では計算が破綻してしまった．LES モデルを用いた場合は，レイノルズ数 1000 以上でも安定して計算でき，抗力係数は実験値とよく一致している．Interpolated bounce-back に基づく移動境界条件により，低レイノルズ数から高レイノルズ数まで，流体が球に与える力を十分な精度で計算できていることが確認できる．

2.5.6 球の沈降速度

流体-構造連成の検証として，単一球形粒子の沈降計算を行い，沈降速度を既往研究 [86] で行われた実験と比較する．計算領域は 100 mm × 100 mm × 160 mm，球の直径は 15 mm，球の重心の初期位置は床から 127.5 mm である．格子解像度を 1 mm とし，粒子直径に対して 15 格子を割り当てて計算する．重力加速度を 9.8 $\mathrm{m/s^2}$ とし，壁面に non-slip 境界条件を課す．固体粒子の密度は $1,120\ \mathrm{kg/m^3}$ である．流体の密度と粘度の条件を Table2.1 に示す．実験で得られた終端速度に対するレイノルズ数が 1.5, 4.1, 11.6, 31.9 の条件で計算する．

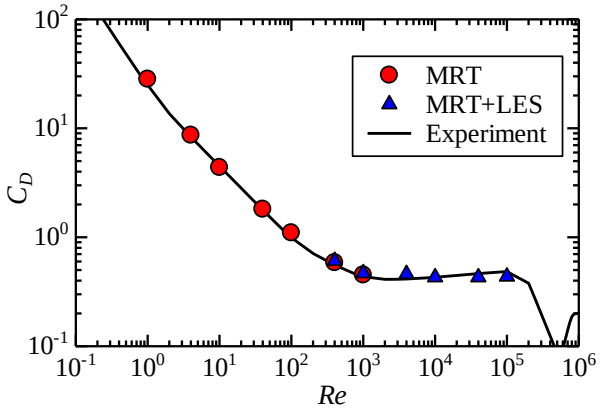


Fig.2.23: The drag coefficient of a moving sphere as a function of the Reynolds number.

Table2.1: Conditions of the sedimentation simulation.

Case number	ρ [kg/m^3]	μ [Ns/m^2]	Re
Case 1	970	373×10^{-3}	1.5
Case 2	965	212×10^{-3}	4.1
Case 3	962	113×10^{-3}	11.6
Case 4	960	58×10^{-3}	31.9

計算結果と実験結果の沈降速度の時間変化の比較を Fig.2.24 に示す．実線が計算結果であり，プロットが実験結果 [86] である．粒子は重力に加速され，沈降速度が一定になる終端速度に達する．床が近づくと，床の影響を受けて粒子の沈降速度が遅くなり，床に衝突して粒子は停止する．どのレイノルズ数の条件においても計算結果と実験結果がよく一致しており，本手法の妥当性を確認できる．

2.5.7　ダム崩壊計算

自由界面流れの検証として，濡れた床へのダム崩壊計算を行う．流体は水を仮定し，0.72 m × 0.12 m × 0.36 m の計算領域を設定する．床に水深 0.018 m の浅い水面を設定し，計算領域の端に 0.15 m × 0.12 m × 0.36 m の水注を設置する．重力加速度を 9.8 m/s^2，格子幅は 0.9375 mm である．時間刻み幅は 2.8125×10^{-6} s であり，物理時間で 1.475 s の計算を行う．壁面の境界条件として，滑りなし境界条件を課す．保存型 Allen-Cahn 方程式のモビリティを $M = 0.05$ と $M = 0.01$ と設定し，拡散・逆拡散項の影響を調べる．

計算結果を Fig.4.41 に示す．ダム崩壊とともに砕波が生じているのが確認でき，これは先行研究の実験で報告されている現象であり [87]，本計算結果は濡れた床へのダムブレーク問題の特徴をよく捉えた計算が行えている．また，モビリティを小さくすると，波が壁に衝突した際により小さい液滴が発生している．モビリティが大きいと拡散・逆拡散の影響が大きくなり界面を集める効果が強くなるためだと

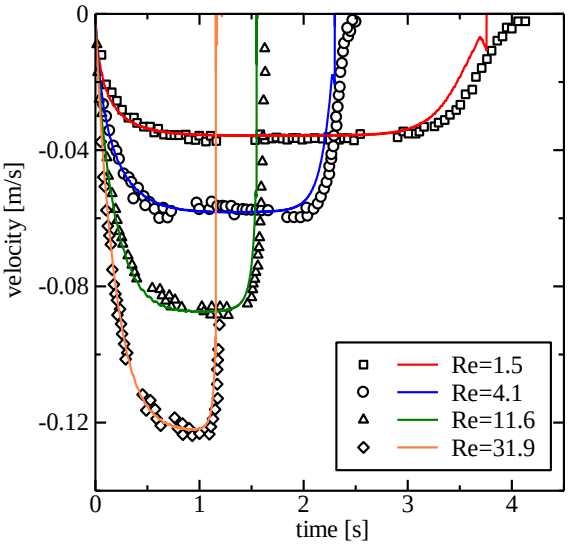


Fig.2.24: Comparison between experimented (plots) and simulated (solid lines) sedimentation velocity of a single sphere.

考えられる．一方，波が壁にぶつかるタイミングや水面高さなどの飛沫以外に関しては，モビリティによる影響は小さい．このことから，飛沫などの現象に注目したい場合はモビリティを小さく設定し，微小な液滴が重要でない流れ場や水面高さなどに注目する場合はモビリティを大きく設定すれば良いことが確認できた．

2.5.8 障害物を含むダム崩壊計算

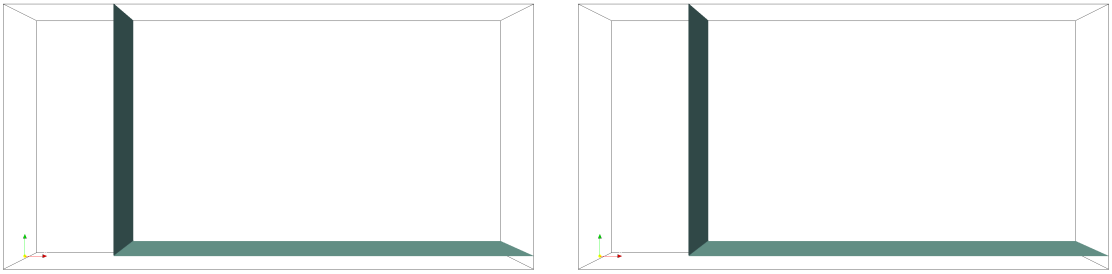
固定された物体を含む自由界面流れの検証として，Kleefsman らによって行われた障害物を含むダム崩壊問題の実験 [88] と同じ条件で数値シミュレーションを行った．Fig.2.26 に示すように計算領域を 3.22 m × 1.0 m × 1.0 m とし，右端から 1.22 m の領域に高さ 0.55 m の水柱を配置した．左端から 0.744 m の位置に大きさ 0.16 m × 0.4 m × 0.16 m の直方体の障害物を設置した．格子ボルツマン法の格子解像度は 10 mm であり，計算格子点数は 322 × 100 × 100 である．流体は水を仮定し，密度は 1000 kg/m^3，動粘度は 1.0×10^{-6} m^2/s である．壁には滑りなし境界条件を課した．格子ボルツマンに SRT モデルを用い LES を導入した計算，キュムラントモデルを導入した計算を行う．

キュムラントモデルを用いた場合の計算結果のスナップショットを Fig.2.27 に示す．水色の等値面が自由界面，黒色の等値面が障害物の表面を示す．実験と同様に障害物により自由界面が跳ね上げられる様子が確認できる．

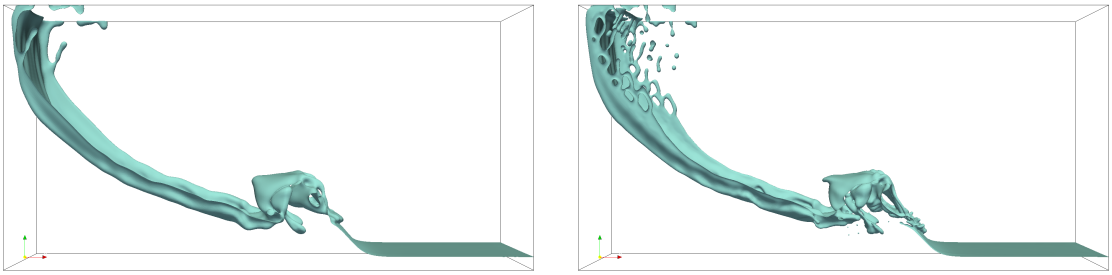
Fig.2.26 の H1，H2，H3，H4 における水面高さの時間変化のグラフを Fig.2.28 に示す．黒線が Kleefsman らによる実験結果，青線が SRT+LES モデルによる結果，赤色がキュムラントモデルによる結果である．どちらの衝突モデルにおいても水面高さは実験と計算で概ね一致しているが，壁で跳ね返る波が実験よりも若干遅れているのが確認できる．このような傾向は，他の界面捕獲手法を用いた格子

ボルツマン法による計算でも現れていおり [89]，本研究の結果は妥当だと考えられる．

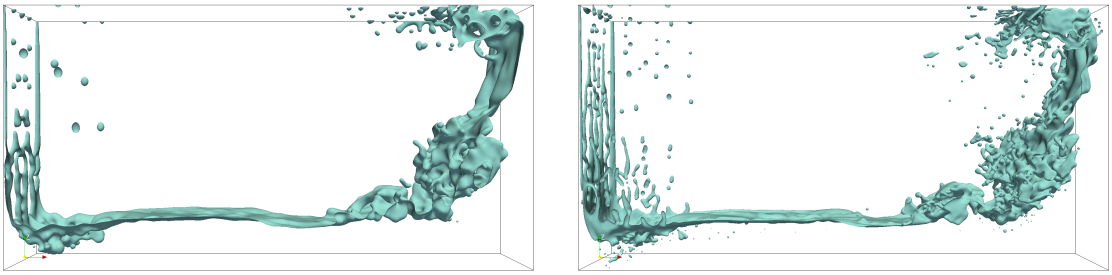
物体表面の圧力の時刻歴を Fig.2.29 に示す．青色の SRT モデルに LES を導入した計算では，物体表面の圧力が大きく振動していることが確認でき，これは速度場が振動しているのが原因だと考えられる．赤色のキュムラントモデルを用いた計算では，圧力の振動を SRT モデルよりも抑えられており，黒色の実験結果と概ね一致した圧力となった．このことから，キュムラントモデルを用いることでダム崩壊問題のような高レイノルズ数の問題をより正確に計算できることが確認できた．$t = 5$ s 付近で圧力が実験よりも遅れているが，このような傾向は既往研究でも報告されている [90]．今後，解像度を変えた計算などを行い，実験よりも波が遅れる原因を検討する予定である．



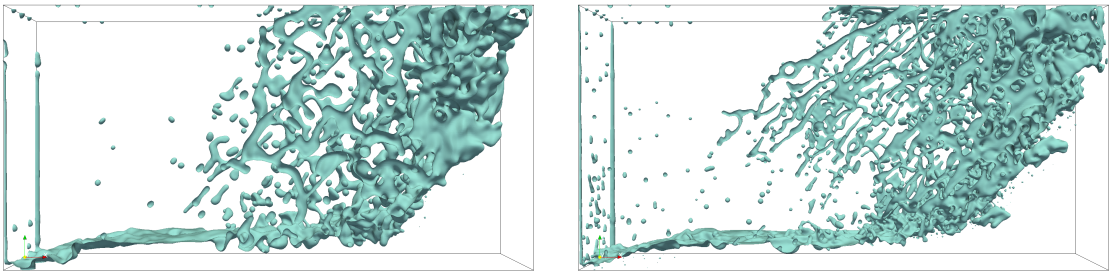
(a) $t = 0.0$ s (left:$M = 0.05$, right:$M = 0.01$).



(b) $t = 0.2304$ s (left:$M = 0.05$, right:$M = 0.01$).

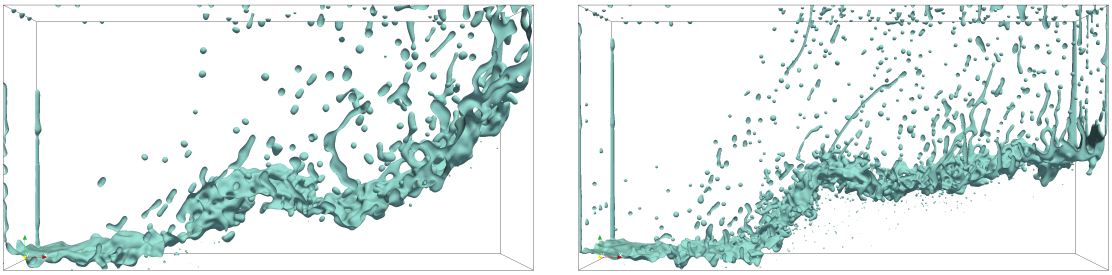


(c) $t = 0.4608$ s (left:$M = 0.05$, right:$M = 0.01$).

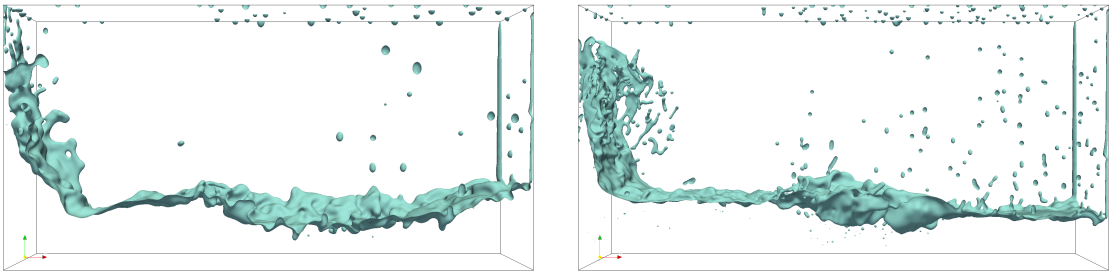


(d) $t = 0.6912$ s (left:$M = 0.05$, right:$M = 0.01$).

Fig.2.25: Simulation results of a dam breaking on a wet floor.



(e) $t = 0.9216$ s (left:$M = 0.05$, right:$M = 0.01$).



(f) $t = 1.152$ s (left:$M = 0.05$, right:$M = 0.01$).

Fig.2.25: Simulation results of a dam breaking on a wet floor. (Continued.)

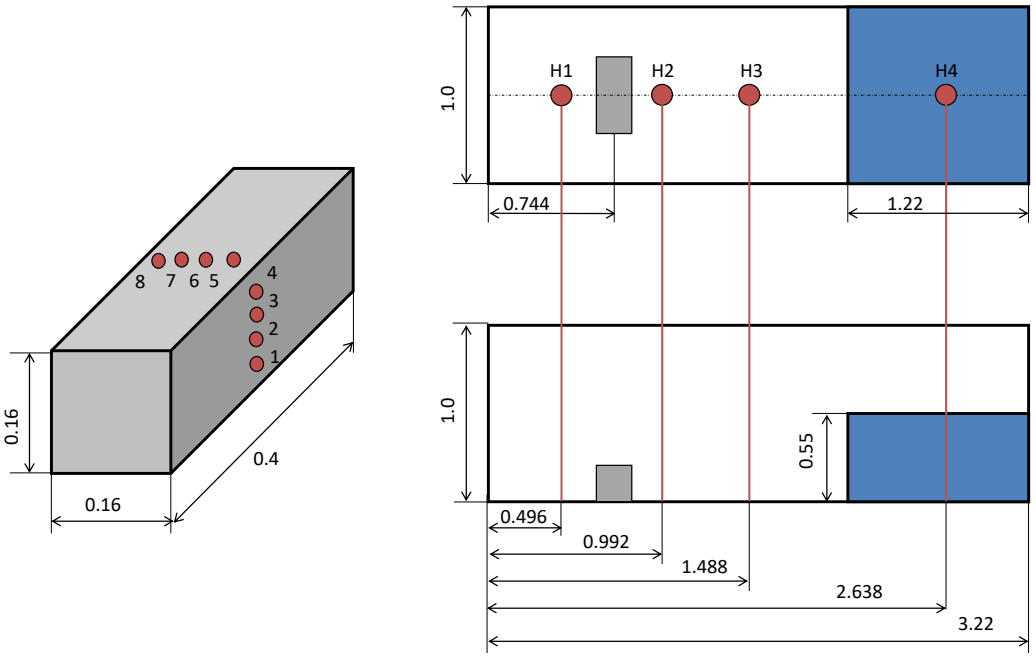


Fig.2.26: The setup of the breaking dam problem.

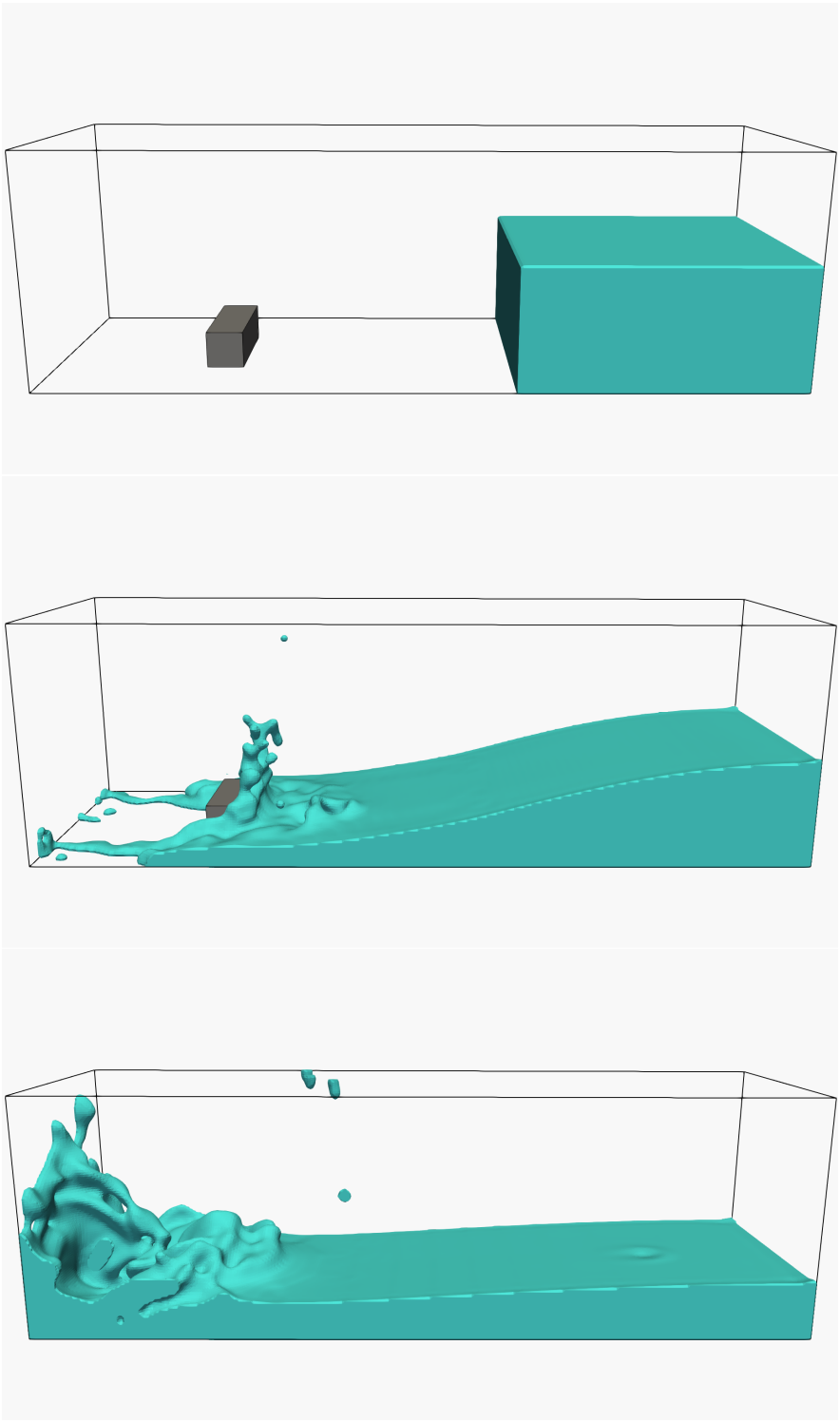


Fig.2.27: Snapshots of the breaking dam simulation using the cumulant LBM at 3 physical time instants t=0.0, 0.6 and 1.2 s (from the upper panel to the lower panel).

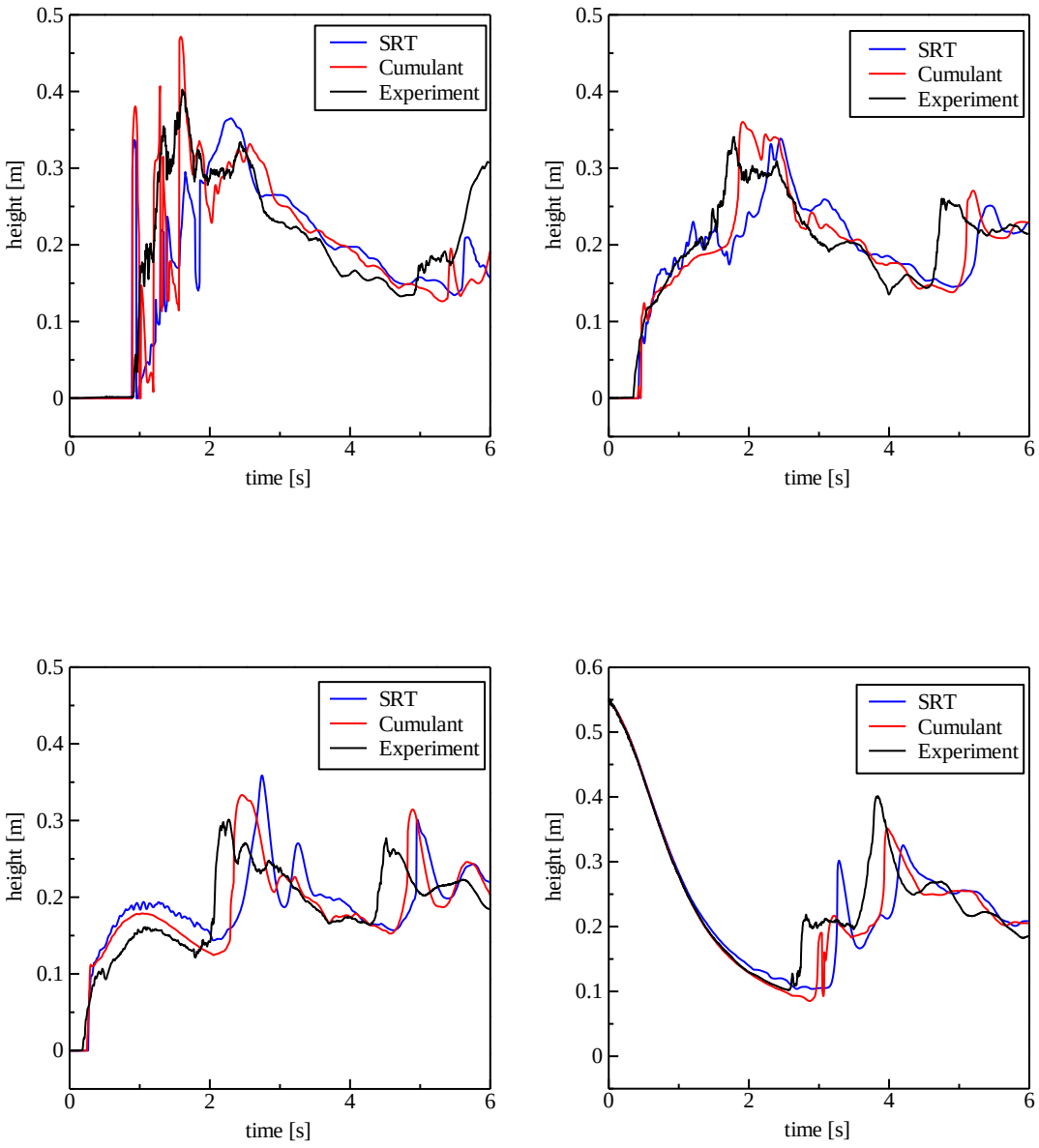


Fig.2.28: Time histories of the water height (upper left: H1, upper right: H2, lower left: H3, lower right: H4).

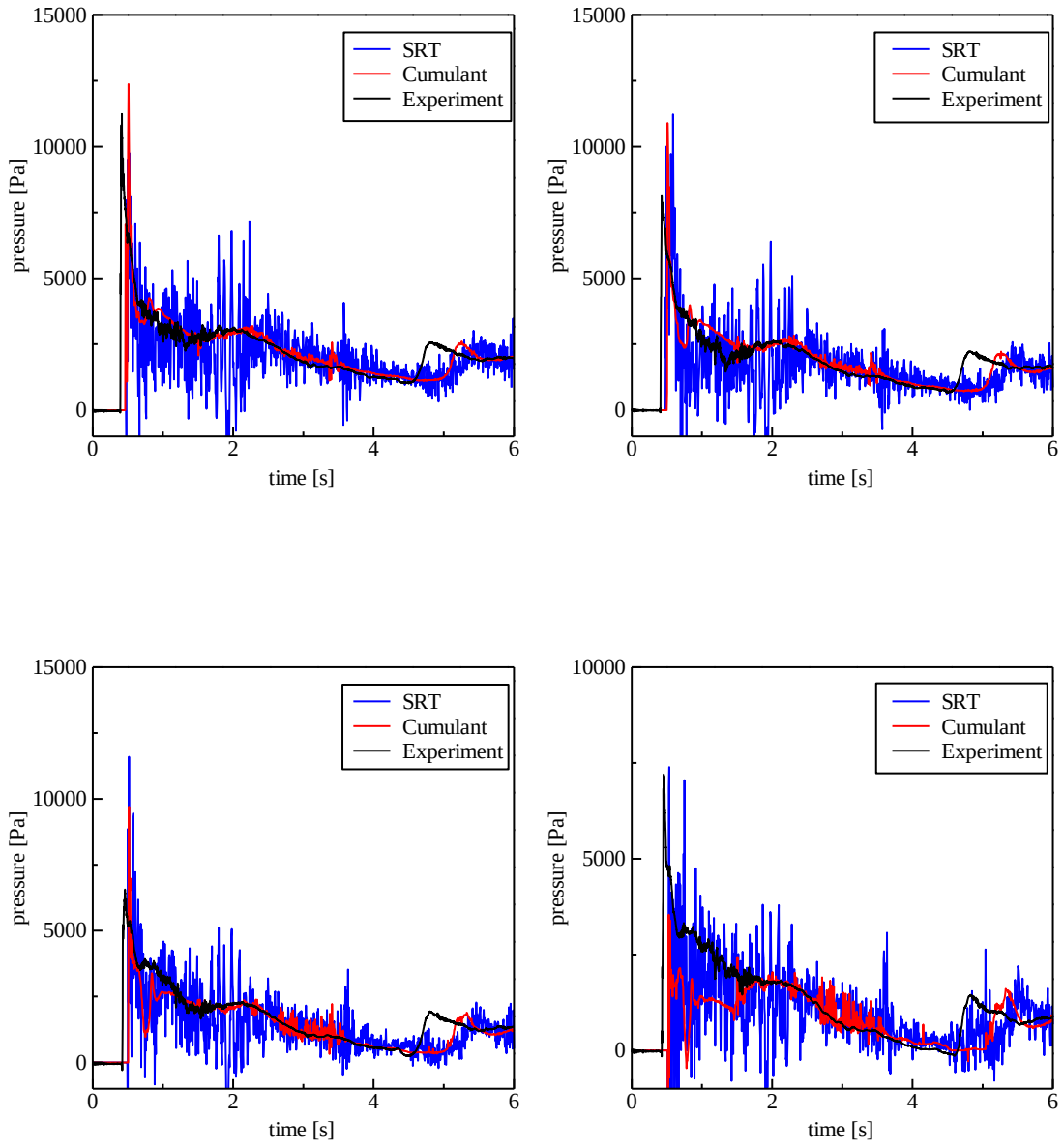


Fig.2.29: Time histories of the pressure on the object surface (upper left: P1, upper right: P2, lower left: P3, lower right: P4).

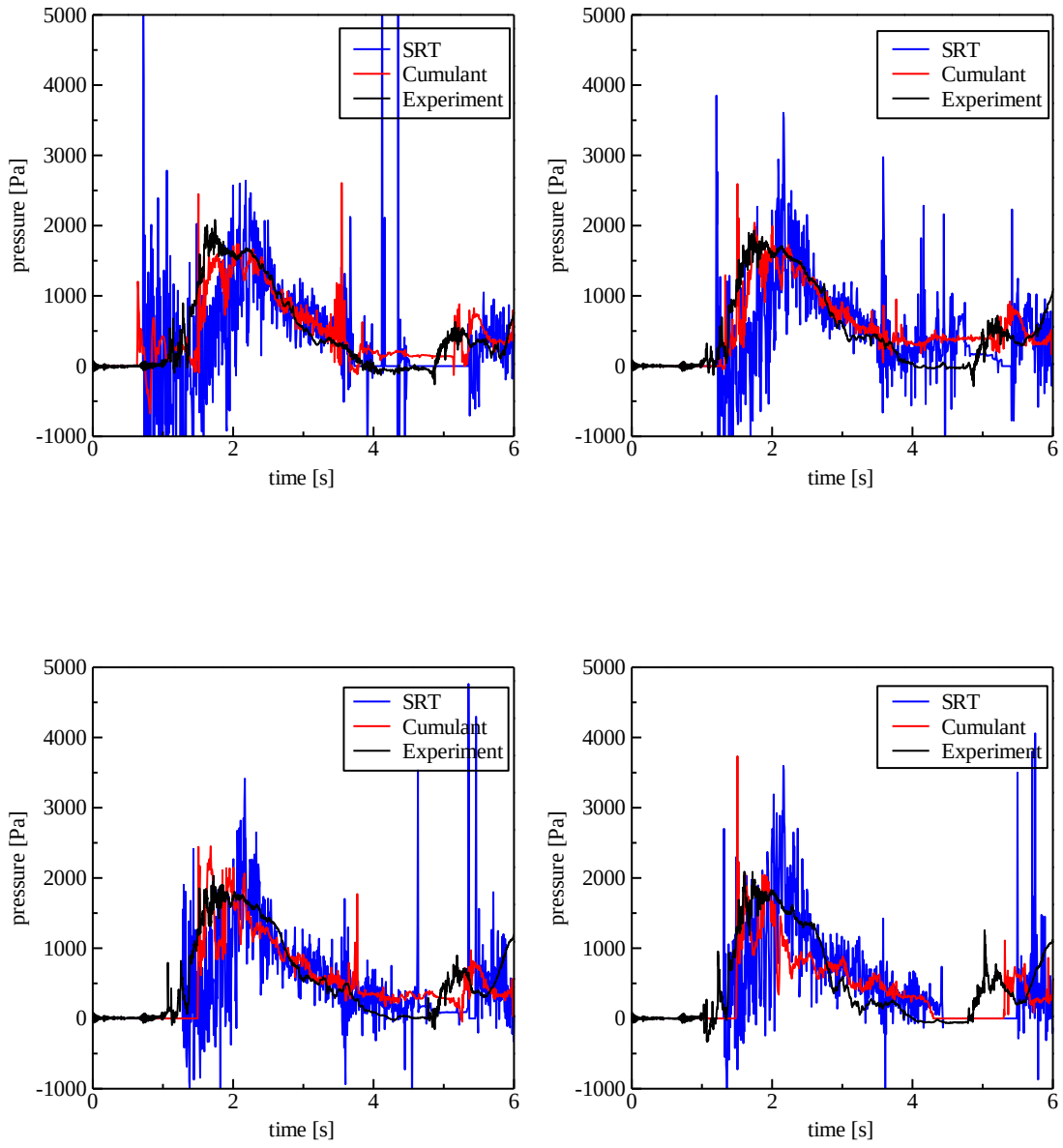


Fig.2.29: Time histories of the pressure on the object surface (upper left: P5, upper right: P6, lower left: P7, lower right: P8). (continued)

2.5.9 円柱の自由界面への落下

自由界面流れと固体の連成計算の検証として，自由表面に円柱を落下させるシミュレーションを行い，先行研究の実験結果 [91] と水面形状と円柱の位置を比較する．実験では深さ 0.3 m の水に，直径 0.11 m で密度 500 kg/m^3 の円柱を，円柱の中心が水面からの 0.5 m の位置から落下させている．本計算では，着水時の円柱の速度が実験と等しくなるように，円柱の中心が水面から 0.1 m の位置から初速 2.8 m/s で落下させる．シミュレーションの初期時刻を $t = 0.286$ s と設定する．壁面には滑りなし境界条件を課す．

計算結果を Fig.2.30 に示す．自由表面の形状は先行研究の実験 [91] や他の手法での計算結果 [92] とよく一致している．Fig.2.31 は円柱の位置の時間変化のグラフであり，計算結果と実験結果がよく一致している．本計算手法で自由界面流れと固体の連成を適切に計算できることが確認できた．

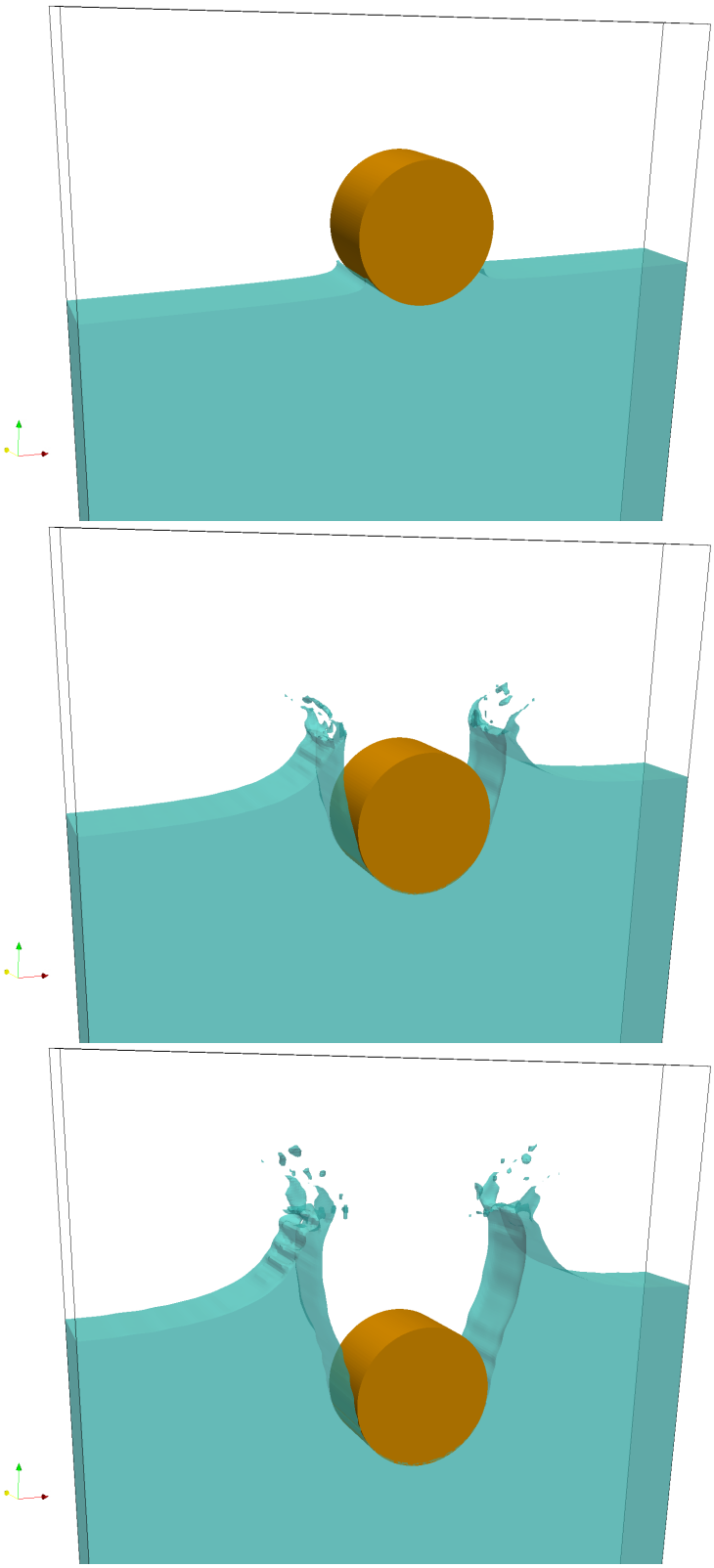


Fig.2.30: Snapshots of the water entry simulation of a cylinder.

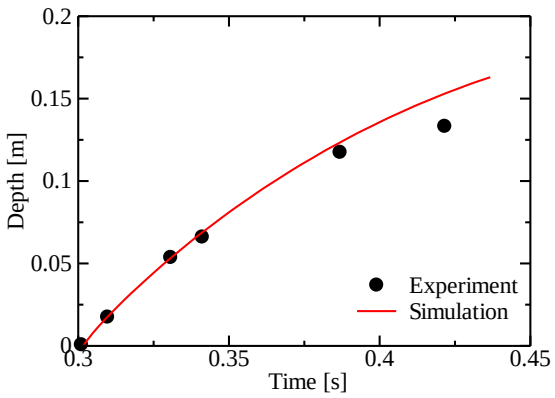


Fig.2.31: The time history of the cylinder position.

2.6　格子ボルツマン法に基づく混相流解析手法の開発のまとめ

本章では，混相流計算の高速化・大規模化に向けた，圧力のポアソン方程式を解く必要がない完全な陽解法計算である格子ボルツマン法に基づく混相流解析手法の開発を行った．格子ボルツマン法による単相流解析，フェーズフィールド法による界面捕獲，個別要素法による固体粒子と複雑形状の物体の計算を組み合わせた，自由界面を含む混相流の解析手法を構築した．

具体的には，単純な SRT モデルの格子ボルツマン法では高レイノルズ数で不安定になるため，MRT やキュムラントの衝突モデル，LES 乱流モデルを混相流計算の導入することで計算を安定化させた．壁面境界を精度良く扱える Interpolated Bounce-back 法に基づく流体と物体の相互作用計算を導入した．高密度比かつ高レイノルズ数の気液二相流は計算が不安定になるため，格子ボルツマン法で液相領域の速度場のみを計算し，気相領域には Velocity extension により速度場を補外することで，高レイノルズ数の自由界面を含む流れの計算を可能にした．個別要素法には複数の球形粒子を剛体連結した非球形粒子モデルを導入し，複雑形状の固体物体を混相流解析を可能にした．

提案した混相流解析手法の検証として，典型的なベンチマーク問題を行った．3 次元速度場での界面移流計算では，保存型 Allen-Cahn 方程式により十分な精度で移流計算を扱えることを確認するとともに，拡散・逆拡散項のパラメータに界面が分裂する際の挙動が変わることを確認した．非球形粒子を用いた個別要素法で斜面を砂が滑り落ちる計算を行い，先行研究の実験と比較し，砂の空間分布をよく再現できた．一定速度で移動する球の抗力係数や単一球形粒子の沈降速度を実験と比較し，Interpolated Bounce-back 法により十分な精度で流体と物体の相互作用を計算可能であることを示した．濡れた床へのダム崩壊計算では，キュムラントモデルを用いることで激しい流れを安定に計算可能であることを確認し，ダム崩壊により砕波が生じる過程を再現することに成功した．また，保存型 Allen-Cahn 方程式のモビリティの大きさにより液滴の飛沫の大きさが変わることが明らかになった．障害物を含むダム崩壊計算では，キュムラントモデルを導入することで圧力振動を大幅に抑えられ，水面の高さや障害物表面の圧力が Kleefsman らの実験とよく一致することを確認した．円柱の自由界面の落下の計算では，自由界面の形状や円柱の位置の時間変化が計算と実験でよく一致した結果が得られ，提案手法により自由界面流れと物体の連成計算を妥当に行えることを示した．以上の結果より，提案した格子ボルツマン法に基づく混相流の解析手法は，高レイノルズ数の激しい流れ場においても安定して十分な精度で計算可能であることを示した．

第 3 章

複数 GPU を用いた大規模混相流解析

3.1 GPU コンピューティング

画像表示用に開発された GPU を科学技術計算に利用する GPU コンピューティングが盛んに行われるようになり，スパコンにも GPU が搭載されるようになっている．本研究では，NVIDIA 社の GPU を利用し，C 言語を拡張した統合開発環境である CUDA を用いる．粒度の高い並列計算を実行する数千個の演算コアや階層的なメモリ構造など，GPU のアーキテクチャは CPU と大きく異なる．そのため，GPU の演算性能を発揮するには，GPU のアーキテクチャを意識したプログラミングが必要である．本節では，GPU のアーキテクチャや CUDA による並列計算の概念について述べる．

3.1.1 ホストとデバイス

CUDA プログラミングでは，CPU を動作させるホストコードと GPU を動作させるデバイスコードから構成される．GPU はマザーボードと PCI Express で接続されている．CPU と GPU のメモリは独立していて，GPU は CPU のメモリに直接アクセスできない．そのため，CPU は GPU のメモリ領域の確保・開放を行い，必要な CPU メモリ上のデータを GPU に転送する．GPU の演算コア（CUDA コア）から GPU 上のメモリ（グリーバルメモリ）へのアクセス速度は最新の Tesla V100 で 900 GB/s であり，それに比べて PCI Express を介して行われるマザーボード上のメイン・メモリへのアクセス速度は 8 GB/s と非常に遅いため，GPU と CPU 間の通信を極力抑える必要がある．そのため，GPU で高効率に計算するには，計算に必要なデータを GPU のグローバルメモリに確保し，メイン・メモリへのデータ出力以外，全ての計算ルーチンを GPU 上で実行することが重要である．また，GPU のグローバルメモリの容量は 16 GB であり，CPU のメインメモリよりも非常に少ない．

3.1.2 CUDA による並列プログラミング

GPU での並列処理は，グリッド，ブロック，スレッドという実行単位で階層的に管理されている．スレッドが最小の実行単位であり，GPU の計算では大量のスレッドが生成され，関数に記述された処理を各スレッドが実行していく．CUDA では Fig.3.1 のようにスレッドの管理は階層的に行われている．複

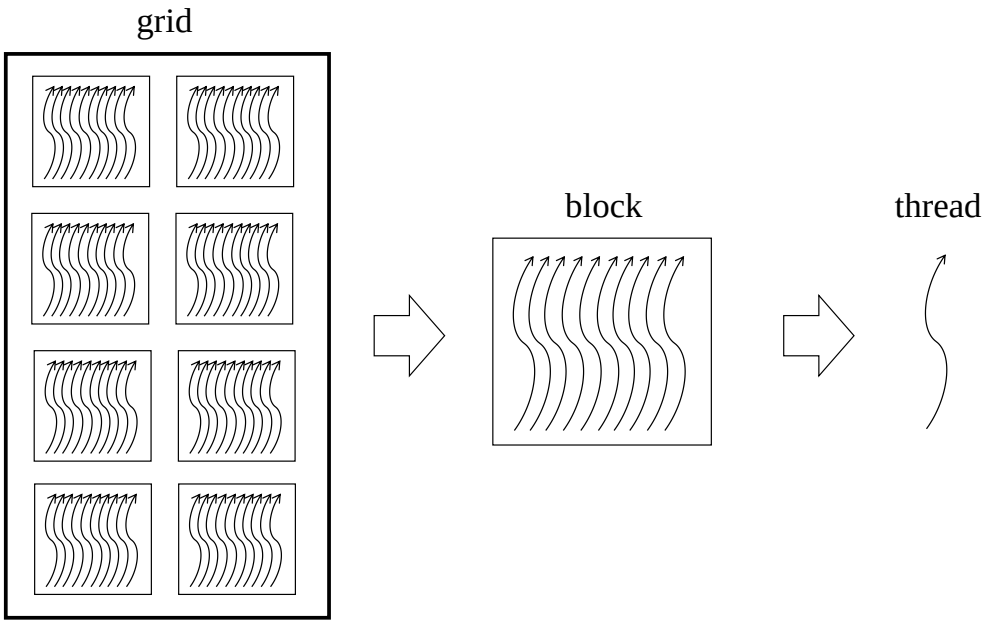


Fig.3.1: Concepts of a grid, blocks and threads in CUDA.

数のスレッドからなるブロック，その上の階層で複数のブロックをまとめるグリッドの二階層の構造で管理している．つまり，一つの関数で生成されたスレッドをすべてまとめたものがグリッドとなる．同じブロックに所属しているスレッドは協調して動作し，ブロック内のスレッドの同期と共有のメモリ領域（シェアードメモリ）を利用することができる．スレッドを識別するために，各スレッドにはグリッド内のブロックのインデックスである `blockIdx`，ブロック内のスレッドのインデックスである `threadIdx` が与えられる．グリッド内のブロックの数は `gridDim`，ブロック内のスレッドの数は `blockDim` である．また，32 個のスレッドをまとめたものをワープといい，ワープ内のスレッドは完全に同期されて実行される．

3.1.3　GPU の階層的なメモリ構造

GPU のメモリは Fig.3.2 のように階層的な構造になっており，CUDA で利用できるメモリには

- グローバルメモリ
- レジスタ
- シェアードメモリ
- ローカルメモリ
- コンスタントメモリ
- テクスチャメモリ

の 6 種類がある．グローバルメモリは最も容量の大きなメモリであり，全てのスレッドから読み込みと

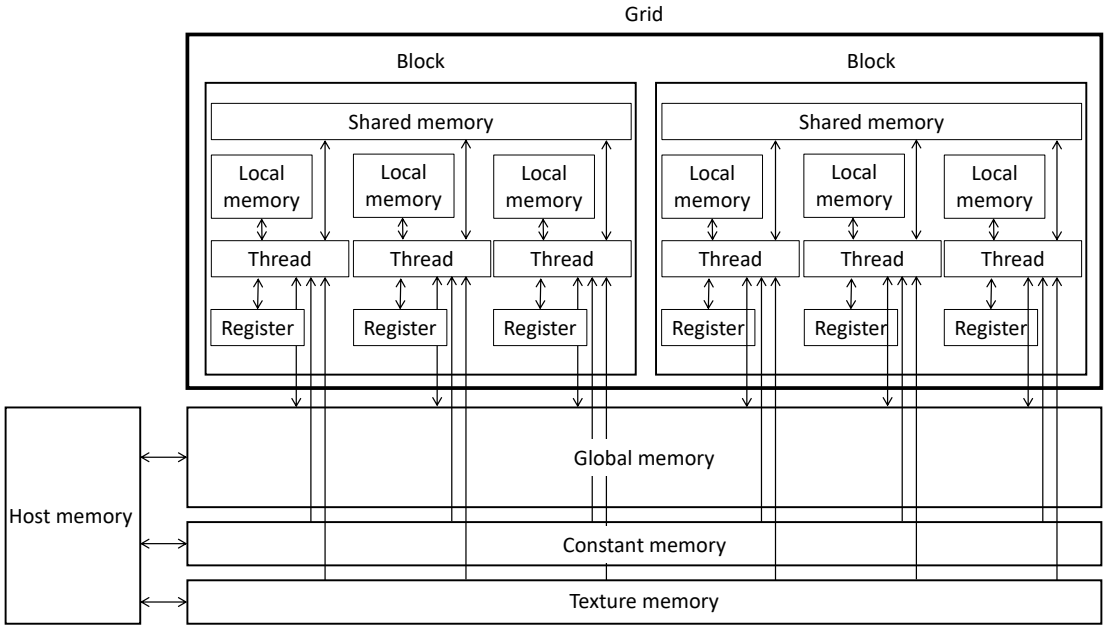


Fig.3.2: The hierarchical memory structure of GPU.

書き込みが可能である．メモリアクセス速度は最も遅く，効率的にアクセスできる条件も限られている．レジスタは各スレッドがもつメモリ領域であり，基本的には他のスレッドが利用するレジスタを参照することはできない．アクセス速度が最も速く，メモリ容量が最も少ないメモリ領域であり，グローバルメモリからロードしたデータの一時的保持などに利用する．シェアードメモリはブロック内のスレッドが共有できるメモリ領域であり，アクセスが高速である．ブロック内の他のスレッドが計算した値を利用したい場合にシェアードメモリを利用する．ローカルメモリはレジスタの容量が足らずにレジスタに保持できない値を保持するためのメモリである．コンスタントメモリは GPU 側から値を書き換えることができないメモリであり，計算に用いる定数などを格納するのに用いる．テクスチャメモリはグローバルメモリの一種であり，テクスチャメモリから読まれるデータはリードオンリーキャッシュにキャッシュされる．

3.1.4 グローバルメモリへのアクセス

GPU による計算では，メモリ容量の観点からグローバルメモリに流体の圧力や速度などの従属変数を格納する．GPU での数値解析では，各スレッドが必要なデータをグローバルメモリから読み込み，計算を行い，その結果をグローバルメモリに書き出す．これを繰り返し行うことで所望の状態まで計算を進めていくことになる．流体計算や粉体計算などでは，浮動小数点演算の時間よりもメモリアクセスの時間が長い，高い実行性能を出すためには効率的なグローバルメモリへのアクセスを意識してプログラミングする必要がある．

グローバルメモリは，特定の条件を満たすと高速にアクセスできるが，条件を満たさないとアクセス

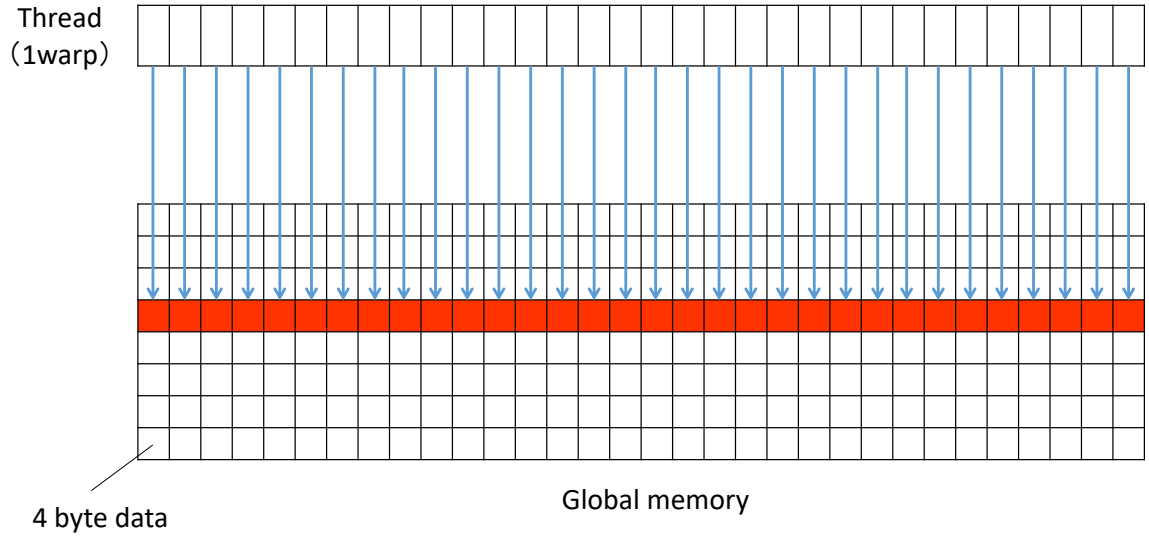


Fig.3.3: Aligned and coalesced memory access.

効率が急激に悪くなる．メモリアクセスを最適化する条件として，アラインメモリアクセスとコアレスメモリアクセスがある．GPU のメモリアクセスは 32 個のスレッドからなるワープ単位で実行され，32 byte，64 byte，128 byte 毎にアクセスされる．アラインメモリアクセスとは，ワープがアクセスするメモリの最初のアドレスが 128 byte の倍数であることである．コアレスメモリアクセスとは，ワープ内の 32 個のスレッドが連続したメモリにアクセスすることである．アラインメモリアクセスとコアレスメモリアクセスを両方とも満たしているときの状態のアクセスを Fig.3.3 に示す．なお，Fig.3.3 では一つの小領域が 4 byte のメモリを示し，一行で 128 byte となっていて，左端が 128 byte の倍数のメモリアドレスである．この状態では，128 byte のデータを 1 回のロードで実行でき，アクセスの効率は 100% になる．Fig.3.4 はコアレスメモリアクセスの条件は満たしているが，アラインメモリアクセスの条件を満たしていない状態である．128 byte のデータを 1 回と，はみ出している 8 byte のメモリを読むための 32 byte のロードが行われる．アラインメモリアクセスの条件を満たしていなくても，極端にアクセス効率は悪くならない．Fig.3.5 はランダムメモリアクセスの状態で，アラインメモリアクセスとコアレスメモリアクセスを両方とも満たしていない．GPU では非常に効率が下がるため，ランダムメモリアクセスが発生しないようプログラミングする必要がある．

3.1.5　ワープダイバージェンスによる実行効率の低下

GPU の処理ではワープ内の 32 個のスレッドが完全に同期して実行される．例えば，スレッド番号が奇数の場合は処理 1 を実行し，偶数の場合は処理 2 を実行する命令を考える．ワープ内のスレッドは全く同じサイクルで同じ処理を実行するので，処理 1 と処理 2 が同時に実行されることはない．処理 1 を

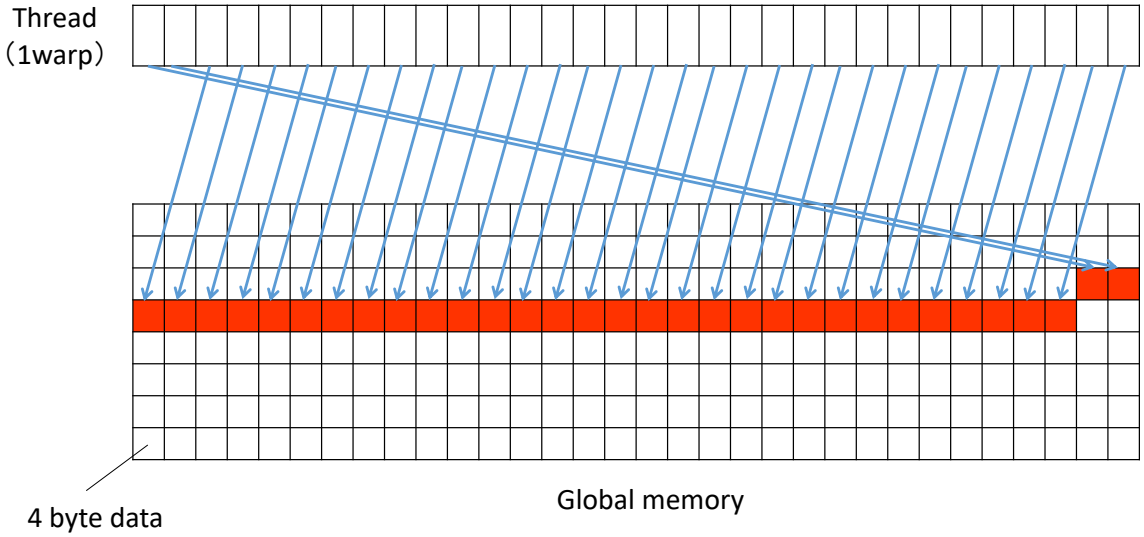


Fig.3.4: Misaligned and coalesced memory access.

行うときは，奇数のスレッドは処理を実行し，偶数のスレッドは休止状態となる．そして，処理 2 を行うときは，奇数のスレッドが休止状態になり，偶数のスレッドが処理を行う．この処理では半分のスレッドが休止状態になるため，効率は 50% になってしまう．このように，ワープ内のスレッドが別々の処理を行うことをワープダイバージェンスといい，GPU 計算の実行性能を大幅に下げる原因となる．CUDA プログラミングでは，ワープダイバージェンスの発生を抑えるため，`if` 文の条件分岐を少なくする必要がある．

3.2 格子計算の GPU 実装

格子ボルツマン法は格子計算に基づく計算手法であり，計算領域を格子で分割して各格子点での速度分布関数の時間発展を計算する．GPU を用いて格子計算を行う場合，各格子点にスレッドを割り当てて計算する．2 次元計算の場合のグリッド，ブロック，スレッドの割り当ての概念図を Fig.3.6 に示す．例えば，Fig.3.6 の場合は一つのブロックに 16×16 のスレッドを割り当て，16×16 の格子を計算する．4×4 のブロックを配置することで，全体で 64×64 の格子点を計算している．格子点数が $N_x \times N_y$ の計算領域を計算する場合，グリッド内のブロック数は

$$\begin{aligned} \texttt{gridDim.x} &= \texttt{floor}\left(\frac{N_x + \texttt{blockDim.x} - 1}{\texttt{blockDim.x}}\right) \\ \texttt{gridDim.y} &= \texttt{floor}\left(\frac{N_y + \texttt{blockDim.y} - 1}{\texttt{blockDim.y}}\right) \end{aligned} \tag{3.1}$$

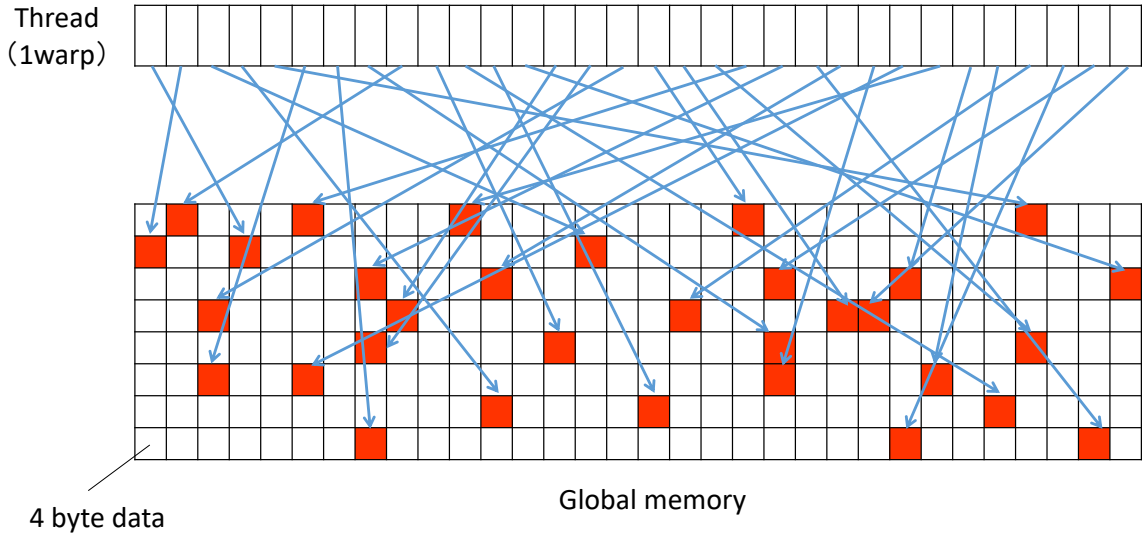


Fig.3.5: Random memory access.

と設定する．ここで，floor は小数点以下を切り捨てる処理である．各スレッドが担当する格子点 $(i,\ j)$ のインデックスは

$$\begin{aligned} i &= \texttt{blockIdx.x} \times \texttt{blockDim.x} + \texttt{thereadIdx.x} \\ j &= \texttt{blockIdx.y} \times \texttt{blockDim.y} + \texttt{thereadIdx.y} \end{aligned} \tag{3.2}$$

とする．

格子ボルツマン法では，各スレッドが担当する格子点における仮想粒子の衝突と並進の計算を実行する．並進過程では隣接の格子点の速度分布関数 f をグローバルメモリから読み込み，f の値をレジスタに格納し，衝突項の計算をして，自身の格子点のグローバルメモリにその計算結果を書き込む．格子点の並びとグローバルメモリに格納した速度分布関数の並びを合わせることで，グローバルメモリへのアクセスは Fig.3.3 のアライン・コアレスメモリアクセス，または Fig.3.4 のミスアライン・コアレスメモリアクセスになる．そのため，等間隔の格子を用いた格子ボルツマン法の計算では，メモリアクセスの高速化を行わなくても GPU で効率よく計算できる．

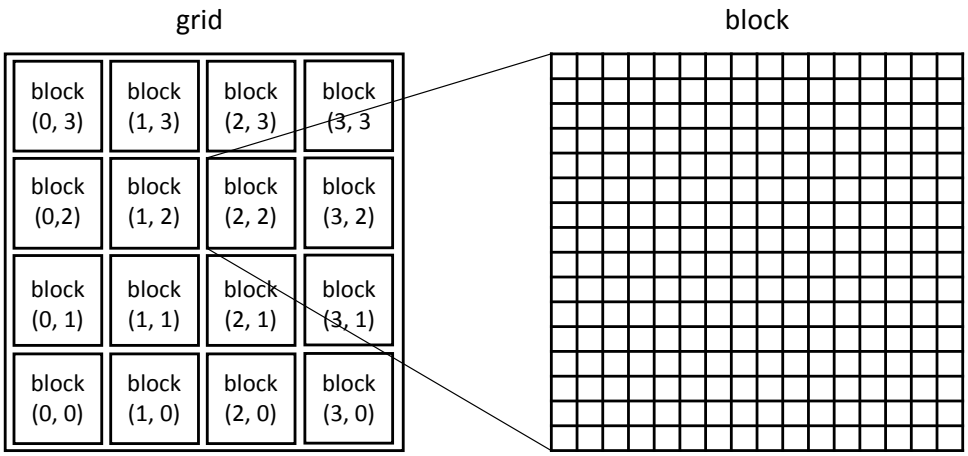


Fig.3.6: Allocation of CUDA blocks and CUDA threads for stencil computation.

3.3　個別要素法の GPU 計算

3.3.1　個別要素法の GPU 実装

GPU での個別要素法の計算では，メモリ容量とアクセス速度の観点から各粒子の位置や速度などの従属変数を GPU 上のグローバルメモリに保持する．粒子間相互作用計算のスレッド並列化の方法として，1 スレッドが 1 つの粒子を担当する方法 [93][94] や，1 スレッドが 1 つの接触点について計算する方法 [95] などがある．本研究では，多くの既往研究で行われていて，実装が容易である 1 スレッドが 1 つの粒子を担当するスレッド並列化を行う．また，CPU の計算では，粒子 j–i 間の接触力 $\boldsymbol{f}_{ji}$ を，作用反作用の法則を利用して粒子 i–j 間の接触力 $\boldsymbol{f}_{ij}$ の符号を反転したものから計算する方法がよく用いられている．GPU 計算の場合，接触力 $\boldsymbol{f}_{ij}$ を粒子 j の接触力の合力 $\boldsymbol{f}_j$ に足し合わせるときに，メモリへの書き込み競合が発生するため，計算の逐次処理が必要となる．本研究の実装では，作用反作用の法則を利用せず，粒子 i–j 間の接触力 $\boldsymbol{f}_{ij}$ と粒子 j–i 間の接触力 $\boldsymbol{f}_{ji}$ を重複して計算している．計算時間は最も計算量の多いスレッドに律速されているため，計算量は 2 倍に増加するが計算時間はそれほど増加しないと考えられる．粒子の時間積分などの相互作用計算以外の処理に関しては，1 スレッドが 1 つの粒子を計算するスレッド並列化を行う．

接触相互作用の計算では，接触相手の粒子の位置や速度などの従属変数をグローバルメモリに読みに行く．このとき，同一ワープ内のスレッドが連続的なメモリにアクセスすることはほとんどなく，Fig.3.5 のようなランダムメモリアクセスになる．メモリアクセスの効率が非常に悪くなるため，接触相手の粒子の従属変数へのメモリアクセスを改善することが計算高速化につながる．また，接触力の計算の前には接触しているかの判定があり，接触していれば接触力の計算，接触していなければ接触力の計算は行わずに次の粒子との接触判定を行う．同一ワープ内のスレッドでは，あるスレッドでは接触力を計算するが，別のスレッドでは接触判定をした粒子と接触していなくて，次の粒子との接触判定に移行したい場合がある．しかし，ワープ内のスレッドは完全に同期されて実行されるため，次の粒子との接触判定に移行したいスレッドは，接触力を計算しているスレッドの処理が終わるまで待っていなければならない．ワープダイバージェンスによる計算効率の低下を防ぐためには，周囲の粒子との接触判定の `if` 文ができるだけ真となるように，接触していない粒子との接触判定を削減する必要がある．

GPU での個別要素法の計算効率を低下させる原因となるワープダイバージェンスを防ぐために，無駄な接触判定を減らすための近傍探索手法の検討を行う．

3.3.2　近傍粒子探索の比較評価

この項では，既往研究で提案されてきた粒子法の近傍探索手法の GPU 実装と個別要素法での計算速度の比較を行い，個別要素法の計算に適した手法の検討を行う．本項の内容は研究業績論文 (3) の内容を含む．

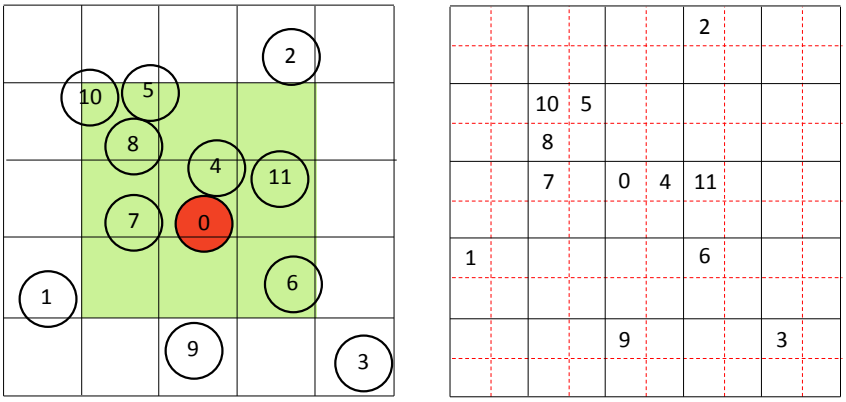


Fig.3.7: Neighbor-particle searching using a cell-list.

近傍粒子探索手法

個別要素法では，接触している粒子との相互作用を毎ステップ計算する．その時刻で接触する可能性がない粒子との距離計算および相互作用計算は行う必要がない．そのため，接触している粒子を効率よく探索すれば計算時間の短縮が可能となる．

個別要素法の近傍探索手法として，Fig.3.7 の左図のように計算領域を一様な格子で分割し，ある着目している粒子において，その粒子自身が所属するセルおよび周囲のセルに所属する粒子とのみ接触判定を行うことで効率的に近傍粒子を探索する方法が一般的に用いられる [48][93]．Fig.3.7 の番号が 0 の粒子の相互作用計算では，緑色の領域に含まれる粒子との接触判定と接触力の計算を行えばよい．各粒子がどのセルに含まれるかを判定し，Fig.3.7 の右図のように各セルが所属する粒子を記憶する．個別要素法では，粒子直径とセルの幅を等しく設定すると効率よく探索でき，その場合，一つのセルに含まれる最大の粒子数は 3 次元計算で 8 個である．各セルはセル内に粒子が存在していない場合でも，粒子のインデックスを格納するメモリ領域を持っておく必要があるため，格子数の 8 倍の数の粒子番号を格納するメモリが必要となり，計算領域を広く取るとメモリが枯渇する可能性がある．メモリ使用量を抑えてセル内の粒子番号を管理する方法として，リンク・リスト法 [47][94] とハッシュ法 [46] がある．

リンク・リストによるセル内の粒子の管理を用いた近傍探索手法の概念図を Fig.3.8 に示す．各粒子は同一セル内の他の粒子を一つ記憶することで一方向のリンク・リストを作成する．各セルはセル内の粒子を全て記憶するのではなく，リンク・リストの先頭の粒子インデックスのみを記憶する．これにより，各セルは粒子を一つ記憶するメモリをもてばよく，メモリ使用量を削減できる．GPU 計算では，1 スレッドが 1 粒子を担当して，粒子が所属しているセル番号の計算と他のセル内粒子とのリンク・リストを作成する．リンク・リストを作成するときに逐次処理が必要となるため，CUDA により提供されている atomic 関数の `atomicExch` を利用して行う．

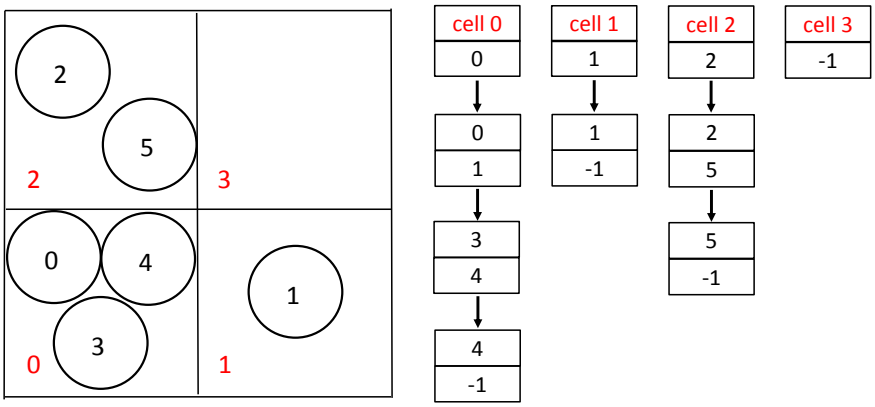


Fig.3.8: Handling of particles in cells using linked-list.

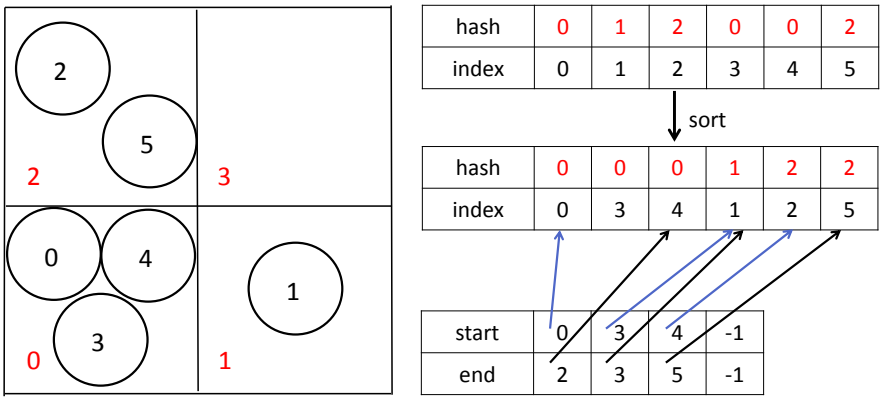


Fig.3.9: Handling of particles in cells using hash value.

セル番号をハッシュ値としたソートによるセル内の粒子の管理を用いた近傍探索手法の概念図を Fig.3.9 に示す．各粒子のインデックスを index 配列に，各粒子の所属するセル番号を粒子番号に対応した hash 配列に格納する．hash 配列の値をキーとして index 配列をソートし，粒子番号をセル番号が小さい順に並べる．要素数がセルの数である start 配列に，index 配列におけるセル内粒子の始点と終点の位置を記憶させる．セル内に粒子が含まれていない場合は，粒子が存在しないことを表す -1 を記憶させる．GPU 上での hash 配列と index 配置のソートには，Thrust ライブラリの `sort_by_key` を使用する．

セル分割を行わない近傍探索手法として，各粒子が数タイムステップのうちに相互作用する可能性のある近傍の粒子を近傍リストとして記憶し，数タイムステップ中は近傍リスト内の粒子とのみ接触判定

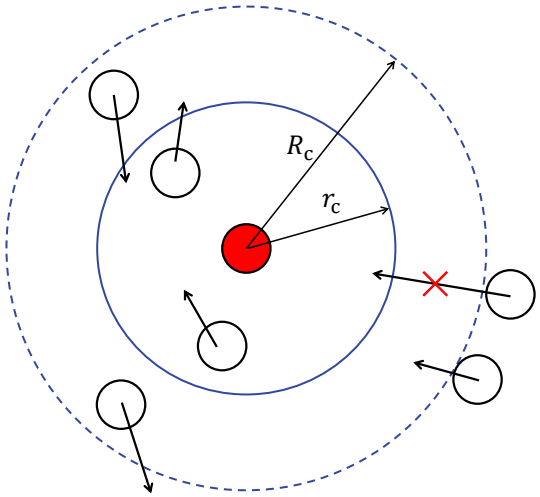


Fig.3.10: Neighbor-particle searching using the book-keeping method.

を行う粒子登録法（Book-keeping method）がある [48]．粒子登録法の概念図を Fig.3.10 に示す．注目している粒子に対して，影響半径 r_c よりも大きい半径 R_c を設定し，R_c 内に含まれる粒子を近傍リストに記憶する．個別要素法の計算では，影響半径 r_c は粒子直径 D と等しいので，R_c の大きさは

$$R_c = (1+\alpha)\, D \tag{3.3}$$

と設定する．ここで，α は近傍リストの更新頻度を調整するパラメータである．Fig.3.10 の赤いバツ印がついた粒子のように，近傍リストに含まれていない粒子が影響半径内に入った場合は正確な近傍探索が行えなくなるため，近傍リストを更新する必要がある．そのため，以下に記すようにして近傍リストの更新が必要であるかの判定を行う．現在の時間ステップにおける全粒子の中で最大の速さ $v_{\max}$ を求めて，1 タイムステップでの最大の移動距離を粒子移動距離の積載値 x_{book} に加算していく．

$$x_{\mathrm{book}}\,(t+\Delta t) = x_{\mathrm{book}}\,(t) + v_{\max}\,(t)\,\Delta t \tag{3.4}$$

x_{book} が $(R_c - r_c)/2$ を上回った場合に全粒子の近傍リストの更新を行うことで，近傍リスト外の粒子が影響半径内に入ることを防ぐ．近傍リストの更新が行われたら，粒子移動距離の積載値 x_{book} を 0 に初期化する．

粒子登録法の近傍リストの作成で全粒子との距離を計算すると膨大な時間がかかるため，Fig.3.11 のように近傍リストの作成にセル分割法による近傍探索を用いて，時間を大幅に短縮する．空間を分割するセルの大きさは，近傍リストを作成する半径 R_c とする．また，メモリ使用量を抑えるために，粒子登録法とハッシュ法のハイブリット手法 [95] や粒子登録法と連結リスト法のハイブリット手法 [96] が提案されている．

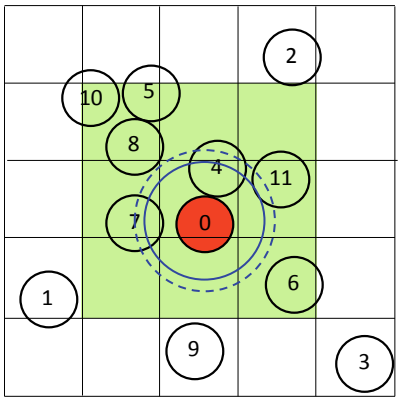


Fig.3.11: The combination method of the book-keeping method with a cell-list.

近傍探索手法の実行性能の比較

各近傍探索手法を GPU での個別要素法計算に実装したプログラムで，512 個から 8,388,608 個の大きさの等しい球形粒子を用いたダム崩壊シミュレーションを行い，計算時間を比較する．詳しい物理条件と計算条件を Table3.1，Table3.2 に示す．粒子の物理量は単精度であり，時間発展には Leap-frog 法を用いる．粒子登録法の近傍リストの更新頻度を調整するパラメータ α は 0.05 に設定する．格子サイズは，粒子登録法を用いない場合は粒子直径と等しく，粒子登録法を用いる場合は近傍リストを作成する半径（粒子直径の 1.05 倍）と等しく設定した．Fig.3.12 の上図のように計算領域の隅に粒子を配置し，Fig.3.12 の下図までの 10 万タイムステップの計算に要した時間を測定する．粒子数を変化させた場合の計算効率を比較するために，1 秒間に 1 ステップ更新できる粒子数を表す Cundall Number を求める．

$$\text{Cundall Number} = \frac{N_{\text{particle}}}{\text{TIME/STEP}} \tag{3.5}$$

ここで，N_{particle} は計算に使用した粒子数，TIME は全計算ステップの実行時間，STEP は実行ステップ数である．

各近傍探索手法でダム崩壊シミュレーションを行った場合の実行時間を Fig.3.13 に示す．横軸に粒子数を，縦軸に実行時間を示す．リンク・リスト法とハッシュ法を比較すると，どの粒子数の計算条件においてもリンク・リスト法を用いた場合のほうが計算時間が短く，粒子数が少なくなるほど相対的な差は大きくなる．リンク・リスト法，ハッシュ法を用いた場合の計算時間の内訳を Table3.3 に示す．ハッシュ法のソートは，リンク・リストの作成時間の 3.8 倍から 11.0 倍の時間がかかり，この差により，計算全体でもリンク・リスト法のほうが速いことが確認できる．全体の計算時間に対するリンク・リストの作成および index 配列と hash 配列のソートにかかる時間の割合は，粒子数が多い 8,388,608 粒子の場合はそれぞれ 3.8%, 1.1% であるが，粒子数が最も少ない 512 粒子の条件ではそれぞれ 41.6%, 9.8% である．

Table3.1: Physical conditions of DEM for performance study.

Particle diameter	[m]	2.0×10^{-3}
Particle mass	[kg]	1.0×10^{-5}
Spring constant (Normal)	[N/m]	4.0×10^{3}
Spring constant (Tangential)	[N/m]	1.6×10^{3}
Damping coefficient (Normal)	[Ns/m]	0.4
Damping coefficient (Tangential)	[Ns/m]	0.25
Coefficient of friction	[-]	0.3
Discrete time	[s]	5.0×10^{-6}

Table3.2: Computational conditions of DEM for performance study.

Number of particles	Size of dam [m^3]	Number of cells (without book-keeping)	Number of cells (with book-keeping)
512	$0.004 \times 0.004 \times 0.256$	$9 \times 9 \times 151$	$8 \times 8 \times 143$
2,048	$0.008 \times 0.008 \times 0.256$	$17 \times 17 \times 151$	$16 \times 16 \times 143$
8,192	$0.016 \times 0.016 \times 0.256$	$33 \times 33 \times 151$	$31 \times 31 \times 143$
32,768	$0.032 \times 0.032 \times 0.256$	$65 \times 65 \times 151$	$61 \times 61 \times 143$
131,072	$0.064 \times 0.064 \times 0.256$	$129 \times 129 \times 151$	$122 \times 122 \times 143$
524,288	$0.128 \times 0.128 \times 0.256$	$257 \times 257 \times 151$	$244 \times 244 \times 143$
2,097,152	$0.256 \times 0.256 \times 0.256$	$513 \times 513 \times 151$	$488 \times 488 \times 143$
8,388,608	$0.512 \times 0.512 \times 0.256$	$1025 \times 1025 \times 151$	$976 \times 976 \times 143$

粒子数が少なくなるに連れてリンク・リストの作成やソートに費やす時間の割合が増えて全体の計算時間への影響が大きくなるため，粒子数が少ないほど全体の計算時間の相対的な差が大きくなったと考えられる．Fig.3.13 の三角形のマーカは粒子登録法を併用した場合は，併用しない場合と比べて大幅に計算時間を削減できている．リンク・リスト法やハッシュ法は，相互作用計算のときに 1 辺が粒子直径の 3 倍の大きさの立方体の領域内の粒子を参照するため，接触していない粒子が多く含まれる．粒子登録法を併用することにより，相互作用計算で参照する領域が半径が粒子直径よりもわずかに大きい球形となり，接触判定で参照する近傍粒子は，ほとんど接触している状態となる．ワープダイバージェンスによる効率の低下を防ぎ，かつ接触していない粒子との無駄な計算を削減したため，2.7 倍から 4.2 倍の大幅な高速化となった．

各近傍探索手法での Cundall Number を Fig.3.14 に示す．横軸が粒子数，縦軸が Cundall Number である．最も計算性能の高い近傍探索手法は粒子登録法とリンク・リスト法のハイブリット手法であり，性能が低いのはハッシュ法である．粒子数が 10 万個以上の計算条件では，どの近傍探索手法でも CundallNumber はほぼ一定の値を取っていて，粒子数が計算性能に与える影響は少ないことがわかる．

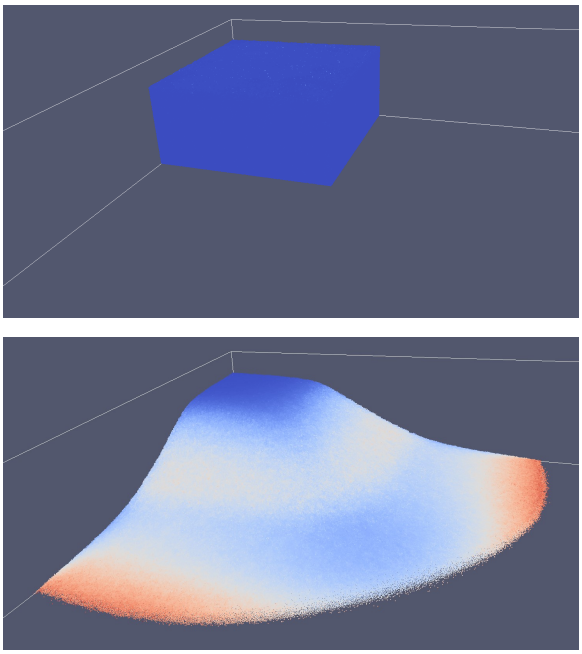


Fig.3.12: The dam-breaking problem used for performance study (top: initial condition, bottom: after 100,000 steps).

Table3.3: The computation time for particle handling of the linked-list method and the hash method.

	Linked-list method			Hash method		
Number of particles	Total [s]	Linked-list [s]	Ratio [%]	Total [s]	Hash [s]	Ratio [%]
512	69.72	6.80	9.8	105.42	43.90	41.6
2,048	78.44	7.38	9.4	143.15	62.99	44.0
8,192	81.05	7.13	8.8	156.36	78.18	50.0
32,768	236.38	8.70	3.7	321.11	81.69	25.4
131,072	812.81	15.53	1.9	986.84	150.28	15.2
524,288	3,230.36	42.68	1.3	3,637.28	292.37	8.0
2,097,152	13,220.62	151.42	1.1	14,368.74	665.74	4.6
8,388,608	53,383.79	583.70	1.1	57,463.59	2,205.29	3.8

粒子数が 10 万個以下の少ない条件では，粒子数が少なくなるに連れて計算効率も低下し，特に粒子数が 1 万個以下になると性能が急激に低下する．このことから，粒子数が 10 万個以下の小規模な問題では，GPU の性能を十分に利用できず，GPU による高速化の効果は低いことが明らかになった．

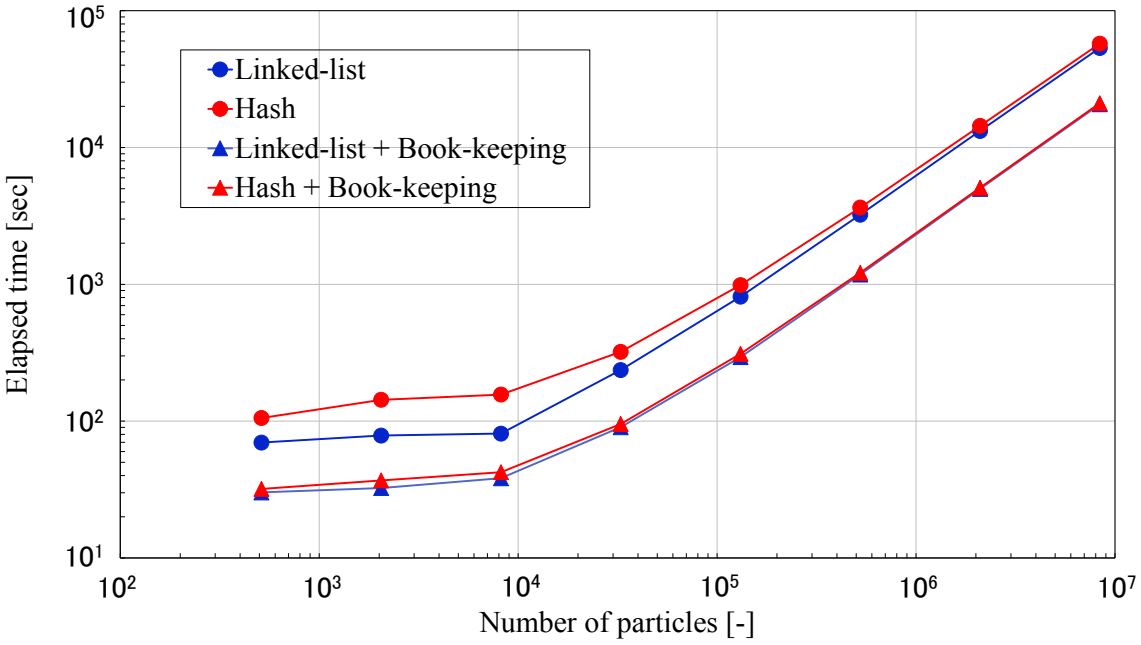


Fig.3.13: The computation time of the dam-breaking problem.

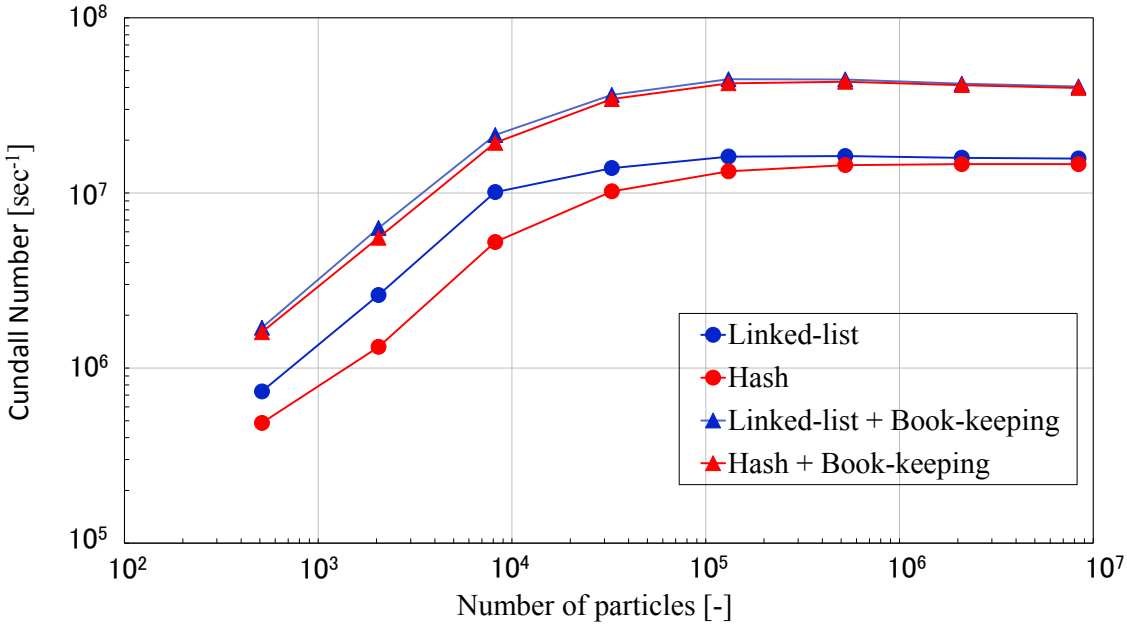


Fig.3.14: The result of performance measurement of the dam-breaking problem.

3.4　MPI ライブラリを用いた複数 GPU 計算

GPU のメモリ容量は最新の Tesla V100 で 16 GB であり，単一 GPU の計算ではメモリ容量の制約から十分な解像度を用いた混相流解析は困難なため，複数台の GPU を用いた並列計算を行う必要がある．この節では，複数 GPU を用いた格子ボルツマン法とフェーズフィールド法，個別要素法の連成手法の並列計算について述べる．

3.4.1　領域分割法

等間隔直交格子による計算では，Fig.3.15 のように全体の計算領域を小領域に分割し，各 GPU に小領域を割り当てることによる並列計算が一般的である [7][15]．1 つの MPI プロセスが 1 台の GPU を担当する Flat-MPI で並列化を行う．格子ボルツマン法と界面捕獲の計算では，小領域の表面にある格子点が隣接する小領域の格子点を参照する．各 GPU のメモリ領域は完全に独立していて，隣接の小領域を担当している GPU のグローバルメモリを直接参照することはできないので，分散メモリ型の並列計算のインターフェースである MPI（Message Passing Interface）を用いたデータ転送を行う．各 GPU が担当する小領域のまわりに，ハロー領域と呼ばれる隣接する小領域とのデータ交換用の格子点を配置する．別のノードにある GPU と GPU の直接通信は行えないため，Fig.3.16 のように，送信側の GPU は転送する格子点のデータをグローバルメモリからホスト側の MPI 通信用のバッファにコピーする．送信側の CPU は受信側の GPU が接続されている CPU に MPI 通信を用いてデータを送る．受信側の CPU が受信側の GPU のハロー領域にデータを転送することで，GPU 間の通信が完了する．

本研究の混相流解析では，固体粒子や物体よりも細かい流体計算格子を用いるため，粒子や物体の数は格子点数よりも非常に少ない．そのため，個別要素法による固体粒子や物体の計算コストは格子ボルツマン法よりも小さく，個別要素法を MPI 並列化しなくても全体の計算時間への影響は小さいと考えられる．そこで，格子ボルツマン法と個別要素法の連成計算の複数 GPU 実装として，Fig.3.17 のように，格子ボルツマン法と界面捕獲法の計算については 2 次元領域分割に基づく単純な並列化を行い，個別要素法については MPI による並列化は行わずに各 GPU が全ての固体粒子の計算を冗長に実行する方法を提案する．個別要素法の並列計算に領域分割法を用いた場合，相互作用のためのハロー領域の粒子の通信，粒子が境界を横切り別の領域へ移動したことに伴う通信，境界にまたがる粒子に作用する流体力の総和計算のための通信といった 3 種類の通信が必要となる．提案手法では多くの煩雑な処理が必要である個別要素法の並列計算を行わないため，複数 GPU 実装を容易に行える．全ての粒子の運動を計算するために，各 GPU は全ての粒子に作用する流体力を知る必要があるが，自身の小領域に存在している粒子に働く流体力しか計算できない．そのため，MPI で集団通信を行う `MPI_Allreduce` 関数を用いて，ノード間の流体力の総和計算を実行する．

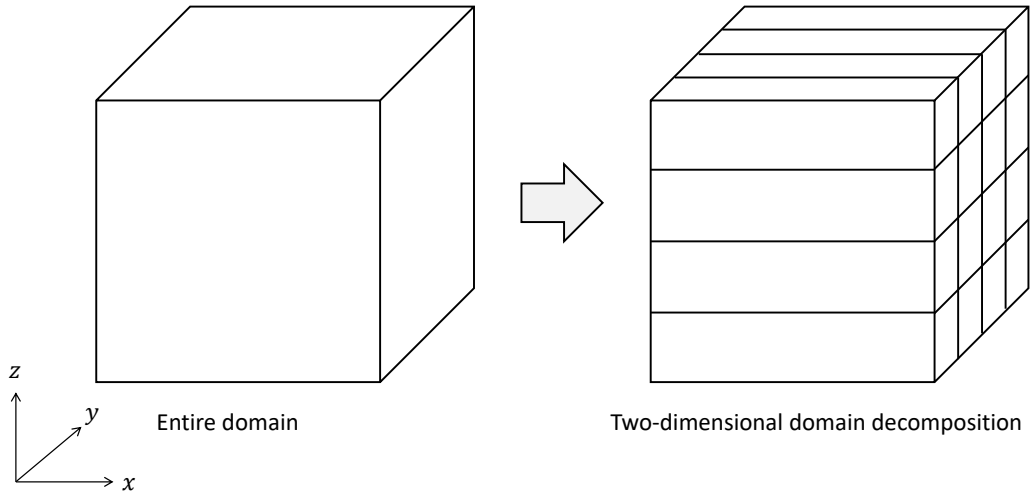


Fig.3.15: Two-dimensional domain decomposition for stencil computation.

3.4.2 TSUBAME3.0 を用いた実行性能測定

東京工業大学・学術国際情報センターのスパコン TSUBAME3.0 は 540 ノードから構成され，1 ノードは 2 ソケットの CPU（14-cores Intel Xeon E5-2680 V4 2.4 GHz）と 4 台の GPU（NVIDIA Tesla P100 with NVLink）から構成される．各ノードは Intel Omni-Path 100Gbps × 4 で接続されている．

TSUBAME3.0 を用いて水面に複数の角柱を自由落下させる計算を行い，本シミュレーションコードの実行性能を測定した．実行性能 P は 1 秒間に更新できる格子点数（MLUPS : Mega Lattice Update Per Second）

$$P = \frac{N^{\mathrm{lattice}}}{T/N^{\mathrm{step}}} \times 10^{-6}\ [\mathrm{MLUPS}] \tag{3.6}$$

で評価した．ここで，N^{lattice} は総格子点数，N^{step} はステップ数，T は実行時間である．格子ボルツマン法の計算には単精度浮動小数点数，個別要素法の計算には倍精度浮動小数点数を用いた．

弱スケーリングの測定

弱スケーリングは「同じ実行時間で計算規模をどこまで大きくできるか」を評価する方法であり，1 台の GPU 当たりの計算サイズを固定し，GPU 数を増やしつつ計算規模を大きくしていき実行性能と並列化効率を評価する．2 倍の計算規模に対して 2 倍の数の GPU を利用すれば，理想的には 1 タイムステップを元の計算と同じ実行時間で計算できるはずである．複数 GPU で並列計算をしたときに，理想的な性

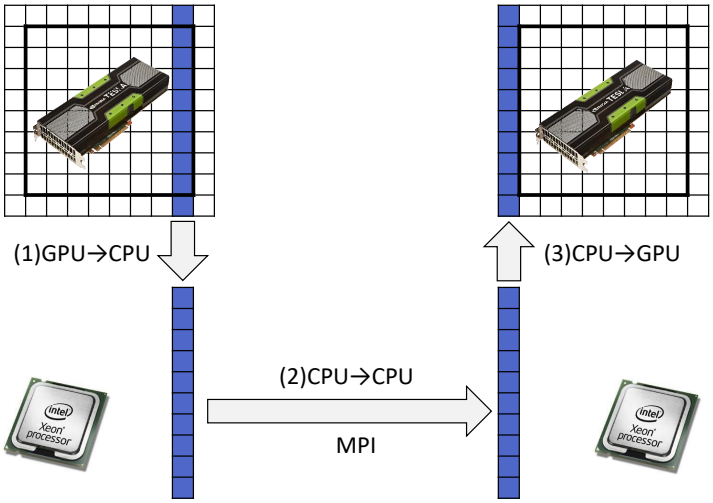


Fig.3.16: Data transfer between GPUs by CPUs.

能向上に対してどの程度の性能であるかを確認する．

弱スケーリングの評価を Table.3.4 に示す条件で行った．1 台の GPU あたりの格子点数を 128 × 2048 × 1024，物体数を 16 と設定し，16 台の GPU から 256 台の GPU までの実行性能を測定した．100 ステップの計算を行い，その実行時間から実行性能を測定した．256 台の GPU の条件で，総格子点数は約 43 億（128 × 8192 × 4096），物体数は 4096 である．

弱スケーリングの測定結果を Fig.3.18 に示す．プロット点が測定結果であり，点線は理想的な実行性能である．16 台の GPU で 1,506 MLUPS，256 台の GPU で 20,110 MLUPS の実行性能が得られ，16 台の GPU に対して 256 台の GPU で 13.3 倍の性能向上が得られている．並列化効率 E を理想的な実行性能 P^{ideal} から

$$E = \frac{P}{P^{\mathrm{ideal}}} \times 100\,[\%] \tag{3.7}$$

と計算し，その結果を Table3.5 に示す．256GPU での並列化効率は 83.4% であり，良好な弱スケーリングが得られた．100 ステップの計算時間の内訳を Fig.3.19 に示す．MPI 並列化をしていない個別要素法の計算時間は，16 台の GPU では計算全体の 2.2%，256 台の GPU では 8.4% であり，並列数の増加に伴い個別要素法の計算時間が長くなっているが，物体の計算を MPI 並列化していないことによるオーバーヘッドは小さいことが確認できた．

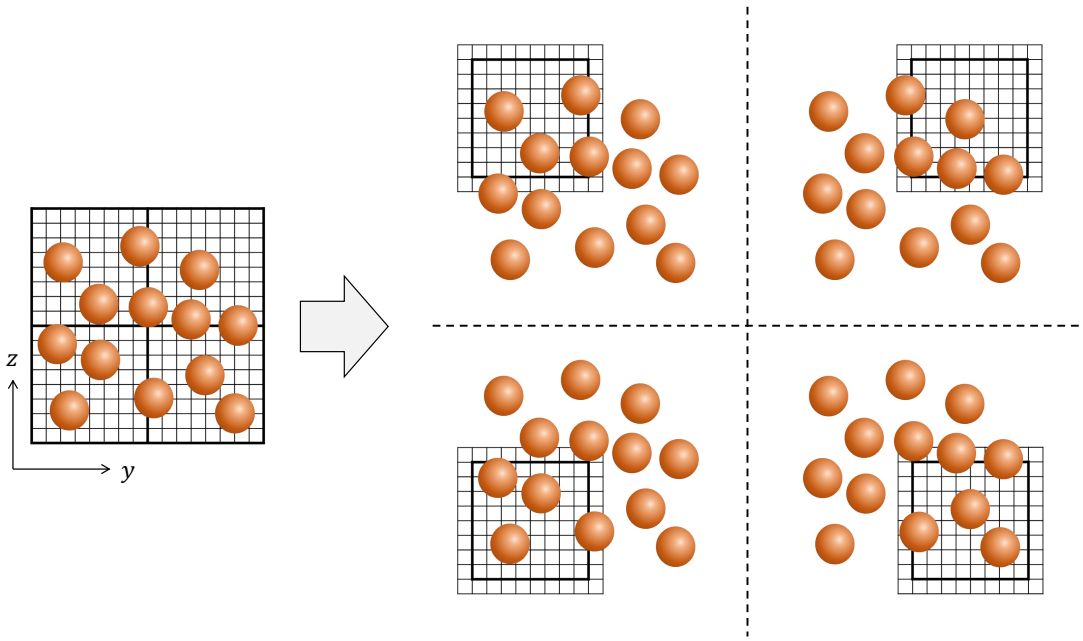


Fig.3.17: The multiple GPU implementation using 2-d domain decomposition for coupled LBM and DEM simulation.

Table3.4: Conditions for weak scaling measurement of the uniform LBM-DEM code on the TSUBAME3.0. p_y and p_z are the number of sub-domains in the y-axis and the z-axis direction, respectively.

Number of GPUs $(p_y \times p_z)$	Number of lattice points $(N_x \times N_y \times N_z)$	Number of objects	Number of particles
16 (4×4)	$128 \times 2048 \times 1024$	256	147,968
32 (4×8)	$128 \times 2048 \times 2048$	512	295,936
64 (8×8)	$128 \times 4096 \times 2048$	1024	591,872
128 (8×16)	$128 \times 4096 \times 4096$	2048	1,183,744
256 (16×16)	$128 \times 8192 \times 4096$	4096	2,367,488

強スケーリングの測定

強スケーリングは「同じ計算をどれだけ短時間で実行できるか」を評価する方法であり，計算規模を固定し，GPU の台数を増やしていき実行性能と並列化効率を確認する．同じ計算に対して，GPU の台数を 2 倍にすれば理想的には計算時間を 1/2 にできるはずであり，実際の性能が理想的な性能に対してどの程度の性能か評価する．

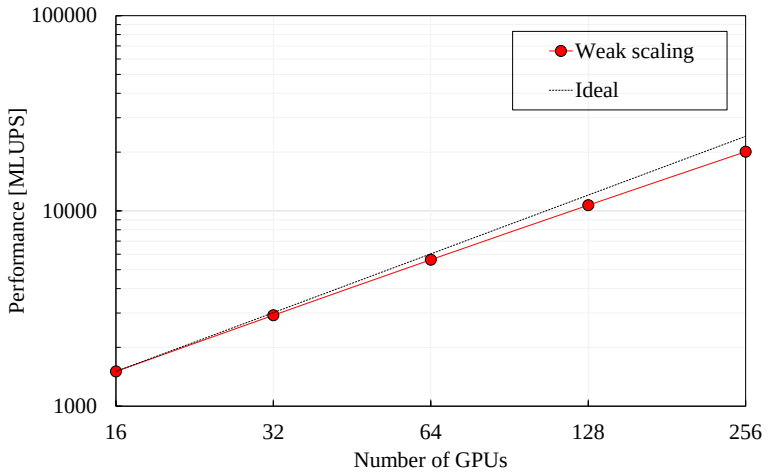


Fig.3.18: Weak scaling results of free-surface flow simulations with objects.

Table3.5: Parallel efficiency of weak scaling.

Number of GPUs	Parallel efficiency [%]
16	—-
32	96.97
64	93.35
128	88.78
256	83.44

強スケーリングの評価を Table.3.6 に示す条件で行った．全格子点数を 128 × 2048 × 1024 とし，物体数を 256 と設定し，16 台の GPU から 256 台の GPU までの実行性能を測定した．強スケーリングの測定結果を Fig.3.20 に示す．GPU 数を増加させると実行性能は増加するが，増加量は理想的な増加量よりも低く，GPU 数が多くなるほど実行性能の伸びが悪くなる傾向が確認できる．16GPU での実行性能を基準としたときの並列化効率を Table3.7 に示す．GPU 数を 2 倍にした 32GPU での並列化効率は 79.49% であり，16GPU 数の計算に対して 1.59 倍の高速化ができている．一方，256GPU での並列化効率は 28.73% であり，理想的には 16GPU に対して 16 倍の高速化となるはずが，実際は 4.60 倍の高速化である．このように，計算規模に対して必要以上に GPU を割り当てても，効率的に計算時間を短縮できないといえる．

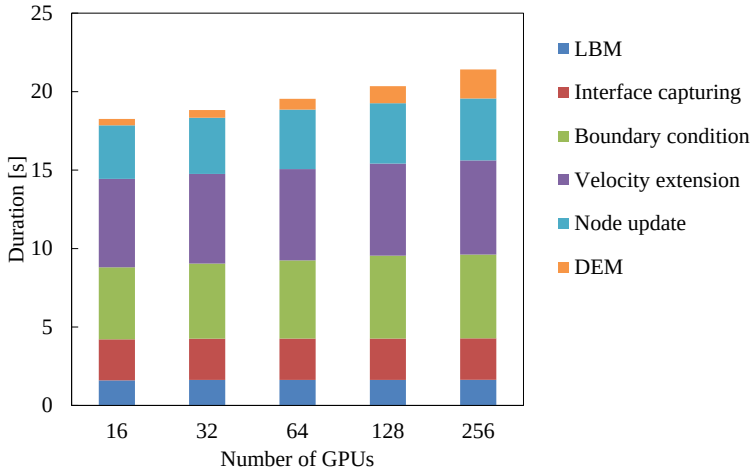


Fig.3.19: Breakdown of calculation time of weak scaling.

Table3.6: Conditions for weak scaling measurement of the uniform LBM-DEM code on the TSUBAME3.0. p_y and p_z are the number of sub-domains in the y-axis and the z-axis direction, respectively.

Number of GPUs $(p_y \times p_z)$	Number of lattice points $(N_x \times N_y \times N_z)$	Number of objects	Number of particles
16 (4×4)			
32 (4×8)			
64 (8×8)	$128 \times 2048 \times 1024$	256	147,968
128 (8×16)			
256 (16×16)			

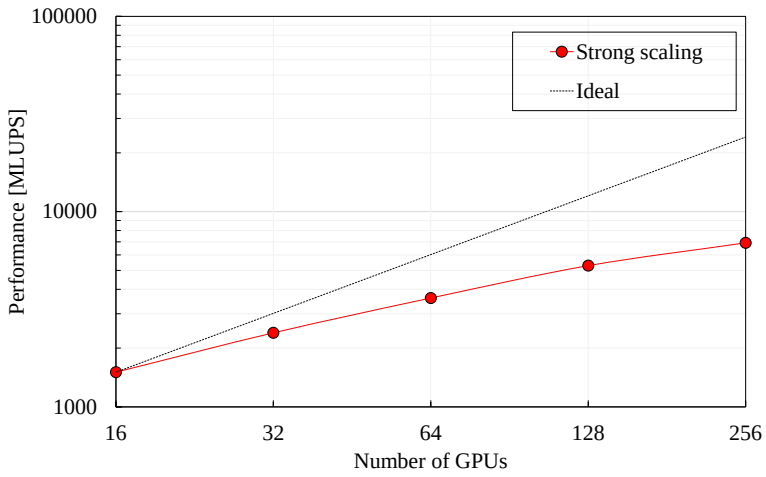


Fig.3.20: Strong scaling results of free-surface flow simulations with objects.

Table3.7: Parallel efficiency of strong scaling.

Number of GPUs	Parallel efficiency [%]
16	—-
32	79.49
64	59.93
128	43.89
256	28.73

3.5　等間隔直交格子を用いた大規模混相流解析

この説では，第 2 章で提案した格子ボルツマン法に基づく混相流解析手法を用いた大規模な混相流解析を実行する．計算格子には等間隔の直交格子を用い，実行には東京工業大学のスーパコンピュータ TSUBAME2.5 および TSUBAME3.0 を用いた．多数の固体粒子を含む固気二相流として，粉体を流動化させて混合や乾燥を行う噴流層のシミュレーションを行う．複雑な形状をした物体を含む流れとして，イチョウの葉が舞い落ちる計算を行う．自由界面を含む流れとして，流木を含む津波流の実験を模擬した計算を行う．

3.5.1　噴流層

流動層や噴流層は固体粒子を充填した容器の底から水や空気などの流体を流すことで，粉体の混合，乾燥，流体との反応などを行う技術である [97]．下からの流量が少ないと固体粒子は運動せず，粒子間の隙間を流体が流れる．一方，流量が増えると流体により固体粒子は容器内で浮遊し，流動化した状態となる．流動化した状態では固体粒子の混合が促進され，また，固体粒子と流体の接触面が増えることで乾燥や反応がより行われる．そのため，固体粒子の流動化は化学プロセスで非常に重要である．

流動層は容器の底から一様な流れを入れることで固体粒子の流動化を行い，固体粒子は容器内でランダムな運動をする．流入速度により流動化の状態は変わり，流入速度が低いと固体粒子は流体のような振る舞いをし，流速を上げていくと沸騰のように粒子群に気泡が発生する．一方，底から一様な流れを入れるのではなく，容器の底に開けたオリフィスから流体を流入させて流動化を行う装置を噴流層という．噴流層での流動化では，Fig.3.21 のように 3 つの領域が形成されることが知られており，固体粒子は規則的な運動をする [97][98]．噴流は固体粒子を押しのけることで容器の中央に流路を形成し（Fig.3.21 の Spout），容器の下にある粒子はこの領域を通り上昇し，粒子群の外に巻き上げられる．噴流により巻き上げられた粒子は粒子群の上部で噴水のようになり（Fig.3.21 の Fountain）粒子群表面に落下する．流路の周囲にある粒子（Fig.3.21 の Annulus）は徐々に下に沈んでいきながら，少しずつ噴流に巻き込まれて上昇させられる．このような循環運動により固体粒子は混合されていく．噴流層は流動層よりも流動化に必要な流量が少なく，比較的直径の大きい粒子の流動化が可能である．

格子ボルツマン法による流体計算と個別要素法による粉体計算を組み合わせ，噴流層のシミュレーションを行う．固体粒子は球形を仮定し，下部が絞られた矩形容器に，直径 0.5 mm で密度 100 kg/m^3 の球形粒子を 81,920 個充填する．粒子を容器内に自由落下させることで初期の粒子層を作成した．オリフィスから空気を 1.5 m/s で流入させる．空気の密度は 1.205 kg/m^3，動粘度は 1.512×10^{-5} m^2/s である．流入速度に対する粒子レイノルズ数は 50 である．格子ボルツマン法の格子解像度を固体粒子よりも細かい 0.05 mm とし，直径に対して 10 格子を割り当てて計算する．矩形容器の形状は符号付き距離関数で表現した．境界条件として，球形粒子に non-slip 境界を，容器の頂面に流出境界を，絞られた部分の底面に流入境界条件を，壁面に non-slip 境界を設定した．総格子点数は $512 \times 512 \times 1680$ であり，TSUBAME2.5 に搭載された GPU（Tesla K20X）を 48 台用いる．計算安定化のために，格子ボルツマ

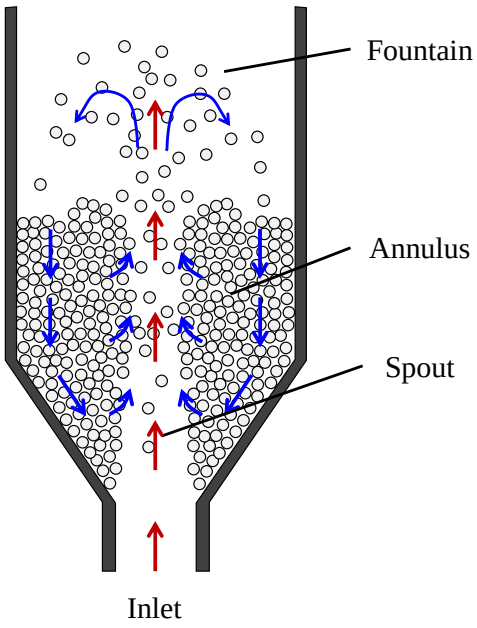


Fig.3.21: A flow of the inlet gas (red) and the solid particle movement (blue) of a typical spouted bed.

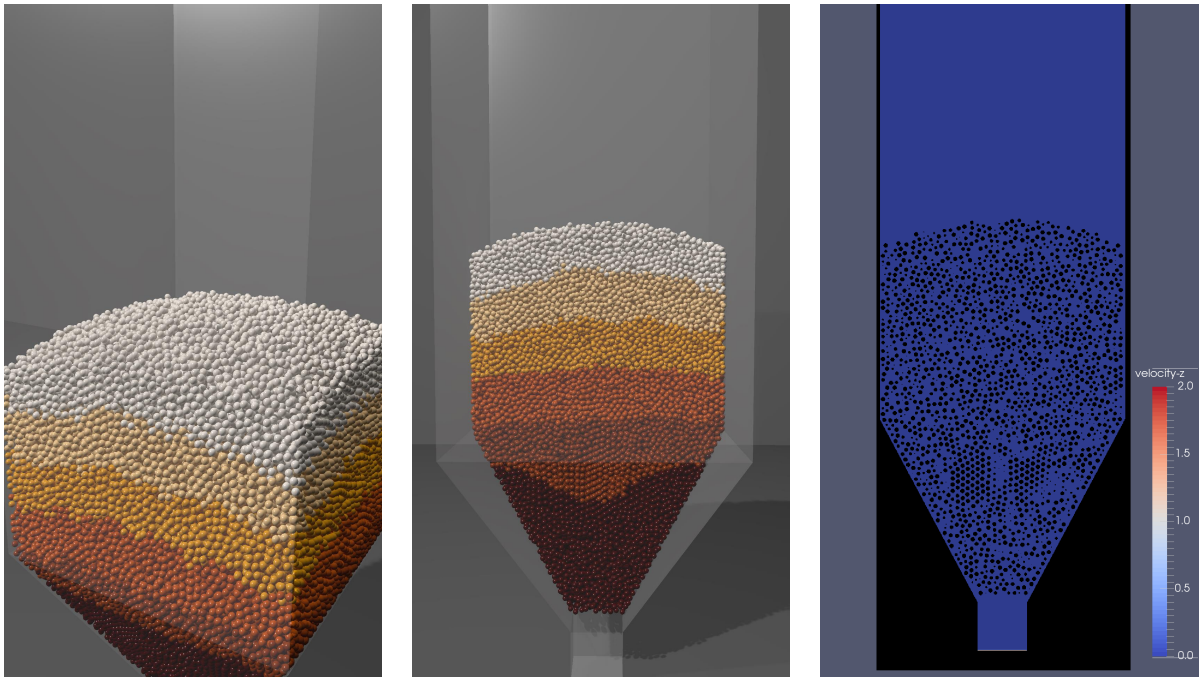
ン法に MRT モデルの衝突項と LES 乱流モデルを導入する．時間刻み幅は 1.67 μs であり，格子ボルツマン法と個別要素法で同じ時間刻みを設定した．計算ステップ数は 750,000 である．

計算結果を Fig.3.22 に示す．左図が計算領域内のすべての粒子を可視化したものであり，粒子の色は初期配置で色分け（下の粒子ほど濃色）している．中央の図は，計算領域の中央から手前側の粒子を表示せず，空気が流れる領域を表示したものである．右図は計算領域の中央で断面をとり，鉛直方向の速度場を可視化した図である．空気の流入を開始すると，噴流は固体粒子を押しのけ，単一の気泡を生成しているのが確認できる（Fig.3.22(b)(c)）．その後，噴流は粒子群を抜けて，固体粒子を上方に巻き上げて Fountain を形成している（Fig.3.22(d)(e)）．速度場から容器の中心を空気が流れていることが確認でき，固体粒子よりも細かい格子を用いた流体計算のため，粒子周りの流れを捉えられているのが確認できる．750,000 ステップ後の Fig.3.22(f) では，粒子郡の上部で各色の粒子が混ざり合っているのがわかる．Fig.3.21 に示す典型的な噴流層による流動化と粒子の混合を定性的に再現できた．また，750,000 ステップの計算に対し，計算時間は 168 時間ほどであった．

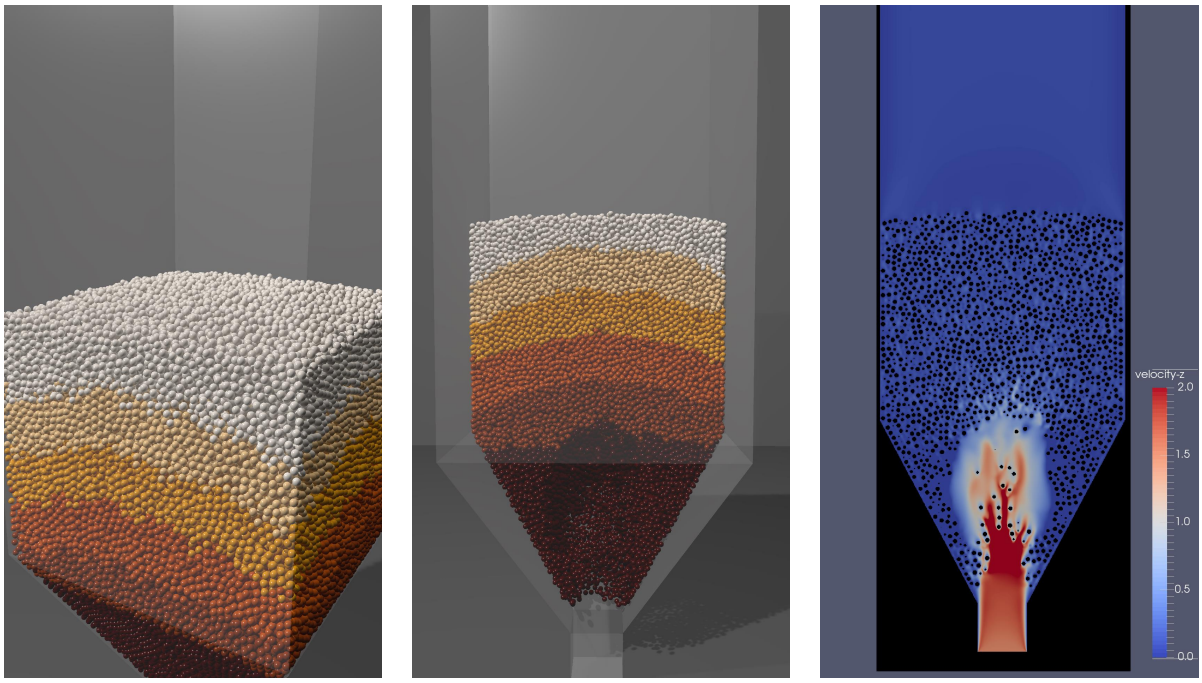
計算結果を定量的に評価するため，噴流層における最小噴流化速度を実験式から求め，本シミュレーションの流入条件と比較する．萩野ら [99] は次元解析と実験的検討により最小噴流化速度 u_{ms} の実験式を

$$u_{\mathrm{ms}} = 1.51 \times 10^{-2} \left[\frac{\epsilon^4}{2\left(1-\epsilon\right)}\right]^{1/2} \left[\left(1-\epsilon\right)\frac{\rho\left(\rho_p-\rho\right)gD^3}{\mu^2}\right]^{1/4} \left(\frac{d_p}{D}\right) \left(\frac{d_0}{D}\right)^{1/3} \left[\frac{2gH\left(\rho_p-\rho\right)}{\rho}\right]^{1/2} \tag{3.8}$$

と提案している．ここで，d_p は粒子直径，ρ_p は粒子の密度，ρ は流体の密度，μ は流体の粘度，D は容器直径，d_0 は流入口の直径，H は粒子充填高さ，ϵ は空隙率である．なお，この実験式は円筒容器での噴流層における最小噴流化速度である．シミュレーションは矩形容器での噴流層であるが，円筒容器と仮定すると，シミュレーションで用いたパラメータは，$d_p = 0.5$ mm，$\rho_p = 100$ kg/m^3，$\rho = 1.2$ kg/m^3，$\mu = 1.82 \times 10^{-5}$ Pa$\cdot$s，$D = 25$ mm，$d_0 = 5$ mm，$H = 40$ mm，$\epsilon = 0.57$ となる．これを実験式に代入すると最小噴流化速度 $u_{\mathrm{ms}} = 0.0344$ m/s を得る．一方，シミュレーションの流入速度は $u_{\mathrm{in}} = 1.5$ m/s であり，容器内の平均流速は 0.06 m/s である．このことから，シミュレーションの流入条件は実験式から計算される最小噴流化速度を上回っており，流動化する条件であることが確認できた．

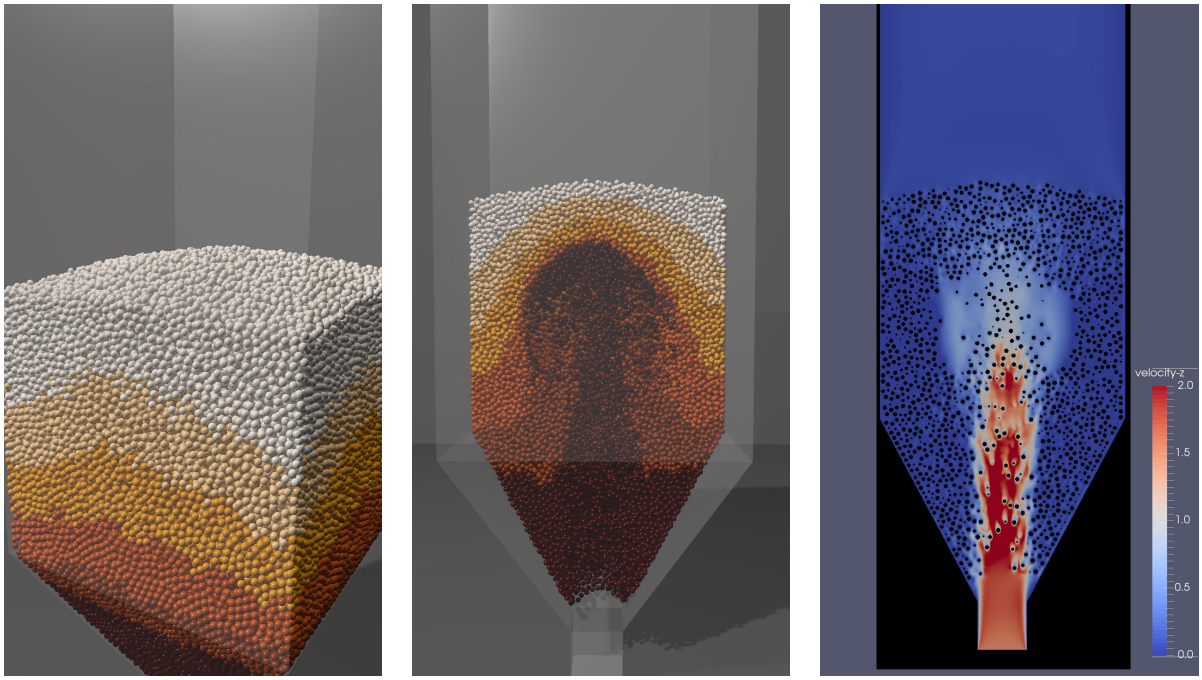


(a) $t = 0.0$ ms.

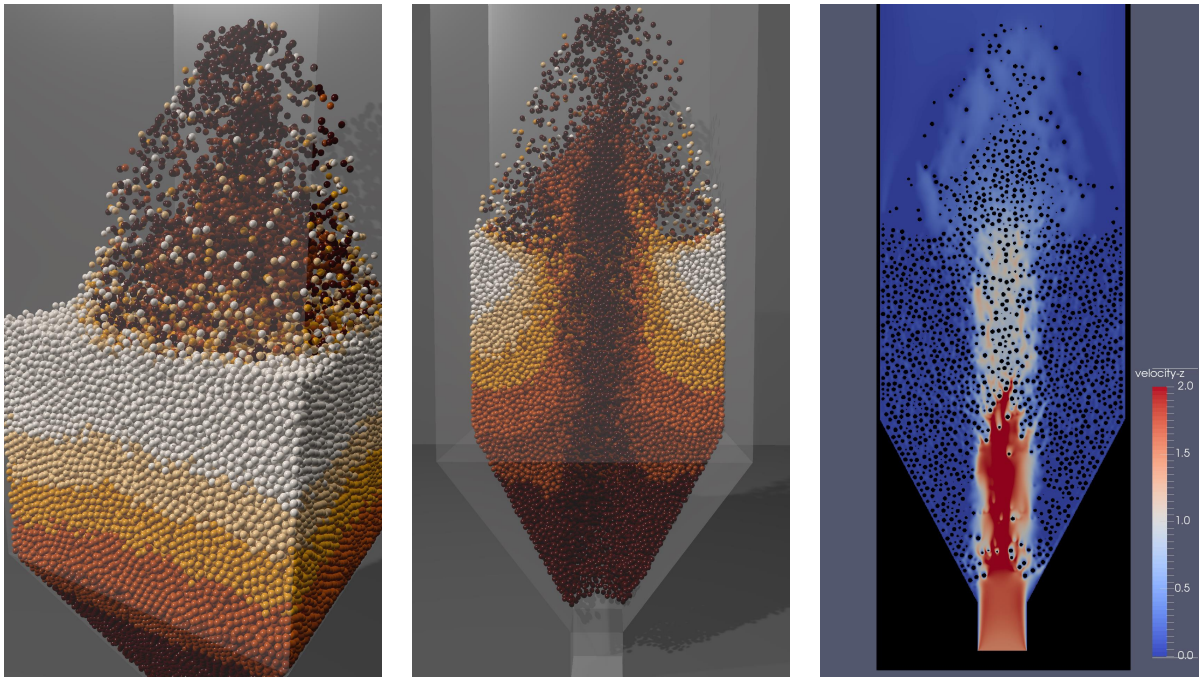


(b) $t = 0.83$ ms.

Fig.3.22: Results of a resolved-particle simulation for a spouted bed. The left panels show all particles colored by their initial position. The center panels show the particles on the far side cut at the center of the domain. The right panels indicate the velocity field in the vertical direction.

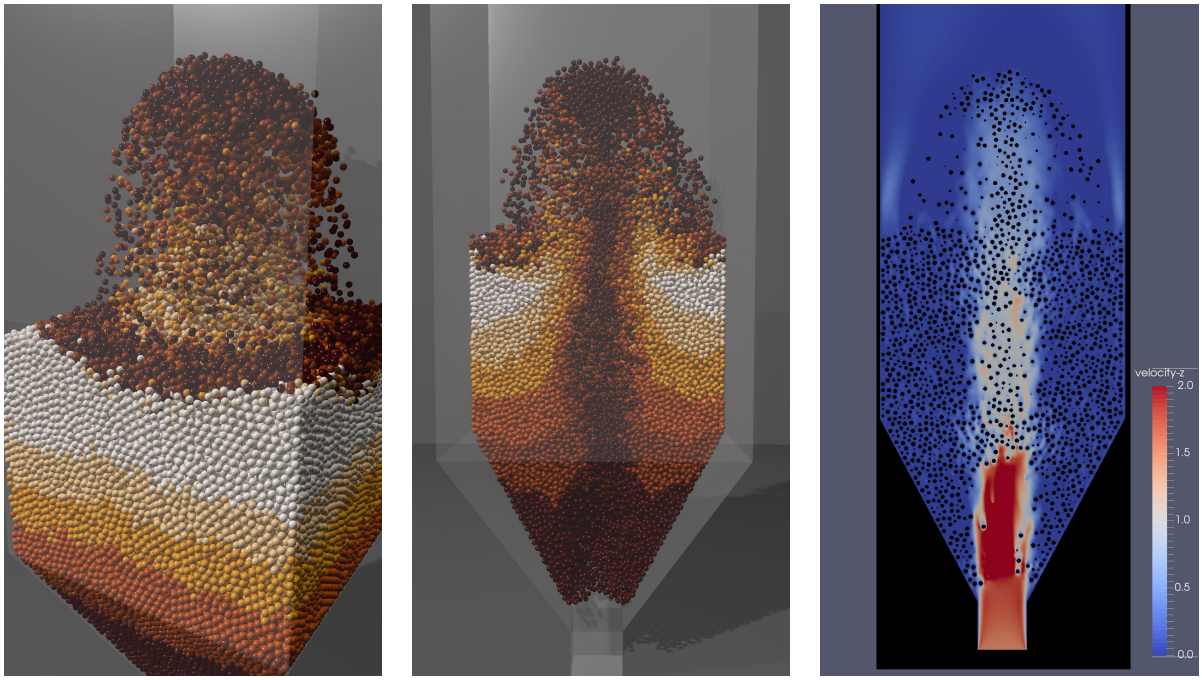


(c) $t = 1.67$ ms.

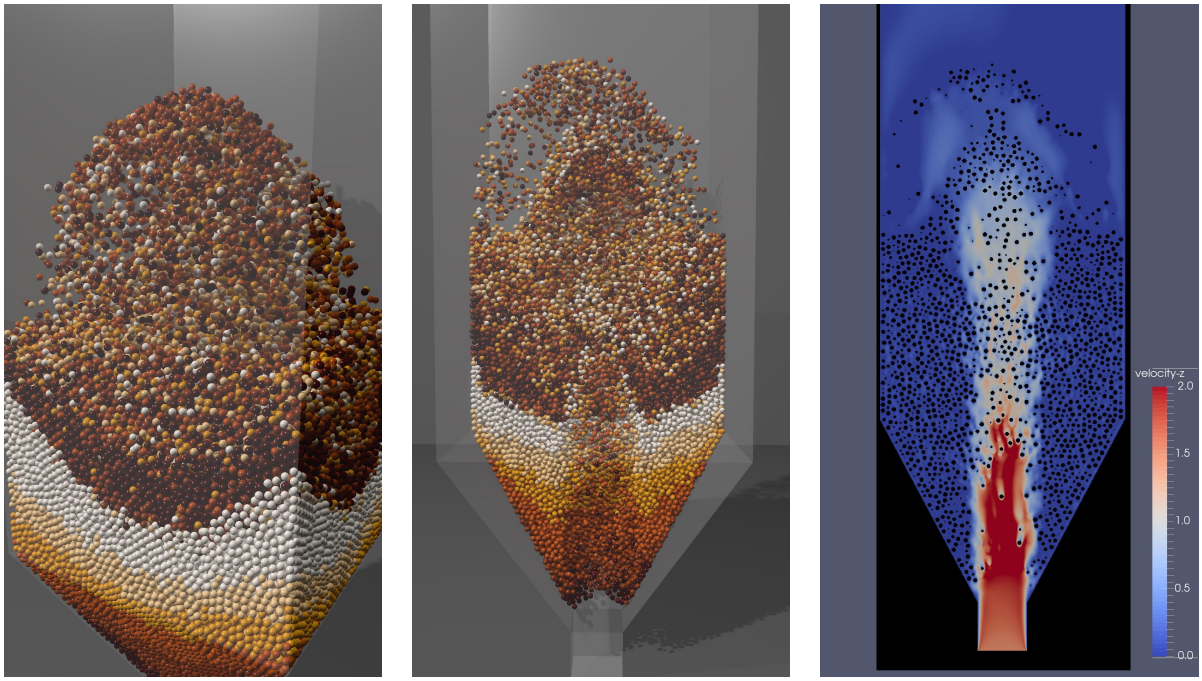


(d) $t = 2.50$ ms.

Fig.3.22: Results of a resolved-particle simulation for a spouted bed. The left panels show all particles colored by their initial position. The center panels show the particles on the far side cut at the center of the domain. The right panels indicate the velocity field in the vertical direction. (Continued.)



(e) $t = 3.33$ ms.



(f) $t = 12.5$ ms.

Fig.3.22: Results of a resolved-particle simulation for a spouted bed. The left panels show all particles colored by their initial position. The center panels show the particles on the far side cut at the center of the domain. The right panels indicate the velocity field in the vertical direction. (Continued.)

3.5.2　舞い落ちるイチョウの葉の流体構造連成解析

名刺や木の葉などの薄く軽い物体が空気中を落下する際，それらは空気からの抵抗を受けてひらひらと揺れながら落下する．左右に規則的に揺れながらの落下やランダムに揺れながらの落下，水平軸周りの回転を伴う落下など，薄く軽い物体の落下は非常に複雑である [100][101]．特に，イチョウの葉は薄肉の葉の部分と長い柄の部分から構成され，鉛直軸周りの回転を伴いながら落下する場合もあり，その落下挙動は特徴的である．

格子ボルツマン法と非球形粒子モデルを用いた個別要素法を組み合わせ，複数のイチョウの葉が舞い落ちる流体構造連成解析を行う．Fig.3.23 のように，実物を参考にして 3 次元のポリゴンデータを作成する．イチョウの複雑な形状を表現するために，ポリゴンデータを覆う直交格子を設定し，各格子点で最近傍のポリゴンからの距離を計算して距離関数を作成する．イチョウの葉の並進と回転に合わせて符号付き距離関数の座標変換を行い，格子ボルツマン法の移動境界条件における壁の位置の計算に用いる．距離関数のゼロ等値面に微小な球形粒子を配置して粒子モデルを作成し，他のイチョウの葉との接触判定と接触力の計算を行う．

イチョウの葉の大きさは 100 mm 程度であり，厚さは 5.0 mm と実際よりも厚く設定している．2.0 m × 2.0 m × 4.1 m の計算領域を 2.0 mm の流体計算格子で解像する．イチョウの葉の厚さに対して 2.5 メッシュを割り当てている．流体は空気を仮定し，密度 1.205 kg/m^3，動粘度 1.512×10^{-5} m^2/s と設定する．時間刻み幅は 0.1 ms である．格子点数は $1024 \times 1024 \times 2048$ であり，TSUBAME2.5 の GPU Tesla K20X を 128 台用いる．1GPU あたり $1024 \times 128 \times 64$ 格子を割り当てる．初期角度をランダムに設定したイチョウの葉を計算領域の上方に配置し，初速を 0 として落下させる．境界条件として，イチョウの葉の表面に non-slip 条件，計算領域の境界に non-slip 条件を課す．計算安定化のために，格子ボルツマン法に MRT モデルの衝突項と LES 乱流モデルを導入する．

計算結果を Fig.3.24 に示す．イチョウの葉がひらひらと落下する様子や，回転しながら落下する様子を再現できた．イチョウの葉同士の接触を考慮することで床に堆積する様子を計算できた．計算領域中央での y-z 平面における速度場を Fig.3.25 に示す．イチョウの葉の運動により空気が乱され，多くの細かい渦が形成されていることがわかる．また，40,000 ステップの計算に対し，計算時間は 48 時間ほどであった．

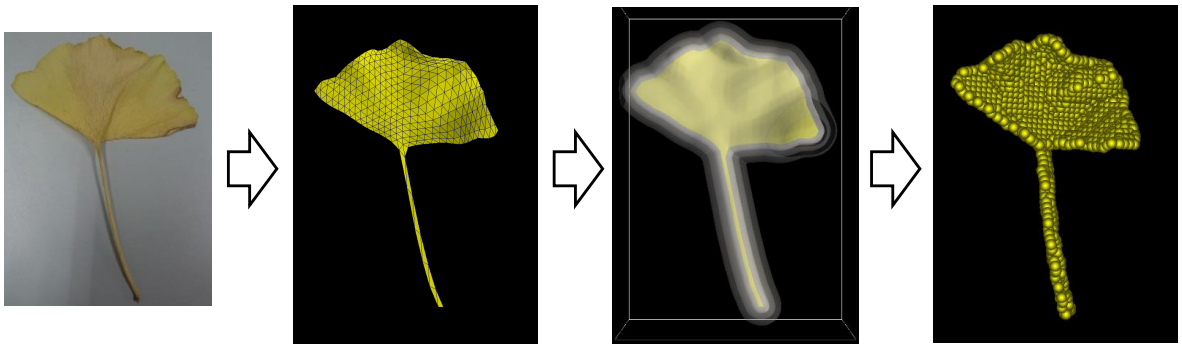
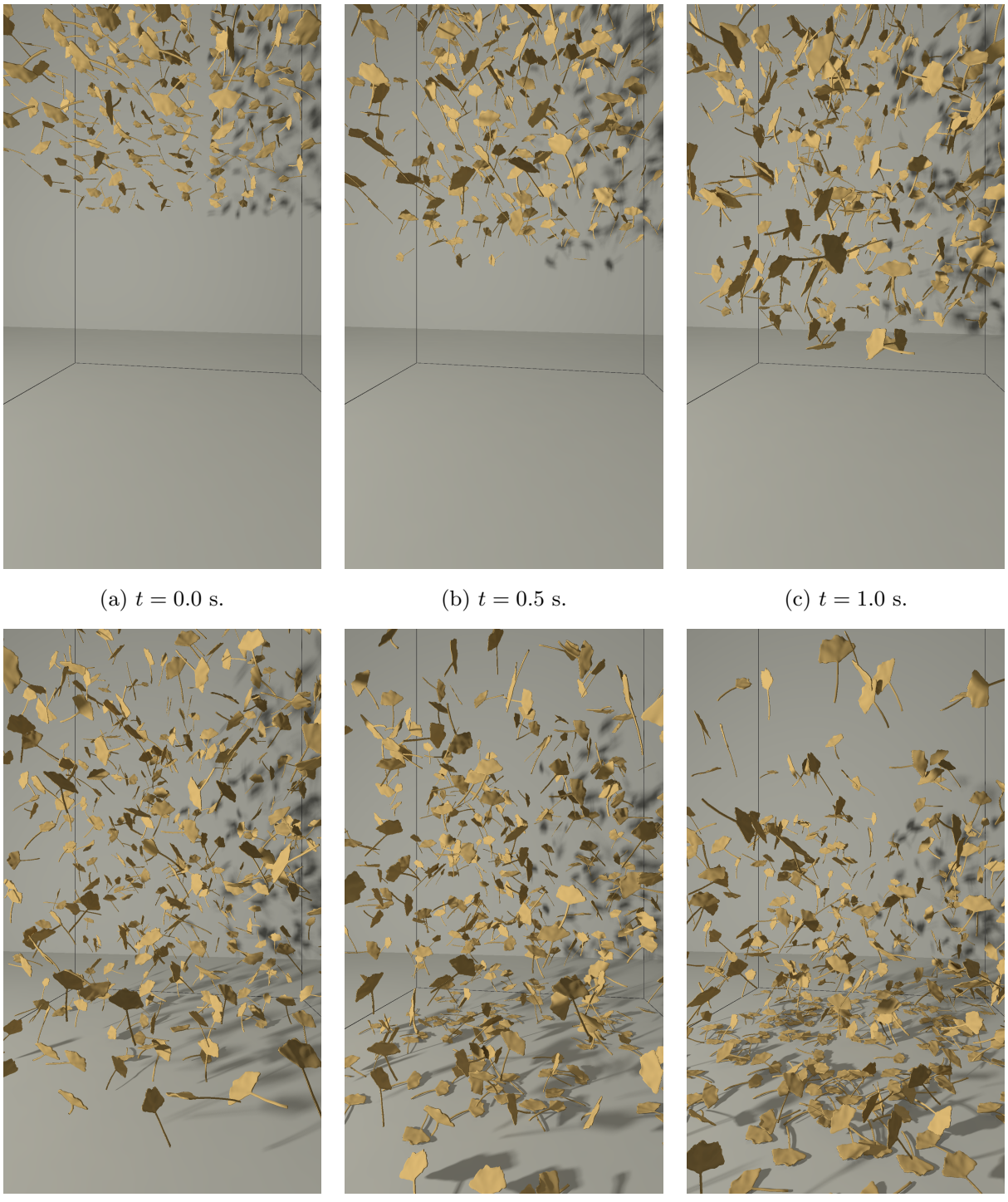


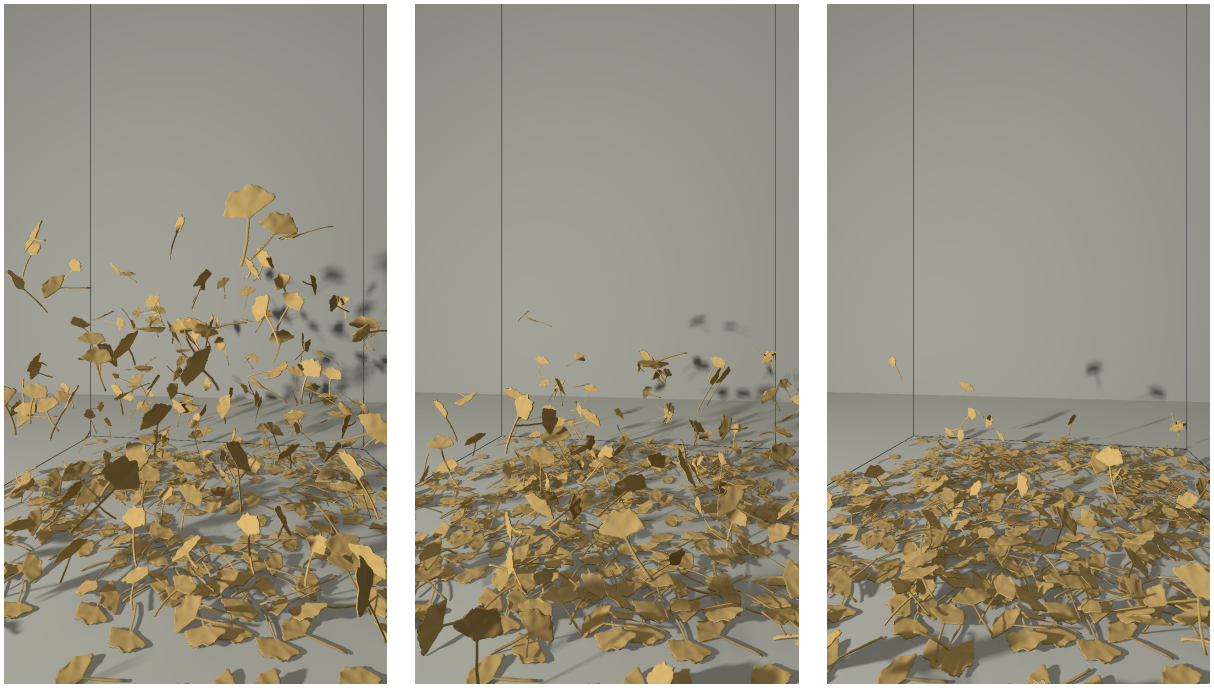
Fig.3.23: Modeling of a ginkgo leaf.



(a) $t = 0.0$ s.　(b) $t = 0.5$ s.　(c) $t = 1.0$ s.

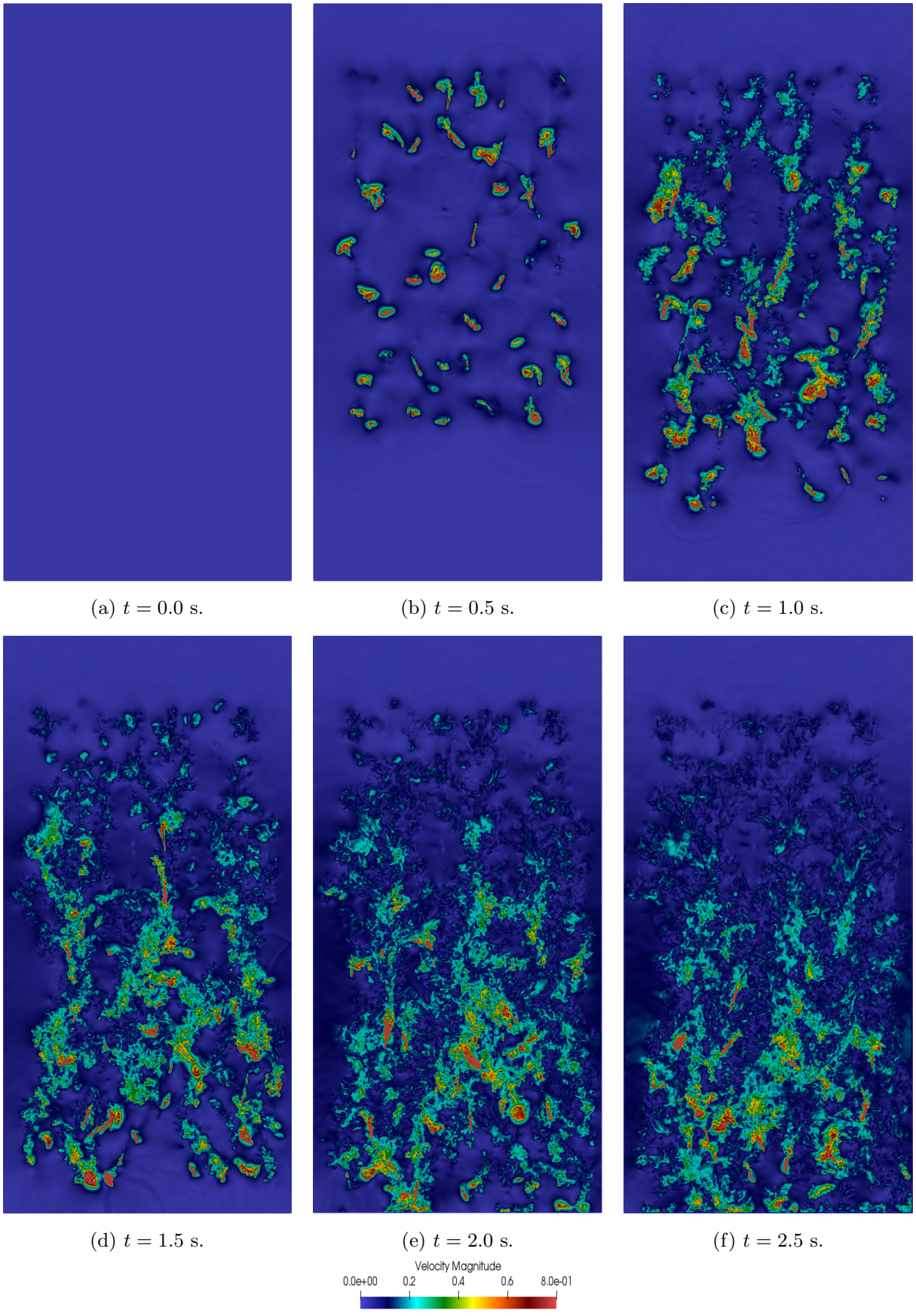
(d) $t = 1.5$ s.　(e) $t = 2.0$ s.　(f) $t = 2.5$ s.

Fig.3.24: Snapshots of a simulation of falling ginkgo leaves.



(g) $t = 3.0$ s.　(h) $t = 3.5$ s.　(i) $t = 4.0$ s.

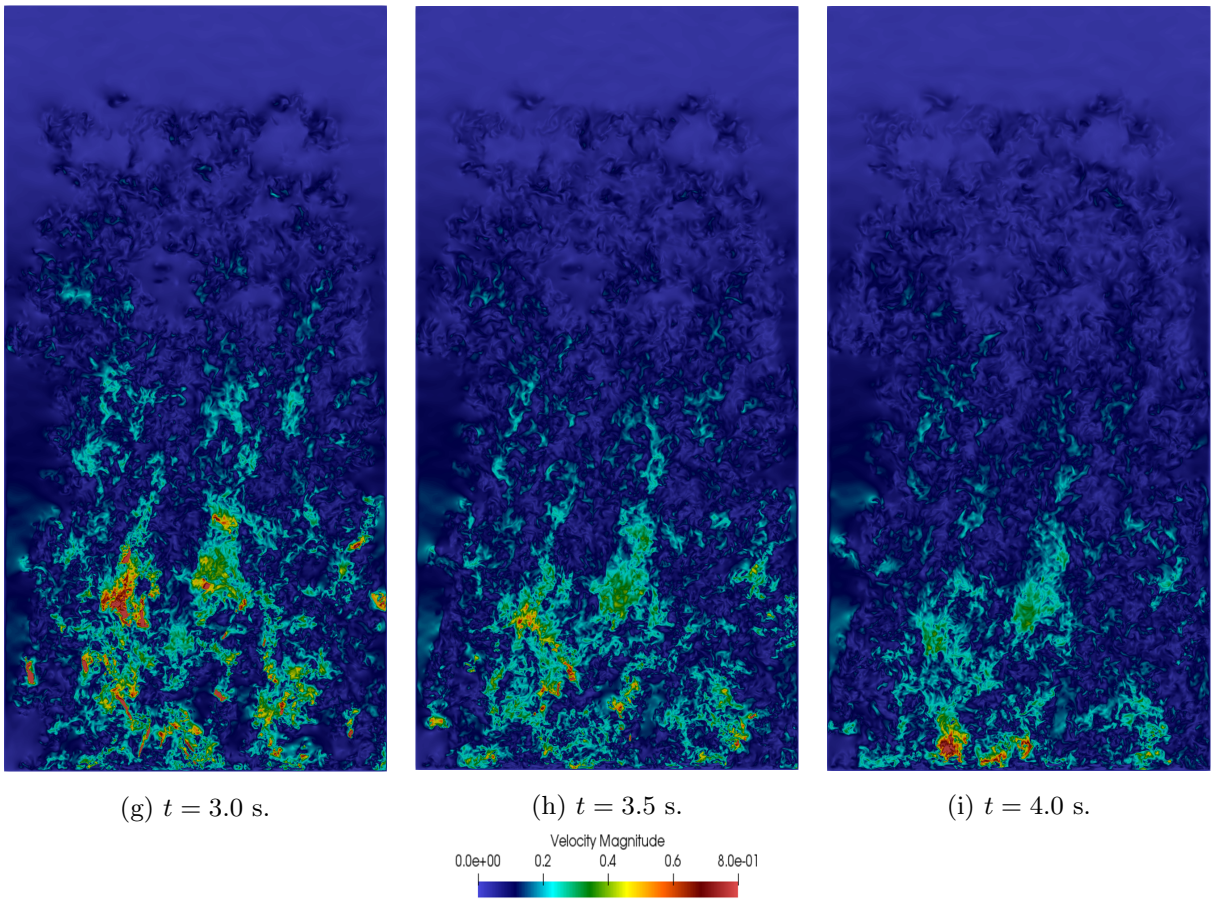
Fig.3.24: Snapshots of a simulation of falling ginkgo leaves. (Continued.)



(a) $t = 0.0$ s.　(b) $t = 0.5$ s.　(c) $t = 1.0$ s.

(d) $t = 1.5$ s.　(e) $t = 2.0$ s.　(f) $t = 2.5$ s.

Fig.3.25: Velocity profile of a simulation of falling ginkgo leaves.



(g) $t = 3.0$ s.　(h) $t = 3.5$ s.　(i) $t = 4.0$ s.

Fig.3.25: Velocity profile of a simulation of falling ginkgo leaves. (Continued.)

3.5.3　流木を含む津波流

本項の内容は研究業績論文 (1) の内容を含む．沿岸に建設する建築構造物の設計には，津波が構造物に与える影響を十分に理解する必要があるが，構造物に作用する津波の影響はまだ十分に理解されておらず，内閣府のガイドライン [102] の設計式では静水圧で近似した津波波力が用いられている．津波の実スケールの実験は困難であるため数値シミュレーションは有効な手段であり，VOF（Volume of Fluid）法を用いた橋梁に作用する津波波力の数値解析では，津波が構造物に衝突した直後に大きな波力が発生することが明らかになっている [103]．東日本大震災の津波では，非常に多くの瓦礫が津波とともに流されたことは良く知られている．そのため，津波波力だけでなく，津波で流された船や瓦礫などの浮遊物体が建物などの構造物に衝突した際の影響をする必要があると考えられる．

格子ボルツマン法による自由界面流れの計算と個別要素法による物体の計算を組み合わせ，浮遊物体を含む津波のシミュレーションを行う．神戸大学にある津波の水槽実験の設備を模擬し，計算領域を長さ 30 m，幅 6 m，高さ 2 m とする．計算領域の端に長さ 10 m，幅 6 m，高さ 1 m の台を配置し，その上に水位 0.4 m の水を配置する．長さ 6 m のスロープを水が下ることで津波流を発生させる．流体は水を仮定し，密度 1000 kg/m^3，動粘度 1.004×10^{-6} m^2/s とする．流木を模擬した直方体の物体を床に 18 個配置する．物体の密度は 800 kg/m^3 であり，寸法は長さ 2 m，幅 0.21 m である．浮遊物体の影響を評価するため，物体を含まない計算も行う．流体計算の格子幅を 1 cm とし，物体の幅に対して 21 格子を割り当てて計算する．格子点数は $200 \times 600 \times 3000$ である．格子ボルツマン法の時間刻み幅を 60 μs，個別要素法の時間刻み幅を 12 μs と設定する．壁面と物体境界に対して non-slip 境界条件を設定する．計算安定化のために，格子ボルツマン法の計算にキュムラントモデルを用いる．TSUBAME3.0 の GPU Tesla P100 を 24 台用い，物理時間で 20 秒までの計算を行う．タイムステップ数は 33 万である．

物体を含まない津波流の計算結果を Fig.3.26 に，物体を含む津波流の結果を Fig.3.27 に示す．スロープを下ることで波が発生し，物体を含む条件では物体が波に流されていく様子が確認できる．流された物体は右の壁面に衝突し，その後，密度が水よりも小さいため水面に浮遊している．

流木が津波流に与える影響を評価するために，右端の壁面に作用する津波波力と流木の衝突力を測定し，流木を含まない場合と比較する．津波波力は壁面近傍の圧力から見積もることができ，右側の壁面に作用する津波波力の時刻歴を Fig.3.28 に示す．赤が流木を含む場合，黒が流木を含まない場合の結果である．右側の壁面に作用する流木の衝突力を個別要素法のバネとダッシュポットのモデルから計算し，その時刻歴を Fig.3.29 に示す．流木を含む場合の津波波力は，流木を含まない場合よりも小さく，流木により流れが妨げられたためだと考えられる．流木を含む場合では，8 秒付近で波力のスパイクが観測された．Fig.3.29 から 8 秒付近で流木が右の壁面に衝突していることが分かる．流木と壁に挟まれた流体の圧力が大きくなったため波力のスパイクが現れたと考えられる．また，流木の衝突力の最大値は約 300 kN であり，津波波力の 15 倍以上の力が働いていることが分かった．このことから，構造物への津波の影響を評価する際には，瓦礫や流木などを考慮することの必要性が明らかになった．

次に，既に浸水した領域に津波の第 2 波が来た場合を模擬した計算を行う．先程の計算と異なり，初期時刻において床に水位 0.4 m の水を配置する．物体は床に置くのではなく，水面に自由落下させる．他

の条件は先程の計算と同じであり，物体の影響を評価するために物体を含まない場合の計算も実施する．

初期水位を設定した物体を含まない津波流の計算結果を Fig.3.30 に，物体を含む津波流の結果を Fig.3.31 に示す．水面を伝搬する波により物体は多少移動するが，初期水位を設定しない場合ほど流されず，右の壁に衝突しなかった．右側の壁面に作用する津波波力の時刻歴を Fig.3.32 に示す．初期水位を設定した場合は，物体を含まない場合と含む場合で津波波力に大きな差異はないことが確認できた．

実際の津波実験を模擬した 3.6 億格子点の津波解析を，24 台の GPU を用いて 33 万タイムステップの計算を約 24 時間で完了した．数億格子点の自由界面流れと物体の連成解析が，GPU を用いた並列計算を行うことにより容易に実行可能であることが示された．

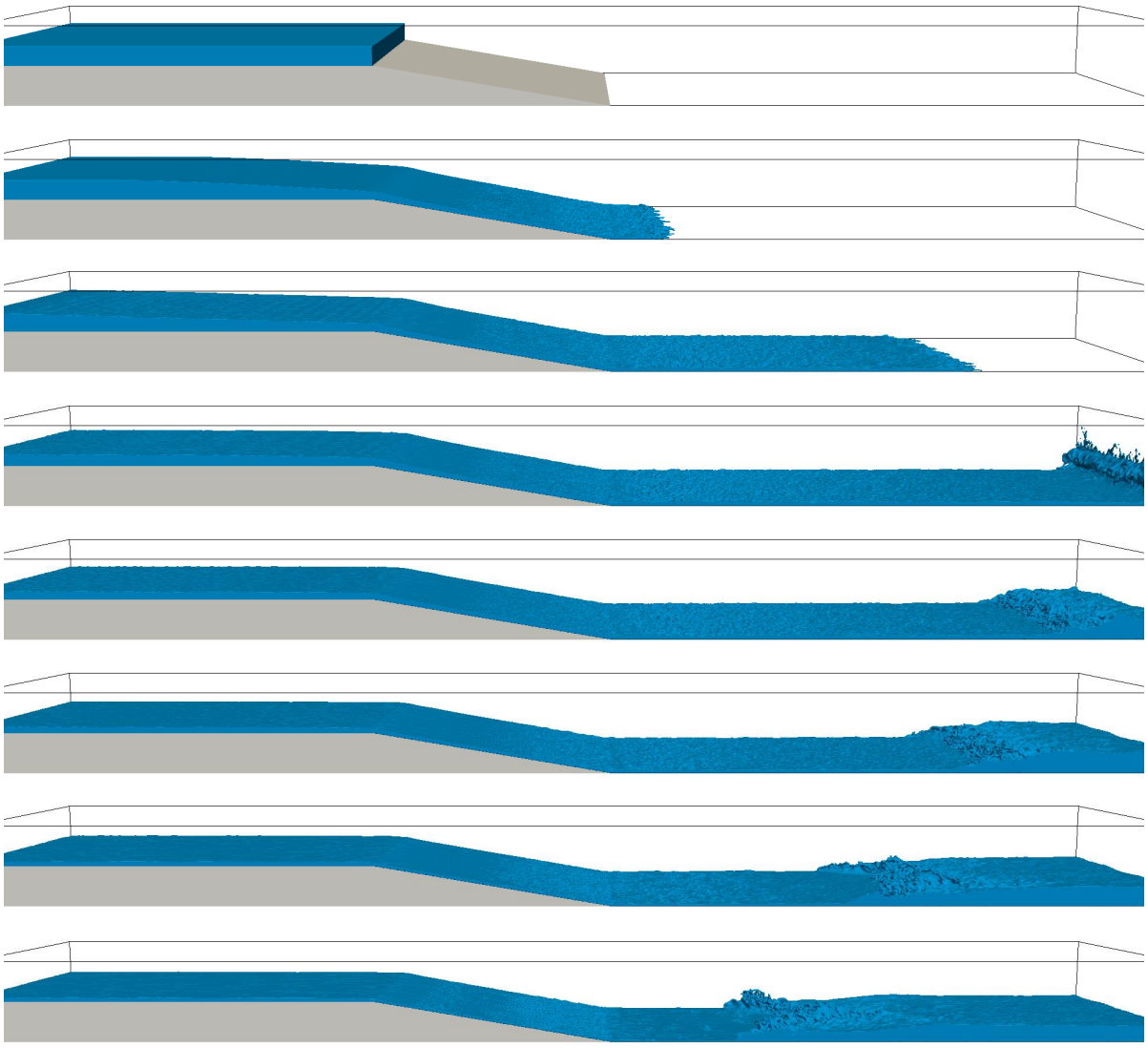


Fig.3.26: Simulation results of the tsunami on the dry floor without driftwoods at 8 physical time instants $t = 0, 2.4, 4.8, 7.2, 9.6, 12.0, 14.4, 16.8$ s (from the upper panel to the lower panel).

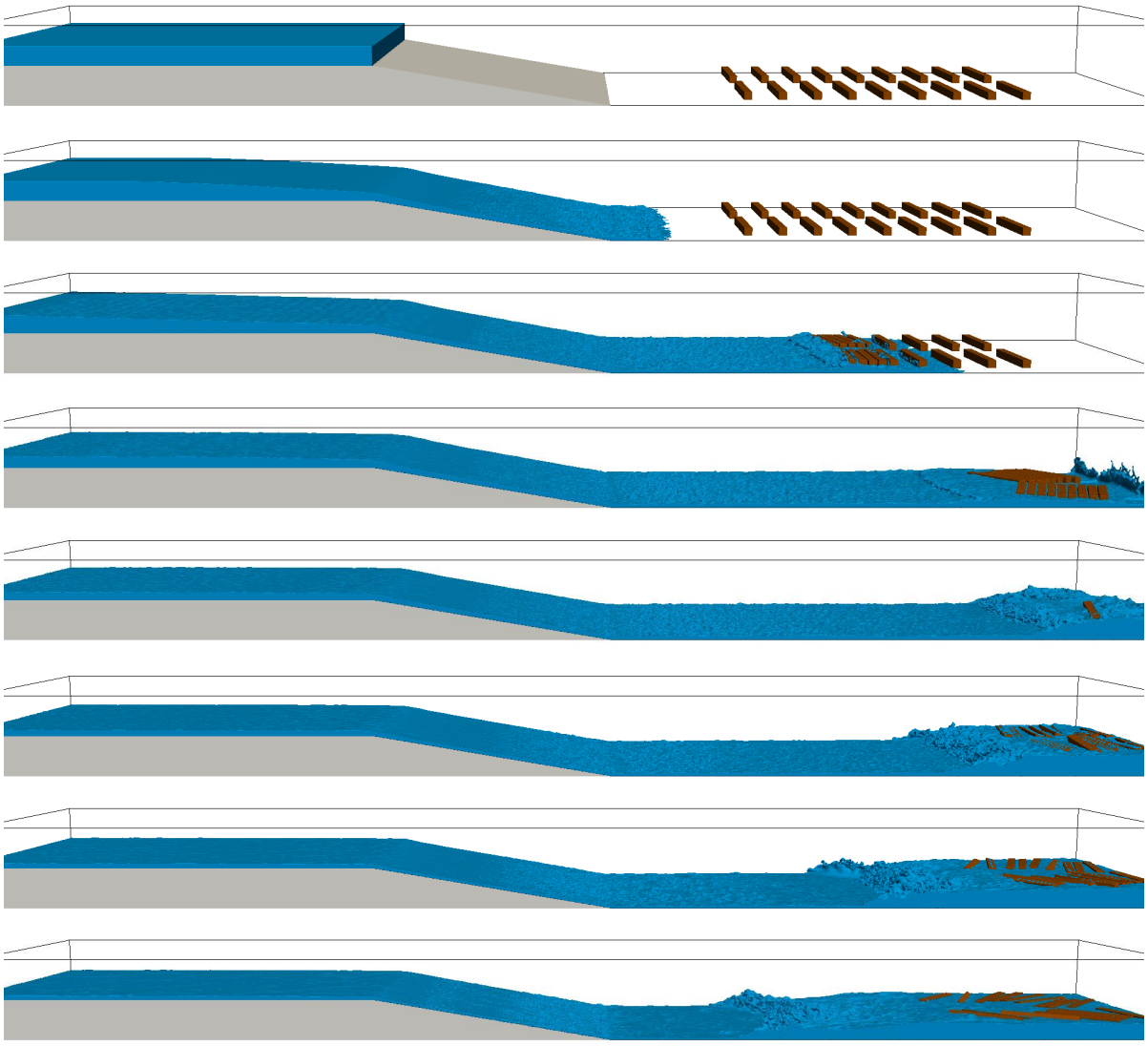


Fig.3.27: Simulation results of the tsunami on the dry floor with 18 driftwoods at 8 physical time instants $t = 0, 2.4, 4.8, 7.2, 9.6, 12.0, 14.4, 16.8$ s (from the upper panel to the lower panel).

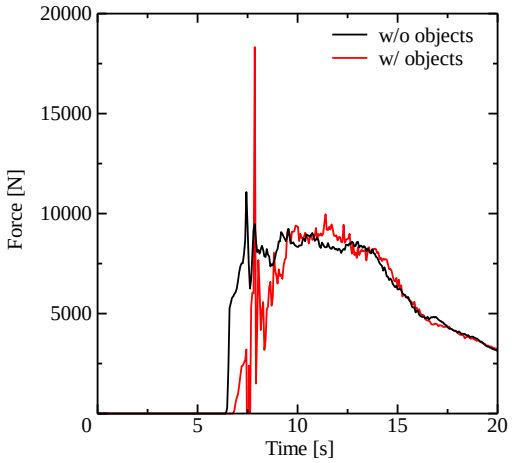


Fig.3.28: Tsunami wave forces acting on the right wall.

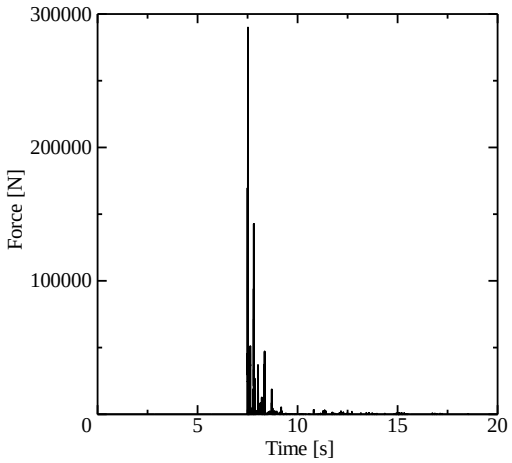


Fig.3.29: Impact force of driftwoods acting on the right wall.

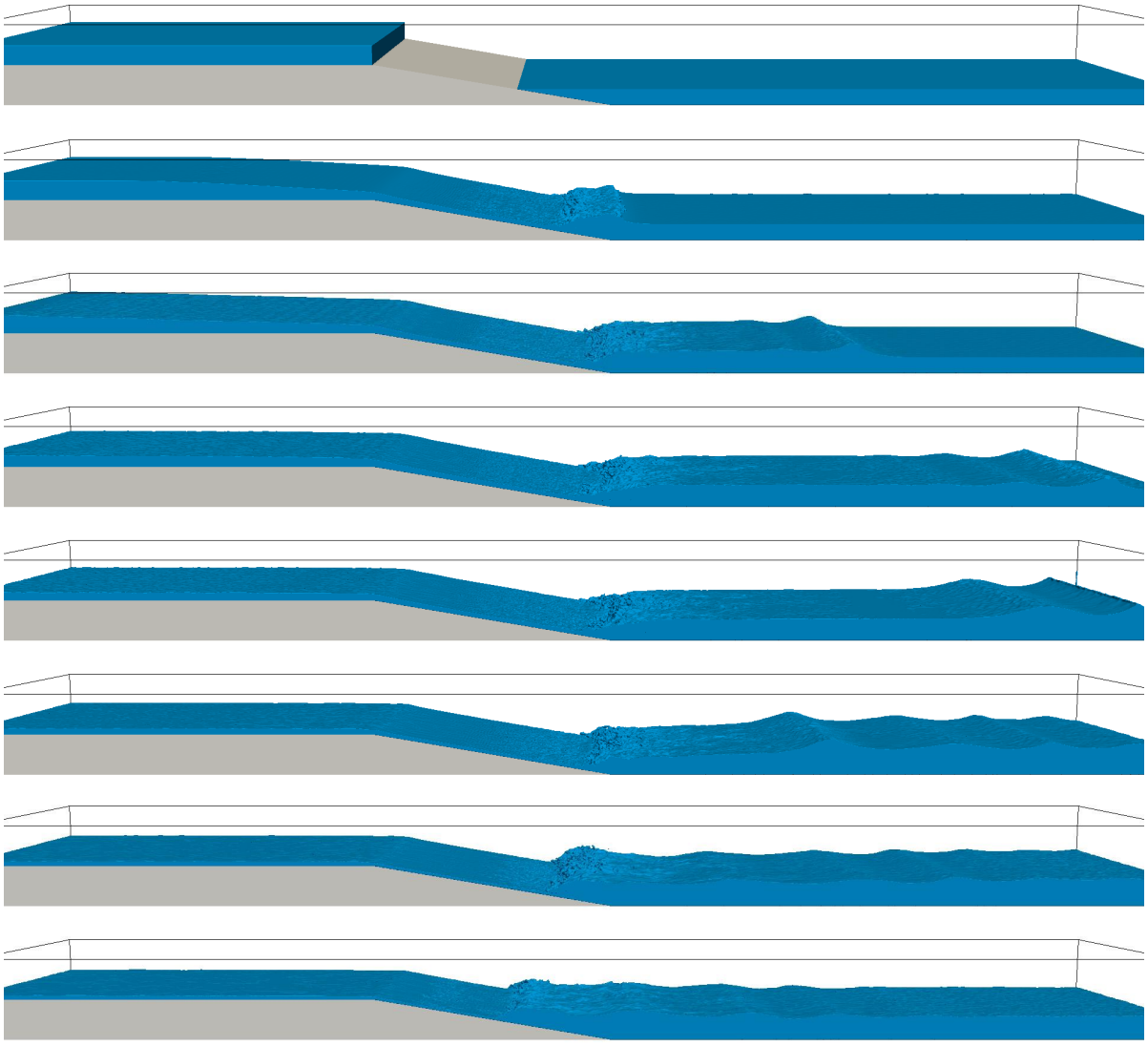


Fig.3.30: Simulation results of the tsunami on the wet floor without driftwoods at 8 physical time instants $t = 0, 2.4, 4.8, 7.2, 9.6, 12.0, 14.4, 16.8$ s (from the upper panel to the lower panel).

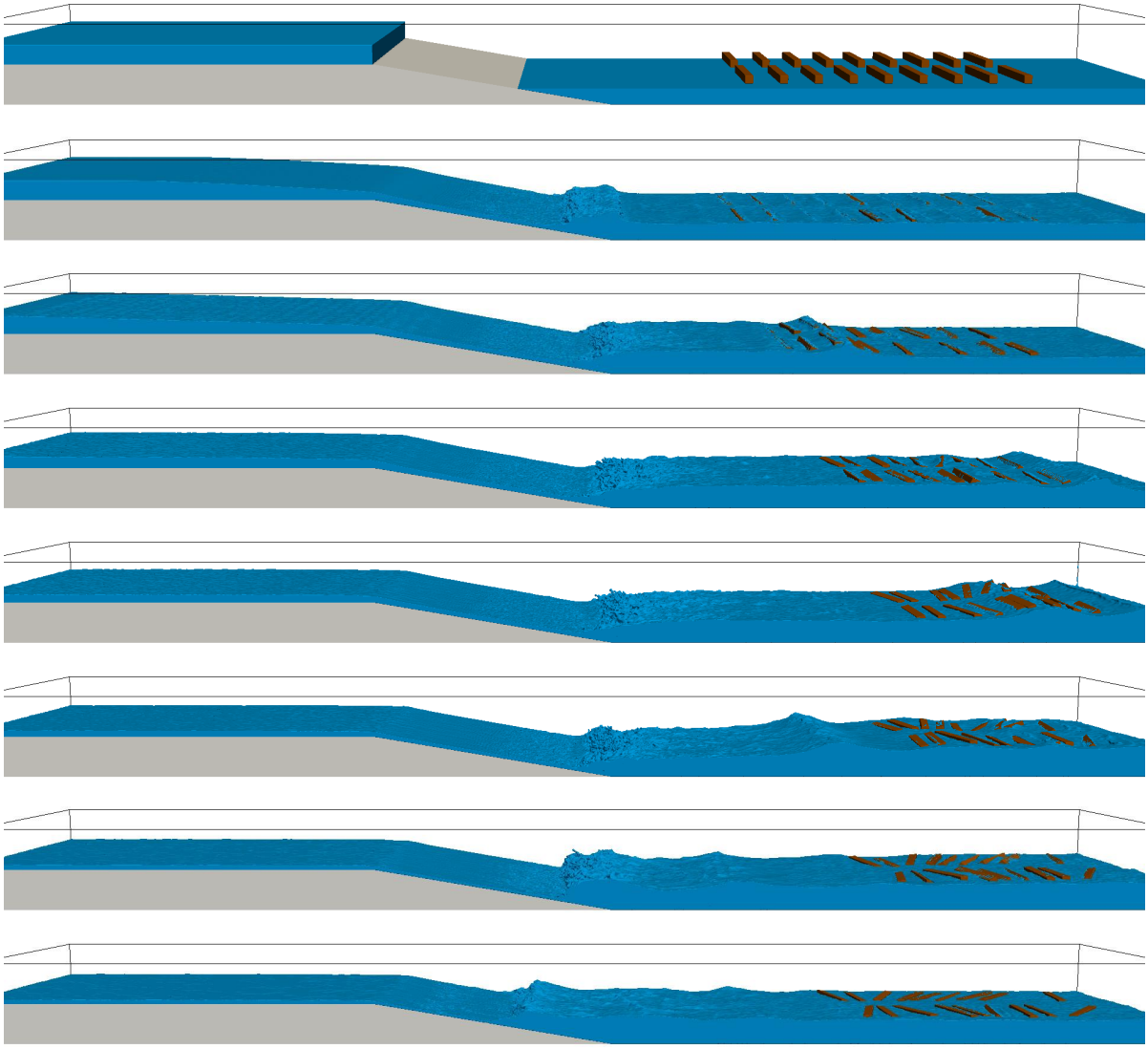


Fig.3.31: Simulation results of the tsunami on the wet floor with 18 driftwoods at 8 physical time instants $t = 0, 2.4, 4.8, 7.2, 9.6, 12.0, 14.4, 16.8$ s (from the upper panel to the lower panel).

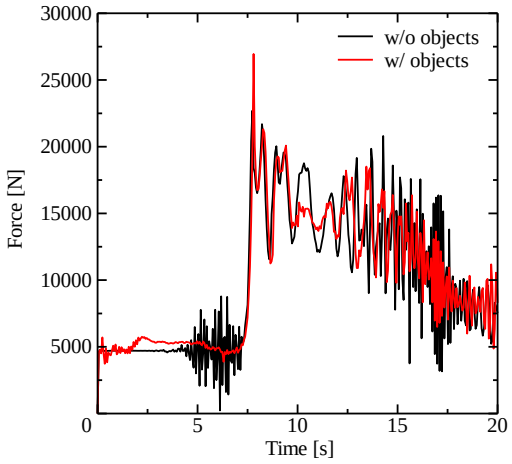


Fig.3.32: Wave forces of the tsunami on the wet floor acting on the right wall.

3.6 複数 GPU を用いた大規模混相流解析のまとめ

本章では，2 章で提案した計算手法の複数 GPU 実装を行い，複数台の GPU を用いた大規模混相流シミュレーションを実施した．

格子ボルツマン法やフェーズフィールド法などの格子計算と個別要素法の粒子計算の単一 GPU 実装について述べた．個別要素法の計算で最も計算時間のかかる近傍粒子探索に対し，リンク・リスト法，ハッシュ法，リンク・リスト法を粒子登録法に併用した手法，ハッシュ法を粒子登録法に併用した手法の 4 つ近傍探索手法を実装し，計算時間を比較した．リンク・リスト法を粒子登録法に併用した手法が最も計算時間が短く，GPU による個別要素法の計算に適していることが明らかになった

複数 GPU 実装では，格子ボルツマン法とフェーズフィールド法の格子点数に比べて個別要素法で計算する物体数や粒子数が非常に少ないことに着目し，格子計算は 2 次元領域分割法による複数 GPU 実装を行い，個別要素法は並列化を行わずに各 GPU が全ての粒子に関して冗長に計算する方法を提案した．東京工業大学の GPU スパコン・TSUBAME3.0 の GPU を用いて弱スケーリングと強スケーリングの性能測定を行った．弱スケーリングでは，256GPU での並列化効率は約 83% であり 16GPU に対して 13.3 倍の高い実行性能を得られ，複数 GPU 計算をしていない個別要素法の計算のオーバーヘッドは 8.4% であった．一方，強スケーリングでは GPU 数の増加に伴う並列化効率の低下し，これは GPU 間の通信のオーバーヘッドが原因であると考えられる．

等間隔直交格子を用いて混相流の実問題に対する大規模計算を東京工業大学の TSUBAME2.5 および TSUBAME3.0 を用いて実行した．多数の固体粒子を含む粒子系混相流の解析として，粉体を流動化させて混合や乾燥を行う噴流層のシミュレーションを行った．典型的な噴流層で現れる空気が流れる領域（Spout），固体粒子が巻き上げられる領域（Fountain），流路の周りで粒子が沈んでいく環状の領域（Annulus）の 3 つの領域が形成され，循環運動による粒子の混合をシミュレーションで再現することに成功した．複雑形状の物体を含む流れとして，512 枚のイチョウの葉が舞い降りる計算を 128 台の GPU を用いて行い，イチョウの葉がひらひらと落下する様子や，回転しながら落下するイチョウの葉の特徴的な現象を再現することに成功した．自由界面を含む流れとして，水槽実験を模擬した 18 個の流木を含む津波流の計算を実行した．津波で流された流木が壁面に衝突した際の接触力は津波波力の 15 倍以上になることが明らかになり，この結果から，構造物への津波の影響を評価する際には，瓦礫や流木などの物体の影響を考慮することの重要性を示せた．この津波計算は 3.6 億格子を用いた大規模計算であるが，TSUBAME3.0 の GPU Tesla P100 を 24 台用いて 24 時間で計算を完了しており，提案した格子ボルツマン法による完全陽解法の混相流解析手法と GPU を用いた並列計算により，大規模計算が短時間で実現可能になった．

第 4 章

AMR 法を導入した格子ボルツマン法の大規模 GPU 計算

4.1　木構造を用いたブロック構造 AMR 法

直交格子を用いる場合，必要とされる解像度に合わせて，計算領域全体を同一の解像度の格子で分割する．単純なメッシュ構造であるため，スーパーコンピュータによる直交格子を用いた大規模計算が数多く報告されている [104][105][106]．しかし，実際の流体現象で計算領域全体に高い計算精度が要求されることは稀であり，より大規模な計算を実現するためには，計算コストとメモリ使用量を削減する必要がある．直交格子の計算精度を維持しつつ，メモリ使用量と計算コスト減らす方法として，AMR（Adaptive Mesh Refinement）法が Beger ら [18] により提案された．高い計算精度が要求される領域に高解像度の格子を割り当て，高解像度の格子が必要でない領域には粗い格子を配置する．Beger らが提案した方法はパッチ型と呼ばれ，解像度の異なる直交格子を任意の位置に配置する方法である．もう一つの方法として，木データ構造を用いて再帰的に計算領域を分割し，分割の深度によって解像度を変えるツリー型 [22] がある．ツリー型は格子の細分化がパッチ型よりも容易に行なえるため，ツリー型の AMR の法を用いた計算が多く行われている．特に，木構造で表現された領域に直交格子を割り当てるブロック構造は，計算セルの並びが規則的で単純であるため，GPU などのアクセラレータや並列計算に適している．本研究では，ブロック構造格子を用いたツリー型の AMR 法を格子ボルツマン法に導入する．

4.1.1　木データ構造

木データ構造を用いた AMR 法での再帰的な格子生成の図を Fig.4.1 に示す．木構造に基づく格子生成では，まず，計算領域全体に対応する根ノードを作成する．木データ構造の各節点はノードと呼ばれ，始点となる一番上のノードを根ノード，実線でつながれた上側のノードを親ノード，下側のノードを子ノードと呼ぶ．子ノードを持たないノードはリーフと呼ばれる．２次元計算の場合は各ノードが４つの子ノードを持つ Quadtree データ構造，３次元計算の場合は各ノードが８つの子ノードを持つ Octree データ構造を用いる．計算領域は４つ（２次元）または８つ（３次元）の等しい領域に分割され，これら

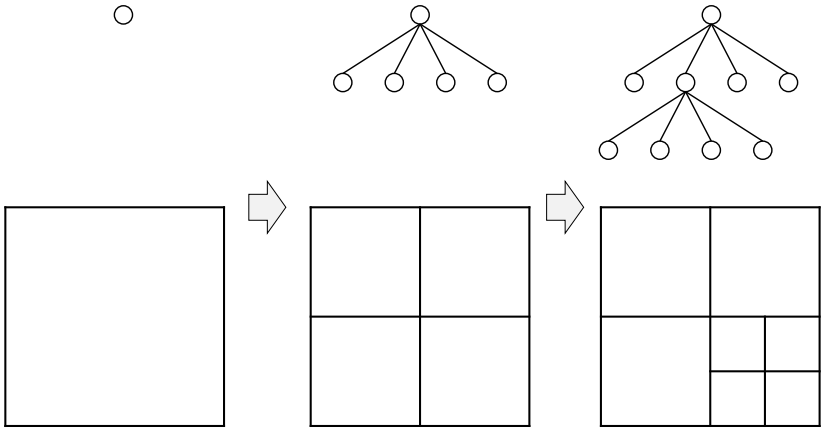


Fig.4.1: Recursive mesh generation based on a tree structure.

の領域はそれぞれ根ノードの子ノードに割り当てられる．これらの子ノードに割り当てられた領域をさらに再帰的に分割していく．ノードを細かく分割するかの決定は，分割領域の物理的特性に基づく．計算の安定性を維持するために，Fig.4.2 の左図ように隣接する領域の大きさが 2 倍以上にならないようにする．

ひとつの木構造は正方形（2 次元）または立方体（3 次元）の領域にしか対応することができない．しかし，流体シミュレーションでは長方形や直方体の計算領域を用いることも多い．ひとつの木構造で細分化した領域の一部を利用する方法もあるが，高アスペクト比の領域を表現する場合に無駄な領域が多くなってしまう．本研究では，Fig.4.3 のように複数の木構造を用いて長方形や直方体の計算領域を表現する．例えば，Fig.4.3 のようにアスペクト比が 3:2 の領域の細分化を行う場合は，木構造を 6 個（3×2）用いる．これにより，計算に利用されない無駄な領域を削減する．

4.1.2　ブロック構造格子

木構造で分割された領域に対して，一定数のセル数で構成される等間隔直交格子を割り当てる．Fig.4.4 は Quadtree で分割された 2 次元の領域に対して，木構造の末端であるリーフに対して 4×4 の計算セルを配置した例でり，根ノードから離れたリーフほど高解像度の格子が割り当てられる．リーフに割り当てられる均一格子の塊はブロックと呼ばれる．ブロック内の格子点数は AMR 法の計算の性能に大きく影響する．ブロック内の格子点数が少ないほど Fig.4.5 のようにプロファイルにより適合した格子が生成でき，格子点数を削減できる．一方，ブロックの格子点数が多いと高解像度格子は大まかに割り当てられるが，リーフ数が少なく木構造を小さくできるため，木構造の大きさに性能が律速される格子生成のプロセスは高速化できる．

ブロック構造格子を格子ボルツマン法に導入する場合，計算点の配置にはセルの頂点に配置するノードセンター（Fig.4.6(a)）とセルの中心に配置するセルセンター（Fig.4.6(b)）が考えられる．ノードセ

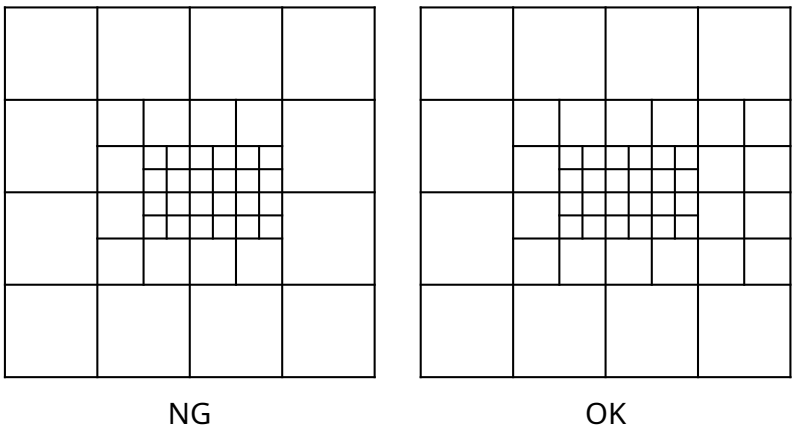


Fig.4.2: Proper refinement. In the left mesh, the resolution of neighbor regions is four times higher, which causes instability of stencil computations.

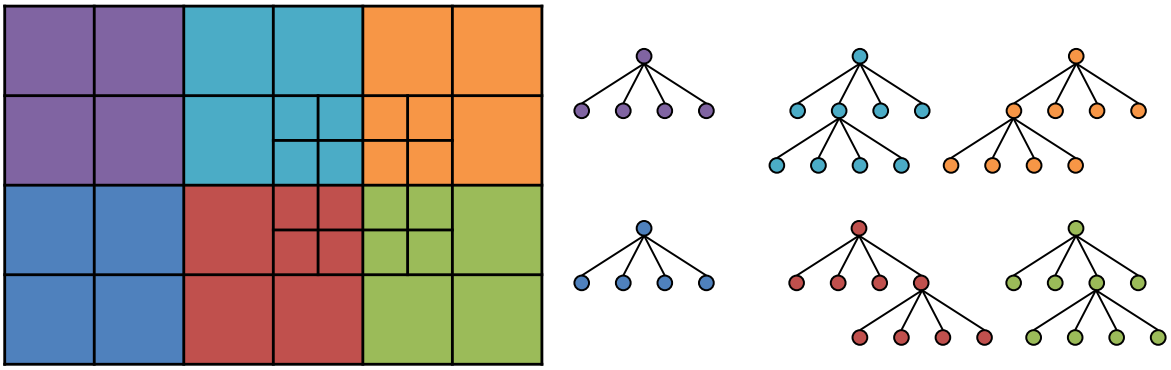


Fig.4.3: Rectangular computational domain represented by multiple tree structures. Colors indicate each tree structures and the corresponding domain.

ンターはセルの頂点に計算点を定義するため，例えばブロック内が $4 \times 4 \times 4$ のセルであった場合は，$5 \times 5 \times 5$ の計算点がブロックに割り当てられる．ブロックの境界に配置された計算点は，隣接しているブロックの計算点と重なって配置される．一方，セルセンターではセル中心に定義するため，ブロック内の計算点数はセル数と同じ $4 \times 4 \times 4$ であり，ノードセンターよりも格子点数が少なくなる．ブロックのセル数が $4 \times 4 \times 4$ の場合はノードセンターに比べて格子点数を 51.2%，$8 \times 8 \times 8$ の場合は 70.2%，$16 \times 16 \times 16$ の場合は 83.3% に削減できる．ノードセンターを用いる利点として，格子の細分化（格子を細かくする処理）や粗大化（格子を粗くする処理）の処理を行う際に，自身のブロック内の計算点の値のみを用いて新しい格子の値を保管することができる点が挙げられる．セルセンターの格子配置を用いる場合は，細分化の処理に隣接するブロックにある計算点の値が必要となり，処理が複雑となる．本研究では，メモリ使用量を考慮し，セルの中心に計算点を定義するセルセンターを用いる．

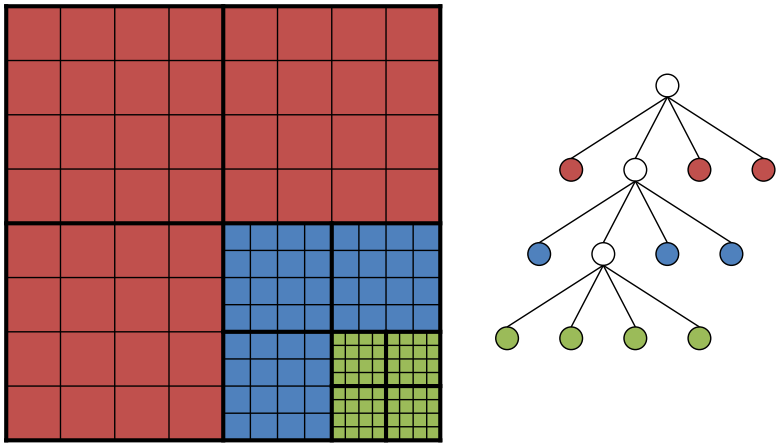
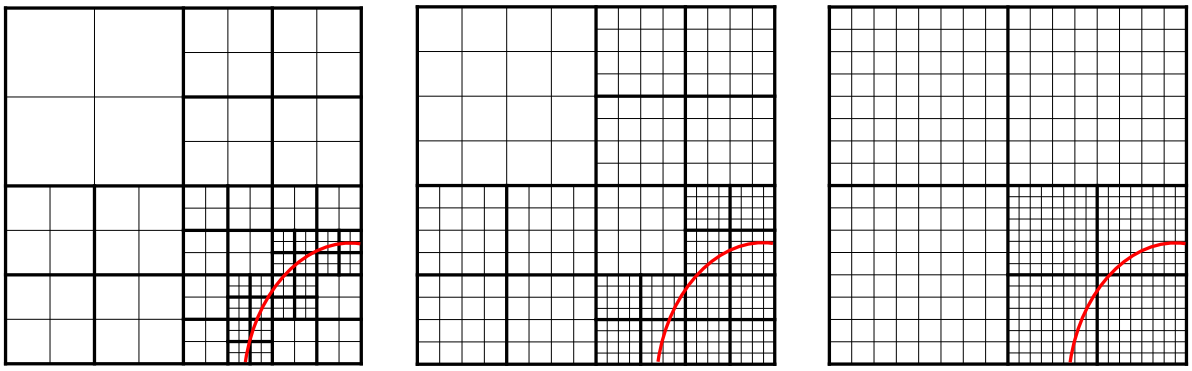


Fig.4.4: A block-structured mesh using a quadtree. Each leaf has a block of 4×4 cells. Colors indicate the mesh resolution.



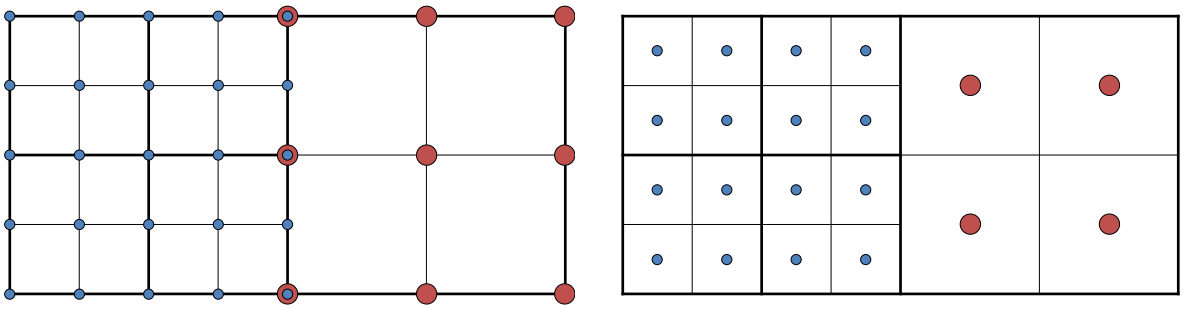
(a) 2×2 cells in a block.　(b) 4×4 cells in a block.　(c) 8×8 cells in a block.

Fig.4.5: Effects of the number of cells in a block. Thick lines and thin lines indicate blocks and cells, respectively. High-resolution cells are assigned to the red line.

4.1.3　解像度差の補間

複数の格子解像度を用いて格子ボルツマン法の計算をする場合，計算領域全体で音速

$$c_{\mathrm{s}} = \sqrt{\frac{1}{3}}\frac{\Delta x}{\Delta t} \tag{4.1}$$



(a) node-centred lattice.　　(b) cell-centred lattice.

Fig.4.6: Placement of lattice points in a block-structured mesh.

を一定にする必要がある．木構造に基づいて格子細分化を行う場合，木構造が深くなるに従い格子幅 Δx は半分になるため，粗い格子の格子幅 Δx^{c} と一段細かい格子の格子幅 Δx^{f} には

$$\Delta x^{\mathrm{f}} = \frac{1}{2}\Delta x^{\mathrm{c}} \tag{4.2}$$

の関係があり，音速を一定にするため細かい格子の時間刻み幅 Δt^{f} を

$$\Delta t^{\mathrm{f}} = \frac{1}{2}\Delta t^{\mathrm{c}} \tag{4.3}$$

と設定する．そのため，Fig.4.7 のように，粗い格子が 1 タイムステップ進む間に，一段細かい格子は 2 ステップ，さらに細かい格子は 4 ステップの計算が行われる．このように，解像度ごとに時間刻み幅が異なり，ある物理時間まで計算するまでに必要なステップが異なることを以降マルチタイムステップと呼ぶ．

マルチタイムステップの計算では，粗い格子と細かい格子の時刻が一致するステップ（Fig.4.7 の赤い丸で囲まれたステップ）で，粗い格子と細かい格子が接している境界において同期の処理が必要である．同期の処理では，解像度が異なる境界近傍において，Fig.4.8 のように細かい格子に重ねて粗い格子を仮想的に定義し（ピンク色の点），細かい格子の分布関数 f^{f} から粗い格子の分布関数 f^{c} を補間により内挿する．補間には線形補間を用い，Fig.4.8 の点 (I, J) における粗い格子の分布関数 $f^{\mathrm{c}}_{I,J}$ を

$$f^{\mathrm{c}}_{I,J} = \frac{f^{\mathrm{f}}_{i,j} + f^{\mathrm{f}}_{i+1,j} + f^{\mathrm{f}}_{i,j+1} + f^{\mathrm{f}}_{i+1,j+1}}{4} \tag{4.4}$$

と計算する．マルチタイムステップでは，粗い格子が 1 ステップ進む間に細かい格子は 2 ステップの計算を行う．そのため，粗い格子から細かい格子の分布関数 f^{f} を内挿するときには，解像度の境界から 2 格子分の計算点に値を与える．Fig.4.9 の点 (i, j) や $(i, j+1)$ などの境界から 1 点目の細かい格子点の値を補間する場合，粗い格子点（Fig.4.9 の赤）だけでは補間の計算を行えないため，仮想的に定義された粗い計算点（Fig.4.9 のピンク）の値を利用する．補間には線形補間を用い，例えば Fig.4.9 の点 $(i, j+1)$ における分布関数 $f^{\mathrm{f}}_{i,j+1}$ は，点 (I, J) からの距離 α と β を用いて

$$f^{\mathrm{f}}_{i,j+1} = (1-\alpha)(1-\beta) f^{\mathrm{c}}_{I,J} + \alpha(1-\beta) f^{\mathrm{c}}_{I+1,J} + (1-\alpha)\beta f^{\mathrm{c}}_{I,J+1} + \alpha\beta f^{\mathrm{c}}_{I+1,J+1} \tag{4.5}$$

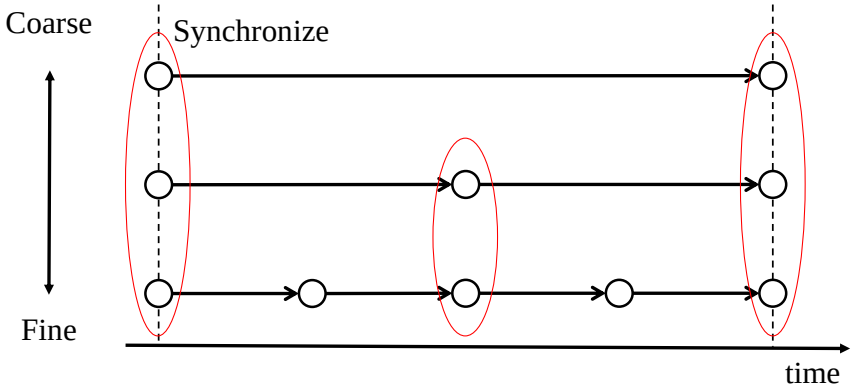


Fig.4.7: Time integration with different time steps for each resolution. The arrow indicates one time step.

と計算される．距離 α と β は粗い格子点間の距離で無次元化されており，$0 \leq \alpha \leq 1$，$0 \leq \beta \leq 1$ の値をとる．

3 次元計算では，線形補間を用いて粗い格子点の分布関数 $f^{\mathrm{c}}_{I,J,K}$ を周囲の細かい格子点から

$$f^{\mathrm{c}}_{I,J,K} = \frac{f^{\mathrm{f}}_{i,j,k} + f^{\mathrm{f}}_{i+1,j,k} + f^{\mathrm{f}}_{i,j+1,k} + f^{\mathrm{f}}_{i+1,j+1,k} + f^{\mathrm{f}}_{i,j,k+1} + f^{\mathrm{f}}_{i+1,j,k+1} + f^{\mathrm{f}}_{i,j+1,k+1} + f^{\mathrm{f}}_{i+1,j+1,k+1}}{8} \tag{4.6}$$

と計算する．細かい格子の速度分布関数 $f^{\mathrm{f}}_{i,j+1,k+1}$ は，点 (I,J,K) からの距離 α,β,γ を用いて

$$\begin{aligned} f^{\mathrm{f}}_{i,j+1,k+1} = {} & (1-\alpha)(1-\beta)(1-\gamma) f^{\mathrm{c}}_{I,J,K} + \alpha(1-\beta)(1-\gamma) f^{\mathrm{c}}_{I+1,J,K} \\ & + (1-\alpha)\beta(1-\gamma) f^{\mathrm{c}}_{I,J+1,K} + \alpha\beta(1-\gamma) f^{\mathrm{c}}_{I+1,J+1,K} \\ & + (1-\alpha)(1-\beta)\gamma f^{\mathrm{c}}_{I,J,K+1} + \alpha(1-\beta)\gamma f^{\mathrm{c}}_{I+1,J,K+1} \\ & + (1-\alpha)\beta\gamma f^{\mathrm{c}}_{I,J+1,K+1} + \alpha\beta\gamma f^{\mathrm{c}}_{I+1,J+1,K+1} \end{aligned} \tag{4.7}$$

と計算する．

密度や流速などの流体のマクロ量は補間で計算された分布関数から求める．フェーズフィールド法の分布関数 h に関しても同様に線形補間で値を求め，フェーズフィールド変数 ϕ は分布関数の総和で求める．

4.1.4　動的な格子細分化

シミュレーションの進行に伴い，各ブロック内の物理特性に基いて細分化や粗大化される．例えば界面に高解像度格子を割り当てる場合，隣接するブロックに界面が含まれる場合に，細分化の処理を行い格子を細かくする．細分化の処理では，必要な解像度を満たしていないリーフ（ブロック）に 4 つ（2 次元）または 8 つ（3 次元）の新しい子ノードを作成し，次の計算ステップから作られた子ノードをリーフ

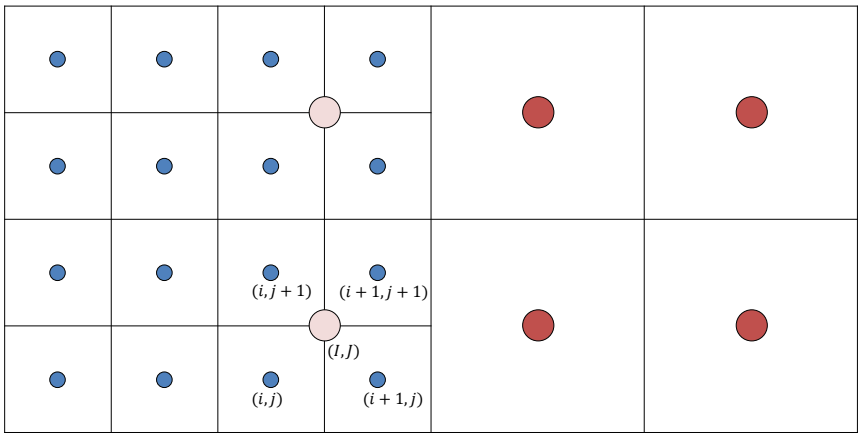


Fig.4.8: Interpolation from higher resolution points (blue) to a lower resolution point (pink) in two dimensional.

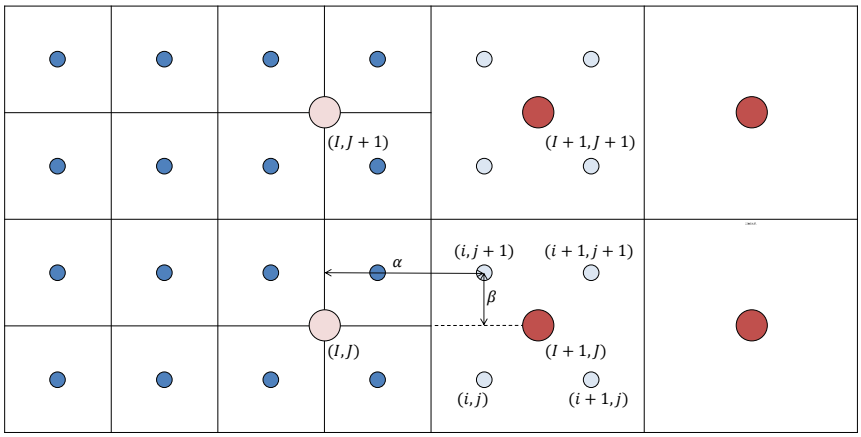


Fig.4.9: Interpolation from lower resolution points (red and pink) to higher resolution points (light blue) in two dimensional.

をして扱う．粗大化の処理では，4 つ（2 次元）または 8 つ（3 次元）のリーフを破棄し，その親ノードがリーフとして設定される．新しいリーフの値は，同期の処理と同様に線形補間を用いて計算する．

4.2 GPU 実装

4.2.1 データ構造

本研究の AMR 法の実装では，細分化を行う木構造の情報，リーフに割り当てられたブロックの情報，ブロックに割り当てられた格子点の情報を扱う必要があり，Fig.4.10 のように 3 つのデータ構造を用い

て管理する．

1 つ目は木構造のための構造体「Node」であり，所属する木構造の番号（idForest），リーフの番号（idLeaf），木構造の深さ（level），子ノードのアドレス（child），隣接するリーフのアドレス（neighbor），格子生成の基準となる物理量（criteria）などを変数に持つ．木構造は以下の手順で作成される．正方形や立方体の計算領域の場合はひとつの根ノード，長方形や直方体の場合は複数の根ノードに対応する Node を確保し，木構造の深さを 0 とする．Fig.4.1 のように計算領域を再帰的に分割しながら，子ノードのメモリを確保していき，子ノードのアドレスを child に記憶する．細分化はノードが対応する領域の物理量（例えば物体からの距離など）に基づいて行う．子ノードの深さは親ノードに 1 を足した値であり，例えば根ノードの子ノードの深さは 1 である．計算領域の細分化後，深さ優先探索を用いて子ノードを持たないリーフノードにリーフ番号を振る．リーフでないノードのリーフ番号を-1 とする．その後，各リーフノードは隣接している領域に対応するリーフノードを探索し，隣接のリーフノードのアドレスを neighbor に記憶する．Node は構造体の中に子ノードの情報を持つ再帰的なデータ構造であるため，逐次処理が必要であり，並列処理を行う GPU で扱いにくい．そのため，CPU のみが木構造を持つ．

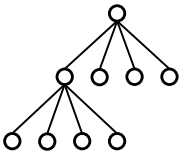
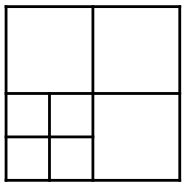
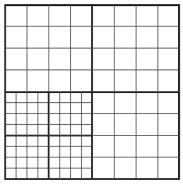
2 つ目はリーフノード（ブロック）のみの情報をまとめた配列「BlockInfo」である．格子ボルツマン法の計算に必要なブロックの情報を，木構造の深さ（格子解像度に対応）ごとにリーフの番号が小さい順に並べて配列に記憶させる．ブロックの情報として，所属する木構造の番号（idForest），データのオフセット（offset），木構造の深さ（level），子ノードのオフセット（child），隣接するリーフのオフセット（neighbor），格子生成の基準となる物理量（criteria）などを変数に持つ．木構造では子ノードや隣接リーフのアドレスを保持していたが，ブロック情報では子ノードや隣接ブロックのデータが配列の何番目か（オフセット）を記憶する．木構造に基づいて CPU がブロックの情報 blockInfo を作成し，それを GPU 側に転送する．

3 つ目が格子ボルツマン法の計算に用いる分布関数や速度などの従属変数である．マルチタイムステップの計算では各解像度のステップ数が異なるため，異なる解像度の従属変数を同じ配列に保存すると，アップデート前の値（f）とアップデート後の値（fn）が混在してしまう．そこで，Fig.4.11 のように解像度ごとに配列を分けることで，アップデート前（f）と後（fn）の変数が同じ配列内で混在しないようにする．GPU でコアレスアクセスとするため，ブロック内のデータは連続アドレスとなるように並べる．格子ボルツマン法の計算は全て GPU 上で実行するため，従属変数は基本的に GPU 上にのみ保存する．CPU 側の従属変数のメモリは主にデータ出力に用いる．

4.2.2　隣接ブロックへのステンシルアクセス

ブロックの内側にある計算点では同一のブロック内のデータしか計算に用いないが，ブロックの表面にある計算点はステンシル計算で隣接するブロックの計算点にアクセスする．そのため，ブロック間のデータ交換が必要であり，本節ではその実装方法について述べる．ブロック間のデータ交換の実装として 3 つの方法が考えられる．1 つ目は，ステンシル計算を行う関数内で隣接ブロックの格子点を直接アクセスする方法である．データ交換用の格子点を割り当てる必要がなく，メモリ使用量を少なくできる．しかし，隣接ブロックの解像度が異なる場合，同じレベルの隣接格子点が存在しないため，その値を補

	Tree	Block information	LBM variables

	
	Node{
	int idLeaf;
	int idForest
	int level;
	Node* child[ ];
	Node* neighbor[ ];
	real* criteria;
	};
		
		BlockInfo{
		int offset;
		int idForest;
		int level;
		int child[ ];
		int neighbor[ ];
		real* criteria;
		};
			
			LBM{
			float* f, fn;
			float* h, hn;
			float* u, v, w;
			float* rho, phi;
			float* levelset;
			};

CPU	✔	✔	used for output only
GPU		✔	✔

Fig.4.10: Data structures used in the AMR code.

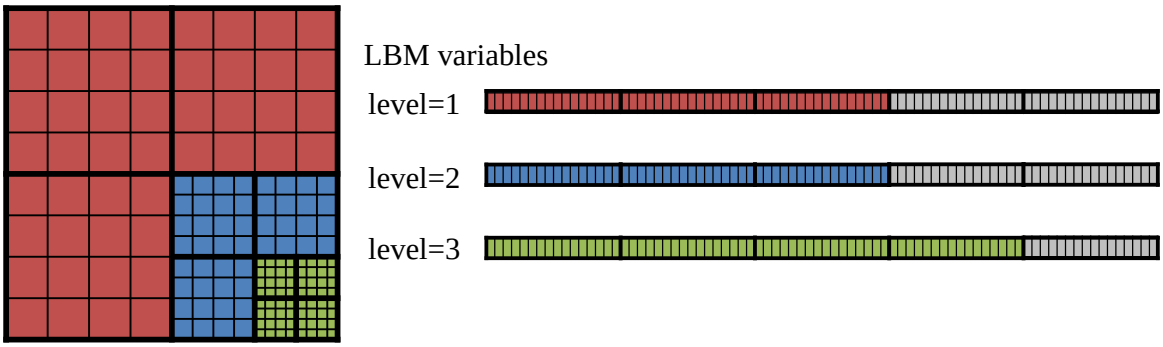


Fig.4.11: Memory layout on a GPU device memory. The color indicates the mesh resolution. Gray cells are unused.

間で求める必要がある．CUDA では 32 個のスレッドがワープという単位で完全に同期して処理を行うため，ブロックの内側と表面で異なる処理をする際，どちらかの処理が完了するまでワープ内の他のスレッドが待機状態となり，GPU 計算の性能が著しく低下する．そのため，隣接ブロックを直接参照する実装では，ブロックの内側と外側のカーネルを分けるなどのチューニングが必要である [107].

2 つ目は GAMER[24] などの AMR 法のフレームワークで採用されている方法であり，Fig.4.12 のように各ブロックの周囲に隣接ブロックとのデータ交換用の領域（ハロー領域）を確保する方法である．ステンシル計算を行う前に，同一レベルの場合は隣接ブロックの値をコピーし，レベル差がある場合は値を補間してハロー領域に記憶する．ステンシル計算では，ブロック表面の計算点はハロー領域に記憶さ

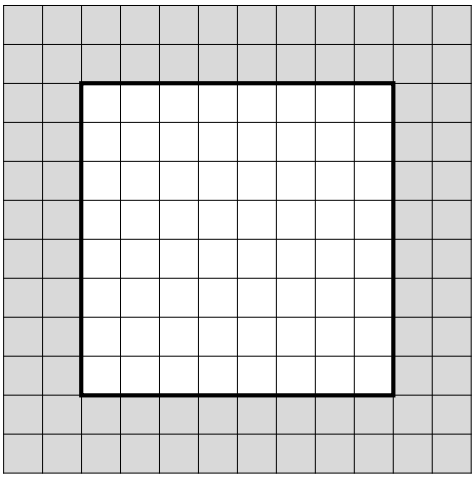


Fig.4.12: A block including halo region (gray) for data exchanges between neighboring blocks.

れた値を参照するため，ブロックの内側と表面で同一の処理となる．GPU で高い実行性能を出しやすいが，メモリ使用量が増加してしまう．特に，ひとつのブロックあたりの格子点数が少ない場合は，計算点よりもハロー領域のメモリ量のほうが多くなってしまう．例えば，ひとつのブロックあたり $4 \times 4 \times 4$ 格子で，ハロー領域として周囲に 2 格子づつ配置した場合，ブロックあたりの格子点は $8 \times 8 \times 8$ となり，メモリ使用量は 8 倍になってしまう．ブロックあたりの格子点数を増やすことでメモリ使用量の増加を抑えることができるが，Fig.4.5 に示した通り，ブロック内の格子点数が多いと格子点数が増えてしまう．GPU のメモリ容量は少ないため，可能な限りメモリ使用量は削減する必要がある．

3 つ目は Fig.4.13 のように解像度差の境界近傍に補間用のブロック（ゴーストブロック）を配置する方法である．ステンシル計算を行う前にゴーストブロックの値を補間で求めておく．隣接ブロックの解像度が異なる場合は，ブロック表面の計算点は隣接のゴーストブロックの値を参照する．隣接ブロックが同一解像度の場合は，隣接ブロックのメモリを直接参照する．直接アクセスする場合よりもメモリ使用量は増加してしまうが，各ブロックがハロー領域を持つ場合と異なり，ブロックあたりの格子点が少ないほどメモリ使用量の増加を抑えられる．本研究では，解像度の境界近傍にゴーストブロックを配置してステンシルアクセスの GPU 実装を行う．

4.2.3　スレッドの割り当て

格子ボルツマン法の計算はマルチタイムステップであるため，タイムステップごとに計算する解像度が異なる．また，物体の境界条件などの処理は物体近傍に割り当てられた高解像度格子でしか行う必要がなく，解像度ごとに処理が異なる．そのため，本研究では，解像度ごとにカーネル関数を分離して実行する．ひとつのスレッドが 1 格子点を計算するスレッド並列化を行う．レベル l のブロック数を N_l^{block}，

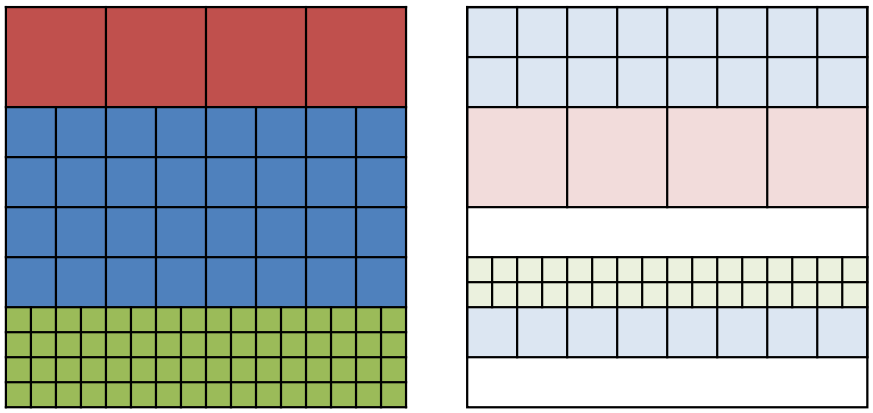


Fig.4.13: The arrangement of ghost blocks for interpolation at the boundaries of resolution difference. The blocks on the right mesh are ghost blocks corresponding to the left mesh.

ブロック内の計算点数を N^{patch} とし，レベル l の格子を計算するときの CUDA ブロックの数を

$$\begin{aligned} \texttt{gridDim.x} &= \texttt{floor}\left(\frac{N^{\mathrm{patch}} + \texttt{blockDim.x} - 1}{\texttt{blockDim.x}}\right) \\ \texttt{gridDim.y} &= \texttt{min}\left(N_l^{\mathrm{block}}, \mathrm{GRIDMAX}\right) \\ \texttt{gridDim.z} &= \texttt{floor}\left(\frac{N_l^{\mathrm{block}} + \mathrm{GRIDMAX} - 1}{\mathrm{GRIDMAX}}\right) \end{aligned} \tag{4.8}$$

と設定する．ここで，`floor` は小数点以下を切り捨てる処理，`min` は最小値を取る処理である．GRIDMAX は `gridDim.y` と `gridDim.z` の最大値であり，CUDA7.5 では 65535 である．`gridDim.x` はひとつのブロックに割り当てる CUDA ブロックの数を指定する．`gridDim.y` はブロックの数を指定するが，CUDA7.5 では 65535 以上の値を指定できないため，ブロック数が 65535 以上の場合は `gridDim.z` も利用してブロック数を指定する．

各スレッドが担当するブロック `blockId` とブロック内の格子点のインデックス `cellId` は

$$\begin{aligned} \texttt{blockId} &= \texttt{blockIdx.y} + \texttt{blockIdx.z} \times \mathrm{GRIDMAX} \\ \texttt{cellId} &= \texttt{blockIdx.x} \times \texttt{blockDim.x} + \texttt{threadIdx.x} \end{aligned} \tag{4.9}$$

と計算する．

4.2.4 動的な格子細分化と粗大化

物体や界面の運動に伴い，格子の動的な細分化と粗大化を行う．格子ボルツマン法は完全な陽解法であるため時間刻み Δt を小さくする必要がある．そのため，界面や物体が 1 格子分移動するのに数十ステップかかり，例えばクーラン数

$$\mathrm{CFL} = \frac{U\Delta t}{\Delta x} \tag{4.10}$$

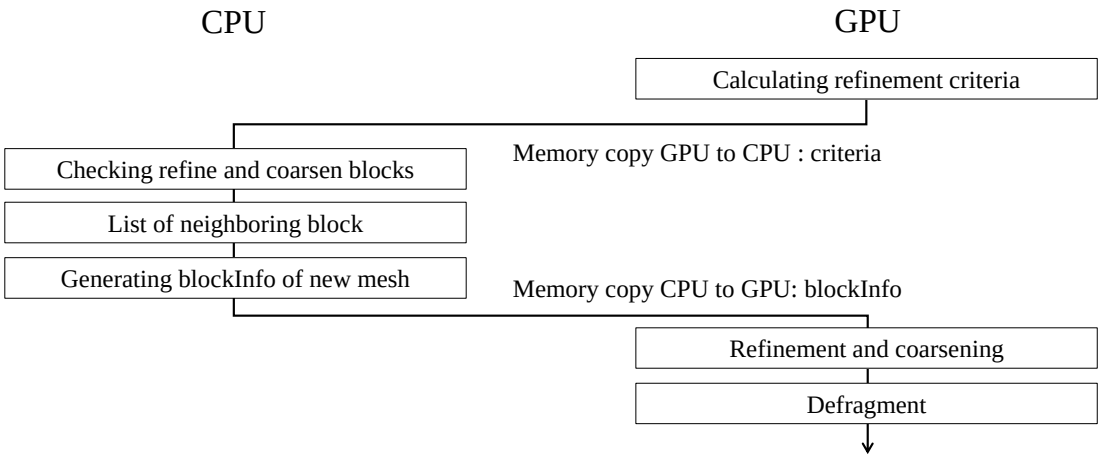


Fig.4.14: Flowchart of block refinement and coarsening.

を 0.03 とした場合は，界面などのプロファイルが 1 格子移動するのに約 33 ステップが必要になる．そのため，毎タイムステップの格子生成は必要ではなく，格子生成の頻度を下げることが可能である．代表速度 U で移動する仮想的なプロファイルが，ブロックの半分の距離を移動するステップ数に一度の頻度で格子生成を行う．

格子生成を高速に実行するため，CPU と GPU 間の通信を最小限に抑える必要がある．格子の細分化と粗大化の処理の流れを Fig.4.14 に示す．まず，各ブロックの格子の細分化と粗大化の基準となる物理量を GPU 側で計算し，各ブロックの計算された閾値を CPU に転送する．各ブロックの閾値に従い粗大化されるブロックと細分化されるブロックを判定し，木構造を修正する．修正された新しい木構造から新しいブロック情報を作成し，GPU 側に通信する．新しいブロック情報を元に GPU 側で細分化と粗大化の処理を実施し，新しい格子点の値を補間する．このとき，プロファイルの変化はブロックの半分以下であるため，解像度差の境界に接しているブロックが細分化や粗大化される．本研究では，解像度差の境界近傍のブロックには補間のためのゴーストブロックがオーバーラップして配置してあり，それを用いた GPU メモリの管理を提案する．解像度差の境界近傍のブロックが全て細分化する処理の例を Fig.4.15 に示す．境界近傍の粗い計算ブロックは細分化され，もともと利用されていたメモリは新しいゴーストブロック用に利用される．不要なゴーストブロックは削除され，ゴーストブロックに利用されていたメモリは空になる．細かいゴーストブロックに使用していた水色の GPU メモリを新しい計算ブロック用のメモリとし，新しく配置されるゴーストブロックのメモリは配列の後ろに追加される．粗大化の処理でも同様にし，Fig.4.16 のように，粗大化されるブロックとオーバーラップして配置されているゴーストブロックを計算ブロックに変更する．粗大化された細かい格子はゴーストブロックに変更される．

GPU でのメモリの確保と開放は CPU の場合のように高速に実行できないため，毎回メモリの確保と開放を行うのではなく，格子点数よりも多くのメモリ・プールを用意して細分化と粗大化を管理し，メモリの動的確保と開放の頻度を削減する．繰り返される細分化と粗大化によりメモリ・プールが断片化するため，GPU メモリのデフラグメンテーションを行う．

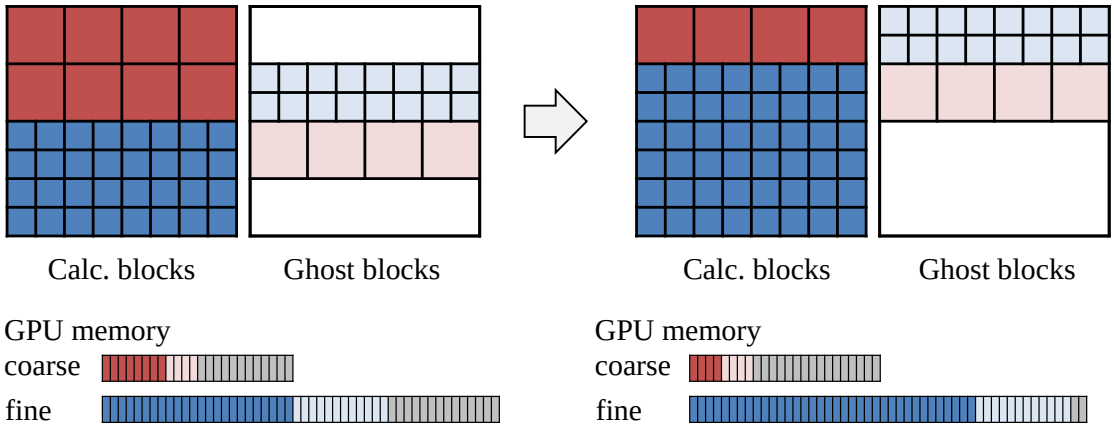


Fig.4.15: Block refinement using ghost blocks.

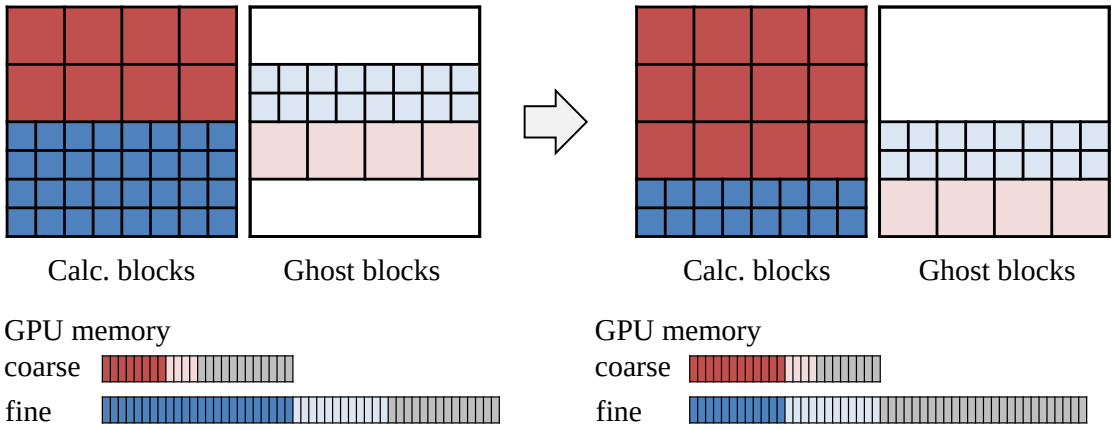


Fig.4.16: Block coarsening using ghost blocks.

4.3 MPI 実装

複数 GPU を用いた大規模な AMR 計算を実現するために，MPI ライブラリを用いた並列化を行う．1 つの MPI プロセスに 1 台の GPU を割り当てる Flat-MPI で並列化を行う．本節では，AMR 法の MPI 実装について述べる．

4.3.1 空間充填曲線による領域分割

均一格子の場合と同様に，計算領域をプロセス数の小領域に分割し，各小領域の計算を各プロセスに割り当てることで AMR 法の MPI 並列計算を行う．Fig.4.17 のように，均一格子の並列計算と同様に計算領域を同じ大きさの立方体や直方体の小領域に分割しただけでは，各小領域のブロック数が異なり，各プロセスの計算コストが異なってしまう．ブロック数が最も多い領域を担当している GPU の計算が終

わるまで他の GPU は待機するため，GPU の使用率が低下してしまう．高効率に複数 GPU 計算を行うには，各 GPU の計算コストを均一化する必要があり，そのためには，各小領域のブロック数を等しくする必要がある．

AMR 法の領域分割の方法として，空間充填曲線 [38] を用いた領域分割 [108][109] とグラフ理論に基づいた領域分割 [39] が提案されている．グラフ理論に基づく領域分割法は，METIS[49] などのフレームワークが有名であり，AMR 格子の分割だけでなく，FEM で用いられる非構造格子の分割や粒子計算の粒子割り当てにも利用されている．グラフ分割により計算コストと通信コストを最適化した領域分割が行えるが，分割する頂点と辺の数が増加すると，グラフ分割は時間がかかる．空力解析などで用いられる静的な細分化格子は最初に一度分割するだけなので，グラフ分割が有効である．しかし，AMR のように，計算の進行に伴い格子が動的に変化する場合は，何回もグラフ分割をする必要があり，グラフ分割がオーバーヘッドになってしまう．

空間充填曲線には Fig.4.18 に示すモートン曲線やヒルベルト曲線がある．モートン曲線は深さ優先探索に従ってリーフを一次元マッピングする．小領域内のリーフ数が均一となるようにモートン曲線を切断することで，計算コストが均一な領域分割が可能である．モートン曲線の考え方は非常に単純であり，かつ高速でロバストであるため AMR の並列計算でよく利用されている．AMR のフレームワークである p4est[23] や DAINO[50] でもモートン曲線による負荷分散が採用されている．しかし，モートン曲線による領域分割は，計算コストの負荷にのみ着目しているため，小領域の計算コストの最適化はされていない．メッシュ形状によっては，一つのプロセスが担当する小領域が 2 つの小領域に分断され，小領域の表面積が増加する場合がある [110]．これを解消する方法として，ヒルベルト曲線を用いた領域分割がある [111]．ヒルベルト曲線を用いることで，ひとつのプロセスが担当する小領域が 2 つ以上に別れることはなくなる．しかし，小領域の形状が凹形状になってしまうため，小領域の表面積が増加し，通信コストが大きくなってしまう．そのため，本研究では，空間充填曲線であるモートン曲線による領域分割を行う．

ひとつの木構造の領域が正方形や立方体であるため，モートン曲線を引ける領域もまた正方形や立方体である．流体解析でよく用いられる長方形や直方体の領域にモートン曲線を描くことはできない．本研究では，直方体の領域を細分化するために複数の木構造を用いているため，Fig.4.19 のように，各木構造が担当する正方形または立方体領域にそれぞれモートン曲線を描く．各モートン曲線の終点と始点を x 軸，y 軸，z 軸の順序でつなぐことで，長方形またや直方体の領域に対してモートン曲線に基づく一本の曲線を引く事ができる．本研究では，この曲線に基づきブロックを番号付けし，その順序に従い各小領域のブロック数が均一になるように領域を分割する．

格子ボルツマン法の計算はマルチタイムステップであるため，粗い格子に対して細かい格子のステップ数は 2 倍必要であり，計算コストも 2 倍になる．そのため，各小領域のブロックの数を均一にするだけでは，計算コストを均一化することはできない．解像度に従ってブロックに重みをつけることで計算コストの均一化を行えるが，その場合，各 GPU のメモリ使用量が均一にならない．GPU のメモリ容量は CPU に比べて少ないため，大規模計算を行うためには各 GPU のメモリ使用量を均一化することも重要である．特に，AMR 法では計算の進行に伴い格子点数が変化し，それに伴い必要なメモリ量も増減す

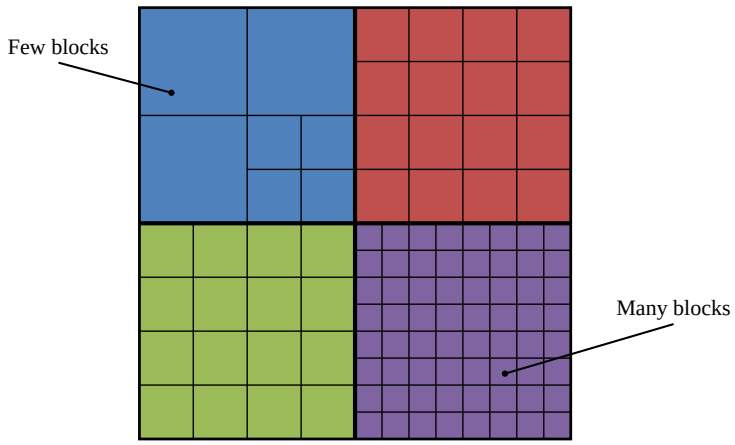


Fig.4.17: Domain decomposition of refined mesh using rectangular sub-domains. The mesh indicates blocks. The colors show sub-domains.

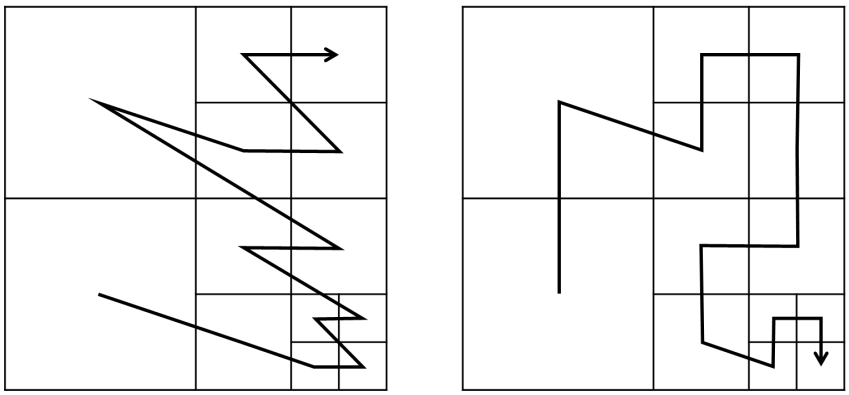


Fig.4.18: Two common space filling curves: Morton curve (left) and Hilbert curve (right).

る．プロファイルの変化により初期状態よりも格子点数が増える場合が多く，メモリ使用量を均一化しないと GPU のメモリ容量が足らずに計算が破綻する場合がある．そのため，本研究では，各小領域のブロック数を等しくし，GPU のメモリ使用量を均一化を優先する．

4.3.2 袖領域の GPU 間通信

格子ボルツマン法の計算では，隣接の計算点へのメモリアクセスがあるため，小領域表面の速度分布関数を GPU 間で通信する必要がある．隣接の小領域を担当している GPU のメモリに直接参照することはできないので，分散メモリ型の並列計算のインターフェースである MPI を用いたデータ転送を行う．

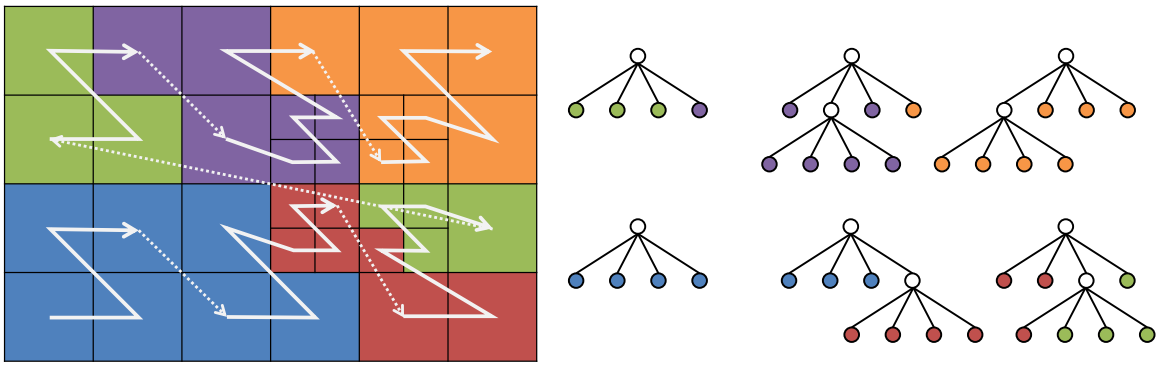


Fig.4.19: Domain decomposition of a rectangular computational domain using the Morton curve and corresponding tree structures. The solid arrows are the Morton curves of each tree. The dot arrows indicate the connections between the Morton curves.

Fig.4.20 のように各 GPU が担当する小領域のまわりに，隣接する小領域とのデータ交換用のハローブロックを配置する．小領域表面の計算点の全方向の速度分布関数は必要でないため [15]，Fig.4.21 のように並進過程で隣接小領域に移動する分布関数のみを通信する．これにより，通信量を 1/3 以下に削減できる．GPU 間の通信は CPU を介して実行する．TSUBAME3.0 の環境では，同一のノード内の GPU 間では NVLINK によって直接通信が可能であり，異なるノードにある GPU 間の通信は GPU Direct RDMA で行えるが，本コードでは CPU を介した通信の方が通信時間が短かったため，GPU 間の直接通信を行う GPU Direct を用いていない．

複数 GPU 計算の単純な実装では，Fig.4.22 の左図のように，袖領域の通信が完了したのち各ブロックの計算を実行する．しかし，GPU スパコンではノードあたりのパフォーマンスが高く計算を高速に実行できるため，ノード間の通信がオーバーヘッドになる場合が多い．そのため，複数 GPU 計算では，格子点の計算と袖領域の通信をオーバーラップさせることで，通信のオーバーヘッドを隠蔽する方法が提案されている [104][104]．本研究では，AMR 法の計算に演算と通信のオーバーラップを実装する．Fig.4.22 の右図に示すように，各小領域のブロックを隣接小領域のデータが必要な小領域表面のブロック（青色）と通信が必要でない内側のブロック（水色）に分ける．CPU が小領域間のデータ通信を行うのと同時に，通信が必要でない内側のブロックを GPU で計算する．通信が終わり次第，小領域表面のブロックの計算を GPU で行うことで，通信時間の隠蔽し実行性能を向上させる．

4.3.3　動的な格子生成

本研究の複数 GPU 実装では，各 MPI プロセスが木構造全体を保持している．木構造の処理は再帰的な処理を含み，並列化が困難であるため，木構造の処理に関しては MPI 並列化を行わない．複数 GPU 計算の場合の格子細分化と粗大化の処理の流れを Fig.4.23 に示す．複数 GPU で処理を行うために，Fig.4.14 の単一 GPU 実装の処理に追加したサブルーチンを赤色で示している．各 GPU は自身が担当しているブロックの細分化と粗大化の基準となる閾値を計算し，CPU に細分化の閾値を転送する．担当

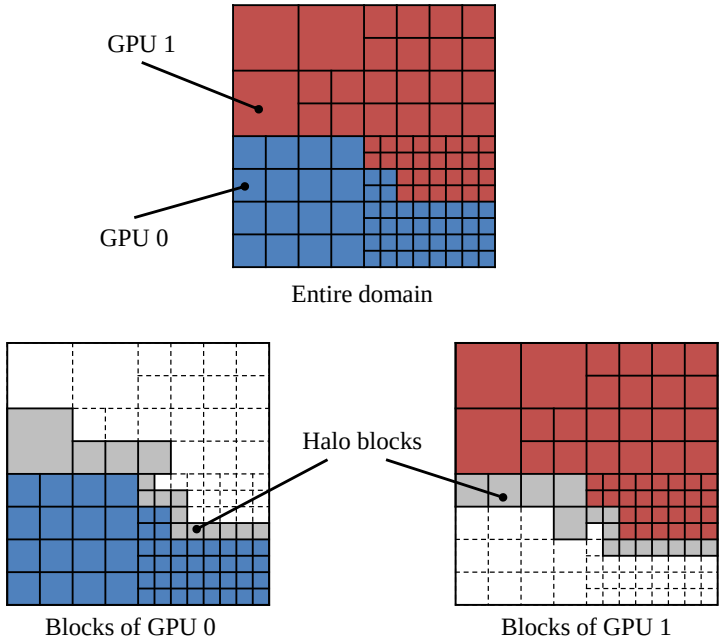


Fig.4.20: Halo blocks to store the data exchanged between neighboring sub-domains.

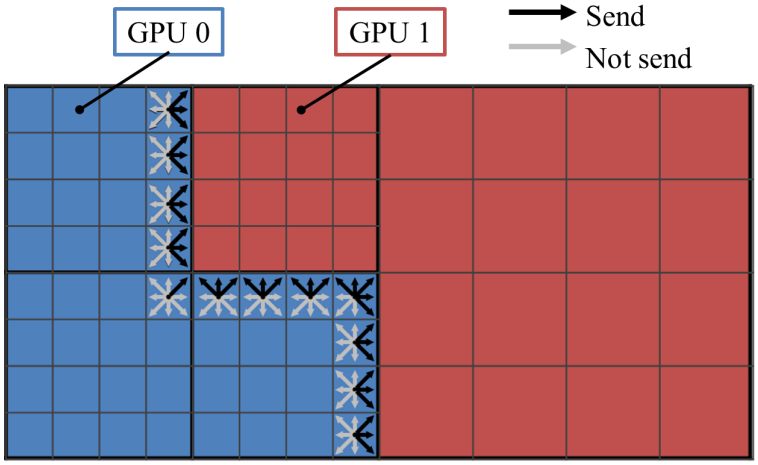


Fig.4.21: Particle distribution functions sent to neighboring sub-domains.

しているブロック以外の閾値は 0 としておく．MPI の集団通信を行う関数である `MPI_Allreduce` を用いて，全 MPI プロセスでその閾値を共有する．各 MPI プロセスでブロックの細分化と粗大化の処理を行い，新しい木構造を作成する．このとき，自身の小領域内のブロックに関しては，ブロックへの計算セルの割り当てや削除を行う．その後，新しい木構造に対して負荷分散のための領域分割を行う．領域分割の変更より，担当していたブロックが他の MPI プロセスの担当が変わり，通信が必要となる．古い分割領域と新しい分割領域を比較し，CPU 側で領域分割の変更に伴うデータ転送が必要なブロックのリス

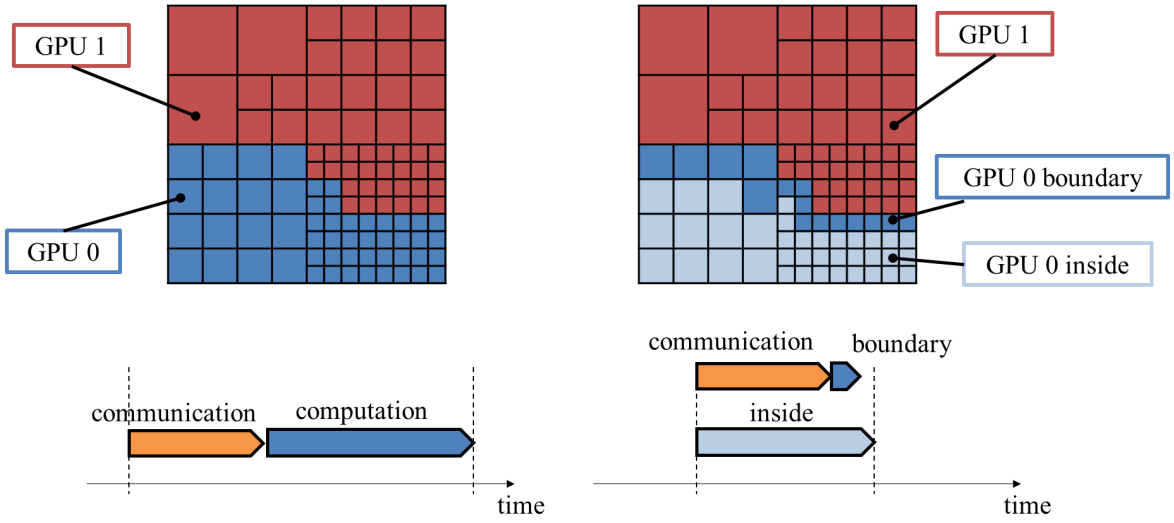


Fig.4.22: The non-overlapping communication method (left) and the overlapping communication method (right) for halo data exchange.

トを作成する．他のプロセスに通信するブロックを GPU 側で一次元配列にパッキングし，その配列を CPU に転送する．CPU 側でパッキングされたブロックのデータを MPI 通信し，他のプロセスから受け取ったブロックの情報を GPU に転送する．格子の細分化と粗大化，および領域分割に伴うブロックの移行によりメモリが断片化するため，GPU メモリのデフラグを行う．領域分割が変更されたため，GPU 間の袖領域通信のリストを作成する．

4.3.4　AMR 法の複数 GPU 計算の実行性能測定

東京工業大学学術国際情報センターの GPU スパコン・TSUBAME3.0 を用い，AMR 法を導入した格子ボルツマン法の複数 GPU 計算の実行性能を評価する．1 つの MPI プロセスに 1 台の GPU を割り当てる Flat-MPI で実装し，1 つのノードに 4MPI プロセスを割り当てる．実行性能の評価には 1 秒間に更新可能な格子点数である MLUPS（Mega Lattice Updata Per Second）を用いる．

固定した格子でのスケーリング測定

動的な格子生成を行わない計算で実行性能を測定し，袖領域通信のオーバーヘッドと通信と計算のオーバーラップの効果を評価する．単相のキュムラント型の格子ボルツマン法を用いた球周り流れの計算により，8GPU から 128GPU までの弱スケーリングを評価する．8GPU での計算を基準とし，16GPU の場合は 8GPU で計算した領域を z 軸方向に 2 つ並べた計算を行う．このようにして，計算コストと通信コストを一定にして並列数を増加させる．弱スケーリングの測定の条件を Table4.1 に示す．1 ブロック

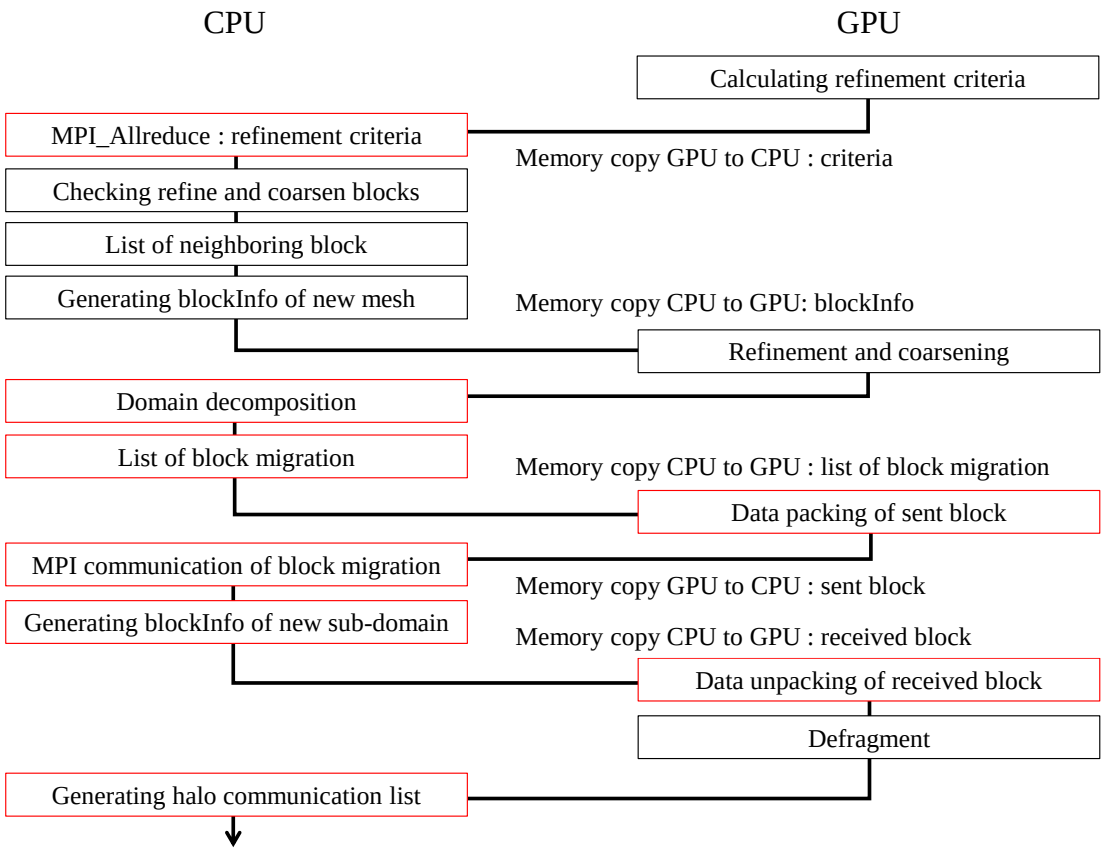


Fig.4.23: Flowchart of dynamic mesh adaptation of multiple GPU implementations. In multiple GPU implementations, red subroutines are added to the single GPU implementation shown in Fig.4.14.

あたりの格子点数を $8 \times 8 \times 8$ とする．1 台の GPU 当たり 28,864,768 格子を割り当て，512 ステップの計算を実行する．最も粗い格子の解像度は最も細かい格子の 128 倍である．弱スケーリングの結果を Fig.4.24 に示す．青いプロットは計算と通信のオーバーラップなしの結果，赤色がオーバーラップありの結果である．GPU を増やすと理想的な性能向上を示す点線と同じ傾きで性能が向上しており，どちらも良好なスケーリングである．128GPU の条件において，1GPU 当たりの実行性能はオーバーラップなしで 397 MLUPS，オーバーラップありで 546 MLUPS であり，オーバーラップにより通信時間を隠蔽することで実行性能を 1.37 倍に向上することができた．また，均一格子を用いた複数 GPU 計算の実行性能と比較すると，オーバーラップ法を用いた場合は約 45% の性能であり，細分化格子を用いた格子ボルツマン法の計算を十分に高速化できているといえる．並列化効率を Table4.2 に示す．128GPU の場合の並列化効率はオーバーラップなしでは 98.2%，オーバーラップありでは 98.7% であり十分な並列化効率を達成できた．

次に，計算サイズを固定し並列数を増加させて実行性能を評価する強スケーリングを測定する．8GPU から 64GPU までの実行性能を測定する．総格子点数を 230,918,144 と設定し，512 ステップの計算を行

Table4.1: Conditions for weak scaling study of LBM computation on locally refined meshes.

Number of GPUs	Number of blocks	Number of lattice points
8	451,012	230,918,144
16	902,024	461,836,288
32	1,804,048	923,672,576
64	3,608,096	1,847,345,152
128	7,216,192	3,694,690,304

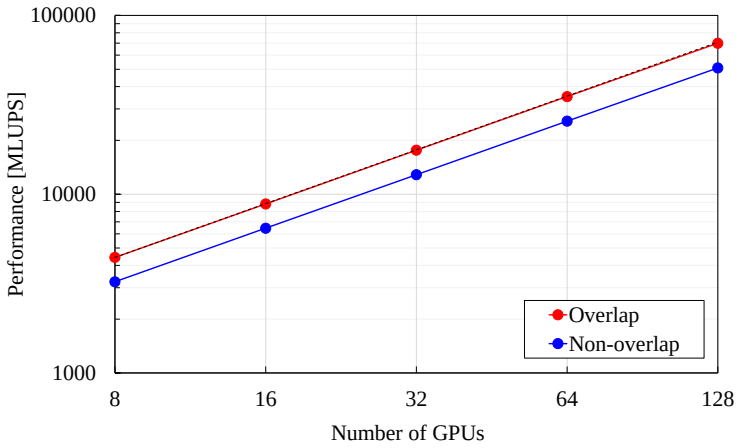


Fig.4.24: Weak scaling results of LBM computation on locally refined meshes.

Table4.2: Parallel efficiency of weak scaling of LBM computation on locally refined meshes.

Number of GPUs	Parallele efficiency [%] non-overlap	Parallele efficiency [%] overlap
8	—	—
16	99.66	99.53
32	99.22	99.55
64	98.93	99.28
128	98.19	98.66

う．強スケーリングの結果を Fig.4.25 に示す．青色のプロットは計算と通信のオーバーラップなしの結果，赤色がオーバーラップありの結果である．青色の点線は計算時間と通信時間の内訳を測定するために頻繁に MPI の同期を行った場合の結果である．通信と計算をオーバーラップすることでオーバーラップなしよりも高い実行性能を示しているが，理想的な性能向上を示す点線よりも性能が下がってしまっている．並列化効率を Table4.3 に示す．オーバラップを行った場合，GPU 数を 2 倍にした 16GPU で

Table4.3: Parallel efficiency of strong scaling of LBM computation on locally refined meshes.

Number of GPUs	Parallele efficiency [%] non-overlap	Parallele efficiency [%] overlap
8	—	—
16	71.50	81.39
32	46.84	49.59
64	30.74	27.70

の並列化効率は 81.39% であり，8GPU 数の計算に対して 1.63 倍の高速化ができている．並列効率は GPU 数の増加に伴い低下していく傾向が確認できる．

512 ステップの計算時間の内訳を Fig.4.26 に示す．GPU 数を 2 倍にしても青色の計算時間を半分にすることはできていない．この理由として，本研究では，メモリ使用量のバランスを優先し，各 GPU が担当するブロック数を等しくしているため，マルチタイムステップの格子ボルツマン法の計算では負荷バランスが乱れたためだと考えられる．そこで，GPU 番号を i とし，各 GPU の計算コスト C_i を

$$C_i = \sum_{l=l_{\min}}^{l_{\max}} 2^{l-l_{\min}} N_{i,l}^{\mathrm{block}} \tag{4.11}$$

と見積もる．ここで，l は木構造のレベル，$l_{\min}$ は最も粗い格子のレベル，$l_{\max}$ は最も細かい格子のレベル，$N_{i,l}^{\mathrm{block}}$ は小領域 i に含まれるレベル l のブロックの数である．最も計算コストが高い GPU の計算コストを $C_{\max}$，各 GPU の計算コストの平均値を C_{ave} とし，負荷分散の誤差 Er を

$$Er = \frac{C_{\max} - C_{\mathrm{ave}}}{C_{\mathrm{ave}}} \tag{4.12}$$

と定義し，強スケーリングにおける負荷分散の誤差を Table4.4 に示す．GPU 数の増加に伴い，計算負荷のバランスの誤差が少しづつ大きくなる傾向が確認できる．また，並列数の増加に伴い，全体の計算時間に対して赤色の通信時間の割合が増え，32GPU の条件で計算時間よりも通信時間のほうが長くなっている．16GPU では計算時間のほうが通信時間よりも長く，理想的にはオーバラップ法により通信時間を完全に隠蔽できるはずであるが，オーバーラップ法を行っても並列化効率は 81.39% であった．この原因として，格子ボルツマン法の計算と MPI 通信のサブルーチンを解像度ごとに分けて実行している．そのため，通信を隠蔽できる解像度と隠蔽できない解像度があり，通信を完全には隠蔽できず並列化効率が下がったと考えられる．強スケーリングを向上するためには，通信コストを削減する必要があることが明らかになった．今後の課題として，GPU 間の直接通信を行う NVLink や，ホストメモリを介すること無く GPU 間通信を行う GPU Direct RDMA などを利用することで通信時間の削減を行う予定である．また，袖領域を広く取ることで総通信回数を減らし，通信レイテンシのコストを削減できるテンポラルブロッキング法 [112] の導入を計画している．

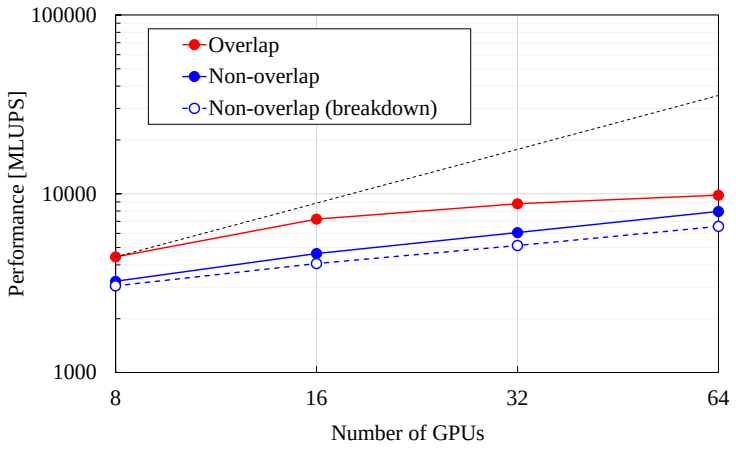


Fig.4.25: Strong scaling results of LBM computation on locally refined meshes.

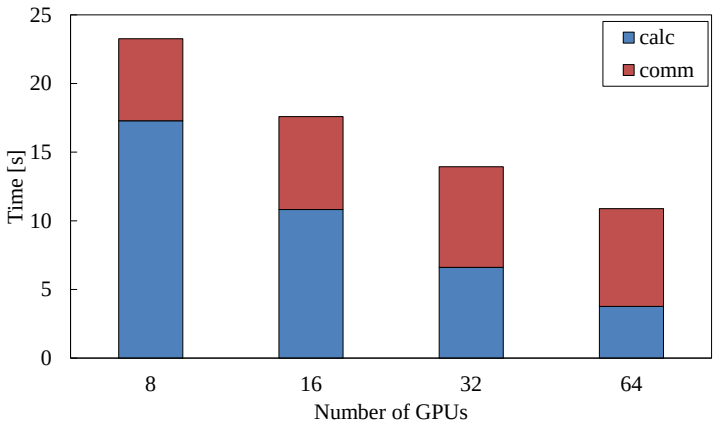


Fig.4.26: Breakdown of strong scaling results of LBM computation on locally refined meshes.

動的な格子生成を含む計算でのスケーリング測定

動的な格子細分化と粗大化を行う計算を GPU 数を変えて実行し，格子生成のオーバーヘッドと並列数の関係を調べる．計算性能の測定には，格子ボルツマン法とフェーズフィールド法による界面捕獲手法を組み合わせた自由界面流れの計算で行う．2 章の検証計算で実施した濡れた床へのダム崩壊計算を 256 タイムステップ行い，格子生成は 128 ステップに一度行う．並列数の増加に合わせて計算領域を奥行方向に広げていき，各 GPU の格子点数を一定にしつつ並列数を増加させる弱スケーリングを評価する．弱スケーリングの条件を Table4.5 に示す．1 ブロックあたりの格子点数を $8 \times 8 \times 8$ とする．1 台の GPU あたり 15,037 ブロック，7,698,944 格子点を割り当て，16GPU から 256GPU までの測定を行った．

弱スケーリングの測定結果を Fig.4.27 に示す．固定格子の場合はほぼ理想的な弱スケーリングが得ら

Table4.4: Load balance error of strong scaling of LBM computation on locally refined meshes.

Number of GPUs	Load balance error Er
8	0.363
16	0.389
32	0.561
64	0.602

Table4.5: Conditions for weak scaling study of LBM computation on adaptively refined meshes.

Number of GPUs	Number of blocks	Number of lattice points
16	240,592	123,183,104
32	481,184	246,366,208
64	962,368	492,732,416
128	1,924,736	985,464,832
256	3,849,472	1,970,929,664

れたが，動的な格子生成を行った場合は，理想的な実行性能よりも低い性能となっている．並列化効率を Table4.6 に示す．32GPU での並列化効率は 93.14% であり，16GPU に対してほぼ 2 倍の性能向上である．しかし，並列数の増加に伴い並列化効率は低下し，256GPU での並列化効率は 47.87% である．並列化効率の原因を調べるため，自由界面流れの計算と動的な格子生成にかかる時間を測定した．弱スケーリングの測定における 256 タイムステップの実行時間の内訳を Fig.4.28 に示す．赤色が自由界面流れの計算時間であり，GPU 間の袖領域の通信時間も含まれる．緑色が動的な格子細分化と粗大化にかかる時間であり，Fig.4.23 に含まれるサブルーチンの実行時間の和である．動的な格子生成は 128 ステップに一度の頻度で実行し，Fig.4.28 の時間は，2 回の格子生成にかかった時間である．格子生成にかかる時間は並列数の増加に伴い長くなっていることが確認できる．16GPU の場合では格子生成にかかる時間は全計算時間の 15% ほどであるが，GPU 数の増加に伴い格子生成の実行時間の割合が増え，256GPU では全体の約 49% が格子生成の時間に費やされている．ブレークダウンの結果より，並列数の増加に伴う格子生成の時間の増加が弱スケーリングを低下させている原因であることが確認できた．木構造に基づく格子細分化の処理には，根ノードから再帰的に行う処理が含まれているため並列化が困難であり，本研究では，木構造の処理は並列化をせずに各 MPI プロセスが冗長に行っている．そのため，並列数の増加に伴い木構造が大きくなる弱スケーリングでは，木構造の処理時間が増加したのだと考えられる．このような，並列数の増加に伴う格子細分化のオーバヘッドの増加は，他の研究でも示されている [113]．今後の課題として，大規模 AMR 計算を高効率に実行するためには，木構造の処理や負荷分散に関しても MPI による並列化が必要であることがわかった．木構造の処理の並列化方法として，木構造を分割し，各プロセスは自身が担当しているブロック，および小領域に隣接しているブロックが所属する分割された木構造の処理を行う方法が考えられる．

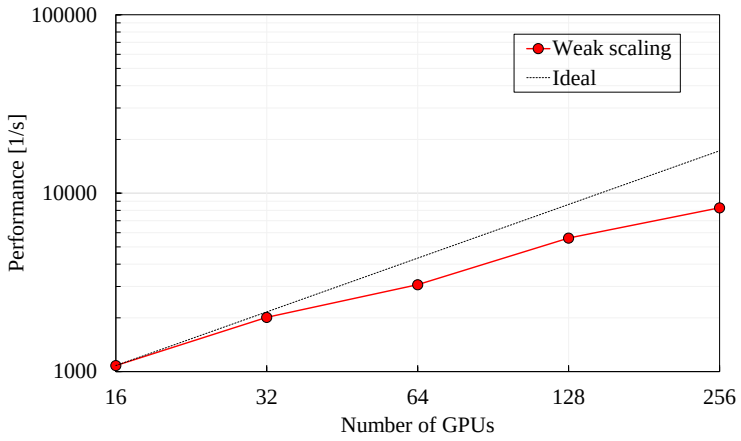


Fig.4.27: Weak scaling results of LBM computation on adaptively refined meshes.

Table4.6: Parallel efficiency of weak scaling of LBM computation on adaptively refined meshes.

Number of GPUs	Parallele efficiency [%]
16	—
32	93.14
64	71.03
128	64.78
256	47.87

4.4　AMR 法の検証計算

4.4.1　3 次元速度場における界面移流計算

AMR 法を用いて 2 章で行ったフェーズフィールド法の検証計算である 3 次元速度場における界面移流計算を行う．高解像度の格子を界面近傍に配置する．均一格子の結果と界面のプロファイルを比較し，AMR 法の計算精度を確認する．また，格子点数を均一格子と比べることで，AMR 法によりどの程度計算コストを削減できるのかを評価する．

計算領域 $[0,1]\times[0,1]\times[0,1]$ を設定し，半径 0.15 の球形プロファイルを位置 $(0.35, 0.35, 0.35)$ に配置する．球形のプロファイルを 3 次元の非圧縮性速度場 [83]

$$u\left(x,y,z,t\right)=2\sin^2\left(\pi x\right)\sin\left(2\pi y\right)\sin\left(2\pi z\right)\cos\left(\frac{\pi t}{T}\right) \tag{4.13}$$

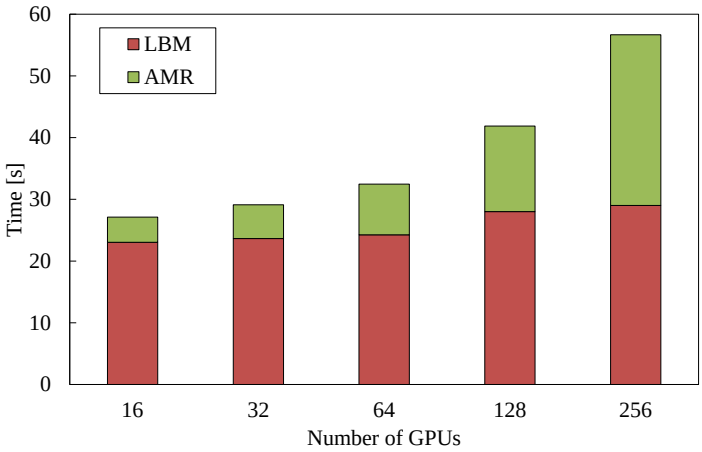


Fig.4.28: Breakdown of weak scaling results of LBM computation on adaptively refined meshes.

$$v\left(x,y,z,t\right) = -\sin\left(2\pi x\right)\sin^2\left(\pi y\right)\sin\left(2\pi z\right)\cos\left(\frac{\pi t}{T}\right) \tag{4.14}$$

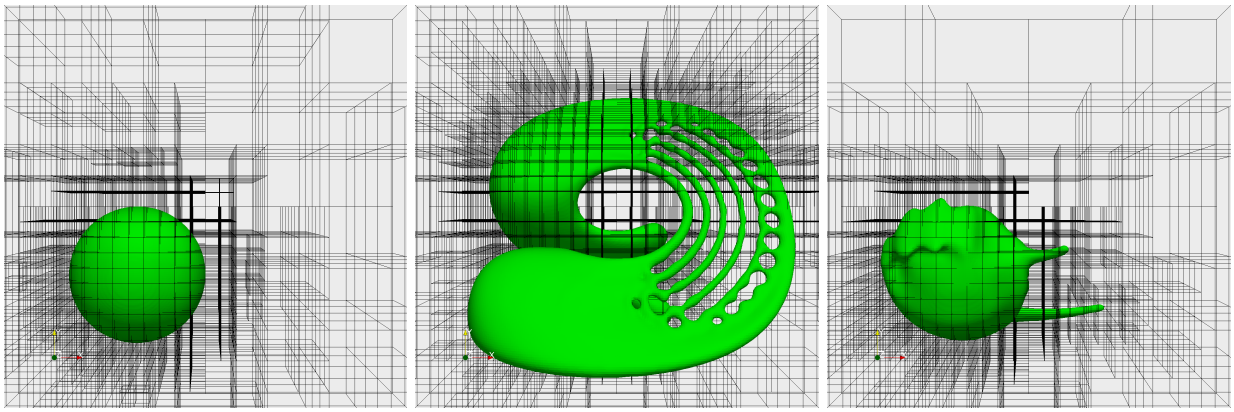
$$w\left(x,y,z,t\right) = -\sin^2\left(2\pi x\right)\sin\left(2\pi y\right)\sin^2\left(\pi z\right)\cos\left(\frac{\pi t}{T}\right) \tag{4.15}$$

で移流させる．ここで，t は時刻，T は周期であり，$T = 3$ を設定する．

フェーズフィールド法の拡散・逆拡散項のパラメータであるモビリティを $M = 0.01$ とする．均一格子で $256 \times 256 \times 256$，または $512 \times 512 \times 512$ 相当の解像度の格子を界面近傍に配置する．ブロック内に界面が含まれる場合，または隣接のブロックに界面が含まれる場合にブロックを細分化する．界面から離れた領域の格子は，隣接するブロックとの解像度差が 2 倍になるように粗くする．$256 \times 256 \times 256$ の均一格子での計算結果を Fig.4.29 の上図に，$256 \times 256 \times 256$ 相当の格子を界面近傍に割り当てた AMR 法の計算結果を下図に示す．AMR 法を導入した計算結果は均一格子の結果とほぼ一致しており，このことから，界面捕獲の計算には界面近傍にのみ高解像度格子を割り当てれば良いことがわかる．また，Fig.4.30 は $512 \times 512 \times 512$ 相当の格子を界面に配置した計算であり，界面が最も引き伸ばされる $t = 1.5$ においても界面が引きちぎれていない．$t = 3$ では，$256 \times 256 \times 256$ 相当の格子よりも初期プロファイルである球形に戻っている．

AMR 法を用いた場合の計算格子点数の時刻歴（左）と，均一格子と AMR 法の格子点数の比（右）を Fig.4.31 に示す．黒色のプロットが $256 \times 256 \times 256$ 相当の格子を界面に割り当てた計算，赤色のプロットが $512 \times 512 \times 512$ 相当の格子を用いた計算，黒色の実線は $256 \times 256 \times 256$ の均一格子の場合の格子点数である．AMR 法を用いることで均一格子よりも格子点数を削減できていることが確認でき，均一格子よりも高解像度の $512 \times 512 \times 512$ 相当の格子を用いた場合も，$256 \times 256 \times 256$ の均一格子よりも格

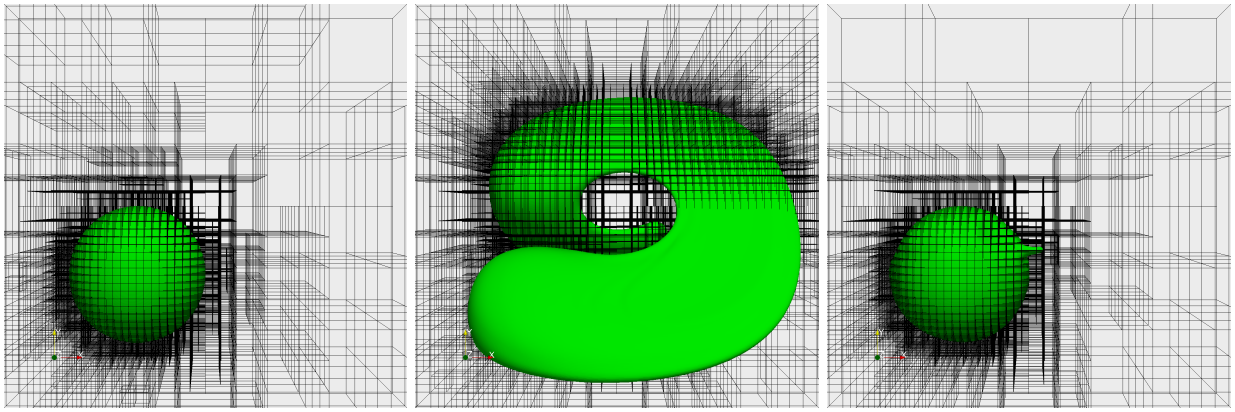
(a) Uniform mesh ($t = 0, 1.5, 3.0$).



(b) Adaptivelly refined mesh ($t = 0, 1.5, 3.0$). The mesh indicates the blocks.

Fig.4.29: Comparison of uniform mesh and adaptivelly refined mesh in three-dimensional interface capturing on the vortex velocity field. The grid resolution corresponds to $256 \times 256 \times 256$ lattice points.

子点数が少ない．このことから，AMR 法を用いることで計算コストを削減できていることがわかる．界面が引き伸ばされて表面積が増加するため，AMR 法の格子点数は $t = 1.5$ まで増加し，その後は初期と同程度まで格子点数が少なくなる．右図は AMR 法が同一の格子解像度の均一格子よりもどの程度格子点を削減できるかを確認するために，AMR 法と均一格子の格子点数の比を表している．$256 \times 256 \times 256$ 相当の細分化格子では，均一格子の約 8% から 22% に格子点数を削減できている．$512 \times 512 \times 512$ 相当の細分化格子では，均一格子の約 4% から 10% に格子点数を削減できている．解像度が高いほど AMR 法による格子点数の削減が効果的であることがわかる．これは，3 次元計算で均一格子の場合は解像度を 2 倍にすると格子点数は 8 倍になるが，AMR 法で界面に格子を集める場合は解像度を 2 倍にしても格子点数は 4 倍にしか増加しないためである．このことから，界面に高解像度格子を適合させる AMR 法は大規模計算になればなるほど効果的であることが明らかになった．



(a) $t = 0.$ (b) $t = 1.5.$ (c) $t = 3.0.$

Fig.4.30: Three-dimensional interface capturing on the vortex velocity field using AMR method. The grid resolution corresponds to $512 \times 512 \times 512$ lattice points. The mesh indicates the blocks.

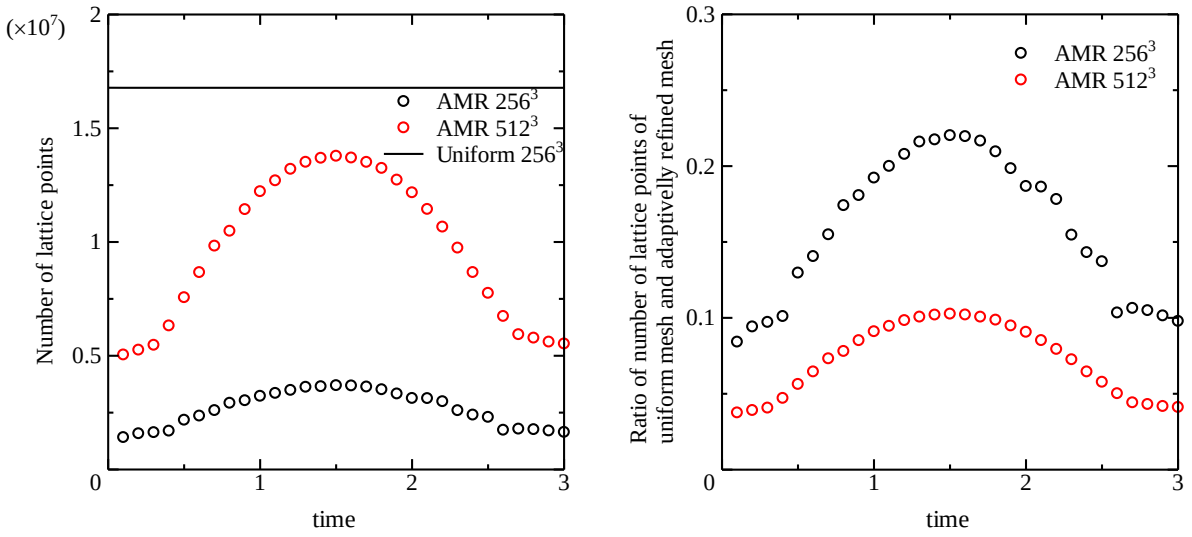


Fig.4.31: Time history of the number of lattice points (left) and the ratio of the number of lattice points of uniform mesh and adaptively refined mesh (right) in the three-dimensional interface capturing.

4.4.2 単一球形粒子の沈降

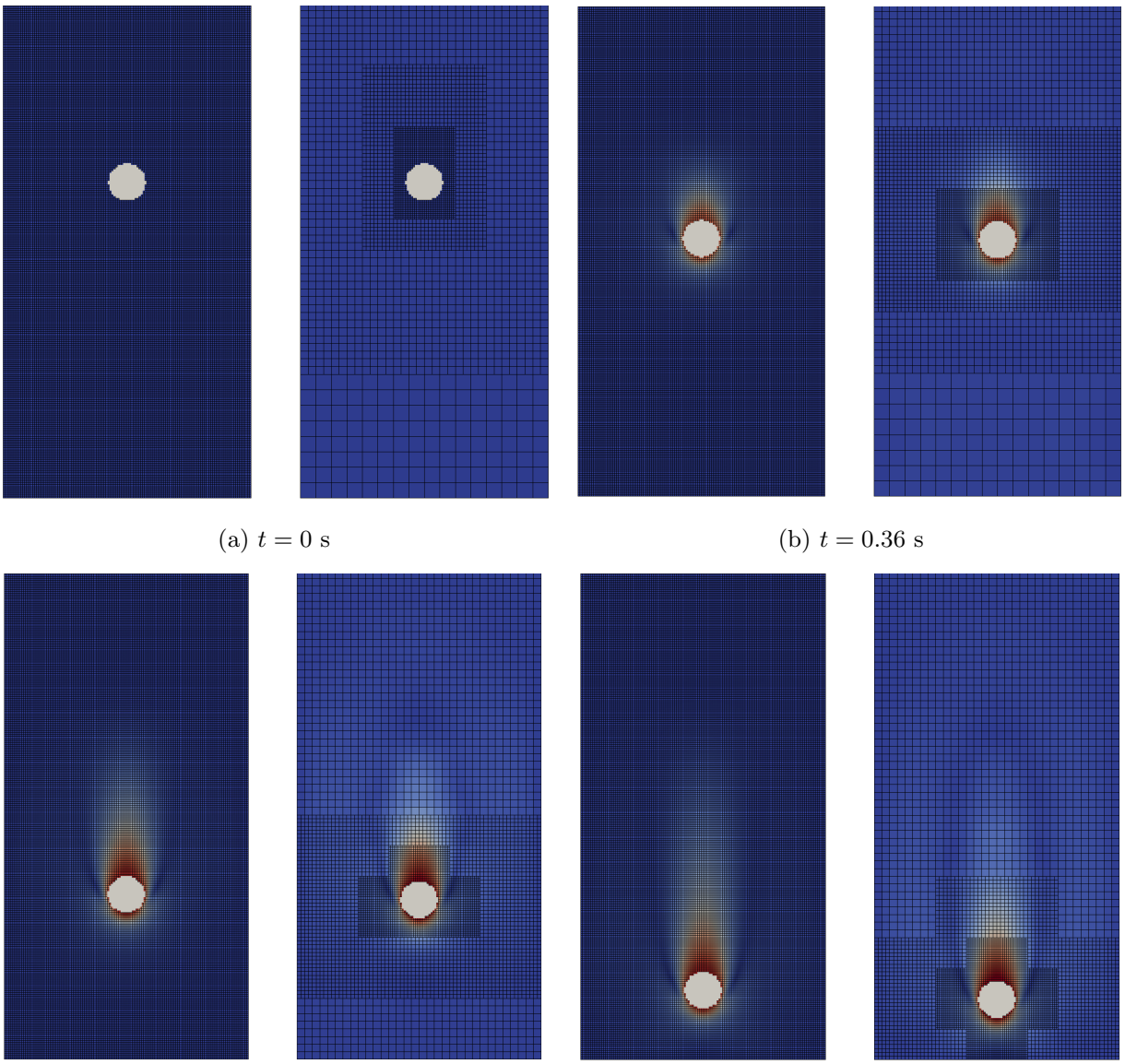
AMR 法を用いて単一球形粒子の沈降計算を行い，速度場のプロファイルと沈降速度を均一格子の結果と既往研究 [86] で行われた実験と比較する．球形粒子の近傍に高解像度の格子を配置し，それ以外の領域は解像度差が 2 倍以下になるように格子を粗くする．2 章で行った検証計算や先行研究の実験では

領域の大きさが 100 mm × 100 mm × 160 mm であるが，複数の木構造を用いてこの領域に対して細分化を行うと，多くの木構造が必要となり，粗い格子を割り当てるのが困難になる．そのため，計算領域を 100 mm × 100 mm × 200 mm と設定し，2 つの木構造で表現できる領域を用いる．格子解像度は 128 × 128 × 256 格子相当である．実験条件に合わせて球の直径を 15 mm，密度を 1,120 kg/m^3 と設定し，球の重心を床から 127.5 mm の位置に初期時刻に配置する．終端速度に対するレイノルズ数が 31.9 である Table2.1 の Case 4 の条件で計算を行う．

0.36 秒ごとの速度場のプロファイルを Fig.4.32 に示す．左が均一格子（128 × 128 × 256 格子），右が AMR 法を用いた計算結果である．格子は計算セルを示す．均一格子と同一の解像度を割り当てた球近傍の速度場は，均一格子を用いた場合とよく一致している．球の後方に発生する流れに粗い格子が割り当てらるため，後流のプロファイルは均一格子よりも拡散している．球の位置は僅かに AMR 法を用いたほうが下にあり，これは後流の影響によるものだと考えられる．

沈降速度の時刻歴を Fig.4.33 に示す．青色が均一格子を用いた場合の球の沈降速度，赤色が AMR 法を用いた場合の沈降速度，プロットは先行研究 [86] の実験の結果である．AMR 法を用いた法が球の沈降速度が速いことがわかる．均一格子の場合は沈降速度の最大値は 0.117 m/s，AMR 法を用いた場合は 0.121 m/s であり，誤差は 3.4% ほどである．AMR 法を用いても均一格子と同等の計算が行えることが確認できた．

この計算における格子点数の時間変化を Fig.4.34 に示す．AMR 法により格子点数を 6% から 8% に削減できた．



(a) $t = 0$ s (b) $t = 0.36$ s

(c) $t = 0.72$ s (d) $t = 1.08$ s

Fig.4.32: Simulation results of single particle settling on the uniform mesh (left) and adaptively refined mesh (right). The color indicates the velocity magnitude. The mesh indicates cells.

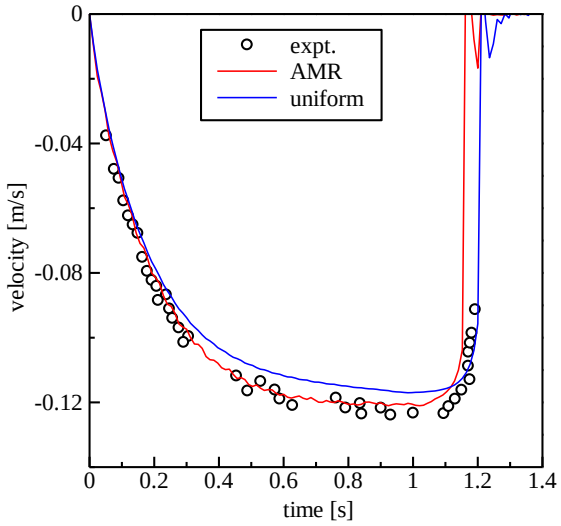


Fig.4.33: Simulation results of settling velocity of a sphere using uniform mesh (blue) and adaptively refined mesh (red). Black plots are the experimental result.

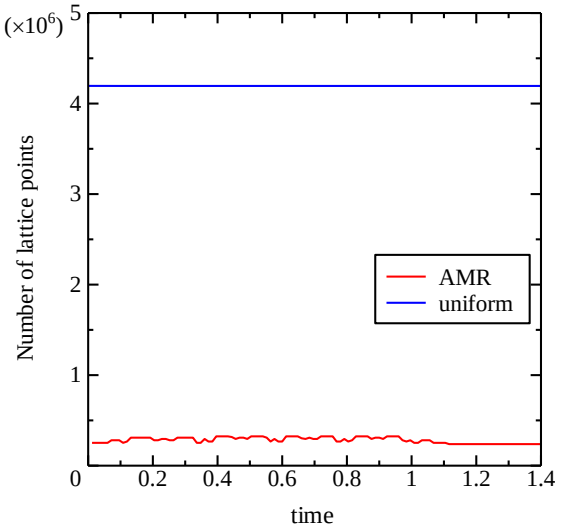


Fig.4.34: Number of lattice points of the single sphere settling simulation using uniform mesh (blue) and adaptively refined mesh (red).

4.5 AMR 法を用いた大規模流体解析

この節では，第 2 章で提案した混相流の解析手法に AMR 法を導入し，東京工業大学 GPU スパコン TSUBAME3.0 を用いた大規模計算を行う．

4.5.1 高レイノルズ数の球周りの流れ

球の抗力係数はレイノルズ数が 20 万から 30 万で急激に低下し，この現象はドラッグクライシスと呼ばれる．ドラッグクライシスは境界層が乱流化により渦の剥離点が球の後方に移動すること原因であると言われている [114].

AMR 法を導入したキュムラント型の格子ボルツマン法を用いて 3 次元の球周り流れの計算をレイノルズ数 10 万，20 万，30 万，50 万の条件で行った．球の壁面には Interpolated bounce-back 法 [77] を適用する．計算領域を $2 \times 1 \times 1$ とし，直径が 0.0625 の球を $(0.5, 0.5, 0.5)$ の位置に配置した．計算格子を Fig.4.35 に示す．動的な格子細分化を行わない固定格子で計算を行った．ブロックあたりの格子点数は $8 \times 8 \times 8$ に設定した．上図が全体の計算領域を示し，球の後方に発生する渦を計算するための解像度 $\Delta x = d/256$ の格子を球の後方に配置する．下図は球近傍の格子を示し，球の表面には粘性低層を解像するための高解像度格子を配置する．球の直径を d とすると，球周りの流れにおける粘性低層の厚さ δ_b は

$$\delta_\mathrm{b} = 3d\sqrt{\frac{1}{2Re}} \tag{4.16}$$

と見積もられる [115]．この式から，粘性低層の厚さはレイノルズ数 10 万で $\delta_\mathrm{b} \approx d/149$，50 万で $\delta_\mathrm{b} \approx d/333$ と計算できる．球表面の格子解像度を $\Delta x = d/1024$ として直径に対して 1024 格子を割り当てた計算と，$\Delta x = d/2048$ として直径に 2048 格子を割り当てた計算を実施する．それぞれ，レイノルズ数 50 万の条件で境界層に対して約 3 格子，約 6 格子を割り当てた条件である．最も粗い格子の解像度は計算領域に対し $128 \times 64 \times 64$ 相当であり，直径に対して 4 格子である．速度 0.03 の一様流れを流入させる．総格子点数は球直径に 2048 格子を用いた場合で約 12.2 億であり，最も細かい格子で領域全体を分割した均一格子（$65,536 \times 32,768 \times 32,768$）と比較すると，格子点数はわずか 0.0173% である．計算には 48 台の GPU Tesla P100 を用い，タイムステップ数は約 157 万である．計算時間は約 18 時間である．

抗力係数 C_D を球が受ける力を F_x，投影断面積を S として

$$C_D = \frac{F_x}{\frac{1}{2}\rho U^2 S} \tag{4.17}$$

と計算し，レイノルズ数に対する抗力係数の計算結果を Fig.4.36 に示す．黒と灰色の実線は実験値に基づく参照値である [85][116]．青いプロットが球直径に 1024 格子を割り当てた結果，赤いプロットが球直径に 2048 格子を割り当てた結果である．直径に 1024 格子を割り当てた計算では，レイノルズ数によら

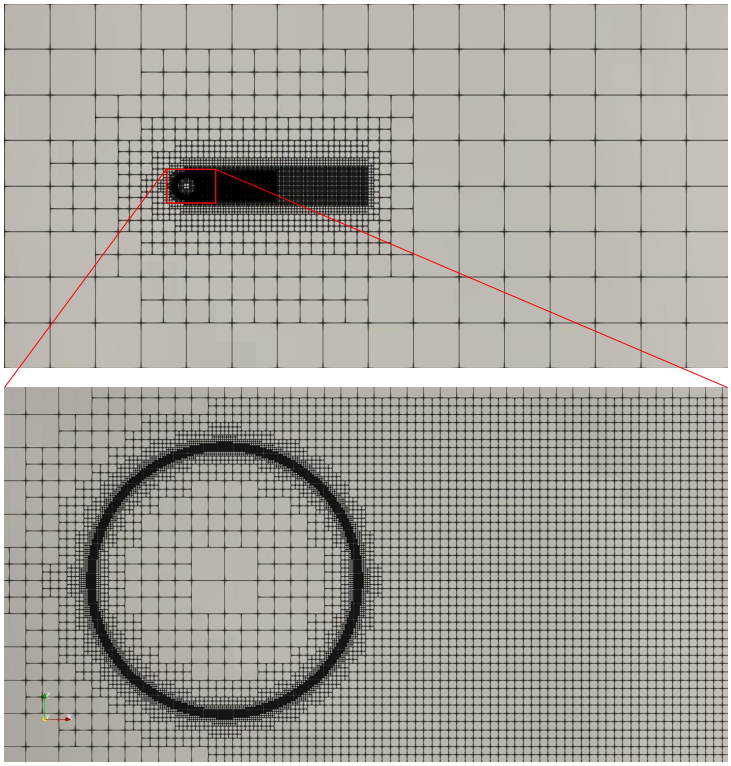


Fig.4.35: The locally refined mesh for simulations of flow around a sphere (top: entire domain, bottom: near the sphere). The mesh shows the block with $8 \times 8 \times 8$ lattice points.

ず効力はほぼ同じ値を取っている．直径に 2048 格子を割り当てた計算では，レイノルズ数が 30 万で効力が下がり始め，レイノルズ数 50 万で抗力係数が急激に低下している．高レイノルズ数で抗力係数が低下するドラッグクライシスを再現できており，妥当な結果だと言える．キュムラントモデルを提案した Geier らは，2017 年に緩和係数を適切に設定した新しいキュムラントモデル [58] を提案しており，緩和係数を設定した新しいモデルでは直径に対してわずか 512 格子でドラッグクライシスの再現に成功している [117]．この論文では 2015 年のキュムラントモデルではドラッグクライシスを再現できないとしているが，本研究の計算により，キュムラントモデルでも球直径に 2048 格子を割り当てた大規模計算によりドラッグクライシスが再現できることが明らかになった．

流れ場を確認するため，渦の可視化によく用いられる速度勾配テンソルの第二不変量である Q 値を

$$Q = \frac{1}{2}(\nabla \times \boldsymbol{u}) \cdot (\nabla \times \boldsymbol{u}) - \frac{1}{4}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)^2 \tag{4.18}$$

と計算する．ここで，$\boldsymbol{u} = \{u_1, u_2, u_3\} = \{u, v, w\}$，$\{x_1, x_2, x_3\} = \{x, y, z\}$ である．速度の絶対値で色付した Q 値の等値面を Fig.4.37 に示す．球の直径に 2048 格子を割り当てた計算では，レイノルズ数 50 万で渦の剥離点が球の後方に移動しているのが確認できた．しかし，Fig.4.36 の抗力係数は実験値に比べて抗力係数が低下するレイノルズ数が高く，数値粘性によって粘性を過大評価していると考えられる．武藤らは非構造格子を用いて層流境界層に対して 20 メッシュを割り当てることでドラッグクライシスを

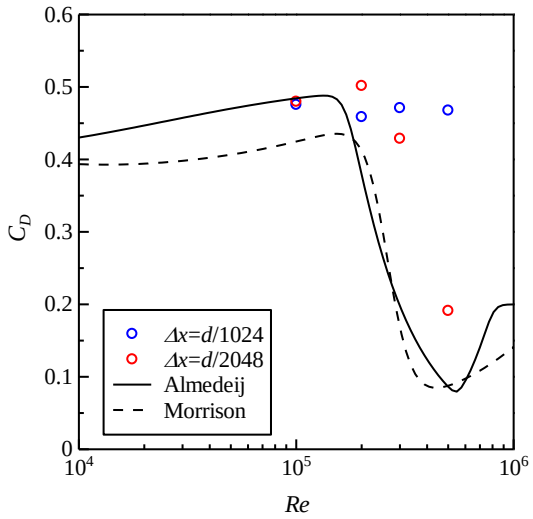
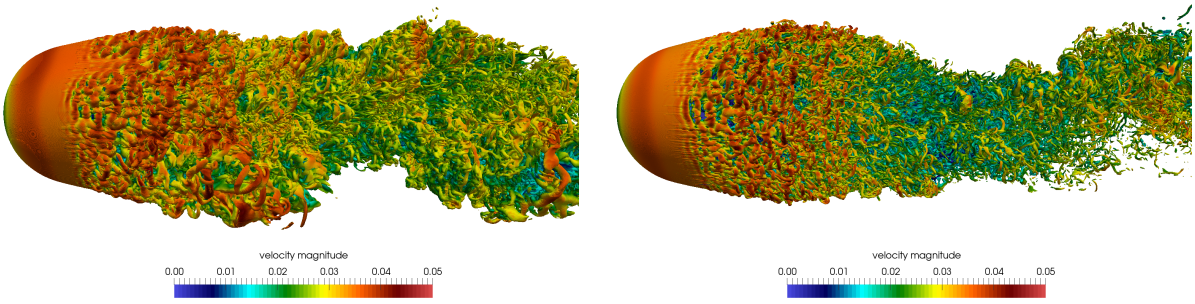
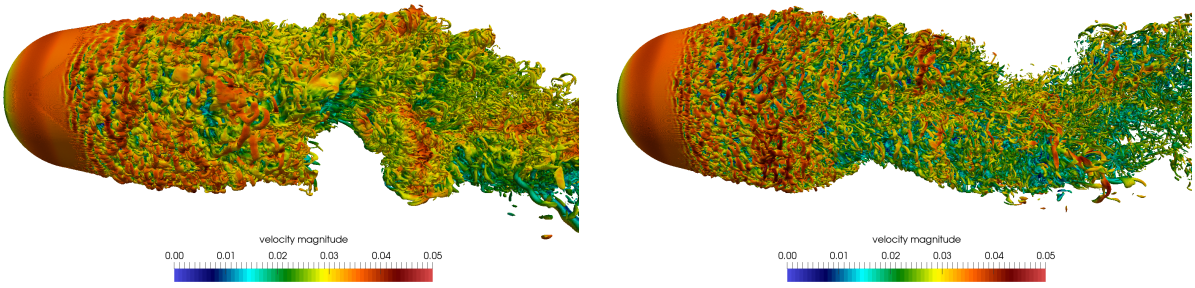


Fig.4.36: The drag coefficient of a sphere.

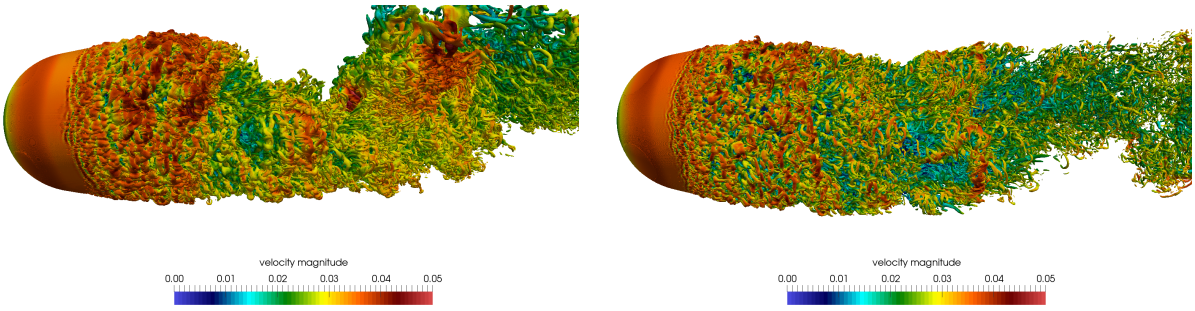
再現しており [115]，このことからも層流境界層に対して 6 格子を割り当てた本計算は解像度が十分ではないと言え，より高解像度の格子を用いる必要があると考えられる．



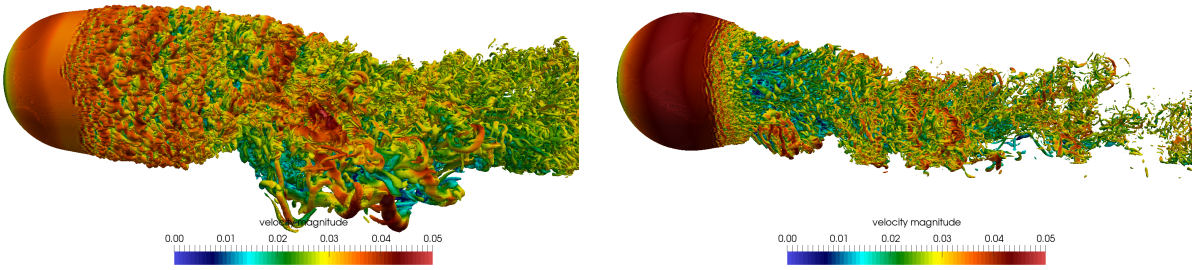
(a) $Re = 100,000$ (left:$\Delta x = d/1024$, right:$\Delta x = d/2048$).



(b) $Re = 200,000$ (left:$\Delta x = d/1024$, right:$\Delta x = d/2048$).



(c) $Re = 300,000$ (left:$\Delta x = d/1024$, right:$\Delta x = d/2048$).



(d) $Re = 500,000$ (left:$\Delta x = d/1024$, right:$\Delta x = d/2048$).

Fig.4.37: A contour plot on the Q criterion. The color shows the velocity magnitude.

4.5.2　噴流により浮遊するピンポン玉

上に向けてドライヤーの風を出し，その上でピンポン玉を放すとピンポン玉は落ちることなく浮遊する．この現象に関する実験的研究が行われており，噴流中の球に作用する効力は一様流中の球と大きく異なることが報告されている．Neve らは噴流ノズルと球の距離を変えて効力を測定し，噴流がノズルから出てしばらくは球に作用する効力は一定になるが，ノズルからの距離が離れると乱流が弱まり効力が距離の-1.8 乗で下がることを示した [118]．また，Lopex-Arias らは噴流の中心軸から球をずらして効力を測定し，噴流の流入口が球よりも太いか細いかにより球に作用する効力の傾向が変わり，流入口が太い場合はジェットの中央で効力の極大値が発生するが，ジェットが細い場合は中央に極小値があることを実験から明らかにした [119]．このように，噴流中の球の効力はレイノルズ数だけでなく，ノズルと球の位置関係やノズルの大きさに影響される．

本研究では，AMR 法を導入した格子ボルツマン法を用いて，噴流と球の連成計算を行う．球はピンポン玉を仮定し，直径を 40 mm，質量と 2.72 g とした．計算領域の境界の影響を減らすために，$1.28 \times 1.28 \times 2.56$ m^3 の広い計算領域を設定し，底面の中央から直径 48 mm の噴流を流入させる．初期時刻において流入口の中心に床から 160 mm の位置に球を配置する．流体は 40 ℃の空気とし，密度 1.128 kg/m^3, 動粘度 1.698×10^{-5} m^2/s とする．流入速度は $13.125, 15.0, 16.875, 18.75$ m/s の 4 つの条件で計算を行う．渦度の大きさを格子細分化の閾値とし，球の表面と渦に対して解像度 ($\Delta x = 1.25$ mm) の格子を割り当てる．ブロックあたりの格子点数は $8 \times 8 \times 8$ に設定した．物理時間で 5.24 秒までの計算を行う．計算には Tesla P100 を 4 台用いた．

計算結果の一例として，流入速度が 15 m/s の条件における計算のスナップショットを Fig.4.38 に示す．左図が Q 値を用いた渦の可視化，右図が計算領域中央における速度の大きさである．噴流と球の表面に高解像度格子が割り当てられ計算していることが確認できる．

球の高さの時間変化のグラフを Fig.4.39 に示す．どの条件においても球は上下に運動していることが確認できる．これは，球が落下し流入口に近づくと大きな効力が作用し球は上昇し，流入口から離れると効力が弱くなり落下するためだと考えられる．また，球の上下の運動は噴流の流速が大きいほど大きいことが明らかになった．なお，流入速度を $15.0, 16.875, 18.75$ m/s とした計算では，それぞれ $4.0, 2.3, 3.6$ 秒で球が床に落下したため計算を途中で終了した．

噴流の中心軸と球の重心の水平距離の時間変化を Fig.4.40 に示す．どの条件においてもピンポン玉は水平方向に揺れていることがわかる．安定して浮遊した流入速度 13.125 m/s では，球の重心が流入口よりも外に出ることなく揺れていることがわかる．他の条件では，流入口を外れる場合があり，流入速度 $15.0, 18.75$ m/s では噴流の中心軸から大きく離れ，噴流からの流体力がなったため床に落下してたことがわかる．

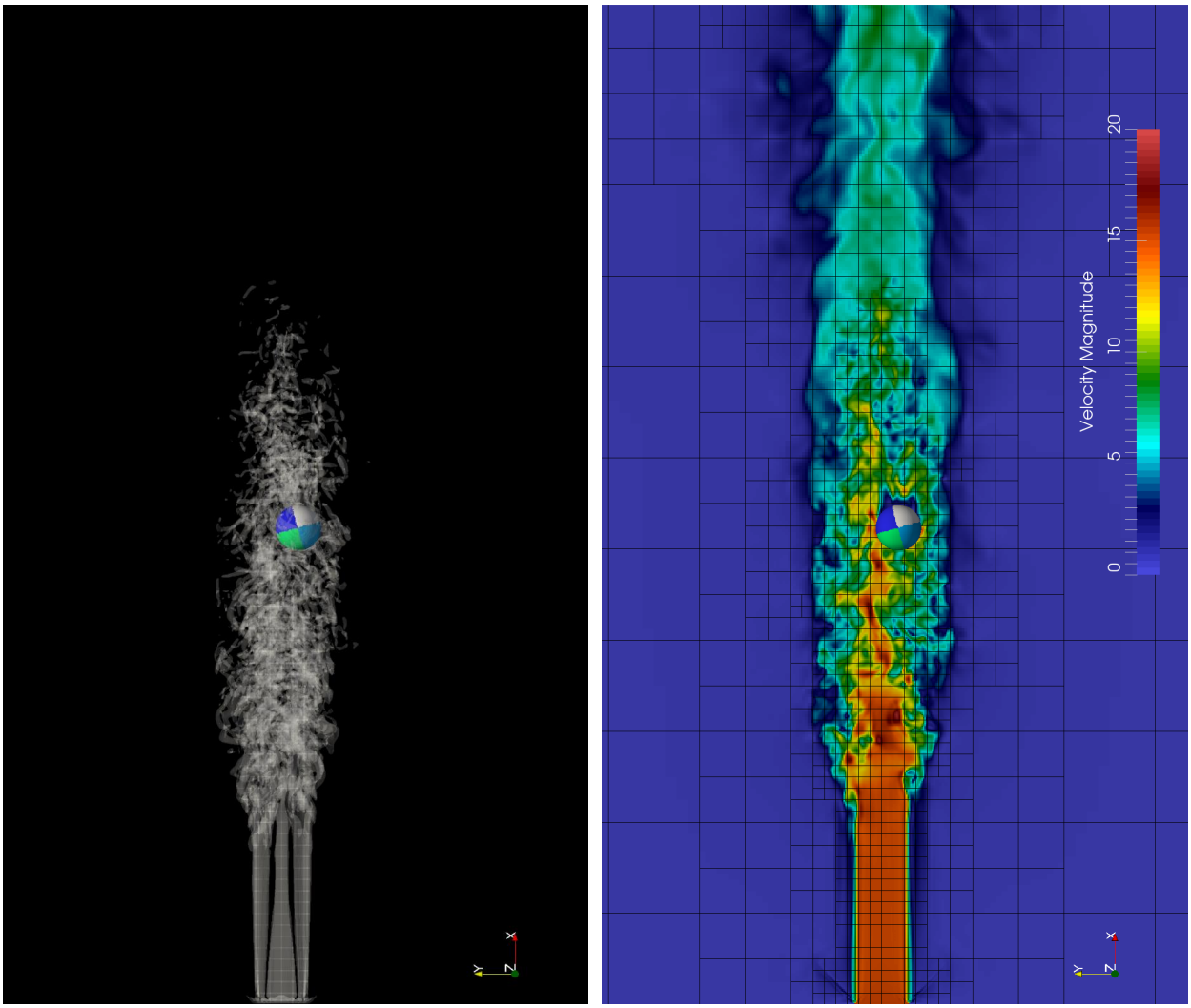


Fig.4.38: The simulation result of a ping-pong ball interacting with a jet flow ($u_{\mathrm{in}} = 15$ m/s). The ping-pong ball is colored in 8 colors to indicate rotation. The left panel shows vortices by the isosurface of Q criterion. The right panel shows the velocity magnitude. The mesh indicates blocks including $8 \times 8 \times 8$ lattice points.

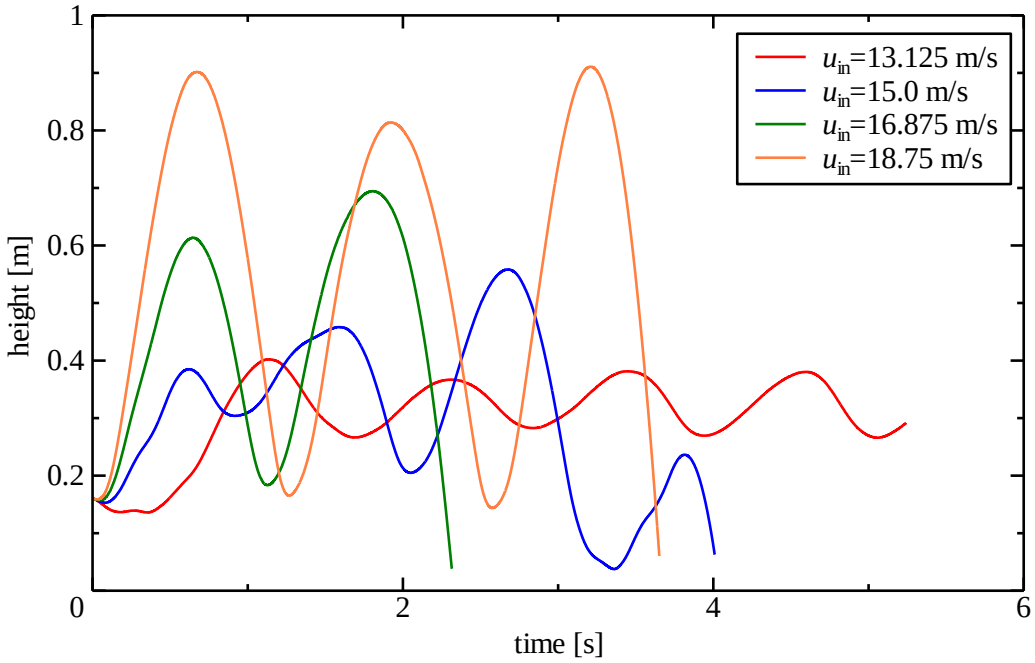


Fig.4.39: Time history of the height of the ping-pong ball.

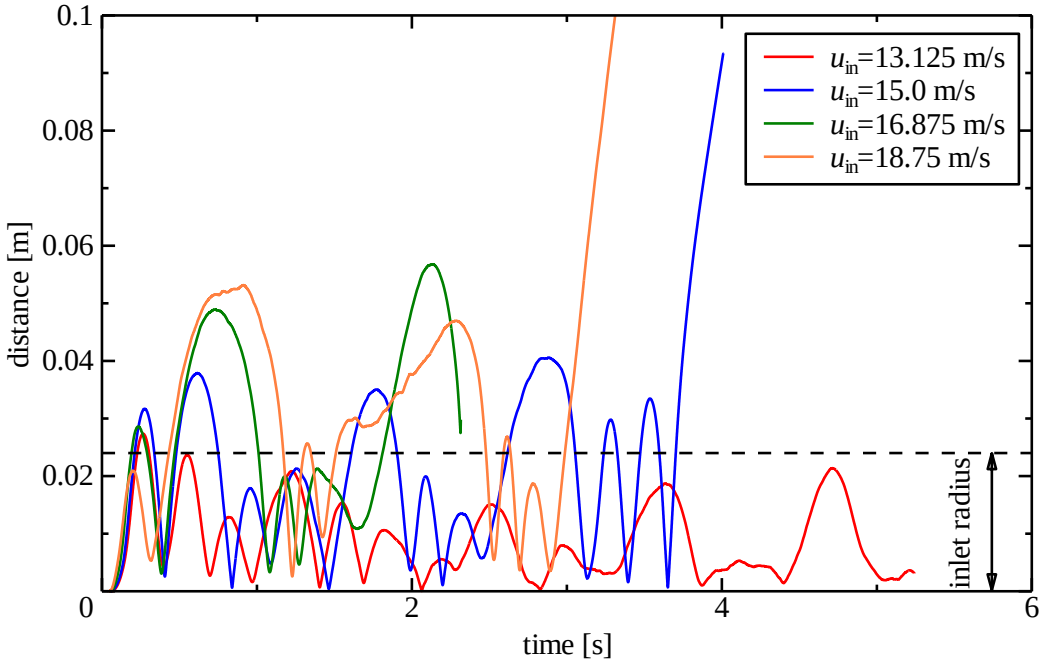


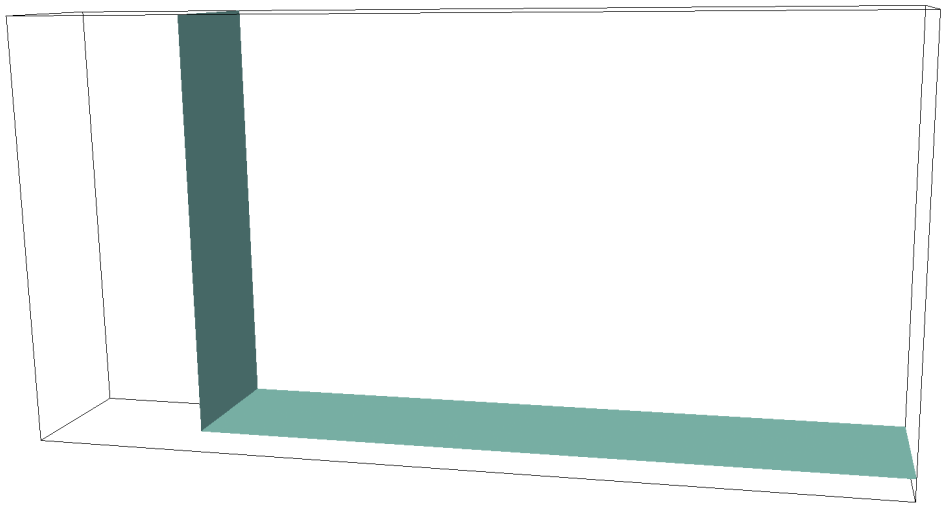
Fig.4.40: Time history of distance between the center of the inlet and the ping-pong ball. The dot line indicates the radius of the inlet nozzle.

4.5.3　大規模ダム崩壊計算

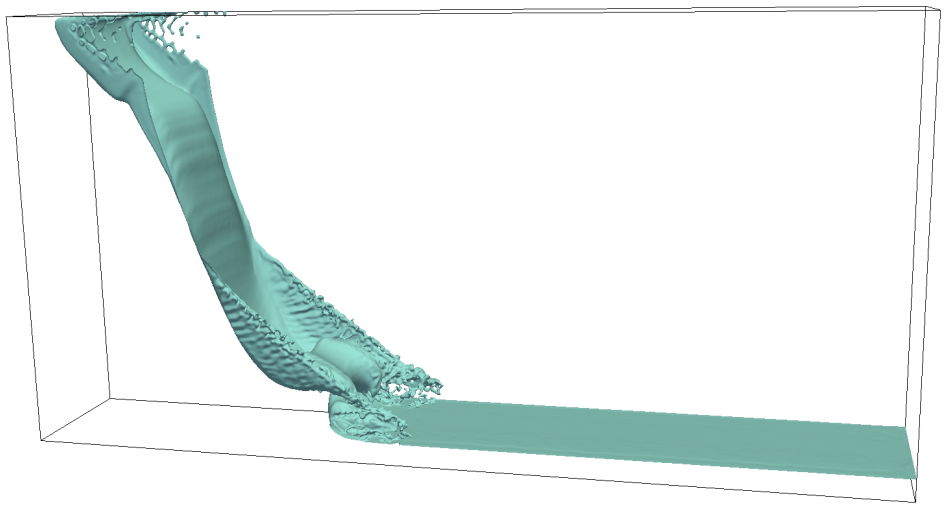
AMR 法を用いて 2 章の検証で行った濡れた床へのダム崩壊問題の大規模計算を行う．流体は水を仮定し，密度 1000 kg/m^3，動粘度 1.0×10^{-6} m^2/s とする．0.72 m × 0.12 m × 0.36 m の計算領域を設定し，計算領域の端に 0.15 m × 0.12 m × 0.36 m の水注を設置する．水深 0.018 m の浅い水面を設定する．格子幅は検証計算の四分の一の 0.23 mm に設定し，均一格子で 3072 × 512 × 1536 格子点相当である．ブロックあたりの格子点数は 8 × 8 × 8 に設定した．壁面の境界条件として，滑りなし境界条件を課す．保存型 Allen-Cahn 方程式のモビリティを $M = 0.05$ とする．時間刻み幅は 0.703×10^{-6} s と設定し，計算には 64 台の GPU を用いる．

計算結果を Fig.4.41 に示し，2 次元平面におけるブロックと領域分割の結果を Fig.4.42 に示す．高解像度格子を界面に割り当てることで，より細かい液滴の飛沫や壁面に張り付く液滴を表現できている．ダム崩壊のようなレイノルズ数の高い激しい流れでは，微小な液滴が発生し，それらを捉えるためには高解像度を用いた大規模計算が必要であることが示された．また，Fig.4.42 から本手法は界面に動的に高解像度格子を配置して計算を行っていることが確認できる．各 GPU が計算している小領域は界面のプロファイルの変化に合わせて変更され，モートン曲線を用いた負荷分散が行えていることがわかる．

格子点数の時間変化のグラフを Fig.4.43 に示す．ダム崩壊による界面の変形と壁面近傍に張り付いた液滴，飛沫などにより界面の表面積が増加し，時間の経過と主に格子点数が増加しているのが確認できる．格子点数が最も多い $t = 0.34$ s における格子点数は約 4 億 1 千万であり，3072 × 512 × 1536 の均一格子の 17% に削減できている．計算には 64 台の GPU（Tesla P100）を用い，524,288 ステップの計算に対して 24 時間で実行した．

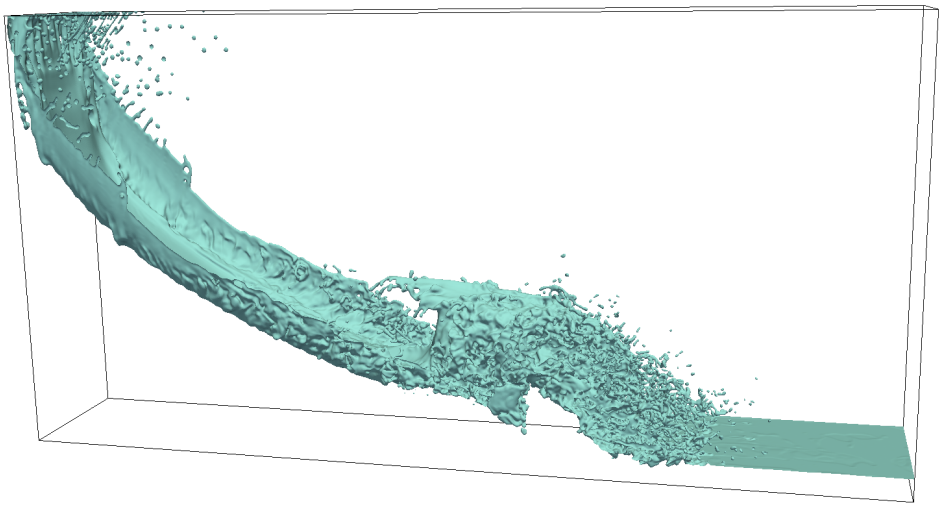


(a) $t = 0.0$ s.

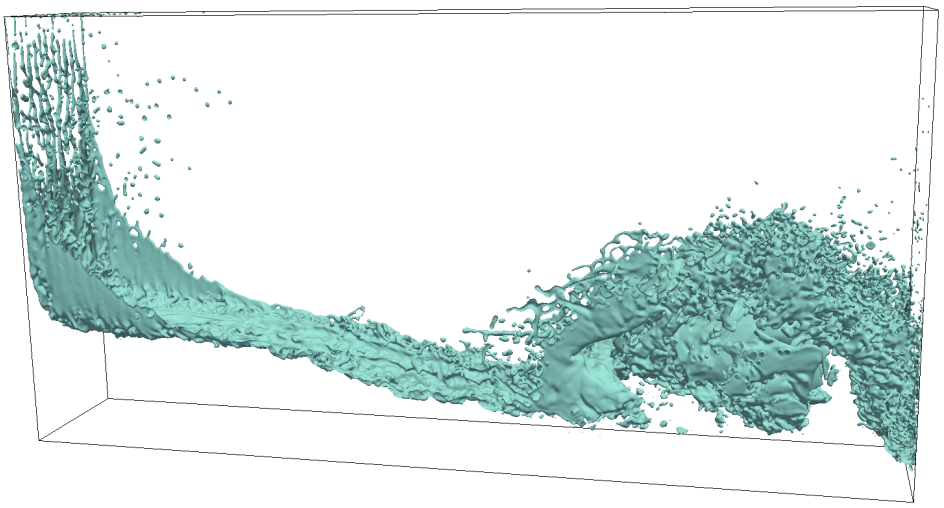


(b) $t = 0.1152$ s.

Fig.4.41: A large-scale simulation for a breaking dam process on a wet floor with the AMR method using 64 GPUs. The mesh resolution correspond to $3072 \times 512 \times 1536$ uniform mesh.

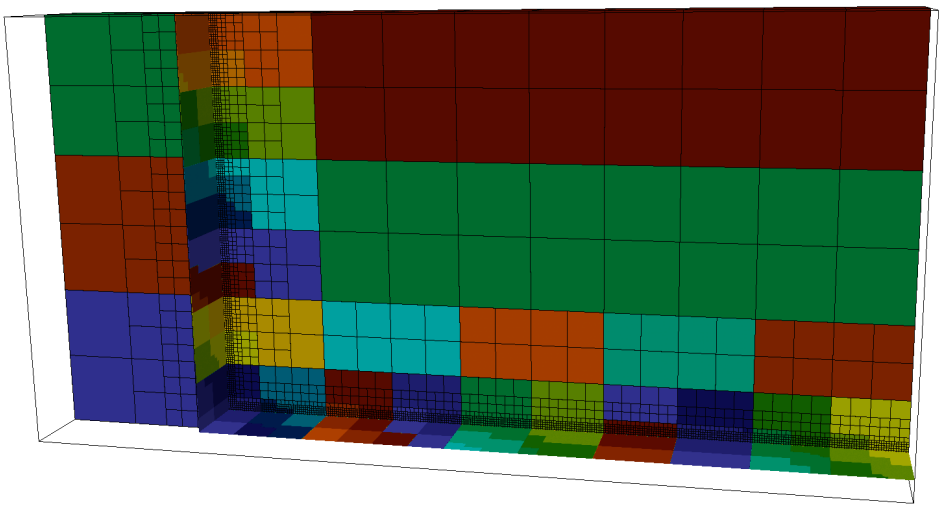


(c) $t = 0.2304$ s.

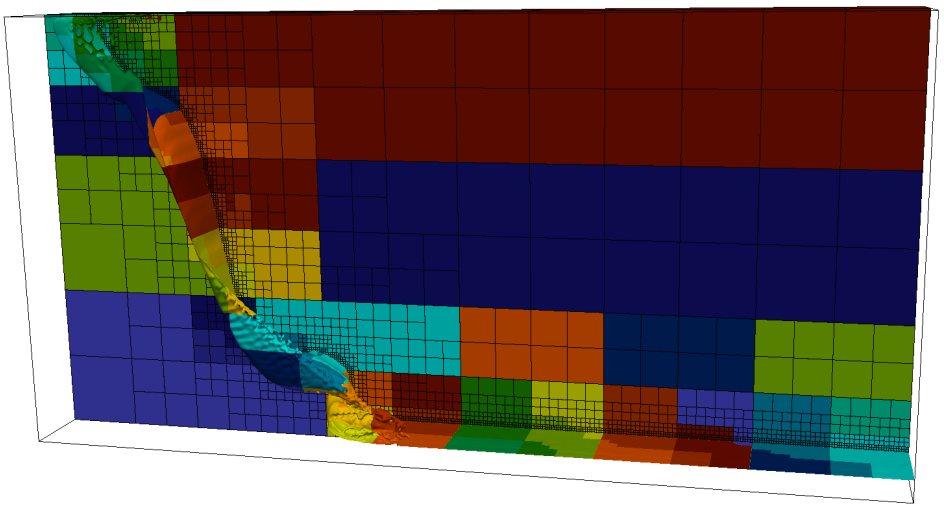


(d) $t = 0.3456$ s.

Fig.4.41: A large-scale simulation for a breaking dam process on a wet floor with the AMR method using 64 GPUs. The mesh resolution correspond to $3072 \times 512 \times 1536$ uniform mesh. (Continued.)

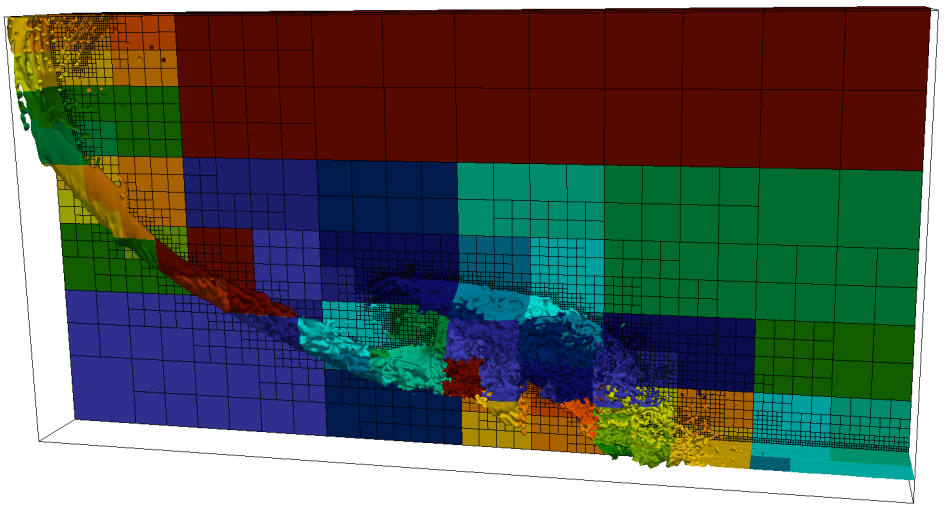


(a) $t = 0.0$ s.

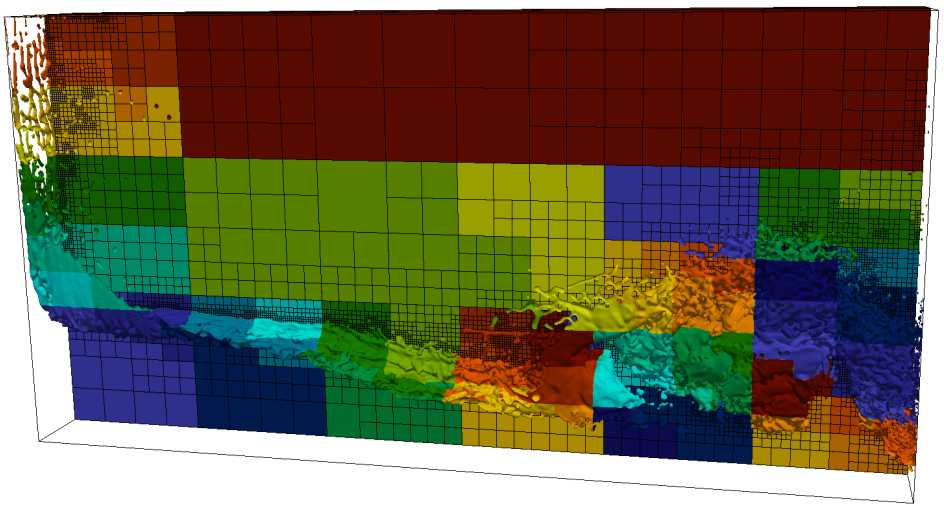


(b) $t = 0.1152$ s.

Fig.4.42: A large-scale simulation for a breaking dam process on a wet floor with the AMR method using 64 GPUs. The mesh resolution correspond to $3072 \times 512 \times 1536$ uniform mesh. The mesh indicates the blocks and the colors indicate the domain decomposition.



(c) $t = 0.2304$ s.



(d) $t = 0.3456$ s.

Fig.4.42: A large-scale simulation for a breaking dam process on a wet floor with the AMR method using 64 GPUs. The mesh resolution correspond to $3072 \times 512 \times 1536$ uniform mesh. The mesh indicates the blocks and the colors indicate the domain decomposition. (Continued.)

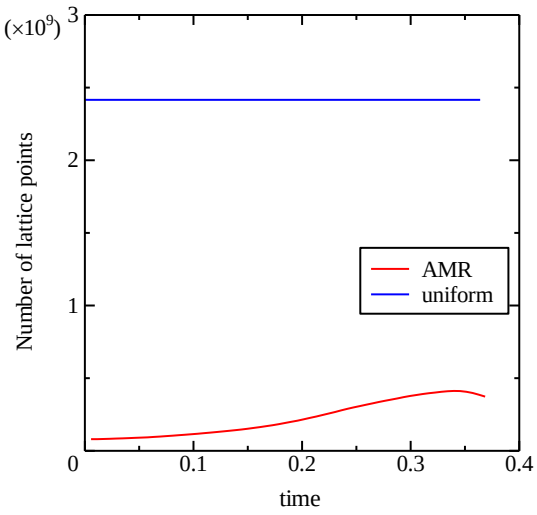


Fig.4.43: Number of lattice points of the breaking dam simulation using uniform mesh (blue) and adaptively refined mesh (red).

4.5.4　回転板による自由界面の撹拌

移動物体を含む自由界面流れの計算として，Fig.4.44 のように水面の上に回転する 3 枚の板を配置し，3 枚の板を互い違いに回転させて界面を撹拌する．計算領域を 10.24 m × 5.12 m × 10.24 m と設定し，水深 5.12 m の水を配置する．板の寸法は 6 m × 1 m × 0.01 m とし，手前と奥側の板を時計回り，中央の板を反時計回りに回転させる．回転速度を 8 rpm と 16 rpm と設定した 2 つの条件で計算し，板の先端の移動速度はそれぞれ 2.5 m/s，5.0 m/s である．最も細かい格子の解像度を 2 cm とし，板の表面と自由界面に格子を適合する．ブロックあたりの格子点数は $8 \times 8 \times 8$ に設定した．時間刻み幅は 8 rpm の条件では 60 μs，16 rpm の条件では 30 μs と設定する．計算には 8 台の GPU を用い，262,144 ステップの計算を実行する．境界条件には滑りなし条件を課した．

回転速度を 8 rpm とした条件の計算結果を Fig.4.45 に，16 rpm とした結果を Fig.4.46 に示す．回転速度が速いほど自由界面が大きく乱れていることがわかる．回転速度 16 rpm の計算では，回転板が水中から出る際に水を持ち上げられ，水が落ちる際に液滴が形成されている．提案手法はこのように移動物体により界面が大変形する問題に対しても安定に計算が可能であることが示せた．

格子点数の時間変化のグラフを Fig.4.47 に示す．青色が均一格子で計算した場合の格子点数，赤色が AMR 法を用いた板の回転速度が 8 rpm の計算，緑が板の回転速度が 16 rpm の計算での格子点数である．横軸は板が回転した回数である．どちらの条件でも周期的に格子点数が増減していることがわかる．格子点数が増加している箇所は，Fig.4.46 の (f) の前後の状態であり，板により界面が乱されるため格子点数が増加している．8 rpm の条件よりも 16 rpm の条件のほうが格子点数が多く，これは回転速度が速いほど界面が乱れるためだと考えられる．$512 \times 256 \times 512$ の均一格子に対して，AMR 法を導入することで格子点数を 8 rpm の条件では 25% 以下に，16 rpm の条件では 33% 以下に削減できた．16 rpm の計算に対して 8 台の GPU を用いて，計算時間は 7.2 時間であった．

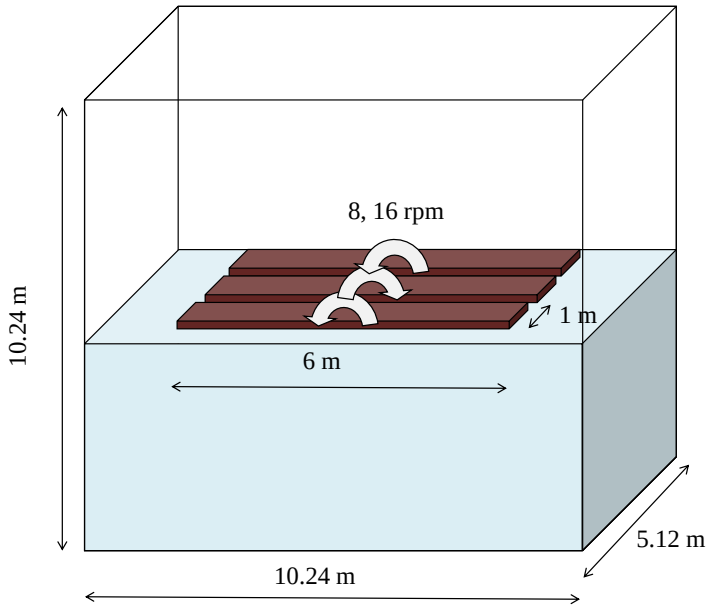
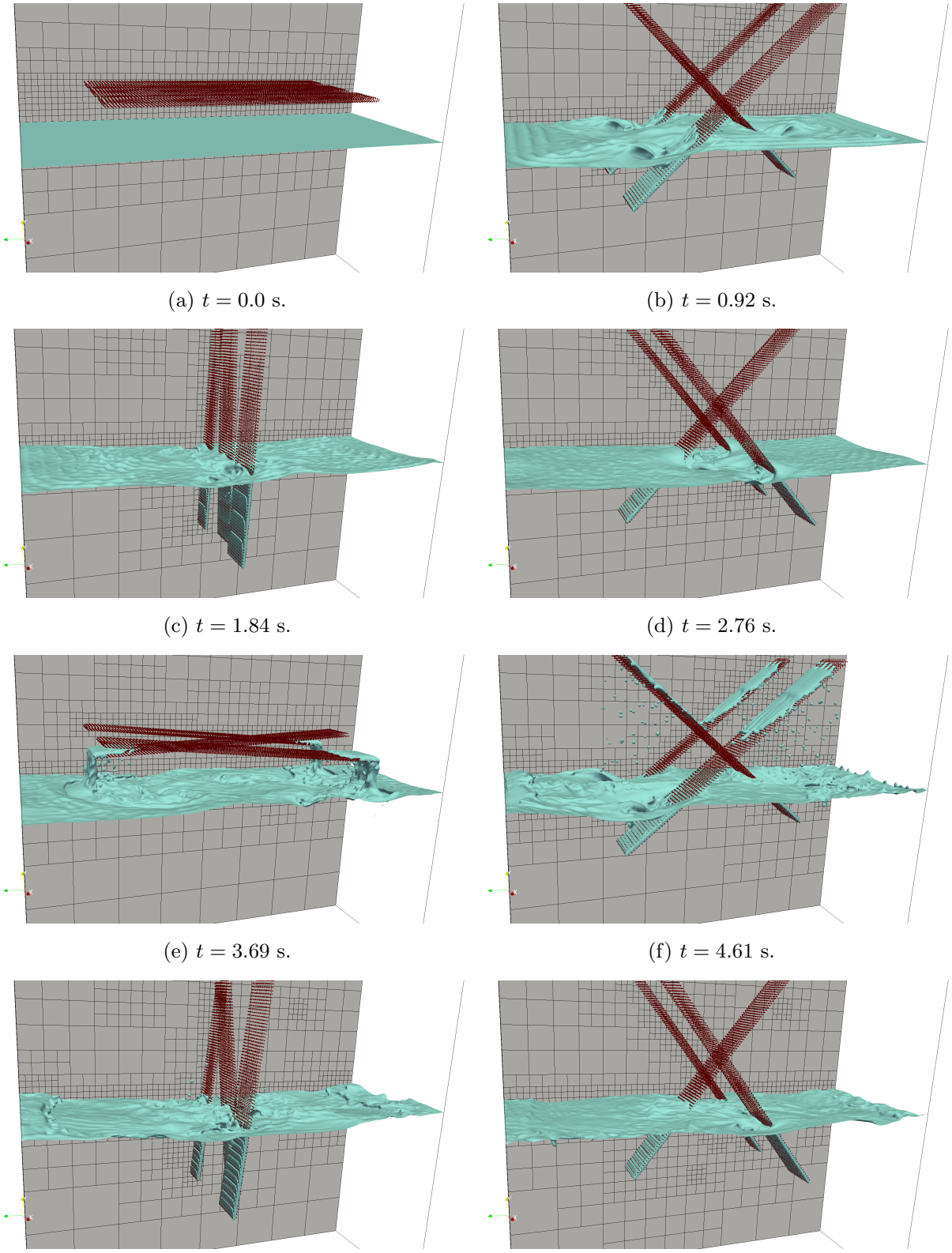


Fig.4.44: A simulation setting for agitation of water surface by rotating plates.



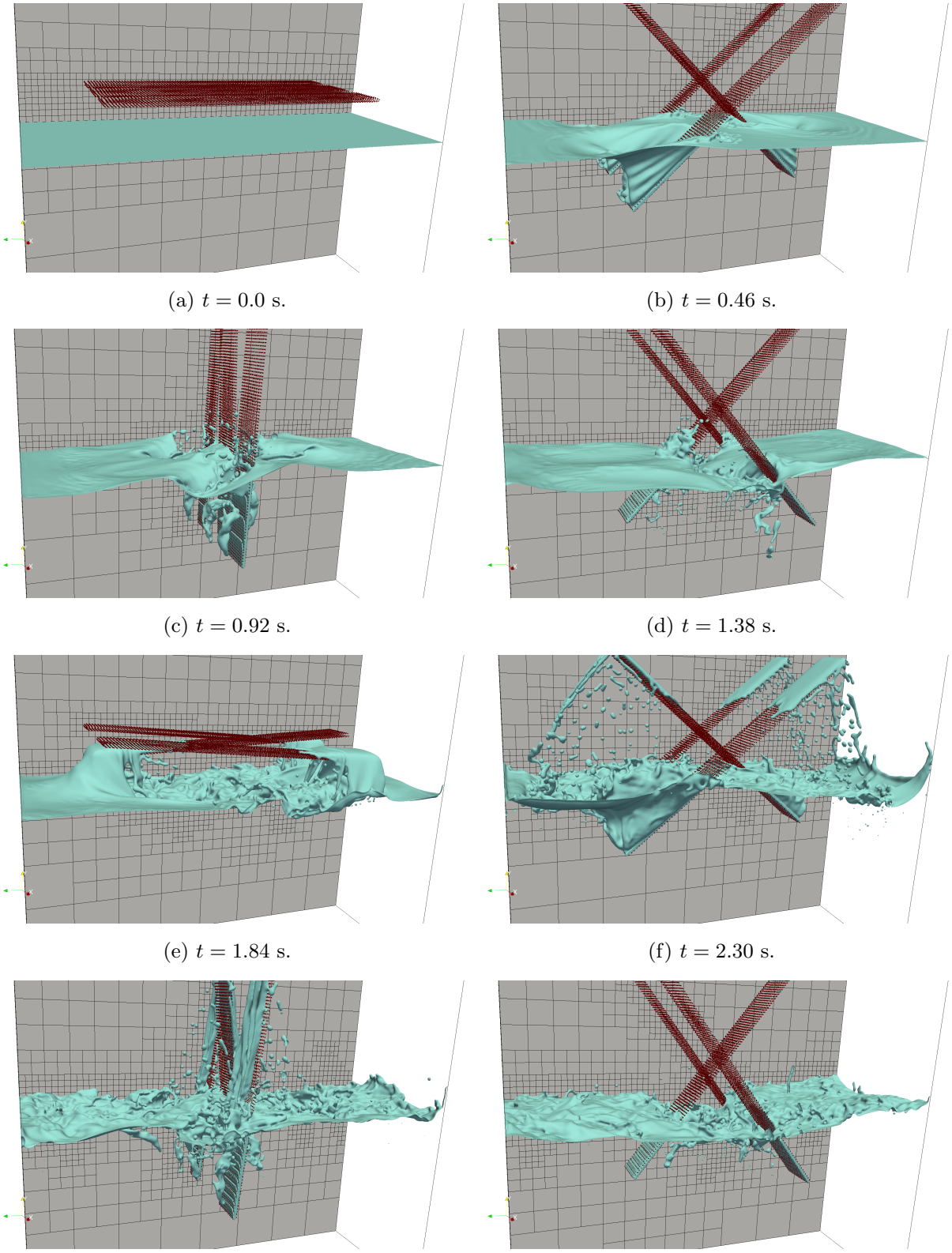
(a) $t = 0.0$ s.　(b) $t = 0.92$ s.

(c) $t = 1.84$ s.　(d) $t = 2.76$ s.

(e) $t = 3.69$ s.　(f) $t = 4.61$ s.

(g) $t = 5.53$ s.　(h) $t = 6.45$ s.

Fig.4.45: A simulation result of agitation of water surface by plates rotating at 8 rpm. The mesh indicates blocks.



(a) $t = 0.0$ s. (b) $t = 0.46$ s.

(c) $t = 0.92$ s. (d) $t = 1.38$ s.

(e) $t = 1.84$ s. (f) $t = 2.30$ s.

(g) $t = 2.76$ s. (h) $t = 3.32$ s.

Fig.4.46: A simulation result of agitation of water surface by plates rotating at 16 rpm. The mesh indicates blocks.

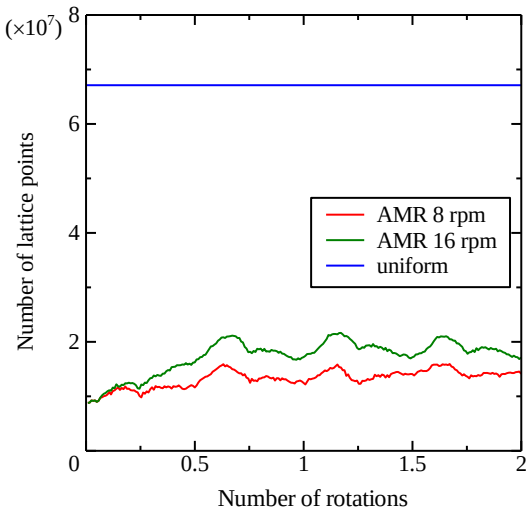


Fig.4.47: Number of lattice points of agitation of water surface using uniform mesh (blue) and adaptively refined mesh (red: 8 rpm, green: 16 rpm).

4.6 AMR 法を導入した格子ボルツマン法の大規模 GPU 計算のまとめ

本章では，2 章で提案した格子ボルツマン法による混相流解析手法に動的な AMR 法を導入し，複数 GPU 並列計算を実装した．木構造に基づく細分化を行い，木構造の末端であるリーフに $8 \times 8 \times 8$ の均一格子のブロックを割り当てることで，ブロック内の格子点のメモリアドレスを連続にし，GPU 実装における実行性能の向上を図った．格子解像度が変わる境界近傍に補間した値を記憶するためのゴーストブロックを配置することで，各ブロックが袖領域を持つよりもメモリ使用量を抑えつつ，補間のための条件分岐を削減した．動的な格子細分化の実装では，ゴーストブロックを利用したメモリ管理を行うことで，GPU 上で細分化と粗大化の処理を実行し，CPU と GPU 間の通信を抑えた．

複数 GPU 実装では，空間充填曲線であるモートン曲線を用いた領域分割を行い，各 GPU に割り当てられるブロック数を均一にした．袖領域の通信では，小領域表面の計算点に保存されている全方向の速度分布関数を通信する必要はないため，隣接の小領域に移流する分布関数のみをパッキングして隣接 GPU に転送することで，GPU 間の通信量を削減した．隣接領域のデータが必要な小領域表面のブロックと通信が必要でない内側のブロックに分け，別々のカーネル関数で非同期実行することによる演算と通信のオーバーラップ手法を提案した．固定された細分化格子を用いて単相の格子ボルツマン法の弱スケーリングを測定し，8GPU から 128GPU で 98% 以上の並列化効率を達成し，オーバーラップ法を用いることで実行性能を 1.37 倍に向上させた．強スケーリングでは，オーバーラップ法を導入ても並列化効率の低下が確認でき，これは主に並列数の増加に伴い通信時間の割合が増えることが原因であり，より高いスケーラビリティを達成するためには通信コストの削減が必要であることが明らかになった．一方，動的な格子細分化・粗大化を行う場合の弱スケーリングでは，GPU 数の増加に伴い格子細分化に要する時間が長くなり，並列化効率が低下することが確認でき，大規模 AMR 計算を高効率に実行するためには，木構造の処理や負荷分散に関しても MPI による並列化が必要であることが明らかになった．

AMR 法の検証として，界面移流の 3 次元計算と単一球形粒子の沈降計算を AMR 法を用いて行い，均一格子の結果とよく一致する結果が得られた．界面移流の計算では，格子解像度が $256 \times 256 \times 256$ 格子相当と $512 \times 512 \times 512$ 格子相当の計算を行い，界面に高解像度を適合する AMR 法では大規模計算になるほど格子点数の削減の効果が高いことが明らかになった．

東京工業大学の TSUBAME3.0 を用いて，AMR 法を導入した大規模な乱流計算と混相流計算を実行した．48 台の GPU を用いて球の直径に 2048 格子を割り当てた大規模な球周りの流れの計算を行い，レイノルズ数が 20 万から 30 万で抗力係数が急激に低下するドラッククライシスをキュムラントモデルの格子ボルツマン法で再現可能であることを示した．噴流によるピンポン玉の浮遊の計算では，ピンポン玉が水平方向にも揺れながら浮遊し，その振幅が速度に応じて大きくなる傾向があることが明らかになった．自由界面に高解像度格子を適合した大規模なダム崩壊計算を行い，格子解像度を細かくするとより細かい液滴の飛沫や壁に張り付く液滴が発生し，自由界面を含む激しい流れの計算ではより細かい格子が必要であることが確認できた．格子点数は界面の変形や飛沫により時間とともに増加する傾向があり，最も格子点数が多い時刻でも，格子点数を均一格子に対して 17% に削減できた．回転する板による水面の撹拌の計算を行い，提案手法が安定して自由界面流れと移動物体の計算可能であることを確認すると

ともに，板の回転速度が速く界面が乱れるほど格子点数が増える傾向があることが明らかになった．

格子ボルツマン法に動的な AMR 法を導入し，GPU スパコンで大規模計算を行った例は無く，AMR 法と GPU を用いることで大規模な混相流解析が可能になることを示した．また，提案した AMR の実装方法の一部は格子ボルツマン法以外のステンシル計算にも適用可能であり，その有用性は高い．

第 5 章

マルチフェーズフィールド法に基づくノード間通信削減のための動的負荷分散手法の開発

AMR 法の並列計算には，モートン曲線やヒルベルト曲線などの空間充填曲線 [38] による動的領域分割が用いられることが多く，4 章ではモートン曲線を用いて格子ボルツマン法の AMR 計算の複数 GPU 実装を行った．空間充填曲線を用いて分割を行えば，分割のコストも少なく，計算の負荷バランスを均一にすることができる．しかし，各領域が凸形状にならないために領域間通信が増え，GPU 計算のようにステンシル計算の時間が短い場合は，通信時間が大きなボトルネックとなる．演算と通信のオーバーラップにより通信時間を隠蔽することもできるが，強スケーリングにおいて並列数が増加すると演算時間よりも通信時間が長くなり，スケーリングが悪化することが明らかになった．大規模並列計算で高い実行性能と良好なスケーラビリティを実現するためには，通信コストを削減する必要がある．METIS[49] などのグラフ理論に基づく分割は各プロセスの負荷バランスを取りつつ，通信コストを最小化することができる．しかし，グラフ分割は分割のコストが大きいため，動的 AMR 法のように計算格子が時間変化する動的負荷分散への適用は難しい．計算の負荷バランスと領域間の通信低減を両立する分割コストの低い，新たな動的領域分割法の開発が必要である．

本研究では，材料分野において多結晶の組織形成過程の計算で使われているマルチフェーズフィールド法 [52][53] の動的領域分割への適用を試みる．マルチフェーズフィールド法では，各結晶粒は界面エネルギーを駆動力として成長し，結晶粒界面の表面積を最小化する．そこで，Fig.5.1 のように，左図の各結晶粒・各相を各プロセスで計算する領域と見なし，右図のブロック構造の細分化格子を分割する．計算負荷を均一化するような制限を与えた各結晶粒・各相の競合成長により，良好な負荷バランス保ちつつ表面積最小化の効果による通信コストの陰的な最適化を試みる．本章では，マルチフェーズフィールド法による細分化格子の分割法を開発し，AMR 法を用いた界面移流計算の複数 GPU コードに提案手法を導入し，従来の空間充填曲線との比較を行う．

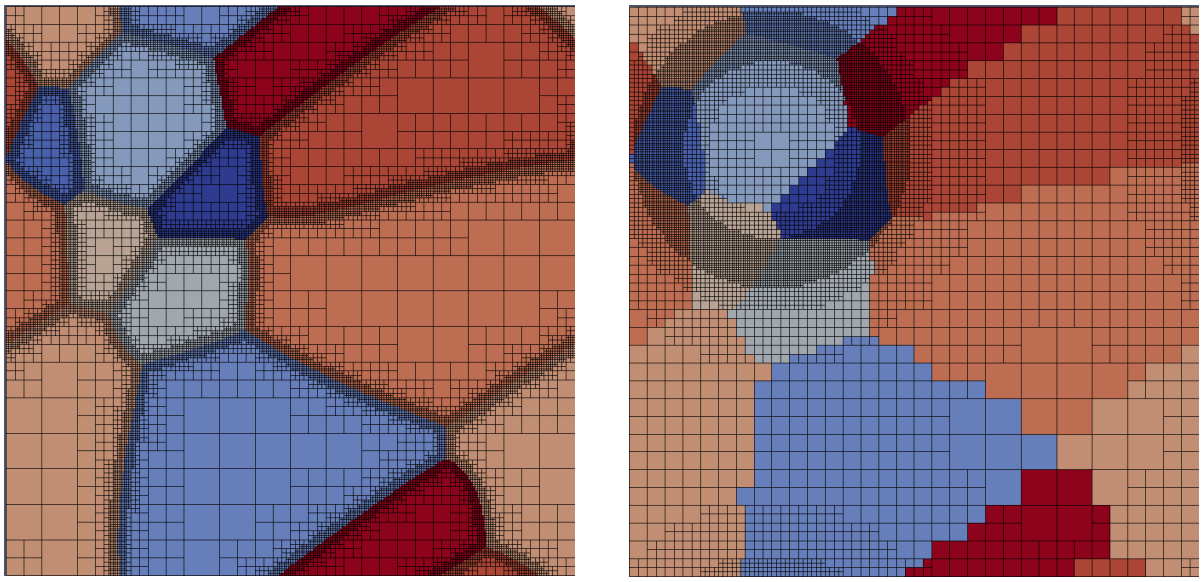


Fig.5.1: Domain partitioning of block-structured mesh based on the multi-phase-field method. The left panel shows the multi-phase-field profile, and the right panel shows domain decomposition of block-structured mesh. The color indicates the phase and corresponding sub-domains.

5.1　マルチフェーズフィールド法に基づく領域分割手法

5.1.1　マルチフェーズフィールド法

マルチフェーズフィールド法は多結晶粒の組織形成のシミュレーション手法であり，結晶粒界面のエネルギーの減少を界面駆動力として各結晶粒が成長していく．そのため，結晶粒と結晶粒の界面の表面積が最小となるような多結晶粒組織が最終的に形成される．異なる N 個の結晶粒から構成される組織をシミュレーションする場合，結晶粒の存在を表す秩序変数 $\phi_i \in \{1, 2, \cdots, N\}$ を定義する．結晶粒 i が存在する座標では $\phi_i = 1$，存在しない場合は $\phi_i = 0$ とする．結晶粒と結晶粒の界面は有限の厚みがあると仮定し，界面では $0 < \phi < 1$ の値を取る．界面では異なる複数の結晶粒のフェーズフィールド変数が共存するため，各格子点において，各結晶粒のフェーズフィールド変数は以下の拘束条件を満たす．

$$\sum_{i=1}^{N} \phi_i = 1 \tag{5.1}$$

任意座標で共存している結晶粒の数を n とし，一般的な粒成長で用いられるフェーズフィールド変数

ϕ_i の時間発展方程式は

$$\frac{\partial \phi_i}{\partial t} = -\frac{2}{n}\sum_{j=1}^{N}\omega_{ij}\left[\sum_{k=1}^{N}\left(\omega_{ik}-\omega_{jk}\right)\left(\phi_k + \frac{\delta^2}{\pi^2}\nabla^2\phi_k\right)\right] \tag{5.2}$$

である．ここで，δ は結晶粒の界面厚さ，ω_{ij} は同一の結晶粒を識別するパラメータであり，

$$\omega_{ij} = \begin{cases} 0, & i = j \\ 1, & i \neq j \end{cases} \tag{5.3}$$

と設定する．

2 次精度中心差分で空間微分を計算し，時間積分には 1 次オイラー法を用いる．格子幅と界面厚さはそれぞれ $\Delta x = 1$，$\delta = 6\Delta x$ と設定する．計算の安定性を確保するため，時間刻み幅は

$$\Delta t \leq \frac{\pi^2}{2D\delta^2}\Delta x^2 \tag{5.4}$$

を満たすように設定する．ここで，D は次元数である．

5.1.2　領域分割への適用

メモリ分散型の並列計算では，計算領域をプロセス数の小領域に分割し，一つの小領域を一つのプロセスに割り当てて計算する．小領域表面の計算セルは，隣接する小領域のデータが必要であるため，隣接する小領域の間で通信が発生する．高い実行性能を達成するためには，各プロセスの計算負荷を均一にしつつ，通信のオーバーヘッドを最小化することが重要である．小領域の表面積は通信量に大きく影響するため，各小領域の形状は通信量を削減するために重要な要因である．

マルチフェーズフィールド法では，結晶粒の表面積を最小化するように各結晶粒が成長するため，その結晶組織に基づき計算領域を分割することで通信コストを削減できると考えられる．一般的な粒成長では，Fig.5.2 のように，界面エネルギーが最小となるように大きい結晶粒が優先的に成長し，小さい結晶粒は収縮・消滅する．そのため，一般的なマルチフェーズフィールド法をそのまま領域分割に利用するのは難しい．

隣接する小領域との計算コストの差に基づき界面を駆動させる補正項を導入することで，結晶粒の数を指定したプロセス数に固定し，かつ各結晶粒で割り当てられた小領域の計算コストを均一化する．上原により提案された結晶粒の体積補正項 [120][121] を参考にし，領域分割のための時間発展方程式を

$$\frac{\partial \phi_i}{\partial t} = -\frac{2}{n}\sum_{j=1}^{N}\omega_{ij}\left[\sum_{k=1}^{N}\left(\omega_{ik}-\omega_{jk}\right)\left(\phi_k + \frac{\delta^2}{\pi^2}\nabla^2\phi_k\right) + k\left(C_i - C_j\right)\left|\nabla\phi_i\right|^2\left|\nabla\phi_j\right|^2\right] \tag{5.5}$$

とする．ここで，k はパラメータ，C_i は結晶粒 i で割り当てられた小領域の計算コストである．補正項により，結晶粒の界面は隣接している結晶粒との計算コストの差に基づいて駆動される．計算コストが小さい結晶粒を粗大化，計算コストが大きい結晶粒を縮小するように補正項が作用することで，各結晶粒で分割された領域の計算コストを均一化する．各小領域のブロック数 N^{block} に基づき，計算コスト C_i を

$$C_i = \frac{N_i^{\mathrm{block}}}{N_{\mathrm{ave}}^{\mathrm{block}}} \tag{5.6}$$

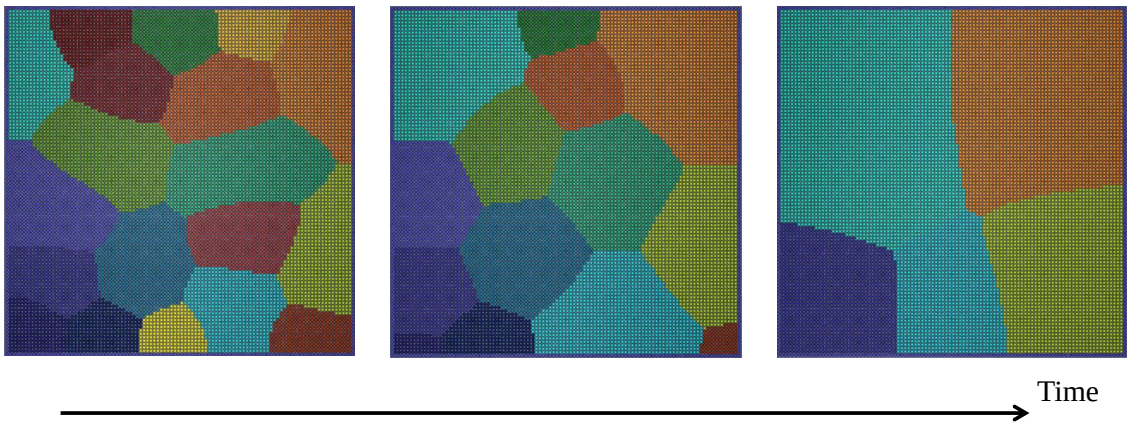


Fig.5.2: General grain growth process by the MPF method.

と見積もる．計算コスト C_i は各少領域のブロック数の平均値

$$N_{\mathrm{ave}}^{\mathrm{block}} = \frac{1}{N}\sum_{i=1}^{N} N_i^{\mathrm{block}} \tag{5.7}$$

で無次元化されており，全計算領域のブロック数によらず同一のパラメータ k を設定できる．

各ブロックはブロック内に存在しているフェーズフィールド変数 ϕ_i に基づいて，割り当てられる小領域が決定される．複数のフェーズが存在している場合は，最も体積が大きいフェーズに割り当てられる．

マルチフェーズフィールド法の計算では，計算領域に配置した N 個の初期シードから各結晶粒が成長する．一つの小領域に対して複数の初期シードを割り当てると，小領域が分裂する場合があるため，初期シードの数は分割したい領域の数（プロセス数）にする．動的領域分割の場合は，前の領域分割の結果を初期値とすることで，収束性を改善することができる．

初期化の後，以下の 3 ステップの処理を繰り返し行うことで，領域分割を行う．

1. マルチフェーズフィールド法の時間発展を 1 ステップ計算する．
2. 結晶粒組織に基づき計算領域を分割する．
3. 各小領域の計算コスト C_i を見積もる．

ロードバランスの誤差が設定した閾値以下になる，または設定した反復回数に達した場合，繰り返し処理を終了し，最終的な結果を領域分割とする．

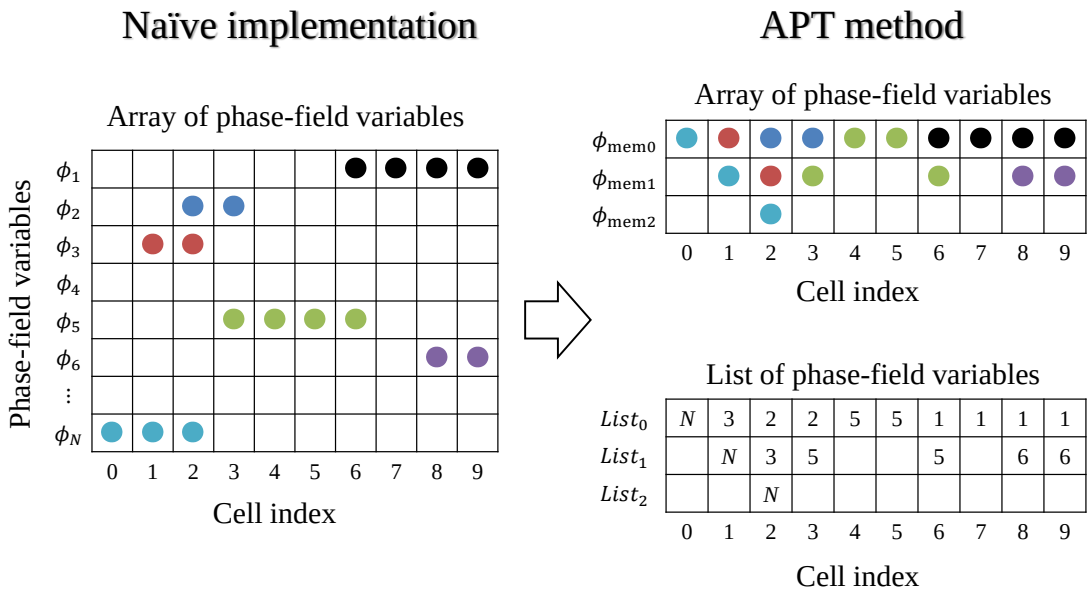


Fig.5.3: Illustration of the Active Parameter Tracking (APT) method (right). Circles indicate active phase-field variables ($\phi_i > \epsilon$). In this case, the maximum number n_a of active variables per a cell is three.

5.1.3 Active Parameter Tracking

マルチフェーズフィールド法では，N 個の結晶粒からなる組織を計算する場合，N 個のフェーズフィールド変数 $\phi_i \in \{1, 2, \cdots, N\}$ の時間発展を計算する必要がある．例えば，計算領域を 1000 ノードに領域分割する場合は 1000 個の変数 ϕ が必要になり，全計算セルに全ての変数 ϕ を保持するとメモリ使用量が膨大となる．

マルチフェーズフィールド法の領域分割に必要なメモリ使用量を削減するために，Fig.5.3 に示す APT（Active Parameter Tracking）法 [53] を導入する．APT 法では，各計算セルに存在している微小な閾値 ϵ 以上の変数 $\phi > \epsilon$ のみを動的にメモリに保存する．各セルに存在しうる最大の変数の数 n_a を設定し（Fig.5.3 では $n_a = 3$），セルに存在しているフェーズフィールド変数を配列に詰めて保存する．各変数を識別するために，各セルに存在している変数のリストを作成する．

APT 法では，各セルに登録されているフェーズフィールド変数に対してのみ時間発展を計算する．更新した変数が ϵ よりも小さい場合は，リストから削除し，その変数の値を 0 とする．一方，閾値 ϵ よりも大きな値に更新された場合は，その計算セルおよび隣接セルのリストに変数を登録する．

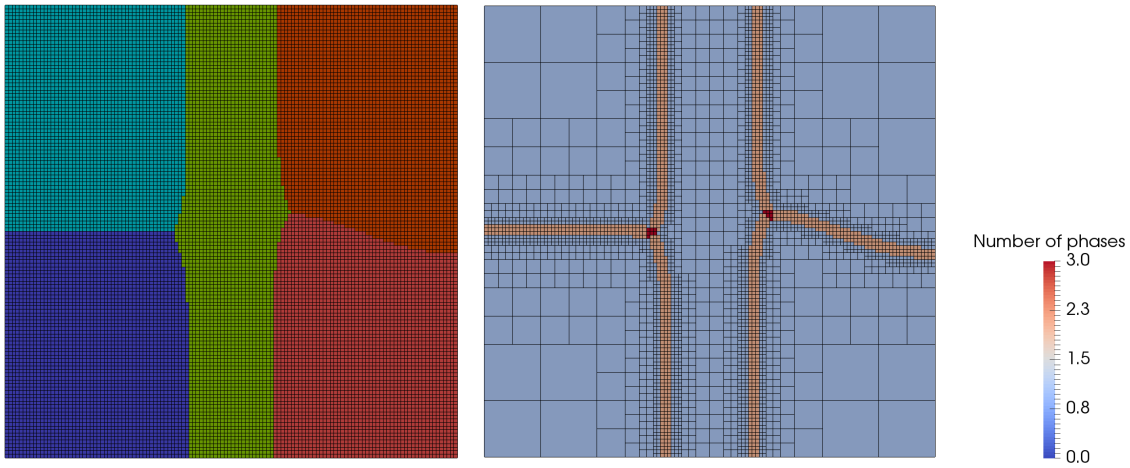


Fig.5.4: Relation of the MPF result with the domain partitioning. A computational domain shown in the left figure with a uniform mesh is partitioned into 5 sub-domains. The right figure illustrates the cell-based AMR mesh for solving the MPF equation and the color indicates the number of active phases registered in the APT list (one active phase in light blue, two active phases in orange and three active phases in red).

5.1.4　セルベース AMR 法

大規模計算では，最小ブロックと同程度の解像度のセルを用いても，マルチフェーズフィールド法の計算が大規模になってしまう．そこで，マルチフェーズフィールド方程式を計算するためだけの細分化格子を用いて，領域分割の計算を効率的に実行する．木構造に基づき格子細分化を行い，木構造の末端であるリーフに一つの計算セルを割り当てるセルベース AMR 法を用いる．Fig.5.4 は 2 次元均一格子を 5 つの小領域に分割した例であり，右図はセルベース AMR 法を導入したマルチフェーズフィールド法の結果で，色は APT 法のリストに登録されている変数 ϕ の数を示す．2 つ以上の変数が存在している領域とその近傍にのみ高解像度の計算セルを割り当てることで，計算精度を維持したまま格子点数を削減する．領域分割の計算は各 MPI プロセスが冗長に実行する．

5.2　細分化格子の静的な領域分割

この節では，提案するマルチフェーズフィールド法により細分化格子の静的な分割を行い，各結晶粒の初期シードの配置，体積補正項のパラメータ k，格子解像度の影響を評価する．分割対象の細分化格子を

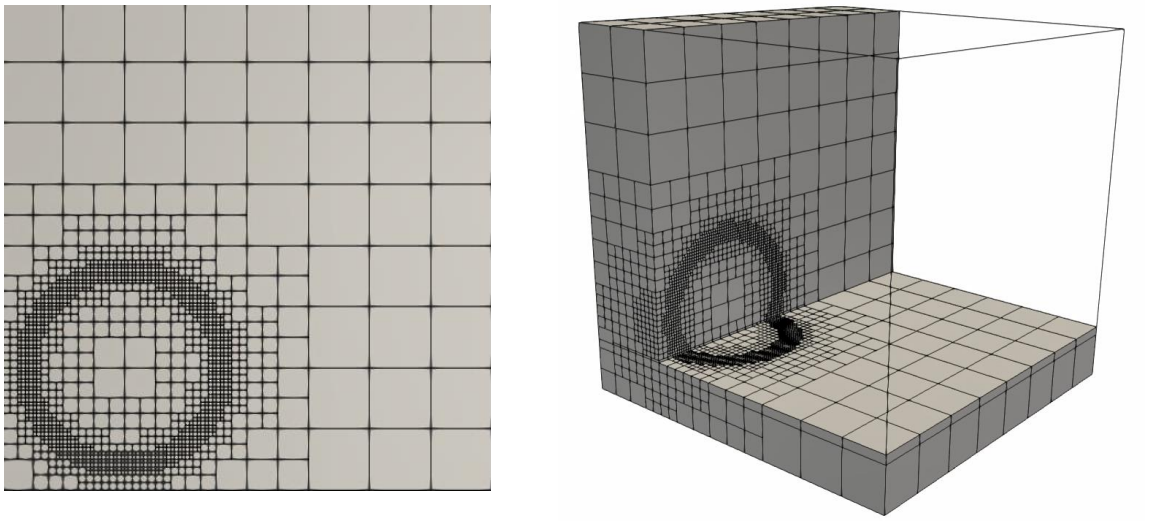


Fig.5.5: The two- (left) and three-dimensional (right) block-structured mesh for static domain partitioning. The mesh indicates the blocks.

Fig.5.5 に示す．2 次元領域分割の評価では，計算領域 $[0,1]\times[0,1]$ の左下 $[0.25,0.25]$ に配置された半径 0.2 の円表面に高解像度格子を適合させた格子を用いる．最も粗い解像度は 8×8 の均一ブロック相当であり，最も細かいものは 128×128 相当である．3 次元領域分割の評価では，計算領域 $[0,1]\times[0,1]\times[0,1]$ の左下 $[0.25,0.25,0.25]$ に配置された半径 0.2 の球の表面に高解像度格子を適合させた格子を用いる．最も粗い解像度は $8\times8\times8$ の均一ブロック相当であり，最も細かいものは $128\times128\times128$ 相当である．

5.2.1　初期シードの配置の影響

マルチフェーズフィールド法による領域分割では，計算領域に配置された分割する小領域の数（プロセス数）と同じ数の初期シードから結晶粒の成長過程を計算する．この説では，最終的に得られる領域分割への初期シードの配置の影響を調べる．

初期シードの配置を変え，Fig.5.5 の 2 次元細分化格子を 36 個の小領域に分割する．空間充填曲線（モートン曲線とヒルベルト曲線）で分割された領域の中心に初期シードを配置したもの，等間隔に 36 個のシード（6×6）を配置したものを比較する．マルチフェーズフィールド法の解像度を 128×128 セル相当とし，10,000 タイプステップの計算を行う．境界条件にはノイマン境界条件を課す．体積補正項のパラメータは $k=400$ である．

初期シードの配置と，10,000 ステップ後のマルチフェーズフィールド法のプロファイルを Fig.5.6 に示す．どの条件においても，高解像度が割り当てられている左下の円近傍に，小さい結晶粒が形成されていることが確認できる．また，結晶粒の形状は表面積が小さい凸形状をしているのが確認できる．初

期シードの配置により，結晶粒の配置は異なっていることがわかる．

領域分割の性能を評価するため，ロードバランスの誤差 Er と通信ブロック Cb を

$$Er = \frac{N_{\mathrm{max}}^{\mathrm{block}} - N_{\mathrm{ave}}^{\mathrm{block}}}{N_{\mathrm{ave}}^{\mathrm{block}}} \tag{5.8}$$

$$Cb = \frac{\sum_{i=1}^{N} N_i^{\mathrm{boundary}}}{\sum_{i=1}^{N} N_i^{\mathrm{block}}} \tag{5.9}$$

と見積もる．ここで，$N_{\mathrm{max}}^{\mathrm{block}}$ はブロック数が最大の小領域におけるブロック数，N^{boundary} は小領域の表面にあり小領域間の通信が必要なブロックの数である．通信ブロック Cb は通信が必要なブロック数と全ブロック数の比であり，小領域の表面積に大きく影響される．通信ブロック Cb は通信コストおよび通信時間に影響する要素であり，値が小さいほど通信は最適化されている．

2 次元領域分割におけるロードバランス誤差 Er と通信ブロック Cb の時刻歴を Fig.5.7 に示す．空間充填曲線を用いて初期シードを配置した条件では，1,000 ステップの計算でロードバランス誤差 Er が 0.1 以下になり，定常状態に達している．一方，初期シードを 6×6 に均一に配置した条件では，定常状態に達するまでに空間充填曲線を用いた場合よりも多くのステップ数が必要である．定常状態におけるロードバランス誤差は初期シードに影響されず，どの条件でも同程度の値となっている．定常状態における通信ブロック Cb はモートン曲線とヒルベルト曲線で同程度となっている．均一の初期シードを用いた場合は，空間充填曲線よりも大きい値を取っている．

Fig.5.5 に示す 3 次元細分化格子を 64 個の小領域に分割し，初期シードの配置の影響を評価する．小領域の数を 64 個とし，2 次元領域分割と同様に空間充填曲線に基づく初期配置と均一な初期配置（$4 \times 4 \times 4$）の条件で分割を行う．マルチフェーズフィールド法の解像度は $128 \times 128 \times 128$ セル相当であり，10,000 ステップの計算を実行する．パラメータは $k = 400$ と設定した．ロードバランス誤差 Er と通信ブロック Cb の時刻歴を Fig.5.8 に示す．2 次元領域分割の結果と同様に，均一な初期配置よりも空間充填曲線を用いた初期配置のほうが定常状態に少ないステップ数で達している．定常状態における負荷バランス誤差と通信ブロックはどの初期シード配置でも大きな違いはない．

収束性が良いため，ヒルベルト曲線に基づく初期シード配置を用いて以下の領域分割の計算を行う．

5.2.2　パラメータ k の影響

体積補正項のパラメータ k が領域分割の結果に与える影響を調べるため，パラメータを $k \in \{100, 200, 400, 800, 1600\}$ とし，Fig.5.5 の格子を 16 個の小領域に分割する．2 次元領域分割ではマルチフェーズフィールド法の解像度を 128×128 セル相当とし，3 次元領域分割では $128 \times 128 \times 128$ セル相当とする．

2 次元領域分割でのロードバランス誤差 Er と通信ブロック Cb の時刻歴を Fig.5.9 に，3 次元領域分割での結果を Fig.5.10 に示す．パラメータ k を大きくするほど，ロードバランス誤差の収束性が良くなり，定常状態における値も小さくなることが確認できる．一方，定常状態における通信ブロックの値は

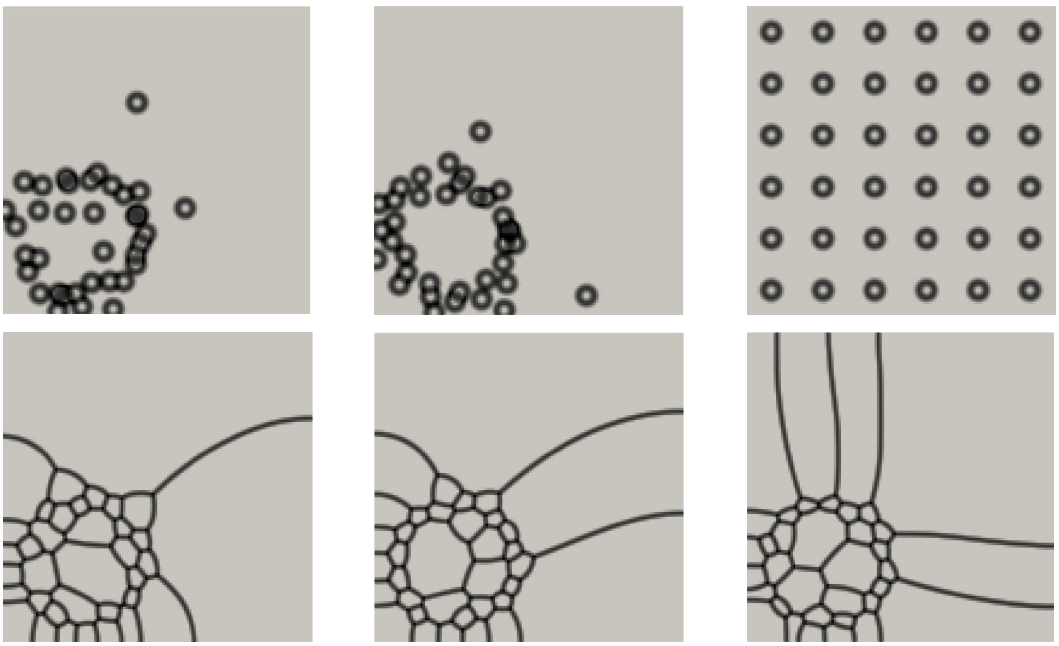


Fig.5.6: Static domain partitioning for two-dimensional refined mesh by MPF method with different initial seed setting. The black lines indicate the phase interface. We compare initial seed arrangement using the Morton curve (left), the Hilbert curve (center) and uniform (right). The upper panels show the initial conditions. The lower panels show the results after 10,000 iterations.

どのパラメータにおいてもほぼ同じであり，通信最適化にはパラメータ k はあまり影響しないことが確認できた．

5.2.3　格子解像度の影響

マルチフェーズフィールド法の格子解像度が領域分割の性能に与える影響を評価するため，マルチフェーズフィールド法の解像度を 64×64，128×128，256×256 と設定し，Fig.5.5 の 2 次元細分化格子（ブロックの最大解像度は 128×128 相当）を 16 個の小領域に分割する．パラメータは $k = 800$ である．

5000 ステップ後の結晶組織の様子とそれに対応する領域分割の結果を Fig.5.11 に示す．ブロックの解像度よりもマルチフェーズフィールド法の解像度が粗い 64×64 の条件でも，計算領域を凸形状の小領域に分割できている．なお，解像度を 32×32 とした場合は，指定した個数の小領域に分割することができなかった．

ロードバランス誤差 Er と通信ブロック Cb の時刻歴を Fig.5.12 に示す．マルチフェーズフィールド法の解像度が高いほどロードバランス誤差を小さくできることが確認できる．しかし，格子が粗いほど早く定常状態に達しており，これは解像度を高くするほど時間刻み幅を小さく設定する必要があるためだと考えられる．

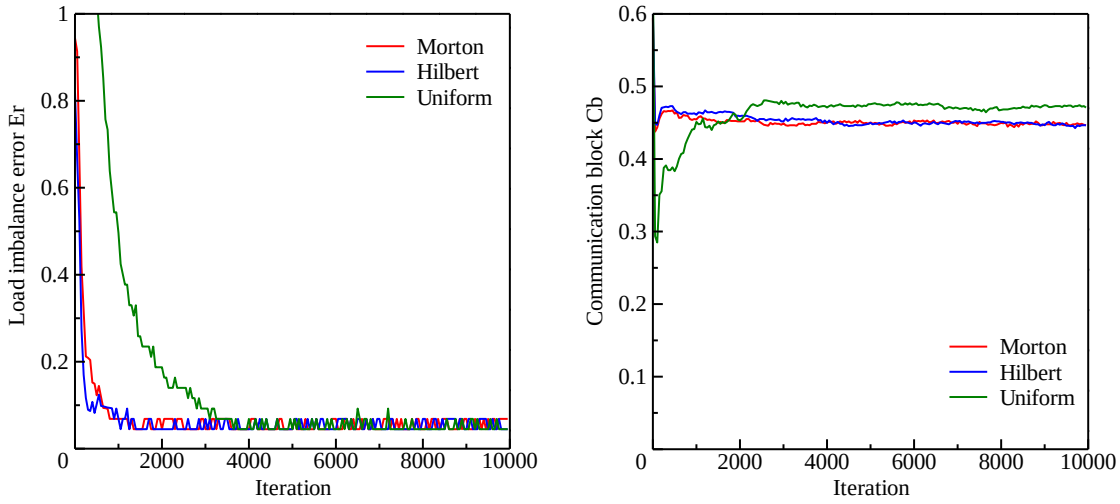


Fig.5.7: The time histories of the load imbalance error (left) and the communication block (right) in two-dimensional static domain partitioning with different initial seed placement.

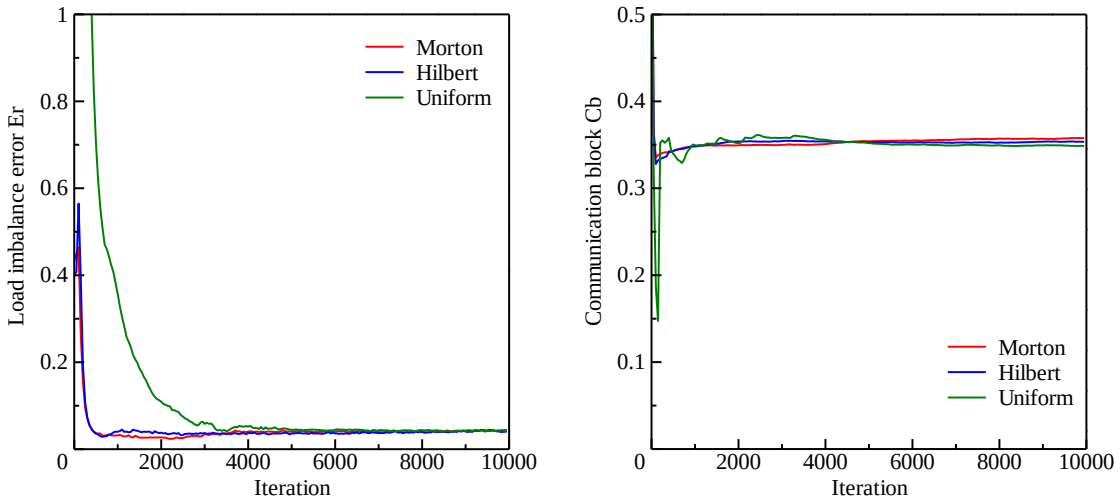


Fig.5.8: The time histories of the load imbalance error (left) and the communication block (right) in three-dimensional static domain partitioning with different initial seed placement.

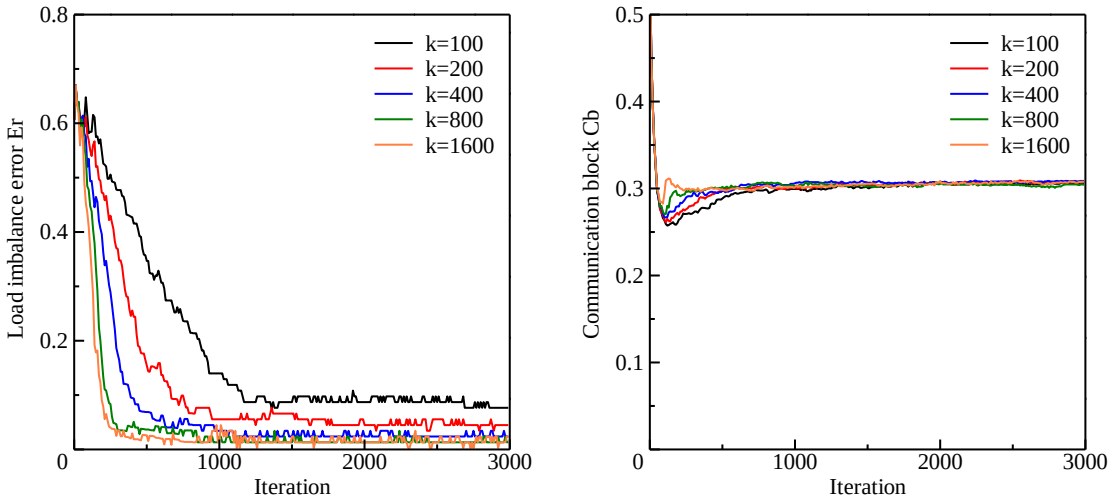


Fig.5.9: Time history of the load imbalance error (left) and the communication block (right) in two-dimensional with different parameter k.

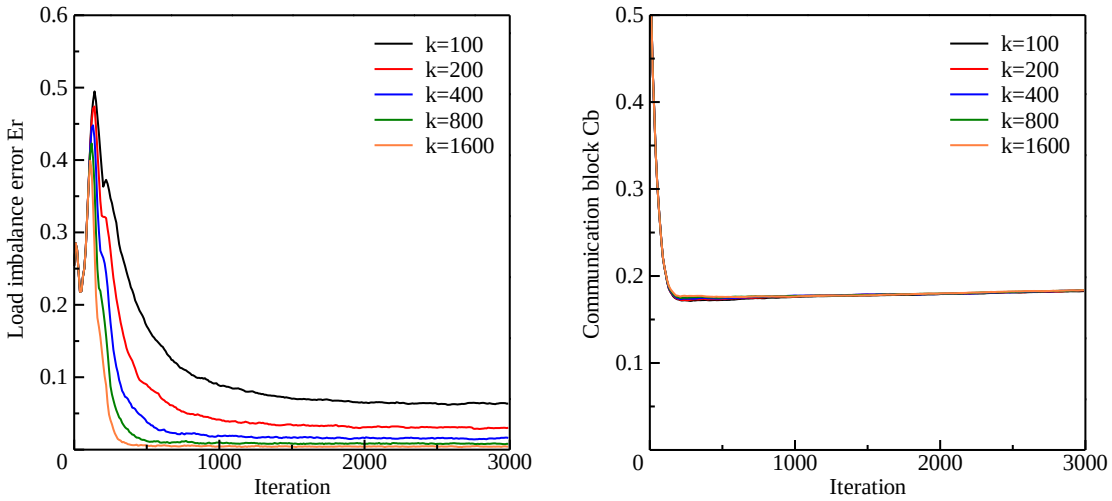


Fig.5.10: Time history of the load imbalance error (left) and the communication block (right) in three-dimensional with different parameter k.

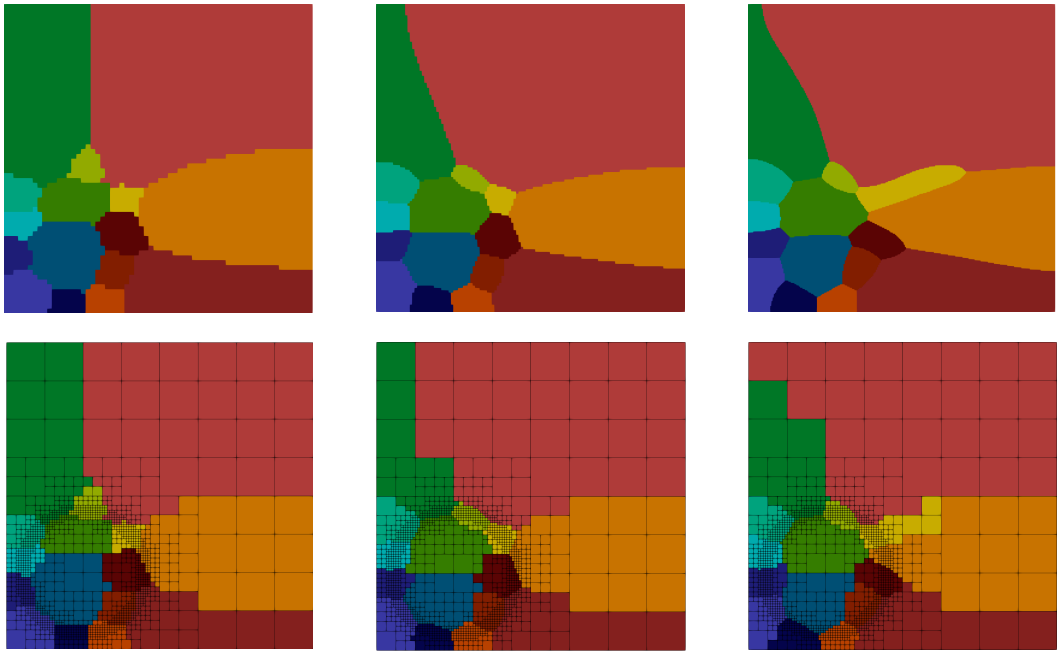


Fig.5.11: The results of MPF simulations (upper) and partition sub-domains (lower) obtained by different MPF resolutions 64×64 (left), 128×128 (center) and 256×256 (right). The mesh of lower panels indicates the blocks of partitioned block-structured mesh. The color indicates the phase index and the corresponding sub-domain.

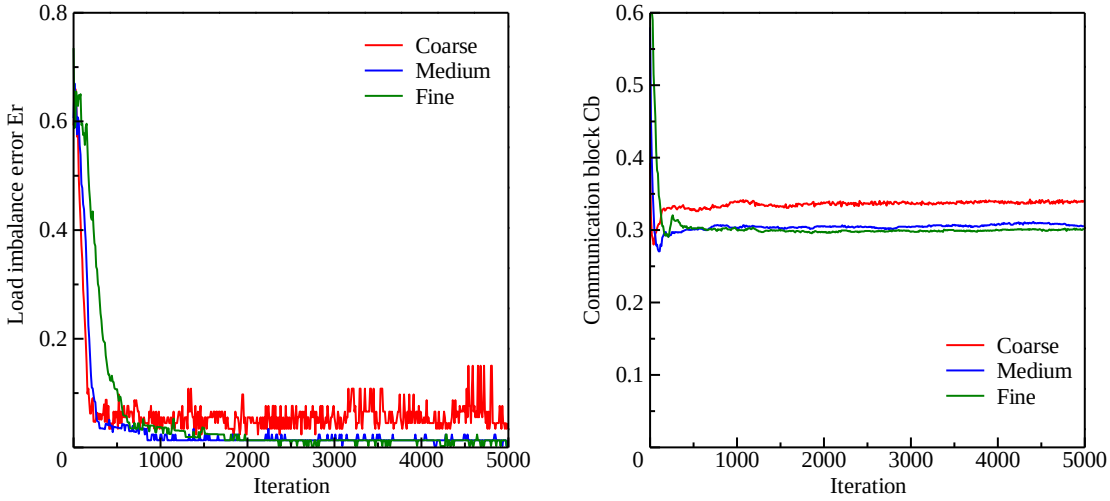


Fig.5.12: Time history of the load imbalance error (left) and the communication block (right) when the MPF resolution is changed.

5.3 AMR 計算の動的領域分割での評価

この節では，提案するマルチフェーズフィールド法による領域分割手法を，AMR 法を導入した界面捕獲の複数 GPU 計算に実装し，領域分割の性能を空間充填曲線であるモートン曲線と比較する．

実行性能評価は東京工業大学の GPU スパコン TSUBAME3.0 を用いて行う．TSUBAME3.0 は 540 ノードから構成され，ひとつのノードには 2 ソケットの CPU（14-core Intel Xeon E5-2680 V4 2.4 GHz）と 4 台の GPUU（NVIDIA Tesla P100 with NVLink）から構成されるノード間は 50 GB/s の Intel Omni-Path で接続されている．

本節の実行性能測定に用いる界面捕獲法の AMR コードでは，1 ノードあたり 4 つの MPI プロセスが割り当てられ，1 つの MPI プロセスには 1 台の GPU と 7 個の OpenMP のスレッドが割り当てられる．

5.3.1 界面捕獲手法

気液二相流などの混相流の計算では，相界面の挙動を捉えるための界面捕獲手法が必要となる．格子上で大変形する界面を高精度かつ安定に捉えるために，界面にある程度の厚みをもたせた拡散界面を表現する Diffuse interface model（DIM）[40] が注目されている．DIM の計算では，相界面近傍に高解像度の格子が必要であり，AMR 法の導入により計算コストの削減が期待できる．

拡散界面の厚さを均一に保ち，高精度に界面の移流計算を行える DIM として，保存型 Allen-Cahn 方程式を解く手法 [41] があり，界面プロファイル ϕ^{AC} の時間発展方程式は

$$\frac{\partial \phi^{\mathrm{AC}}}{\partial t} + \nabla \cdot \left(\boldsymbol{u} \phi^{\mathrm{AC}} \right) = \bar{\gamma} \epsilon \nabla^2 \phi^{\mathrm{AC}} - \bar{\gamma} \nabla \left[\phi^{\mathrm{AC}} \left(1 - \phi^{\mathrm{AC}} \right) \frac{\nabla \phi^{\mathrm{AC}}}{|\nabla \phi^{\mathrm{AC}}|} \right] \tag{5.10}$$

である．ここで，$\boldsymbol{u}$ は速度場，$\bar{\gamma}$ と ϵ はパラメータである．左辺が ϕ^{AC} の保存方程式であり，右辺の第一項が界面の拡散項，第二項が界面の逆拡散項である．パラメータ $\bar{\gamma}$ と ϵ は

$$\epsilon = \frac{\delta^{\mathrm{AC}}}{2} \frac{\tanh\left(1 - 2\lambda\right)}{2} \tag{5.11}$$

$$\bar{\gamma} = \frac{3}{2} M_\phi \left| \boldsymbol{u} \right|_{\max} \gamma \delta^{\mathrm{AC}} \tanh\left(1 - 2\lambda\right) \tag{5.12}$$

と設定する．ここで，δ^{AC} は界面厚さ，M_ϕ はモビリティ，γ は表面張力，λ は界面厚さを決めるパラメータである．$\delta^{\mathrm{AC}} = 3\Delta x^{\mathrm{AC}}, M_\phi = 500$，$\gamma = 72.8 \times 10^{-3}$，$\lambda = 0.1$ と設定する．

本節では，保存型 Allen-Cahn 方程式を有限体積法で解く．時間積分には 2 段 2 次ルンゲクッタ法を用いる．空間の離散化には，左辺の保存方程式には 3 次 MUSCLE 法，右辺の拡散・逆拡散項には 2 次中心差分を用いる．

計算格子の生成は木構造に基づいて行い，木構造で細分化した領域に均一格子のパッチを割り当てるブロック構造格子を用いる．ブロックあたりの格子点数を 2 次元計算では 16×16，3 次元計算では $16 \times 16 \times 16$ とする．界面近傍に高解像度格子を割り当てる．

5.3.2　2 次元動的領域分割

一様速度場における移流計算

提案するマルチフェーズフィールド法による負荷分散を用いて，一様速度場における 2 次元移流問題の AMR 計算を行う．一辺の大きさが 1 の正方形の計算領域を設定し，位置 $[0.25, 0.25]$ に半径 0.2 の円形プロファイルを配置し，速度 $u(x, y, t) = v(x, y, t) = 1$ で移流させる．

各ブロックに 16×16 セルを割り当てたブロック構造格子を用い，高解像度格子を界面に適合させる．最も粗い格子は 128×128 セル相当であり，最も細かい格子は 2048×2048 セル相当である．初期時刻におけるブロックの数は 2305 個であり，計算セルは 590,080 である．移流計算は $t = 0.5$ まで計算し，円形プロファイルは位置 $[0.75, 0.75]$ まで移動する．計算領域の境界条件にはノイマン境界条件を課し，計算には 16 台の GPU を用いる．移流計算 500 ステップに一度の頻度で格子生成と領域分割を行い，これはプロファイルがおおよそ 1 ブロック移動したら格子生成と負荷分散を行う頻度である．

マルチフェーズフィールド法の格子解像度は 128×128 セル相当である．体積補正項のパラメータを $k = 800$ と設定する．初期時刻における領域分割では，マルチフェーズフィールド法の時間発展方程式を 2000 ステップ計算する．動的な領域分割では，前の時刻における領域分割の結果を初期状態とし，ロードバランスの誤差 Er が 0.05 以下になるまで反復計算する．

領域分割の結果を Fig.5.13 に示す．凸形状の小領域で計算領域を分割できていることが確認できる．

ロードバランスの誤差 Er，通信ブロック Cb，最も多くの小領域と隣接してる小領域の隣接小領域の数，領域分割の変更により発生するデータ移行のコストの時刻歴を Fig.5.14 に示す．マルチフェーズフィールド法による領域分割の性能を評価するために，空間充填曲線であるモートン曲線と結果を比較する．どの時刻においても，提案手法はロードバランスの誤差（Fig.5.14 の左上）を 5% 以下（設定した領域分割の反復を止める閾値以下）にキープすることができている．

マルチフェーズフィール法を用いた場合の通信ブロック（Fig.5.14 の右上）はモートン曲線を用いた場合よりも小さく，通信コストを削減できている．時間平均した値は，マルチフェーズフィールド法では 0.309，モートン曲線では 0.397 であり，提案手法は通信ブロックの割合をモートン曲線の 77.8% に削減できた．

通信時間は通信量だけでなく，通信相手の数にも影響される場合がある [122]．各小領域に隣接している小領域の数（通信相手の数）を数え，最も隣接数が多い小領域の隣接数を Fig.5.14 の左下の図に示す．マルチフェーズフィールド法の領域分割では，隣接の小領域の数を最小化する効果は導入していないが，隣接している小領域の数の最大値はモートン曲線と同程度かそれ以下であった．

AMR 計算の負荷分散では，領域分割が変更されることに伴い，ブロックのデータを他のプロセスに移行する必要があり，データ移行のための通信がオーバーヘッドにある場合がある．特に，ひとつの計算セルが多くの従属変数を持つ場合は，データ移行の通信量が大きくなる．Fig.5.14 の右下は，全ブロックにおける他のプロセスに通信されたブロック数の割合を示す．提案手法では，新しい格子の分割の際に前の時刻における領域分割の情報を利用しているため，モートン曲線よりもデータ移行の通信コストを削減できている．

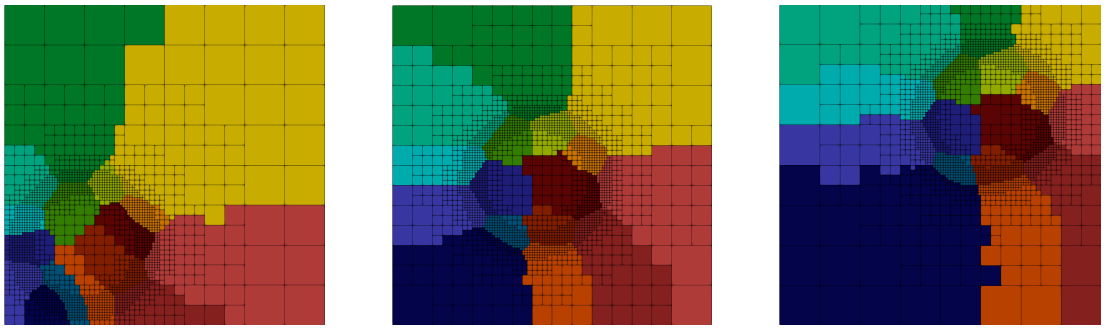


Fig.5.13: Results of dynamic domain partitioning for adaptively refined mesh in two-dimensional at 3 physical time instants t=0.0,0.2 and 0.4 s (from the left panel to the right panel). The mesh indicates the blocks.

ロードバランスの誤差を 5% 以下にするために必要だった反復回数を Fig.5.15 に示す．時間平均した反復回数は 103 回であった．本研究の実装では，マルチフェーズフィールド法による領域分割の計算を各プロセスが 7 個の OpenMP スレッドを用いて冗長に計算していて，一回の領域分割にかかった時間は 0.15 秒であり，全計算時間に対して約 7.4% である．

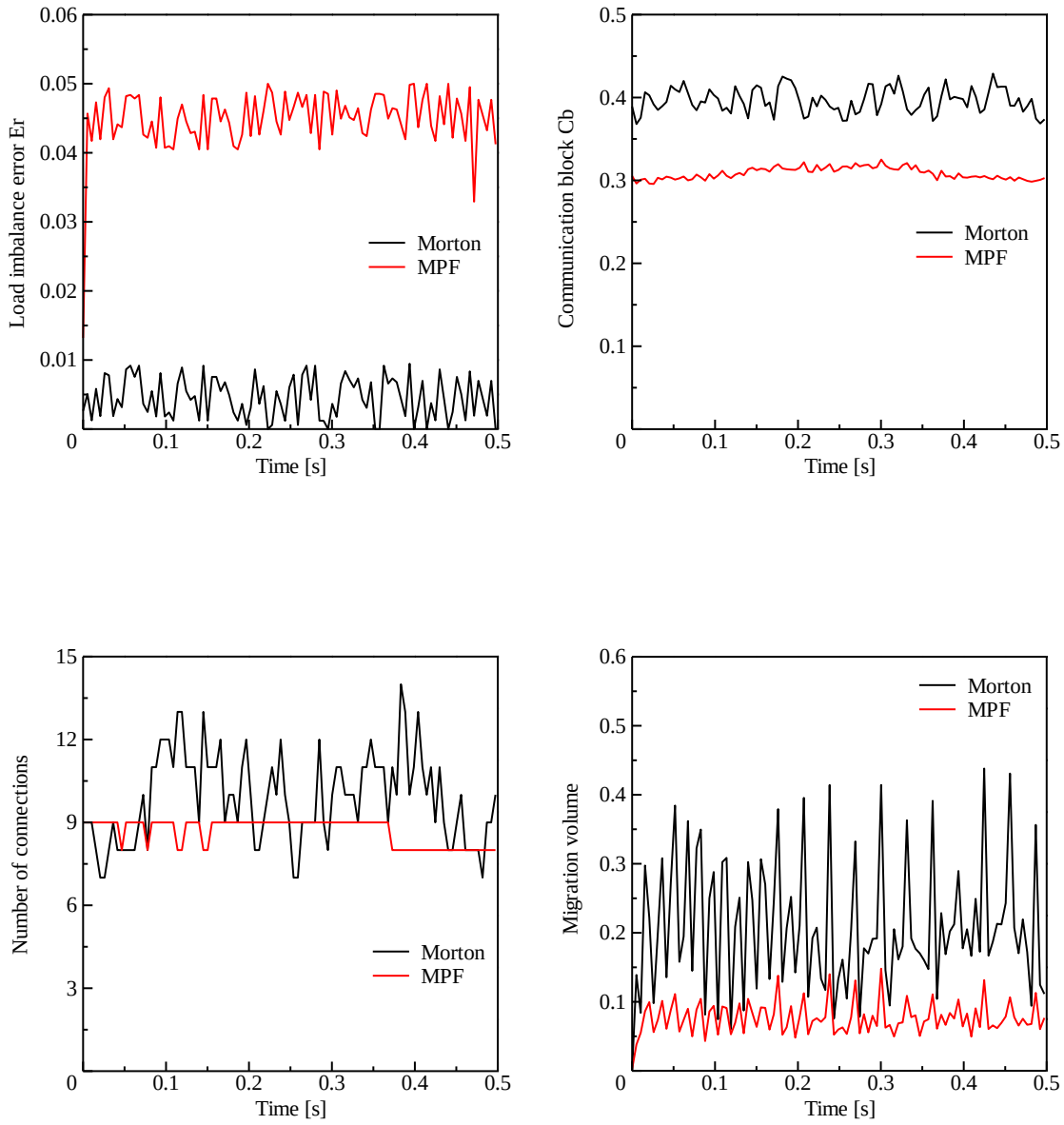


Fig.5.14: Time history of the load imbalance error (upper left), the communication block (upper right), the maximum number of connections (lower left) and the data migration cost (lower right) of two-dimensional advection in uniform velocity field. The migration cost is defined as the number of blocks allocated to different processes normalized by the number of total blocks.

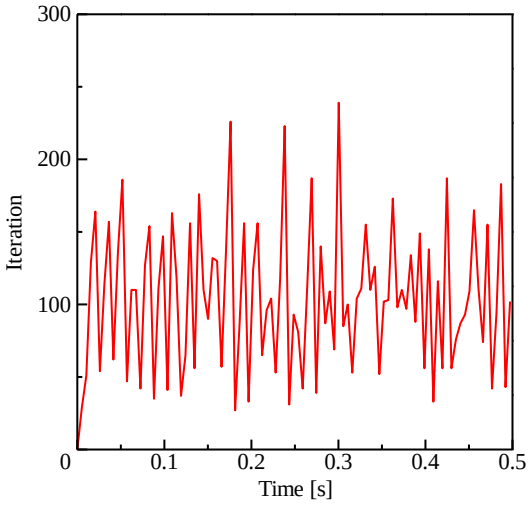


Fig.5.15: Time history of the number of iterations of the MPF method in two-dimensional dynamic domain partitioning.

渦速度場における移流計算

界面捕獲手法のベンチマークである Single vortex 問題 [123] に対し AMR 法を適用した 2 次元の保存型 Allen-Cahn 方程式で解く．一辺の大きさが 1 の正方形の計算領域を設定し，半径が 0.2 の円形プロファイルを $[0.25, 0.25]$ 配置する．円形プロファイルは以下の速度場で移流する．

$$u(x,y,t) = 2\sin^2(\pi x)\sin(\pi y)\cos(\pi y)\cos\left(\frac{\pi t}{T}\right) \tag{5.13}$$

$$v(x,y,t) = -2\sin(\pi x)\cos(\pi x)\sin^2(\pi y)\cos\left(\frac{\pi t}{T}\right) \tag{5.14}$$

ここで，T は周期であり，$T = 8.0$ と設定する．

界面捕獲手法の計算に用いる AMR 格子には，各ブロックに 16×16 の均一格子が割り当てられる．最も粗いブロックの解像度は 16×16 相当であり，128×128 セルに対応する．最も細かいブロックの解像度は 128×128 相当であり，2048×2048 セルに対応する．高解像度の格子は界面近傍に割り当てられる．初期状態において，全ブロック数は 2305 で，計算セル数は 590,080 である．16GPU（Tesla P100）を用いて周期の半分にあたる $t = 4.0$ まで計算する．計算領域の境界条件にはノイマン条件を用いる．AMR 格子の細分化と粗大化の処理は 500 ステップに一度行う．これはクーラン数を 0.03 と設定した場合に，プロファイルがおおよそ 1 ブロック移動したら格子生成と負荷分散を行う頻度である．

領域分割に用いるマルチフェーズフィールド法の計算に 128×128 セルを用い，各 MPF プロセスが領域分割の計算を冗長に行う．パラメータは $k = 800$ と設定し，初期状態における領域分割では，マルチフェーズフィールド法の反復計算を 2000 回行う．動的領域分割では，負荷バランスの誤差が 5% 以下になるまで反復計算を行う．

2 次元 Single vortex 問題における動的領域分割の結果を Fig.5.16 に示す．どの時刻においても，MPF 法で分割された小領域は凸形状であることが確認できる．界面プロファイルの変化が激しい $t = 2.0$ までは，小領域の形状と位置も大きく変化している．界面プロファイルの変化が緩やかな $t = 2.0$ 以降では，小領域の形状の変化は小さく，時間が経過しても似た形状を保っている．小領域の形状と位置の時間変化が小さいことにより，ブロックの割り当てられるプロセスが変わることにより発生するデータマイグレーションのコストを小さくすることができる．

2 次元 Single vortex 問題における負荷バランスの誤差と通信ブロック，小領域の接続数，データ移行のコストの時刻歴を Fig.5.17 に示す．MPF 法の領域分割の性能を比較するため，モートン曲線を用いた場合の結果（黒線）と比較する．MPF 法は設定した負荷バランス誤差 5% 以内をキープすることが出来ている．MPF 法による領域分割を用いた場合の通信ブロックの割合は，全時刻に置いてモートン曲線を用いた場合よりも小さく，MPF 法の領域分割は通信コストを削減できている．なお，どちらの結果においても計算の進行に伴い通信ブロックの割合が低下しているのは，界面プロファイルの形状の変化によりブロック数が増加したためである．時間平均した値はマルチフェーズフィールドでは 0.189，モートン曲線では 0.253 であり，マルチフェーズフィールド法は通信ブロックをモートン曲線の 74.7% に削

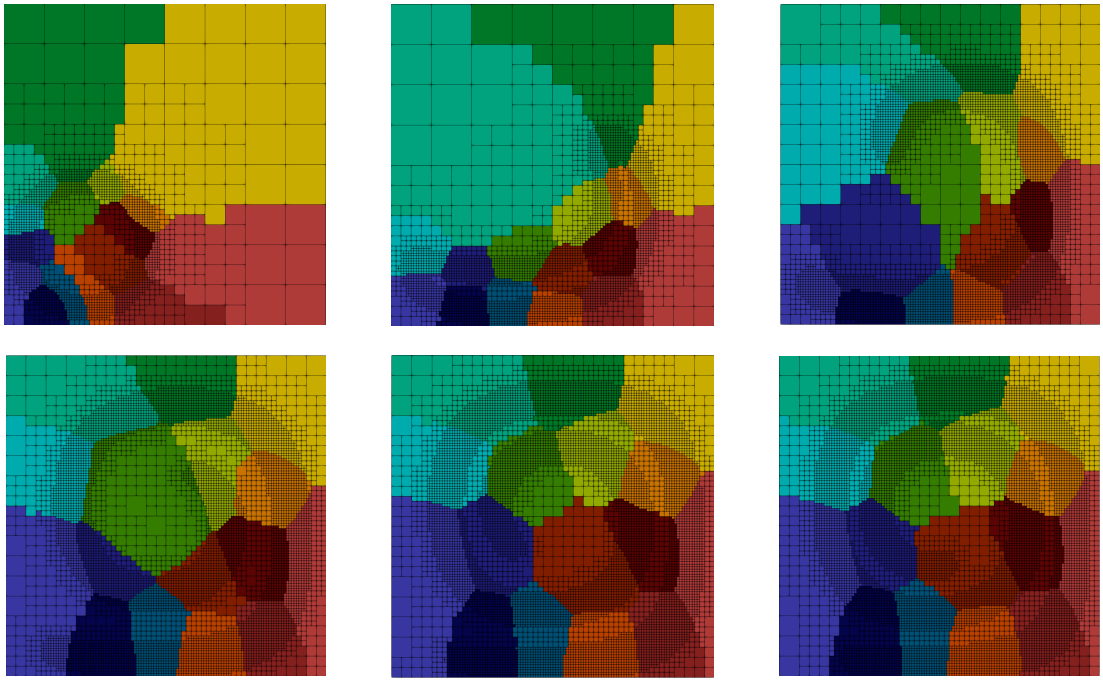


Fig.5.16: Dynamic domain decomposition with 16 sub-domains for adaptively refined mesh in a two-dimensional single vortex simulation at 6 physical time instants t=0.0, 0.5, 1.0, 2.0, 3.0 and 4.0 s (from the upper left panel to the lower right panel). The mesh indicates the blocks.

減できた．小領域の接続数の最大値は，マルチフェーズフィールド法とモートン曲線に大きな差はない．渦速度場における移流問題においても，マルチフェーズフィールド法は領域分割の変更に伴うデータ移行のコストを削減できている．どちらの手法も界面の移動が小さい $t = 2.0$ 移行では，データ移行のコストが小さく，マルチフェーズフィールド法では約 1%，モートン曲線では約 2% である．

Single Vortex 問題における動的領域分割において，一回の領域分割に要した MPF 法の反復回数を Fig.5.18 に示す．動的領域分割では，前の分割結果を初期値として新しい細分化格子の分割を行うことで，100 回程度の反復計算で十分に最適化された分割結果を得られた．特に，界面プロファイルの変化が緩やかな $t = 2.0$ 以降では，数十回の反復で分割を行えている場合が多い．一回の領域分割に要した時間は平均で 0.06 秒であり，全計算時間に対して領域分割に要した時間は約 3.2% であった．

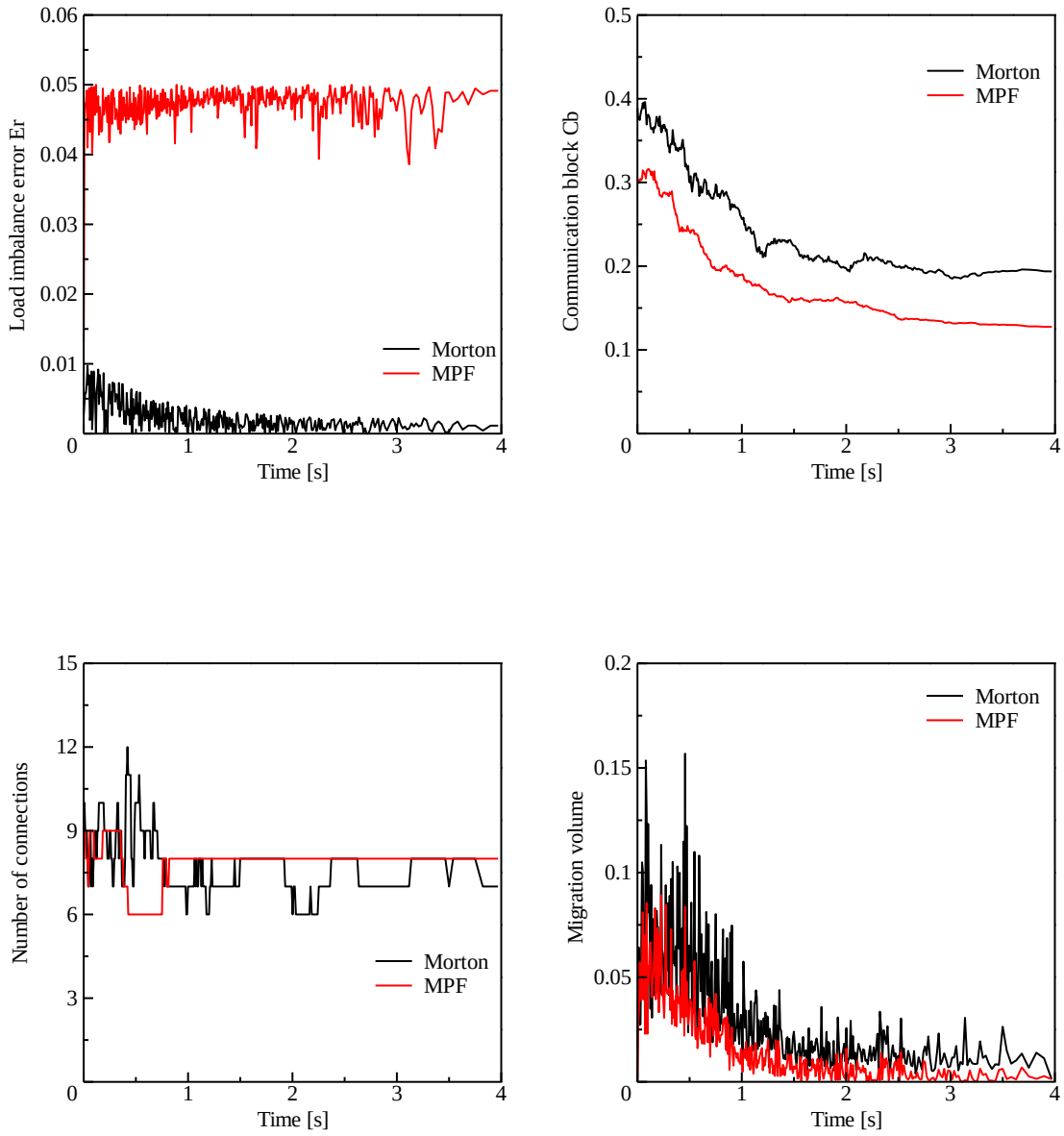


Fig.5.17: Time history of the load imbalance error (upper left), the communication block (upper right), the number of connections (lower left) and the data migration volume (lower right) of two-dimensional advection in uniform velocity field.

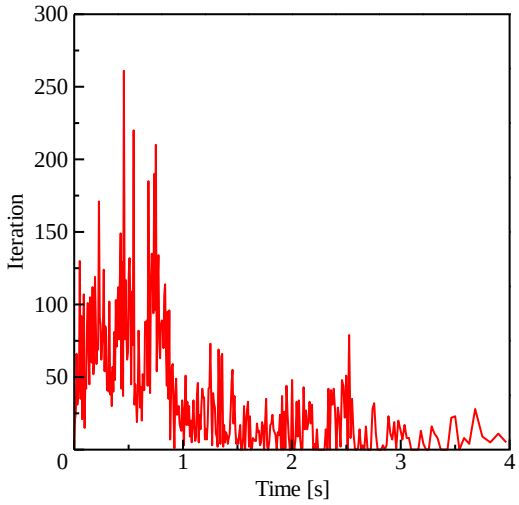


Fig.5.18: Time history of the number of iterations of the MPF method in two-dimensional dynamic domain partitioning.

5.3.3　3 次元動的領域分割

一辺の大きさが 1 の立方体の計算領域を設定し，半径が 0.2 の球形プロファイルを配置し，一様速度場 $u(x,y,z,t) = v(x,y,z,t) = w(x,y,z,t) = 1$ で移流させる．並列数を 16, 32, 64, 128GPU とする．AMR 格子では，各ブロックに $16 \times 16 \times 16$ の均一格子が割り当てられる．最も粗いブロックの解像度は $16 \times 16 \times 16$ 相当であり，$128 \times 128 \times 128$ セルに対応する．最も細かいブロックの解像度は $128 \times 128 \times 128$ 相当であり，$2048 \times 2048 \times 2048$ セルに対応する．各並列数の条件で 2500 ステップの計算を実行し，実行時間を測定する．AMR 法の格子生成と負荷分散は 500 ステップに一度の頻度で実行する．これはクーラン数を 0.03 と設定した場合に，プロファイルがおおよそ 1 ブロック移動したら格子生成と負荷分散を行う頻度である．

領域分割に用いるマルチフェーズフィールド法の計算には $128 \times 128 \times 128$ セル相当の細分化格子を用いる．パラメータは $k = 1600$ と設定する．動的領域分割では，負荷バランスの誤差が 5% 以下になるまで反復計算を行う．

32GPU の条件における，負荷バランスの誤差と通信コスト（小領域表面のブロック数の割合），小領域の接続数，データ移行のコストの時刻歴を Fig.5.19 に示す．3 次元計算においても，設定した負荷バランス誤差 5% 以内をキープすることが出来ている．マルチフェーズフィールド法を用いることで，モートン曲線よりも通信コストを約 83% に削減できている．3 次元移流問題においては，マルチフェーズフィールド法はモートン曲線よりも小領域の接続数の最大値を削減できている．時間平均したデータ移行のコストはマルチフェーズフィールド法で 0.068，モートン曲線で 0.414 であり，マルチフェーズフィールド法はデータ移行のコストを 16.4% に削減できている．

32GPU の条件において，一回の領域分割に要した MPF 法の反復回数を Fig.5.20 に示す．一度の領域分割に要した反復回数は平均で 21 回であり，分割に要した時間は 1.84 秒である．全計算時間に対してマルチフェーズフィールド法の領域分割の時間は 1.2% である．

3 次元移流問題を 16, 32, 64, 128GPU で計算した際の実行時間の内訳を Fig.5.21 に示す．青色の計算時間はマルチフェーズフィールド法とモートン曲線で同程度であり，マルチフェーズフィールド法が適切に負荷分散を行えていることが確認できる．どの並列数の条件においてもマルチフェーズフィールド法の領域間通信の時間（赤色）はモートン曲線の通信時間よりも短い．マルチフェーズフィールド法により通信コストの削減が行えているのが確認できる．紫色で表示される領域分割の時間は，並列数が少ない場合は全体の実行時間に対して非常に短い．しかし，並列数の増加に伴い分割に領域分割に時間がかかり，128GPU では，全体の実行時間の 17% がマルチフェーズフィールド法による領域分割に費やされている．並列数の増加に伴い解くべきフェーズフィールド変数 ϕ の数が増えること，界面の表面積が増えマルチフェーズフィールド法を解くためのセルベースの細分化格子のセル数が増加すること，領域分割にかかる反復回数が増加することなどが原因だと考えられる．領域分割の変更に伴うデータ移行の時間はマルチフェーズフィールド法を用いることで削減できている．本計算では，データ移行のコストは非常に小さく，最もデータ移行に時間がかかったモートン曲線を用いた 128GPU の条件でも，その実行時間は全体の 3.5% ほどであった．保存型 Allen-Cahn 方程式を用いて与えられた速度場で界面を移流

させる計算では，データ移行で通信する変数は界面の秩序変数 ϕ^{AC} だけであるため，この実行性能測定では，データ移行のコストが小さかったと考えられる．

128GPU の計算でマルチフェーズフィールド法による領域分割のオーバヘッドが大きくなってしまったことを受け，領域分割法の高速化が今後の課題として挙げられる．格子生成と領域分割の頻度を下げることで，領域分割のオーバーヘッドを小さくすることができるが，そのためには界面近傍により広い高解像度格子の領域が必要である．格子点数の増加に伴い計算時間や通信時間は長くなるが，格子生成や領域分割のオーバーヘッドが非常に大きい場合は，これにより全体の実行時間を削減できる．高解像度格子を割り当てる領域と AMR 法の格子生成と領域分割の頻度の設定について，今後，様々な条件での計算を行うことで適切な頻度を推定する予定である．

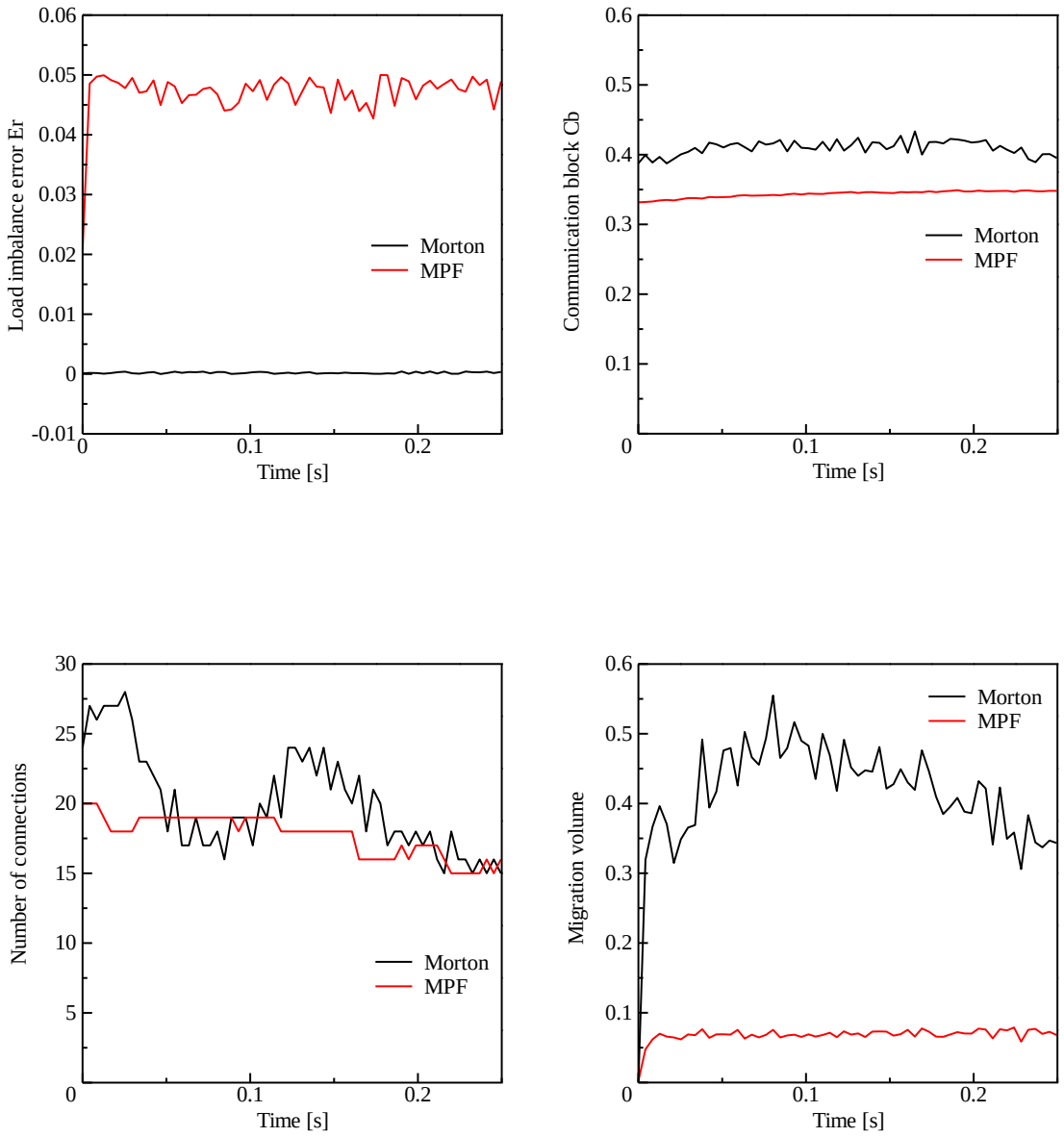


Fig.5.19: Time history of the load imbalance error (upper left), the communication block (upper right), the number of connections (lower left) and the data migration cost (lower right) of three-dimensional advection in uniform velocity field using 32 GPUs.

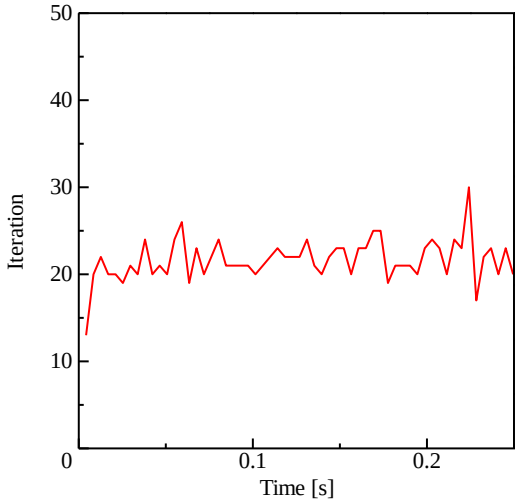


Fig.5.20: Time history of the number of iterations of the MPF method in three-dimensional dynamic domain partitioning using 32 GPUs.

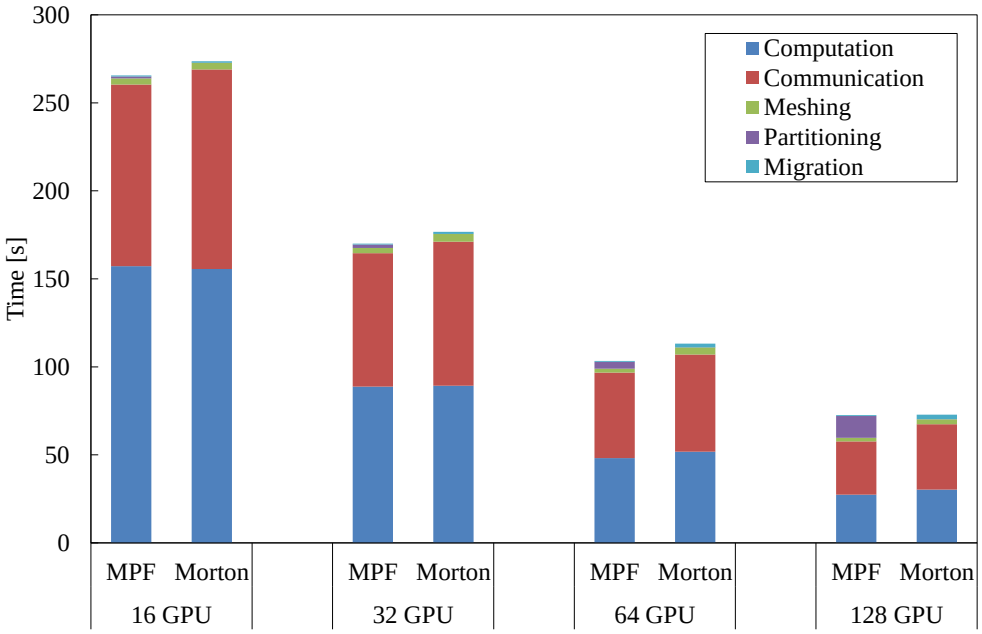


Fig.5.21: The breakdown of the wall-clock time of three-dimensional AMR application.

5.4　マルチフェーズフィールド法に基づくノード間通信削減のための動的負荷分散手法の開発のまとめ

第 4 章のモートン曲線を用いた負荷分散では，並列数の増加に伴い通信時間が増加し，強スケーリングが低下することが明らかになった．本章では，GPU 間の通信コストを削減するために，多結晶組織の成長過程のシミュレーション手法であるマルチフェーズフィールド法を用いた新しい領域分割手法の開発を行った．マルチフェーズフィールド法では各結晶粒が界面エネルギーの減少により成長するため，界面の表面積が最小化された結晶組織が得られる．この特徴に着目し，マルチフェーズフィールド法で計算された結晶組織に基づいて細分化格子を小領域に分割することで，小領域の表面積を小さくし通信量を削減できると考えた．マルチフェーズフィールド法の時間発展方程式に，各結晶粒に対応する小領域の計算コスト（ブロック数）を均一にする体積補正項を導入することで，負荷分散を行いつつ，界面エネルギーの減少により通信コストを陰的に最小化する領域分割法を提案した．

AMR 法を用いた界面移流計算の複数 GPU コードに提案した領域分割手法を実装し，動的負荷分散の性能をモートン曲線を比較した．提案手法は動的領域分割においても各 GPU の負荷バランス誤差を 5% 以下に保つことができ，通信コストをモートン曲線よりも削減することに成功した．また，提案手法は領域再分割に以前の領域分割の情報を利用するため，領域分割の変更に伴うデータ移動のコストをモートン曲線よりも削減することが可能であることを示した．

より通信時間を削減するために，今後，マルチフェーズフィールド法の補正項などの改善を進めていく予定である．具体的には，現在のモデルでは結晶粒の表面積を最小化しているが，通信時間は表面積だけでなく，通信相手の数や通信レイテンシ，通信相手がノード内かノード外かなど様々な要因に依存する．領域分割のパラメータとして通信時間を導入することで，通信時間の最小化を目指す．

第 6 章

結言

複数の相が混ざり合う混相流は化学プラントでの流動層，土石流や津波などの自然災害で現れ，界面および物体表面での現象を捉えることのできる高解像度計算が必要であるが，流体計算のコストが高い相互作用の直接計算による混相流の直接シミュレーションは多く行われてこなかった．本研究では，GPU スパコンで混相流の直接シミュレーションを高効率に実行するための計算手法を提案し，複数 GPU を用いた大規模並列計算により混相流の詳細な解析が実現可能であることを明らかにした．

各章の内容を以下にまとめる．

格子ボルツマン法に基づく混相流解析手法の開発

第 2 章では，混相流計算の高速化・大規模化に向けた，圧力のポアソン方程式を解く必要がない完全な陽解法計算である格子ボルツマン法に基づく混相流解析手法の開発を行った．格子ボルツマン法による単相流解析，フェーズフィールド法による界面捕獲，個別要素法による固体粒子と複雑形状の物体の計算を組み合わせた，自由界面を含む混相流の解析手法を構築した．具体的には，単純な SRT モデルの格子ボルツマン法では高レイノルズ数で不安定になるため，MRT やキュムラントの衝突モデル，LES 乱流モデルを混相流計算の導入することで計算を安定化させた．壁面境界を精度良く扱える Interpolated Bounce-back 法に基づく流体と物体の相互作用計算を導入した．高密度比かつ高レイノルズ数の気液二相流は計算が不安定になるため，格子ボルツマン法で液相領域の速度場のみを計算し，気相領域には Velocity extension により速度場を補外することで，高レイノルズ数の自由界面を含む流れの計算を可能にした．個別要素法には複数の球形粒子を剛体連結した非球形粒子モデルを導入し，複雑形状の固体物体を混相流解析を可能にした．

提案した混相流解析手法の検証として，典型的なベンチマーク問題を行った．3 次元速度場での界面移流計算では，保存型 Allen-Cahn 方程式により十分な精度で移流計算を扱えることを確認するとともに，拡散・逆拡散項のパラメータに界面が分裂する際の挙動が変わることを確認した．非球形粒子を用いた個別要素法で斜面を砂が滑り落ちる計算を行い，先行研究の実験と比較し，砂の空間分布をよく再現できた．一定速度で移動する球の抗力係数や単一球形粒子の沈降速度を実験と比較し，Interpolated Bounce-back 法により十分な精度で流体と物体の相互作用を計算可能であることを示した．濡れた床へのダム崩壊計算では，キュムラントモデルを用いることで激しい流れを安定に計算可能であることを確

認し，ダム崩壊により砕波が生じる過程を再現することに成功した．また，保存型 Allen-Cahn 方程式のモビリティの大きさにより液滴の飛沫の大きさが変わることが明らかになった．障害物を含むダム崩壊計算では，キュムラントモデルを導入することで圧力振動を大幅に抑えられ，水面の高さや障害物表面の圧力が Kleefsman らの実験とよく一致することを確認した．円柱の自由界面の落下の計算では，自由界面の形状や円柱の位置の時間変化が計算と実験でよく一致した結果が得られ，提案手法により自由界面流れと物体の連成計算を妥当に行えることを示した．以上の結果より，提案した格子ボルツマン法に基づく混相流の解析手法は，高レイノルズ数の激しい流れ場においても安定して十分な精度で計算可能であることを示した．

複数 GPU を用いた大規模混相流解析

第 3 章では，2 章で提案した計算手法の複数 GPU 実装を行い，複数台の GPU を用いた大規模混相流シミュレーションを実施した．格子ボルツマン法やフェーズフィールド法などの格子計算と個別要素法の粒子計算の単一 GPU 実装について述べた．個別要素法の計算で最も計算時間のかかる近傍粒子探索に対し，リンク・リスト法，ハッシュ法，リンク・リスト法を粒子登録法に併用した手法，ハッシュ法を粒子登録法に併用した手法の 4 つ近傍探索手法を実装し，計算時間を比較した．リンク・リスト法を粒子登録法に併用した手法が最も計算時間が短く，GPU による個別要素法の計算に適していることが明らかになった

複数 GPU 実装では，格子ボルツマン法とフェーズフィールド法の格子点数に比べて個別要素法で計算する物体数や粒子数が非常に少ないことに着目し，格子計算は 2 次元領域分割法による複数 GPU 実装を行い，個別要素法は並列化を行わずに各 GPU が全ての粒子に関して冗長に計算する方法を提案した．東京工業大学の GPU スパコン・TSUBAME3.0 の GPU を用いて弱スケーリングと強スケーリングの性能測定を行った．弱スケーリングでは，256GPU での並列化効率は約 83% であり 16GPU に対して 13.3 倍の高い実行性能を得られ，複数 GPU 計算をしていない個別要素法の計算のオーバーヘッドは 8.4% であった．一方，強スケーリングでは GPU 数の増加に伴う並列化効率の低下し，これは GPU 間の通信のオーバーヘッドが原因であると考えられる．

等間隔直交格子を用いて混相流の実問題に対する大規模計算を東京工業大学の TSUBAME2.5 および TSUBAME3.0 を用いて実行した．多数の固体粒子を含む粒子系混相流の解析として，粉体を流動化させて混合や乾燥を行う噴流層のシミュレーションを行った．典型的な噴流層で現れる空気が流れる領域（Spout），固体粒子が巻き上げられる領域（Fountain），流路の周りで粒子が沈んでいく環状の領域（Annulus）の 3 つの領域が形成され，循環運動による粒子の混合をシミュレーションで再現することに成功した．複雑形状の物体を含む流れとして，512 枚のイチョウの葉が舞い降りる計算を 128 台の GPU を用いて行い，イチョウの葉がひらひらと落下する様子や，回転しながら落下するイチョウの葉の特徴的な現象を再現することに成功した．自由界面を含む流れとして，水槽実験を模擬した 18 個の流木を含む津波流の計算を実行した．津波で流された流木が壁面に衝突した際の接触力は津波波力の 15 倍以上になることが明らかになり，この結果から，構造物への津波の影響を評価する際には，瓦礫や流木などの物体の影響を考慮することの重要性を示せた．この津波計算は 3.6 億格子を用いた大規模計算であるが，

TSUBAME3.0 の GPU Tesla P100 を 24 台用いて 24 時間で計算を完了しており，提案した格子ボルツマン法による完全陽解法の混相流解析手法と GPU を用いた並列計算により，大規模計算が短時間で実現可能になった．

AMR 法を導入した格子ボルツマン法の大規模 GPU 計算

第 4 章では，開発した格子ボルツマン法による混相流解析手法に動的な AMR 法を導入し，複数 GPU 並列計算を実装した．木構造に基づく細分化を行い，木構造の末端であるリーフに $8 \times 8 \times 8$ の均一格子のブロックを割り当てることで，ブロック内の格子点のメモリアドレスを連続にし，GPU 実装における実行性能の向上を図った．格子解像度が変わる境界近傍に補間した値を記憶するためのゴーストブロックを配置することで，各ブロックが袖領域を持つよりもメモリ使用量を抑えつつ，補間のための条件分岐を削減した．動的な格子細分化の実装では，ゴーストブロックを利用したメモリ管理を行うことで，GPU 上で細分化と粗大化の処理を実行し，CPU と GPU 間の通信を抑えた．

複数 GPU 実装では，空間充填曲線であるモートン曲線を用いた領域分割を行い，各 GPU に割り当てられるブロック数を均一にした．袖領域の通信では，小領域表面の計算点に保存されている全方向の速度分布関数を通信する必要はないため，隣接の小領域に移流する分布関数のみをパッキングして隣接 GPU に転送することで，GPU 間の通信量を削減した．隣接領域のデータが必要な小領域表面のブロックと通信が必要でない内側のブロックに分け，別々のカーネル関数で非同期実行することによる演算と通信のオーバーラップ手法を提案した．固定された細分化格子を用いて単相の格子ボルツマン法の弱スケーリングを測定し，8GPU から 128GPU で 98% 以上の並列化効率を達成し，オーバーラップ法を用いることで実行性能を 1.37 倍に向上させた．強スケーリングでは，オーバーラップ法を導入ても並列化効率の低下が確認でき，これは主に並列数の増加に伴い通信時間の割合が増えることが原因であり，より高いスケーラビリティを達成するためには通信コストの削減が必要であることが明らかになった．一方，動的な格子細分化・粗大化を行う場合の弱スケーリングでは，GPU 数の増加に伴い格子細分化に要する時間が長くなり，並列化効率が低下することが確認でき，大規模 AMR 計算を高効率に実行するためには，木構造の処理や負荷分散に関しても MPI による並列化が必要であることが明らかになった．

AMR 法の検証として，界面移流の 3 次元計算と単一球形粒子の沈降計算を AMR 法を用いて行い，均一格子の結果とよく一致する結果が得られた．界面移流の計算では，格子解像度が $256 \times 256 \times 256$ 格子相当と $512 \times 512 \times 512$ 格子相当の計算を行い，界面に高解像度を適合する AMR 法では大規模計算になるほど格子点数の削減の効果が高いことが明らかになった．

東京工業大学の TSUBAME3.0 を用いて，AMR 法を導入した大規模な乱流計算と混相流計算を実行した．48 台の GPU を用いて球の直径に 2048 格子を割り当てた大規模な球周りの流れの計算を行い，レイノルズ数が 20 万から 30 万で抗力係数が急激に低下するドラッククライシスをキュムラントモデルの格子ボルツマン法で再現可能であることを示した．噴流によるピンポン玉の浮遊の計算では，ピンポン玉が水平方向にも揺れながら浮遊し，その振幅が速度に応じて大きくなる傾向があることが明らかになった．自由界面に高解像度格子を適合した大規模なダム崩壊計算を行い，格子解像度を細かくするとより細かい液滴の飛沫や壁に張り付く液滴が発生し，自由界面を含む激しい流れの計算ではより細かい

格子が必要であることが確認できた．格子点数は界面の変形や飛沫により時間とともに増加する傾向があり，最も格子点数が多い時刻でも，格子点数を均一格子に対して 17% に削減できた．回転する板による水面の撹拌の計算を行い，提案手法が安定して自由界面流れと移動物体の計算可能であることを確認するとともに，板の回転速度が速く界面が乱れるほど格子点数が増えることが明らかになった．

マルチフェーズフィールド法に基づくノード間通信削減のための動的負荷分散手法の開発

第 5 章では，GPU 間の通信コストを削減するために，多結晶組織の成長過程のシミュレーション手法であるマルチフェーズフィールド法を用いた新しい領域分割手法の開発を行った．マルチフェーズフィールド法では各結晶粒が界面エネルギーの減少により成長するため，界面の表面積が最小化された結晶組織が得られる．この特徴に着目し，マルチフェーズフィールド法で計算された結晶組織に基づいて細分化格子を小領域に分割することで，小領域の表面積を小さくし通信量を削減できると考えた．マルチフェーズフィールド法の時間発展方程式に，各結晶粒に対応する小領域の計算コスト（ブロック数）を均一にする体積補正項を導入することで，負荷分散を行いつつ，界面エネルギーの減少により通信コストを陰的に最小化する領域分割法を提案した．

AMR 法を用いた界面移流計算の複数 GPU コードに提案した領域分割手法を実装し，動的負荷分散の性能をモートン曲線を比較した．提案手法は動的領域分割においても各 GPU の負荷バランス誤差を 5% 以下に保つことができ，通信コストをモートン曲線の約 80% に削減することに成功した．また，提案手法は領域再分割に以前の領域分割の情報を利用するため，領域分割の変更に伴うデータ移動のコストをモートン曲線よりも削減することが可能であることを示した．

以上，本研究では，混相流の直接シミュレーションを GPU スパコンで高効率に実行するための計算手法を開発し，複数 GPU を用いた大規模な混相流シミュレーションに成功した．

非圧縮性流体の計算に一般的に用いられる半陰解法では，大規模計算で圧力のポアソン方程式の収束性の悪化により計算効率が低下するため，完全な陽解法である格子ボルツマン法を用いることで半陰解法から脱却し，大規模計算での高い実行性能を達成した．一方，格子ボルツマン法では高レイノルズ数の流れで計算が不安定になるため，MRT モデルやキュムラントモデルなどの乱流解析に提案された手法を混相流解析に適用し，津波などのスケールが大きく激しい流れの計算を可能にした．

混相流解析では，物体表面や気液界面における現象が重要であることに着目し，界面と物体表面に高解像度の格子を局所的に集める AMR 法を格子ボルツマン法に導入し，計算コストとメモリ使用量の削減を大幅に削減した．AMR 法を導入した場合，計算負荷が空間・時間的に変化するため，計算領域を均等に分割する領域分割法では，各 GPU の計算格子点数が均一にならず並列化効率が低下するだけでなく，GPU のメモリ容量を上回る格子点数の割り当てが生じ，メモリ不足により大規模計算を実行することが出来ない．AMR 法の複数 GPU 計算で生じるこの問題を解決するために，本研究では，計算領域を空間充填曲線に従い動的に分割することで各 GPU が処理する格子点数を均一化し，GPU スパコンでの AMR 法を用いた大規模な格子ボルツマン法の計算を実現した．通信と演算のオーバーラップによる通

信時間の隠蔽手法を AMR 法に導入することで実行性能を 1.37 倍に向上させることに成功し，固定した格子の計算において 16GPU から 128GPU までほぼ理想的な弱スケーリングを達成した．一方，動的な格子細分化・粗大化の処理時間は並列数と計算規模の増加に伴い長くなり，これらが弱スケーリングを低下させる原因であることが確認でき，より大規模な AMR 法の計算を実現するためには木構造の処理の並列化が課題となることが明らかになった．

空間充填曲線による領域分割により AMR 法を導入した格子ボルツマン法の大規模 GPU 計算を可能にしたが，演算と通信のオーバーラップを導入してもなお，強スケーリングでは GPU 間の通信時間が大きなオーバーヘッドになることが明らかになった．このことから，GPU スパコンで AMR 法の実行性能をより向上させるには，GPU 間の通信時間の削減が重要であると考え，GPU 間の通信コストを削減するためのマルチフェーズフィールド法に基づく動的領域分割法を提案した．界面エネルギーの減少に基づく結晶粒の成長により，各 GPU が計算する小領域の表面積を最小化することで，提案した領域分割法は通信コストをモートン曲線の約 80% に削減することに成功した．提案手法は結晶粒成長という物理現象に基づく領域分割法であり，AMR 法の動的負荷分散問題に対して，既存の空間充填曲線やグラフ理論と全く異なるアプローチを提案することが出来た．

格子ボルツマン法に動的な AMR 法を導入し，GPU スパコンで大規模計算を行った例は無く，本研究では，AMR 法と GPU を用いることで格子ボルツマン法の大規模計算が実現可能であることを明らかにした．演算性能が高く低消費電力である GPU をアクセラレータとして搭載したスパコンは今後さらに増えると考えられ，今後のエクサスケールに向けた超大規模流体解析手法として本研究は貢献できる．また，本研究で提案した AMR の実装方法の一部は格子ボルツマン法以外のステンシル計算にも適用可能であり，その有用性は高い．

参考文献

[1] Yutaka Tsuji, Toshihiro Kawaguchi, and Toshitsugu Tanaka. Discrete particle simulation of two-dimensional fluidized bed. *Powder technology*, Vol. 77, No. 1, pp. 79–87, 1993.

[2] 酒井幹夫, 越塚誠一. 水平管内固気混相流における離散要素法粗視化モデルの適用. 日本機械学會論文集. B 編, Vol. 74, No. 742, pp. 1332–1339, jun 2008.

[3] Shinichi Yuu, Toshihiko Umekage, and Yuuki Johno. Numerical simulation of air and particle motions in bubbling fluidized bed of small particles. *Powder Technology*, Vol. 110, No. 1, pp. 158–168, 2000.

[4] 酒井幹夫, 柴田和也, M. Kawasaki Vanessa, 越塚誠一. ＤＥＭ粗視化モデルによる気泡流動層の数値解析. 粉体工学会誌, Vol. 47, No. 1, pp. 17–25, 2010.

[5] DRJ Owen, CR Leonardi, and YT Feng. An efficient framework for fluid–structure interaction using the lattice Boltzmann method and immersed moving boundaries. *International Journal for Numerical Methods in Engineering*, Vol. 87, No. 1-5, pp. 66–95, 2011.

[6] 大槻敏, 松岡俊文. 固液二相達成シミュレーションによる傾斜沈降メカニズムの検討. 混相流, Vol. 23, No. 4, pp. 420–435, 2009.

[7] 杉原健太. 界面追跡法による多相流シミュレーションの GPU を用いた高速化. PhD thesis, 東京工業大学, 2011.

[8] 小野寺直幸, 青木尊之. GPU を用いた固体粒子を含む固気液三相流の大規模シミュレーション. 混相流, Vol. 27, No. 5, pp. 607–613, 2014.

[9] Dieter A Wolf-Gladrow. *Lattice-gas cellular automata and lattice Boltzmann models: An Introduction*. Springer Science & Business Media, 2000.

[10] Massimo Bernaschi, Mauro Bisson, Toshio Endo, Satoshi Matsuoka, Massimiliano Fatica, and Simone Melchionna. Petaflop biofluidics simulations on a two million-core system. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011.

[11] Amanda Randles, Erik W Draeger, Tomas Oppelstrup, Liam Krauss, and John A Gunnels. Massively parallel models of the human circulatory system. In *Proceedings of 2015 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015.

[12] Programming Guide :: CUDA Toolkit Documentation. http://docs.nvidia.com/cuda/cuda-c-programming-guide/. Last access: 19 Dec. 2018.

[13] TOP500 Supercomputer Sites. http://www.top500.org/. Last access: 19 Dec. 2018.

[14] Qingang Xiong, Bo Li, Ji Xu, Xiaowei Wang, Limin Wang, and Wei Ge. Efficient 3D DNS of gas–solid flows on Fermi GPGPU. *Computers & Fluids*, Vol. 70, pp. 86–94, 2012.

[15] Wang Xian and Aoki Takayuki. Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster. *Parallel Computing*, Vol. 37, No. 9, pp. 521–535, 2011.

[16] 小野寺直幸, 青木尊之, 下川辺隆史, 小林宏充. 格子ボルツマン法による 1m 格子を用いた都市部 10km 四方の大規模 LES 気流シミュレーション. ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集, pp. 123–131, 2013.

[17] Wei Ge, Wei Wang, Ning Yang, Jinghai Li, Mooson Kwauk, Feiguo Chen, Jianhua Chen, Xiaojian Fang, Li Guo, Xianfeng He, Xinhua Liu, Yaning Liu, Bona Lu, Jian Wang, Junwu Wang, Limin Wang, Xiaowei Wang, Qingang Xiong, Ming Xu, Lijuan Deng, Yongsheng Han, Chaofeng Hou, Leina Hua, Wenlai Huang, Bo Li, Chengxiang Li, Fei Li, Ying Ren, Ji Xu, Nan Zhang, Yun Zhang, Guofeng Zhou, and Guangzheng Zhou. Meso-scale oriented simulation towards virtual process engineering (VPE)-The EMMS Paradigm. *Chemical Engineering Science*, Vol. 66, No. 19, pp. 4426–4458, 2011.

[18] Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, Vol. 53, No. 3, pp. 484–512, 1984.

[19] Robert Lucas, James Ang, Keren Bergman, Shekhar Borkar, William Carlson, Laura Carrington, George Chiu, Robert Colwell, William Dally, Jack Dongarra, et al. DOE advanced scientific computing advisory subcommittee (ASCAC) report: top ten Exascale research challenges. Technical report, USDOE Office of Science (SC)(United States), 2014.

[20] AMReX-Codes. https://amrex-codes.github.io/. Last access: 19 Dec. 2018.

[21] Marsha J. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, Vol. 82, No. 1, pp. 64–84, 1989.

[22] Tiankai Tu, David R. O'Hallaron, and Omar Ghattas. Scalable parallel octree meshing for terascale applications. *Proceedings of the ACM/IEEE 2005 Supercomputing Conference, SC'05*, 2005.

[23] Carsten Burstedde, Lucas C. Wilcox, and Omar Ghattas. p4est: Scalable Algorithms for Parallel Adaptive Mesh Refinement on Forests of Octrees. *SIAM Journal on Scientific Computing*, Vol. 33, No. 3, pp. 1103–1133, 2011.

[24] Hsi-Yu Schive, Tzihong Chiueh, and Yu-Chih Tsai. GAMER: a GPU-Accelerated adaptive mesh refinement code for astrophysics. *Astrophys. J. Suppl.*, Vol. 186, No. arXiv: 0907.3390, pp. 457–484, 2009.

[25] SIMULIA PowerFLOW. https://exa.com/en/product/simulation-tools/powerflow-cfd-simulation. Last access: 19 Dec. 2018.

[26] SIMULIA XFlow CFD. https://www.simuleon.com/simulia-xflow-cfd/. Last access: 19 Dec. 2018.

[27] Martin Schönherr. *Towards reliable LES-CFD computations based on advanced LBM models utilizing (Multi-) GPGPU hardware*. PhD thesis, Jul 2015.

[28] Naoyuki Onodera and Yasuhiro Idomura. Acceleration of wind simulation using locally mesh-refined lattice Boltzmann method on GPU-rich supercomputers. In *In proceedings of Asian Conference on Supercomputing Frontiers*, pp. 128–145, 2018.

[29] ultraFluidX. https://altairhyperworks.com/product/ultrafluidx. Last access: 19 Dec. 2018.

[30] Abbas Fakhari and Taehun Lee. Finite-difference lattice Boltzmann method with a block-structured adaptive-mesh-refinement technique. Vol. 033310, pp. 1–12, 2014.

[31] Abbas Fakhari, Martin Geier, and Taehun Lee. A mass-conserving lattice Boltzmann method with dynamic grid refinement for immiscible two-phase flows. *Journal of Computational Physics*, Vol. 315, pp. 434–457, 2016.

[32] Kai Feldhusen, Ralf Deiterding, and Claus Wagner. A dynamically adaptive lattice boltzmann method for thermal convection problems. *International Journal of Applied Mathematics and Computer Science*, Vol. 26, No. 4, pp. 735–747, 2016.

[33] Zhao Yu and Liang-Shih Fan. An interaction potential based lattice Boltzmann method with adaptive mesh refinement (AMR) for two-phase flow simulation. *Journal of Computational Physics*, Vol. 228, No. 17, pp. 6456–6478, 2009.

[34] waLBerla. http://walberla.net/index.html. Last access: 19 Dec. 2018.

[35] Christian Godenschwager, Florian Schornbaum, Martin Bauer, Harald Kostler, and Ulrich Rude. A framework for hybrid parallel flow simulations with a trillion cells in complex geometries. In *in proceedings of the 2013 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2013*, 2013.

[36] Florian Schornbaum and Ulrich Rüde. Extreme-scale block-structured adaptive mesh refinement. *SIAM Journal on Scientific Computing*, Vol. 40, , 2017.

[37] Christoph Rettinger, Christian Godenschwager, Sebastian Eibl, Tobias Preclik, Tobias Schruff, Roy Frings, and Ulrich Rüde. Fully resolved simulations of dune formation in riverbeds. In *International Supercomputing Conference*, pp. 3–21, 2017.

[38] Arthur R. Butz. Space filling curves and mathematical programming. *Information and Control*, Vol. 12, No. 4, pp. 314–330, 1968.

[39] Bruce Hendrickson and Tamara G Kolda. Graph partitioning models for parallel computing. *Parallel Computing*, Vol. 26, No. 12, pp. 1519–1534, 2000.

[40] Junseok Kim. A continuous surface tension force formulation for diffuse-interface models.

Journal of Computational Physics, Vol. 204, No. 2, pp. 784–804, 2005.

[41] Pao-hsiung Chiu and Yan-ting Lin. A conservative phase field method for solving incompressible two-phase flows. *Journal of Computational Physics*, Vol. 230, No. 1, pp. 185–204, 2011.

[42] Peter A Cundall and Otto DL Strack. A discrete numerical model for granular assemblies. *Geotechnique*, Vol. 29, No. 1, pp. 47–65, 1979.

[43] Dominique d'Humières. Multiple–relaxation–time lattice Boltzmann models in three dimensions. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, Vol. 360, No. 1792, pp. 437–451, 2002.

[44] Martin Geier, Martin Schönherr, Andrea Pasquali, and Manfred Krafczyk. The cumulant lattice boltzmann equation in three dimensions: Theory and validation. *Computers & Mathematics with Applications*, Vol. 70, No. 4, pp. 507–547, 2015.

[45] Gary S Grest, Burkhard Dünweg, and Kurt Kremer. Vectorized link cell Fortran code for molecular dynamics simulations for a large number of particles. *Computer Physics Communications*, Vol. 55, No. 3, pp. 269–285, 1989.

[46] Simon Green. Particle simulation using cuda. *NVIDIA Whitepaper, December 2010*, 2010.

[47] 平林久義, 佐藤雅弘. 線形リストを用いた粒子法の近傍粒子探索. 日本計算工学会論文集, Vol. 2010, No. 20100001, 2010.

[48] 小口かなえ, 澁田靖, 鈴木俊夫. GPU を用いた分子動力学法解析の高速化. 日本金属学会誌, Vol. 76, No. 7, pp. 462–467, 2012.

[49] George Karypis and Vipin Kumar. METIS* A software package for partitioning unstructured graphs , partitioning meshes , and computing fill-reducing orderings of sparse matrices. *Manual*, pp. 1–44, 1998.

[50] Mohamed Wahib, Naoya Maruyama, and Takayuki Aoki. Daino: A high-level framework for parallel and efficient AMR on GPUs. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, pp. 621–632, 2017.

[51] Hsi-Yu Schive, John A ZuHone, Nathan J Goldbaum, Matthew J Turk, Massimo Gaspari, and Chin-Yu Cheng. GAMER-2: a GPU-accelerated adaptive mesh refinement code–accuracy, performance, and scalability. *Monthly Notices of the Royal Astronomical Society*, Vol. 481, No. 4, pp. 4815–4840, 2018.

[52] I. Steinbach and F. Pezzolla. A generalized field method for multiphase transformations using interface fields. *Physica D: Nonlinear Phenomena*, Vol. 134, No. 4, pp. 385–393, 1999.

[53] Seong Gyoon Kim, Dong Ik Kim, Won Tae Kim, and Yong Bum Park. Computer simulations of two-dimensional and three-dimensional ideal grain growth. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, Vol. 74, No. 6, pp. 1–14, 2006.

[54] P. L. Bhatnagar, E. P. Gross, and M. Krook. A model for collision processes in gases. I.

Small amplitude processes in charged and neutral one-component systems. *Physical Reviews*, Vol. 94, pp. 511–525, 1954.

[55] K Suga, Y Kuwata, K Takashima, and R Chikasue. A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows. *Computers & Mathematics with Applications*, Vol. 69, No. 6, pp. 518–529, 2015.

[56] Abbas Fakhari, Martin Geier, and Taehun Lee. A mass-conserving lattice Boltzmann method with dynamic grid refinement for immiscible two-phase flows. *Journal of Computational Physics*, Vol. 315, pp. 434–457, 2016.

[57] Martin Geier, Andreas Greiner, and Jan G Korvink. Cascaded digital lattice Boltzmann automata for high Reynolds number flow. *Physical Review E*, Vol. 73, No. 6, p. 066705, 2006.

[58] Martin Geier, Andrea Pasquali, and Martin Schönherr. Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion part I: Derivation and validation. *Journal of Computational Physics*, Vol. 348, pp. 862–888, 2017.

[59] Hiromichi Kobayashi. The subgrid-scale models based on coherent structures for rotating homogeneous turbulence and turbulent channel flow. *Physics of Fluids (1994-present)*, Vol. 17, No. 4, p. 045104, 2005.

[60] MJ Fritts and JP Boris. The lagrangian solution of transient problems in hydrodynamics using a triangular mesh. *Journal of Computational Physics*, Vol. 31, No. 2, pp. 173–215, 1979.

[61] Francis H Harlow and J Eddie Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *The physics of fluids*, Vol. 8, No. 12, pp. 2182–2189, 1965.

[62] Cyril W Hirt and Billy D Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of computational physics*, Vol. 39, No. 1, pp. 201–225, 1981.

[63] Mark Sussman, Peter Smereka, and Stanley Osher. A level set approach for computing solutions to incompressible two-phase flow. *Journal of Computational physics*, Vol. 114, No. 1, pp. 146–159, 1994.

[64] Martin Geier, Abbas Fakhari, and Taehun Lee. Conservative phase-field lattice Boltzmann model for interface tracking equation. *Physical Review E*, Vol. 91, No. 6, p. 063309, 2015.

[65] Jeremiah U Brackbill, Douglas B Kothe, and Charles Zemach. A continuum method for modeling surface tension. *Journal of computational physics*, Vol. 100, No. 2, pp. 335–354, 1992.

[66] Danping Peng, Barry Merriman, Stanley Osher, Hongkai Zhao, and Myungjoo Kang. A PDE-based fast local level set method. *Journal of computational physics*, Vol. 155, No. 2, pp. 410–438, 1999.

[67] 和田浩二, 千秋博紀, 松井孝典. DEM による粉体衝突シミュレーション. 遊・星・人: 日本惑星科学会誌, Vol. 13, No. 4, pp. 233–240, 2004.

[68] 重松孝昌, 小田一紀, 田野雅彦, 廣瀬真由. 個別要素法による水中沈降粒子群の 3 次元挙動に関する研究. 海岸工学論文集, Vol. 47, pp. 996–1000, 2000.

[69] 山本通典, 石原真吾, 加納純也. DEM シミュレーションを用いた回転ドラムにおける粒子形状が混合に及ぼす影響評価. 粉体工学会誌, Vol. 52, No. 8, pp. 445–452, 2015.

[70] Xiaoqiang Yue, Hao Zhang, Congshu Luo, Shi Shu, and Chunsheng Feng. Parallelization of a DEM code based on CPU-GPU heterogeneous architecture. In *Parallel Computational Fluid Dynamics*, pp. 149–159. Springer, 2014.

[71] 阪口秀, 尾崎叡司, 五十嵐徹. 円形要素を用いた DEM における回転の抑制に関する研究. 神戸大学農学部研究報告, Vol. 20, No. 2, pp. 239–246, 1993.

[72] Nicolin Govender, Daniel N Wilke, Schalk Kok, and Rosanne Els. Development of a convex polyhedral discrete element simulation framework for NVIDIA Kepler based GPUs. *Journal of Computational and Applied Mathematics*, Vol. 270, pp. 386–400, 2014.

[73] Benjamin Nassauer and Meinhard Kuna. Contact forces of polyhedral particles in discrete element method. *Granular Matter*, Vol. 15, No. 3, pp. 349–355, 2013.

[74] Darius Markauskas and Rimantas Kačianauskas. Investigation of rice grain flow by multi-sphere particle model with rolling resistance. *Granular Matter*, Vol. 13, No. 2, pp. 143–148, 2011.

[75] 田中正幸, 酒井幹夫, 越塚誠一. 粒子ベース剛体シミュレーションと流体との連成. 日本計算工学会論文集, Vol. 2007, No. 20070007, 2017.

[76] Carolin Körner, Michael Thies, Torsten Hofmann, Nils Thürey, and Ulrich Rüde. Lattice Boltzmann model for free surface flow for modeling foaming. *Journal of Statistical Physics*, Vol. 121, No. 1-2, pp. 179–196, 2005.

[77] M' hamedBouzidi, Mouaouia Firdaouss, Pierre Lallemand. Momentum transfer of a Boltzmann-lattice fluid with boundaries. *Physics of Fluids (1994-present)*, Vol. 13, No. 11, pp. 3452–3459, 2001.

[78] Hui Gao, Hui Li, and Lian-Ping Wang. Lattice Boltzmann simulation of turbulent flow laden with finite-size particles. *Computers & Mathematics with Applications*, Vol. 65, No. 2, pp. 194–210, 2013.

[79] Binghai Wen, Chaoying Zhang, Yusong Tu, Chunlei Wang, and Haiping Fang. Galilean invariant fluid–solid interfacial dynamics in lattice Boltzmann simulations. *Journal of Computational Physics*, Vol. 266, pp. 161–170, 2014.

[80] Hui Gao, Hui Li, and Lian-Ping Wang. Lattice Boltzmann simulation of turbulent flow laden with finite-size particles. *Computers & Mathematics with Applications*, Vol. 65, No. 2, pp. 194–210, 2013.

[81] YT Feng, K Han, and DRJ Owen. Coupled lattice Boltzmann method and discrete element modelling of particle transport in turbulent fluid flows: Computational issues. *International*

Journal for Numerical Methods in Engineering, Vol. 72, No. 9, pp. 1111–1134, 2007.

[82] UKNG Ghia, Kirti N Ghia, and CT Shin. High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of computational physics*, Vol. 48, No. 3, pp. 387–411, 1982.

[83] Randall J LeVeque. High-resolution conservative algorithms for advection in incompressible flow. *SIAM Journal on Numerical Analysis*, Vol. 33, No. 2, pp. 627–665, 1996.

[84] Roger P Denlinger and Richard M Iverson. Flow of variably fluidized granular masses across three-dimensional terrain: 2. Numerical predictions and experimental tests. *Journal of Geophysical Research: Solid Earth (1978–2012)*, Vol. 106, No. B1, pp. 553–566, 2001.

[85] Jaber Almedeij. Drag coefficient of flow around a sphere: Matching asymptotically the wide trend. *Powder Technology*, Vol. 186, No. 3, pp. 218–223, 2008.

[86] A Ten Cate, CH Nieuwstad, JJ Derksen, and HEA Van den Akker. Particle imaging velocimetry experiments and lattice-Boltzmann simulations on a single sphere settling under gravity. *Physics of Fluids (1994-present)*, Vol. 14, No. 11, pp. 4012–4025, 2002.

[87] PK Stansby, A Chegini, and TCD Barnes. The initial stages of dam-break flow. *Journal of Fluid Mechanics*, Vol. 370, pp. 203–220, 1998.

[88] KMT Kleefsman, G Fekken, AEP Veldman, B Iwanowski, and B Buchner. A volume-of-fluid based simulation method for wave impact problems. *Journal of computational physics*, Vol. 206, No. 1, pp. 363–393, 2005.

[89] 佐藤兼太, 越村俊一. 非圧縮型格子ボルツマン法による自由表面流れ解析の計算安定化. 土木学会論文集 B2(海岸工学), Vol. 71, No. 2, pp. 145–150, 2015.

[90] Alibek Issakhov, Yeldos Zhandaulet, and Aida Nogaeva. Numerical simulation of dam break flow for various forms of the obstacle by VOF method. *International Journal of Multiphase Flow*, Vol. 109, pp. 191–206, 2018.

[91] Martin Greenhow and Woei-Min Lin. Nonlinear-free surface effects: experiments and theory. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE DEPT OF OCEAN ENGINEERING, 1983.

[92] Xinying Zhu. *Application of the CIP method to strongly nonlinear wave-body interaction problems*. PhD thesis, Norwegian University of Science and Technology, 2006.

[93] 茂渡悠介, 酒井幹夫, 越塚誠一, 山田祥徳. GPU による離散要素法シミュレーションの高速化. 粉体工学会誌, Vol. 45, No. 11, pp. 758–765, 2008.

[94] Yuan Tian, Ji Qi, Junjie Lai, Qingguo Zhou, and Lei Yang. A heterogeneous CPU-GPU implementation for discrete elements simulation with multiple GPUs. In *Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on*, pp. 547–552. IEEE, 2013.

[95] 西浦泰介, 阪口秀. GPU を用いた DEM の高速化アルゴリズム. 日本計算工学会論文集, Vol. 2010,

No. 20100007, 2010.

[96] G Viccione, V Bovolin, and E Pugliese Carratelli. Defining and optimizing algorithms for neighbouring particle identification in SPH fluid simulations. *International Journal for Numerical Methods in Fluids*, Vol. 58, No. 6, pp. 625–638, 2008.

[97] 横川明. 流動層, 噴流層における流動化の特性とその応用. 粉体工学会誌, Vol. 21, No. 11, pp. 715–723, 1984.

[98] Panagiotis N Kechagiopoulos, Spyros S Voutetakis, and Angeliki A Lemonidou. Cold flow experimental study and computer simulations of a compact spouted bed reactor. *Chemical Engineering and Processing: Process Intensification*, Vol. 82, pp. 137–149, 2014.

[99] 荻野文丸, 張来英, 前橋康雄. 噴流層における最小噴流化速度と最大圧力降下. 化学工学論文集, Vol. 16, No. 4, pp. 788–795, 1990.

[100] Yoshihiro Tanabe and Kunihiko Kaneko. Behavior of a falling paper. *Physical Review Letters*, Vol. 73, No. 10, p. 1372, 1994.

[101] David L Finn. Falling paper and flying business cards. *SIAM News*, Vol. 40, No. 4, p. 3, 2007.

[102] 津波避難ビル等に係るガイドライン（案）. http://www.bousai.go.jp/kohou/oshirase/h17/pdf/050323shiryou2.pdf. Last access: 20 Dec. 2018.

[103] 米山望, 直田梓. 橋梁に作用する津波波力評価に対する VOF 法に基づく数値計算法の適用性検討. 土木学会論文集 B2(海岸工学), Vol. 68, No. 2, pp. I246–I250, 2012.

[104] Takashi Shimokawabe, Takayuki Aoki, Tomohiro Takaki, Akinori Yamanaka, Akira Nukada, Toshio Endo, Naoya Maruyama, and Satoshi Matsuoka. Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer. In *in proceedings of the 2011 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2011*, 2011.

[105] Babak Hejazialhosseini, Diego Rossinelli, Christian Conti, and Petros Koumoutsakos. High throughput software for direct numerical simulations of compressible two-phase flows. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 16. IEEE Computer Society Press, 2012.

[106] Diego Rossinelli, Babak Hejazialhosseini, Panagiotis Hadjidoukas, Costas Bekas, Alessandro Curioni, Adam Bertsch, Scott Futral, Steffen J Schmidt, Nikolaus A Adams, and Petros Koumoutsakos. 11 PFLOP/s simulations of cloud cavitation collapse. In *in proceedings of the 2013 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2013*, 2013.

[107] 長谷川雄太, 青木尊之. Octree 型 AMR を導入した格子ボルツマン法の C++ テンプレートを用いたカーネル生成による GPU 計算の高速化. 情報処理学会論文誌コンピューティングシステム（ACS）, Vol. 9, No. 2, pp. 34–45, 2016.

[108] Paul M Campbell, Karen D Devine, Joseph E Flaherty, Luis G Gervasio, and James D Teresco. Dynamic octree load balancing using space-filling curves. *Williams College Department of Computer Science, Tech. Rep. CS-03-01*, 2003.

[109] Carsten Burstedde, Omar Ghattas, Michael Gurnis, Tobin Isaac, Georg Stadler, Tim Warburton, and Lucas Wilcox. Extreme-scale AMR. In *Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12. IEEE Computer Society, 2010.

[110] Girish V Nivarti, M Mahdi Salehi, and W Kendal Bushe. A mesh partitioning algorithm for preserving spatial locality in arbitrary geometries. *Journal of Computational Physics*, Vol. 281, pp. 352–364, 2015.

[111] Hua Ji, Fue-Sang Lien, and Eugene Yee. A new adaptive mesh refinement data structure with an application to detonation. *Journal of Computational Physics*, Vol. 229, No. 23, pp. 8981–8993, 2010.

[112] 深沢圭一郎, 梅田隆行, 南里豪志ほか. 超並列惑星磁気圏電磁流体シミュレーションに向けた隣接通信の効率化. ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集, Vol. 2012, pp. 101–106, 2012.

[113] Carsten Burstedde, Omar Ghattas, Michael Gurnis, Tobin Isaac, Georg Stadler, Tim Warburton, and Lucas Wilcox. Extreme-scale AMR. pp. 1–12, 2010.

[114] George Constantinescu and Kyle Squires. Numerical investigations of flow over a sphere in the subcritical and supercritical regimes. *Physics of fluids*, Vol. 16, No. 5, pp. 1449–1466, 2004.

[115] 武藤昌也, 坪倉誠, 大島伸行. 回転球に作用する負のマグヌス力の数値解析. 日本機械学会論文集B編, Vol. 77, No. 775, pp. 781–792, 2011.

[116] Faith A Morrison. *An introduction to fluid mechanics*. Cambridge University Press, 2013.

[117] Martin Geier, Andrea Pasquali, and Martin Schönherr. Parametrization of the cumulant lattice Boltzmann method for fourth order accurate diffusion Part II: Application to flow around a sphere at drag crisis. *Journal of Computational Physics*, Vol. 348, pp. 889–898, 2017.

[118] RS Neve, R Nelson, and P Kotsiopoulos. The drag force on spheres in thin jets. *Journal of Fluid Mechanics*, Vol. 107, pp. 521–531, 1981.

[119] T López-Arias, LM Gratton, G Zendri, and S Oss. Forces acting on a ball in an air jet. *Physics Education*, Vol. 46, No. 2, p. 146, 2011.

[120] Takuya Uehara. Numerical simulation of foam structure formation and destruction process using phase-field model. *Advanced Materials Research*, Vol. 1042, pp. 65–69, 2014.

[121] Takuya Uehara. Phase-Field modeling for the three-dimensional space-filling structure of metal foam materials. No. July, pp. 120–125, 2015.

[122] 都築怜理. GPU スパコンにおける動的負荷分散を用いた大規模粒子法シミュレーション. PhD thesis, 東京工業大学, 2016.

[123] William J. Rider and Douglas B. Kothe. Reconstructing volume tracking. *Journal of Computational Physics*, Vol. 141, No. 2, pp. 112–152, 1998.

研究業績

論文

(1) 渡辺勢也，青木尊之，長谷川雄太，河原淳，橋本博公，”格子ボルツマン法による物体を含む自由界面流れの大規模シミュレーション”，日本混相流学会誌（Accepted）
(2) 渡辺勢也，青木尊之，都築怜理，”GPU を用いた個別要素法による粉体シミュレーションに対するメモリ使用量を抑えた高速化手法”，日本計算工学会論文集，日本計算工学会, 2016, Paper No.20160013, 2016
(3) 渡辺勢也，青木尊之，都築怜理，下川辺隆史，“接触による粒子間相互作用の GPU 計算での近傍探索手法”，情情報処理学会論文誌コンピューティングシステム（ACS)，情報処理学会，8-4, 50–60，2015

その他論文

(1) 渡辺勢也，青木尊之，都築怜理，”GPU による非球形粒子を用いた DEM の大規模シミュレーション”，粉体工学会誌，粉体工学会，52-12，730–734，2015（技術資料）

国際会議

(1) Seiya Watanabe, Takayuki Aoki, Yuta Hasegawa, “GPU-accelerated Lattice Boltzmann Simulations with Octree-based Dynamic AMR Method” , DSFD 2018, Worcester, MA, June, 2018
(2) Seiya Watanabe, Takayuki Aoki, Yuta Hasegawa, “Simulations of Free Surface Flows Interacting with Solid Objects by Lattice Boltzmann Method Using Multiple GPUs” , ECCM-ECFD 2018, Glasgow, UK, June, 2018
(3) Seiya Watanabe, Takayuki Aoki, Yuta Hasegawa, “Large-scale Simulations for Fluid-particle Systems Using Coupled LBM-DEM on a GPU Supercomputer”, PARTICLES 2017, Hannover, Germany, September, 2017
(4) Seiya Watanabe, Takayuki Aoki, Yuta Hasegawa, “Large-Scale Simulations For Multiphase Flows By Coupled LBM-DEM Using Multiple GPUs” , DSFD2017, Erlangen, Germany, July, 2017
(5) Seiya Watanabe, Takayuki Aoki, Yuta Hasegawa, “A large-scale LBM-DEM simulation based

on direct calculation of fluid-solid interactions using multiple GPUs”, COUPLED PROBLEMS 2017, Rhodes island, Greece, June, 2017

(6) Seiya Watanabe, Takayuki Aoki, Yuta Hasegawa, “Large-scale Simulations for Fluidization using Coupled Lattice Boltzmann Method and Discrete Element Method on a GPU Supercomputer”, ECCOMAS Congress 2016, Crete Island, Greece, June, 2016

(7) Seiya Watanabe, Takayuki Aoki, Yuta Hasegawa, “Large-scale Simulations for Fluid-particle System using Multiple GPUs”, Parallel CFD 2016, Kobe, Japan, May, 2016

(8) Seiya Watanabe, Takayuki Aoki, Satori Tsuzuki, “Large-scale DEM Simulations using Non-spherical Elements on GPU”, PARTICLES 2015, Barcelona, Spain, September, 2015

国内会議

(1) 渡辺勢也，青木尊之，長谷川雄太,”GPU スパコンにおける動的負荷分散を用いた AMR 法による大規模 LBM 計算”，日本流体力学会・第 32 回数値流体力学シンポジウム，東京，12 月，2018

(2) 渡辺勢也，青木尊之，松下慎太郎，Christian Conti，高木知宏,”AMR 法の動的領域分割へのマルチフェーズフィールド法の適用”，日本機械学会・第 31 回計算力学講演会，徳島，11 月，2018

(3) 渡辺勢也，青木尊之，長谷川雄太,”動的 AMR 法を導入した格子ボルツマン法の複数 GPU による大規模計算”，日本機械学会・第 31 回計算力学講演会，徳島，11 月，2018

(4) 渡辺勢也，青木尊之，長谷川雄太,”AMR 法を導入した格子ボルツマン法による大規模流体構造連成解析”，日本流体力学会・日本流体力学会年会 2018，大阪，9 月，2018

(5) 渡辺勢也，青木尊之，長谷川雄太，河原淳,”多数の瓦礫を含んだ津波が構造物へ与える影響の数値解析”， 日本混相流学会・混相流シンポジウム 2018，宮城，8 月，2018

(6) 渡辺勢也，青木尊之，黄遠雄，長谷川雄太，高木知弘,” マルチフェーズフィールド法に基づく領域分割を用いた AMR 計算の動的負荷分散”， 日本計算工学会・第 23 回計算工学講演会，名古屋，6 月，2018

(7) 渡辺勢也，青木尊之，長谷川雄太,”格子ボルツマン法を用いた浮遊物体を含む自由界面流れのシミュレーション”，日本流体力学会・第 31 回数値流体力学シンポジウム，京都，12 月，2017

(8) 渡辺勢也，青木尊之，長谷川雄太，松下真太郎,”Phase field 法による界面捕獲手法を用いた自由界面流れと固体粒子の連成解析”， 日本機械学会・第 30 回計算力学講演会，大阪，9 月，2017

(9) 渡辺勢也，青木尊之，長谷川雄太，松下真太郎,”自由界面流れと固体粒子の連成シミュレーション”，日本混相流学会・混相流シンポジウム 2017，東京，8 月，2017

(10) 渡辺勢也，青木尊之，長谷川雄太,”流体-粒子間相互作用の直接計算に基づく非球形粒子を含む流動層シミュレーション”，日本計算工学会・第 22 回計算工学講演会，埼玉，6 月，2017

(11) 渡辺勢也，青木尊之，長谷川雄太,”非球形粒子を含む流動層の大規模直接シミュレーション”，粉体工学会・2017 年度春期研究発表会，東京，5 月，2017

(12) 渡辺勢也，青木尊之，長谷川雄太,”格子ボルツマン法による舞い落ちるイチョウの葉の大規模流体-構造連成シミュレーション”，日本流体力学会・第 30 回数値流体力学シンポジウム，東京，12

月，2016

(13) 渡辺勢也，青木尊之，長谷川雄太,”流体－固体粒子間相互作用の直接計算による大規模流動化シミュレーション”， 化学工学会・第 22 回流動化・粒子プロセッシングシンポジウム，東京，12 月，2016

(14) 渡辺勢也，青木尊之，長谷川雄太,”LBM-DEM の直接相互作用計算による流動層の大規模シミュレーション”， 日本機械学会・第 29 回計算力学講演会，名古屋，9 月，2016

(15) 渡辺勢也，青木尊之，長谷川雄太,”舞い落ちる多数のイチョウの葉のシミュレーション”，日本計算工学会・第 21 回計算工学講演会，新潟，5 月，2016

(16) 渡辺勢也，青木尊之，長谷川雄太,”非球形固体粒子と流体の直接相互作用による混相流の大規模シミュレーション”，粉体工学会・2016 年度春期研究発表会，京都，5 月，2016

(17) 渡辺勢也，青木尊之，長谷川雄太,”LBM-DEM 固液二相流の GPU 計算”，日本流体力学会・第 29 回数値流体力学シンポジウム，博多，12 月，2015

(18) 渡辺勢也，青木尊之，都築怜理,”GPU による実形状粉体モデルを用いた大規模 DEM シミュレーション”，日本機械学会・第 28 回計算力学講演会，横浜，10 月，2015

(19) 渡辺勢也，青木尊之，都築怜理,”接触相互作用に基づく粒子法の GPU 計算におけるメモリアクセスの改善による高速化”，情報処理学会・第 149 回ハイパフォーマンスコンピューティング研究発表会，東京，6 月，2015

(20) 渡辺勢也，青木尊之，都築怜理，下川辺隆史,”GPU を用いた個別要素法計算における近傍探索手法の比較評価”， 日本計算工学会・第 21 回計算工学講演会，筑波，6 月，2015

(21) 渡辺勢也，青木尊之，都築怜理,”非球形粒子を用いた個別要素法による大規模粉体シミュレーション”, 粉体工学会・2015 年度春期研究発表会，東京，5 月，2015

(22) 渡辺勢也，青木尊之，都築怜理，下川辺隆史,”GPU による近接相互作用に基づく粒子計算の近傍探索手法”, 情報処理学会・第 205 回 ARC・第 147 回 HPC 合同研究発表会（HOKKE-22)，小樽，12 月，2014

受賞

(1) ベスト CFD グラフィックスアワード静止画部門 最優秀作品賞，日本流体力学会・第 32 回数値流体力学シンポジウム，2018 年 12 月 12 日

(2) Phase-Field Student Award，日本機械学会・第 31 回計算力学講演会，2018 年 11 月 24 日

(3) Best Presentation Award for Thermal Engineering, The 11th China-Japan-Korea Student Symposium, China, March, 2017

(4) 優秀講演賞，日本機械学会・第 29 回計算力学講演会，2016 年 9 月 23 日

(5) グラフィックスアワード MicroAVS 賞，日本計算工学会・第 21 回計算工学講演会，2016 年 6 月 1 日

(6) グラフィックスアワード MSC Apex 賞，日本計算工学会・第 21 回計算工学講演会，2016 年 6 月 1 日

(7) 東京工業大学 総合理工学研究科 創造エネルギー専攻 最優秀論文賞，2016 年 2 月 16 日

(8) ベスト CFD グラフィックスアワード動画部門第 2 位，日本流体力学会・第 29 回数値流体力学シンポジウム，2015 年 12 月 16 日

(9) 粉体工学会ベストプレゼンテーション賞（BP 賞），粉体工学会・2015 年度春期研究発表会，2015 年 5 月 19 日

職歴

(1) 日本学術振興会 特別研究員 DC2（2017 年 4 月-2019 年 3 月）

謝辞

本研究を進めるにあたり，熱心にご指導してくださった学術国際情報センター青木尊之教授に心から感謝申し上げます．修士課程と博士課程で青木先生からは研究の進め方や論文の書き方，申請書の書き方など研究者にとって必要不可欠なことを非常に多くの知恵や技術を教えて頂きました．また，ECCOMAS などの様々な国際会議へ参加する機会を頂き，貴重な体験をさせていただきました．この 5 年間は本当に充実してあっという間でしたが，青木研究室で学んだ事を糧にし，青木先生のように最前線で活躍する研究者を目指して行きたいと思います．来年以降も，精力的に研究を進めていきます．宜しくお願い致します．

科学研究費研究員の杉原健太さんには Povray での計算結果のリアリスティックな可視化に関して多くのことを教わりました．実は大学院進学のとき，青木研ホームページにある杉原さんのダムブレークの動画に感動し，青木研に進学しようと考えました．青木研究室の先輩である東京大学情報基盤センターの下川辺隆史准教授と日本原子力研究開発機構の小野寺直幸さんに感謝致します．修士課程で，論文の書き方や発表の仕方，大規模計算での実行性能測定や注意点など研究に必要な技術を下川辺さんは教えてくださいました．研究者としての心構えや，学会の懇親会などで教員や他大学の学生との繋がりを作ることの大切さを学べました．小野寺さんとは修士課程の最初の 1 年間だけ青木研究室で一緒でしたが，その 1 年間で数値解析に関して多くのことを教わりました．小野寺さんが青木研究室を離れてからも，電話や学会で格子ボルツマン法や AMR 法の GPU 実装に関して多くのアドバイスをしていただきました．

研究生活を事務的な面から支えてくださった青木研究室の秘書の高橋千恵子さんに感謝申し上げます．博士課程に進学してからは学会参加のための国内外の出張が多く，毎回の出張の手配やフライトの予約，学会費の精算などいろいろありがとうございました．

粒子法の実装や検証方法，最先端の粒子法の研究などについて多くのディスカッションをした都築さんに感謝致します．修士課程のときは，ご自身の研究で忙しいにも関わらず，論文や発表スライドを添削してていただき，毎回助かっていました．都築さんの熱心に研究に取り組む姿勢には本当に憧れています．チェスターさんには CUDA プログラミングやグラフィックス業界の最新の話題に関して教えて頂きました．研究員だった Conti さんには，SC18 の Tecnical Paper の投稿の際に英文の添削をしていただき，ありがとうございました．

同期の長谷川君と 5 年間お互いに切磋琢磨して研究を勧めていくことが出来て，嬉しく思います．学会などで一緒にいろいろなお酒を飲みに行けて楽しかったです．来年からもお互い頑張りましょう．修士課程で同期だった泉田君と杉山君，修士課程での日々の研究を楽しくしてくれてありがとうございま

した．泉田君はプレゼン準備や論文にとりかかるのがとても早くて，いつも見習いたいと思っていました．杉山君は数学と数値解析の理解がすごく，頼りにしていました．二人ともまた青木研究室に遊びに来てください．

後輩の松下君とは，界面捕獲法や AMR 法などに関して多くの議論をし，研究を進める上で大きな助けとなりました．同じ格子ボルツマン法が研究テーマである Yos さんとは，格子ボルツマン法での気液二相流計算に関してディスカッションできました．中国からの留学生の Kai 君はいつも格子ボルツマン法や GPU に関してたくさん質問してくれて，質問に答えることで私の理解も深めることが出来ました．修士 1 年の河原君には，私が開発した格子ボルツマン法コードを利用して津波シミュレーションを実行してもらい，私のコードの問題点などに気づくことが出来ました．また，共に研究をした青木研究室の他の学生に深く感謝致します．

最後に，いままでの学生生活を絶えず支えてくれた家族に深く感謝致します．好きなことをやらせてくれて博士進学に関しても背中を押してくれた父，一人暮らしでの食事や健康面をいつも心配してくれた母に対し心より感謝致します．