

論文 / 著書情報
Article / Book Information

題目(和文)	適合細分化格子法を導入した格子ボルツマン法による複雑形状物体周りの複数GPUを用いた空力解析
Title(English)	
著者(和文)	長谷川雄太
Author(English)	Yuta Hasegawa
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11123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,長崎 孝夫
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第11123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

適合細分化格子法を導入した格子ボルツ
マン法による複雑形状物体周りの複数
GPUを用いた空力解析

東京工業大学
工学院機械系機械コース

長谷川雄太

指導教員

青木尊之 教授

2019 年 1 月 29 日

目次

第 1 章	緒言	1
1.1	研究の背景	1
1.2	研究の目的	2
1.3	関連研究	5
1.4	論文の構成	5
第 2 章	格子ボルツマン法および適合細分化格子法	7
2.1	格子ボルツマン法	7
2.1.1	基礎方程式	7
2.1.2	格子ボルツマン法の適用範囲	10
2.1.3	衝突項の改良モデル	10
2.1.4	境界条件	18
2.1.5	抗力の計算	20
2.1.6	ラージエディシミュレーションの導入	20
2.2	適合細分化格子法	23
2.2.1	データ構造	23
2.2.2	時間刻みの取り扱い	24
2.2.3	時空間補間及びレベル変換	26
2.3	検証計算	29
2.3.1	Fourier 解析による数値安定性と精度の検証	29
2.3.2	Standing vortex	39
2.3.3	2 次元キャビティ流れ	42
2.3.4	3 次元球周りの流れ	48
2.3.5	平行平板間乱流	49
2.4	本章のまとめ	51
第 3 章	単一 GPU 計算の実装と高速化	55
3.1	CUDA フレームワークのプログラミングモデル	55

3.2	適合細分化格子法の GPU 実装と高速化手法	57
3.2.1	Naïve な実装	57
3.2.2	内部格子と外殻格子のカーネル分割および内部格子計算の高速化	59
3.2.3	マザーリーフ法による外殻格子点数の削減	60
3.2.4	C++ テンプレートによる外殻格子のカーネル生成	61
3.3	実行性能測定	65
3.3.1	実行時間の測定	66
3.3.2	Roofline モデルによる評価	67
3.4	本章のまとめ	68
第 4 章	複数 GPU 計算の実装	69
4.1	空間充填曲線に基づく領域分割法	69
4.1.1	Hilbert 曲線に基づく単一 Octree 内のリーフの走査	72
4.1.2	アコーディオン曲線による Forest-of-octree 内の Octree の走査	74
4.1.3	アコーディオン曲線と Hilbert 曲線の組み合わせによる Forest-of-octrees の リーフの走査	77
4.2	GPU 間通信の実装と最適化	80
4.2.1	袖領域のメモリ確保	80
4.2.2	GPU 間通信の最適化	80
4.2.3	データ転送量および通信時間の測定	82
4.3	計算性能測定	84
4.3.1	単純配置の格子における計算性能	86
4.3.2	複雑配置の格子における計算性能	87
4.4	本章のまとめ	88
第 5 章	球周りの流れの大規模計算によるドラッグ・クライシスの再現計算	89
5.1	計算条件	91
5.2	計算結果	92
5.3	本章のまとめ	94
第 6 章	自転車競技における集団走行の大規模空力解析	95
6.1	自転車の 3D モデル	96
6.2	計算条件	97
6.3	検証計算	102
6.3.1	単体走行の検証計算	102
6.3.2	集団走行の検証計算	102
6.4	集団走行の応用計算	106
6.4.1	8 人の集団の計算	106

6.4.2	2つの小集団で構成される16人の集団の計算	109
6.4.3	72人の集団の大規模計算	111
6.5	本章のまとめ	113
第7章	結言	115
付録A	Chapman-Enskog 展開	119
参考文献		127

目次

2.1	Discrete velocity model of Lattice Boltzmann Method	8
2.2	Bounce-back scheme.	18
2.3	Interpolated bounce-back scheme.	19
2.4	Data structure of octree-based Adaptive Mesh Refinement.	25
2.5	Refined mesh of rectangular computational domain with multiple octrees.	26
2.6	Multiple timestep sequences in refined mesh.	26
2.7	Two-dimensional spacial interpolation from coarse to fine mesh.	27
2.8	Coarse/fine distribution functions on the level-interface lattice.	29
2.9	Eigenvalues' ω - U dependency with $\tau = 0.5$	33
2.10	Eigenvalues' ω - τ dependency with $U = 0$	34
2.11	Exact and numerical solution of advection equation.	36
2.12	Error of Lattice Boltzmann Method for 1D advection equation.	37
2.13	Exact and numerical solution of diffusion equation.	38
2.14	Error of Lattice Boltzmann Method for 1D diffusion equation.	39
2.15	Initial vorticity profile of standing vortex.	40
2.16	Vorticity profile of staning vortex at $N = 129, t = 3$	40
2.17	Energy error of standing vortex.	41
2.18	Lid-driven cavity flow with oblique computational area.	42
2.19	Results of lid-driven cavity flow with oblique computational area at $Re = 100$. . .	43
2.20	Results of lid-driven cavity flow with oblique computational area at $Re = 1000$. . .	44
2.21	Results of lid-driven cavity flow with oblique computational area at $Re = 3200$. . .	44
2.22	Results of lid-driven cavity flow with oblique computational area at $Re = 5000$. . .	45
2.23	Results of lid-driven cavity flow with oblique computational area at $Re = 7500$. . .	45
2.24	Computational area of lid-driven cavity flow with 2-levels refined mesh.	46
2.25	Computational area of lid-driven cavity flow with 3-levels refined mesh.	47
2.26	Result of lid-driven cavity flow with 2-levels refined mesh ($Re = 1000$).	47
2.27	Result of lid-driven cavity flow with 3-levels refined mesh ($Re = 1000$).	48
2.28	Result of lid-driven cavity flow with 2-levels refined mesh ($Re = 7500$).	49

2.29	Drag coefficient of a sphere with various Reynolds numbers.	50
2.30	Computational condition of a flow in a parallel-plate channel.	50
2.31	Result of a turbulent channel flow with uniform mesh.	52
2.32	Result of a turbulent channel flow with adaptive mesh refinement.	53
3.1	Hierarchical structure of CUDA threads.	56
3.2	Example of warp divergence.	56
3.3	Coalescing of memory access in GPU.	58
3.4	Mother-leaf method (2D example).	62
3.5	Classification of leaf lattices based on stencil access patterns.	63
4.1	Example of space-filling curve on single octree.	70
4.2	Domain partitioning for forest-of-octrees by the space-filling curve using ijk-order and Morton curve.	71
4.3	Domain partitioning for forest-of-octrees by the space-filling curve using accordion curve and Hilbert curve (to be utilized in this study).	71
4.4	Binary indicator of 8 children of a node. (Left) octree expression of a node and its children. (Right) coordinates and binaries of the children.	73
4.5	Tracing order of 8 children regulated by terminal table $T_{\gamma i}^{\text{trm}}$ shown in Table 4.1. . .	74
4.6	Sample of octree-scan in forest-of-octrees (to be utilized in this study).	76
4.7	Example of the rectangular space-filling curve without/with in-out property matching. .	79
4.8	Example of the rectangular space-filling curve by the accordion curve and Hilbert curve.	79
4.9	Setup of halo leaves on multiple GPUs computation.	80
4.10	Optimized halo grids in the halo leaves.	81
4.11	Optimized halo grids with multiple timestep method.	82
4.12	Optimized lattice Boltzmann distribution functions by direction.	83
4.13	Complex refined mesh for performance measuring.	84
4.14	Performance of inter-GPU data transfer with each optimization.	85
4.15	Simple refined mesh for performance measuring.	86
4.16	Scalability of the computation on simple refined mesh.	87
4.17	Scalability of the computation on the complex refined mesh.	88
5.1	Drag coefficient of a sphere with the drag crisis on $Re \approx 2 \times 10^5$	90
5.2	Example of invalid result in the flow around a sphere at drag crisis with the coarse mesh ($\Delta x = D/25.6$).	90
5.3	Setup of the computational domain for the flow around a sphere at drag crisis. . . .	92
5.4	Numerical result of the drag coefficient on a sphere at drag crisis.	93

6.1	Specification of the 3D cyclist model in upright posture. TurboSquid, utilized with royalty free license [69].	96
6.2	Specification of the 3D cyclist model in drop posture. TurboSquid, utilized with royalty free license [69].	97
6.3	Computational domain and mesh near the cyclist with drop posture.	98
6.4	Computational mesh near the 4 cyclists with serial arrangement.	98
6.5	Time history of the drag area on the isolated cyclist with drop posture.	100
6.6	Frequency spectrum of the drag area on the cyclist with drop posture.	100
6.7	Velocity field behind a cyclist with different mesh resolution in the wake.	101
6.8	Result of the drag area on alone running cyclist in upright/drop posture.	103
6.9	Result of the drag area on the group running of 4 cyclists in a row.	104
6.10	Top view of the disturbed row arrangement appeared in Shirasaki et al. 2017 [73]: the trailing cyclist is X [m] behind from the other 3 riders in a row.	104
6.11	Top view of the disturbed row arrangement appeared in Shirasaki et al. 2017 [73]: the trailing cyclist is Y [m] laterally displaced from the other 3 riders in a row. . .	105
6.12	Result of the drag on the trailing cyclist with longitudinal distance X in the group of 4 cyclists.	105
6.13	Result of the drag on the trailing cyclist with lateral offset Y in the group of 4 cyclists.	106
6.14	Top view of the diamond arrangement appeared in Blocken et al. 2018 [67]	106
6.15	Drag ratio on each cyclist in the diamond arrangement to solo cyclist.	107
6.16	Arrangement of the in-line group of 8 cyclists.	107
6.17	Arrangement of the rhomboid group of 8 cyclists.	107
6.18	Result of the drag area and drag ratio on each cyclist in the in-line group of 8 cyclists.	108
6.19	Result of the drag area and drag ratio on each cyclist in the rhomboid group of 8 cyclists.	108
6.20	Result of drag area and drag ratio on each cyclist in the two closer in-line groups.	110
6.21	Result of drag area and drag ratio on each cyclist in the two closer rhomboid groups.	110
6.22	Result of drag area and drag ratio on each cyclist in the two in-line groups with 10 m displacement.	110
6.23	Result of drag area and drag ratio on each cyclist in the two rhomboid groups with 10 m displacement.	110
6.24	Result of drag area and drag ratio on each cyclist in the two in-line groups with 20 m displacement.	111
6.25	Result of drag area and drag ratio on each cyclist in the two rhomboid groups with 20 m displacement.	111

6.26	Arrangement of the large-scale running group of 72 cyclists.	112
6.27	Result of the drag area on each cyclist in the group of 72 cyclists.	112

表目次

2.1	Applicable scope of lattice Boltzmann method.	10
2.2	Energy error of standing vortex.	41
3.1	Execution environment.	65
3.2	Computational condition without mother-leaf method.	66
3.3	Computational condition with mother-leaf method.	66
3.4	Computation time on each tuning applied.	67
3.5	Results of performance.	68
4.1	Terminal table $T_{\gamma i}^{\text{trm}}$ for generating Hilbert curve.	72
4.2	Induction table $T_{\gamma i}^{\text{ind}}$ for generating Hilbert curve.	73

第 1 章

緒言

1.1 研究の背景

現代には自動車や自転車，鉄道をはじめ様々な移動手段が存在している．より早くかつ低コストな輸送を実現するため，あるいは，スポーツや興行，余暇に利用するため，乗り物の高速化や高効率化を目的とした研究開発や改良が続けられている．

古典力学に従えば水平移動に伴う仕事量は零であるが，現実には摩擦などの抵抗力を受けるため，散逸によって大量のエネルギーが消費される．資源エネルギー庁の総合エネルギー統計 [1] によれば，我が国の最終エネルギー消費のうち実に 2 割強もが運輸部門で占められており，輸送の効率化によるエネルギー消費の低減は極めて重要な課題である．また，自転車のような人力の移動手段においては，人体が発揮することのできるパワーの上限が存在するため，移動速度を上げるためには抵抗力によるエネルギー散逸を低減するほかない．この点においても，エネルギー消費の低減は重要である．

移動時にエネルギーを消費する要因として，地上を走る交通手段においては，(1) 静止時から走行速度を得るための慣性エネルギーへの転換，(2) 登坂に伴う位置エネルギーへの転換，(3) 地面との摩擦による散逸，(4) 空気抵抗（空気抗力とも呼ばれる）による散逸などが考えられるが，中でも影響が大きいのが (4) の空気抵抗である．(1) の慣性エネルギーへの転換は，軽量化によって低減することができる．(2) の位置エネルギーへの転換は，平地を長時間移動することを仮定すればほぼ無視できる．(3) の地面との摩擦については，人類は車輪を発明することでこれを克服した．車輪と地面との摩擦抵抗，いわゆる転がり抵抗は，走行速度によらずほぼ一定であるから走行速度を上げるほど影響は小さくなる．これらに対して (4) の空気抗力は走行速度の 2 乗に比例して増大するため，走行速度が高くなるほど影響は大きくなる．たとえば，自家用車では 60 km/h，自転車では 30 km/h 程度の速度に達すると空気抗力が無視できない大きさになると言われている．さらに，自転車の例では，40 km/h まで加速すると全抗力の 90% 以上を空気抗力が占めるほど深刻になる [2]．このように，空気抗力はあらゆる移動手段において無視できない要素であるため，走行時に発生する空気抵抗を低減することを目的とした空力解析が非常に重要となる．

空力の中でも低マッハの非圧縮性流れの解析は，自動車をはじめとする産業界からの需要が大きい．数値流体力学（Computational fluid dynamics, CFD）の典型的な解析対象であり，長年研究開発が行われ

てきたが、計算が困難な問題であり未だ改善の余地がある。困難さをもたらす最大の要因は乱流である。空気の動粘度はおよそ $10^{-5} \text{ m}^2/\text{s}$ のオーダーであるから、たとえば代表長さ 1 m 、代表速度 1 m/s 程度（歩行者くらいの遅い流れ）であっても、レイノルズ数は 10^5 のオーダーに達する。他の例を挙げると、たとえば自転車では速度がおよそ 1 桁上がるため $Re \approx 10^6$ 、自動車では更に 1 桁高くなり $Re \approx 10^7$ にもなる。このように、空気の流れは容易に乱流化するため、空力解析において乱流計算は必要不可欠となる。

他の数値計算の例に漏れず、乱流計算の発展は計算機の性能向上とともにあった。乱流場を最も精度良く計算できる Direct numerical simulation (DNS) は、3次元計算ではレイノルズ数の $9/4$ 乗に比例する膨大な格子点数が必要であることが知られており、多くの場合、計算資源の問題により計算不可能である。計算機が飛躍的な発展を遂げ、計算性能がエクサスケールに到達しようとしている現在においても、DNS による空力解析は困難を極める。このため乱流をモデル化し、より少ない資源で計算できる手法が必要である。乱流モデルとして最もよく用いられているのが Reynolds averaged Navier-Stokes simulation (RANS) である。RANS は乱流による流れ場の時間的変動の平均のみを扱う定常計算で、計算負荷が極めて小さいモデルである。物体境界には壁モデルが導入されることが多く、境界層を解像する必要がなく計算格子点数も非常に少なく済む。乱流モデルの中では最も少ない計算資源で解析を行うことができるもので、商用コード ANSYS Fluent [3] などで広く用いられてきた。しかし、複雑形状境界に対する壁の精度が十分でない場合もあり、より精度の高い Large eddy simulation (LES) への移行が求められている。LES は RANS に比べると多くの計算資源を必要とするものの、RANS の産業応用が本格的に始まった 1990 年代に比べて現在のコンピュータの性能は十万～百万倍にも向上しており、LES の実用化が進みつつある。このような RANS から LES への過渡期にあつて、高解像度の LES 計算を高効率に実行できる計算手法を実装し、実用的な大規模計算を実施することは、空力解析シミュレーションに新たなパラダイムをもたらすものとして極めて重要である。

1.2 研究の目的

本研究では、最新の高性能計算機において LES による複雑形状物体周りの非圧縮性流れの大規模空力解析を実現することを目的とする。解決すべき課題は以下のとおりである。

完全陽解法である格子ボルツマン法の導入の必要性

LES は RANS に比べると多くの格子点数が必要で、その数は、問題にも因るが数億、数十億、あるいは数百億ともいわれる。これほどの大規模計算を行うとなると、従来の流体計算手法ではいくつかの問題が生じる。

第1の問題は、半陰解法の反復計算の計算量が膨大となることである。非圧縮性流れの支配方程式は Navier-Stokes 方程式および連続の式であるが、このうち連続の式を満たすため一般に圧力のポアソン方程式が解かれる。これは格子点数 N と同じだけの次元を持つ連立一次方程式の解を反復法により求めるものであり、その計算量は、逐次アルゴリズムである SOR 法 (Successive over-relaxation) を用いた場合で $O(N^2)$ 、再帰アルゴリズムであるマルチグリッド法を導入した場合でも $O(N \log N)$ で増大する。

格子点数が増えるほど計算時間の増加が深刻になり、数十億や数百億を超える格子点数を扱う大規模計算の実施は困難となる。このため、計算量が $O(N)$ となる完全陽解法型の計算手法の導入が必要である。

非圧縮性流れは音速を無限大と仮定しており、局所的な圧力の変動は瞬時に全領域へと伝達される。したがって無限大の音速の極限においては、圧力ポアソン方程式を含む半陰解法型スキームを選択せざるを得ない。これを回避するには、流速に比べて十分に速いが無限大ではない有限の音速を仮定することが有効である。これは非圧縮性流れの弱圧縮近似（または低マッハ近似）と呼ばれるもので、圧力変動の伝達速度が音速で規定されるため、音速の CFL 条件が満たされる範囲において完全陽解法型のスキームを構築することが可能となる。低マッハ近似の手法のひとつで乱流計算において成功を収めているものに格子ボルツマン法 [4, 5] がある。格子ボルツマン法は、気体分子集団の統計的振る舞いを記述するボルツマン方程式を解くメゾスコピックなモデルであるが、Chapman-Enskog 展開 [6] により Navier-Stokes 方程式を近似的に満たすことが示されており、非圧縮性流れの計算に適用可能である。格子ボルツマン法の隣接参照はただか 1 つ隣の格子点までであるため、プログラムの実装上、高次精度差分を必要とする差分法や有限体積法よりもメモリ参照の局所性が高くなる利点がある。また、斜め方向の隣接点への参照が方程式に自然に含まれており、格子に沿わない複雑形状物体の境界条件を精度良く計算できることが期待される。これらの利点から、格子ボルツマン法は、チャネル乱流や多孔質内乱流 [7]、都市気流 [8, 9]、人体全域の血流 [10] など、様々な乱流計算に適用されている。そのほか、Exa 社の PowerFLOW [11] や Next Limit Dynamics 社の XFlow [12]、FluiDyna 社の ultraFluidX [13] などの商用コードにも格子ボルツマン法が組み込まれており、産業界からの関心も高まりつつあることが伺える。大規模空力解析を実現するという本研究の目的にも合致しており、格子ボルツマン法に基づく大規模 LES 計算による空力解析手法およびその計算コードの開発が必要である。

直交格子ベースの適合細分化格子法の必要性

LES において格子点数が膨大になることによる第 2 の問題は、格子生成に長い時間を要することである。従来、RANS では四面体要素による非構造格子が用いられてきたが、非構造格子は複雑形状物体に適合する格子を生成するために多大な時間を要する。格子生成の時間が計算のボトルネックになることも多く、数十億や数百億格子の規模では格子生成自体が困難になることも予想される。このため、大規模空力解析においては、格子生成が比較的簡単な構造格子を用いることが望ましい。

典型的な構造格子としては、計算領域全体に一樣な解像度の格子を割り当てる等間隔格子がある。しかし、空力においてスケールの小さい乱流渦が発生するのは物体近傍に限られ、遠方では波長の大きい流れが支配的である。等間隔格子では物体遠方の関心の低い領域にも過剰に細かい格子を割り当ててしまうため、格子点数に対する解像度の効率が悪い。そのほかの構造格子として不等間隔格子や円筒座標系の格子なども考えられるが、これらは複雑形状物体に適合する格子を割り当てることが困難であり、また、格子ボルツマン法において計算精度の低下を招くため、本研究の目的に対しては不適である。

格子ボルツマン法において高精度かつ高効率な空力解析を実現するためには、等間隔格子の規則的な格子配置を維持しながら、物体近傍に局所的に高い解像度を割り当てる格子が必要となる。この目的のため、本研究では適合細分化格子法 (Adaptive mesh refinement) [14, 15] を導入する。適合細分化格

子法には、物体境界などに合わせて細かい格子を割り当てる境界適合の静的な細分化、および、数値解の微細な変動を追跡して動的に解像度を変更する解適合の動的な細分化がある。本研究は、実在の複雑形状物体を対象として空力解析を実施することが目的であるから、物体近傍に細かい格子を割り当てる静的な細分化により計算可能である。また、格子の割り当て方としては、複雑形状物体に対して一定のアルゴリズムで自動的に細分化格子を生成できるものであって、格子生成に時間を要さないものが望ましい。これには木データ構造を用いた再帰的なアルゴリズム [16, 17] が有効である。計算領域を木構造に基づいて再帰的に分割していくもので、再帰深度が高い領域ほど細かい格子を割り当てることが可能となる。木構造に基づく適合細分化格子法により、格子点数あたりの格子解像度の効率を改善することができ、大規模空力解析コードの大幅な効率化が期待される。

複数 GPU 計算への実装の必要性

近年における計算機の発展の動向に鑑みて、大規模計算を行うためには Graphics processing unit (GPU) を用いた計算の実装は重要である。GPU はもともと画像処理専用に開発されたプロセッサであるが、CPU よりも演算性能やメモリアクセス速度に優れ、格子計算のような高並列計算において高い計算性能が得られることから、数値計算をはじめ様々な汎用計算に用いられるようになった。黎明期には Cg (C for Graphics) を始めとするシェーディング言語を用いた実装が用いられていたが、コンピュータグラフィックスの専門知識を必要とするためプログラミングの難易度が高かった。しかし、2006 年に NVIDIA 社から GPU の汎用計算プログラミングのフレームワークである CUDA [18] が発表されると、これがブレークスルーとなって GPU 計算が広く普及するようになった。2008 年に東京工業大学が世界初の GPU を搭載したスーパーコンピュータ TSUBAME1.2 の稼働を開始すると、他のスーパーコンピュータもこれを追随し、GPU は高性能計算 (High performance computing, HPC) の分野からも注目されるようになった。その後の 2010 年代に始まったビッグデータ処理や機械学習などのトレンドも GPU の普及を後押しし、今日までいくつもの GPU スーパーコンピュータが開発されてきた。世界のスーパーコンピュータの性能ランキングである Top500 [19] の最新版 (2018 年 11 月分) を参照すると、世界 1 位の性能を誇る米国 Oak Ridge National Laboratory の Summit, 同 2 位である米国 Lawrence Livermore National Laboratory の Sierra, 同 5 位である Swiss National Supercomputing Centre の Piz Daint, 同 7 位かつ日本国内 1 位である産業技術総合研究所の AI Bridging Cloud Infrastructure (ABCI) など、GPU を搭載したスーパーコンピュータが世界各国で稼働しており、その存在感を増していることがわかる。したがって、最新のスーパーコンピュータにおいて大規模空力解析を実現するためには、上で述べた格子ボルツマン法および適合細分化格子法を GPU 計算に実装することは必須である。また、スーパーコンピュータには数千台から数万台と多数の GPU が搭載されており、それらの性能を活用するためには、Message passing interface (MPI) により複数 GPU 計算コードを開発することも求められる。

上記の課題に対し本研究では、格子ボルツマン法に適合細分化格子法を導入し、複数 GPU に実装することで、最新の GPU スーパーコンピュータ上での大規模 LES 空力解析を実現することを目的とす

る．適宜実装の高速化を行い，応用計算において充分実用的な計算性能を達成することを目指す．また，計算コードの開発のみで終わるのではなく，実用的な応用計算を示すことで，本研究で開発する計算手法およびコードの有用性を示す． $Re \approx 10^5$ オーダーの空力の応用計算として，球周りの流れにおけるドラッグ・クライシスの再現計算を実施する．細分化格子を導入し，かつ多数の GPU を用いることにより，これまでの規模では得られなかったドラッグ・クライシスの再現計算を行う．さらに， $Re \approx 10^6$ オーダーの空力の応用計算として，自転車競技における集団走行の大規模空力解析を実施する．これまで広く RANS で行われてきた自転車競技の空力計算について，LES 解析への進展を目指すとともに，本手法のスポーツ工学への応用可能性を示す．

1.3 関連研究

本研究の主要な要素技術である「格子ボルツマン法」，「適合細分化格子法」，「複数 GPU 計算」のうち，2 つを組み合わせた研究はこれまでも広く行われてきた．格子ボルツマン法に適合細分化格子法を導入するものとしては，黎明期の 1998 年の Filippova らの研究 [20] や 2002 年の Yu らの研究 [21] などが挙げられるほか，2006 年の Rohde らの質量保存スキーム [22] や 2016 年の Kuwata らの乱流における Imbalance-correction 法 [23] など，計算精度を改善したスキームの開発が現在でも続けられている．格子ボルツマン法の複数 GPU 計算の実装は，等間隔格子では既に行われており，2011 年の Wang ら [24] の実装をはじめ，2013 年の小野寺らの都市気流計算 [8] のような実用計算への適用も達成されている．また，適合細分化格子法の GPU 実装の先行事例は数が少ないが，例えば 2010 年の Schive らの天体計算コード GAMER [25]，米国エネルギー省・Exascale Computing Project のフレームワーク AMReX [26, 27] が挙げられる．

「格子ボルツマン法」，「適合細分化格子法」，「複数 GPU 計算」をすべて含む研究は最近になるまで行われていなかった．本研究以外で上記 3 要素を含む事例としては，2017 年の Geier らの球周りの流れの計算 [28]，2018 年の Onodera らによる風況解析 [9]，渡辺の混相流計算 [29] が挙げられる．そのほか，商用コードの ultraFluidX [13] も，2019 年 1 月現在，複数 GPU 計算コードを開発中としている．いずれの事例においても複数 GPU 計算が実現されたのは最近 2 年以内であり，本研究と同様，現在も高速化や大規模化が進められている．また，これら関連研究と比較して，本研究の独自性としては，複雑形状物体を含む実用計算において，最大 192 GPU の大規模計算を達成していることが挙げられる．

1.4 論文の構成

本論文は以下の 7 章から構成される．第 1 章では，研究の背景および目的について述べた．第 2 章では，本研究で用いる数値計算手法である格子ボルツマン法および適合細分化格子法について説明し，検証計算の結果を示す．第 3 章では，NVIDIA 社の CUDA フレームワークを利用した GPU 計算の実装について述べる．適合細分化格子法を導入した格子ボルツマン法の実装について述べ，既存の最適化手法の適用および新しい高速化手法の提案と実装を行う．さらに，実行性能を測定し，高速化手法の有効性を評価する．第 4 章では，MPI (Message passing interface) を用いた複数 GPU 計算の実装を述べる．

適合細分化格子法における計算領域の各 GPU への割当の方法や袖領域間の通信の最適化について述べ、並列計算に対する性能測定を行う。第 5 章では、空力解析の典型的な検証問題である球周りの流れの計算について、複数 GPU による大規模計算の結果を示す。第 6 章では、本空力解析コードのスポーツ工学への応用として、自転車競技における集団走行の大規模計算を実施する。第 7 章では、本論文の結言を述べる。

第 2 章

格子ボルツマン法および適合細分化格子法

本章では、本研究で用いる計算手法である格子ボルツマン法および適合細分化格子法についての説明、および検証計算を示す。

2.1 格子ボルツマン法

2.1.1 基礎方程式

格子ボルツマン法の基礎モデルとして、等温・非圧縮性の単層流体のモデルについて説明する。格子ボルツマン法では、気体分子集団の統計的振る舞いを記述するボルツマン方程式を離散化することで基礎方程式を得る。連続空間上のボルツマン方程式は以下のように表される。

$$\frac{\partial f}{\partial t} + \boldsymbol{\xi} \cdot \nabla f = \Omega(f) - \boldsymbol{a} \cdot \frac{\partial f}{\partial \boldsymbol{\xi}} \quad (2.1)$$

f は位相空間上の気体分子の密度分布、 $\boldsymbol{x}, \boldsymbol{\xi}$ はそれぞれ位相空間の空間ベクトル、速度ベクトル、 t は時刻である。 $\boldsymbol{a}$ は外力により生じる加速度、 $\Omega(f)$ は分子同士の衝突を記述する項である。

格子ボルツマン法では、まず位相空間の速度ベクトルを有限個の要素 $\boldsymbol{c}_\alpha$ ($\alpha = 0, 1, \dots, Q-1$) に離散化する。本研究では、 $\boldsymbol{c}_\alpha$ として Fig. 2.1 に示す 2 次元 9 速度 (D2Q9) モデル

$$\boldsymbol{c}_\alpha = \begin{cases} (0, 0) & (\alpha = 0) \\ (\pm c, 0), (0, \pm c) & (1 \leq \alpha \leq 4) \\ (\pm c, \pm c) & (5 \leq \alpha \leq 8) \end{cases} \quad (2.2)$$

または、3 次元 27 速度 (D3Q27) モデル

$$\boldsymbol{c}_\alpha = \begin{cases} (0, 0, 0) & (\alpha = 0) \\ (\pm c, 0, 0), (0, \pm c, 0), (0, 0, \pm c) & (1 \leq \alpha \leq 6) \\ (\pm c, \pm c, 0), (0, \pm c, \pm c), (\pm c, 0, \pm c) & (9 \leq \alpha \leq 18) \\ (\pm c, \pm c, \pm c) & (19 \leq \alpha \leq 26) \end{cases} \quad (2.3)$$

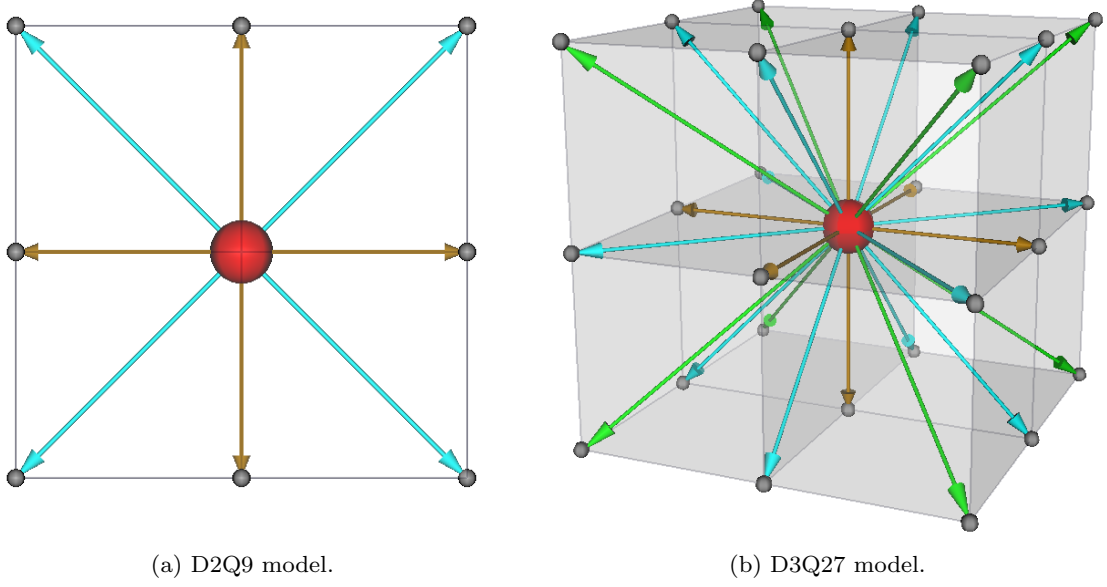


Fig. 2.1: Discrete velocity model of Lattice Boltzmann Method

を用いる．速度に掛かる定数 c は，格子ボルツマン法の擬似的な音速を規定する．

$$c_s = \frac{c}{\sqrt{3}} \quad (2.4)$$

擬似的な音速のために本モデルは圧縮性を有するが，マッハ数が 0.3 以下と十分に小さければ非圧縮性流体に近似することができる．実用的には，流れの最大速度 $U_{\max}$ に対して $c = U_{\max}/0.1$ や $c = U_{\max}/0.03$ などが選択される．なお，3 次元の速度モデルには，他にも D3Q27 から $9 \leq \alpha \leq 18$ を除いた 15 速度 (D3Q15) や， $19 \leq \alpha \leq 26$ を除いた 19 速度 (D3Q19) のモデルがあるが，乱流の計算において軸方向に依存する非対称性が出ることが報告されており [30]，高レイノルズ数の計算には不適であるため採用しない．以上のような速度方向の離散化により，下記の方方程式を得る．

$$\frac{\partial f_\alpha}{\partial t} + \mathbf{c}_\alpha \cdot \nabla f_\alpha = \Omega_\alpha + K_\alpha \quad (2.5)$$

式 (2.5) は離散ボルツマン方程式 (discrete Boltzmann equation) と呼ばれる． f_α は離散速度上での分子の密度分布であり，離散速度上では特に速度分布関数 (velocity distribution function) や確率密度関数 (probability distribution function)，または粒子分布関数 (particle distribution function) などと呼ばれる．また， Ω_α は衝突項， K_α は外力項 ($-\mathbf{a} \cdot \partial f / \partial \boldsymbol{\xi}$ に相当する項) である．

次に空間および時間の離散化を行う．速度定数 c に対し， $c = \Delta x / \Delta t$ を満たす一定の格子幅 Δx および時間刻み Δt を設定する．これにより，1 時刻ステップでの移流先がちょうど隣の格子点となるため，(2.5) の左辺にラグランジュ積分を適用することで次式が得られる．

$$f_\alpha(\mathbf{x} + \mathbf{c}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t) + \Omega_\alpha + K_\alpha \quad (2.6)$$

式 (2.6) は格子ボルツマン方程式 (lattice Boltzmann equation) と呼ばれる。衝突項は、流体の密度および速度を保存し、かつ区間 $[t, t + \Delta t]$ の時間内に生じる粘性散逸を記述する必要がある。衝突項のモデルにはいくつか種類があるが、最も簡単なものとして Bhatnagar-Gross-Krook (BGK) モデル [31, 4] がある。これは、気体分子が衝突を繰り返すことで分布関数が熱平衡状態に漸近すると仮定した衝突モデルである。格子ボルツマン方程式に BGK モデルを適用することから、格子 BGK モデル (lattice BGK model) と呼ばれる。格子 BGK モデルでは衝突項は以下のように表される。

$$\Omega_\alpha = -\frac{1}{\tau}(f_\alpha - f_\alpha^{\text{eq}}) \quad (2.7)$$

ここで、 τ は緩和時間係数であり、以下の式で与えられる。

$$\tau = \frac{1}{2} + \frac{\nu}{c_s^2 \Delta t} \quad (2.8)$$

ν は動粘度である。(2.8) の導出過程は煩雑であるため、詳細は付録 A で述べる。 f_α^{eq} は平衡速度分布関数で、気体分子の熱平衡分布である Maxwell-Boltzmann 分布を流速 $\mathbf{u}$ の 2 次で Taylor 展開した式から次のように定義される。

$$f_\alpha^{\text{eq}}(\rho, \mathbf{u}) = w_\alpha \rho \left(1 + \frac{\mathbf{c}_\alpha \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{c}_\alpha \cdot \mathbf{u})^2 - c_s^2 |\mathbf{u}|^2}{2c_s^4} \right) \quad (2.9)$$

ここで、 w_α は各速度方向の重みづけ係数である。 w_α は、総和計算が位相空間の速度方向全体に積分する操作と等価となるように、下記のように定義される。

$$\text{D2Q9: } w_\alpha = \begin{cases} 4/9, & |\mathbf{c}_\alpha| = 0 \\ 1/9, & |\mathbf{c}_\alpha| = 1 \\ 1/36, & |\mathbf{c}_\alpha| = \sqrt{2} \end{cases} \quad (2.10)$$

$$\text{D3Q27: } w_\alpha = \begin{cases} 8/27, & |\mathbf{c}_\alpha| = 0 \\ 2/27, & |\mathbf{c}_\alpha| = 1 \\ 1/54, & |\mathbf{c}_\alpha| = \sqrt{2} \\ 1/216, & |\mathbf{c}_\alpha| = \sqrt{3} \end{cases} \quad (2.11)$$

流体の密度 ρ や速度 $\mathbf{u}$ は、以下のように速度分布関数の総和から求められる。

$$\rho = \sum_{\alpha=0}^{Q-1} f_\alpha \quad (2.12)$$

$$\mathbf{u} = \frac{1}{\rho} \sum_{\alpha=0}^{Q-1} \mathbf{c}_\alpha f_\alpha \quad (2.13)$$

最後に、外力項 K_α は、以下のように定義される。外力によって加速度 $\mathbf{a}$ を印加された流体は、時刻刻み Δt の間に $\rho \mathbf{a} \Delta t$ だけ運動量を増加させるから、 K_α は

$$\sum_{\alpha=0}^{Q-1} \mathbf{c}_\alpha K_\alpha = \rho \mathbf{c}_\alpha \cdot \mathbf{a} \Delta t \quad (2.14)$$

Table 2.1: Applicable scope of lattice Boltzmann method.

	$Kn \ll 1$	$Kn \gtrsim 1$
$Ma \lesssim 0.3$	applicable	not applicable
$Ma > 0.3$	not applicable	not applicable

を満たす必要がある． K_α は一意には定まらないが，本研究では (2.14) を満たす組の一つとして，以下の定義を採用する．

$$K_\alpha = w_\alpha \rho \frac{\mathbf{c}_\alpha \cdot \mathbf{a} \Delta t}{c_s^2} \quad (2.15)$$

圧力は，格子ボルツマン法では陽に計算されない．比熱比 1 の流体を仮定することで，密度と音速の従属変数として次式で求められる．

$$p = c_s^2 \rho \quad (2.16)$$

2.1.2 格子ボルツマン法の適用範囲

格子ボルツマン法が Navier-Stokes 方程式に帰着されることは，Chapman-Enskog 展開 [6] により示されている．詳細については付録 A で述べるが，Chapman-Enskog 展開ではいくつかの仮定が置かれ，それにより格子ボルツマン法が適用できる流体の種類は制限される．まず，クヌーセン数は微小であることが仮定されており，すなわち $Kn \ll 1$ である．したがって適用範囲は連続流体に制限され，クヌーセン数の大きい流れである希薄気体には適用できない．また，非圧縮近似 ($\rho \approx \text{const.}$) が仮定されているため，非圧縮とみなすことのできる低マッハ数流れ ($Ma \lesssim 0.3$) にのみ適用可能であり，マッハ数の高い亜音速や超音速などの流れには適用できない．これらの関係を表にまとめると，Table 2.1 のようになる．

2.1.3 衝突項の改良モデル

格子ボルツマン法の BGK モデルはアルゴリズムが単純かつ精度が良いため，広く用いられてきたが，高レイノルズ数の計算において数値振動が増大し，計算が不安定となることが指摘されていた．乱流における数値安定性の改善のため，後年になっていくつかの改良モデルが提案されている．Charman-Enskog 展開によれば，分布関数の総和系のうち 2 次以下の項 ($\sum f_\alpha, \sum c_{\alpha i} f_\alpha, \sum c_{\alpha i} c_{\alpha j} f_\alpha$) のみが巨視的方程式に関与するのであって，3 次以上の項 ($\sum c_{\alpha i} c_{\alpha j} c_{\alpha k} f_\alpha, \sum c_{\alpha i} c_{\alpha j} c_{\alpha k} c_{\alpha l} f_\alpha, \dots$) は巨視的量に無関係である．したがって 3 次以上の項に対する衝突緩和係数は式 (2.8) に依らず任意の数を取ることができ，それらを自由に調整することで計算精度を保ったまま数値安定性を高めることができる．速度分布関数のままでは各次数の項が混ざり合っていて緩和係数を分離することができないため，速度分布関数をモーメント（次数ごとに分離された項）に変換し，モーメント空間で衝突緩和を計算して最後に速度分布関

数に逆変換する，というのが改良モデルの基本的な手順である．本項では，改良モデルとして代表的な MRT モデルおよびキュムラントモデルについて説明する．

Multiple-relaxation-time モデル

Multiple-relaxation-time (MRT) モデルは，衝突項の改良モデルの中でも初期に登場した手法である [32, 33]．MRT モデルでは，速度分布関数を直接緩和させるのではなく，モーメントに変換し，それぞれのモーメントを各々異なる衝突緩和係数で緩和し，速度分布関数に戻す操作を行う．ここでいうモーメントとは，速度分布関数の総和系であり，たとえば，密度

$$\rho = \sum_{\alpha=0}^{Q-1} f_{\alpha}, \quad (2.17)$$

運動量

$$j_x = \sum_{\alpha=0}^{Q-1} c_{\alpha x} f_{\alpha}, \quad j_y = \sum_{\alpha=0}^{Q-1} c_{\alpha y} f_{\alpha}, \quad j_z = \sum_{\alpha=0}^{Q-1} c_{\alpha z} f_{\alpha}, \quad (2.18)$$

応力テンソルに対応する項

$$p_{xy} = \sum_{\alpha=0}^{Q-1} c_{\alpha x} c_{\alpha y} f_{\alpha}, \quad p_{yz} = \sum_{\alpha=0}^{Q-1} c_{\alpha y} c_{\alpha z} f_{\alpha}, \dots \quad (2.19)$$

などがある．速度分布関数からモーメントへの変換は，線形写像であり，また逆変換が存在している必要がある．このような変換は行列の形式で簡潔に記述することができるため，MRT モデルもまた行列で記述される．MRT 衝突項は，最も簡潔に書くと，次の式で表される．

$$|\Omega\rangle = |f\rangle - M^{-1} \hat{S} M (|f\rangle - |f^{\text{eq}}\rangle) \quad (2.20)$$

式 (2.20) について，D2Q9 を例に詳しく説明する．まず，ケットベクトルは速度分布関数またはモーメントを並べた列ベクトルであり，

$$|f\rangle := (f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8)^{\text{T}} \quad (2.21)$$

のように定義される．ケットベクトルを定義することにより，モーメントへの変換を行列の形式で記述できるようになる．たとえば，密度は

$$\rho = \langle \rho | f \rangle, \quad (2.22)$$

$$\langle \rho | := \langle 1 | = (1, 1, 1, 1, 1, 1, 1, 1, 1) \quad (2.23)$$

と書き換えることができるし，運動量は

$$j_x = \langle j_x | f \rangle, \quad (2.24)$$

$$\langle j_x | := \langle c_x | = (0, 1, -1, 0, 1, -1, -1, 1, 1) \quad (2.25)$$

などとすることができる．同様にして計9個の線形独立なモーメントを用意すれば，速度分布関数とモーメントの変換行列が定義できる．D2Q9において，Lallemand ら [32] の定義に従い，モーメントの組を

$$|m\rangle := (\rho, e, \varepsilon, j_x, q_x, j_y, q_y, p_{xx}, p_{xy})^\top, \quad (2.26)$$

$$\langle e| := \langle -4 + 3(c_x^2 + c_y^2)|, \quad (2.27)$$

$$\langle \varepsilon| := \left\langle 4 - \frac{21}{2}(c_x^2 + c_y^2) + \frac{9}{2}(c_x^2 + c_y^2)^2 \right|, \quad (2.28)$$

$$\langle j_x| := \langle c_x|, \quad (2.29)$$

$$\langle q_x| := \langle (-5 + 3(c_x^2 + c_y^2))c_x|, \quad (2.30)$$

$$\langle j_y| := \langle c_y|, \quad (2.31)$$

$$\langle q_y| := \langle (-5 + 3(c_x^2 + c_y^2))c_y|, \quad (2.32)$$

$$\langle p_{xx}| := \langle c_x^2 - c_y^2|, \quad (2.33)$$

$$\langle p_{xy}| := \langle c_x c_y|. \quad (2.34)$$

とすると，速度分布関数からモーメントへの変換を

$$|m\rangle = M |f\rangle, \quad (2.35)$$

$$M = \begin{pmatrix} \langle \rho| \\ \langle e| \\ \langle \varepsilon| \\ \langle j_x| \\ \langle q_x| \\ \langle j_y| \\ \langle q_y| \\ \langle p_{xx}| \\ \langle p_{xy}| \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ -4 & -1 & -1 & -1 & -1 & 2 & 2 & 2 & 2 \\ 4 & -2 & -2 & -2 & -2 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & -1 & 0 & 1 & -1 & -1 & 1 \\ 0 & -2 & 0 & 2 & 0 & 1 & -1 & -1 & 1 \\ 0 & 0 & 1 & 0 & -1 & 1 & 1 & -1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 1 & 1 & -1 & -1 \\ 0 & 1 & -1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & -1 \end{pmatrix}. \quad (2.36)$$

と書くことができる． M はモーメント変換行列である．次に，得られたモーメントのそれぞれに対して BGK 衝突項の計算を適用する．BGK 衝突項では密度 ρ および運動量 j_x, j_y は保存されるから，残りの6モーメントについて衝突緩和が行われる．衝突緩和後のモーメントを上付き添字*で表すとする，衝突項は次のように書くことができる．

$$m_\beta^* = m_\beta - s_\beta(m_\beta - m_\beta^{\text{eq}}) \quad (\beta = 0, 1, 2, \dots, 8) \quad (2.37)$$

あるいは，行列で書けば，以下のようになる．

$$|m^*\rangle = |m\rangle - \hat{S}(|m\rangle - |m^{\text{eq}}\rangle) \quad (2.38)$$

ここで， $|m^{\text{eq}}\rangle$ は平衡モーメントであり，平衡分布関数のモーメントとして与えられる．

$$|m^{\text{eq}}\rangle = M |f_\alpha^{\text{eq}}\rangle \quad (2.39)$$

$\hat{S}$ は衝突緩和係数行列であり、次のように定義される.

$$\hat{S} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & s_1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & s_2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & s_4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & s_6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & s_8 \end{pmatrix} \quad (2.40)$$

緩和係数 $s_1, s_2, s_4, s_6, s_7, s_8$ のうち, s_7, s_8 (p_{xx}, p_{xy} の緩和係数) は流体の応力テンソルに対応する項であるため, 動粘度から式 (2.8) によって決定される. 残りの係数 s_1, s_2, s_4, s_6 については任意の数を選ぶことができる. 高レイノルズ数条件において安定な係数の組を選べば, 以下のようになる.

$$s_1 = s_2 = 1.4, \quad s_4 = s_6 = 1.2, \quad s_7 = s_8 = \tau^{-1}. \quad (2.41)$$

以上により, モーメント空間での BGK 衝突項が記述された. 最後に, 衝突後のモーメントを速度分布関数に逆変換することで MRT モデルによる衝突緩和計算が完了する.

$$|f^*\rangle = M^{-1} |m^*\rangle \quad (2.42)$$

ただし,

$$|f^*\rangle = |f\rangle + |\Omega\rangle \quad (2.43)$$

である.

3 次元の D3Q27 モデルにおいても同様に, 27 個の線形独立なモーメントを用意することで MRT 衝突項を構成することができる. D2Q9 に比べるとモーメントや緩和係数の選択に任意性があるが, D3Q27 における有効な行列のうちの一種が Suga ら [7] によって導出されている. Suga らの導出によれば, モー

[illegible]

$$s_5 = s_7 = \tau^{-1},$$

ただし, $\text{diag}(\cdot)$ は, 引数を対角成分に持つ対角行列である.

キュムラントモデルは、2015 年に Geier ら [34] によって提案された、比較的新しい衝突モデルである。MRT モデルから更に数値安定性が向上しており、MRT モデルでは計算が破綻するような極めて高いレイノルズ数の流れも安定的に計算することができる。キュムラントモデルでは、

- の 2 点により，数値安定性を大幅に改善している．それぞれ以下で説明する．まず，中心モーメントとは，モーメントに変換する際に分子速度の累乗を掛けて総和を取るのではなく，分子速度と流速の差の累乗を掛けて総和を取ったものである．すなわち，MRT で用いられるモーメントが

$$m_{lmn} = \sum_{\alpha=0}^{Q-1} c_{\alpha x}^l c_{\alpha y}^m c_{\alpha z}^n f_{\alpha} \quad (2.47)$$

と定義されるのに対し，中心モーメントは

$$\kappa_{lmn} = \sum_{\alpha=0}^{Q-1} (c_{\alpha x} - u)^l (c_{\alpha y} - v)^m (c_{\alpha z} - w)^n f_{\alpha} \quad (2.48)$$

と定義される． $l, m, n \in \mathbb{N}$ はそれぞれモーメントまたは中心モーメントの x, y, z 軸方向の次数である．次に，キュムラントについて説明する．モーメントもキュムラントも，分布関数の或る種の統計的性質を表す量であるが，キュムラントは特に確率変数の統計的独立性が最も高くなるように設計された統計量である．また，Gaussian 分布において 3 次以上のキュムラントがすべて 0 になるという性質を持つ．速度分布関数 f_{α} に対するキュムラントは，分子速度の変数として速度分布関数のラプラス変換を求め，その自然対数を母関数とし Taylor 展開することで得られる．すなわち，

$$F(\Xi, \Upsilon, Z) = \mathcal{L}\{f_{\alpha}\} = \sum_{\alpha=0}^{Q-1} f_{\alpha} e^{-(\Xi c_{\alpha x} + \Upsilon c_{\alpha y} + Z c_{\alpha z})}, \quad (2.49)$$

$$M(\Xi, \Upsilon, Z) = \ln F(\Xi, \Upsilon, Z), \quad (2.50)$$

$$C_{lmn} = c^{-l-m-n} \left. \frac{\partial^l \partial^m \partial^n M(\Xi, \Upsilon, Z)}{\partial \Xi^l \partial \Upsilon^m \partial Z^n} \right|_{\Xi=\Upsilon=Z=0} \quad (2.51)$$

である．D3Q27 においては， $l, m, n \in \{0, 1, 2\}^3$ とすれば 27 個の独立なキュムラントが定義でき，MRT と同様に変換および逆変換が可能である．

式 (2.51) は速度分布関数 f_{α} の従属変数として展開できるため，実装上は微係数を求めるような操作は必要ない．キュムラントへの変換，逆変換および衝突項の実装は Geier らによって導出されており，下記の手順で行われる．まず，変換に必要な流体の密度，流速を求める．

$$\rho = \sum_{\alpha=0}^{Q-1} f_{\alpha}, \quad (2.52)$$

$$\mathbf{u} = (u, v, w)^{\top} = \frac{1}{\rho} \left(\frac{\mathbf{a} \Delta t}{2} + \sum_{\alpha=0}^{Q-1} \mathbf{c}_{\alpha} f_{\alpha} \right). \quad (2.53)$$

ここで， $\mathbf{a}$ は加速度である．次に，密度，流速を用いて中心モーメントを求める．式 (2.48) で求めてもよいが，総和計算を軸方向ごとに分離することで，より少ない計算量で中心モーメントを求めることができる．

$$\kappa_{ij|n} = \sum_k f_{ijk} \left(\frac{c_{\alpha z}}{c} - \frac{w}{c} \right)^n, \quad (2.54)$$

$$\kappa_{i|mn} = \sum_j \kappa_{ij|n} \left(j - \frac{w}{c} \right)^m, \quad (2.55)$$

$$\kappa_{lmn} = \sum_i \kappa_{i|mn} \left(k - \frac{w}{c} \right)^l. \quad (2.56)$$

これらは、以下のように行列形式で書くこともできる。

$$\begin{pmatrix} \kappa_{ij|0} \\ \kappa_{ij|1} \\ \kappa_{ij|2} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ -1 - w/c & -w/c & 1 - w/c \\ (-1 - w/c)^2 & (-w/c)^2 & (1 - w/c)^2 \end{pmatrix} \begin{pmatrix} f_{ij\bar{1}} \\ f_{ij0} \\ f_{ij1} \end{pmatrix} \quad (2.57)$$

$$\begin{pmatrix} \kappa_{i|0n} \\ \kappa_{i|1n} \\ \kappa_{i|2n} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ -1 - v/c & -v/c & 1 - v/c \\ (-1 - v/c)^2 & (-v/c)^2 & (1 - v/c)^2 \end{pmatrix} \begin{pmatrix} \kappa_{i\bar{1}|n} \\ \kappa_{i0|n} \\ \kappa_{i1|n} \end{pmatrix} \quad (2.58)$$

$$\begin{pmatrix} \kappa_{0mn} \\ \kappa_{1mn} \\ \kappa_{2mn} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ -1 - u/c & -u/c & 1 - u/c \\ (-1 - u/c)^2 & (-u/c)^2 & (1 - u/c)^2 \end{pmatrix} \begin{pmatrix} \kappa_{\bar{1}|mn} \\ \kappa_{0|mn} \\ \kappa_{1|mn} \end{pmatrix} \quad (2.59)$$

ここで、添字の i, j, k は速度分布関数の速度方向の x, y, z 軸方向成分で、 $(i, j, k) \in \{-1, 0, 1\}^3$ である。また、 $\bar{1} = -1$ である。 f_{ijk} は、 $\mathbf{c}_\alpha = \mathbf{c}(i, j, k)^\top$ を満たす添字 α の速度分布関数 f_α である。次に、中心モーメントからキュムラントへの変換を行う。0 次、1 次のキュムラントは衝突項に影響しないため、求める必要はない。2 次、3 次のキュムラントは、中心モーメントに一致する。

$$C_{110} = \kappa_{110}, \quad (2.60)$$

$$C_{200} = \kappa_{200}, \quad (2.61)$$

$$C_{120} = \kappa_{120}, \quad (2.62)$$

$$C_{111} = \kappa_{110}. \quad (2.63)$$

4 次以上のキュムラントは中心モーメントの非線形結合として与えられ、以下のように求められる。

$$C_{211} = \kappa_{211} - \frac{\kappa_{200}\kappa_{011} + 2\kappa_{110}\kappa_{101}}{\rho}, \quad (2.64)$$

$$C_{220} = \kappa_{220} - \frac{\kappa_{220}\kappa_{020} + 2\kappa_{110}^2}{\rho}, \quad (2.65)$$

$$C_{122} = \kappa_{122} - \frac{\kappa_{002}\kappa_{120} + \kappa_{020}\kappa_{102} + 4\kappa_{011}\kappa_{111} + 2(\kappa_{101}\kappa_{021} + \kappa_{110}\kappa_{012})}{\rho}, \quad (2.66)$$

$$\begin{aligned} C_{222} = & \kappa_{222} - (4\kappa_{111}^2 + \kappa_{200}\kappa_{022} + \kappa_{020}\kappa_{202} + \kappa_{002}\kappa_{220} + 4(\kappa_{011}\kappa_{211} + \kappa_{101}\kappa_{121} + \kappa_{110}\kappa_{112})) \\ & + \frac{2(\kappa_{120}\kappa_{102} + \kappa_{210}\kappa_{012} + \kappa_{201}\kappa_{021})}{\rho} \\ & + \frac{16\kappa_{110}\kappa_{101}\kappa_{011} + 4(\kappa_{101}^2\kappa_{020} + \kappa_{011}^2\kappa_{200} + \kappa_{110}^2\kappa_{002}) + 2\kappa_{200}\kappa_{020}\kappa_{002}}{\rho^2}. \end{aligned} \quad (2.67)$$

なお、添字の順序を並べ替えたキュムラントは、右辺の中心モーメントの添字を同じ順序に並べ替えることで得られる。たとえば、 C_{011} は式 (2.60) より

$$C_{011} = \kappa_{011} \quad (2.68)$$

であり、他のキュムラントも同様である。次に、次式で表されるキュムラント衝突項を計算する。

$$C_{lmn}^* = C_{lmn} + (1 - \omega_{lmn})C_{lmn}^{\text{eq}} \quad (2.69)$$

C_{lmn}^* は衝突後のキュムラント, ω_{lnm} は各キュムラントの緩和係数, C_{lmn}^{eq} は平衡キュムラントである. 緩和係数は 2 次のキュムラントの非対角成分において τ^{-1} , 他のキュムラントにおいて 1 である. 平衡キュムラントは, 2 次の対角成分において

$$C_{200}^{\text{eq}} = C_{020}^{\text{eq}} = C_{002}^{\text{eq}} = \frac{\rho}{3} \quad (2.70)$$

であり, 他の平衡キュムラントは 0 である. 衝突計算ののち, キュムラントから中心モーメントを経由して, 速度分布関数への逆変換を行う. キュムラントから中心モーメントへの変換は, 以下の式で表される.

$$\kappa_{110}^* = C_{110}^*, \quad (2.71)$$

$$\kappa_{200}^* = C_{200}^*, \quad (2.72)$$

$$\kappa_{120}^* = C_{120}^*, \quad (2.73)$$

$$\kappa_{111}^* = C_{111}^*, \quad (2.74)$$

$$\kappa_{211}^* = C_{211}^* - \frac{\kappa_{200}^* \kappa_{011}^* + 2\kappa_{110}^* \kappa_{101}^*}{\rho}, \quad (2.75)$$

$$\kappa_{220}^* = C_{220}^* - \frac{\kappa_{220}^* \kappa_{020}^* + 2(\kappa_{110}^*)^2}{\rho}, \quad (2.76)$$

$$\kappa_{122}^* = C_{122}^* - \frac{\kappa_{002}^* \kappa_{120}^* + \kappa_{020}^* \kappa_{102}^* + 4\kappa_{011}^* \kappa_{111}^* + 2(\kappa_{101}^* \kappa_{021}^* + \kappa_{110}^* \kappa_{012}^*)}{\rho}, \quad (2.77)$$

$$\begin{aligned} \kappa_{222}^* = & C_{222}^* - (4(\kappa_{111}^*)^2 + \kappa_{200}^* \kappa_{022}^* + \kappa_{020}^* \kappa_{202}^* + \kappa_{002}^* \kappa_{220}^* + 4(\kappa_{011}^* \kappa_{211}^* + \kappa_{101}^* \kappa_{121}^* + \kappa_{110}^* \kappa_{112}^*)) \\ & + \frac{2(\kappa_{120}^* \kappa_{102}^* + \kappa_{210}^* \kappa_{012}^* + \kappa_{201}^* \kappa_{021}^*)}{\rho} \\ & + \frac{16\kappa_{110}^* \kappa_{101}^* \kappa_{011}^* + 4((\kappa_{101}^*)^2 \kappa_{020}^* + (\kappa_{011}^*)^2 \kappa_{200}^* + (\kappa_{110}^*)^2 \kappa_{002}^*) + 2\kappa_{200}^* \kappa_{020}^* \kappa_{002}^*}{\rho^2}. \end{aligned} \quad (2.78)$$

1 次の中心モーメントは, 符号を反転させることで, 外力項 (式 (2.6) の K_α) に相当する速度の変分 ($\Delta \mathbf{u} / \Delta t = \mathbf{a} \Delta t$) を記述することができる.

$$\kappa_{100}^* = -\kappa_{100} \quad (2.79)$$

最後に, 中心モーメントから速度分布関数への逆変換は, 式 (2.57), (2.58), (2.59) の逆行列を求めることで, 以下のように計算される.

$$\begin{pmatrix} \kappa_{1|mn}^* \\ \kappa_{0|mn}^* \\ \kappa_{1|mn}^* \end{pmatrix} = \begin{pmatrix} 1 - (u/c)^2 & -2 & -1 \\ ((u/c)^2 - u/c)/2 & (2u/c - 1)/2 & 1/2 \\ ((u/c)^2 + u/c)/2 & (2u/c + 1)/2 & 1/2 \end{pmatrix} \begin{pmatrix} \kappa_{0mn}^* \\ \kappa_{1mn}^* \\ \kappa_{2mn}^* \end{pmatrix} \quad (2.80)$$

$$\begin{pmatrix} \kappa_{i1|n}^* \\ \kappa_{i0|n}^* \\ \kappa_{i1|n}^* \end{pmatrix} = \begin{pmatrix} 1 - (v/c)^2 & -2 & -1 \\ ((v/c)^2 - v/c)/2 & (2v/c - 1)/2 & 1/2 \\ ((v/c)^2 + v/c)/2 & (2v/c + 1)/2 & 1/2 \end{pmatrix} \begin{pmatrix} \kappa_{i|0n}^* \\ \kappa_{i|1n}^* \\ \kappa_{i|2n}^* \end{pmatrix} \quad (2.81)$$

$$\begin{pmatrix} f_{ij1}^* \\ f_{ij0}^* \\ f_{ij1}^* \end{pmatrix} = \begin{pmatrix} 1 - (w/c)^2 & -2 & -1 \\ ((w/c)^2 - w/c)/2 & (2w/c - 1)/2 & 1/2 \\ ((w/c)^2 + w/c)/2 & (2w/c + 1)/2 & 1/2 \end{pmatrix} \begin{pmatrix} \kappa_{ij0}^* \\ \kappa_{ij1}^* \\ \kappa_{ij2}^* \end{pmatrix} \quad (2.82)$$

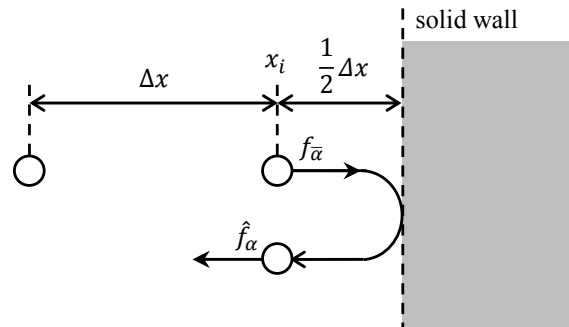


Fig. 2.2: Bounce-back scheme.

2.1.4 境界条件

Bounce-back

Bounce-back 条件は、すべりなし壁に対する境界条件として広く用いられている境界条件である。壁から流体へと流れ込む分布関数 $\hat{f}_{\alpha}$ として、流体側の格子点で逆向きの速度を持つ分布関数の値を代入する。

$$\hat{f}_{\alpha}(\mathbf{x}, t + \Delta t) = f_{\bar{\alpha}}(\mathbf{x}, t) \quad (2.83)$$

ただし、 $\bar{\alpha}$ は $\mathbf{c}_{\bar{\alpha}} = -\mathbf{c}_{\alpha}$ を満たす速度方向である。このスキームを模式的に表すと、Fig. 2.2 のようになる。速度分布関数は粒子的な振る舞いにより壁で反射し、逆向きの速度で元の格子点に戻ってくる。

Bounce-back 条件は静止壁だけでなく移動壁も扱うことが出来る [35]。壁面位置での流速が壁の移動速度に一致するように、移動壁で反射する分布関数に対して次式のような外力項を加える。

$$\hat{f}_{\alpha}(\mathbf{x}, t + \Delta t) = f_{\bar{\alpha}}(\mathbf{x}, t) + 2w_{\alpha}\rho(\mathbf{x}, t)\frac{\mathbf{c}_{\alpha} \cdot \mathbf{u}_{\text{wall}}}{c_s^2} \quad (2.84)$$

$\mathbf{u}_{\text{wall}}$ は壁面の移動速度である。なお、 $\mathbf{u}_{\text{wall}} = \mathbf{u}(\mathbf{x}, t)$ とすることで、滑り壁の境界条件を与えることもできる。

Interpolated bounce-back

Bounce-back 境界は非常に簡便なスキームであるが、格子点からちょうど $\Delta x/2$ の距離にある壁しか扱うことができないため、複雑形状物体を表現するときの精度が悪いという問題がある。境界条件の精度を改善するため、Bounce-back に補間関数を取り入れた Interpolated bounce-back が Bouzidi ら [36] から提案されている。概念図を Fig. 2.3 に示す。横方向は座標位置を表す。実線の丸印は、格子点上の値、または、その値が移流した先の値を表し、破線の丸印は補間によって求められた値であることを示す。また、破線の矢印は補間操作を表し、始点には補間で用いる値、終点には補間で求められる値が示されている。Interpolated bounce-back では、格子点と壁との距離が格子幅の半分よりも小さい場合と大

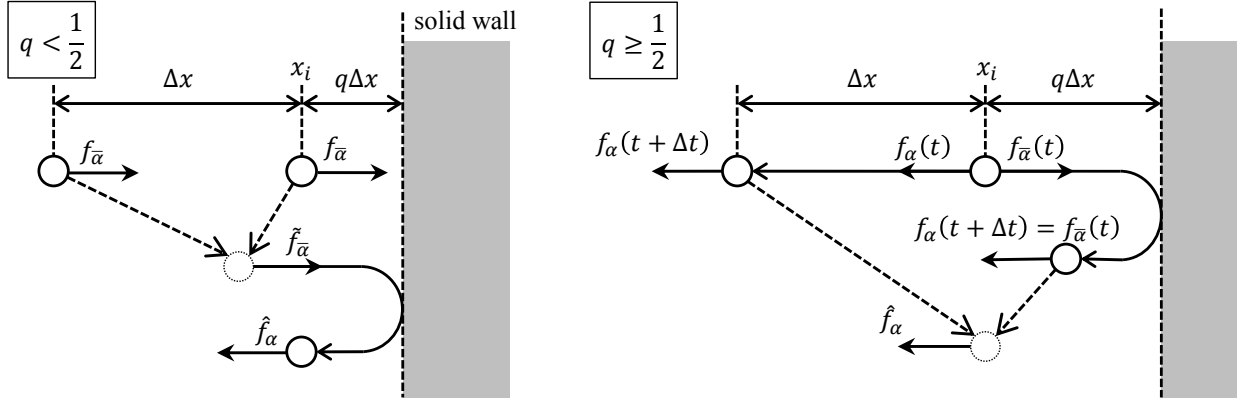


Fig. 2.3: Interpolated bounce-back scheme.

きい場合で処理が異なる． $q < 1/2$ のときは，壁からの反射を経てちょうど格子点上に移動してくるような位置の分布関数を空間補間によって求め，それを移流させることで壁から流体の方向に移流する速度分布関数を求める． $q \geq 1/2$ のときは，まず速度分布関数を移流させ，移流後の速度分布関数を空間補間することで速度分布関数を求める．これらを式で表すと以下のようになる．

$$f_{\alpha}(\mathbf{x}, t + \Delta t) = 2q f_{\bar{\alpha}}(\mathbf{x}, t) + (1 - 2q) f_{\bar{\alpha}}(\mathbf{x} + \mathbf{c}_{\alpha} \Delta t, t) + 2w_{\alpha} \rho \frac{\mathbf{c}_{\alpha} \cdot \mathbf{u}_{\text{wall}}}{c_s^2} \quad \left(q < \frac{1}{2} \right), \quad (2.85)$$

$$\hat{f}_{\alpha}(\mathbf{x}, t + \Delta t) = \frac{1}{2q} f_{\bar{\alpha}}(\mathbf{x}, t) + \frac{2q - 1}{2q} f_{\alpha}(\mathbf{x}, t) + \frac{1}{2q} \left(2w_{\alpha} \rho \frac{\mathbf{c}_{\alpha} \cdot \mathbf{u}_{\text{wall}}}{c_s^2} \right) \quad \left(q \geq \frac{1}{2} \right). \quad (2.86)$$

q ($0 < q \leq 1$) は流体の格子点と壁面との距離に関する変数， $\mathbf{u}_{\text{wall}}$ は壁の移動速度， $\hat{f}_{\alpha}$ は Interpolated bounce-back 境界条件によって与えられる速度分布関数の値である．

流入境界

流入境界条件では，密度 ρ_{in} および流速 $\mathbf{u}_{\text{in}}$ を一定値に固定する Dirichlet 条件を用いる．流入する速度分布関数が平衡分布であると仮定し，境界の速度分布関数に次のような固定値を設定する．

$$\hat{f}_{\alpha}(\mathbf{x}, t) = f_{\alpha}^{\text{eq}}(\rho_{\text{in}}, \mathbf{u}_{\text{in}}) = \rho_{\text{in}} w_{\alpha} \left(1 + \frac{\mathbf{c}_{\alpha} \cdot \mathbf{u}_{\text{in}}}{c_s^2} + \frac{(\mathbf{c}_{\alpha} \cdot \mathbf{u}_{\text{in}})^2 - c_s^2 |\mathbf{u}_{\text{in}}|^2}{c_s^4} \right) \quad (2.87)$$

流出境界

流出境界条件では，速度分布関数の進行方向に対して物理量の勾配がゼロであると仮定する Neumann 条件を用いる．境界条件によって与えられる密度・流速を用いて平衡速度分布関数を求め，境界の速度

分布関数とする.

$$\hat{f}_\alpha(\mathbf{x}, t) = f_\alpha^{\text{eq}}(\rho_{\text{out}}, \mathbf{u}_{\text{out}}) = \rho_{\text{out}} w_\alpha \left(1 + \frac{\mathbf{c}_\alpha \cdot \mathbf{u}_{\text{out}}}{c_s^2} + \frac{(\mathbf{c}_\alpha \cdot \mathbf{u}_{\text{out}})^2 - c_s^2 |\mathbf{u}_{\text{out}}|^2}{c_s^4} \right), \quad (2.88)$$

$$\rho_{\text{out}} = \rho(\mathbf{x} + \mathbf{n}c\Delta t, t), \quad (2.89)$$

$$\mathbf{u}_{\text{out}} = \mathbf{u}(\mathbf{x} + \mathbf{n}c\Delta t, t). \quad (2.90)$$

ただし, $\mathbf{n}$ は流出境界面から流体側に向かう法線ベクトルである.

2.1.5 抗力の計算

流体から物体に加わる抗力を求めるため, Momentum exchange method [37] を用いる. 壁と粒子の衝突のアナロジーから, Bounce-back または Interpolated bounce-back 条件において壁と速度分布関数の間で運動量が交換されると仮定する. 物体表面で速度分布関数が反射する前後における運動量の変化量を全て足し合わせたのち, 次元を合わせるため時間刻み幅で除算する.

$$\mathbf{F} = \frac{1}{\Delta t} \sum_{\mathbf{x} \in \Omega} \sum_{\alpha=0}^{Q-1} \mathbf{c}_\alpha B(\mathbf{x} - \mathbf{c}_\alpha \Delta t) \left(f_\alpha(\mathbf{x} - \mathbf{c}_\alpha \Delta t, t) + \hat{f}_\alpha(\mathbf{x} - \mathbf{c}_\alpha \Delta t, t + \Delta t) \right), \quad (2.91)$$

$$B(\mathbf{x}_b) = \begin{cases} 1 & (\mathbf{x}_b \text{ is fluid lattice}) \\ 0 & (\text{elsewhere}) \end{cases}. \quad (2.92)$$

ただし, $\hat{f}_\alpha$ は Bounce-back または Interpolated bounce-back の式 (2.84), (2.85) または (2.86) により求められる速度分布関数である. Ω は物体内部の格子点全体の集合, $B(\mathbf{x}_b)$ は物体内部の参照を無視するための項である.

2.1.6 ラージエディシミュレーションの導入

レイノルズ数の高い乱流の3次元計算では, Kolmogorov スケールの変動を捉えるために必要な格子点数はレイノルズ数の4分の9乗に比例して増大するため, 直接計算 (direct numerical simulation; DNS) を実施することは計算資源の観点から現実的でない. 乱流モデルを導入する必要があるが, 本研究ではラージエディシミュレーション (Large Eddy Simulation; LES) モデルを用いる. LES では, 格子解像度相当の変動 (Grid Scale; GS) までは直接計算し, 格子解像度以下の変動 (Sub-Grid Scale; SGS) のみをモデル化する. RANS と比較して LES は, GS 成分にモデル誤差が混入しないため計算精度が高い, 本質的に非定常解析であるため時間的な変動も捉えることができるという利点がある.

フィルタ理論

格子ボルツマン法や他の格子法と呼ばれる数値計算手法では, 物理空間を或る格子幅で離散化する. モデル無しで離散化した場合, 物理量は空間格子幅で平均化されたものだけが計算されるため, SGS 成分は無視され, 離散化誤差に混入する. SGS 成分をモデル化するため, フィルタ理論によって GS 成分と SGS 成分の相互作用項を導出する.

ある物理量 $f(x)$ について，以下のようなフィルタ化された物理量 $\bar{f}(x)$ を導入する．

$$\bar{f}(x) = \int_{-\infty}^{\infty} f(y)w(x-y)dy \quad (2.93)$$

ここで， $w(x)$ はフィルタ関数である．フィルタ関数の形は任意であるが，代表的なものとしてトップハットフィルタ

$$w(x) = \begin{cases} \frac{1}{\bar{\Delta}} & \left(-\frac{1}{2\bar{\Delta}} \leq x < \frac{1}{2\bar{\Delta}}\right) \\ 0 & (\text{otherwise}) \end{cases} \quad (2.94)$$

や，ガウシアンフィルタ

$$w(x) = \sqrt{\frac{6}{\pi\bar{\Delta}^2}} \exp\left(-\frac{6x^2}{\bar{\Delta}^2}\right) \quad (2.95)$$

および，スペクトルカットオフフィルタ

$$w(x) = \frac{2 \sin \frac{\pi x}{\bar{\Delta}}}{\pi x} \quad (2.96)$$

が挙げられる．ただし， $\bar{\Delta}$ はフィルタ幅であり，LES では $\bar{\Delta} = \Delta x$ とするのが一般的である．フィルタ化によって真値から GS 成分のみが取り出されると仮定すると，真値 f と GS 成分 $\bar{f}$ の差から物理量の SGS 成分が求められる．

$$f' = f - \bar{f} \quad (2.97)$$

フィルタ化された方程式系を考える．非圧縮性流体の支配方程式である連続の式および Navier-Stokes 方程式にフィルタ操作を施すと，以下の式が得られる．

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \quad (2.98)$$

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{1}{\rho} \frac{\partial \bar{p}}{\partial x_i} + \nu \frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} - \frac{\partial \tau_{ij}}{\partial x_j} \quad (2.99)$$

ここで， τ_{ij} は SGS 成分の変動が GS 成分に与える影響を表す項（SGS 応力項）で，次のように定義される．

$$\tau_{ij} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j \quad (2.100)$$

さらに， $\overline{u_i u_j}$ は以下のように展開することが出来る．

$$\overline{u_i u_j} = \overline{(\bar{u}_i + u'_i)(\bar{u}_j + u'_j)} \quad (2.101)$$

$$= \bar{u}_i \bar{u}_j + \overline{\bar{u}_i u'_j} + \overline{u'_i \bar{u}_j} + \overline{u'_i u'_j} \quad (2.102)$$

これにより, τ_{ij} は以下のように変形される.

$$\tau_{ij} = L_{ij} + C_{ij} + R_{ij}, \quad (2.103)$$

$$L_{ij} = \overline{\bar{u}_i \bar{u}_j} - \bar{u}_i \bar{u}_j, \quad (2.104)$$

$$C_{ij} = \overline{\bar{u}_i u'_j} + \overline{u'_i \bar{u}_j}, \quad (2.105)$$

$$R_{ij} = \overline{u'_i u'_j}. \quad (2.106)$$

L_{ij} は Leonard 項, C_{ij} は Cross 項, R_{ij} は Reynolds 応力項と呼ばれる. これらの項を GS 成分から直接求めることは出来ないため, モデルを導入する必要がある.

Smagorinsky モデル

Smagorinsky モデル [38] は, SGS 応力項のモデルとして広く使われているモデルである. フィルタ操作がレイノルズ平均の性質を近似的に満たすこと, および Reynolds 応力が分子粘性による応力と同様に作用することを仮定し, SGS 応力項に以下のモデル式を代入する.

$$L_{ij} \approx 0, C_{ij} \approx 0, \quad (2.107)$$

$$R_{ij} - \frac{1}{3} \delta_{ij} R_{kk} = -\nu_{\text{SGS}} \bar{S}_{ij}. \quad (2.108)$$

ただし, $\bar{S}_{ij}$ はひずみ速度テンソルであり, 以下のように定義される.

$$\bar{S}_{ij} = \frac{1}{2} \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) \quad (2.109)$$

また, ν_{SGS} は SGS 成分の変動による影響を表す渦粘性係数であり, 以下のように与えられる.

$$\nu_{\text{SGS}} = C \bar{\Delta}^2 |\bar{S}|, \quad (2.110)$$

$$|\bar{S}| = \sqrt{2 \bar{S}_{ij} \bar{S}_{ji}}. \quad (2.111)$$

C はモデル係数, $\bar{\Delta}$ はフィルタ幅である.

フィルタ化された連続の式および Navier-Stokes 方程式に Smagorinsky モデルを適用すると, 式 (2.98) および (2.99) は以下のように書き換えられる.

$$\frac{\partial \bar{u}_i}{\partial x_i} = 0 \quad (2.112)$$

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{1}{\rho} \frac{\partial \bar{p}}{\partial x_i} + \nu_t \frac{\partial^2 \bar{u}_i}{\partial x_j \partial x_j} \quad (2.113)$$

ただし, ν_t は SGS 応力項を加味した動粘性係数であり, モデルなしの動粘度 ν と渦粘性係数 ν_{SGS} の和で与えられる.

$$\nu_t = \nu + \nu_{\text{SGS}} \quad (2.114)$$

これらの式をモデルなしの方程式と比べると, 物理量がフィルタ化され GS 成分のみを値に持つこと, および動粘度に渦粘性係数 ν_{SGS} が加えられていることのみが異なる. 実装上は, 渦粘性係数を求めて分子粘性に足し合わせるだけでよく, 計算コードの僅かな変更のみで LES を導入することができる.

■**コヒーレント構造 Smagorinsky モデル** Smagorinsky モデルにおけるモデル係数 C は問題依存の定数または可変の係数である．一様等方性乱流のような物体境界を含まない問題では定数のモデル係数が設定できるが，物体境界を含む場合は壁関数を導入するか可変のモデル係数を設定する必要がある．任意の複雑形状物体では，壁関数の定義が困難であるため，可変のモデル係数を設定するほうが簡便である．モデル係数を決定する手法はいくつか存在するが，本研究では，演算の局所性が高く大規模計算に適した手法であるコヒーレント構造 Smagorinsky モデル (Coherent-structure Smagorinsky Model; CSM) [39] を用いる．CSM では，モデル係数 C を以下の式によって求める．

$$C = C_{\text{CSM}} |F_{\text{CS}}|^{\frac{3}{2}}, \quad (2.115)$$

$$C_{\text{CSM}} = \frac{1}{20}, \quad (2.116)$$

$$F_{\text{CS}} = \frac{Q}{E}, \quad (2.117)$$

$$Q = -\frac{1}{2} \frac{\partial \bar{u}_j}{\partial x_i} \frac{\partial \bar{u}_i}{\partial x_j}, E = \frac{1}{2} \frac{\partial \bar{u}_j}{\partial x_i} \frac{\partial \bar{u}_j}{\partial x_i}. \quad (2.118)$$

格子ボルツマン法への LES の導入

格子ボルツマン法では，衝突項に現れる緩和時間係数 τ を以下の τ_t に置き換えることで LES を導入することができる．

$$\tau_t = \frac{1}{2} + \frac{\nu_t}{c_s^2 \Delta t} \quad (2.119)$$

ただし， ν_t は SGS 応力項を加味した動粘性係数であり，式 (2.114) で定義される．また，格子ボルツマン法では，ひずみ速度テンソルを求めるために流速の空間差分を用いるのではなく，速度分布関数を用いて求めることができる．ひずみ速度テンソルと速度分布関数の関係は Chapman-Enskog 展開から以下のように導かれる．

$$S_{ij} = -\frac{1}{2\rho c_s^2 \tau \Delta t} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} (f_{\alpha} - f_{\alpha}^{\text{eq}}) \quad (2.120)$$

2.2 適合細分化格子法

2.2.1 データ構造

細分化格子の構造には，Fig. 2.4 に示すような木データ構造で表される再帰的な領域分割によって生成される構造を用いる．3 次元格子の場合，1 回の領域分割によって元の領域は 8 つに分割され，octree データ構造となる．Octree の各節点 (Fig. 2.4 (a) において白色の丸，および Fig. 2.4 (b) において白色の丸または色付きの四角の記号で表される要素) をノードと呼ぶ．上下に実線でつながれたノード同士は親子関係にあると言い，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．親ノードが共通であるような 2 つ以上のノードは互いに兄弟ノードであるという．子を持たないノードはリーフと呼ばれる．

GPU 計算での並列性を確保するため、それぞれのリーフには一定数の格子点で構成される等間隔格子を割り当てる。ノードセンターの格子配置を用い、1 リーフに割り当てる格子点数は $9 \times 9 \times 9$ または $17 \times 17 \times 17$ とする。

Octree のリーフのノード深さを、細分化レベル、または単にレベルとよぶ。細分化レベルはリーフの格子解像度を規定し、レベル l のリーフの格子解像度 Δx_l は、

$$\Delta x_l = 2^{l_{\max}-l} \Delta x_{l_{\max}} \quad (2.121)$$

に設定される。 $l_{\max}$ は最大の細分化レベルである。例えば、Fig. 2.4 (a) では、最も粗いリーフは 2,3,4, 最も細かいリーフは 5,6,7,8 であり、全体で 2 倍の解像度（格子間隔）の差がある。Fig. 2.4 (b) の例では、最も細かい橙色リーフの細分化レベルは 2 で、全体で 4 倍の格子解像度差がある。なお、計算の安定性を確保するため、互いに隣接するリーフのレベル差は 0 または 1 に限定する。

本データ構造において、単一の Octree は正方形または立方体形状の計算領域に対応する。より一般的な長方形または直方体の形状の計算領域を扱うため、複数の Octree を連結する Forest-of-octrees 構造 [40] を導入する。複数の Octree を連結したデータ構造および細分化格子の例を Fig. 2.5 に示す。図左側は Forest-of-octrees の木データ構造、図右側は木構造に対応する細分化格子である。細分化格子の図で実線は単一の Octree に対応する領域を、破線は Octree 内のリーフを表している。この図の例では、6 個の Octree を連結し、縦横比 4:3 の 2 次元の計算領域を設定している。3 次元の場合も同様にして、 x 軸方向に N_x 個、 y 軸方向に N_y 個、および z 軸方向に N_z 個の Octree を並べることで、任意の整数比の長さを持った直方体状の計算領域を取ることができる。

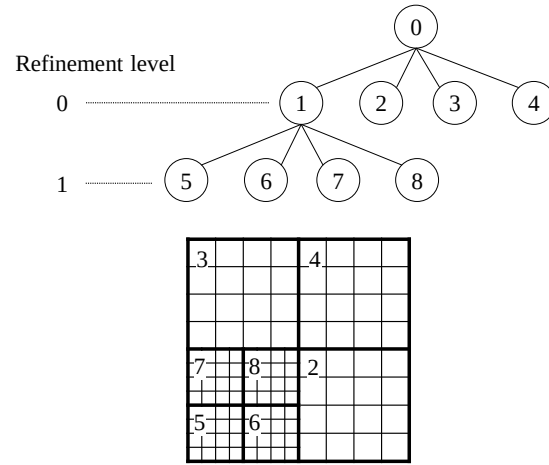
2.2.2 時間刻みの取り扱い

格子ボルツマン法では、速度分布関数の移動速度 c は格子の細分化レベルに依らず一定である。速度 c 、格子幅 Δx 、時間刻み幅 Δt との間には $c = \Delta x / \Delta t$ の関係があるため、細分化格子のように異なる解像度の格子が存在する場合、時間刻み幅も同様に異なる値を取る必要がある。最も細かい格子の細分化レベルを $l_{\max}$ 、時間刻み幅を $\Delta t_{l_{\max}}$ とすると、細分化レベルが l である格子に対する時間刻み幅 Δt_l は、次の式で定められる。

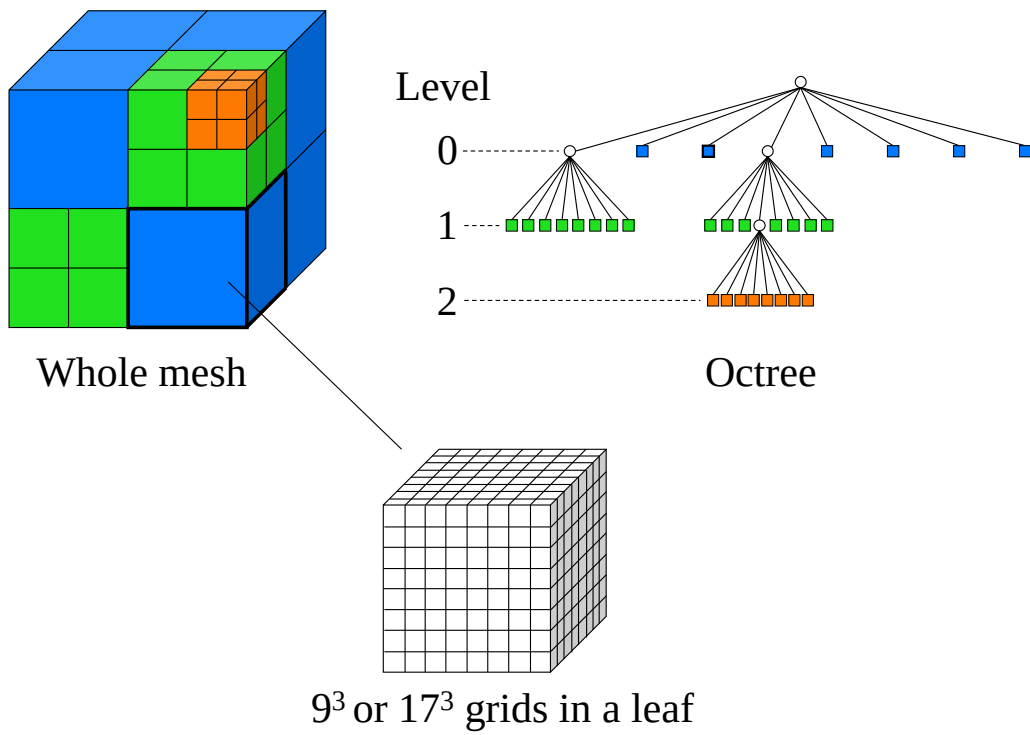
$$\Delta t_l = 2^{l_{\max}-l} \Delta t_{l_{\max}} \quad (2.122)$$

すなわち、細分化レベルがひとつ高くなるごとに時間刻み幅は半分になり、ステップ数は 2 倍になる。例として、3 つの細分化レベルの格子が存在する場合の時間発展の順序を Fig. 2.6 に示す。実線の矢印は時間発展計算を表し、破線の矢印は異なる細分化レベルの格子の間での時間補間の操作を表す。このように、粗い格子が 1 ステップの時間発展を計算する間に、1 段細かい格子では 2 ステップの計算を行うマルチタイムステップの実装が必要である。

時間刻み幅が細分化レベルによって異なることに伴い、格子ボルツマン法の緩和時間係数 τ も変化させる必要がある。細分化レベル l の格子における緩和時間係数 τ_l は、式 (2.8) の Δt を細分化レベルに



(a) 2D example.



(b) 3D mesh.

Fig. 2.4: Data structure of octree-based Adaptive Mesh Refinement.

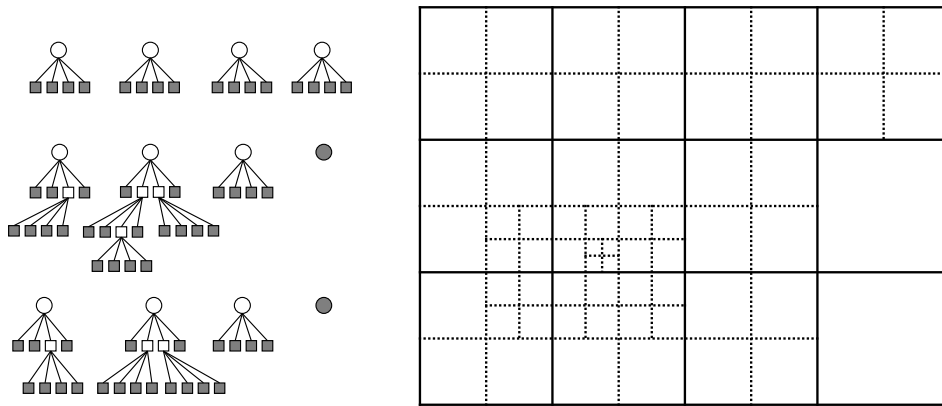


Fig. 2.5: Refined mesh of rectangular computational domain with multiple octrees.

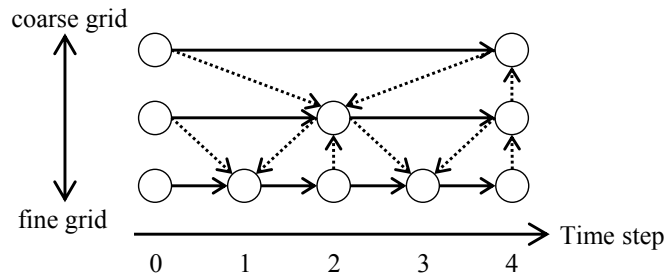


Fig. 2.6: Multiple timestep sequences in refined mesh.

応じて変化させて求める．

$$\tau_l = \frac{1}{2} + \frac{\nu}{c_s^2 \Delta t_l} \quad (2.123)$$

2.2.3 時空間補間及びレベル変換

格子ボルツマン法での隣接点参照において，隣接格子点の細分化レベルが異なる場合，速度分布関数の時空間補間を行ったのち，レベル変換によって速度分布関数の値を修正する．

粗い格子から細かい格子への補間

時間方向および各座標方向のそれぞれについて線形補間を行う．時間方向の補間は，Fig. 2.6 の破線の矢印で示す通りである．粗い格子に同一時刻に時刻ステップが存在する場合にはその時刻の値を用い，粗い格子に同一時刻の時刻ステップが存在しない場合には前後の時刻ステップの平均値を取る．空間方向の補間では，座標方向ごとに同一座標上に粗い格子が存在する場合としない場合があり，それに応じて近傍の1点，2点，4点または8点の平均値を取る．2次元格子の場合の格子点の位置と空間補間のパターンを Fig. 2.7 に示す．白抜きの大きい丸は粗い格子である．黒，赤，青の丸印は空間補間された細

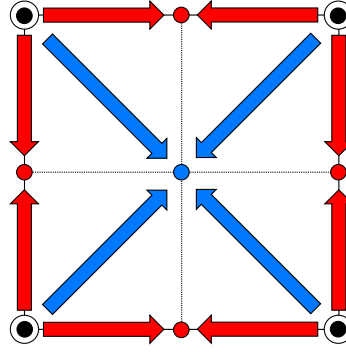


Fig. 2.7: Two-dimensional spacial interpolation from coarse to fine mesh.

かい格子であり、それぞれ 1 点、2 点、4 点の粗い格子から補間される点を表す．赤および青色の矢印の始点および終点は、それぞれ補間に用いられる粗い格子の点および補間された細かい格子の点を表す．また、3 次元の場合、これらに加えて 8 点の粗い格子から補間される格子も存在する．

細かい格子から粗い格子への補間

粗い格子における時刻ステップおよび座標には、細かい格子が必ず存在している．したがって、粗い格子の点の時刻・座標に重複して存在する細かい格子の点の値をそのまま補間値とする．

速度分布関数のレベル変換

格子ボルツマン法における時空間補間は、流体の物理量ではなく速度分布関数に対して適用される．そのため、ひずみ速度テンソルが補間操作に対して不連続となり、非物理的な応力が生じる．ひずみ速度テンソルを連続にするため、補間された速度分布関数の値を修正する必要がある [20, 21]．本節では衝突項に BGK モデルを用いることとしてレベル変換の式を導出するが、MRT モデルやキュムラントモデルに対しても同一の式を適用することが可能である．

Chapman-Enskog 展開によれば、速度分布関数とひずみ速度テンソルの間には、以下の関係がある．

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} f_{\alpha}^{\text{neq}} = -2\rho c_s^2 \tau \Delta t S_{ij} \quad (2.124)$$

ただし、 f_{α}^{neq} は速度分布関数の非平衡成分で、次の式で定義される．

$$f_{\alpha}^{\text{neq}} = f_{\alpha} - f_{\alpha}^{\text{eq}} \quad (2.125)$$

f_{α} , f_{α}^{eq} は補間操作に対して連続であるから、 f_{α}^{neq} もまた連続である．しかし、緩和時間係数 τ および時間刻み Δt が細分化レベルによって異なるため、ひずみ速度テンソルは細分化レベル差の境界で不連続となる．異なる細分化レベルの間でひずみ速度テンソルを連続にするためには、すべての細分化レベル

の格子で以下の関係を満たす必要がある。

$$\frac{f_{\alpha}^{(l,\text{neq})}}{\tau_l \Delta t_l} = \text{const.} \quad (2.126)$$

上式を満たすため、非平衡成分 f_{α}^{neq} の値を修正する式（レベル変換）を適用する。細かい格子から粗い格子への補間に対するレベル変換は、次の式で表される。

$$\frac{f_{\alpha}^{\text{neq,coarse}}}{\tau^{\text{coarse}} \Delta t^{\text{coarse}}} = \frac{f_{\alpha}^{\text{neq,fine}}}{\tau^{\text{fine}} \Delta t^{\text{fine}}} \quad (2.127)$$

$$\begin{aligned} \therefore f_{\alpha}^{\text{neq,coarse}} &= \frac{\tau^{\text{coarse}} \Delta t^{\text{coarse}}}{\tau^{\text{fine}} \Delta t^{\text{fine}}} f_{\alpha}^{\text{neq,fine}} \\ &= 2 \frac{\tau^{\text{coarse}}}{\tau^{\text{fine}}} f_{\alpha}^{\text{neq,fine}} \end{aligned} \quad (2.128)$$

同様に、粗い格子から細かい格子への補間に対するレベル変換は、以下の式で表される。

$$f_{\alpha}^{\text{neq,fine}} = \frac{1}{2} \frac{\tau^{\text{fine}}}{\tau^{\text{coarse}}} f_{\alpha}^{\text{neq,coarse}} \quad (2.129)$$

レベル変換によりひずみ速度テンソルはレベル差の境界面で連続となるが、ノードセンターの格子を用いている場合、高レイノルズ数の条件において流体の密度および運動量に不均衡が生じる。これは、Fig. 2.8 のように、境界面の格子点上に粗い格子から来る速度分布関数と細かい格子から来る速度分布関数が共存しているためである。一部の速度分布関数にしかレベル変換の式が適用されないため、非平衡成分の総和が非ゼロの値を取ってしまう。密度と速度の保存性を改善するため、レベル変換に Kuwata らの Imbalance-correction [23] を導入する。Imbalance-correction では、レベル変換における平衡分布関数を計算するための密度および速度に補正項を加える。

$$f_{\alpha}^{\text{eq}} = f_{\alpha}^{\text{eq}}(\rho^*, \mathbf{u}^*), \quad (2.130)$$

$$\rho^* = \rho + \lambda_{\rho} \varepsilon_{\rho}, \quad (2.131)$$

$$\mathbf{u}^* = \mathbf{u} + \lambda_{\mathbf{u}} \varepsilon_{\mathbf{u}}. \quad (2.132)$$

λ_{ρ} および $\lambda_{\mathbf{u}} = (\lambda_u, \lambda_v, \lambda_w)^{\top}$ は重み係数である。細い格子から粗い格子への変換で、レベル差の境界面の方向が x 平面である場合、重み係数は次式で定義される。

$$\lambda_{\rho} = \lambda_u = \left(1 - \frac{1}{6} \left(1 - n \frac{\tau^{\text{coarse}}}{\tau^{\text{fine}}} \right) \left(1 + \frac{c^2}{c_s^2} \right) \right)^{-1} \quad (2.133)$$

$$\lambda_v = \lambda_w = \left(1 - \frac{1}{18} \left(1 - n \frac{\tau^{\text{coarse}}}{\tau^{\text{fine}}} \right) \left(\frac{c^2}{c_s^2} \right) \right)^{-1} \quad (2.134)$$

$\varepsilon_{\rho}, \varepsilon_{\mathbf{u}}$ は Imbalance-correction なしのレベル変換で発生する密度と速度の不均衡量で、以下の式で計算される。

$$\varepsilon_{\rho} = \sum_{\alpha \in A_c} f_{\alpha}^{\text{neq,coarse}} + \sum_{\alpha \in \hat{A}_c} f_{\alpha}^{\text{neq,fine} \rightarrow \text{coarse}} \quad (2.135)$$

$$\varepsilon_{\mathbf{u}} = \sum_{\alpha \in A_c} \mathbf{c}_{\alpha} f_{\alpha}^{\text{neq,coarse}} + \sum_{\alpha \in \hat{A}_c} \mathbf{c}_{\alpha} f_{\alpha}^{\text{neq,fine} \rightarrow \text{coarse}} \quad (2.136)$$

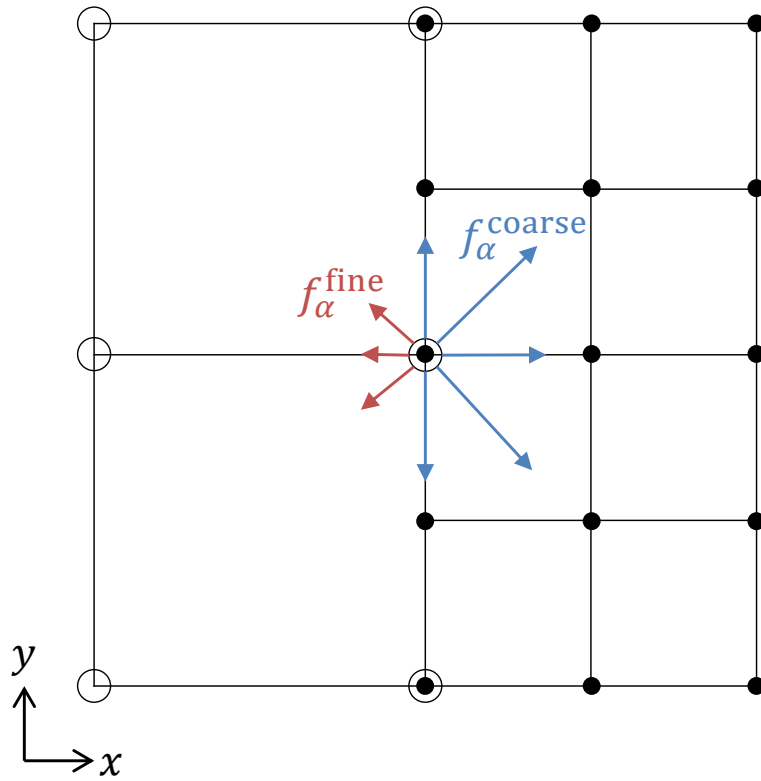


Fig. 2.8: Coarse/fine distribution functions on the level-interface lattice.

2.3 検証計算

2.3.1 Fourier 解析による数値安定性と精度の検証

数値計算手法の安定性と精度を代数的に検証する方法として、解を波数成分に分解して単一の波数成分ごとの応答性を確認する Fourier 解析が良く用いられる。しかし、流体の支配方程式である Navier-Stokes 方程式では、非線形性によって高波数の成分が生成されるため、単一の波数を仮定する Fourier 解析を適用することはできない。本節では、Navier-Stokes 方程式に類似する線形方程式である移流拡散方程式を用いて、格子ボルツマン法の Fourier 解析を実施する。

1 次元移流拡散方程式の格子ボルツマン法による数値解法

一様な速度場における、或る物理量 $\varphi(x, t)$ の 1 次元移流拡散方程式は、以下の式で表される。

$$\frac{\partial \varphi}{\partial t} + U \frac{\partial \varphi}{\partial x} = \kappa \frac{\partial^2 \varphi}{\partial x^2} \quad (2.137)$$

ただし, U は移流速度, κ は拡散係数である. 移流拡散方程式に対応する格子ボルツマン方程式は, 以下のように表される.

$$f_\alpha(x + c_\alpha \Delta t, t + \Delta t) = f_\alpha(x, t) - \frac{1}{\tau} (f_\alpha(x, t) - f_\alpha^{\text{eq}}(x, t)), \quad (2.138)$$

$$f_\alpha^{\text{eq}} = \varphi w_\alpha \left(1 + \frac{c_\alpha U}{c_s^2} \right), \quad (2.139)$$

$$\varphi = \sum_{\alpha=0}^{Q-1} f_\alpha. \quad (2.140)$$

ただし, 衝突項に BGK モデルを用いた. τ は緩和時間係数であり, 拡散係数 κ に対して次の関係を満たす.

$$\kappa = c_s^2 \Delta t \left(\tau - \frac{1}{2} \right) \quad (2.141)$$

速度モデルに 1 次元 3 速度 (D1Q3) モデルを用い, c_α, w_α, c_s を以下のように定義する.

$$(c_0, c_1, c_2) = (0, c, -c), c \equiv \frac{\Delta x}{\Delta t}, \quad (2.142)$$

$$(w_0, w_1, w_2) = \left(\frac{2}{3}, \frac{1}{6}, \frac{1}{6} \right), \quad (2.143)$$

$$c_s = \frac{c}{\sqrt{3}}. \quad (2.144)$$

von Neumann の安定性解析

格子ボルツマン法の安定性解析を行うため, いくつかの式変形を施す. まず, 格子ボルツマン方程式において, 左辺の速度分布関数の座標をそろえる.

$$f_\alpha(x, t + \Delta t) = f_\alpha(x - c_\alpha \Delta t, t) - \frac{1}{\tau} (f_\alpha(x - c_\alpha \Delta t, t) - f_\alpha^{\text{eq}}(x - c_\alpha \Delta t, t)) \quad (2.145)$$

次に、平衡分布関数 f_α^{eq} を定義式で展開し、さらに、 f_α^{eq} 内に現れる φ を速度分布関数の総和で展開する。

$$f_0(x, t + \Delta t) = f_0(x, t) - \frac{1}{\tau} \left(f_0(x, t) - \frac{2}{3} (f_0(x, t) + f_1(x, t) + f_2(x, t)) \right), \quad (2.146)$$

$$f_1(x, t + \Delta t) = f_1(x - c\Delta t, t) - \frac{1}{\tau} \left(f_0(x - c\Delta t, t) - \frac{1 + \frac{3U}{c}}{6} (f_0(x - c\Delta t, t) + f_1(x - c\Delta t, t) + f_2(x - c\Delta t, t)) \right), \quad (2.147)$$

$$f_2(x, t + \Delta t) = f_2(x + c\Delta t, t) - \frac{1}{\tau} \left(f_0(x + c\Delta t, t) - \frac{1 - \frac{3U}{c}}{6} (f_0(x + c\Delta t, t) + f_1(x + c\Delta t, t) + f_2(x + c\Delta t, t)) \right). \quad (2.148)$$

さらに、右辺を f_0, f_1, f_2 で整理する。

$$f_0(x, t + \Delta t) = \left(1 - \frac{1}{3\tau} \right) f_0(x, t) + \frac{2}{3\tau} f_1(x, t) + \frac{2}{3\tau} f_2(x, t), \quad (2.149)$$

$$f_1(x, t + \Delta t) = \frac{1 + \frac{3U}{c}}{6\tau} f_0(x - c\Delta t, t) + \left(1 - \frac{5 - \frac{3U}{c}}{6\tau} \right) f_1(x - c\Delta t, t) + \frac{1 + \frac{3U}{c}}{6\tau} f_2(x - c\Delta t, t), \quad (2.150)$$

$$f_2(x, t + \Delta t) = \frac{1 - \frac{3U}{c}}{6\tau} f_0(x + c\Delta t, t) + \frac{1 - \frac{3U}{c}}{6\tau} f_1(x + c\Delta t, t) + \left(1 - \frac{5 + \frac{3U}{c}}{6\tau} \right) f_2(x + c\Delta t, t). \quad (2.151)$$

ここで、物理量に対して或る波数を持つ擾乱が加わったとことを想定し、速度分布関数の値を次のように定義する。

$$f_\alpha(x, t) = f_\alpha^0 e^{2\pi i k x} \quad (2.152)$$

f_α^0 は擾乱の振幅、 i は虚数単位であり、 k は波数である。格子点数を N とすると、サンプリング定理より、 k は 0 以上 $N/2$ 未満の整数の値を取る。このとき、隣接格子点上の速度分布関数は、式 (2.152) の定義を代入することで、以下のように与えられる。

$$f_\alpha(x \pm c\Delta t, t) = f_\alpha^0 e^{2\pi i k (x \pm c\Delta t)} = f_\alpha(x, t) e^{\pm 2\pi i k c \Delta t} \quad (2.153)$$

$$= f_\alpha(x, t) e^{\pm i \omega} \quad (2.154)$$

ただし, $\omega = 2\pi kc\Delta t$ と置いた. ω は格子幅で規格化された波数で, $0 \leq \omega < \pi$ である. 式 (2.153) を式 (2.149), (2.150), (2.151) に代入することで, 単一格子点上の速度分布関数 $f_\alpha(x, t)$ と物理量 $\varphi(x, t)$ で構成される線形システムの状態方程式が得られる.

$$\begin{pmatrix} f_0(x, t + \Delta t) \\ f_1(x, t + \Delta t) \\ f_2(x, t + \Delta t) \end{pmatrix} = L \begin{pmatrix} f_0(x, t) \\ f_1(x, t) \\ f_2(x, t) \end{pmatrix}, \quad (2.155)$$

$$L = \begin{pmatrix} 1 - \frac{1}{3\tau} & \frac{2}{3\tau} & \frac{2}{3\tau} \\ \frac{1 + \frac{3U}{c}}{6\tau} e^{-i\omega} & \left(1 - \frac{5 - \frac{3U}{c}}{6\tau}\right) e^{-i\omega} & \frac{1 + \frac{3U}{c}}{6\tau} e^{-i\omega} \\ \frac{1 - \frac{3U}{c}}{6\tau} e^{i\omega} & \frac{1 - \frac{3U}{c}}{6\tau} e^{i\omega} & \left(1 - \frac{5 + \frac{3U}{c}}{6\tau}\right) e^{i\omega} \end{pmatrix}, \quad (2.156)$$

$$\varphi(x, t) = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} f_0(x, t) \\ f_1(x, t) \\ f_2(x, t) \end{pmatrix}. \quad (2.157)$$

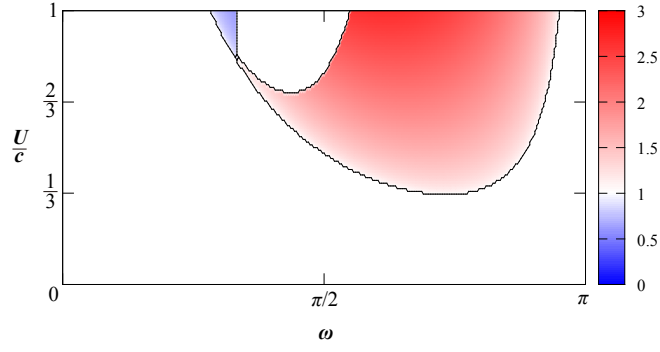
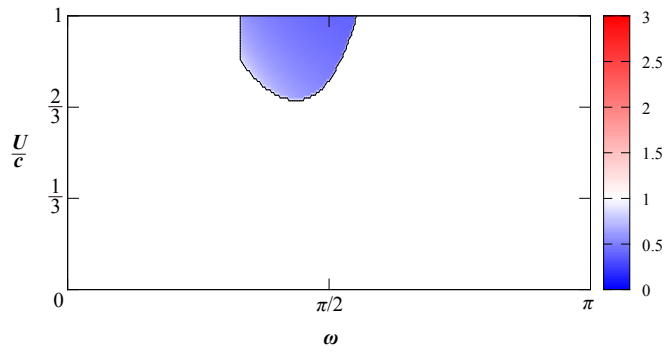
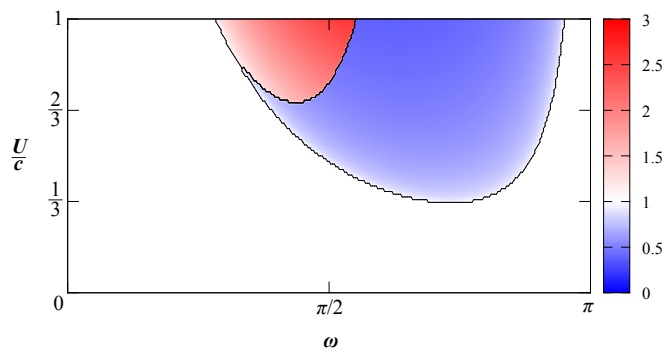
L は係数行列である. 任意のタイムステップ n ($n\Delta t = t$) に対して上式を再帰的に適用することで, 以下の式が得られる.

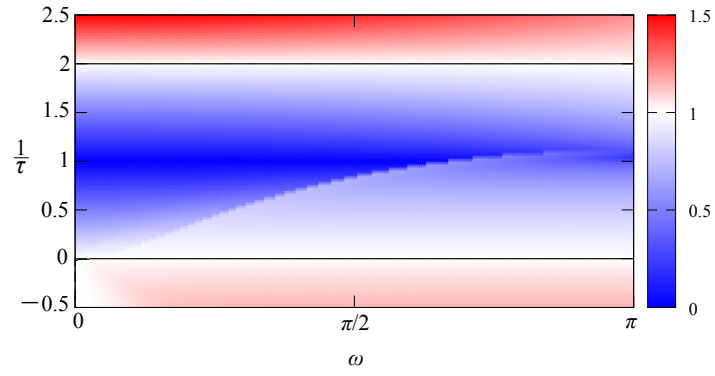
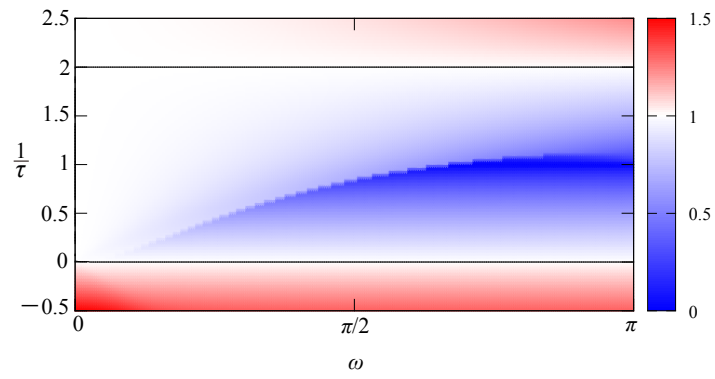
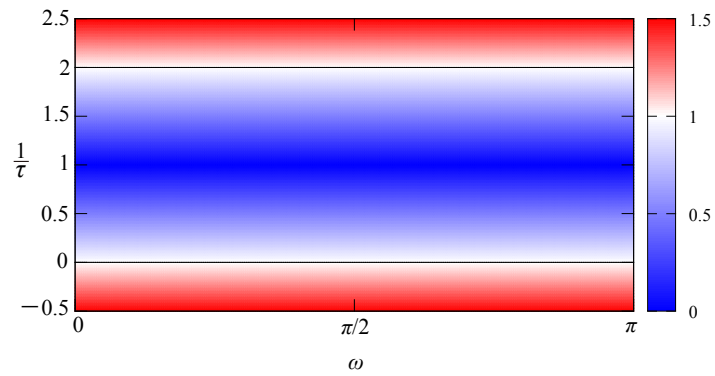
$$\varphi(x, t) = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} L^n \begin{pmatrix} f_0(x, 0) \\ f_1(x, 0) \\ f_2(x, 0) \end{pmatrix} \quad (2.158)$$

上式から, 系が波数 ω の擾乱に対して安定であることの必要十分条件は, 係数行列 L の累乗が発散しないことであり, 固有値問題に帰着することができる. 行列 L の固有値を $\lambda_0, \lambda_1, \lambda_2$ とすると, 系の安定性を以下のように分類することができる.

$$\begin{cases} \max(|\lambda_0|, |\lambda_1|, |\lambda_2|) < 1 & \text{安定} \\ \max(|\lambda_0|, |\lambda_1|, |\lambda_2|) = 1 & \text{中立安定} \\ \max(|\lambda_0|, |\lambda_1|, |\lambda_2|) > 1 & \text{不安定} \end{cases} \quad (2.159)$$

行列の固有値は, 波数 ω , 緩和時間係数 τ および移流速度 U による 3 変数関数となるが, τ または U のいずれかを固定値にすると議論が簡単である. 格子ボルツマン法は, $\tau = 0.5$ のとき移流方程式に, $U = 0$ のとき拡散方程式に一致する. $\tau = 0.5$ のときの固有値の分布を Fig. 2.9 に, $U = 0$ のときの固有値の分布を Fig. 2.10 に示す. 青色の部分は安定域, 白色の部分は中立安定域, 赤色の部分は不安定域である. また, 黒の実線は固有値の絶対値が 1 の等高線を示している. 計算が安定であるためには, すべての波数 ω に対して系が安定である必要がある. したがって, 安定条件は $\frac{U}{c} \leq \frac{1}{3}$ および $0 \leq \frac{1}{\tau} \leq 2$ である.

(a) $|\lambda_0|$ (b) $|\lambda_1|$ (c) $|\lambda_2|$ Fig. 2.9: Eigenvalues' ω - U dependency with $\tau = 0.5$

(a) $|\lambda_0|$ (b) $|\lambda_1|$ (c) $|\lambda_2|$ Fig. 2.10: Eigenvalues' ω - τ dependency with $U = 0$

移流方程式に対する精度

移流方程式に初期擾乱 $\varphi(x, 0) = e^{i\omega x}$ を加えた時、厳密解は以下のようになる.

$$\varphi_{\text{exact}}(x, t) = e^{i\omega(x-Ut)} \quad (2.160)$$

一方、格子ボルツマン法による数値解は、初期の速度分布関数を平衡分布と仮定すると、以下のようになる.

$$\varphi_{\text{numerical}}(x, t) = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} L^n \begin{pmatrix} \frac{2}{3} e^{i\omega x} \\ \frac{1 + \frac{3U}{c}}{6} e^{i\omega x} \\ \frac{1 - \frac{3U}{c}}{6} e^{i\omega x} \end{pmatrix} \quad (2.161)$$

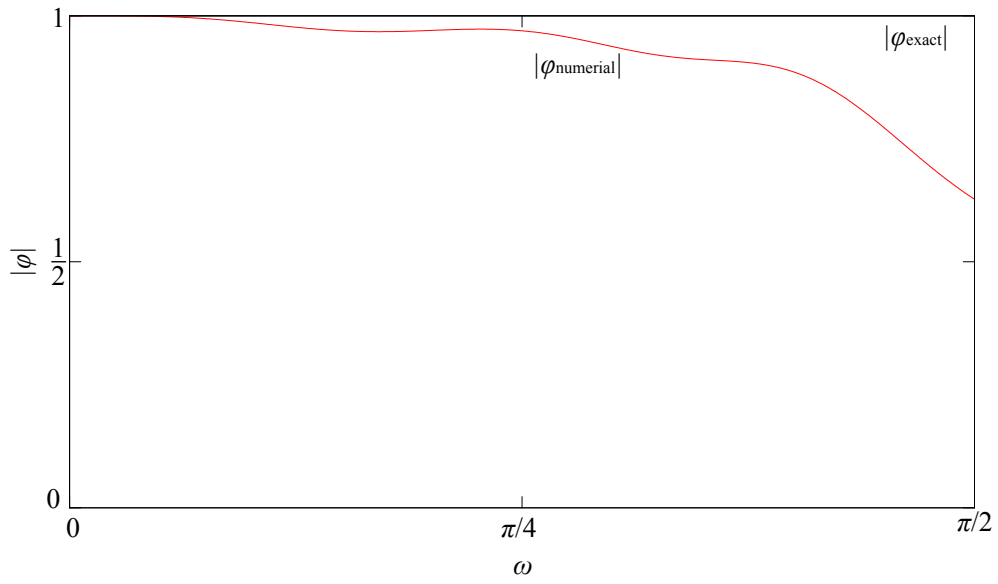
ただし、 $t = n\Delta t$, $\tau = \frac{1}{2}$ である. $U = 0.1c$, $n = 10$ としたときの、 $x = 0$ における $\varphi_{\text{exact}}, \varphi_{\text{numerical}}$ の絶対値と偏角を Fig. 2.11 に示す. 黒の実線は厳密解、赤の実線は数値解である. 高波数の成分ほど数値解と厳密解のずれが大きくなっている. 高波数域において数値解の絶対値が 1 未満であるが、これは、 $\varphi_{\text{numerical}}$ が内部状態 f_0, f_1, f_2 の足し合わせとして記述されるためである. 本条件において係数行列 L の固有値の絶対値は全て 1 であるから、擾乱は時間的に減衰しない.

格子ボルツマン法による計算の数値誤差を評価する. 上で求めた $\varphi_{\text{exact}}, \varphi_{\text{numerical}}$ から、1 ステップあたりの利得誤差 (Gain error) および位相誤差 (Phase error) を以下の式により求める.

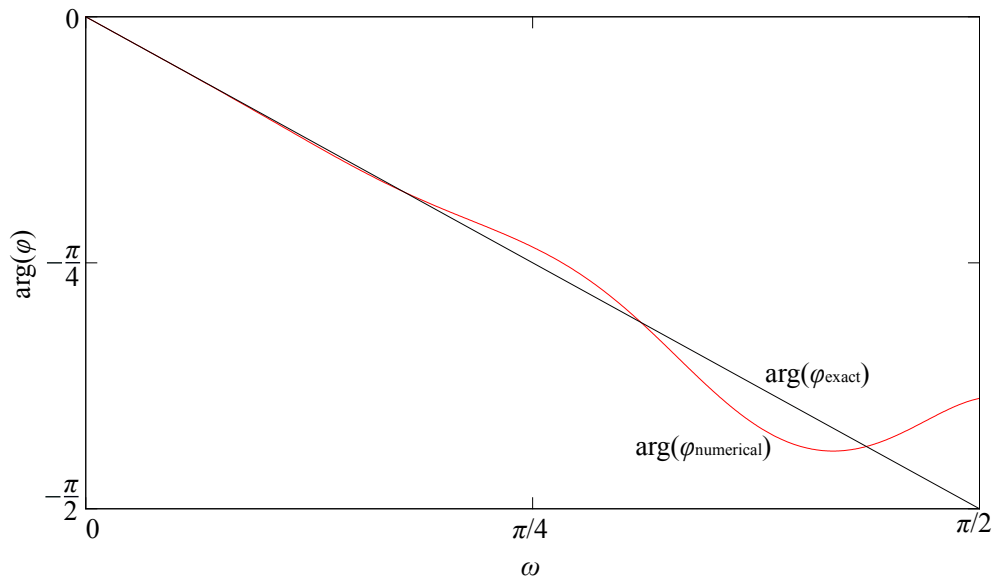
$$\begin{aligned} \text{Gain error} &= \left| \frac{|\varphi_{\text{exact}}(x, n\Delta t)|^{\frac{1}{n}} - |\varphi_{\text{numerical}}(x, n\Delta t)|^{\frac{1}{n}}}{|\varphi_{\text{exact}}(x, n\Delta t)|^{\frac{1}{n}}} \right| \\ &= \left| 1 - |\varphi_{\text{numerical}}(x, n\Delta t)|^{\frac{1}{n}} \right| \end{aligned} \quad (2.162)$$

$$\begin{aligned} \text{Phase error} &= \left| \frac{\arg(\varphi_{\text{exact}}(x, n\Delta t))/n - \arg(\varphi_{\text{numerical}}(x, n\Delta t))/n}{\arg(\varphi_{\text{exact}}(x, n\Delta t))/n} \right| \\ &= \left| 1 - \frac{\arg(\varphi_{\text{numerical}}(x, n\Delta t))}{-\omega n U \Delta t} \right| \end{aligned} \quad (2.163)$$

各波数に対する利得誤差および位相誤差の分布を Fig. 2.12 に示す. 高波数域において誤差が急激に減衰している. これは、系の共振によるものと考えられる. 共振の影響を除けば、利得誤差と位相誤差は、それぞれ波数に対して 4 次と 2 次の傾きになっている. したがって、格子ボルツマン法の移流方程式に対する誤差は位相誤差が支配的であることがいえるとともに、移流方程式に対する格子ボルツマン法の計算精度が時空間 2 次であることが確認できた.

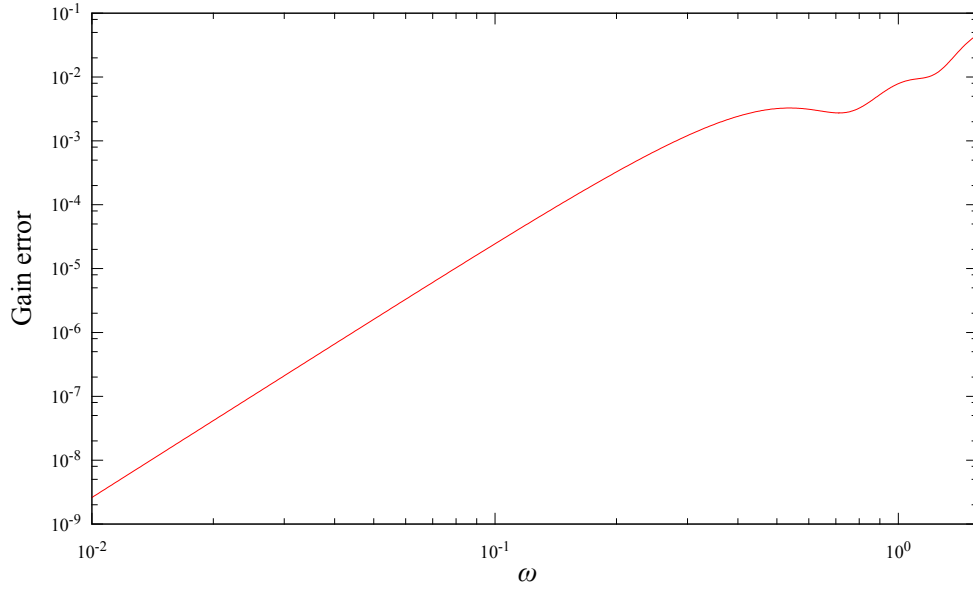


(a) Absolute value.

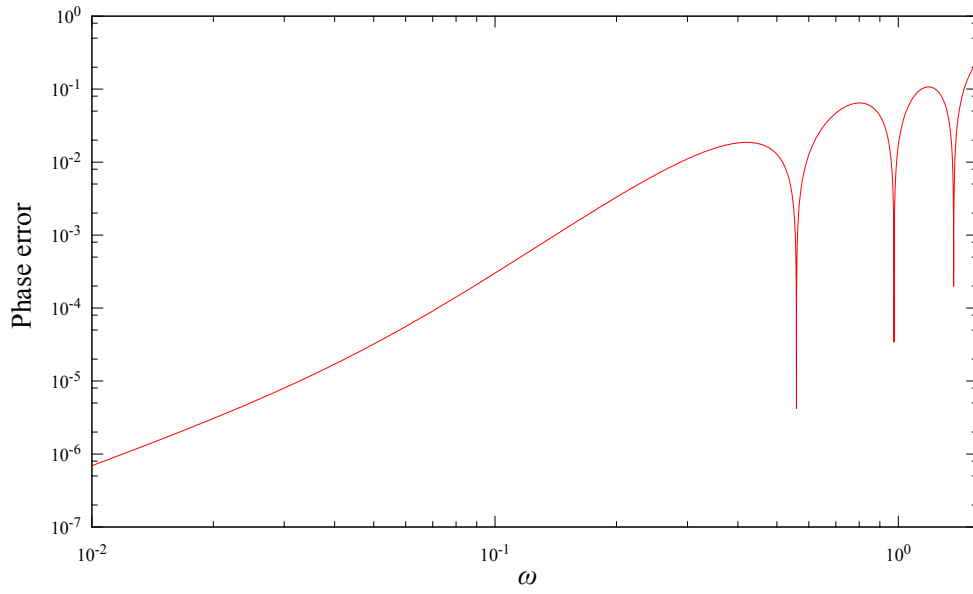


(b) Amplitude.

Fig. 2.11: Exact and numerical solution of advection equation.



(a) Gain error.



(b) Phase error.

Fig. 2.12: Error of Lattice Boltzmann Method for 1D advection equation.

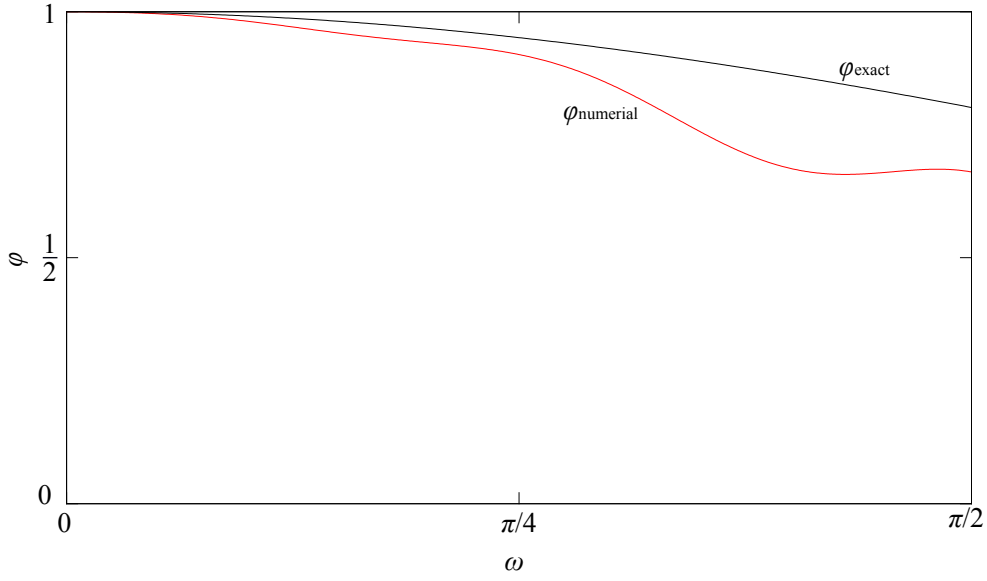


Fig. 2.13: Exact and numerical solution of diffusion equation.

拡散方程式に対する精度

拡散方程式に初期擾乱 $\varphi(x, 0) = e^{i\omega x}$ を加えた時、厳密解は以下のようになる。

$$\varphi_{\text{exact}}(x, t) = e^{-\kappa\omega^2 t} e^{i\omega x} \quad (2.164)$$

一方、格子ボルツマン法による数値解は、初期の速度分布関数を平衡分布と仮定すると、以下のようになる。

$$\varphi_{\text{numerical}}(x, t) = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} L^n \begin{pmatrix} \frac{2}{3} e^{i\omega x} \\ \frac{1}{6} e^{i\omega x} \\ \frac{1}{6} e^{i\omega x} \end{pmatrix} \quad (2.165)$$

ただし、 $t = n\Delta t$ 、 $U = 0$ である。拡散方程式では、位相の変化はなく位相誤差は厳密に 0 であるため、誤差は利得誤差のみとなる。 $\frac{1}{\tau} = 1.9$ 、 $n = 10$ としたときの、 $x = 0$ における $\varphi_{\text{exact}}, \varphi_{\text{numerical}}$ の値を Fig. 2.13 に示す。黒の実線は厳密解、赤の実線は数値解である。移流方程式と同様、波数が大きくなるほど数値解の誤差が大きくなっている。さらに、 $\varphi_{\text{exact}}, \varphi_{\text{numerical}}$ から、1 ステップあたりの誤差を以下の式により求める。

$$\begin{aligned} \text{Error} &= \left| \frac{(\varphi_{\text{exact}}(x, n\Delta t))^{\frac{1}{n}} - (\varphi_{\text{numerical}}(x, n\Delta t))^{\frac{1}{n}}}{(\varphi_{\text{exact}}(x, n\Delta t))^{\frac{1}{n}}} \right| \\ &= \left| 1 - e^{\frac{1}{3}(\tau - \frac{1}{2})\omega^2} (\varphi_{\text{numerical}}(x, n\Delta t))^{\frac{1}{n}} \right| \end{aligned} \quad (2.166)$$

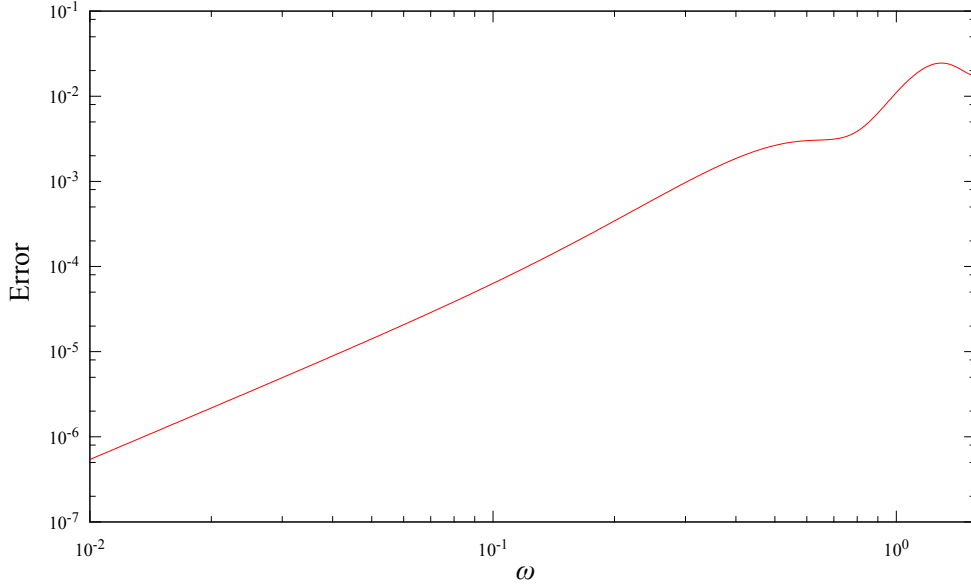


Fig. 2.14: Error of Lattice Boltzmann Method for 1D diffusion equation.

各波数に対する誤差の分布を Fig. 2.14 に示す．誤差は波数に対して 2 次の傾きになっており，したがって，拡散方程式に対する格子ボルツマン法の計算精度が時空間 2 次であることが確認できた．

以上により，移流拡散方程式に対する格子ボルツマン法の計算精度は，移流項および拡散項の双方に対して時空間 2 次であることが確認できた．

2.3.2 Standing vortex

非線形項を含む Navier-Stokes 方程式に対する格子ボルツマン法の精度を検証するため，厳密解の存在する Standing vortex の解析を実施する．計算領域は 2 次元の $-\frac{1}{2} \leq x, y \leq \frac{1}{2}$ の正方領域であり，全ての方向に対して周期境界条件が設定される．初期速度場は以下の式で定義される．

$$u_r = 0, \quad (2.167)$$

$$u_\theta = \begin{cases} 5r & \left(r < \frac{1}{5} \right) \\ 2 - 5r & \left(\frac{1}{5} \leq r < \frac{2}{5} \right) \\ 0 & \left(r \geq \frac{2}{5} \right) \end{cases}. \quad (2.168)$$

ただし， (r, θ) は極座標であり， u_r, u_θ はそれぞれ速度の r 方向成分， θ 方向成分である．本初期条件における渦度の分布は Fig. 2.15 のようになる．無粘性流れ ($\nu = 0, Re \rightarrow \infty$) を仮定すると，厳密解は初期条件そのものであり，ある程度の時間まで計算を進めた時のエネルギーの誤差から計算精度を見積も

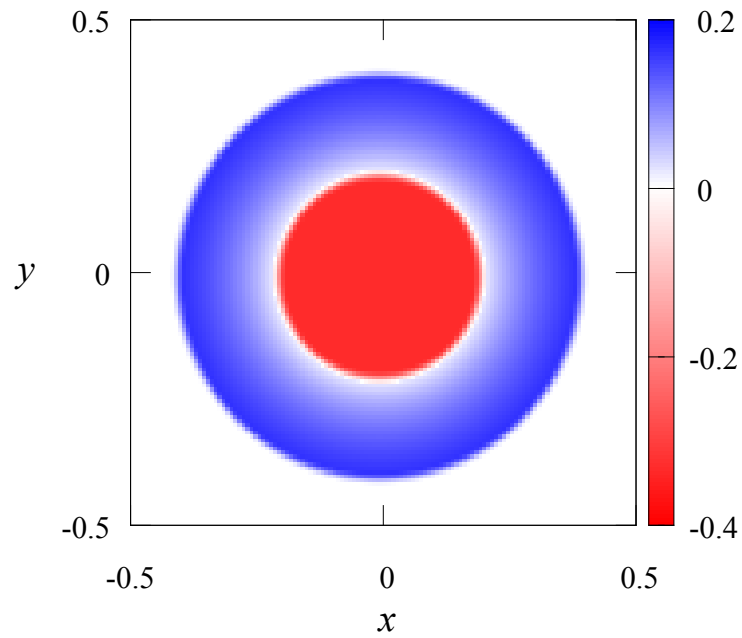


Fig. 2.15: Initial vorticity profile of standing vortex.

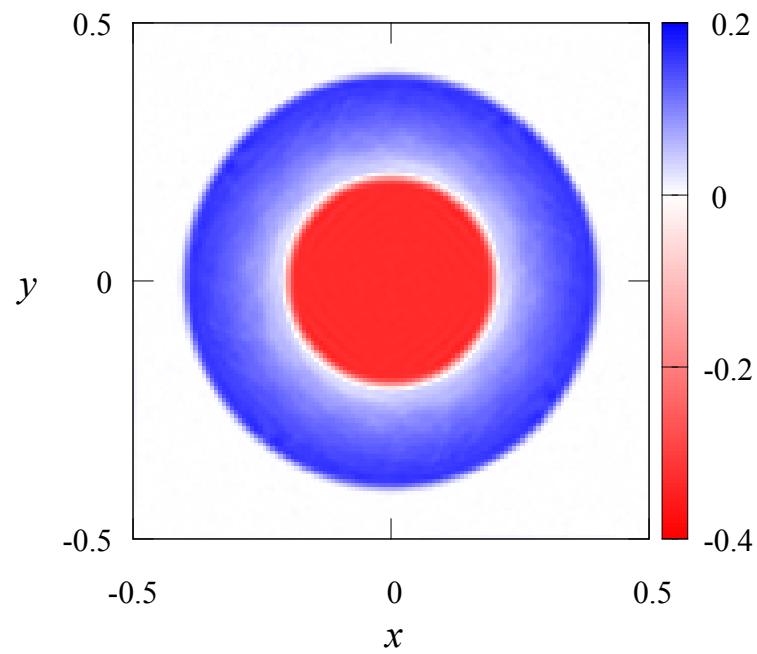


Fig. 2.16: Vorticity profile of standing vortex at $N = 129$, $t = 3$.

Table 2.2: Energy error of standing vortex.

N	energy error	order
33	4.25663×10^{-3}	—
65	7.99351×10^{-4}	2.70349
129	1.92211×10^{-4}	2.09548
257	4.69807×10^{-5}	2.05355

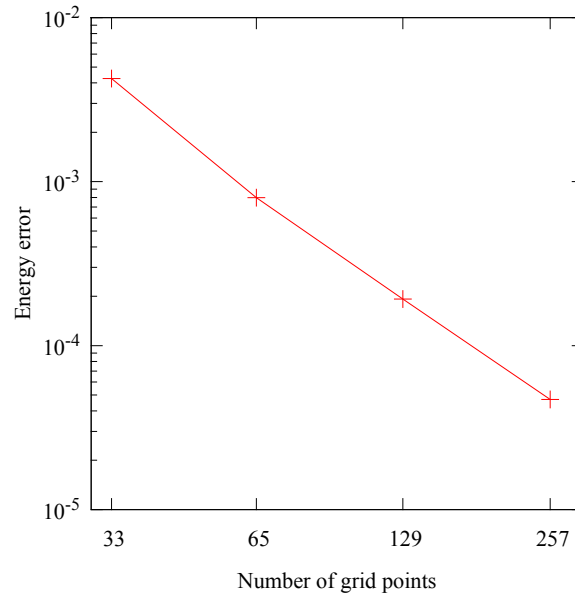


Fig. 2.17: Energy error of standing vortex.

ることができる。

格子ボルツマン法の速度モデルおよび衝突項に D2Q9 BGK モデルを用いて計算した。格子点数を $N \times N$ ($N = 33, 65, 129, 257$)、最大速度に対するクーラン数 (u/c) を 0.1 とし、それぞれ無次元時間 $t = 3$ まで計算を行った。 $N = 129, t = 3$ における渦度の分布を Fig. 2.16 に示す。色は渦度の値であり、赤色は負の渦度、青色は渦度を表す。白色の領域の渦度は 0 であり、渦が存在しないことを意味している。初期の渦の形状が良く保存されており、格子ボルツマン法による流体計算の妥当性が確認できる。さらに、各格子点数の計算結果に対してエネルギーの誤差の絶対値を求め、全格子点・全時間の平均値を取った。結果を Table 2.2 および Fig. 2.17 に示す。エネルギーの誤差が格子点数に対しておよそ 2 次の精度で減衰している。したがって、非線形項に対する格子ボルツマン法の精度はエネルギーに対して時空間 2 次であると推定できる。

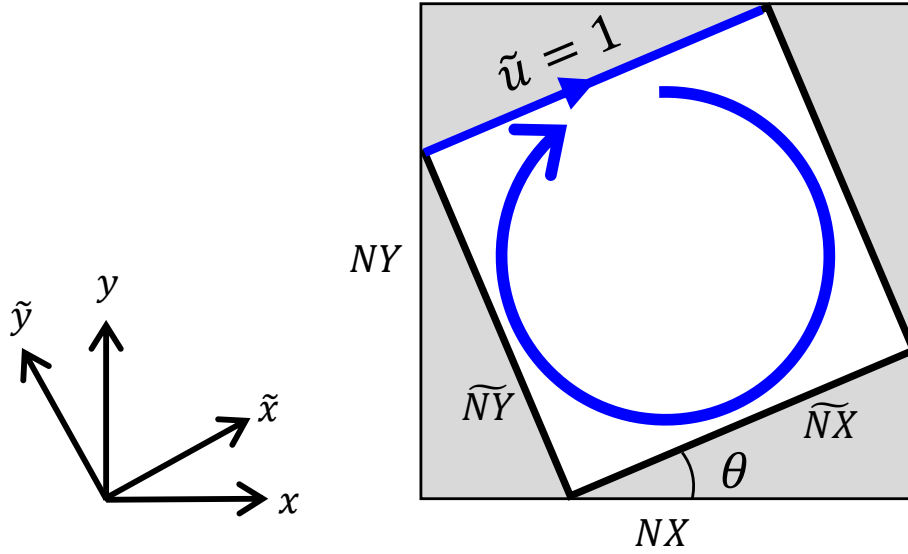


Fig. 2.18: Lid-driven cavity flow with oblique computational area.

2.3.3 2次元キャビティ流れ

計算領域を傾けたキャビティ流れ

キャビティ流れの計算は壁面近傍の応力テンソルの評価に生じる誤差によって計算結果が大きく変化するため、境界条件の精度を検証するベンチマーク問題として用いることが出来る。計算領域を傾けたキャビティ流れの計算を行うことで、境界条件の精度の座標方向依存性について確かめた。計算格子には等間隔格子を用いた。Fig. 2.18 に示すように、直交格子から角度 θ だけ傾いた計算領域を取る。直交格子に沿った軸を x, y とし、直交格子に対して角度 θ だけ傾いた計算領域の座標を $\tilde{x}, \tilde{y}$ とする。白色の部分はキャビティ流れの計算領域であり、灰色の部分は計算を行わない余白領域である。 $\tilde{x}, \tilde{y}$ は、以下のような座標変換によって求めることが出来る。

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \quad (2.169)$$

ここで、 (x_0, y_0) は座標の回転の中心座標である。流速 $\tilde{u}, \tilde{v}$ も同様に座標変換から求めることが出来る。

$$\begin{pmatrix} \tilde{u} \\ \tilde{v} \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} \quad (2.170)$$

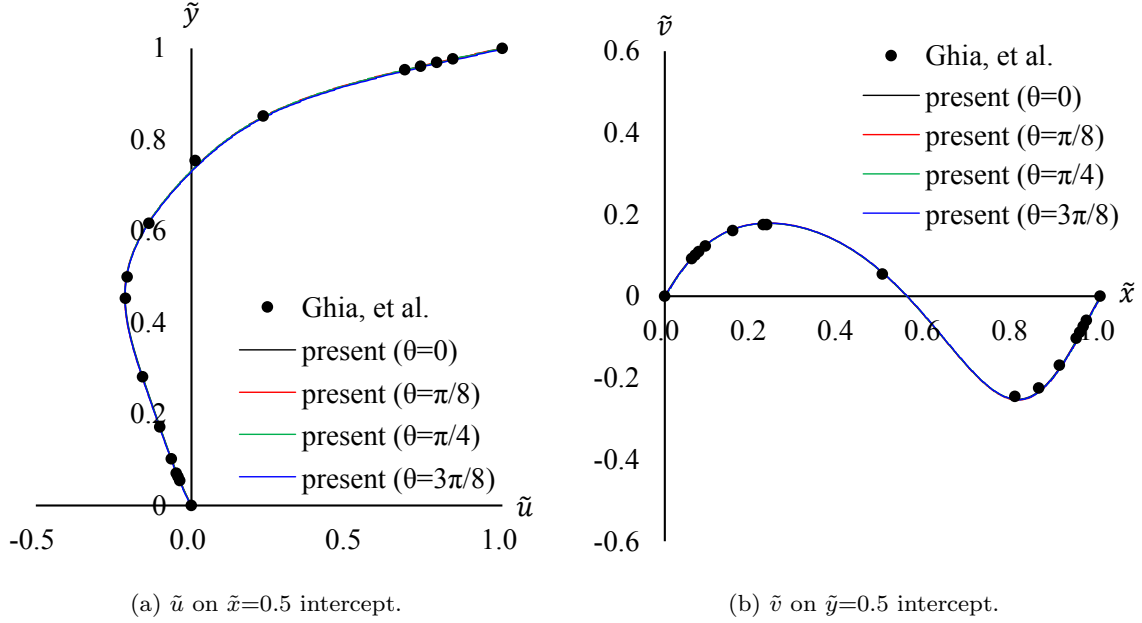


Fig. 2.19: Results of lid-driven cavity flow with oblique computational area at $Re = 100$.

キャビティ流れが計算される実質の計算格子点数は $\widetilde{NX} \times \widetilde{NY}$ で与えられ、計算を行わない格子点も含めた総格子点数は $NX \times NY$ で与えられる。 $NX, NY, \widetilde{NX}, \widetilde{NY}$ は、以下のような関係を持つ。

$$NX = NY, \widetilde{NX} = \widetilde{NY}, \quad (2.171)$$

$$NX \geq \widetilde{NX}(\sin \theta + \cos \theta). \quad (2.172)$$

計算領域の大きさは $0 \leq \tilde{x}, \tilde{y} \leq 1$ とした。計算領域の4つの端辺すべてに Interpolated bounce-back 境界条件を適用し、 $\tilde{x} = 0, \tilde{x} = 1, \tilde{y} = 0$ の辺を速度0の静止壁、 $\tilde{y} = 1$ の辺を速度1の移動壁 ($\tilde{u} = 1; u = \cos \theta, v = \sin \theta$) に設定した。格子ボルツマン法は速度モデルには D2Q9 を、衝突項には BGK モデルを用いた。計算領域の傾き角 $\theta = 0, \frac{\pi}{8}, \frac{\pi}{4}, \frac{3\pi}{8}$ 、レイノルズ数 $Re = 100, 1000, 3200, 5000, 7500$ の条件で計算を行った。計算格子点数は $\widetilde{NX} = \widetilde{NY} = 513$ 、上面境界の移動速度 $\tilde{u}$ に対するクーラン数 ($\tilde{u}/c$) は0.1である。計算ステップ数は、流れが定常になるのに十分に大きい数として、レイノルズ数に応じて数十万から数百万ステップに設定した。

計算結果を Fig. 2.19–2.23 に示す。図には計算領域の傾き角ごとの計算結果を実線で示すとともに、Ghia ら [41] による参照値を点で示している。それぞれの図の (a) は縦方向の中央断面 $\tilde{x} = 0.5$ における横方向の速度 $\tilde{u}$ の分布であり、(b) は横方向の中央断面 $\tilde{y} = 0.5$ における縦方向の速度 $\tilde{v}$ の分布である。いずれのレイノルズ数においても、傾き角に依らず計算結果がほぼ同一であることがわかる。また、それらの計算結果は参照値とよく一致しており、格子に沿った方向の境界だけでなく、斜め方向の境界に対しても応力テンソルを正しく評価できていることがいえる。したがって、Interpolated bounce-back 境界は座標方向に依存しない高精度な境界条件であることが確かめられた。

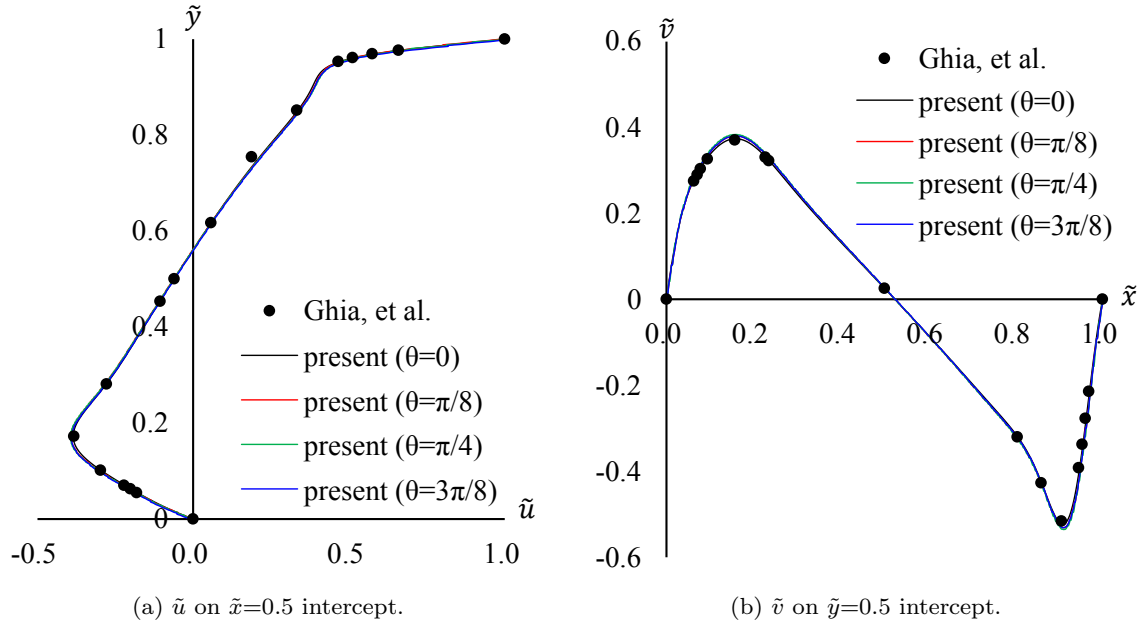


Fig. 2.20: Results of lid-driven cavity flow with oblique computational area at $Re = 1000$.

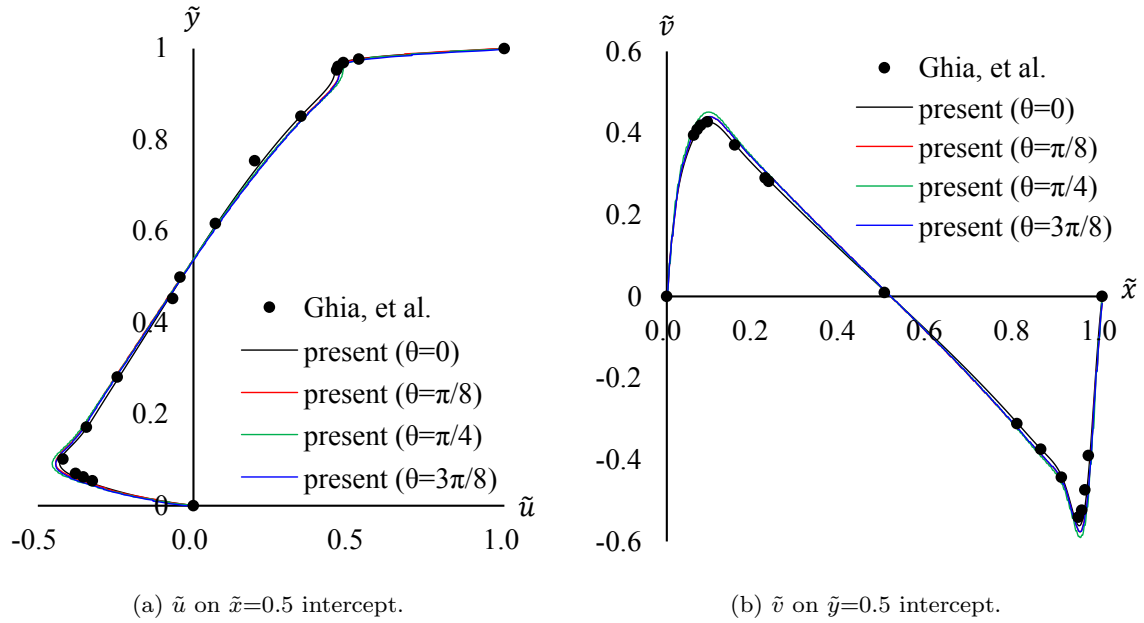


Fig. 2.21: Results of lid-driven cavity flow with oblique computational area at $Re = 3200$.

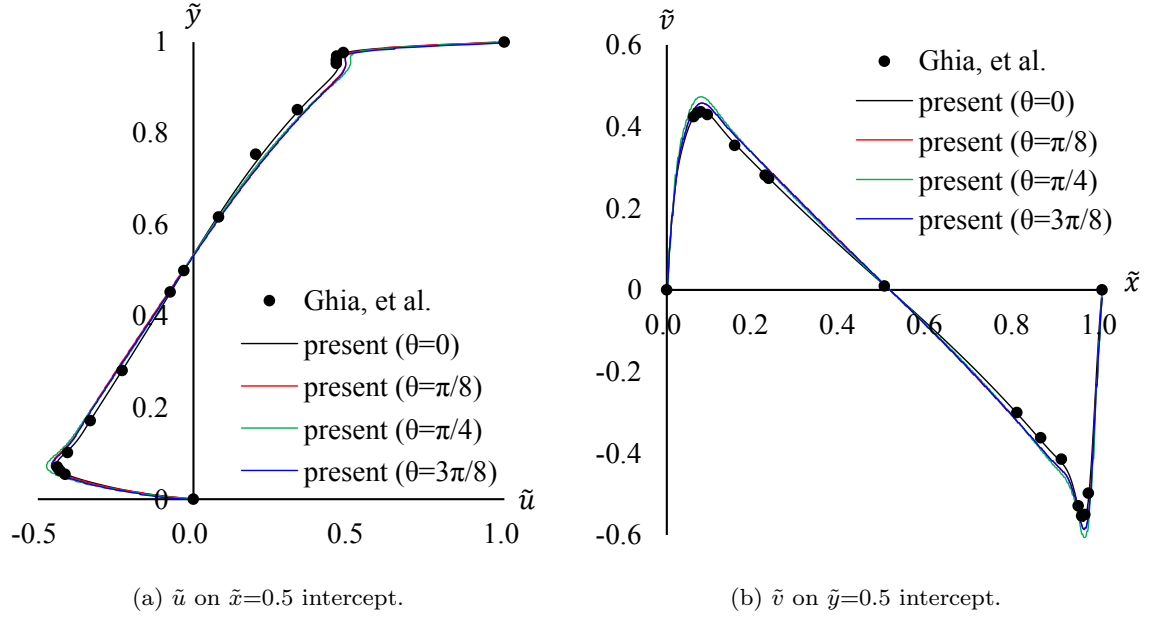


Fig. 2.22: Results of lid-driven cavity flow with oblique computational area at $Re = 5000$.

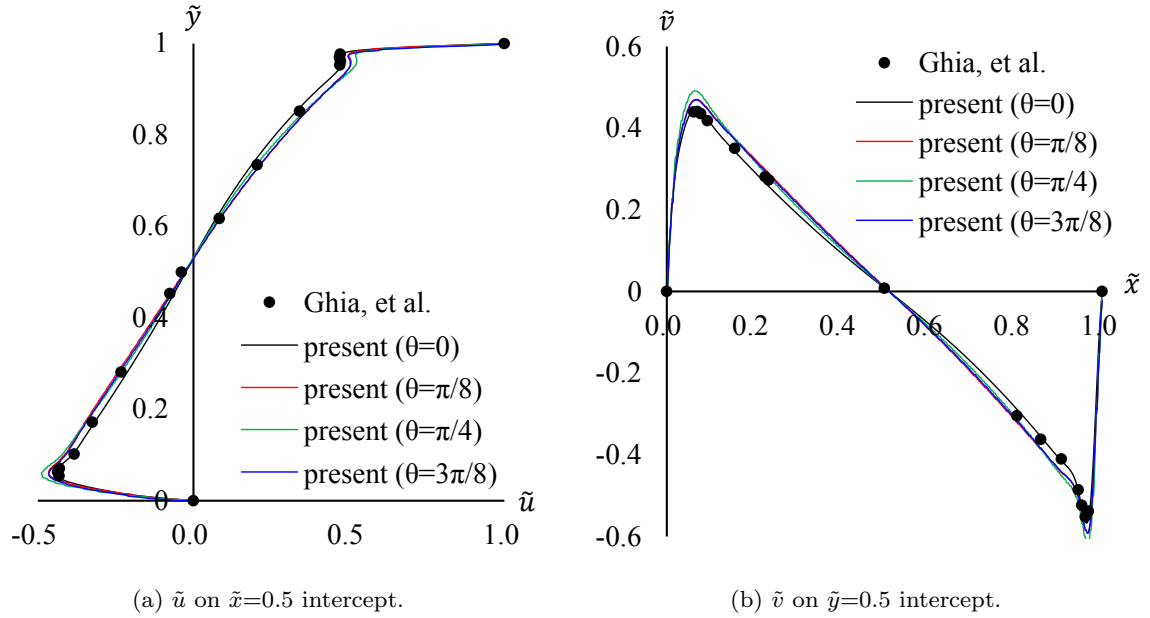


Fig. 2.23: Results of lid-driven cavity flow with oblique computational area at $Re = 7500$.

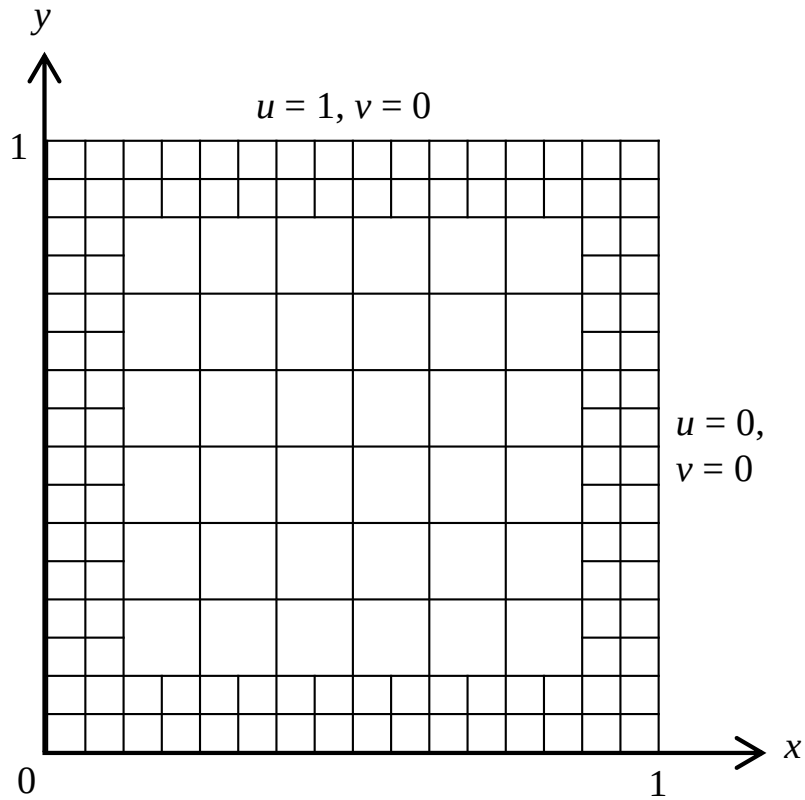


Fig. 2.24: Computational area of lid-driven cavity flow with 2-levels refined mesh.

適合細分化格子法を導入した計算

キャビティ流れにおいて、壁面近傍は速度の変動が大きいため高解像度の格子が必要であるが、計算領域の中央付近には壁面近傍ほど高い解像度は必要ない。そこで、適合細分化格子法により中央付近の解像度を粗くした計算を行うことで、適合細分化格子法の計算精度を検証する。Fig. 2.24, 2.25 に、計算領域 $((x, y) = ([0, 1], [0, 1]))$ のリーフ配置を示す。Fig. 2.24 では細分化レベル差 1, Fig. 2.25 では細分化レベル差 2 の計算を行った。格子ボルツマン法のモデルには D2Q9 BGK を用いた。境界条件として、上面を速度 1 で移動する移動壁、その他の壁面を静止壁に設定した。境界条件の計算には Bounce-back 条件を適用した。また、上面の移動速度に対するクーラン数は 0.1 とした。低レイノルズ数 ($Re = 1000$) および高レイノルズ数 ($Re = 7500$) の条件で計算を行い、解析結果を以下に示す。

■低レイノルズ数 ($Re = 1000$) の計算 Fig. 2.24 および Fig. 2.25 のリーフ配置を用いて、低レイノルズ数 ($Re = 1000$) の解析を実施した。リーフの格子点数には 9×9 を設定した。最も細かい格子は、 129×129 の等間隔格子と同一の解像度を持つ。本計算の格子点数は等間隔格子の計算と比較して、Fig. 2.24 のリーフ配置で 28.0%, Fig. 2.25 のリーフ配置で 33.8% 削減されている。

計算結果を Fig. 2.26 および Fig. 2.27 に示す。グラフには $y = 0.5$ 切片における y 方向の速度 $v|_{y=0.5}$ および $x = 0.5$ 切片における x 方向の速度 $u|_{x=0.5}$ を示している。実線は本計算による結果、点は Ghia

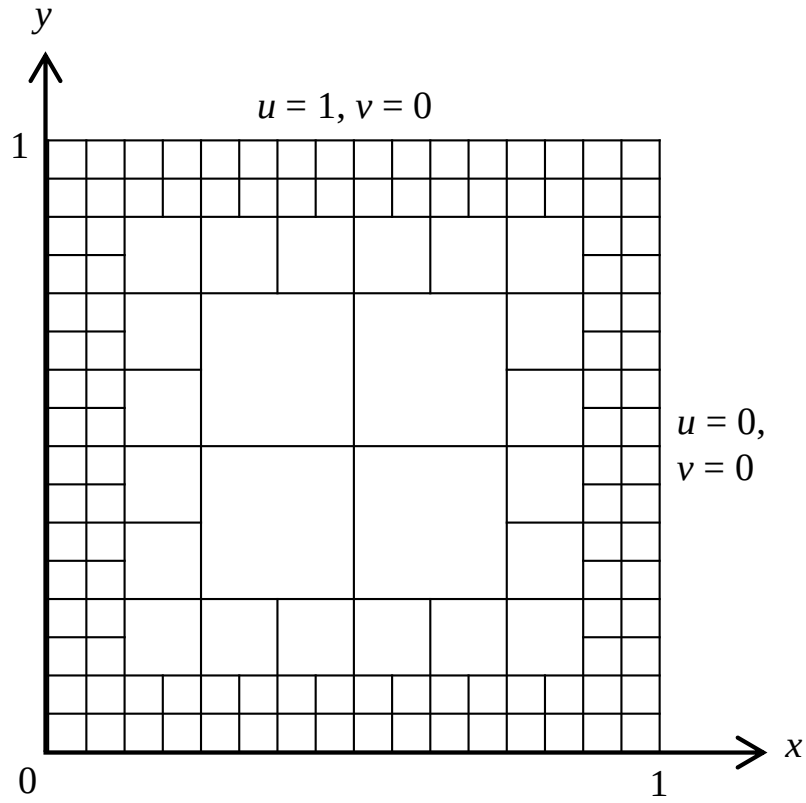
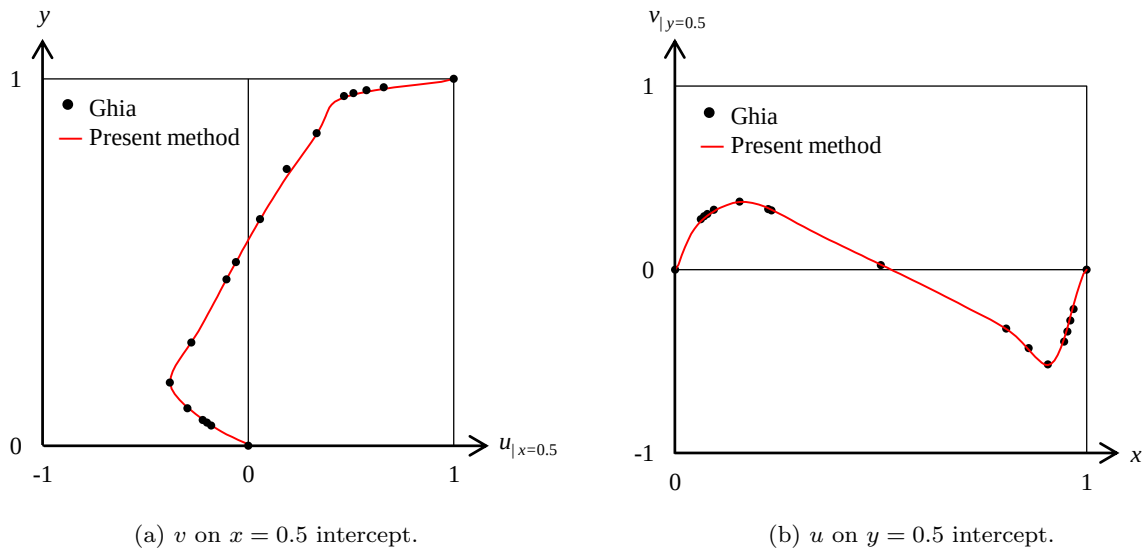


Fig. 2.25: Computational area of lid-driven cavity flow with 3-levels refined mesh.

Fig. 2.26: Result of lid-driven cavity flow with 2-levels refined mesh ($Re = 1000$).

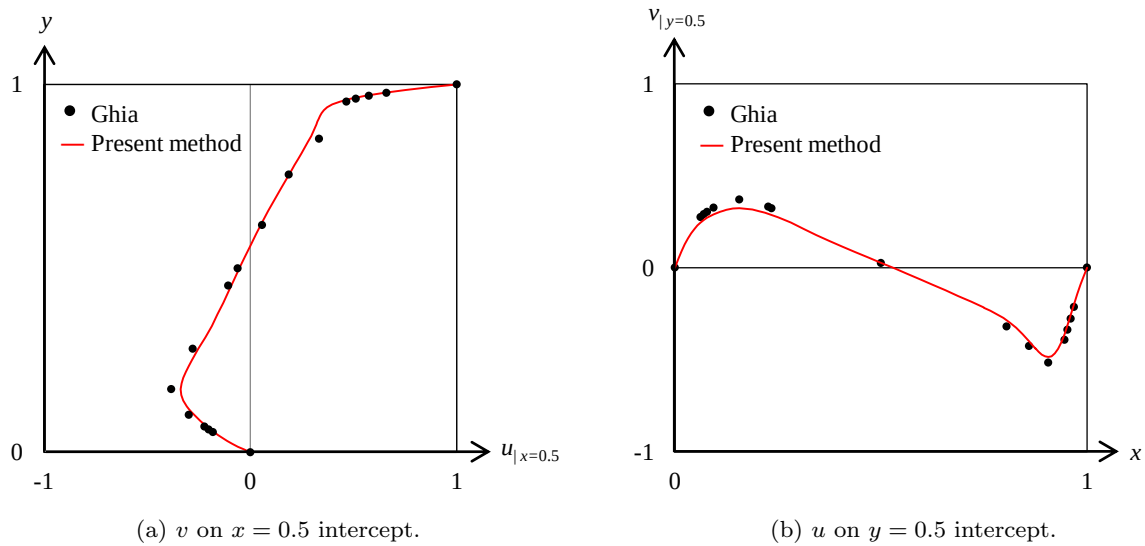


Fig. 2.27: Result of lid-driven cavity flow with 3-levels refined mesh ($Re = 1000$).

ら [41] による参照値である．細分化レベル差 1 の Fig. 2.26 では，計算結果は参照値と非常に良く一致している．細分化レベル差 2 の Fig. 2.27 では，壁面近傍の速度場は良く一致しているが，中心付近の格子をより粗く設定したため，Ghia らの結果に対して若干のずれが生じている．

■高レイノルズ数 ($Re = 7500$) の計算 Fig. 2.24 のリーフ配置を用いて，高レイノルズ数 ($Re = 7500$) の解析を実施した．リーフの格子点数は 17×17 と設定した．最も細かい格子は，細分化格子を適用しない場合の等間隔格子 257×257 と同一の解像度を持つ．本計算の格子点数は，等間隔格子の計算と比較して 40.5% 削減されている．

計算結果を Fig. 2.28 に示す．図のように計算結果は参照値と良く一致しており，本手法は高レイノルズ数の問題に対しても適用可能であるといえる．

以上により，格子ボルツマン法に適合細分化格子法を適用することで壁面近傍の応力テンソルを高精度に捉えらると共に，格子点数を削減可能であることを示した．

2.3.4 3次元球周りの流れ

本計算手法による抗力の評価が妥当であるか確認するため，3次元球周りの流れの計算を行った．計算領域の大きさは $0 \leq x, y, z \leq 1$ とし， $(0.3, 0.5, 0.5)$ の位置に直径 0.1 の球を配置した．適合細分化格子法により，球の近傍に高解像度の格子を集中させた．格子ボルツマン法のモデルには D3Q27 MRT を用い，さらに乱流モデルとして CSM を導入した．球近傍の格子は，球の直径を 25.6 格子で解像しており，周辺領域には解像度が半分の粗い格子を配置した．計算格子点数は，球近傍の格子解像度で計算領域を満たした等間隔直交格子に比べて 79.7% 削減されている．境界条件として， $x = 0, y = 0, y = 1, z = 0, z = 1$

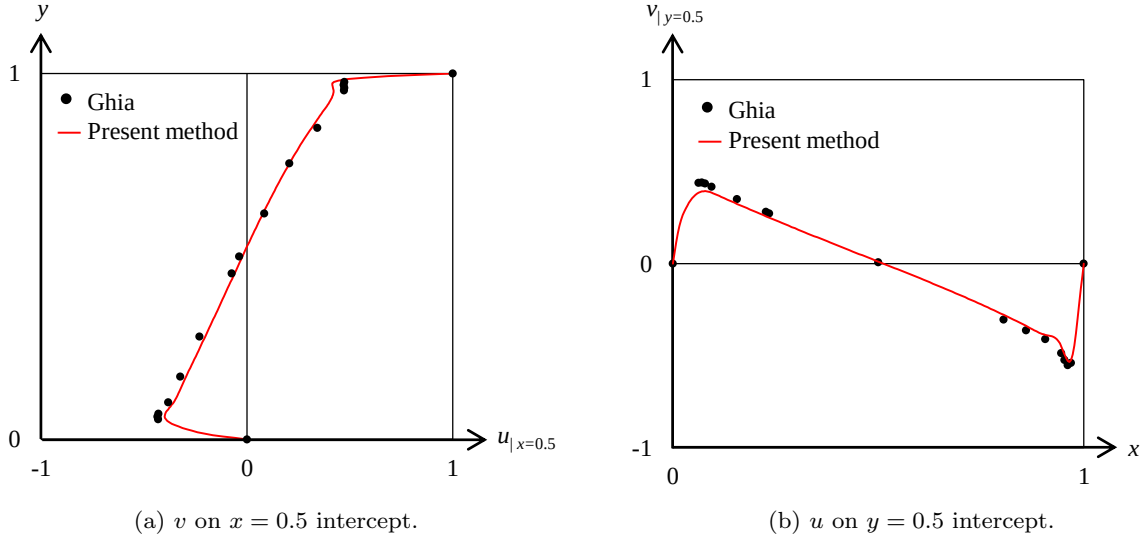


Fig. 2.28: Result of lid-driven cavity flow with 2-levels refined mesh ($Re = 7500$).

の面に $\rho = 1$, $\mathbf{u} = (1, 0, 0)$ の流入境界条件を, $x = 1$ の面に $\partial\rho/\partial x = 0$, $\partial\mathbf{u}/\partial x = \mathbf{0}$ の流出境界を設定した. また, 球表面に Interpolated bounce-back 境界条件を適用した.

流れが十分発達したのち, 球に加わる抗力の時間平均値を求めた. 抗力の時間平均値から, 抗力係数を以下の式により求めた.

$$C_D = \frac{F_x}{\frac{1}{2}\rho_0 U^2 \pi \left(\frac{D}{2}\right)^2} \quad (2.173)$$

ただし, F_x は抗力の主流方向成分, ρ_0 は主流の流体密度, U は主流速度である. D は球の直径で, 本計算条件では $D = 0.1$ である.

レイノルズ数の条件を変化させながらいくつか計算を行い, それぞれ抗力係数を求めた. 結果を Fig. 2.29 に示す. 横軸はレイノルズ数, 縦軸は抗力係数の値である. 点は本手法による計算結果, 実線は実験値 [42] である. 高いレイノルズ数でも計算を安定的に行うことができ, かつ, $Re = 8 \times 10^4$ まで実験値とよく一致する計算結果を得た. したがって, 球周りの流れにおいて抗力を正しく評価できていることが確認できた.

2.3.5 平行平板間乱流

乱流場の計算に対する本手法の精度を確かめるため, 平衡平板間乱流の計算を行う. 計算領域の設定および計算条件を Fig. 2.30 に示す. $L_x \times L_y \times L_z$ は計算領域の大きさである. 主流方向に対して一様な圧力勾配 $\partial p/\partial x = -1$ を加える. 境界条件として, x 軸方向 (主流方向) および z 軸方向 (スパン方

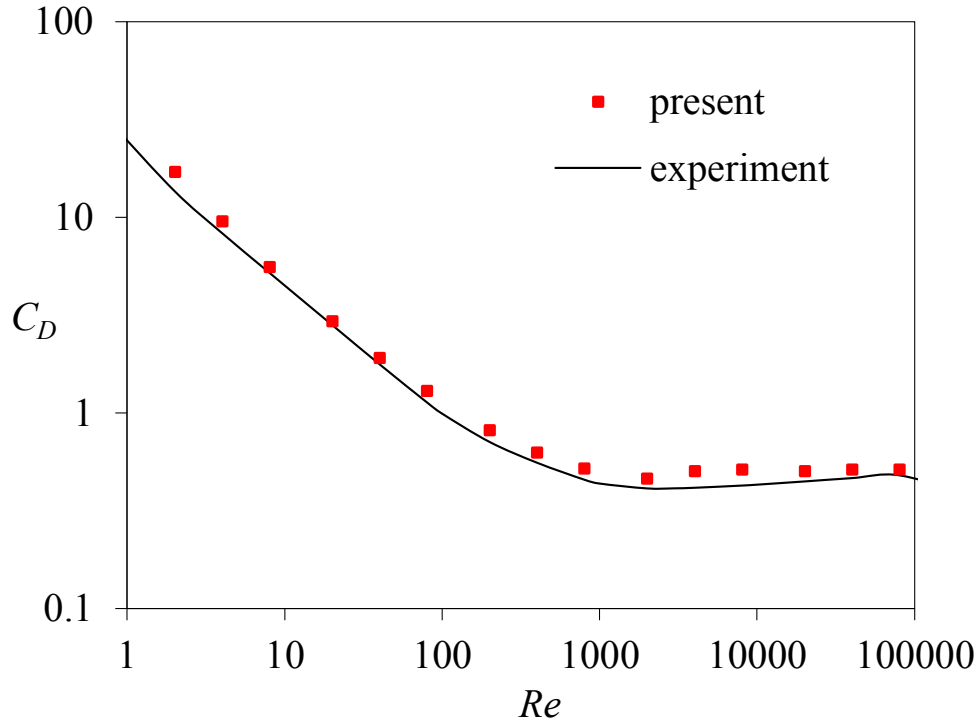


Fig. 2.29: Drag coefficient of a sphere with various Reynolds numbers.

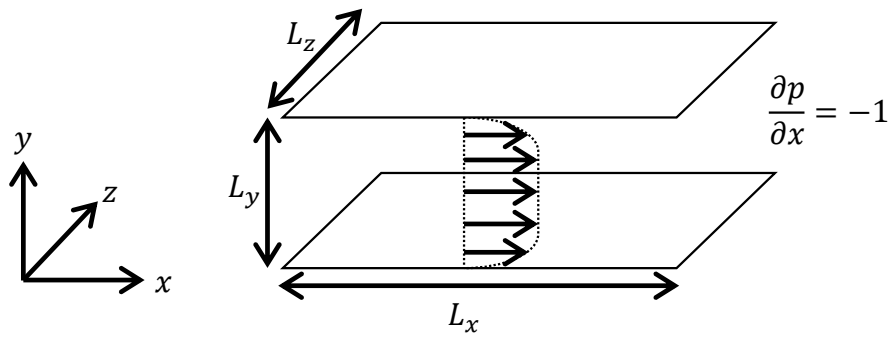


Fig. 2.30: Computational condition of a flow in a parallel-plate channel.

向)に対して周期境界を設定し, y 軸方向(壁方向)の両端に対して Interpolated bounce-back による静止壁境界を設定する. 乱流の成長を促すため, 初期速度に対して摩擦速度の 10 倍程度の振幅の擾乱を与える. 計算領域の壁方向の長さを $L_y = H$, 摩擦レイノルズ数を $Re_\tau = 180$ とし, 計算領域の大きさを $2\pi H \times H \times \pi H$ とする.

計算格子に等間隔格子および適合細分化格子法を用い, 衝突項に D3Q27 MRT モデルを用いて計算す

る．乱流が十分に発達するまで時間発展を繰り返したのち，以下の乱流統計量を測定する．

$$\bar{\mathbf{u}}(y) = \frac{1}{NX \times NZ \times T} \sum_{i=0}^{NX-1} \sum_{k=0}^{NZ-1} \sum_{n=0}^{T-1} \mathbf{u}(x_i, y, z_k, t_0 + n\Delta t) \quad (2.174)$$

$$\sqrt{\mathbf{u}^2}(y) = \sqrt{\frac{1}{NX \times NZ \times T} \sum_{i=0}^{NX-1} \sum_{k=0}^{NZ-1} \sum_{n=0}^{T-1} (\mathbf{u}(x_i, y, z_k, t_0 + n\Delta t) - \bar{\mathbf{u}}(y))^2} \quad (2.175)$$

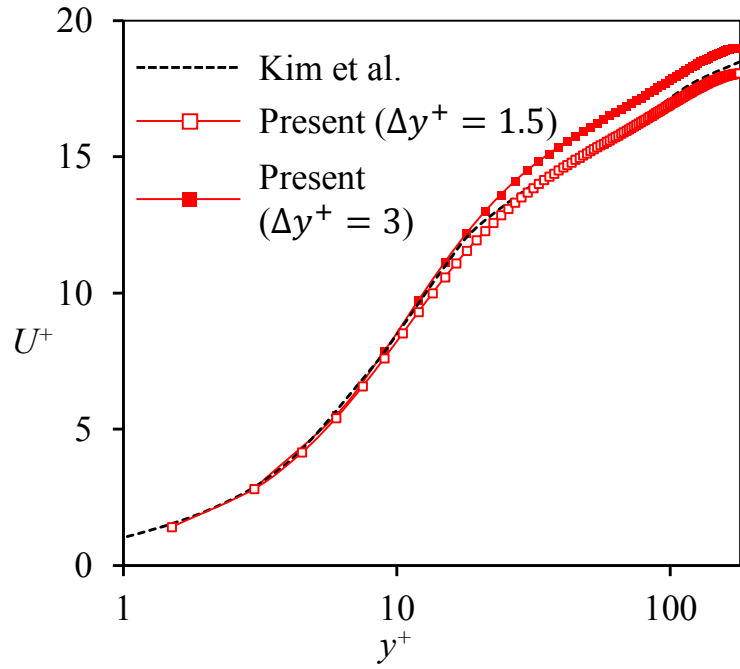
ここで， $\bar{\mathbf{u}}(y)$ は平均流速， $\sqrt{\mathbf{u}^2}(y)$ は乱流強度であり， T は時間方向のサンプリング数， i, k はそれぞれ x, z 方向の座標のインデックス， n は時刻ステップ， t_0 は乱流が十分に発達した時刻である．

等間隔格子での計算には，摩擦速度で無次元化した壁面からの距離 $y^+ = Re_\tau y / (H/2)$ に対して，粘性底層を十分に解像することのできる $\Delta y^+ = 1.5$ ，および，一段粗い $\Delta y^+ = 3$ の2種類の格子解像度で計算を行った．計算結果を Fig. 2.31 に示す．壁面からの距離および乱流統計量は，摩擦速度で無次元化している．破線は Kim ら [43] の DNS 解析による結果であり，実線・塗りつぶし点は $\Delta y^+ = 1.5$ の計算結果，実線・中抜き点は $\Delta y^+ = 3$ の計算結果である． $\Delta y^+ = 1.5$ の計算では参照値とよく一致する結果が得られたが， $\Delta y^+ = 3$ の計算では主流方向速度が参照値よりも大きく見積もられた．

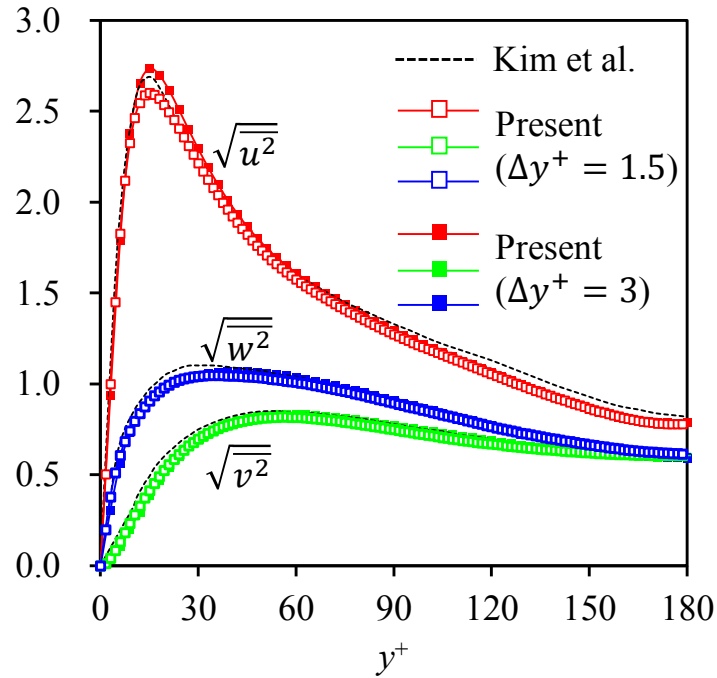
適合細分化格子法では計算領域の長さを整数比で取る必要があるため，参照値の計算条件に近い長さとして $6H \times H \times 3H$ の計算領域を設定した．壁面近傍 ($y^+ \leq H/4, y^+ \geq 3H/4$) に $\Delta y^+ = 3$ の高解像度格子を，壁面から遠い領域 ($H/4 < y^+ < 3H/4$) に $\Delta y^+ = 6$ 粗い格子を配置した．計算結果を Fig. 2.32 に示す．Fig. 2.32 には，Fig. 2.31 と同様の凡例に加えて，解像度の変化する ($y^+ = 90$) の位置を縦の破線で示している．細分化格子の導入により，解像度の変化する境界の付近で乱流強度に非物理的な変動がみられたが，それ以外の領域では $\Delta y^+ = 3$ の等間隔格子と同様の計算結果が得られた．したがって，適合細分化格子法を導入した本計算手法は乱流計算に適用可能である．

2.4 本章のまとめ

本章では，本研究で用いる数値計算手法として適合細分化格子法を導入した格子ボルツマン法について述べるとともに，いくつかの検証計算を実施した．2次元キャビティ流れ，3次元球周りの流れおよび平行平板間乱流の検証計算により，本手法が単層非圧縮流れを妥当に計算できていることを確認した．

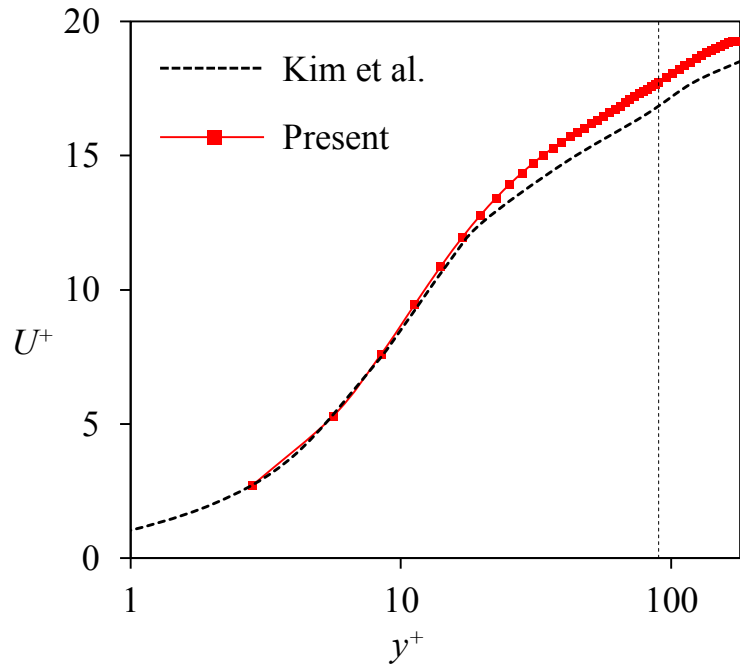


(a) mean velocity.

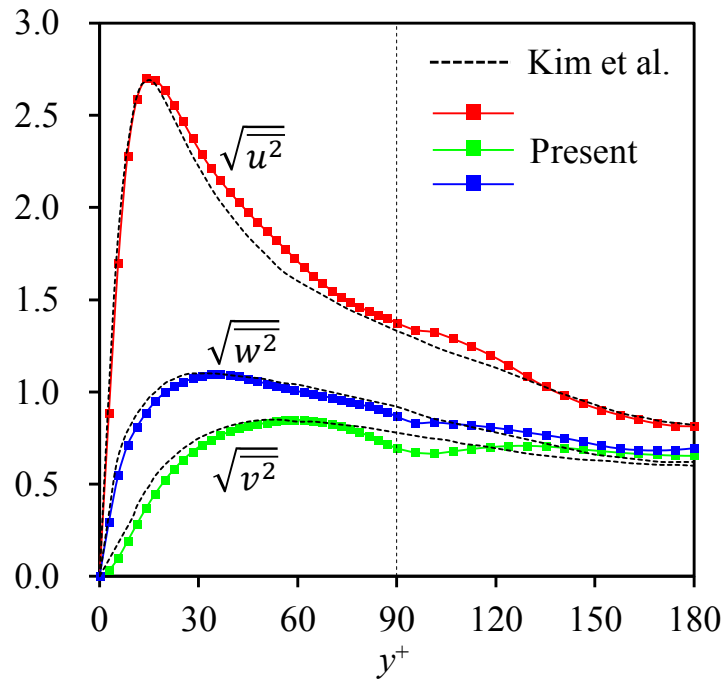


(b) turbulent intensity.

Fig. 2.31: Result of a turbulent channel flow with uniform mesh.



(a) mean velocity.



(b) turbulent intensity.

Fig. 2.32: Result of a turbulent channel flow with adaptive mesh refinement (the vertical dashed lines indicate the coordinate of refinement levels' interface).

第 3 章

単一 GPU 計算の実装と高速化

適合細分化格子法を導入した格子ボルツマン法の単一 GPU 計算への実装および計算の高速化を行う。本章の内容は研究業績 (1) を含む。

3.1 CUDA フレームワークのプログラミングモデル

本研究では NVIDIA 社の GPU を用いており、GPU 計算の実装には CUDA フレームワーク [18] を用いている。CUDA はマルチスレッドのプログラミングモデルである。GPU には数千の演算コアが搭載されており、それらを非常に多数のスレッドで管理する必要があるため、CUDA のスレッドは Fig. 3.1 に示すような階層構造になっている。図中の矢印は、それぞれ 1 つのスレッドを指す。スレッドはまず数個～1024 個ごとに block と呼ばれる単位に纏められ、さらに複数の block が纏められて grid と呼ばれる単位になる。同一 block 内のスレッドは `__syncthreads()` 関数で同期を取ることが可能で、Shared Memory と呼ばれるメモリを用いて高速なデータ通信を行うことも出来る。また、同一 block 内のスレッドは 32 個ごとに Warp と呼ばれる単位に纏められる。同一 Warp 内のスレッドにはレジスタの値を共有する Warp Shuffle という機能があり、CUDA フレームワークの中で最も高速なスレッド間データ通信が可能である。Warp 内のスレッドは命令実行やメモリアクセスにおいても結びつきが非常に強く、計算の高速化を行うにあたり、この Warp という単位は非常に重要である。以下では Warp における命令実行やメモリアクセスの特徴について詳しく述べる。

Warp の 32 スレッドは単一の命令によって完全に同期された演算を実行し、このようなスレッド実行のモデルは SIMT (Single Instruction Multiple Thread) と呼ばれる。Warp に対して 1 度に 1 つの命令しか発行されないため、条件分岐によって各スレッドが異なる命令を実行する必要がある場合、Warp divergence と呼ばれる命令実行の冗長化が発生し、実行性能が著しく低下する。Warp divergence の例を Fig. 3.2 に示す。tid はスレッド番号である。図の左側に示したコードを実行する場合、まず 0 番から 15 番のスレッドで `a=1;` の処理が実行され、次に 16 番から 31 番のスレッドで `a=2;` の処理が実行される。それぞれのステップにおいて、処理を実行しないスレッド (idle threads) が半数存在しているために、スレッドの利用率および実行性能は半分に低下する。Warp divergence の発生を防ぐためには、条件分岐の少ないアルゴリズムを選択したり、Warp 内のスレッドが全て同じ方向に条件分岐するようにス

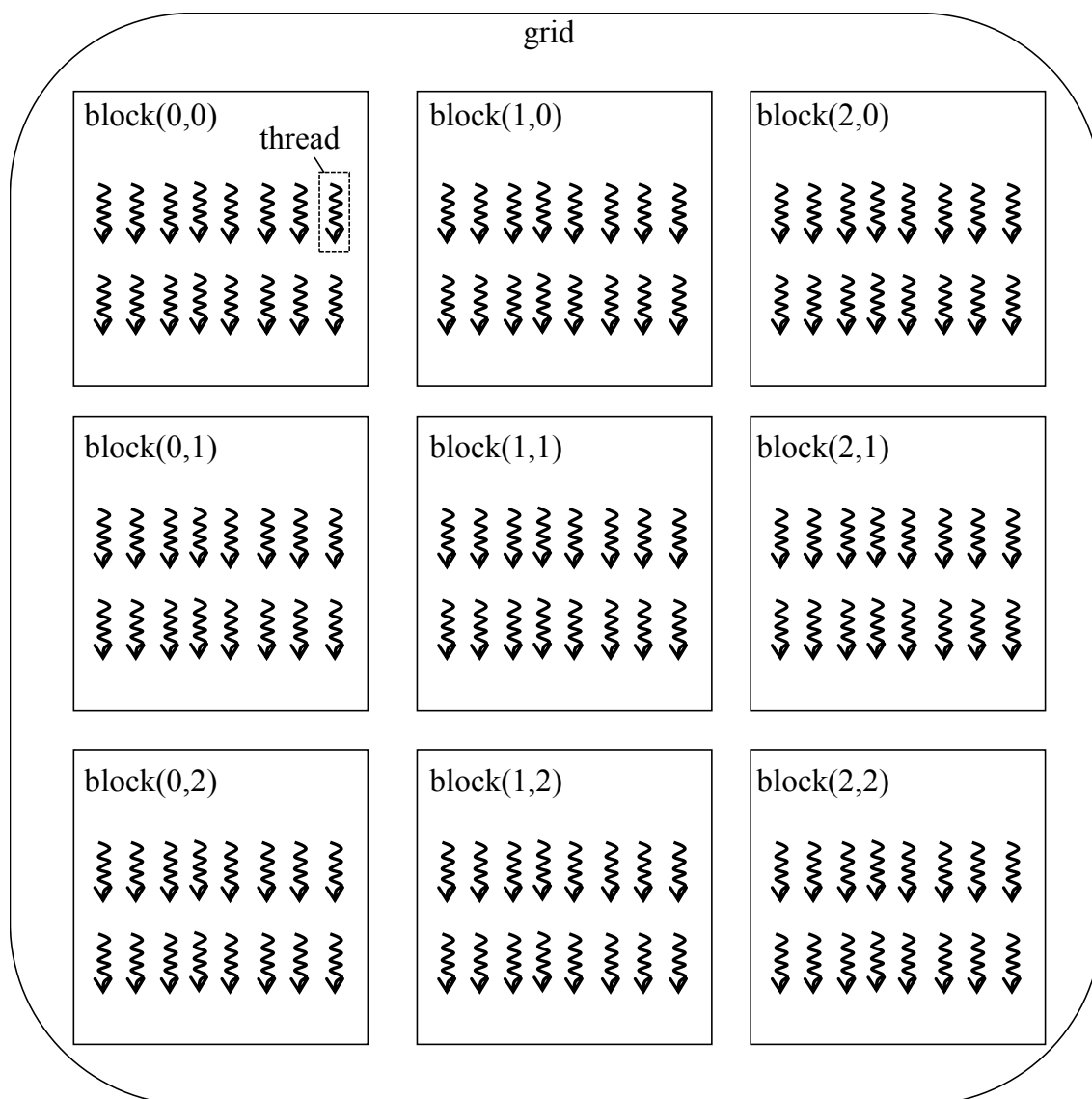


Fig. 3.1: Hierarchical structure of CUDA threads.

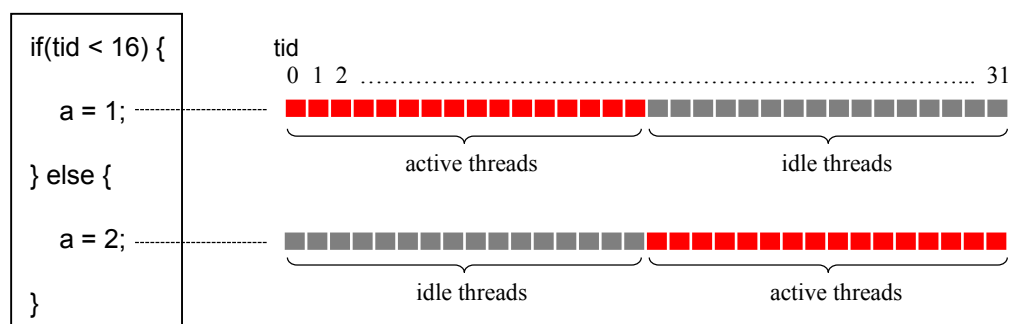


Fig. 3.2: Example of warp divergence.

レッドの配置を変更したりする必要がある。

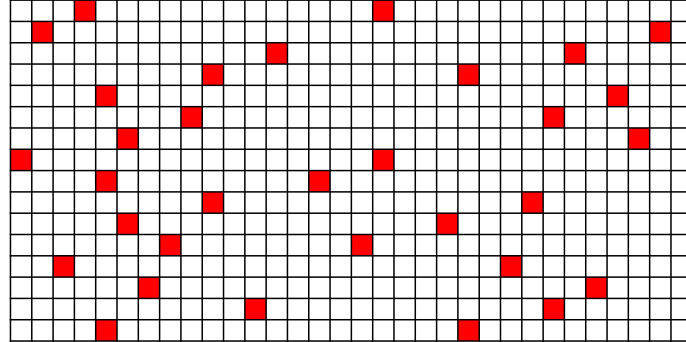
GPU でメモリアクセスを行う際には、小さいデータに何度もアクセスするよりも、大きいデータに一度にアクセスする方が効率が良い。このため、NVIDIA 社の GPU は、Warp 内のスレッドで発行されるメモリアクセス要求を可能な限り纏めて、大きいメモリブロックにアクセスするようなアーキテクチャになっている。複数のメモリアクセス要求を纏めることをコアレスシングという。メモリアクセスとコアレスシングの例を Fig. 3.3 に示す。格子状に並べられたセルはメモリで、横方向に並んだメモリには纏めてアクセスすることができる。赤色のセルは或る Warp 内のスレッドがアクセスするメモリである。Fig. 3.3(a) は完全にランダムなメモリアクセスである。コアレスシング条件が全く満たされておらず、メモリアクセスはほとんど纏められることなく何度も分けて実行される。Fig. 3.3(b) ではメモリアドレスが連続であるが、複数のメモリブロックにまたがっている。コアレスシング条件は一部満たされ、メモリアクセスは数回に分けて実行される。したがって、Fig. 3.3(b) は Fig. 3.3(a) よりも高速なメモリアクセスが可能である。Fig. 3.3(c) ではメモリアドレスが連続であり、かつ同一のメモリブロック内に収まっている。このような場合、コアレスシング条件が完全に満たされ、メモリアクセスは 1 回だけになり、最も高速なアクセスができる。格子ボルツマン法のようなステンスル計算では、ほとんどのメモリアクセスが Fig. 3.3(b) または Fig. 3.3(c) のような規則的なアクセスであり、最適化をあまり意識しなくてもある程度高速なメモリアクセスが可能である。しかし、適合細分化格子法を導入した場合、リーフ間の隣接点参照においてランダムメモリアクセスが発生するため、最適化なしでは実行性能が大幅に低下することが予想される。

3.2 適合細分化格子法の GPU 実装と高速化手法

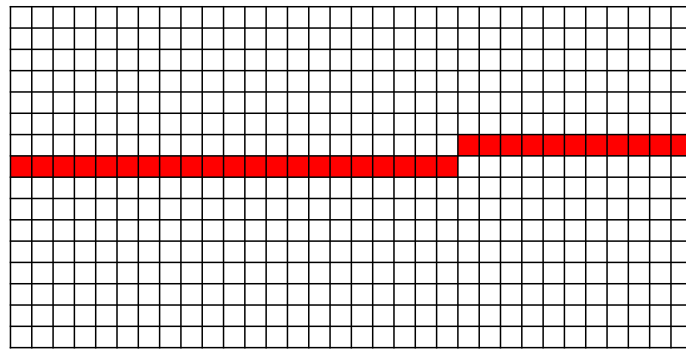
本節では、空力解析に用いる計算コードを開発するため、3 次元計算の実装および高速化を検討する。格子ボルツマン法の速度モデルとして D3Q27、衝突項として格子 BGK モデル、リーフの格子点数として 17^3 を設定した場合に対してチューニングを行う。

3.2.1 Naïve な実装

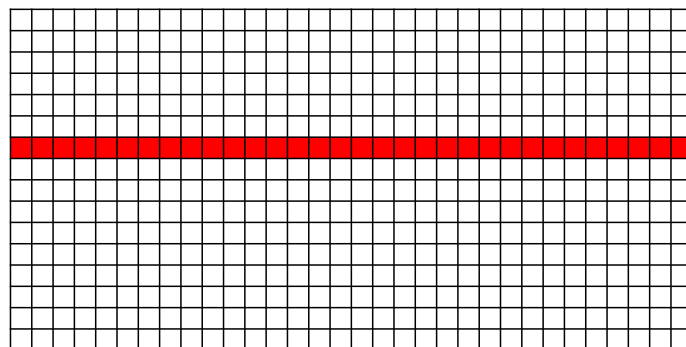
「Naïve な実装」とは、「高速化などを全く考慮せず、アルゴリズムの通りに単純にプログラミングした実装」を意味する。GPU カーネル関数を 1 種類だけ記述し、1 格子点の計算に 1 スレッドを割り当てて計算を行う。GPU カーネル関数のレジスタが溢れないよう、ブロック内のスレッド数のみ調整する。リーフには複数の格子点が割り当てられているため、リーフの内側（内部格子）と外殻部分（外殻格子）で隣接点参照のアクセスパターンが異なる。内部格子は、全計算空間に対して等間隔格子で計算する場合と同じ単純なインデックス計算により隣接点を参照することができる。一方、外殻格子では隣接点の一部が自身のリーフではなく隣接リーフに存在する。隣接リーフの格子点を参照するためにはインデックス計算に加えて隣接リーフの探索を行う必要があり、メモリアクセスの回数が増加する。また、隣接格子点のメモリアドレスが不連続になり、アクセスパターンも連続的でなくなる。隣接リーフにレベル差がある場合には、隣接リーフの格子が自身よりも粗いときと細かいときとで更に処理が分岐する。隣



(a) random access (non-coalesced, non-aligned).



(b) partially-coalescing access (coalesced, non-aligned).



(c) fully-coalescing access (coalesced, aligned).

Fig. 3.3: Coalescing of memory access in GPU.

接リーフのレベルが低い（格子が粗い）場合には、隣接点の位置に格子点が存在しないため、近傍の 8 点から値を補間する．隣接リーフのレベルが高い場合は隣接点の位置に必ず格子点が存在するため、その格子点の値を参照する．従って、外殻格子では多数の条件分岐を含んだ計算となる．

同一 Warp 内のスレッドが条件分岐によって互いに異なる処理を行う場合、Warp divergence と呼ばれる冗長な命令実行が起こる．本研究で用いているリーフの格子点数は 17^3 と小さいため、殆どの Warp に内部格子だけでなく数個の外殻格子が含まれる．このとき、内部格子の計算を担当するスレッドは自身の Warp 内の外殻格子の計算が完了するまで待機することになる．それぞれの外殻格子が異なるアクセスパターンを持つ場合には、外殻格子をアクセスパターンごとに分けて計算する必要がある．スレッドの待機時間は更に長くなる．

殆どの Warp は、28 個の内部格子と 4 個の外殻格子の計算を行う．例えば、外殻格子点が x 軸方向の境界面にある場合、アクセスパターンは x 軸の正方向の（右）面の格子と x 軸の負方向の（左）面の格子で 2 つに分かれる．このため、1 Warp の計算は、内部格子、左面の外殻格子、右面の外殻格子の 3 種類の処理に分岐する．条件分岐が多くなると、待機状態のスレッドの数が増加し、命令を実行しているスレッド（アクティブスレッド）の数が減少する．外殻格子の計算を行っている間のアクティブスレッド数は 1 Warp（32 スレッド）あたりわずか 2 スレッドであり、残りの 30 スレッドが待機状態となるため、並列化効率が非常に低くなる．

3.2.2 内部格子と外殻格子のカーネル分割および内部格子計算の高速化

外殻格子の計算は内部格子に比べて低速で、内部格子と外殻格子の計算スレッドの間で発生する Warp divergence は深刻な実行性能低下を招く．Warp divergence の影響を低減するため、内部格子と外殻格子を異なるカーネル関数に分割して計算する．これにより、内部格子に対する計算カーネルでは実行性能が著しく向上する．しかし、リーフあたりの格子点数が 17^3 と非常に小さいため、 129^3 や 257^3 のように大きな等間隔直交格子に比べると、内部格子の計算で性能を出すのはそれほど容易ではない．

GPU におけるメモリアクセスにはコアレッシング条件が存在し、適切にアライメントされた 32, 64 または 128 Byte のブロックに Warp が一斉にアクセスするような場合に最大の実効メモリバンド幅が得られる．たとえば、Warp が単精度浮動小数点数を読み込む場合、32 個のデータがメモリ上で連続しており、かつ、メモリのブロックの先頭アドレス値が 128, 192 など 64 の倍数となっているときにメモリアクセスが最速になる．リーフの格子は 17^3 の点が規則正しく並べられているが、1 列を 64 の倍数の Byte 数で取ることが出来ず、コアレッシング条件を満たしていない．そこで、リーフの格子のメモリアドレスを GPU が効率的にアクセスできるように並べ替える．リーフの格子を $16 \times 17 \times 17$ と $1 \times 17 \times 17$ の部分に分け、 $1 \times 17 \times 17$ の部分には、それぞれの速度分布関数の参照方向に応じて $i = 0$ または $i = 16$ の部分を取る．すなわち、 x 軸の負方向の隣接点を参照する速度分布関数では $i = 16$ の側を、 x 軸の正方向の隣接点を参照する速度分布関数では $i = 0$ の側を $1 \times 17 \times 17$ に取る．これにより、読み込み・書き込みとも効率よくアクセスすることができる．また、Half Warp（Warp の前半または後半の 16 スレッド）あたり内部格子の 1 列分（15 格子点）の計算を行い、1 つのスレッドでは計算を行わない．これにより、コアレッシング条件が大きく改善され、計算が高速化されることが期待できる．

Half Warp で計算をしていなかった 1 スレッドも活用して完全なコアレスシングを実現する。まず、Half Warp 内の 16 スレッドで、 $16 \times 17 \times 17$ の部分の格子から、読み込むべき 15 個の隣接格子点を含む 1 列 16 個の格子点をコアレスアクセスで読み込む。次に、Warp Shuffle により 16 個読み込んだ格子点のうち必要な 15 個を、必要なスレッドに転送する。ここで、Warp Shuffle とは、同一 Warp 内のスレッド同士でレジスタの値を共有する機能で、1 クロックで値の転送が完了するため、Shared Memory を使うよりも高速にスレッド間で値の転送が出来る。次に、27 個の速度分布関数の相互作用（衝突過程）を計算する。最後に、計算結果をメモリに書き込む際には、読み込み時と逆の方向に Warp Shuffle を行い、完全コアレスなメモリ書き込みを行う。このとき、Half Warp のうち 1 スレッドは外殻格子のメモリ位置に無駄なデータを書き込むことになるが、外殻格子の計算で上書きされるため計算結果に影響はない。無駄な書き込みの分だけメモリアクセス量は増大するが、コアレスシング条件により実行時間は短縮され、Time-to-Solution の実行性能が向上する。

3.2.3 マザーリーフ法による外殻格子点数の削減

リーフの外殻格子は隣接点参照のメモリアクセスポターンが悪く、高い実行性能が得られない。外殻格子の数が内部格子に比べて相対的に少ないほど、計算の実行性能は高くなる。リーフあたりの外殻格子の数は、3 次元の場合リーフの格子点数の 3 分の 2 乗に比例するため、リーフの格子点数を 17^3 から 33^3 , 65^3 などと増加させることで外殻格子の数を相対的に減少させることができる。しかし、リーフの格子点数が増加すると細分化の緻密さが低下する（無駄に細かい格子を多く使う）ことにつながるため、適合細分化格子法の実装として適していない。

格子適合性を維持しつつ外殻格子の数を削減する手法として、マザーリーフ法を導入する。マザーリーフ法では、本来末端のリーフが持つべき格子を、兄弟リーフで纏めて共通の親ノード（マザーリーフ）に持たせる。兄弟ノードの中にリーフと単なるノードの両方が存在する場合には、単なるノードにも仮の格子を割り当て、マザーリーフで纏めて管理する。簡単のため、Fig. 3.4 に示す 2 次元の例を用いて説明する。(a) はリーフに格子を割り当てるような元のデータ構造、(b) はマザーリーフ法を用いた時のデータ構造である。木構造において緑色で示したノードに対して格子を割り当てている。Fig. 3.4 のようにノード 2,3,4,5,6,7,8 がリーフであるような場合、Fig. 3.4 (b) のようにリーフ 5,6,7,8 の格子をマザーリーフ 1 で管理し、リーフ 2,3,4 の格子およびノード 1 の仮の格子をマザーリーフ 0 で管理する。このとき、マザーリーフ 0,1 の重複する領域では複数の解像度の格子が重なって存在しているが、細かい格子（マザーリーフ 1）から粗い格子（マザーリーフ 0）への時空間補間を行い、粗い格子の値を補間値で上書きする。また、マザーリーフ法を導入したことで格子点数が 17^3 から 33^3 に変更されたため、内部格子の高速化手法の実装を以下のように置き換える。メモリアドレスの並び替えでは、「 $16 \times 17 \times 17$ と $1 \times 17 \times 17$ 」を「 $32 \times 33 \times 33$ と $1 \times 33 \times 33$ 」に変更する。冗長なスレッドを含む計算では、「Half Warp で 1 列中の 15 個の内部格子点と冗長な 1 格子点を計算する」を「1 Warp で 1 列中の 31 個の内部格子点と冗長な 1 格子点を計算する」に替える。

マザーリーフ法は細分化格子の構造を変化させることなく外殻格子数を削減することができ、連続アドレスで計算できる格子点数を増加させることができる。計算領域の総格子点数は粗い格子と細かい格

子が重なり合う分だけ増大するが、総格子点数の増大率は、兄弟ノードのうち 7 つのノードに子ノードを生成するような領域分割を繰り返した場合でも最大で 4.50% ($\lim_{d \rightarrow \infty} (\sum_{l=0}^{d-1} 7^l + 7^d \times 8) / \sum_{l=0}^d 7^l$) であり、冗長な計算に対するメモリ使用量や計算量は小さい。マザーリーフの格子点数は、リーフの 17^3 に対して 33^3 となり、そのうち外殻格子点数はマザーリーフで 6146 個 ($33^3 - 31^3$)、リーフで 1538 個 ($17^3 - 15^3$) である。マザーリーフ 1 つあたりの外殻格子点数はリーフに比べて 4 倍程度に増加するが、計算領域中のマザーリーフの総数はリーフの 8 分の 1 程度であるため、計算領域全体の外殻格子点数は 2 分の 1 程度に減少する。

マザーリーフ法と同様に Octree のリーフの一段上を利用する手法は、既にいくつか提案されている。たとえば、Cyril らによるボクセルベースの CG レンダリング手法 GigaVoxels [44] では、メモリコヒーレンスの改善やリーフの隣接関係を記憶するメモリ量の削減に有効であることが主張されている。Hsi-Yu らの天体計算コード GAMER [25] では、リーフに袖領域を取る実装において袖領域の格子点数を大幅に削減できることが示されている。一方、マザーリーフ法は、メモリアクセス効率の点で不利な外殻格子点の数を削減することで計算の高速化を図るものであり、それぞれ目的が異なっている。

3.2.4 C++ テンプレートによる外殻格子のカーネル生成

外殻格子には、格子点の位置によって複数のアクセスパターンが存在する。Naïve な実装ではそのアクセスパターンをカーネル内の条件分岐文で判定していた。多数の条件分岐があるため、条件分岐の評価に多大な計算時間が費やされて、アクセスパターンの異なる外殻格子が同一 Warp 内で計算されることによる Warp divergence も発生し、実行性能は著しく低下していた。

外殻格子計算の高速化のためには、存在するアクセスパターンをすべて列挙し、複数のカーネル関数に分割して記述する方法が考えられる。しかし、D3Q27 格子ボルツマン法においては、外殻格子点の位置によってパターンが 26 通りあることに加え、そのそれぞれに 27 通りの隣接点参照があるため、それらの組み合わせによるアクセスパターンの総数は 702 に達する。これほど多数のパターンを手動で記述することは、プログラムの生産性を考えると現実的でない。

本研究では、Harris の C++ テンプレートを用いた GPU カーネル生成による最適化手法 [45] を応用し、プログラムの生産性を損なうことなく、外殻格子における 702 通りすべてのアクセスパターンを列挙する高速化手法を提案する。まず、格子点の位置を同一のアクセスパターンごとに分類する。リーフの格子点について、隣接点へのアクセスパターンごとの分類を Fig. 3.5 に示す。すなわち、リーフの格子点は次の 27 パターンに分類される。

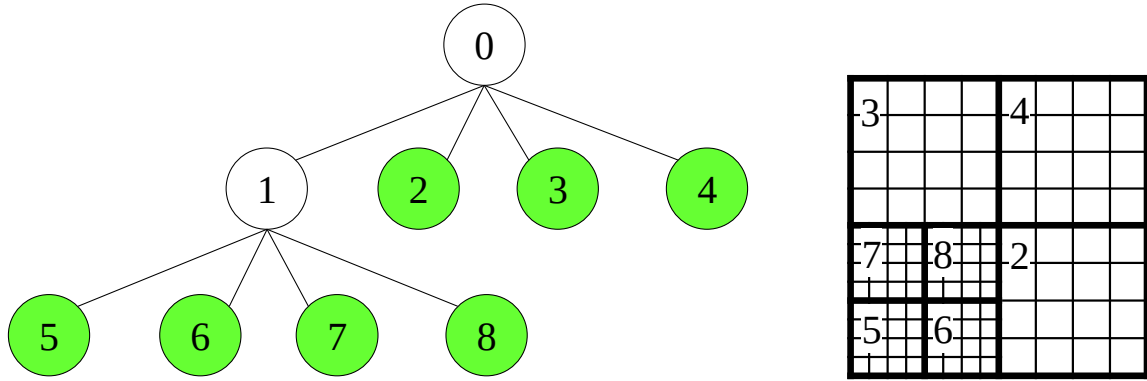
$$(i, j, k) \in \{1, 2, \dots, m-2\}^3 \quad (3.1)$$

$$(i, j, k) \in \{0, m-1\} \times \{1, 2, \dots, m-2\}^2 \quad (3.2)$$

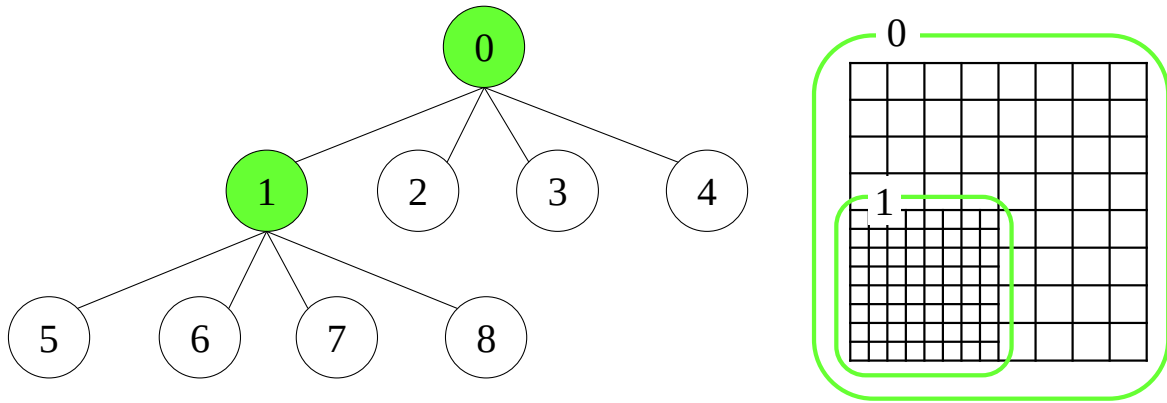
$$(i, j, k) \in \{0, m-1\}^2 \times \{1, 2, \dots, m-2\} \quad (3.3)$$

$$(i, j, k) \in \{0, m-1\}^3 \quad (3.4)$$

m^3 はリーフの格子点数であり、 $(i, j, k) \in \{0, 1, 2, \dots, m-1\}^3$ はリーフの格子点のインデックスである。式 (3.1) は内部格子、式 (3.2–3.4) は外殻格子である。外殻格子はさらに、式 (3.2) の 6 種類の面、



(a) Traditional data structure.



(b) Data structure of mother-leaf method.

Fig. 3.4: Mother-leaf method (2D example).

式 (3.3) の 12 種類の辺, 式 (3.4) の 8 種類の頂点に分けられる. 面の格子は 1 つの隣接リーフを参照し, 辺の格子は 3 つの隣接リーフへの参照を必要とする. 頂点格子には 7 つの隣接リーフにアクセスするパターンが存在する. 隣接点参照のパターンごとに分類した格子のパターンを示す変数として, g_i, g_j, g_k

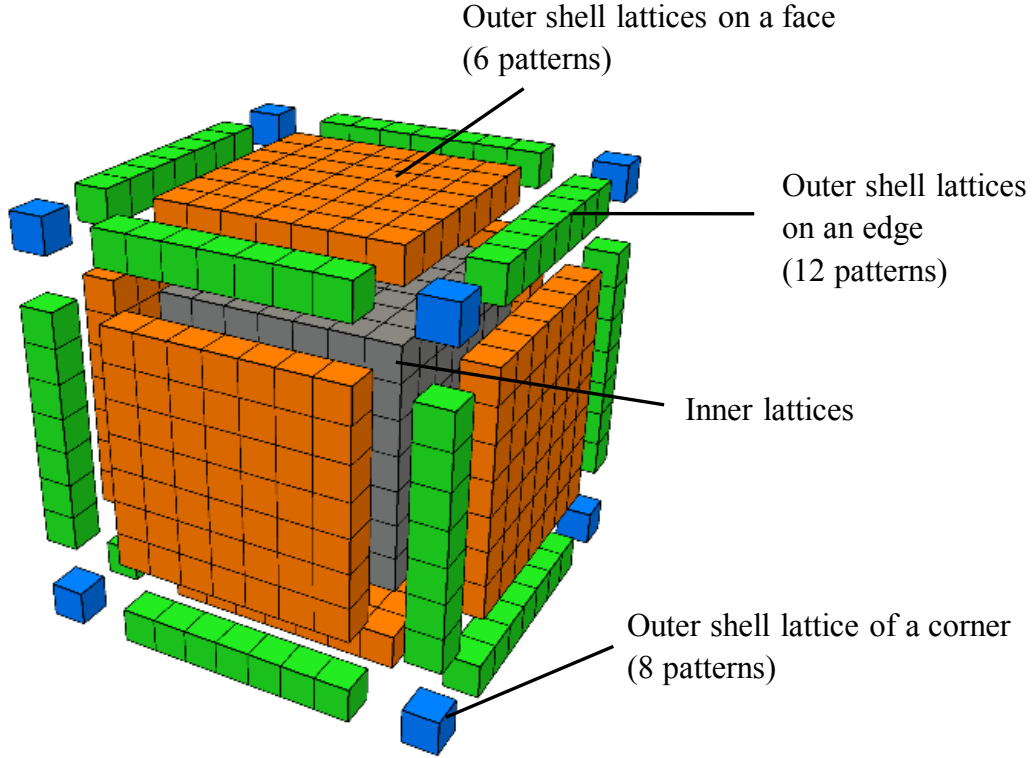


Fig. 3.5: Classification of leaf lattices based on stencil access patterns.

を次のように定義する.

$$gi = \begin{cases} -1 & (i = 0) \\ 0 & (i = 1, 2, \dots, m-2) \\ 1 & (i = m-1) \end{cases} \quad (3.5)$$

$$gj = \begin{cases} -1 & (j = 0) \\ 0 & (j = 1, 2, \dots, m-2) \\ 1 & (j = m-1) \end{cases} \quad (3.6)$$

$$gk = \begin{cases} -1 & (k = 0) \\ 0 & (k = 1, 2, \dots, m-2) \\ 1 & (k = m-1) \end{cases} \quad (3.7)$$

gi, gj, gk をテンプレート引数とし, Program 1 のような GPU カーネル関数を記述する. 各スレッドの担当するリーフの番号や格子点のインデックスは, gi, gj, gk およびスレッド番号 (`threadIdx`, `blockIdx`) の組み合わせによって定める. Program 1 のカーネル関数を,

```
leaf_bound<-1,0,0><<<gridDim,blockDim>>>(...),
leaf_bound<1,0,0><<<gridDim,blockDim>>>(...)
```

などと, gi, gj, gk を変化させて呼び出すことで, C++ テンプレートの機能によってそれぞれのアクセ

Program 1 C++ template kernel for boundary grids.

```

template<int gi, int gj, int gk>
__global__ void leaf_bound(...) {
    /* 格子ボルツマン法の速度分布関数 */
    float f[27];
    /* リーフの番号 l および格子点のインデックス i,j,k を計算 */
    const int l = ...;
    const int i = ..., j = ..., k = ...;
    /* 格子ボルツマン法の並進過程に基づく隣接点参照 */
    f[0] = read_stencil<gi, gj, gk, 0, 0, 0>(l, i, j, k, 0,...);
    f[1] = read_stencil<gi, gj, gk, 1, 0, 0>(l, i, j, k, 1,...);
    f[2] = read_stencil<gi, gj, gk, 0, 1, 0>(l, i, j, k, 2,...);
    ... /* 残り 24 個の read_stencil 関数も同様に呼び出す */
    /* 格子ボルツマン法の衝突項を計算 */
    f[0] = ...;
    f[1] = ...;
    ...
    /* f[] の計算結果をグローバルメモリに書き込む */
    .....
}

```

Program 2 Detail of the subroutine “read_stencil.”

```

template<int gi, int gj, int gk, int si, int sj, int sk>
__device__ float read_stencil(int l, int i, int j, int k, int alpha, ...) {
    float retval;
    const int ni = (gi == si) ? si : 0;
    const int nj = (gj == sj) ? sj : 0;
    const int nk = (gk == sk) ? sk : 0;
    if(ni != 0 || nj != 0 || nk != 0) {
        /* 隣接リーフの格子点を参照 */
        retval = read_stencil_neighbor_leaf<ni, nj, nk>(l, i+si, j+sj, k+sk, alpha);
    } else {
        /* 自分自身のリーフの格子点を参照 */
        retval = fg[ idx(l, i + si, j + sj, k + sk, alpha) ];
    }
    return retval;
}

```

Table 3.1: Execution environment.

CPU	Intel Xeon X5670 (2.93 GHz)
Main memory	54 GB
Operating system	SUSE Linux Enterprise Server 11 SP3
GPU	NVIDIA K20Xm
C++ Compiler	GNU Compiler Collection 4.3.4
CUDA driver	version 6.5

スパターンに対応する 26 通りのカーネル関数が生成される．なお，`leaf_bound<0,0,0>` は内部格子に対応しており，実際には使用しない．

`read_stencil` は隣接点の値を参照するデバイス関数である． gi, gj, gk に加えて隣接点の方向を表す変数 $(si, sj, sk) \in \{-1, 0, 1\}^3$ をテンプレート引数にとり，Program 2 のように記述する． (gi, gj, gk) の組み合わせは 26 通り， (si, sj, sk) の組み合わせは 27 通りあり，全部で 702 通りのアクセスパターンに対応する `read_stencil` 関数がテンプレート展開によって生成される．また，`read_stencil_neighbor_leaf` は，隣接リーフの方向 (ni, nj, nk) に従って隣接リーフの探索および格子の値の参照を行うサブルーチンである．`idx` はリーフ内の格子を参照するためのインデックス計算を行う関数である．`fg` はグローバルメモリ上に記憶されている速度分布関数の値へのポインタである．Program 2 における if 文は隣接点がリーフ内にあるかどうかの判定であり，隣接点がリーフ外（隣接リーフ）にあるとき真，隣接点がリーフ内にあるとき偽となる．格子ボルツマン法の場合は条件分岐が全て静的であるため，以上のようなアクセスパターンを適切に分類したテンプレート関数がコンパイル時に展開される．これにより，条件分岐の評価に要する時間や Warp divergence による実行性能の低下が無くなり，実行性能の向上を図ることができる．

カーネル関数を 26 通りに分割したことでそれぞれのカーネルが担当する格子点数が少なくなっており，単一のカーネルでは GPU 計算に十分な並列数を取ることができない．そこで，CUDA のストリームを用いて複数のカーネルを同時実行することで並列数を確保する．ストリームの実行はランタイム時に動的にスケジューリングされるが，Table 3.1 に示す実行環境において最大で 11 個のカーネルが同時実行されることを確認している．

3.3 実行性能測定

適合細分化格子法を導入した格子ボルツマン法に 3.2 で説明した高速化手法を適用し，実行性能測定を行った．時空間補間や境界条件などの特殊な処理を含まないステンシル計算の部分のみで性能を測定するため，全計算領域でリーフの細分化レベルは一定，境界条件は周期境界条件とした．浮動小数点演算はすべて単精度とした．実行環境を Table 3.1 に示し，計算条件を Table 3.2, Table 3.3 に示す．3.2.3 項で説明したマザーリーフ法の適用により，同一の問題であっても総格子点数，内部格子点数および外殻格子点数が変化するため，マザーリーフ法あり・なし両方の計算条件を示している．

Table 3.2: Computational condition without mother-leaf method.

mesh size per leaf	4913 (17^3)
number of leaves	512
number of total grids	2,515,456
inner lattices	1,728,000
outer shell lattices	787,456
type of floating point number	single precision

Table 3.3: Computational condition with mother-leaf method.

mesh size per mother-leaf	35,937 (33^3)
number of mother-leaves	64
number of total grids	2,299,968
inner lattices	1,906,624
number of outer shell lattices	393,344
type of floating point number	single precision

チューニングを行っていない Naïve な実装に対して、3.2 節で説明した高速化手法を 1 つずつ加える形で適用していき、1 タイムステップの計算時間を測定することで、それぞれの手法の効果を確かめた。計算時間は、内部格子、外殻格子、全格子合計（内部＋外殻）のそれぞれについて測定した。計算時間の測定には CUDA プロファイラ（nvprof および NVIDIA Visual Profiler）を用い、内部格子カーネルの実行時間、および 26 個の外殻格子カーネルのうち最初のカーネルの起動から最後のカーネルの実行完了までの時間を測定した。なお、GPU カーネルの起動に掛かる時間は 1 カーネルあたり数百ナノ秒と非常に短いため、測定から除外した。

3.3.1 実行時間の測定

1 ステップのみの計算を実行し、実行時間を測定した。測定結果を Table 3.4 に示す。Inner は内部格子の計算時間、Outer Shell は外殻格子の計算時間であり、Total は合計の計算時間を表している。Table 3.4 の [3.2.1] は高速化手法を適用していない Naïve な実装を指している。Naïve な実装では内部格子と外殻格子を同一カーネルで計算していたため、合計の計算時間のみを示している。[3.2.2], [3.2.3], [3.2.4] は適用した高速化手法を表しており、その数字は項の番号に対応している。

内部格子と外殻格子のカーネル分割 [3.2.2] によって 5.39 倍の高速化を達成した。外殻格子の計算に内部格子の 8.98 倍程度の時間を要しており、外殻格子は計算効率が非常に低いことがわかる。

マザーリーフ法 [3.2.3] の適用により、外殻格子の格子点数が削減され、外殻格子の計算を 1.96 倍に高速化することができた。内部格子は、格子点数が増加したために計算時間もわずかに増大しているが、

Table 3.4: Computation time on each tuning applied.

	Inner		Outer Shell		Total	
	time [msec]	speedup	time [msec]	speedup	time [msec]	speedup
[3.2.1]	—	—	—	—	142.187	—
[3.2.2]	2.642	—	23.731	—	26.373	5.39
[3.2.3]	2.705	0.98	12.111	1.96	14.816	1.78
[3.2.4]	2.705	1.00	5.270	2.29	7.975	1.86

外殻格子の高速化の方が効果は大きく、全体でも 1.78 倍の高速化を達成した。

C++ テンプレートによる外殻格子のカーネル生成 [3.2.4] により、外殻格子の計算速度を 2.29 倍に向上することができた。内部格子の計算には変化がなく、全体としては 1.86 倍の高速化となった。

以上の高速化手法の適用により、Naïve な実装に対して合計で 17.8 倍の高速化を達成した。特にマザーリーフ法 [3.2.3] および C++ テンプレートによる外殻格子のカーネル生成 [3.2.4] で、計算速度をそれぞれ 1.78 倍および 1.86 倍に高速化した点は、適合細分化格子法に特有の新しい高速化手法である。

3.3.2 Roofline モデルによる評価

3.3.1 項における全ての高速化手法を適用し、実行性能の上限値を予測する Roofline モデルを用いて評価する。単純な Roofline モデル [46] では、計算の所要時間に対しメモリアクセスの時間と浮動小数点演算時間の長いほうを選ぶ。

$$P_{\text{reachable}} = \min(P_{\text{peak}}, B_{\text{peak}} \times I),$$

$$I = \frac{F}{B} \quad (3.8)$$

ここで、 P_{peak} は浮動小数点演算器のピーク性能、 B_{peak} はメモリバンド幅のカタログ性能である。 F は一格子点あたりの浮動小数点数演算の回数、 B は一格子点あたりのメモリアクセス量である。また、 I は演算強度と呼ばれる値で、浮動小数点数演算の回数とメモリアクセス量の比として与えられる。本計算条件においては、 $P_{\text{peak}} = 3953.23$ GFLOPS, $B_{\text{peak}} = 249.6$ GB/sec, $F = 377$ flop, $B = 220$ byte, $I = 1.713$ であり、したがって実行性能の上限値は $P_{\text{reachable}} = 427.6$ GFLOPS と予測される。 $P_{\text{peak}}/B_{\text{peak}} = 15.8$ に対して演算強度が $I = 1.713$ ということから、格子ボルツマン法はメモリアクセスが律速の計算であることが分かる。この場合、以下の改良型 Roofline モデル [47] を用いると上限値の予測精度を少し改善することができる。

$$P_{\text{reachable}} = \frac{F}{F/P_{\text{peak}} + B/B_{\text{peak}}}$$

$$= \frac{I}{I + P_{\text{peak}}/B_{\text{peak}}} P_{\text{peak}} \quad (3.9)$$

実行性能の上限値は $P_{\text{reachable}} = 385.8$ GFLOPS と予測される。

Table 3.5: Results of performance.

	Performance [GFLOPS]	Ratio of resulting and reachable performance [%]
Reachable	385.8	—
Inner	265.7	68.9
Outer Shell	28.14	7.29
Total	108.7	28.2

改良型 Roofline モデルによる実行性能の上限値、ならびに、内部格子、外殻格子および全格子合計の実行性能を Table 3.5 に示す。実行性能 [GFLOPS] に加え、上限値に対する実行性能の比率を示している。内部格子の実行性能は上限値の 68.9% であり、 33^3 という小さいマザーリーフの格子を GPU で計算するという厳しい条件の下では、十分に最適化された性能であるといえる。等間隔格子の計算では実装の最適化を行っていないため直接の比較はできないが、等間隔格子で上限値に等しい実行性能が達成されると仮定して比較を行う。このとき、合計の実行性能から、細分化格子の格子点数が等間隔格子に対して 28.2% よりも少なければ、等間隔格子よりも実行時間が短くなることがいえる。すなわち、格子点数を 4 分の 1 以下に削減することで、メモリ使用量だけでなく Time-to-Solution の計算速度においても適合細分化格子法が優位となる。

3.4 本章のまとめ

格子ボルツマン法に適合細分化格子法を導入した単一 GPU 計算の実装と高速化手法について述べた。適合細分化格子のリーフに割り当てられる格子点数は等間隔直交格子に比べて少ないため、GPU で効率的な並列計算を行うことが難しい。特に、リーフの境界部分にある外殻格子は隣接参照のために他のリーフの格子を参照する必要があり、アクセスパターンが悪く高い実行性能が出ない。外殻格子点数を削減するため、Octree の 1 段上のノードに格子を割り当てる「マザーリーフ法」を提案し、外殻格子計算で 1.96 倍、全体で 1.78 倍の高速化を達成した。さらに、外殻格子計算では隣接格子へのアクセスパターンを適切に分類して実装する必要があるが、D3Q27 格子ボルツマン法による 3 次元計算の場合、リーフの外殻格子のアクセスパターンは 702 通りになり、すべてのパターンを手動で列挙して実装することはプログラムの生産性や保守性の観点から現実的でない。これまで格子ボルツマン法や適合細分化格子法には適用されていない「C++ テンプレートによる GPU カーネル関数生成」の方法により、実行性能の低い外殻格子計算を 2.29 倍に高速化し、全体でも 1.86 倍の高速化を達成した。1 種類のみのテンプレート関数を記述し、テンプレート引数を変化させた複数の関数呼び出しによりコンパイル時にアクセスパターンの判定をすることで、プログラムの生産性を維持しながら計算を高速化することができた。

第 4 章

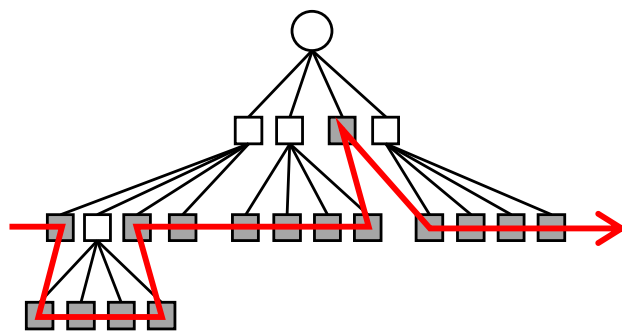
複数 GPU 計算の実装

本章では、適合細分化格子法を導入した格子ボルツマン法の複数 GPU 計算への実装について述べる。MPI (message passing interface) を用い、1 GPU に 1 MPI プロセスを割り当てる Flat MPI で実装する。GPU カーネルの実装は、基本的に前章「単一 GPU 計算の実装と高速化」を踏襲する。ただし、第 3.2.3 項「マザーリーフ法」については、MPI プロセス間通信において性能を得ることができなかったため、実装しないこととした。

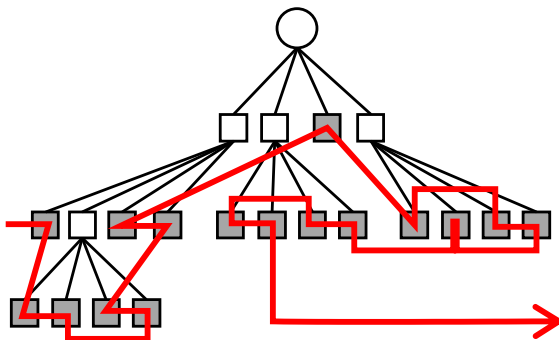
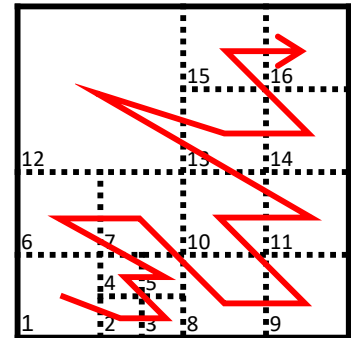
4.1 空間充填曲線に基づく領域分割法

複数 GPU 計算を行うためには各 GPU に格子点を振り分ける必要がある。等間隔格子の場合は領域の大きさに応じて等分割するだけよいが、適合細分化格子法では座標によって解像度が異なるため、単純な分割では各 GPU の担当する格子点数に偏りが生じる問題がある。そこで、適合細分化格子法において格子点数が等しくなるように分割するためのアルゴリズムが必要になる。なお、適合細分化格子法を導入した格子ボルツマン法はマルチタイムステップの実装であり (cf: 第 2.2.2 項)、解像度によって格子点数あたりの計算負荷が異なるため、格子点数のみを等しく分割した場合に一部の GPU に細かい格子が集中し、計算負荷が適切に分散されないおそれがある。しかし、GPU ではメモリ容量の上限が CPU に比べて小さく、計算規模を律速しやすいため、多少の計算負荷バランスの悪化を許容してでもメモリ使用量を均等に分散することが必要である。

適合細分化格子法における領域分割法として比較的簡便かつ効率的な手法として、空間充填曲線を用いる方法 [48] がある。空間充填曲線とは、再帰的に分割された空間について、すべての小領域を必ず 1 回だけ通るような曲線である。四分木または八分木で細分化された任意の領域を一定のアルゴリズムで辿ることができるため、領域を 1 次元配列にマッピングする手法としてよく用いられる。代表的な空間充填曲線として Morton 曲線 [49] および Hilbert 曲線 [50] があり、これらは Fig. 4.1 のように表される。Morton 曲線は Z 字のような順序でリーフを辿るため、Z-curve と呼ばれる。空間の再帰的な分割を表す木構造を深さ優先探索 (Depth first search; DFS) で辿り、その順序でリーフを並べることで、容易に得ることができる。実装が容易であるが、細分化レベルが深くなると、Morton 曲線の順序では隣り合っているリーフが空間的には遠い位置に居るような所謂「ジャンプ」が発生する (たとえば、



(a) Morton curve.



(b) Hilbert curve.

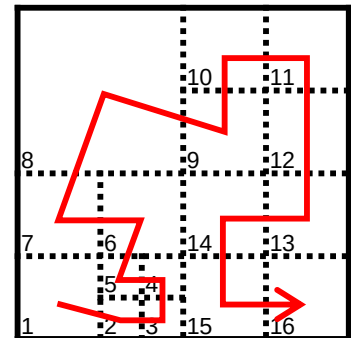


Fig. 4.1: Example of space-filling curve on single octree.

Fig. 4.1 (a) で 11 番目と 12 番目のリーフ). ジャンプの影響により, Morton 曲線に基づく領域分割法は, 分割された各領域の空間的局所性が悪化するおそれがある. 一方 Hilbert 曲線は, コの字型の順序でリーフを辿るもので, レベルごとにコの子を適切に回転させて, 前後の順序にあるリーフが空間中でも必ず隣り合うように設計されている. 領域分割の空間局所性が良くなるが, 深さ優先でリーフを探索する際, 兄弟リーフを辿る順序が一定でないため, 実装が少し煩雑になる.

Morton 曲線も Hilbert 曲線も単一の Octree しか辿ることができないため, 直方体の計算領域を取る Forest-of-octrees 構造においては, 別途 Octree の順序を決める必要がある. 最も簡便な方法は $(0,0,0), (1,0,0), \dots, (NX-1,0,0), (0,1,0), \dots, (NX-1,NY-1,0), (0,0,1), \dots, (NX-1,NY-1,NZ-1)$ のように, 等間隔格子の ijk オーダーに類似した順序で Octree を並べることである. Octree 間を ijk オーダーで辿り, Octree 内のリーフを Morton 曲線で辿る方法は, アルゴリズムが単純で実装が簡単であるため既往研究でもよく用いられている [17, 40, 51, 52]. しかし, この方法で領域分割を行った場合, 分割された領域が空間的に分離される『ジャンプ』が発生する問題がある. ジャンプが発生する領域分割の例を Fig. 4.2 に示す. 図は 2 次元の例で, ijk オーダーと Morton 曲線の組み合わせによりリーフに

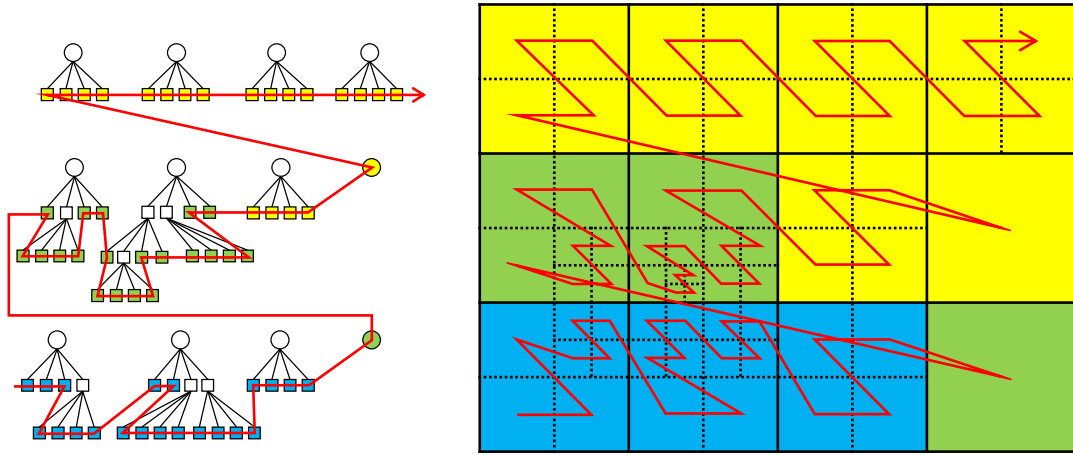


Fig. 4.2: Domain partitioning for forest-of-octrees by the space-filling curve using ijk-order and Morton curve.

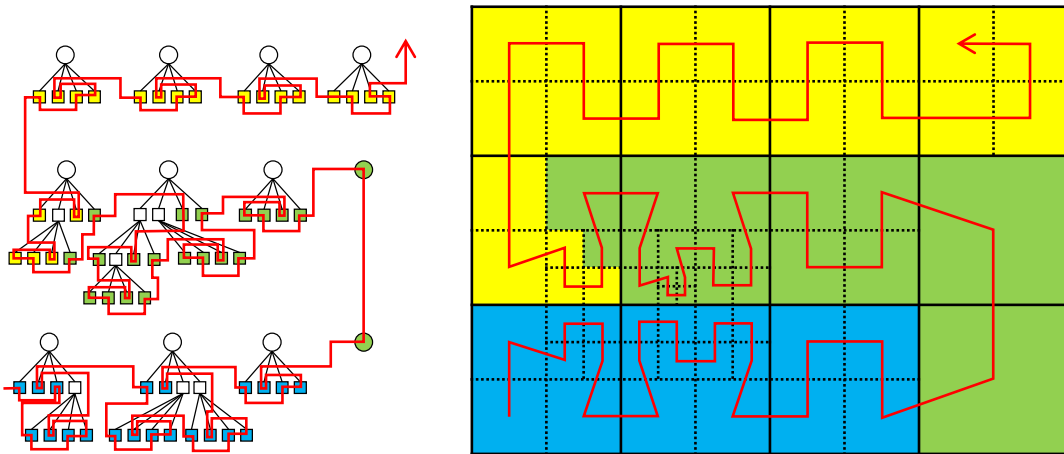


Fig. 4.3: Domain partitioning for forest-of-octrees by the space-filling curve using accordion curve and Hilbert curve (to be utilized in this study).

順序をつけ、3つの領域（青、緑、黄）に分割している．図の例では、緑色の領域について曲線の『ジャンプ』が発生し、2箇所の空間に分離していることがわかる．このようなジャンプが発生しないようにするため、Forest-of-octreesにおいても Hilbert 曲線のようにすべてのリーフを空間的に隣接する順序でマッピングするアルゴリズムが必要である．本研究では、Hilbert 曲線に加えてアコーディオン曲線という独自の曲線を組み合わせることにより、Fig. 4.3 のような空間局所性が改善された空間充填曲線を導入した．本節では、Hilbert 曲線とアコーディオン曲線に基づく直方体形状の空間充填曲線の生成アルゴリズムを説明する．

Table 4.1: Terminal table $T_{\gamma i}^{\text{trm}}$ for generating Hilbert curve.

$\gamma \backslash i$	0	1	2	3	4	5	6	7
A	000	001	011	010	110	111	101	100
B	110	111	101	100	000	001	011	010
C	101	111	101	100	000	010	011	010
D	011	001	000	010	110	100	101	111
E	000	001	101	100	110	111	011	010
F	110	010	011	111	101	001	000	100
G	101	001	000	100	110	010	011	111
H	011	010	110	111	101	100	000	001
I	000	100	110	010	011	111	101	001
J	110	199	000	010	011	001	101	111
K	101	111	011	000	010	110	110	100
L	011	111	101	001	000	100	110	010

4.1.1 Hilbert 曲線に基づく単一 Octree 内のリーフの走査

Forest-of-octrees では、それぞれの Octree のリーフに対しては Hilbert 曲線に基づく順序を与える。このため、はじめに Hilbert 曲線の生成アルゴリズムを述べる。

Hilbert 曲線の生成は、深さ優先探索にルックアップテーブルを導入したアルゴリズム [53, 54] によって行われる。ルックアップテーブルには、ターミナルテーブル $T_{\gamma i}^{\text{trm}}$ インダクションテーブル $T_{\gamma i}^{\text{ind}}$ がある。ここで、 γ は Octree の各節点に付ける属性で、 $\gamma \in \{A, B, C, D, E, F, G, H, I, J, K, L\}$ 、 i は γ に基づいて 8 つの子の順序を辿る順序で、 $i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$ と定義される。

ルックアップテーブルには任意性があるが、本研究では Zhang ら [55] の定義を用い、ターミナルテーブルを Table 4.1、インダクションテーブルを Table 4.2 に示す通りに定義する。ターミナルテーブル $T_{\gamma i}^{\text{trm}}$ は属性が γ である親ノードにおいて、 i 番目に辿る子ノードの位置を規定する 3 ビットの 2 進数で、第 1 ビットから順に x, y, z 軸上の位置を表している。3 ビットの数と子ノードの位置の対応は Fig. 4.4 に示す通りである。また、ターミナルテーブルにより規定される順序は、それぞれの属性 $\gamma \in A, B, \dots, L$ に対して、Fig. 4.5 のようになる。つぎに、インダクションテーブル $T_{\gamma i}^{\text{ind}}$ は属性が γ である親ノードにおいて、 i 番目の子ノードに設定する属性を表している。これらのルックアップテーブルに基づく Hilbert 曲線の生成アルゴリズムの擬似コードを Algorithm 1 に示す。たとえば属性 $\gamma = A$ の親ノードは、子ノードを左下手前 $\rightarrow$ 左下奥 $\rightarrow$ 左上奥 $\rightarrow$ 左上手前 $\rightarrow$ 右上手前 $\rightarrow$ 右上奥 $\rightarrow$ 右下奥 $\rightarrow$ 右下手前の順序で辿り、それぞれの子ノードに $\gamma = I, E, E, D, D, F, F, K$ の属性を与える。それぞれの子ノードは、自身の属性 γ とターミナルテーブルに従って孫ノードを辿り、それぞれの孫ノードに対してインダクションテーブルに基づく属性を与える。このような処理をすべてのリーフの探索が終了する

Table 4.2: Induction table $T_{\gamma i}^{\text{ind}}$ for generating Hilbert curve.

$\gamma \backslash i$	0	1	2	3	4	5	6	7
A	I	E	E	D	D	F	F	K
B	J	F	F	C	C	E	E	L
C	G	K	K	B	B	I	I	H
D	H	L	L	A	A	J	J	G
E	I	A	A	G	G	B	B	L
F	B	J	J	H	H	K	K	A
G	C	K	K	E	E	J	J	D
H	L	D	D	F	F	C	C	I
I	A	E	E	J	J	H	H	C
J	F	B	B	I	I	D	D	G
K	G	C	C	L	L	A	A	F
L	D	H	H	K	K	E	E	B

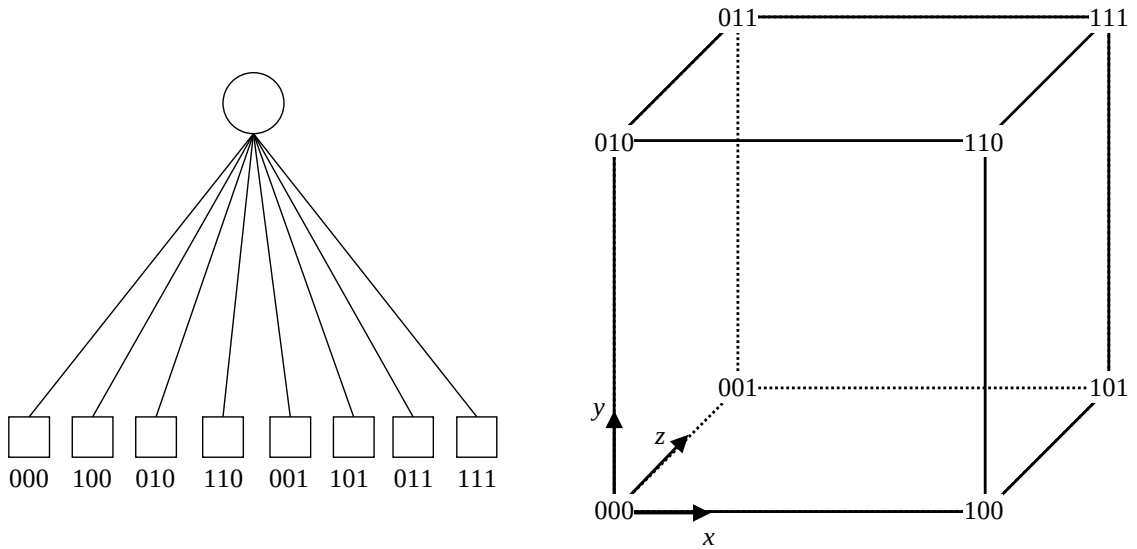


Fig. 4.4: Binary indicator of 8 children of a node. (Left) octree expression of a node and its children. (Right) coordinates and binaries of the children.

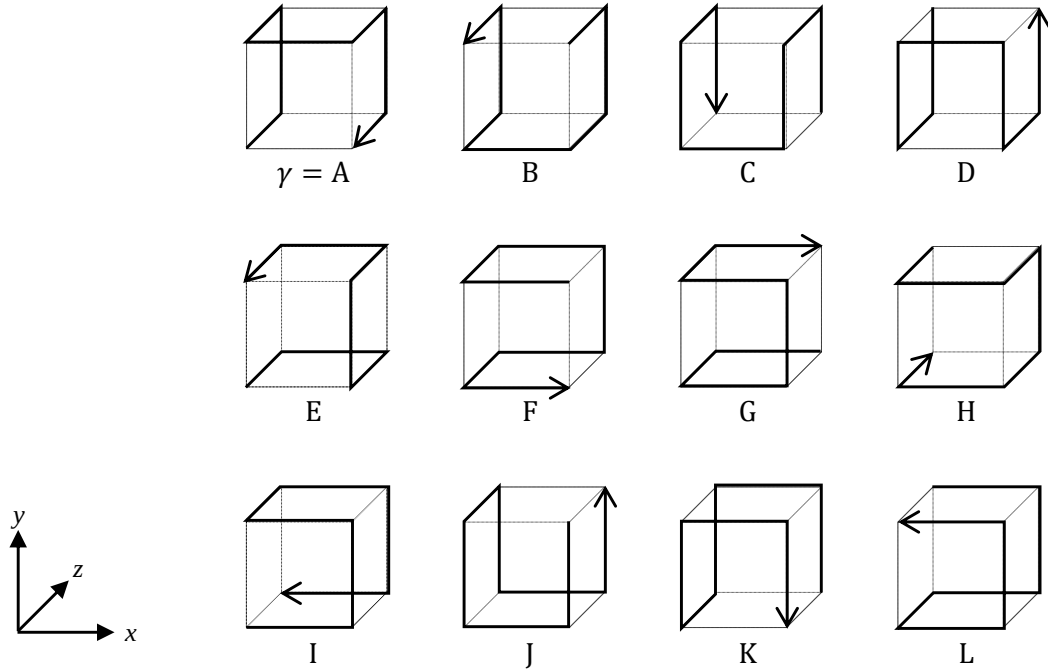


Fig. 4.5: Tracing order of 8 children regulated by terminal table T_{γ}^{trm} shown in Table 4.1.

まで再帰的に繰り返すことで、Octree に対する Hilbert 曲線が生成される。

4.1.2 アコーディオン曲線による Forest-of-octree 内の Octree の走査

複数の Octree を含む Forest-of-octrees 構造では、各 Octree をどのような順序で辿るかを決定するアルゴリズムが必要である。本研究では、アコーディオン曲線に基づく Octree の順序付けアルゴリズムを導入する。アコーディオン曲線は前後の Octree が必ず隣接する順序となるように設計された曲線である。Fig. 4.6 に示す例のように、(a) の ijk オーダーでは列を移動するたびに離れた Octree を走査する「ジャンプ」が発生しているが、(b) のアコーディオン曲線は常に隣り合った Octree を辿っている。アコーディオン曲線は、Algorithm 2 に示す擬似コードのようなアルゴリズムで生成される。Forest 内の各 Octree の 3 次元インデックスを等間隔格子のように (i, j, k) と表すとし、はじめに $(0, 0, 0)$, $(1, 0, 0)$, $(2, 0, 0)$, ..., $(N_x - 1, 0, 0)$ と、 $j = 0, k = 0$ の列について x 軸方向に Octree を走査する。つぎに $j = 1$ の列の走査を行うが、このとき ijk オーダーのように $(0, 1, 0)$ に向かうのではなく、 $(N_x - 1, 0, 0)$ に隣接している Octree を辿るため、 $(N_x - 1, 1, 0)$ を選択する。その後、 x 軸の負の方向に $(N_x - 2, 1, 0)$, $(N_x - 3, 1, 0)$,

Algorithm 1: Generating 1D order of leaves in an octree using Hilbert curve.

```

let  $T$ : octree data structure
let  $M$ : empty array
Function  $H(e$ : a node of octree,  $\gamma$ : property id,  $M$ ):
  if  $e$  is leaf
    append  $(e, \gamma)$  into  $M$ 
  else
    for  $i = 0, 1, 2, 3, 4, 5, 6, 7$ 
      let  $e_i^c$ :  $T_{\gamma^i}^{\text{trm}}$ -th child of  $e$ 
      recursive call  $H(e_i^c, T_{\gamma^i}^{\text{ind}}, M)$ 
    end for
  end if
end Function
let  $n^r$ : root node of  $T$ 
let  $\forall \gamma^r \in \{A, B, C, D, E, F, G, H, I, J, K, L\}$ 
call  $H(n^r, \gamma^r, \text{ref } M)$ 
output  $M$ 

```

..., (1,1,0), (0,1,0) の順に Octree を走査し, $j = 1$ の列の走査を完了する. $j = 2$ の列では (0,2,0) から x 軸の正方向に, $j = 3$ の列では $(N_x - 1, 3, 0)$ から x 軸の負方向に, というように, 1 列ごとに x 軸の走査の方向を反転させながら $j = N_y - 1$ の列まで辿ることで, $k = 0$ の層の走査を完了する. 次に, $k = 1$ の層では $j = N_y - 1$ の列から開始し, 上記と同様に x 軸の走査方向を列ごとに反転しながら $j = 0$ の列まで走査する. 以後同様に, $k = 2$ の層では $j = 0$ から y 軸の正方向に, $k = 3$ の層では $j = N_y - 1$ の列から y 軸の負方向に, というように 1 層ごとに y 軸の走査の方向を反転させながら $k = N_z - 1$ まで辿ることで, すべての Octree の走査が完了する.

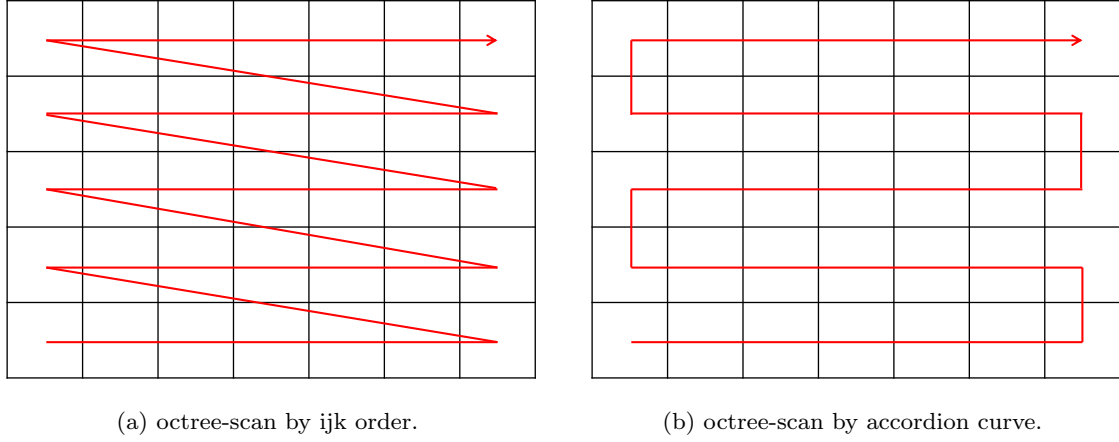


Fig. 4.6: Sample of octree-scan in forest-of-octrees (to be utilized in this study).

Algorithm 2: Generating 1D order of octrees in a forest using accordion curve.

```

let  $F$ : forest-of-octrees data structure
let  $N_x, N_y, N_z$ : number of octrees in  $F$ 
let  $T_{ijk}$ : the  $(i, j, k)$ -th octree in  $F$ 
let  $L$ : empty array
let  $d_i \leftarrow -1$ 
let  $d_j \leftarrow -1$ 
for  $k = 0, 1, \dots, N_z - 1$ 
     $d_j \leftarrow -d_j$ 
    let  $j_0 \leftarrow (N_y - 1)(1 - d_j)/2$ 
    for  $j_v = 0, 1, \dots, N_y - 1$ 
        let  $j \leftarrow j_0 + d_j j_v$ 
         $d_i \leftarrow -d_i$ 
        let  $i_0 \leftarrow (N_x - 1)(1 - d_i)/2$ 
        for  $i_v = 0, 1, \dots, N_x - 1$ 
            let  $i \leftarrow i_0 + d_i i_v$ 
            append  $T_{ijk}$  into  $L$ 
        end for
    end for
end for
output  $L$ 

```

4.1.3 アコーディオン曲線と Hilbert 曲線の組み合わせによる Forest-of-octrees のリーフの走査

上で述べたアコーディオン曲線と Hilbert 曲線を組み合わせることで、直方体形状の計算領域を持つ Forest-of-octree 構造に対してリーフの走査を行う。走査アルゴリズムを Algorithm 3 に示す。ここで重要となるのは、各 Octree の根ノードの属性 γ である。単一の Octree であれば $\gamma \in \{A, B, \dots, L\}$ のどの属性を選んでも支障ないが、Forest-of-octrees 構造では、前後の Octree において、前の Octree の最後のリーフと後の Octree の最初のリーフが隣り合うように適切な属性を選択する必要がある。Fig. 4.7 に示す例のように、(a) のすべての Octree で属性を統一して Hilbert 曲線を生成した場合には一部の Octree を跨ぐときにリーフの「ジャンプ」が発生しており、(b) のようにそれぞれの Octree で根ノードに適切な属性を設定して Hilbert 曲線を生成する必要があることがわかる。

各 Octree の根ノードの属性 γ を決定するため、当アルゴリズムでは「出入り整合条件 (In-out matching; Algorithm 3 における関数 $B(T_{-1}, T_0, T_1)$)」を設定した。整合条件に従えば、 $(0,0,0)$ の Octree の根ノードの属性 γ_{000} は、前の Octree が存在しないため $T_{\gamma_{000},0}^{\text{trm}} = ***$ (ただし $*$ はワイルドカード)、後ろの Octree は $(1,0,0)$ であり走査は x 軸の正方向であるため、 $T_{\gamma_{000},7}^{\text{trm}} = 1**$ である。 $T_{\gamma_{000},0}^{\text{trm}} = ***$, $T_{\gamma_{000},7}^{\text{trm}} = 1**$ を満たす属性は、Table 4.1 を参照すると $\gamma_{000} \in \{A, D, F, G, J, K\}$ である。ここでは仮に $\gamma_{000} = A$ を選択したとする。次に、 $(1,0,0)$ の Octree の根ノードの属性 γ_{100} について考える。前の Octree が $(0,0,0)$ であり、また、 $\gamma_{000} = A$, $T_{A,7}^{\text{trm}} = 100$ であるから、整合条件より $T_{\gamma_{100},0}^{\text{trm}} = 000$ である。さらに、後ろの Octree は $(2,0,0)$ であり走査は x 軸の正方向となるため、 $T_{\gamma_{100},7}^{\text{trm}} = 1**$ である。 $T_{\gamma_{100},0}^{\text{trm}} = 000$, $T_{\gamma_{100},7}^{\text{trm}} = 1**$ を満たす属性は、Table 4.1 より $\gamma_{100} = A$ である。以降 $(N_x - 2, 0, 0)$ の Octree までは同様にして $\gamma = A$ が得られる。 $(N_x - 1, 0, 0)$ の Octree では、前の Octree が $(N_x - 2, 0, 0)$ かつ $\gamma_{N_x-1,00} = A$ であるから、 $T_{\gamma_{N_x-1,00},0}^{\text{trm}} = 000$ 、後ろの Octree が $(N_x - 1, 1, 0)$ であり走査は y 軸の正方向となるため $T_{\gamma_{N_x-1,00},7}^{\text{trm}} = *1*$ である。 $T_{\gamma_{N_x-1,00},0}^{\text{trm}} = 000$, $T_{\gamma_{N_x-1,00},7}^{\text{trm}} = *1*$ を満たす属性として、 $\gamma_{N_x-1,00} = E$ が得られる。このように、アコーディオン曲線の順序でそれぞれの Octree の根ノードに与える属性を逐次的に決定していくことで、Forest-of-octrees においてジャンプの発生しない空間充填曲線を生成することができる。以上のアルゴリズムにより生成される空間充填曲線と、それに基づく領域分割の例を Fig. 4.8 に示す。計算領域中に球を配置し、その周辺に細かい格子を割り当てたもので、(a) は空間充填曲線、(b) は 3 つの領域に分割したときの各領域を表している。図のように、領域のジャンプが無く空間局所性の高い領域分割が実現できている。

Algorithm 3: Generating the 1D order of leaves in a forest using accordion curve and Hilbert curve.

Function $B(T_{-1}, T_0, T_1$: previous, present, next octree):

let $(i_{-1}, j_{-1}, k_{-1}), (i_0, j_0, k_0), (i_1, j_1, k_1)$: indices of T_{-1}, T_0, T_1 in the forest

let γ_{-1} : property id of the root node of T_{-1}

let $b_x^{\text{in}} \leftarrow \begin{cases} 0 & (i_{-1} < i_0) \\ 1 & (i_{-1} > i_0) \\ \text{the first bit of } T_{\gamma_{-1},7}^{\text{trm}} & (i_{-1} = i_0) \\ * & (\text{elsewhere}) \end{cases},$

$b_y^{\text{in}} \leftarrow \begin{cases} 0 & (j_{-1} < j_0) \\ 1 & (j_{-1} > j_0) \\ \text{the second bit of } T_{\gamma_{-1},7}^{\text{trm}} & (j_{-1} = j_0) \\ * & (\text{elsewhere}) \end{cases},$

$b_z^{\text{in}} \leftarrow \begin{cases} 0 & (k_{-1} < k_0) \\ 1 & (k_{-1} > k_0) \\ \text{the thrid bit of } T_{\gamma_{-1},7}^{\text{trm}} & (k_{-1} = k_0) \\ * & (\text{elsewhere}) \end{cases}$

let $b_x^{\text{out}} \leftarrow \begin{cases} 0 & (i_1 < i_0) \\ 1 & (i_1 > i_0) \\ * & (\text{elsewhere}) \end{cases}, \quad b_y^{\text{out}} \leftarrow \begin{cases} 0 & (j_1 < j_0) \\ 1 & (j_1 > j_0) \\ * & (\text{elsewhere}) \end{cases}, \quad b_z^{\text{out}} \leftarrow \begin{cases} 0 & (k_1 < k_0) \\ 1 & (k_1 > k_0) \\ * & (\text{elsewhere}) \end{cases}$

select γ_0 **where** $T_{\gamma_0,0}^{\text{trm}} = (b_x^{\text{in}}, b_y^{\text{in}}, b_z^{\text{in}}) \wedge T_{\gamma_0,7}^{\text{trm}} = (b_x^{\text{out}}, b_y^{\text{out}}, b_z^{\text{out}})\}$

return γ_0

end Function

let F : forest-of-octrees data strucuture

let N : total number of octrees in the forest (also defined as $N = N_x N_y N_z$)

let T_n : n -th octree of F in the accordion order L generated by Algorithm 2
 (valid in range of $n = 0, 1, \dots, N - 1$)

let $T_{-1} \leftarrow \text{NULL}$

let $T_N \leftarrow \text{NULL}$

let M : empty array

for $n = 0, 1, 2, \dots, N - 1$

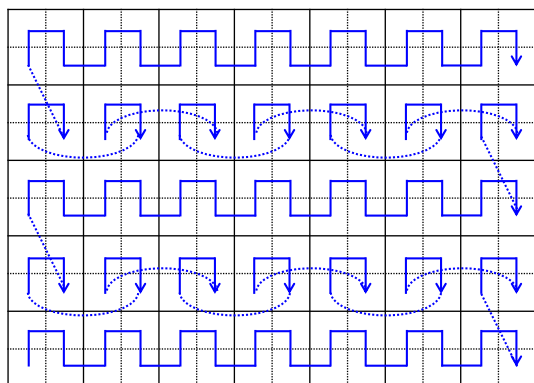
let $\gamma_n \leftarrow B(T_{n-1}, T_n, T_{n+1})$

let r_n : root node of T_n

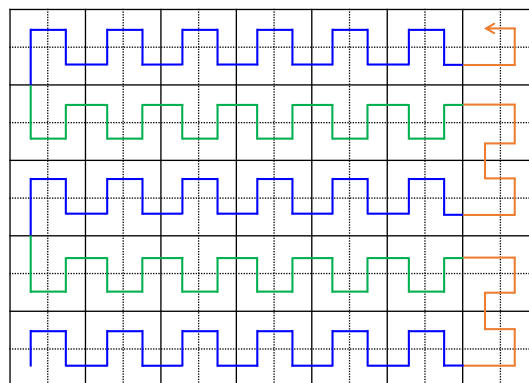
call $H(r_n, \gamma_n, \text{ref } M)$

end for

output M

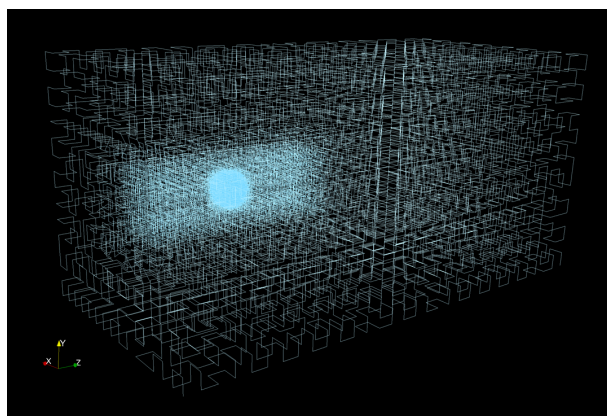


(a) without in-out property matching.

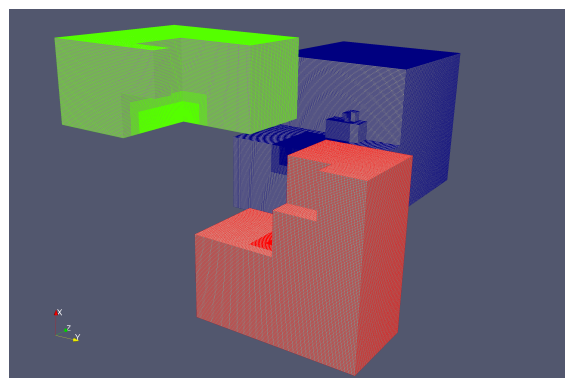


(b) with in-out property matching (to be utilized in this study).

Fig. 4.7: Example of the rectangular space-filling curve without/with in-out property matching.



(a) space-filling curve for the domain.



(b) three partitioned domains by the space-filling curve.

Fig. 4.8: Example of the rectangular space-filling curve by the accordion curve and Hilbert curve.

4.2 GPU 間通信の実装と最適化

4.2.1 袖領域のメモリ確保

複数 GPU 計算においては、他の GPU 上に割り当てられた格子に隣接参照するため、袖領域と呼ばれるデータ転送専用の格子点を確保する必要がある。隣接するリーフのうち袖領域として参照に必要な格子は端の数格子だけであるが、時間発展計算の GPU カーネルにおいて隣接点参照を簡便にするため、リーフ全体の格子点をメモリに取るような実装を採用する。複数 GPU 計算における計算格子と袖領域の例を Fig. 4.9 に示す。2 次元格子を 2 台の GPU に領域分割した例であり、GPU 1 に対応する格子を青色、GPU 2 に対応する格子を赤色で示している。図左側は計算格子の配置を表す。図右側は各 GPU がメモリ確保する格子点を表しており、薄い色の格子が袖領域である。このようにリーフ全体を袖領域に持つことで、隣接点参照で計算格子と袖領域格子の区別をする必要がなく、実装が簡便になる。袖領域リーフ全体の格子を冗長に確保するためメモリ使用量が若干増大するが、予備的な計算において袖領域のメモリ使用量は計算格子の 2 割程度であったため、メモリ使用量の増大の影響は小さいと考えられる。

4.2.2 GPU 間通信の最適化

(1) 通信する格子の最適化

前節の通り本研究では隣接するリーフの全体の格子を袖領域に取っているが、隣接点参照として必要な格子は一部であるため、袖領域リーフ全体を通信するのは無駄なデータ転送が多く効率が悪い。そこ

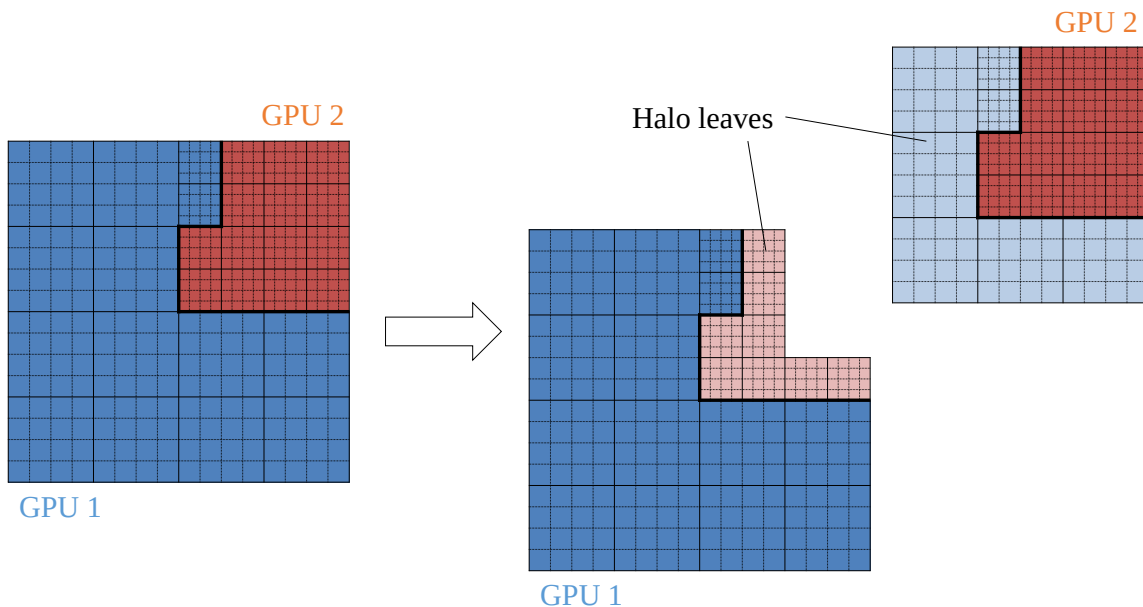


Fig. 4.9: Setup of halo leaves on multiple GPUs computation.

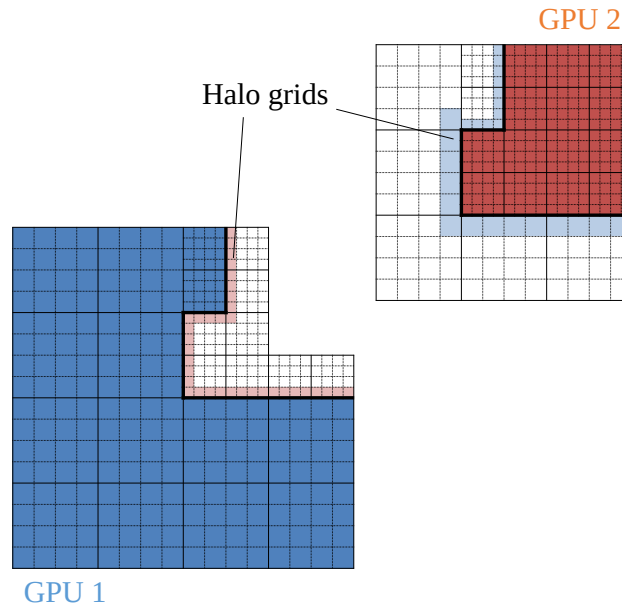


Fig. 4.10: Optimized halo grids in the halo leaves.

で、隣接点参照に必要な格子だけを取り出して通信する最適化を実施する．これにより、通信される格子は Fig. 4.10 に示す例のようになる．図において薄い色付きの格子は隣接点参照に必要であるため通信し、残りの白色の格子点については通信しない．これによりデータ転送量を大幅に削減されることが期待できる．

(2) マルチタイムステップ時間発展における各時刻の通信の最適化

適合細分化格子法を導入した格子ボルツマン法は、格子解像度ごとに時間発展計算の回数が異なるマルチタイムステップの実装になっている．そこで、各時刻ステップごとに値が更新された格子点だけを通信するように最適化する．例を Fig. 4.11 に示す．図の例では粗い格子と細かい格子の 2 種類の解像度が存在しており、(a) の偶数のタイムステップではすべての格子を通信する一方で、(b) の奇数のタイムステップでは粗い格子の通信を行わず細かい格子のみ通信することでデータ転送量を削減できる．また、より粗い格子は通信するタイムステップが更に少なくなるため、解像度差が大きいような条件ではさらに通信量を削減できる可能性がある．

(3) 格子ボルツマン法の速度分布関数の方向による最適化

格子ボルツマン法は 1 格子点に速度方向成分の数だけ速度分布関数を保持しているが、袖領域中の速度分布関数のうち隣接点参照に必要な成分は一部だけである．そこで、速度分布関数のうち必要な成分だけを取り出して通信する最適化を実施する．最適化の例を Fig. 4.12 に示す．2 次元の例で、格子内の矢印と丸い点は D2Q9 モデルの格子ボルツマン法の速度分布関数を表す．図のように、袖領域である薄い赤または青色の格子のうち、黒色の矢印で示される方向の速度分布関数は隣接点参照に必要な成分で

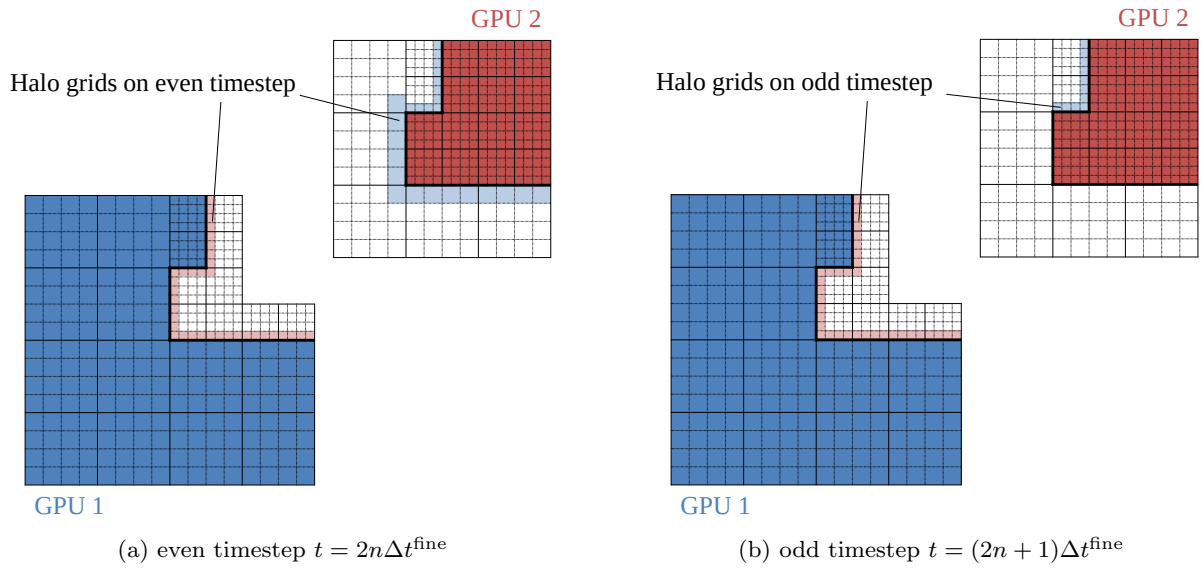


Fig. 4.11: Optimized halo grids with multiple timestep method.

ある．灰色の矢印で示される方向の速度分布関数の通信は不要であるため行わない．これにより，(1)，(2)の最適化から更にデータ転送量を削減することができる．

4.2.3 データ転送量および通信時間の測定

GPU 間通信の最適化の効果を確かめるため，データ転送量および通信時間の測定を実施した．測定に用いる計算格子を Fig. 4.13 に示す．これは，複雑形状物体である自転車の周りに高解像度格子を割り当てたものである．計算領域は $2 \times 1 \times 1$ の Forest-of-octree としており，その大きさは約 $4 \text{ km} \times 2 \text{ km} \times 2 \text{ km}$ の範囲に相当する．計算領域の端面は，6 面すべてで周期境界条件のための袖領域通信を行う．格子の細分化レベルは，自転車の最近傍格子で 16，遠方の最も粗い格子で 4 であり，最大で 4096 倍の解像度差がある．最細格子の解像度は， $1048577 \times 524289 \times 524289$ の等間隔格子と同等であり，その格子幅は 4 mm に相当する．総格子点数は 1.09 億である．なお，本格子配置は第 6 章「自転車競技における集団走行の大規模空力解析」の応用計算でも用いるものであるが，ここでは空力計算の結果には触れず，GPU 間通信の性能についてのみ議論する．

測定には東京工業大学のスパコン TSUBAME3.0 を用いた．TSUBAME3.0 は 1 ノードに 4 GPU (NVIDIA[®] Tesla[®] P100 HBM2; 単精度浮動小数点数演算性能 10.6 TFLOPS, メモリ容量 16 GB, バンド幅 732 GB/s) を搭載しており，ノード内の GPU 間は NVLink (バンド幅 160 GB/s) で接続され，ノード間は 4 本の Intel[®] Omni-Path ネットワーク (バンド幅 100 Gbit/s) で接続される．本研究では，MPI ライブラリとして OpenMPI [56] を用いる．GPU 間通信の実装には GPU Direct RDMA を用いており，通信のバックエンドは NVKink, CPU を経由した PCIe 接続，および Omni-Path ネットワークの中から自動的に選択される．

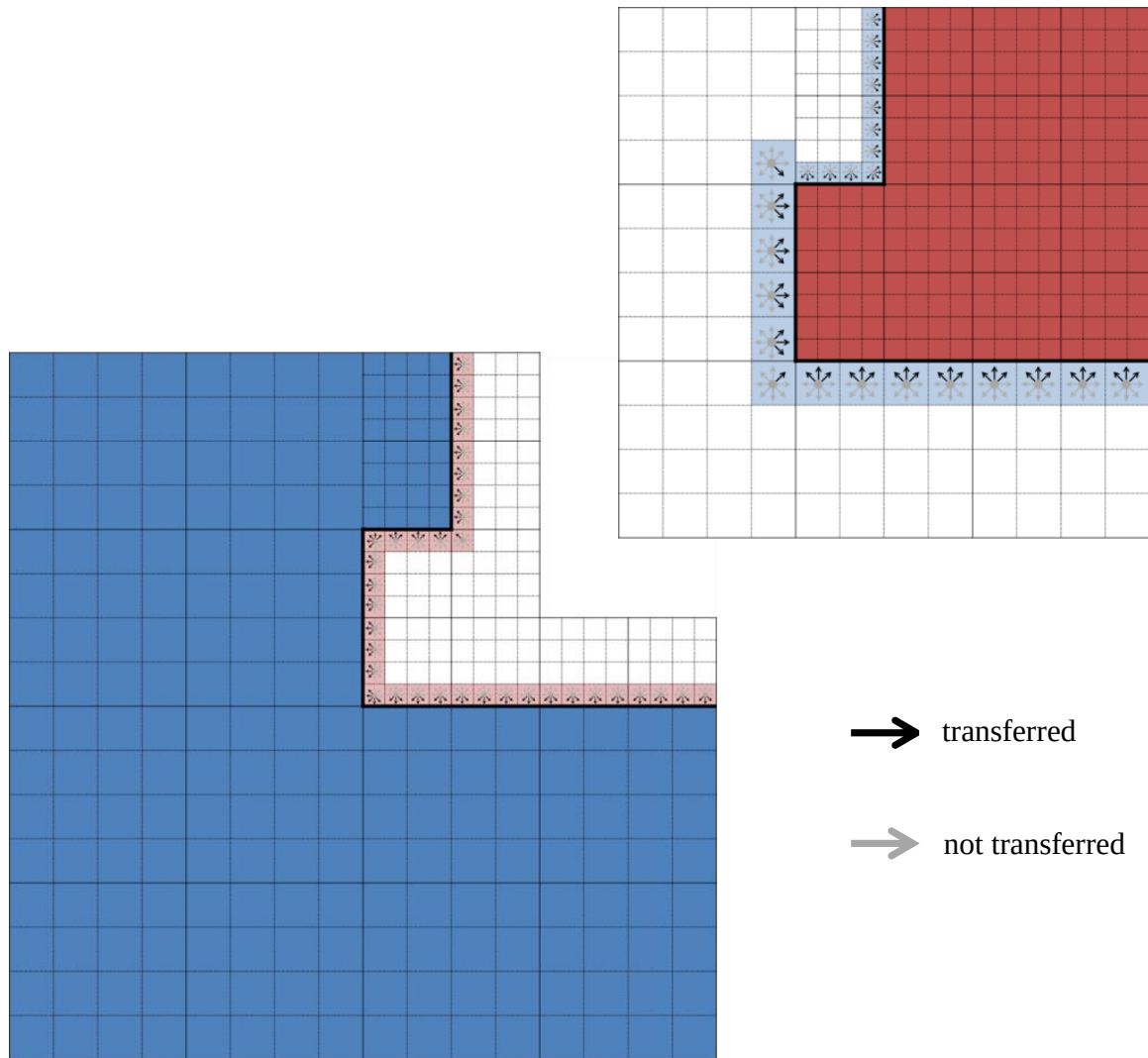


Fig. 4.12: Optimized lattice Boltzmann distribution functions by direction.

2 ノード 8 GPU を用いた並列計算におけるデータ通信の性能測定結果を Fig. 4.14 に示す. (a) はデータ転送量の測定結果, (b) は通信時間の測定結果である. 横軸は最適化の適用の段階であり, 数字は前項「GPU 間通信の最適化」で説明したそれぞれの最適化の番号に対応する. 縦軸は (a) において 1 GPU あたり・1 タイムステップあたりの平均データ転送量, (b) において 1 タイムステップあたりの平均通信時間である. 最適化 (1) 「通信する格子の最適化」により, データ転送量は最適化なしの実装に比べて 16.3% へと大幅に削減され, 通信時間は 56.8% に短縮された. 最適化 (2) 「マルチタイムステップ時間発展における各時刻の通信の最適化」を適用することで, データ転送量は最適化なしの実装に比べて 9.4%, 通信時間は 26.4% にそれぞれ削減された. さらに, 最適化 (3) 「格子ボルツマン法の速度分布関数の方向による最適化」を加えることで, 最終的にデータ転送量は 4.3%, 通信時間は 20.9% まで削減された. データ転送量の削減率に比べて通信時間の短縮の度合いが小さいが, これは, リーフの中から必要

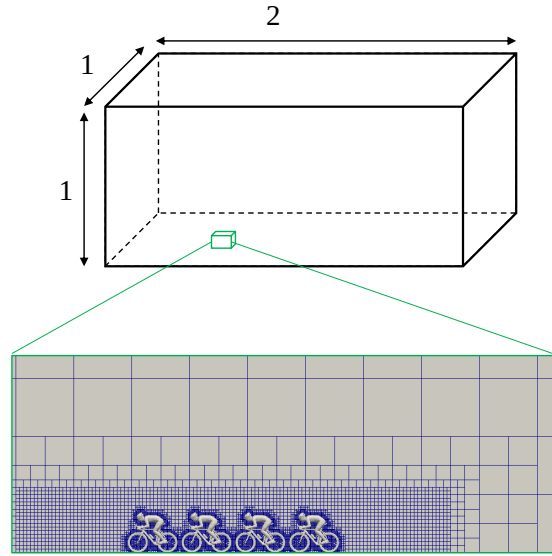


Fig. 4.13: Complex refined mesh for performance measuring.

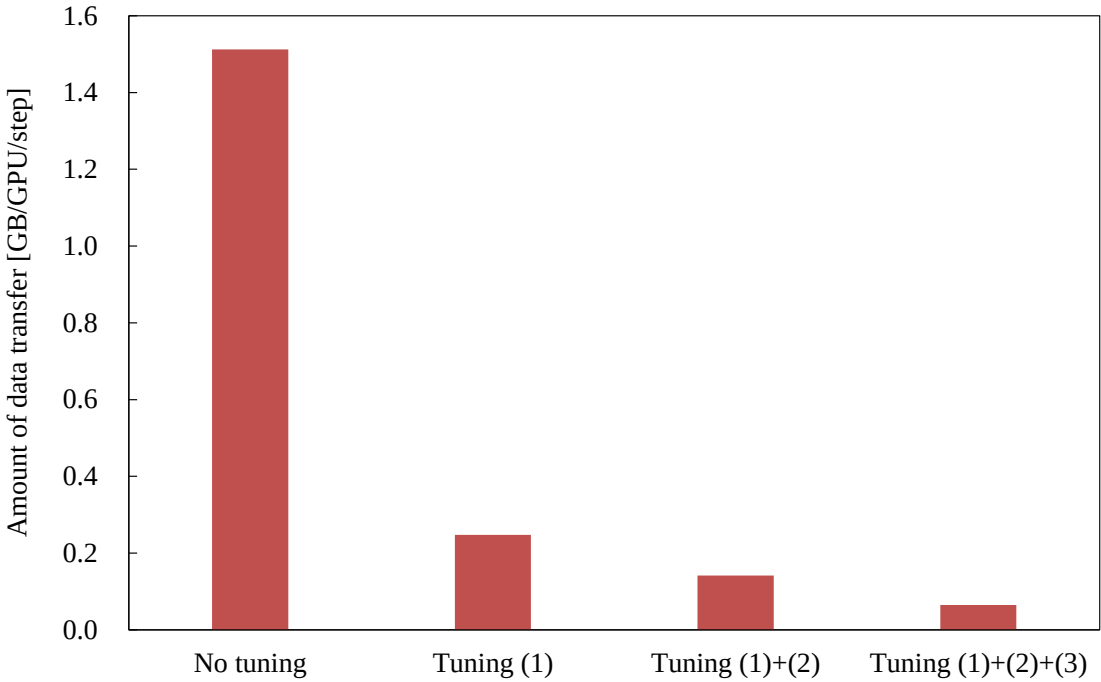
な格子のみを取り出して MPI 通信のバッファに組み込むパッキング，およびその逆であるアンパッキングの処理において，GPU メモリへのアクセスが不規則になり GPU のメモリアクセスのスループットが低下したことが原因として考えられる．そのほか，データ転送量が削減されることで 1 回の通信あたりのデータ量が小さくなったことで，MPI 通信のスループットも低下したと考えられる．しかし，いずれにせよ，総ての最適化を適用することで GPU 間通信に要する時間はおよそ 5 分の 1 にまで短縮されており，GPU 間通信の大幅な高速化が達成されたといえる．また，同条件における時間発展計算に要する GPU カーネルの実行時間は 1 ステップあたり 0.202 sec であった．GPU 間通信の時間は，1 ステップあたり 0.077 sec と時間発展計算の時間に比べても小さく，十分に高速化されていることが確認できた．

4.3 計算性能測定

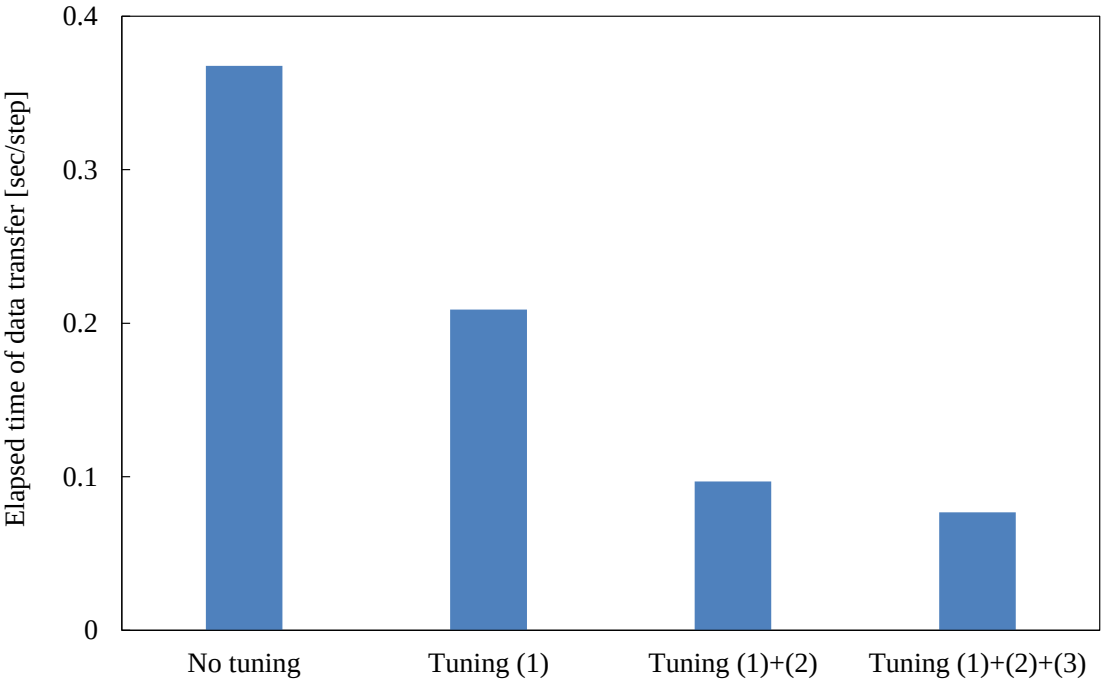
上記の複数 GPU 計算コードの実装において，計算性能の測定を行う．測定する計算性能の指標として，1 秒あたりに時間発展させることのできる格子点の数を表す LUPS (lattice updates per second) を採用する．格子ボルツマン法の計算では LUPS が百万を優に超えるため，Mega の接頭辞をつけて MLUPS の単位で表すのが通例である．適合細分化格子法を導入した格子ボルツマン法では，細分化レベルごとに時間発展計算の頻度が異なることを考慮する必要がある，計算性能 P は次の式で求められる．

$$P = \frac{T \sum_{l=0}^{l_{\max}} 2^{l-l_{\max}} n_l (m-1)^3}{t} \quad [\text{LUPS}] \quad (4.1)$$

ここで， T は最細格子における時間発展の回数，すなわち計算ステップ数を表す．粗い格子の時間発展計算の回数との整合性を取るため，通常 T には 2 の累乗の数を設定する． t は計算ステップ数 T の計算に掛かる時間であり，プログラムの全実行時間から初期設定とファイル出力の時間を取り除いた時間を



(a) amount of data transfer.



(b) elapsed time.

Fig. 4.14: Performance of inter-GPU data transfer with each optimization.

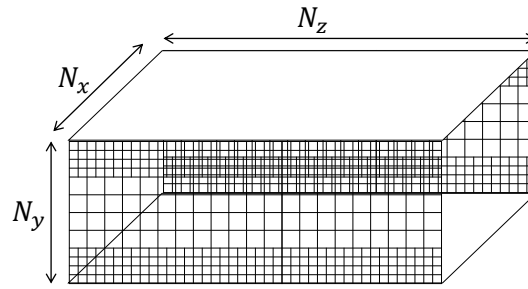


Fig. 4.15: Simple refined mesh for performance measuring.

測定値とする． l は格子の細分化レベルであり， $l_{\max}$ は最細格子のレベル， n_l はレベル l のリーフの数を表す． m はリーフに割り当てる格子の x, y, z 各軸方向に対する分割数であり，本計算では $m = 9$ である．なお，本計算はノードセンターの格子配置であり，隣り合うリーフで端の格子を重複して計算しているため，計算格子の実効的な数は m^3 ではなく $(m - 1)^3$ となる．

計算性能の測定には，前節「GPU 間通信の実装と最適化」の通信性能測定と同様，東京工業大学のパソコン TSUBAME3.0 を用いた．

本性能測定では，単純な配置と複雑な配置の 2 種類の細分化格子を用いる．

4.3.1 単純配置の格子における計算性能

単純配置の格子を Fig. 4.15 に示す．これは，平行平板間流れのように，上下面に細かい格子を割り当てた格子である． x 軸および y 軸方向の計算領域の端は周期境界条件とし，袖領域の MPI 通信を行う．上下面については MPI 通信を行わない．計算領域の大きさは，基準となる 8 GPU 計算において， $N_x = 1, N_y = 2, N_z = 2$ とする．格子の細分化レベルは，壁の最近傍格子で 7，中央の最も粗い格子で 4 であり，最大で 8 倍の解像度差がある．最細格子の解像度は， $N_x = 1, N_y = 2, N_z = 2$ の Forest-of-octrees の場合で $1025 \times 1025 \times 2049$ の等間隔格子と同等である．総格子点数は $N_x = 1, N_y = 2, N_z = 2$ の場合で 2.31 億となり，他の条件では Octree の数 ($N_x \times N_y \times N_z$) に比例する．比較的単純な格子で，複数 GPU 計算における袖領域の通信が少なく高い性能が得られることが予想される．

GPU 数に比例して計算格子数を増大させる弱スケーリング性能 (weak scalability) および，計算格子点数を固定して GPU 数のみを増やす強スケーリング性能 (strong scalability) の測定を行った．8 GPU で $N_x = 1, N_y = 2, N_z = 2$ の条件を起点とし，8, 16, 32, 64, 128, 256 GPU の計算を行った．性能測定結果を Fig. 4.16 に示す．横軸は GPU 数，縦軸は計算性能 [MLUPS] を表し，破線は 8 GPU 計算から予想される理想的なスケーリング性能，赤線は弱スケーリングの測定結果，青線は強スケーリングの測定結果である．また，弱スケーリングと強スケーリング性能について，理想的な性能に対する実行性能の比率 (並列化効率) をパーセンテージで示している．弱スケーリングは，8 GPU から 256 GPU までで 96.4% の並列化効率を達成しており，良好なスケーリングである．一方で強スケーリングの並列化効率は，8 GPU から 256 GPU までで 19.0% と低く，十分な性能が得られていない．これは，強スケー

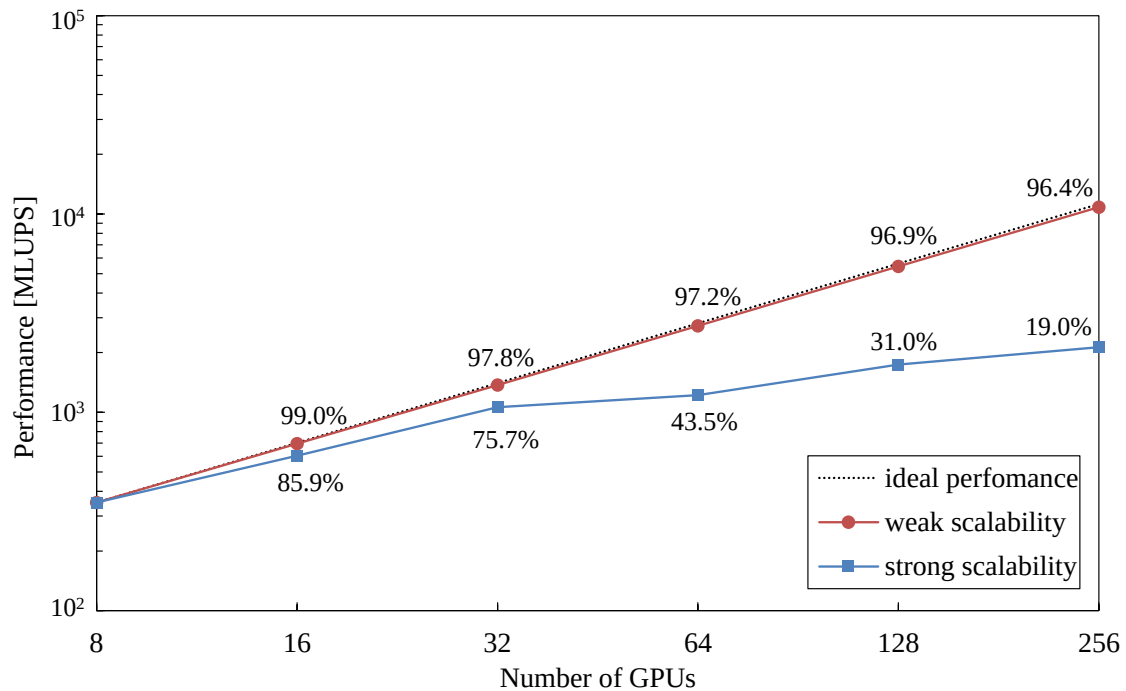


Fig. 4.16: Scalability of the computation on simple refined mesh.

リングにおいて GPU 数が増えると 1 GPU あたりの格子点数が少なくなり、GPU 間通信のデータ量が相対的に大きくなるのが原因と考えられる。

4.3.2 複雑配置の格子における計算性能

複雑配置の格子として、前節「GPU 間通信の実装と最適化」の通信性能測定に用いた Fig. 4.13 のような格子を用いる。これは、複雑形状物体である自転車の周りに高解像度格子を割り当てたもので、格子配置が比較的複雑であるため細分化レベル間の格子の補間や MPI における袖領域の通信が多く、計算性能を得にくいと考えられる。計算領域の大きさは GPU 数によらず固定とし、自転車の台数を変化させることで格子点数を制御する。基準となる 8 GPU 計算において 4 台の自転車を配置する。総格子点数は、4 台の自転車を配置した場合で 1.09 億であり、他の条件では自転車の台数に概ね比例する。

GPU 数に比例して計算領域中の自転車を増やす擬似弱スケーリング (weak-like scalability), および、自転車 4 台の条件で GPU 数のみを増加させる強スケーリング (strong scalability) の性能測定を実施した。8 GPU で自転車 4 台を配置する条件を起点とし、8, 16, 32, 64, 128 GPU の計算を行った。性能測定結果を Fig. 4.17 に示す。横軸は GPU 数、縦軸は計算性能 [MLUPS] を表し、破線は 8 GPU 計算から予想される理想的なスケーリング性能、赤線は擬似弱スケーリング性能、青線は強スケーリング性能の結果である。擬似弱スケーリングは 8 GPU から 128 GPU までで 72.8% の並列化効率である。前項の単純格子の計算における 8 GPU から 128 GPU までの弱スケーリングの並列化効率 96.4% に比べると効率は低いですが、72.8% の並列化効率は応用計算を行ううえで実用上十分な性能であり、多数の自転

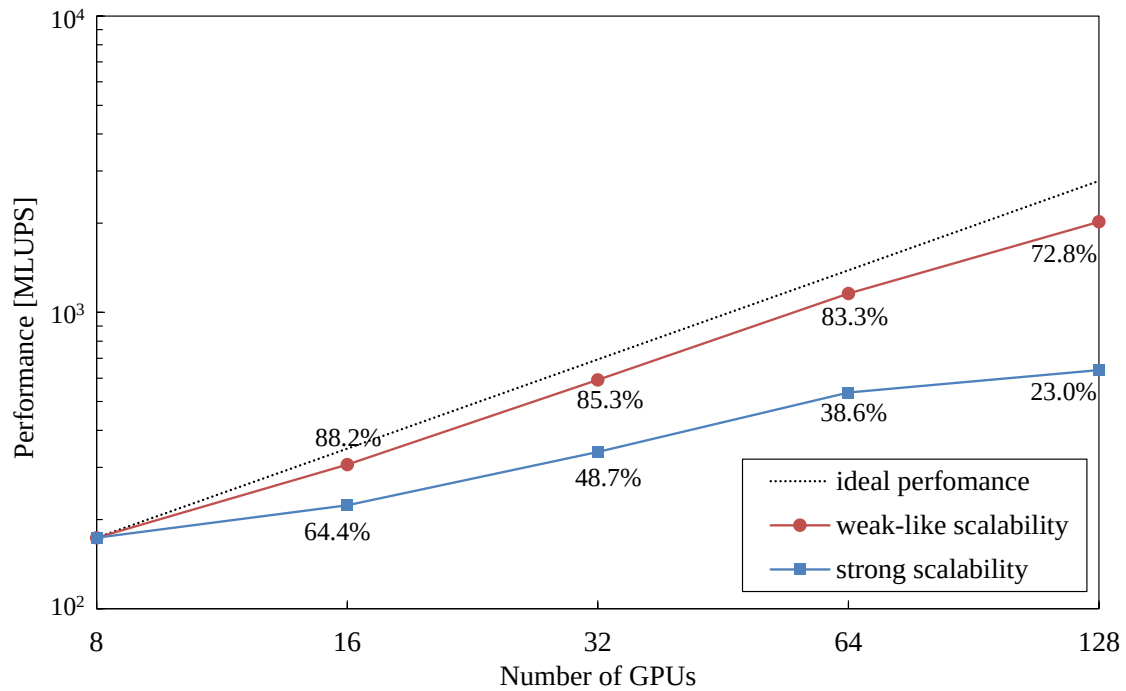


Fig. 4.17: Scalability of the computation on the complex refined mesh.

車を含むような大規模計算が本コードにより実施可能であることが確認できた。一方、強スケーリング性能は、8 GPU から 128 GPU で 23.0% と極めて低い並列化効率であった。これは、単純格子の計算と同様、強スケーリングにおいて GPU 数が増えると 1 GPU あたりの格子点数が少なくなり、MPI 通信のデータ量が相対的に大きくなることが原因であると考えられる。強スケーリングは実用上においても十分とはいえない性能であるため、本コードで応用計算を行う際には GPU メモリの上限近くまで格子点数を取る並列化を行うべきであるといえる。

4.4 本章のまとめ

適合細分化格子法を導入した格子ボルツマン法の複数 GPU 計算の実装およびその計算性能について述べた。直方体形状の計算領域を取る Forest-of-octrees 構造において、アコーディオン曲線と Hilbert 曲線を組み合わせた独自の空間充填曲線を用いることで、空間局所性の高い領域分割法を導入した。また、袖領域通信の最適化を行い、最適化前に比べて MPI 通信のデータ量を 4.3% まで、および通信時間を 20.9% まで、それぞれ大幅に削減した。複数 GPU 計算の弱スケーリングは、単純な格子配置において 90% 以上の並列化効率を達成したほか、自転車のような複雑形状を含む計算条件においても 8 GPU から 128 GPU のスケーリングにおいて効率 72.8% と、実用上十分な性能であった。ただし、1 GPU あたりの格子点数が少なくなる強スケーリングでは十分な性能が得られておらず、本コードは GPU メモリの上限近くまで格子点数を取るような並列化に適していることが示唆された。

第 5 章

球周りの流れの大規模計算によるドラッグ・クライシスの再現計算

本研究で開発した空力解析の複数 GPU 計算コードを用いて、球周りの流れの大規模計算を実施する。

球周りの流れは、第 2.3.4 項でも示したように、非圧縮単層流や空力解析の検証計算としてよく用いられるものであるが、レイノルズ数が 10 万以下の範囲で検証に使われることが多い。これは、下で説明するドラッグ・クライシスと呼ばれる現象を計算で再現することが難しいためである。

球周りの流れにおけるレイノルズ数と抗力係数の関係は、風洞実験から、概ね Fig. 5.1 のようになることが知られている [42]。レイノルズ数が 10^3 から 10^5 の間は抗力係数が 0.4 程度でほぼ一定であるが、レイノルズ数が 20 万程度を超えて上昇すると、抗力係数が突然低下し始め、レイノルズ数 50 万程度で抗力係数は 0.1 を下回るほど小さくなる。ドラッグ・クライシスとは、このような急激に抗力係数が低下する現象のことをいう。なお、ドラッグ・クライシスが始まる臨界レイノルズ数は、球の表面性状や風洞実験の精度の限界から正確な値は知られていないが、球表面が滑面に近い場合は概ね 20 万～30 万であると予想されている。

ドラッグ・クライシスは 20 世紀初頭には既に実験 [57] によって見いだされていた事象であるが、計算においては極めて高い解像度が要求されるため、必然的に大規模計算となり、再現計算を実施することは計算資源の制約から近年まで困難であった。小規模の計算では解像度の不足により計算が破綻するか、または過剰な数値粘性がはたらいてドラッグ・クライシスが再現されないことが想定される。実際、本論文第 2.3.4 項の球周りの流れの検証計算においても、球直径に対して 25.6 格子という粗い格子では LES モデルによる渦粘性が非物理的に過大に評価され、Fig. 5.2 に示す結果のように、ドラッグ・クライシスにおける抗力低下が再現できなかった。LES 計算において初めてドラッグ・クライシスの再現に成功したのは 2012 年の Muto らの非構造格子を用いた有限体積法に基づく計算 [58] であるが、現象の実験的発見から 100 年近い歳月を経ており、計算機の性能向上を待たざるを得なかったことが伺える。

ドラッグ・クライシスの再現計算は、多数の格子点を必要とする高難易度の計算であるため、大規模計算の応用事例に適している。また、極めて高いレイノルズ数の条件であるため計算手法の数値安定性の検証として有用であるほか、レイノルズ数に非常に敏感な現象であるため、過剰な数値粘性が入るような計算スキームではドラッグ・クライシスが得られないことも予想され、計算手法の精度検証としても

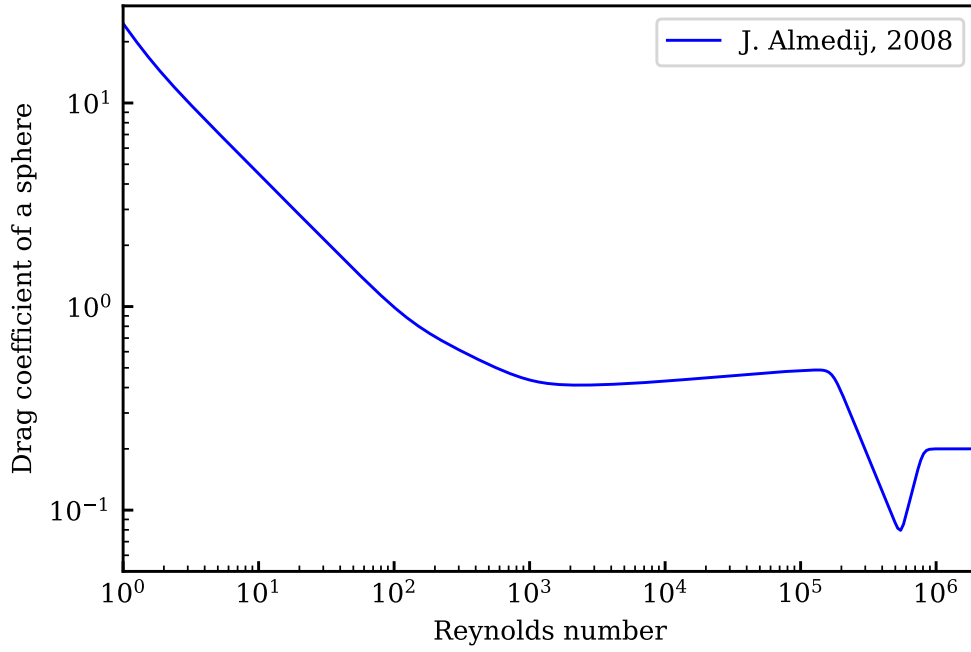


Fig. 5.1: Drag coefficient of a sphere with the drag crisis on $Re \approx 2 \times 10^5$.

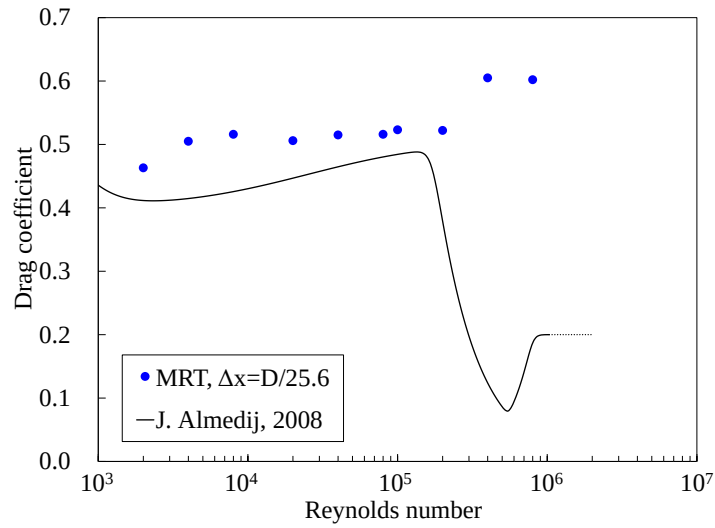


Fig. 5.2: Example of invalid result in the flow around a sphere at drag crisis with the coarse mesh ($\Delta x = D/25.6$).

有益な計算になると考えられる．そこで本章では，本研究で実装した適合細分化格子上の格子ボルツマン法の複数 GPU コードにおいて，MRT モデルおよびキュムラントモデルの 2 種類の衝突モデルで球周りの流れの大規模計算を実施し，各モデルにおける数値安定性やドラッグ・クライシスの再現精度を比較する．

5.1 計算条件

計算領域の設定を Fig. 5.3 に示す．Forest-of-octree により $2 \times 1 \times 1$ の大きさの計算領域を設定し， $(0.5, 0.5, 0.5)$ の位置に直径 $D = 1000/65536$ の球を配置する．流路の球による閉塞比は 0.0182% と十分に小さいため無視できる．計算領域の前端 $x = 0$ の面は速度 $\mathbf{U}_{\text{in}} = (1, 0, 0)$ の流入境界条件，後端 $x = 2$ の面は流出境界条件， $y = 0, 1$ ならびに $z = 0, 1$ の上下左右面は周期境界条件とする．主流の CFL 数は $U\Delta t/\Delta x = 0.03$ である．球表面には Interpolated bounce-back 条件 [36] を適用し，Momentum-exchange method [35] により抗力を求める．格子解像度は，球から 10 メッシュ以内の最近傍格子で $\Delta x = D/1000$ ，球から遠い格子ほど粗く設定しており，最大で $\Delta x = D/2$ の格子幅になる．

Muto ら [58] によれば，ドラッグ・クライシスの再現に必要な格子解像度は球表面の層流境界層厚さ δ_B に依存する．また，境界層厚さの値は，

$$\delta_B = 3\sqrt{\frac{(D/2)\nu}{U_0}} = 3D\sqrt{\frac{1}{2Re}} \quad (5.1)$$

と見積られている [59]．上式に基づけば，たとえばドラッグ・クライシスが起こる直前のレイノルズ数 $Re = 2 \times 10^5$ では，境界層厚さは $\delta_B = 4.743 \times 10^{-3}D$ と見積られる．本計算条件の球表面の格子解像度は $\Delta x = D/1000$ であるから， $\delta_B = 4.734\Delta x$ ，すなわち層流境界層を 4 格子以上で解像していることがわかる．また，抗力係数が極小値を取るレイノルズ数 $Re = 5 \times 10^5$ では，同様に見積もると $\delta_B = 3\Delta x$ であり，層流境界層を 3 格子で解像していることになる．3 格子あれば境界層内に 3 次関数で近似された速度プロファイルが得られるため，ある程度高い精度でドラッグ・クライシスを計算できることが予想される．さらに高いレイノルズ数においては，たとえば $Re = 1 \times 10^6$ では $\delta_B = 2.12\Delta x$ ， $Re = 1 \times 10^7$ では $\delta_B = 0.671\Delta x$ となるため，このような高レイノルズ数の条件下では解像度が不足し，層流境界層を正しく解像できない．正しい解は得られないと考えられるが，数値安定性の検証に役立つため，これらの高レイノルズ数の計算も行う．

格子ボルツマン法の衝突モデルを比較するため，

- 1) すべての格子で MRT モデル．
- 2) 球から 10 メッシュ以内の最近傍格子でキュムラントモデル，他の遠方格子で MRT モデル．
- 3) すべての格子でキュムラントモデル．

の 3 種類の設定で計算をした．いずれの計算も，コヒーレント構造 Smagorinsky モデルに基づく渦粘性を導入した LES 計算である．なお，2) は 2 つの衝突モデルを組み合わせたハイブリッドモデルで，Detached eddy simulation (DES) [60] のように数値安定性と計算精度を両立するような手法が得ら

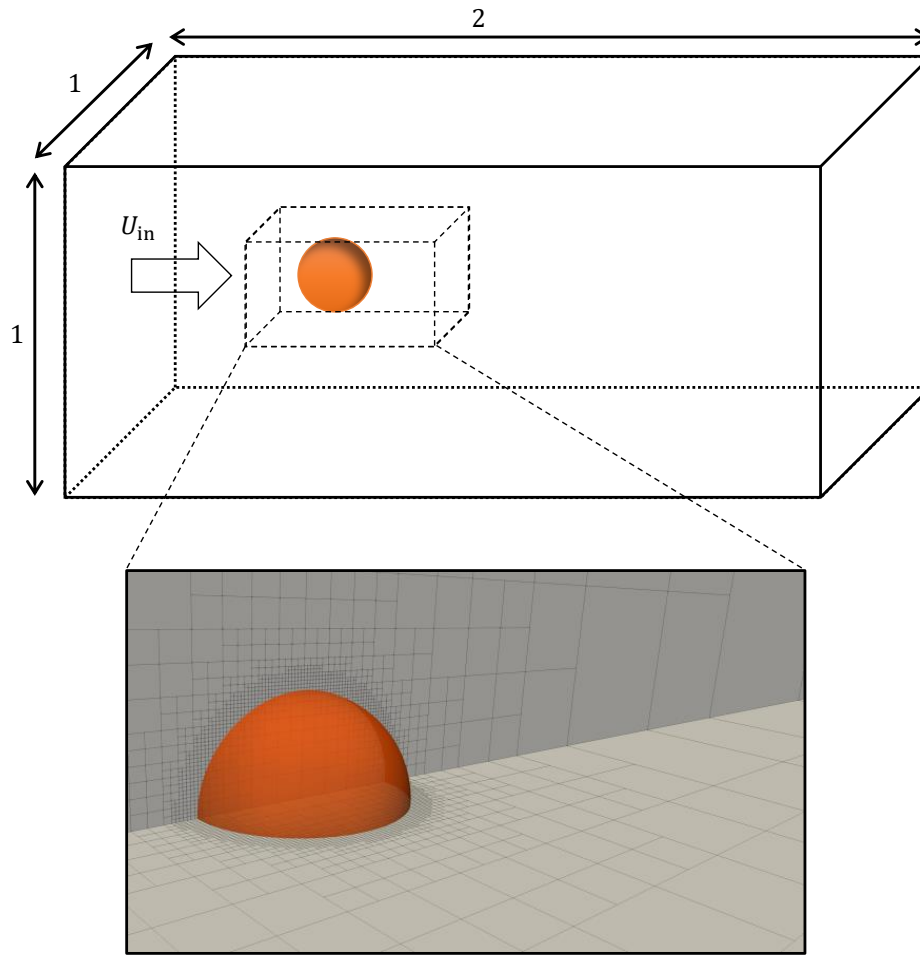


Fig. 5.3: Setup of the computational domain for the flow around a sphere at drag crisis.

れることを期待したものである。

5.2 計算結果

本計算は、流れが十分に発達するまでに 200 万ステップ程度の時間発展を要した。計算には東京工業大学のスパコン TSUBAME2.5 ならびに 3.0 を用いた。TSUBAME2.5 (2017 年 8 月まで稼働) では 64 台の NVIDIA[®] Tesla[®] K20X[®] を用いた並列計算で約 2 週間の計算時間を必要とし、TSUBAME3.0 (2017 年 8 月から稼働) では 16 台の NVIDIA[®] Tesla[®] K20X[®] を用いた並列計算で約 6 日の計算時間が掛かった。

計算結果を Fig. 5.4 に示す。青丸のプロットは「すべての格子で MRT モデル」を用いた計算結果、赤の四角のプロットは「球から 20 メッシュ以内の最近傍格子でキュムラントモデル、他の遠方格子で MRT モデル」の結果、緑の三角は「すべての格子でキュムラントモデル」の結果である。黒色の線は参

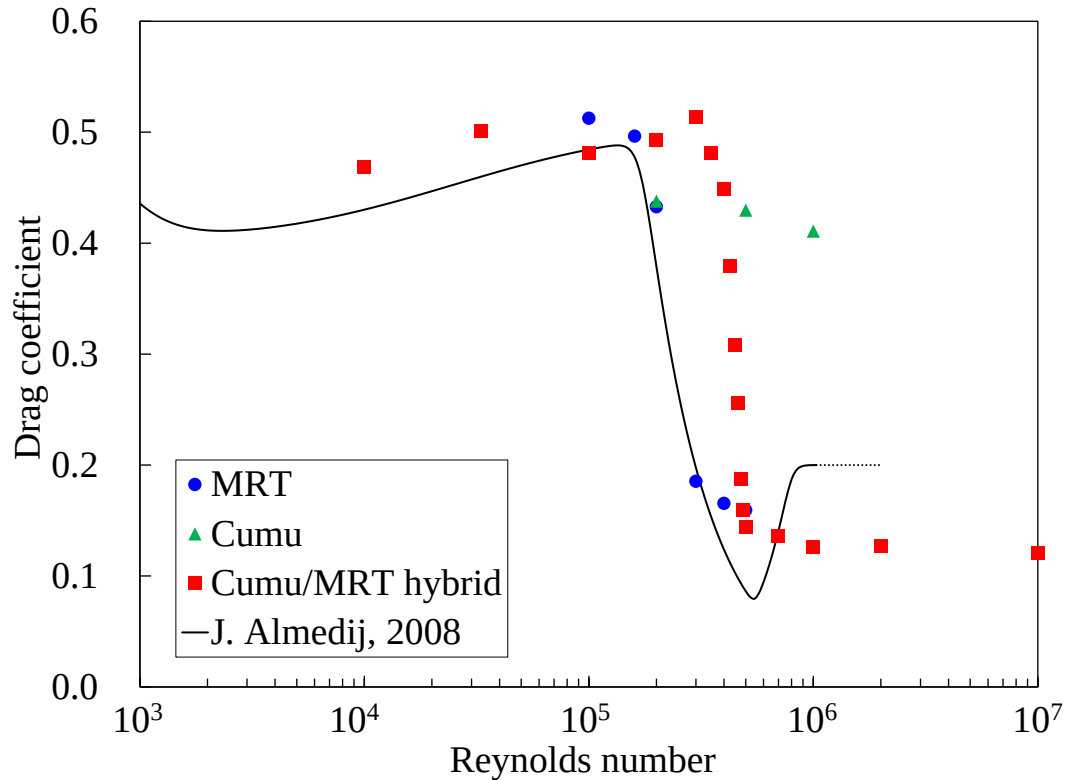


Fig. 5.4: Numerical result of the drag coefficient on a sphere at drag crisis.

照値 [42] である．MRT モデルのみを用いる計算では，ドラッグ・クライシスが再現され，また，抗力係数が低下し始めるレイノルズ数は 20 万で，3 種類の計算条件の中で参照値に最も近い結果になった．ただし，ドラッグ・クライシスよりも高いレイノルズ数においては計算が不安定になり， $Re > 10^6$ では計算が破綻した．本結果から，MRT モデルは計算精度は高いが，格子解像度が十分でない場合には計算が不安定になりやすいことが考えられる．次に，キュムラントと MRT のハイブリッドモデルでは，ドラッグ・クライシスが得られたが，臨界レイノルズ数が 40 万前後と実際よりも大きい値となった．一方で数値安定性は高く， $Re = 10^7$ と非常に高いレイノルズ数でも，抗力係数は参照値と一致しないものの安定に計算することができた．これらの結果から，キュムラントモデルは数値安定性が高いが数値粘性が MRT モデルよりも強く，解像度が不足している場合には数値粘性によって流れの実効レイノルズ数を低く抑える効果を持つと考えられる．最後に，すべての格子でキュムラントモデルを用いた計算では，ドラッグ・クライシスを得ることができなかった．これは，遠方の粗い格子において解像度が大幅に不足し，キュムラントモデルの数値粘性が過剰にはたらくことで，流れの実効レイノルズ数が押し下げられたものと考えられる．以上のように，MRT モデルは計算精度が高いが数値的に不安定になりやすいこと，キュムラントモデルは数値安定性が高いが数値粘性が MRT モデルよりも高い傾向にあることがわかった．また，キュムラントと MRT のハイブリッドモデルは，キュムラントモデル単体ほど数値粘性は強くない，かつ，MRT モデルよりも数値安定性が高い中庸なモデルであることが確認できた．

5.3 本章のまとめ

本章では、複数 GPU の大規模計算により、球周りの流れにおいて $Re > 10^5$ オーダーの高レイノルズ数条件で抗力が急激に低下するドラッグ・クライシスの再現計算を実施した。応用計算のひとつとして球周りの流れにおけるドラッグ・クライシスの再現計算を行った。複数 GPU の大規模計算により、 $Re > 10^5$ オーダーの高レイノルズ数条件で抗力が急激に低下する現象の再現計算を試みた。格子ボルツマン法の衝突モデルとして MRT モデルおよびキュムラントモデル、ならびにそれらを組み合わせたハイブリッドモデルを用い、それぞれのモデルの計算精度と数値安定性を比較した。計算結果から、以下の知見が得られた。

- MRT モデルは、ドラッグ・クライシスの再現精度が最も良いが、 $Re > 10^6$ の高レイノルズ数条件では格子解像度が不足し、数値的に不安定になった。
- キュムラントモデルは数値安定性が高いが数値粘性が MRT モデルよりも高い傾向にあり、キュムラントモデル単体では細分化格子中の粗い格子において過剰な数値粘性がはたらいたため、現状の格子数ではドラッグ・クライシスを再現できなかった。
- キュムラントモデルと MRT モデルを組み合わせることで、キュムラントモデル単体ほど数値粘性は強くなく、かつ、MRT モデルよりも数値安定性が高いハイブリッドモデルを構成することができる。

また、複数 GPU の大規模計算により、球の直径に対して 1000 格子という極めて細かい解像度の計算を実施でき、ドラッグ・クライシスという高レイノルズ数かつレイノルズ数依存性の高い難問を解くことが可能になったことから、本手法の有用性を示すことができた。なお、本計算を行ったのち、2017 年 7 月にキュムラントモデルの数値粘性の問題を改善した 4 次精度キュムラント格子ボルツマン法が発表され [61]、同モデルによるドラッグ・クライシスの再現計算が示された [28]。4 次精度キュムラントモデルは計算精度が大幅に改善されており、MRT モデルに匹敵する精度を有している。初期のキュムラントモデルに比べると 4 次精度モデルは数値安定性が低下するものの、計算が不安定になりつつある格子点において局所的に計算を安定化させるリミターが新たに導入されており、MRT モデル以上の数値安定性を有している。このため、本研究で用いたキュムラントと MRT のハイブリッドモデルよりも、4 次精度キュムラントモデルのほうがより高い精度で計算を行える可能性がある。追計算を行い両モデルの比較を行う必要があるが、本研究では未だ 4 次精度キュムラントモデルの実装が完了しておらず、今後の課題である。

第 6 章

自転車競技における集団走行の大規模空力解析

本研究で実装した格子ボルツマン法および適合細分化格子法の複数 GPU 計算コードを用いた実用計算のひとつとして、自転車競技における集団走行の大規模計算を行う。本章の内容は研究業績 (2) を含む。

高速で走行する自転車は空力の影響を強く受けることが知られている。選手が受ける空気抵抗は平地の走行において全抵抗の 90% 以上を占めており [2]、空気抵抗を低減することは自転車競技のパフォーマンスを向上させる上で非常に重要となる。複数の選手が集団で走行する際には、他の選手の後方を走り、風除けとして利用することで空気抵抗を大幅に削減することができる。たとえば、2 人の選手が縦列で走行する際には、後方の選手にかかる空気抗力は単体走行時の半分程度まで低減される [62]。4 人の選手の縦列走行でも後方の選手の抗力が低減されることは同様であるが、前方の 1 人だけが高い抗力を受け、後方の 3 人の抗力は低くなるため、集団全体で見ると 2 人よりも 4 人の集団のほうが空力的な効率が良くなる [63]。空力解析によって集団走行時の各選手に掛かる空気抗力を評価することで、空力的に効率の良い形状の集団の形成や集団内の位置取りなどの戦略策定に貢献し、自転車競技のパフォーマンスの向上に寄与することが期待される。

自転車競技の空力解析も一般の CFD の例に漏れず、計算制度や規模が計算資源に大きく依存することから、比較的少ない資源で計算できる RANS 計算が用いられることが多かった。既往研究における自転車の CFD 解析事例として、単体走行の計算 [64, 65]、4 人のチームパシュートの計算 [66]、ならびに 121 人のペロトン集団の計算 [67] が挙げられるが、いずれも RANS 解析である。Defraeye らによる単体走行の解析 [68] のように一部では LES による計算も利用され始めているが、計算資源の要求が厳しいため、単体走行の解析に利用されるのみで集団走行への LES の適用事例はまだ見られない。本研究で実装した細分化格子および格子ボルツマン法の複数 GPU 計算コードを用いて、高効率かつ大規模な空力計算を行うことで、LES に基づく自転車競技の集団走行の大規模空力解析を実現する。

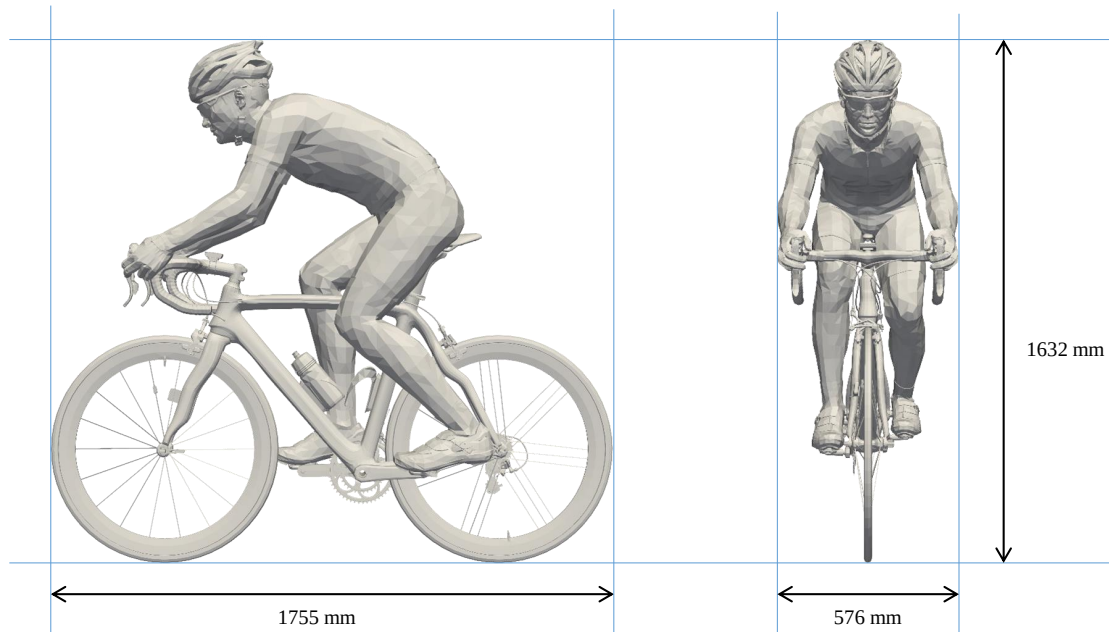


Fig. 6.1: Specification of the 3D cyclist model in upright posture. TurboSquid, utilized with royalty free license [69].

6.1 自転車の 3D モデル

自転車の 3D モデルには、オンラインで販売されているモデル [69] を用いる。同モデルを編集し、選手の姿勢の異なる 2 種類の形状データを作成する。第 1 の形状データは、Fig. 6.1 に示す Upright 姿勢のデータである。Upright 姿勢は上体を起こしているため前面面積が大きくなり空力的には不利だが、無理のない姿勢であり選手の疲労が小さいため、ロードレース競技における巡航でよく用いられる。第 2 の形状データは、Fig. 6.2 に示す Drop 姿勢のデータである。Drop 姿勢は選手がドロップハンドルを持ち上体を前に大きく傾けた姿勢で、Upright 姿勢より前面面積が小さく空力的に有利だが、疲労しやすく姿勢を長時間維持することが難しいため、ゴール直前のスプリントなど瞬間的に高い走行速度を発揮したい場面で用いられる。

本モデルでは足は右足を前、左足を後ろにした位置で静止させている。実際の競技ではペダルを漕ぐため足は回転しているが、これを静止しているものと扱った場合の影響を考える。自転車競技においてペダルの回転数（ケイデンス）は 100 rpm (1.7 s^{-1}) 前後、クランク長さは 0.2 m に満たない程度であるから、ペダルの回転に伴う足の移動速度はたかだか 0.3 m/s となる。これに対してロードレース競技における走行速度はおおよそ 40 km/h (11 m/s) であるから、足の移動速度は自転車の走行速度の 3% にも満たない。このように足の動きは走行速度に比べて十分に遅く、足の動きを無視しても流れへの影響は小さいと考えられる。また、ペダルを漕ぐことで体の揺動が起こることも考えられるが、本稿では考慮していない。

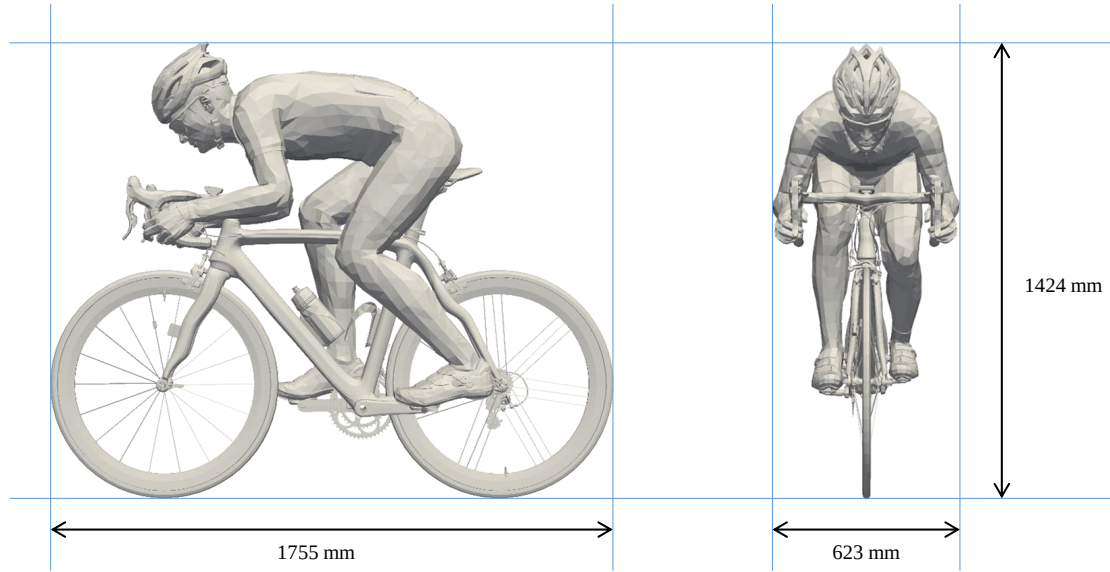


Fig. 6.2: Specification of the 3D cyclist model in drop posture. TurboSquid, utilized with royalty free license [69].

6.2 計算条件

計算領域の大きさは $4194 \text{ m} \times 2097 \text{ m} \times 2097 \text{ m}$ である。格子解像度は、自転車近傍の最も細かい格子で 4 mm ，四方に数メートル程度の範囲で $4, 8$ または 16 mm ，遠方の最も粗い格子で 16384 mm である。Drop 姿勢の選手の単体走行の計算の場合，計算領域および計算格子は Fig. 6.3 に示ようになる。すべての計算で計算領域の大きさは固定，自転車の配置のみを変更するとして，それぞれの自転車の近傍に同様の細分化格子を生成する。また，複数の自転車が配置されている場合，自転車間の空隙には最大で 16 mm まで粗い格子を割り当てる。たとえば，4 台の自転車が縦に並ぶ配置においては，Fig. 6.4 のような細分化格子が生成される。主流流速を 16.67 m/s ，最も細い格子における時間刻み幅を $4.20 \mu\text{s}$ とする。20 °C の空気を仮定し，流体の密度を 1.205 kg/m^3 ，動粘度を $1.512 \times 10^{-5} \text{ m}^2/\text{s}$ に設定する。流れのレイノルズ数は，地面から選手の頭の先までの高さを代表長さにした場合，Upright 姿勢で 180 万，Drop 姿勢で 157 万である。格子ボルツマン法の分子速度は $U/c = 0.03$ となるように設定しており，疑似音速は 320.8 m/s ，マッハ数は 0.052 ，最細格子における時間刻み幅は $7.20 \mu\text{s}$ となる。

境界条件は以下のように設定する。まず，計算領域前方の流入面および後方の流出面には密度および速度固定の Dirichlet 条件を適用し，格子ボルツマン法の分布関数に平衡分布関数を設定する。一般には流出面における速度の境界条件には勾配零の Neumann 条件が与えられるが，本計算は高レイノルズ数条件でありかつ流出面に粗い格子を配置することから計算が不安定になりやすいため，より高い数値安定性を得るために固定値の条件を与えた。ただし，流出面は自転車から 2 km 以上と十分遠方の位置にあるため，自転車周辺の流れにはほとんど影響しないと考えられる。計算領域の左右の端面は周期境界と

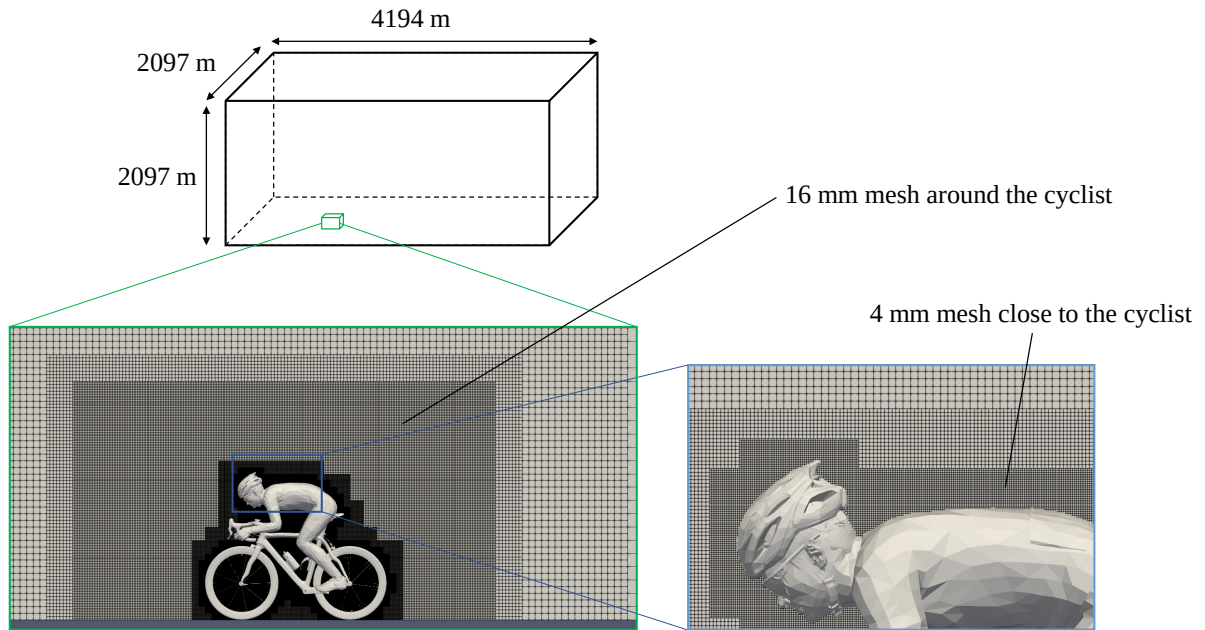


Fig. 6.3: Computational domain and mesh near the cyclist with drop posture.

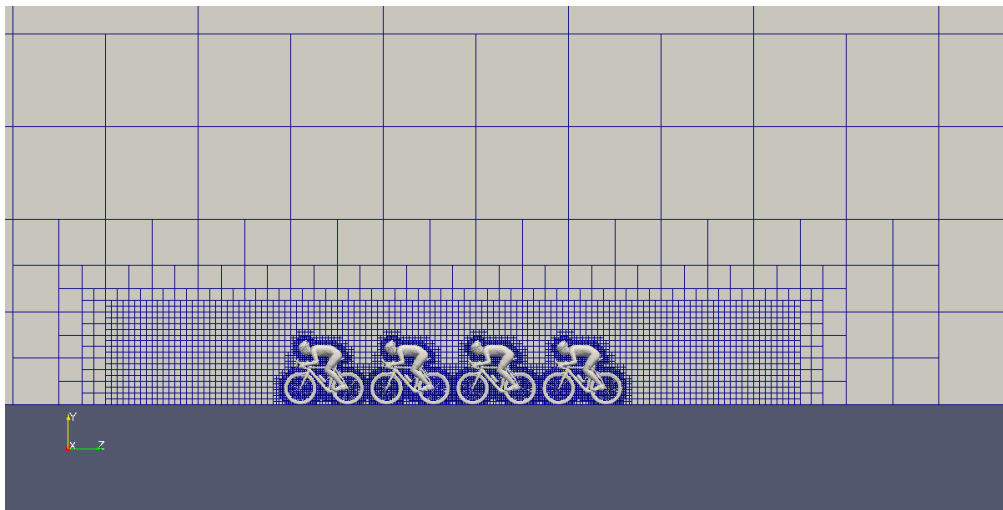


Fig. 6.4: Computational mesh near the 4 cyclists with serial arrangement.

した．計算領域の下面には，16 mm 以下の格子に Interpolated bounce-back による滑りなし壁を設定し，32 mm 以上の粗い格子では上下面で周期境界とした．実際の競技では自転車移動しているため本計算条件とは異なるが，下面を静止させた条件は簡易であるため既往の風洞実験や数値計算でもよく用いられている [68, 64, 70, 66, 62, 63, 71, 72, 73, 67]．

本計算は自転車および選手の複雑な形状を含むため計算が不安定になりやすく，数値安定性を最重要

に計算スキームを選択する必要がある。予備計算として、コヒーレント構造 Smagorinsky モデルに基づく渦粘性を導入した上で、MRT モデルおよびキュムラントモデルモデル、ならびにそれらのハイブリッドモデルによる計算を実施したところ、MRT モデル、MRT/キュムラントのハイブリッドモデルでは計算が破綻し、安定的に計算できたのはキュムラントモデル単体の計算のみであった。そこで、格子ボルツマン法の衝突項には最も数値安定性の高いキュムラント衝突項を用いる。さらに、数値安定性や質量・運動量の保存性を向上させるため、Rohde らの文献 [22] に則り解像度の異なる格子間においてレベル変換を行わないものとした。

抗力を評価するため、一般の空力解析では抗力係数を用いるが、自転車競技では、実験において選手の姿勢や体格を一定に保つことが難しく、前面面積を計測することが困難である。このため自転車競技の空力では、前面面積の測定が不要な抗力面積 (drag area; $C_D A$) がよく用いられる [72]。抗力面積は、次式で定義される。

$$C_D A = \frac{D}{\frac{1}{2}\rho U^2} \quad (6.1)$$

D は抗力である。本計算では、Momentum-exchange method [35] に基づいた抗力の瞬時値が各時刻ステップで得られ、抗力面積も同様に各時刻ステップで瞬時値が得られる。抗力面積の時刻歴の例として、Drop 姿勢の選手の単体走行計算で得られた抗力面積の時刻歴を Fig. 6.5 に示す。横軸は時刻、縦軸は抗力面積である。図のように、2 秒以下の区間 (flow-developing interval) においては初期条件から流れが発達している段階にあり、抗力の変動が安定していない。2 秒以降の区間 (averaging interval) では抗力の変動は一定の範囲に収まっており、流れが十分に発達していると考えられる。本章では、流れが十分に発達している区間において抗力面積の平均値を求め、実験値との比較および評価を行う。なお、流れが十分に発達するまでの時間の閾値は自転車の配置によって異なると考えられる。平均区間を 2~5 秒と一定とするのではなく、それぞれの計算結果について時刻歴を確認し、それぞれに適した閾値を設定する。また、計算の終了時刻についても、5 秒後と一定とするのではなく、計算条件ごとに十分長い averaging interval が得られる時間を設定する。

Fig. 6.5 の例について、averaging interval の区間での抗力面積の時間変動をフーリエ変換し、周波数スペクトル解析を行った結果を Fig. 6.6 に示す。横軸は周波数、縦軸は振幅である。青色の実線はフーリエ変換により得られた値であるが、ノイズのような振動が大きくスペクトルの傾向を見ることができない。そこで、フーリエ変換で得られたスペクトルに対し、周波数空間で移動平均を取った値を赤色の実線で示している。黒色の破線は $-5/3$ 乗則の補助線である。この図から、抗力面積のスペクトルは、80 Hz 以上の領域においてエネルギー Spektral の $-5/3$ 乗則に従っていることがわかる。乱流を妥当に計算できていることが確認でき、また、80 Hz 以下の抗力の変動については LES でモデル化できることがいえる。80 Hz の抗力変動は、主流速度が 16.67 m/s であることから 21 cm 程度の渦に相当すると推測できる。格子解像度は、自転車近傍ならびにその後流で関心のある領域において、それぞれ 4 mm ならびに 16 mm であり、21 cm 程度の渦のスケールに対して十分に細かい。このため、本計算において自転車近傍およびその後流における格子解像度は LES で計算するのに十分な解像度があるといえる。

上でも述べたように、本計算では自転車の後流で関心のある領域には 16 mm までの粗い格子を割り当

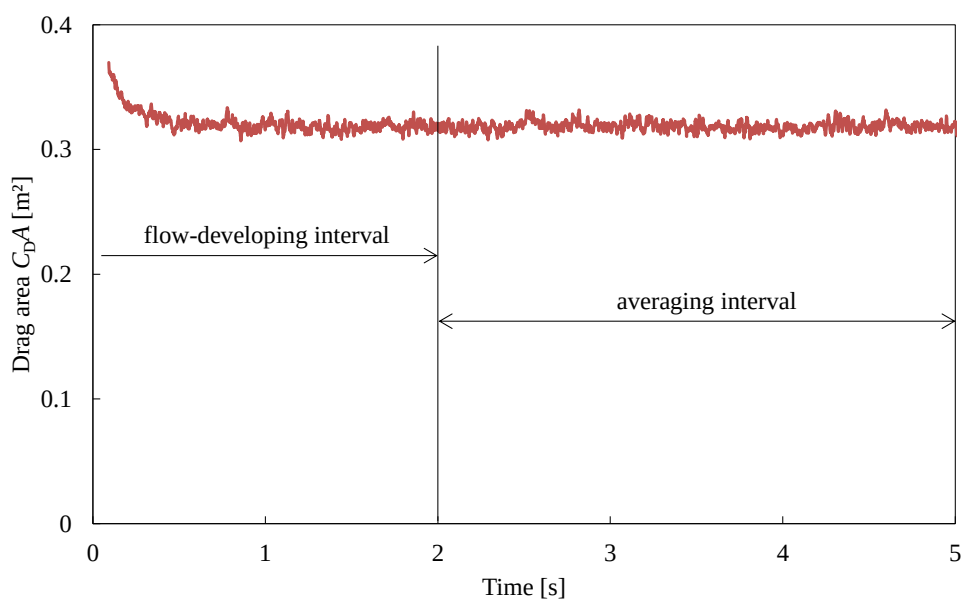


Fig. 6.5: Time history of the drag area on the isolated cyclist with drop posture.

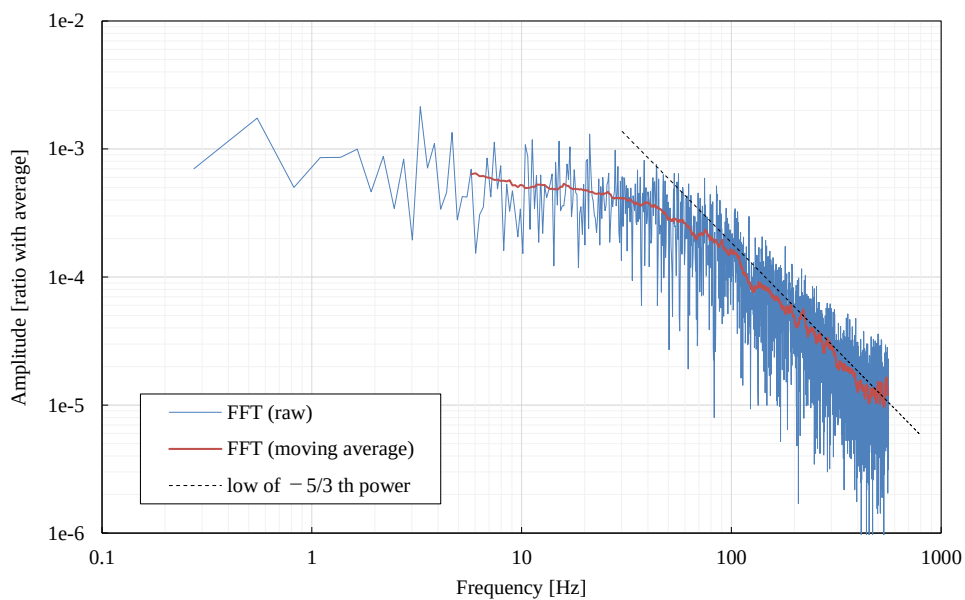
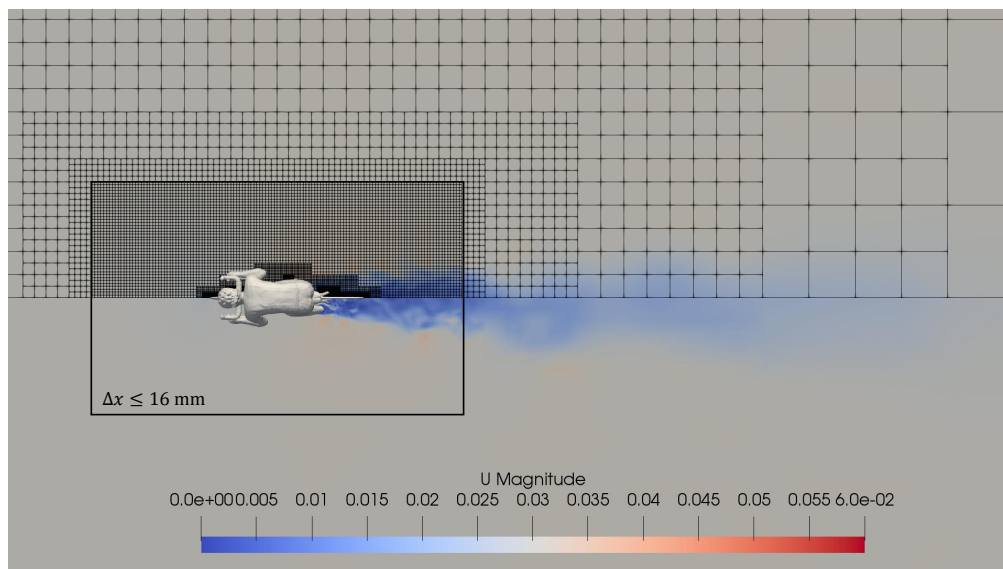
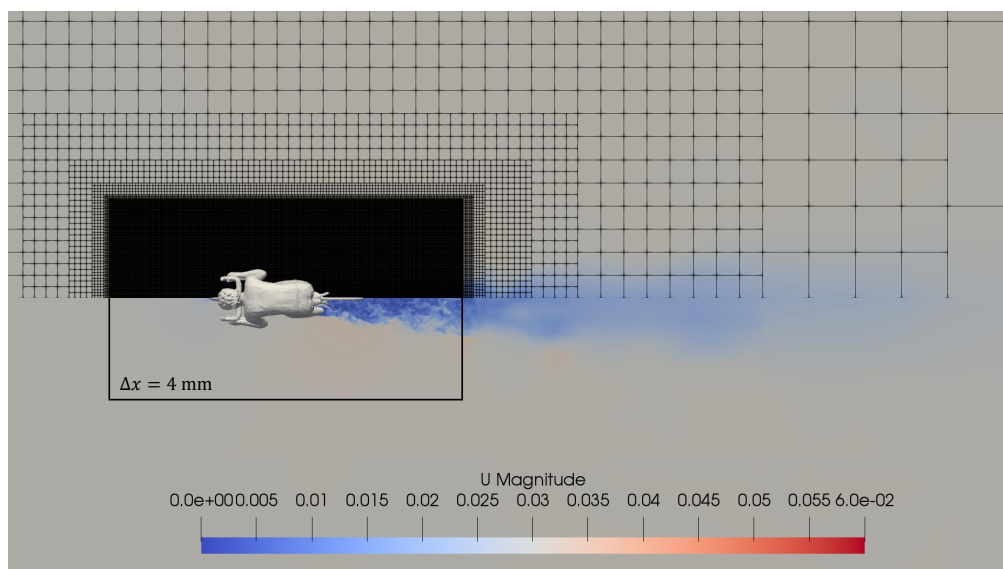


Fig. 6.6: Frequency spectrum of the drag area on the cyclist with drop posture.



(a) 16 mm mesh in the wake.



(b) 4 mm mesh in the wake.

Fig. 6.7: Velocity field behind a cyclist with different mesh resolution in the wake.

てている。抗力変動の周波数スペクトルから、後流の格子解像度は 16 mm で十分に細かいことが示されているが、後流にも 4 mm 格子を割り当てたほうがより細かい渦を捉えることができ、計算精度が向上することも予想される。しかし、後流の広い範囲を 4 mm 格子で解像するには多数の格子点が必要であり、計算資源とのトレードオフの問題が発生するため、後流には格子依存性が十分に収束する限りにおいてなるべく粗い格子を割り当てるべきである。そこで、後流の格子解像度が 16 mm で十分であることを示すため、16 mm 格子の領域を 4 mm 格子で置き換えた場合の計算との比較を行った。後流を 16 mm 格子、4 mm 格子で解像した場合の速度の分布を Fig. 6.7 に示す。計算領域を上から見上げた図であり、

地面から 0.5 m の断面における速度分布および自転車右側の格子配置を示している。黒枠は、自転車の後流で関心のある領域として、16 mm または 4 mm で解像した格子の範囲を表す。(a) は後流を 16 mm で解像した計算結果、(b) は後流を 4 mm で解像した計算結果である。(b) の結果のほうがより細かい速度変動を捉えられていることが確認できる。ただし、2 種類の解像度の計算において、黒枠内の格子ではほぼ同じ幅の後流が得られており、16 mm の格子で後流を十分に解像できている。また、抗力面積の結果も同程度であった。したがって (a) のように後流を 16 mm の少し粗い格子で解像した場合でも、LES によって流れは十分にモデル化されており、4 mm の格子と同様の後流が得られることが確認できた。4 mm 格子の計算は、単体走行であるにもかかわらず、64 台の GPU による並列計算を要しており、集団走行の計算に適用することは計算資源の観点から現実的でない。したがって、本計算においては後流の領域に 16 mm 格子を割り当てるのが妥当であるといえる。

6.3 検証計算

6.3.1 単体走行の検証計算

第 1 の検証計算として、Upright 姿勢と Drop 姿勢それぞれの単体走行の計算を行い、既往研究の風洞実験の結果と比較する。4 台の GPU (NVIDIA[®] Tesla[®] P100) を用いた並列計算により物理時間 5 秒に相当する計算を行い、約 1 日の計算時間を要した。物理時間 2 秒から 5 秒の区間で抗力面積の平均値を求めた。計算結果および既往研究の実験結果 [74, 2, 75, 76, 77, 62] を Fig. 6.8 に示す。棒グラフは計算結果で、青、赤、緑の棒はそれぞれ最細格子を 8, 4, 2 mm としたときの計算結果を示している。エラーバーは抗力面積の時間的変動の範囲を表している。また、黒点は実験値である。縦軸は抗力面積の大きさを表す。実験は異なる体格や姿勢の選手、自転車、風洞設備の条件で行われ、それぞれの実験で得られた抗力面積の平均値が報告されている。足を静止させた条件とペダルを漕ぐ条件の両方の実験があるが、ペダリングの有無よりも選手の体格差に依存して結果が大きく異なっているため、文献からペダリングの影響を評価することは困難である。本計算と完全に同じ条件で行われた実験はなく、定量的な検証は難しい。しかし、いずれにせよ本計算結果は、様々な選手による実験結果と同程度の値であり、また、Upright 姿勢に比べ Drop 姿勢の抗力面積が小さくなる傾向を再現していることから、ある程度妥当な結果である。検証のため異なる解像度の計算として、最細格子を 8 mm または 2 mm とした条件の計算も行ったが、抗力面積の平均値および変動の幅は同程度であった。2 mm と 4 mm の格子解像度による抗力面積の差異は最大 5.0% であり、抗力面積の格子依存性による有意な差は見られなかった。なお、抗力面積の変動の幅は平均値に比べて小さく、また、既往研究においても平均値のみが評価されているため、本論文でも以降は平均値についてのみ議論する。

6.3.2 集団走行の検証計算

自転車競技における集団走行の検証として、4 人の選手が集団で走行する条件の計算を行う。

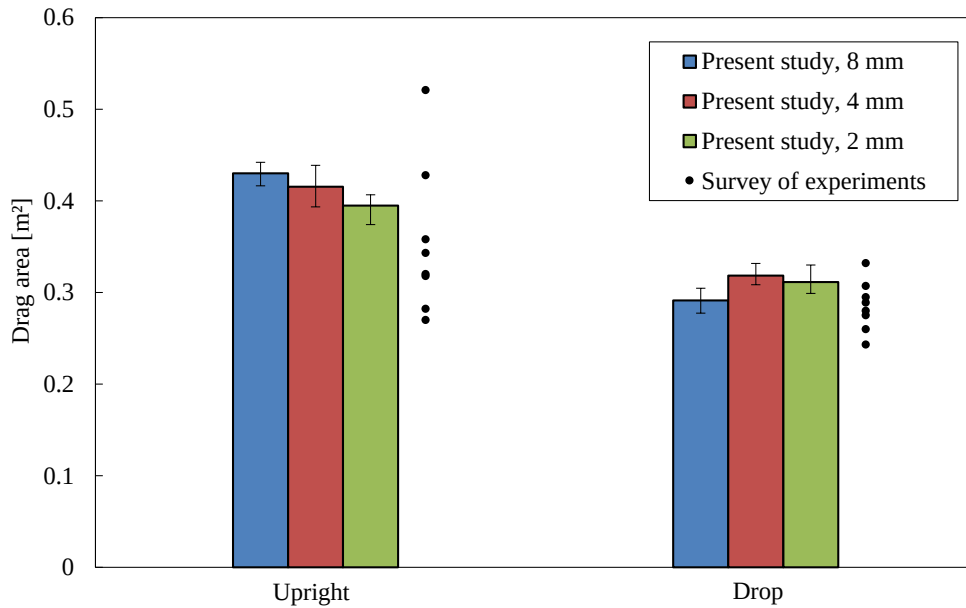


Fig. 6.8: Result of the drag area on alone running cyclist in upright/drop posture.

縦列走行の条件

第一の条件として、4人の選手が1列に整列した縦列走行の条件で計算を行う。選手の姿勢はDropとし、自転車間の距離は0.15 mに設定する。8台のGPU (NVIDIA® Tesla® P100) を用いた並列計算により物理時間で4秒間に相当する計算を行い、約2日の計算時間を要した。物理時間2秒から4秒の区間で抗力の平均値を求めた。計算結果および参照値 [78, 66, 63, 73, 67] を Fig. 6.9 に示す。棒グラフは計算結果、折れ線グラフは既往研究の風洞実験または RANS 解析による参照値である。縦軸は先頭の選手に掛かる抗力を100%としたときの各選手に掛かる抗力の割合である。本計算では2,3人目の抗力低減が参照値に比べて強すぎる傾向が見られるが、文献ごとに選手の体格や姿勢などの諸条件が異なるため定量的な比較は難しい。また、本計算では4人目の抗力が3人目よりも高くなる傾向が見られるが、これは終端後端の負圧の影響によるものと考えられる。自転車競技においても最後尾の選手の抗力が僅かに高くなることは経験的に知られており、また、Blocken らの風洞実験でも3人目の抗力よりも4人目の抗力が1.0%と僅かに高くなっていることから、このような抗力の上昇は妥当な計算結果であると考えられる。いずれにせよ、先頭の選手に比べて後方の3人に掛かる抗力が半分程度まで小さくなっていることや、後方の3人に掛かる抗力は近い値となるが2番目の選手より3番目の選手に掛かる抗力が僅かに小さくなることなど、定性的な傾向が一致しており、本計算結果はある程度妥当である。

Shirasaki ら [73] の条件

次の検証として、Shirasaki らの実験 [73] を模擬した計算を行う。Fig. 6.10 または Fig. 6.11 のように、縦列走行において最後方の選手が規定の位置から縦方向に X [m] または横方向に Y [m] すらして配

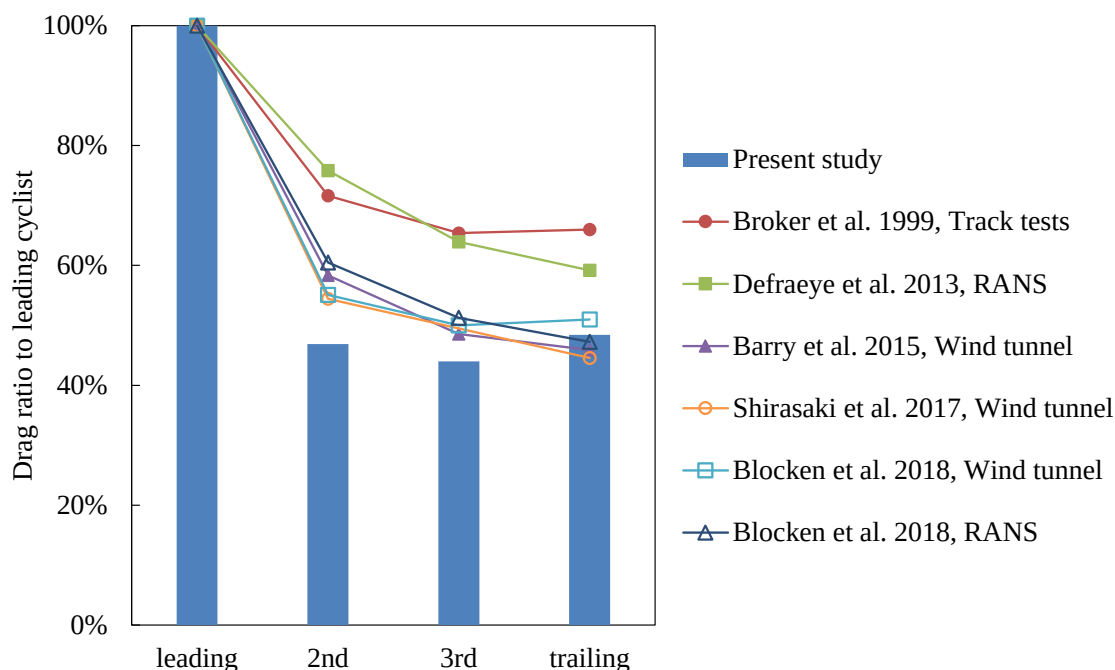


Fig. 6.9: Result of the drag area on the group running of 4 cyclists in a row.

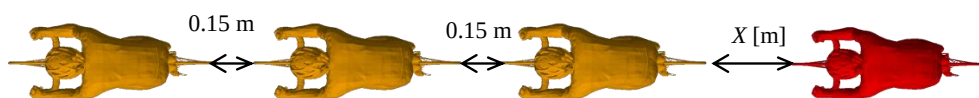


Fig. 6.10: Top view of the disturbed row arrangement appeared in Shirasaki et al. 2017 [73]: the trailing cyclist is X [m] behind from the other 3 riders in a row.

置する．本計算では最後尾の選手の抗力のみを評価する．8 台の GPU (NVIDIA[®] Tesla[®] P100) を用いた並列計算により物理時間で 4 秒間に相当する計算を行い，約 2 日の計算時間を要した．計算結果および既往研究の実験結果を Fig. 6.12, Fig. 6.13 に示す．横軸は最後尾の選手の位置のずれ，縦軸は先頭の選手の抗力を 100% としたときの最後尾の選手の抗力の割合である．これらの結果から，縦方向のずれ (X) による抗力の変動は小さいが，横方向のずれ (Y) は抗力の増加に強く影響することがわかる．参照値は自転車を固定する治具や風洞の閉塞率により誤差を含んでいると考えられる．また，本計算と既往研究の実験で選手の体格や姿勢，自転車の形状などの諸条件が異なるため結果は厳密には一致し得ない．しかし，いずれにせよ本計算は実験で得られる傾向を再現しており，ある程度妥当な計算結果である．

Blocken ら [67] のダイヤモンド配置の条件

最後の検証として，Blocken らのダイヤモンド配置 [67] で計算を行う．Fig. 6.14 のように，1 列目に 1 人，2 列目に 2 人，3 列目に 1 人を並べた配置を設定する．縦方向の車間距離 (wheel-to-wheel distance)

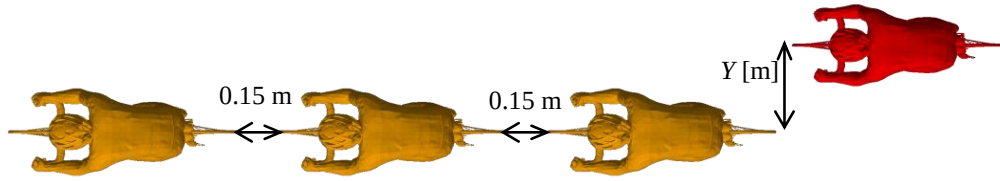


Fig. 6.11: Top view of the disturbed row arrangement appeared in Shirasaki et al. 2017 [73]: the trailing cyclist is Y [m] laterally displaced from the other 3 riders in a row.

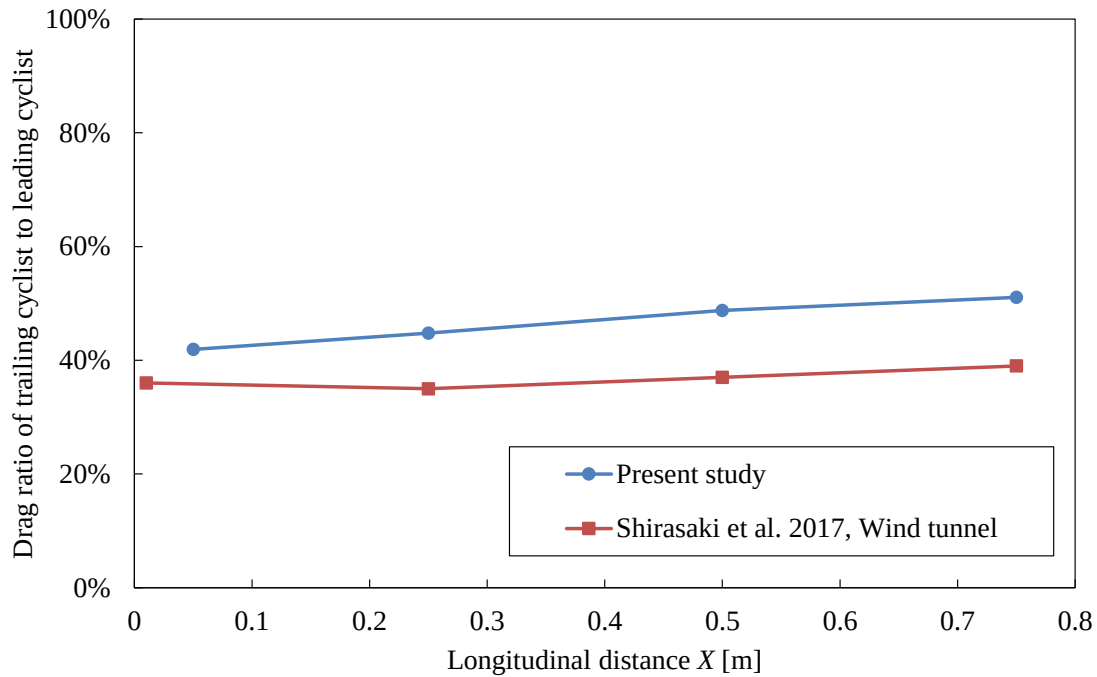


Fig. 6.12: Result of the drag on the trailing cyclist with longitudinal distance X in the group of 4 cyclists.

は 0.15 m であり、横方向の位置は、2 列目において自転車の中央位置から隣の自転車の中央位置までの距離 (center-to-center distance) が 0.8 m となるように設定する。8 台の GPU (NVIDIA[®] Tesla[®] P100) を用いた並列計算により物理時間で 4 秒間に相当する計算を行い、約 2 日の計算時間を要した。

計算結果および参照値を Fig. 6.15 に示す。横軸のラベルは Fig. 6.14 で示した各選手の位置を表し、縦軸は単体走行時に対する各選手の抗力の割合を示す。参照値 [67] は風洞実験および RANS 解析による。本計算結果は、1 列目の選手の抗力が大きいこと、2 列目の選手の抗力は 1 列目の選手に比べてわずかに小さいこと、3 列目の選手の抗力は集団中で最も小さく単体走行時の 40~50% まで低減されることなどの傾向を再現している。これまでの検証計算と同様、計算と参照値で自転車の形状や選手の体格など諸条件が異なるため結果が厳密に一致することはないが、上記のように定性的な傾向は一致しており、妥当な計算結果である。

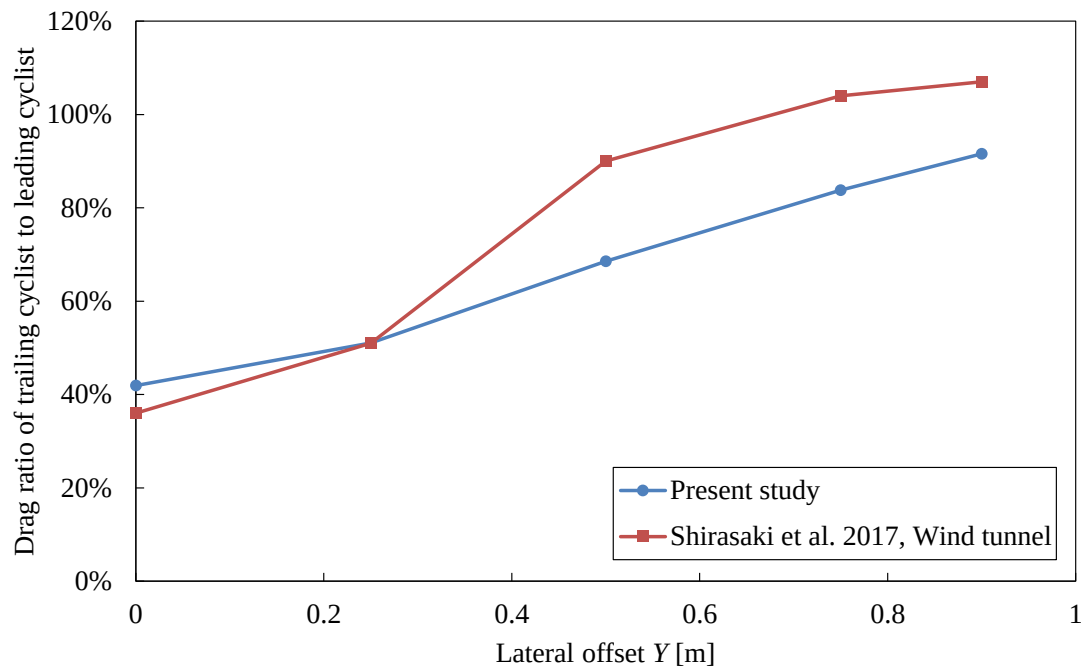


Fig. 6.13: Result of the drag on the trailing cyclist with lateral offset Y in the group of 4 cyclists.

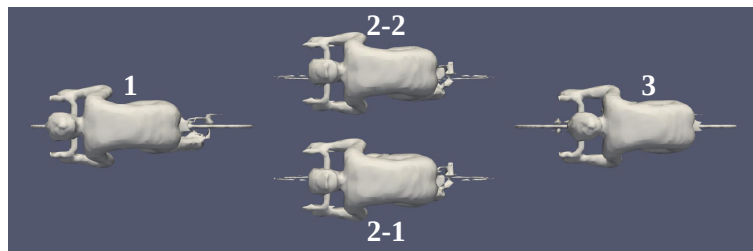


Fig. 6.14: Top view of the diamond arrangement appeared in Blocken et al. 2018 [67]

以上の単体走行および集団走行の検証計算により本手法の妥当性を確認し、本手法が集団走行の空力解析に応用可能であることを示した。

6.4 集団走行の応用計算

本計算コードによる自転車競技の応用計算として、多数の自転車を含む集団走行の大規模計算を実施する。

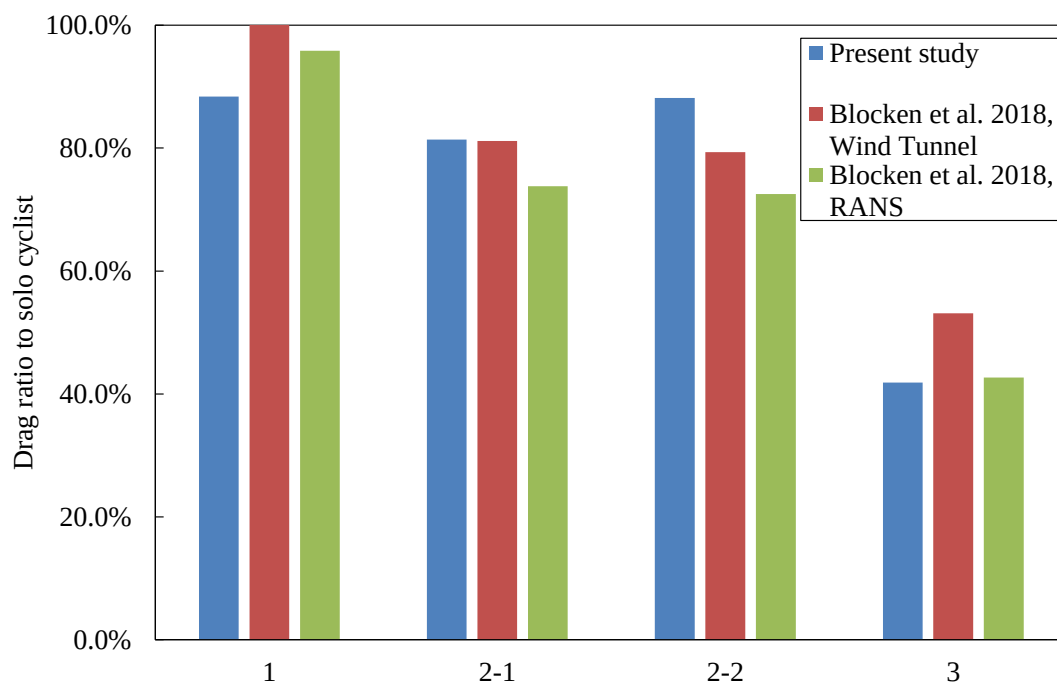


Fig. 6.15: Drag ratio on each cyclist in the diamond arrangement to solo cyclist.

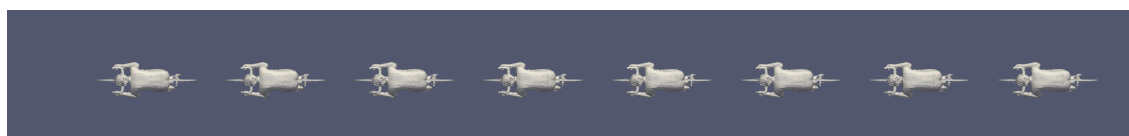


Fig. 6.16: Arrangement of the in-line group of 8 cyclists.

6.4.1 8人の集団の計算

8人の選手で1つの集団を構成し、集団内の選手の配置として、Fig. 6.16に示す縦列配置またFig. 6.17のひし形配置を設定する。これらの配置において、縦方向の車間距離（wheel-to-wheel distance）は0.15 m、横方向の位置は自転車の中央位置から隣の自転車の中央位置までの距離（center-to-center distance）が1 mとなるように設定する。縦列配置とひし形配置のそれぞれについて、単一の集団の計算を行う。計算領域の大きさ、自転車近傍及び遠方の格子解像度など諸計算条件は検証計算と同様である。GPU数は、縦列走行の計算において20台、ひし形配置の計算において32台に設定した。流れが十分に発達して平均抗力を求めるのに十分な時間が経過するまで計算を行った。物理時間で8秒間の計算を行い、約4日の計算時間を要した。

縦列配置の計算結果をFig. 6.18に、ひし形配置の計算結果をFig. 6.19に示す。自転車の色で抗力面積の値を表すとともに、単体走行時の抗力を100%としたときの各選手に加わる抗力の大きさの割合を数字で示している。縦列配置では先頭に1代に単体走行時の99.5%と高い抗力が集中し、後方の7人に

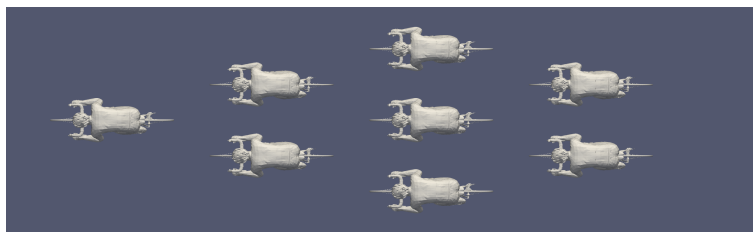


Fig. 6.17: Arrangement of the rhomboid group of 8 cyclists.

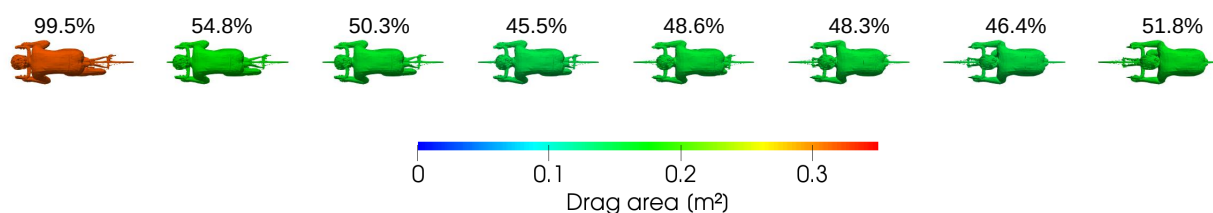


Fig. 6.18: Result of the drag area and drag ratio on each cyclist in the in-line group of 8 cyclists.

掛かる抗力は 50% 程度まで小さくなっている。8 人全員の抗力の平均値は 55.6% と小さい。一方、ひし形配置では、前方外側の 5 人の選手に 90% の高い抗力が掛かっており、全選手の抗力の平均値は 74.4% と縦列配置に比べて高い。しかし、後方左側の選手に掛かる抗力は 35.0% となっており、縦列配置における最小値 45.5% に比べて小さい。これらの結果から、8 人全体の平均を考えると縦列配置が効率的であるが、特定の選手に掛かる抗力を最小化し、同選手の体力を温存させたいような場合には、ひし形配置のように横に広がる配置も有効であると考えられる。なお、ひし形配置では、左側に配置された選手の抗力が右側の選手よりも小さい傾向にあり、抗力が左右非対称に分布している。これは、ペダルや足の角度、自転車のギアなど、自転車および選手の形状が左右非対称に分布しているためである。ペダルを半周させ、左足を前、右足を後ろにした形状データを用いた場合、非対称分布がほぼ逆になり、右側に配置された選手の抗力が左側の選手よりも小さくなる。

6.4.2 2 つの小集団で構成される 16 人の集団の計算

前節で計算した縦列配置またはひし形配置を 1 つの小集団とし、前後に 2 つの小集団が並んだ計算を行った。前後の小集団間の距離を 0.5 m, 10 m, 20 m の 3 種類に設定し、距離ごとに各選手に加わる抗力を求める。計算領域の大きさ、自転車近傍及び遠方の格子解像度など諸計算条件は検証計算と同様である。集団の配置に応じて GPU 数を 40~64 台に設定し、流れが十分に発達して抗力の平均値を求めるのに十分な時間区間が得られるまで計算を行った。物理時間で 8~11 秒の計算を必要とし、4~5 日の計算時間を要した。

2 つの集団が接近しているときの計算結果を Fig. 6.20 および Fig. 6.21 に、前後の集団の距離が 10 m の条件での計算結果を Fig. 6.22 および Fig. 6.23 に、前後の集団の距離が 20 m であるときの計算結果を Fig. 6.24 および Fig. 6.25 に示す。自転車の色は各選手に掛かる抗力面積の値であり、パーセンテージ

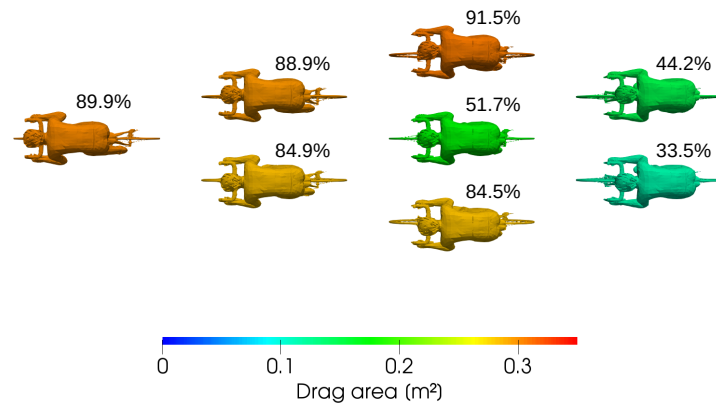


Fig. 6.19: Result of the drag area and drag ratio on each cyclist in the rhomboid group of 8 cyclists.

ジで表示された値は単体走行時の抗力を 100% としたときの各選手に加わる抗力の大きさの割合である。縦列配置およびひし形配置のいずれにおいても、前方の集団の各選手に掛かる抗力は集団間の距離によらずほぼ一定であるが、後方集団の各選手に掛かる抗力は集団間の距離が小さいほど小さくなっている。また、集団間の距離が同じ条件での縦列配置とひし形配置の計算結果を比較すると、ひし形配置のほうが後方集団の抗力の減少が強い傾向がある。これらの計算結果から、前方集団は後方集団に風除けとして利用されてしまうこと、ひし形配置では風除けの効果が増大し後方集団にとって有利（逆説的に、前方集団にとって不利）となってしまうことがいえる。上記の結果から、競技における戦略として、いくつかの提案をすることができる。たとえば、後方の集団に対しては、

- なるべく前方の集団に近い位置を走行する。
- 前方の集団に追いついた場合、すぐに追い抜くのではなく、しばらく後方に付いて、風除けとして利用することで体力を温存する。

などの戦略が考えられるし、前方の集団に対しては、

- なるべく後方の集団から距離を取って走行する。
- ひし形配置を取っているときに、後方集団に追いつかれてしまった場合、風除けの効果を弱めるために縦列配置に転換する。

といった戦略は空力的に有効であると考えられる。なお、ひし形配置の 2 集団の走行では 1 集団の計算よりも左右非対称性が強く、非対称性が 2 倍以上に達するところもあるが、これは、1 集団のときと同様に足の角度による選手の形状の左右非対称に起因すると考えられる。すべての選手が同じ足の角度を取るため、人数が増えるほど非対称性が累積されることが想定される。2 集団の条件においては、後方集団において、自身の形状の左右非対称性と、前方集団が作り出した非対称性が加算された結果、より強い左右非対称性が得られたと考えられる。Blocken らのペロトン集団の RANS 解析でも [67] においても同様

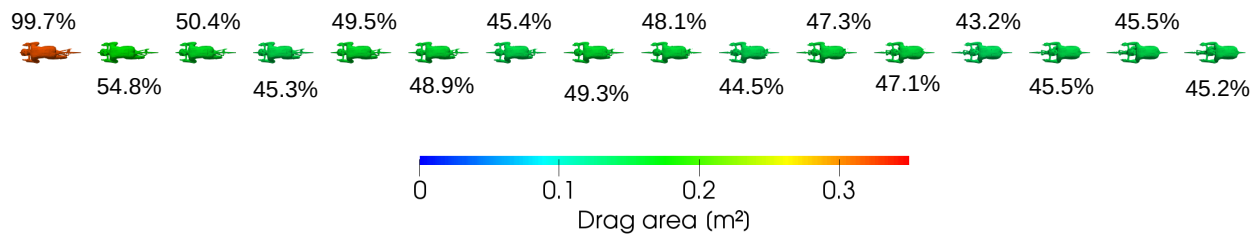


Fig. 6.20: Result of drag area and drag ratio on each cyclist in the two closer in-line groups.

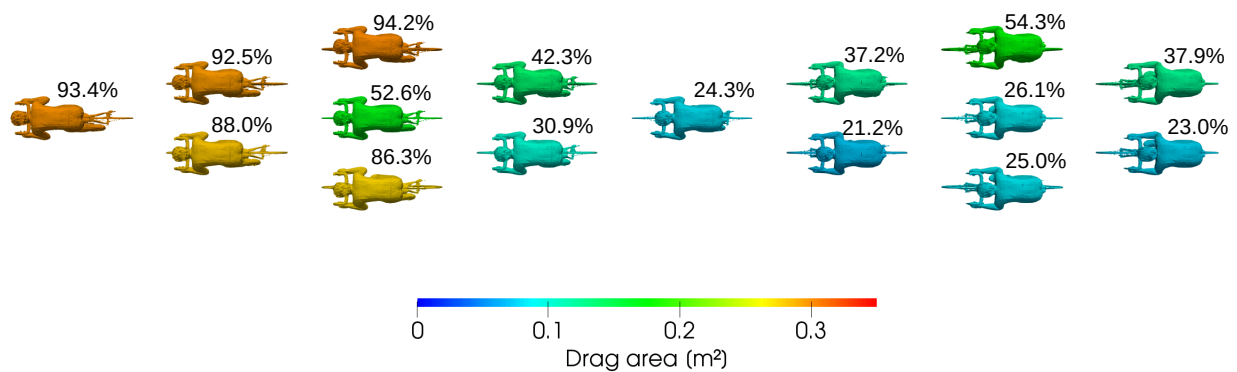


Fig. 6.21: Result of drag area and drag ratio on each cyclist in the two closer rhomboid groups.

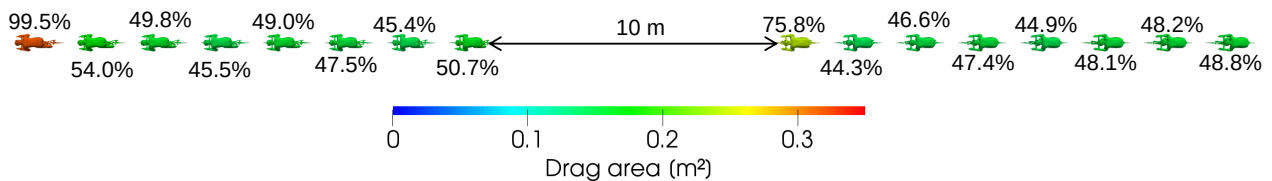


Fig. 6.22: Result of drag area and drag ratio on each cyclist in the two in-line groups with 10 m displacement.

に 2 倍以上の非対称性が見られる。足の角度に起因する非対称性を取り除くため、今後、足の角度を変えた計算またはペダルの回転を含む計算が必要である。

6.4.3 72 人の集団の大規模計算

本手法による自転車の大規模計算のベンチマークとして、ロードレースを想定した 72 人の集団の計算を行い、計算結果ならびに計算時間を確かめた。Fig. 6.26 のように、縦と横に 6×6 に並んだ 36 人の小集団を設定し、前後に 10 m 離れた 2 つの小集団を配置する。計算領域の大きさ、自転車近傍及び遠方の格子解像度など諸計算条件は検証計算と同様である。

本計算条件における格子点数は 22.3 億であった。192 台の GPU (NVIDIA[®] Tesla[®] P100) を用いた大規模並列計算により、物理時間 7.2 秒に相当する計算を行い、4 日の計算時間を要した。計算結果を

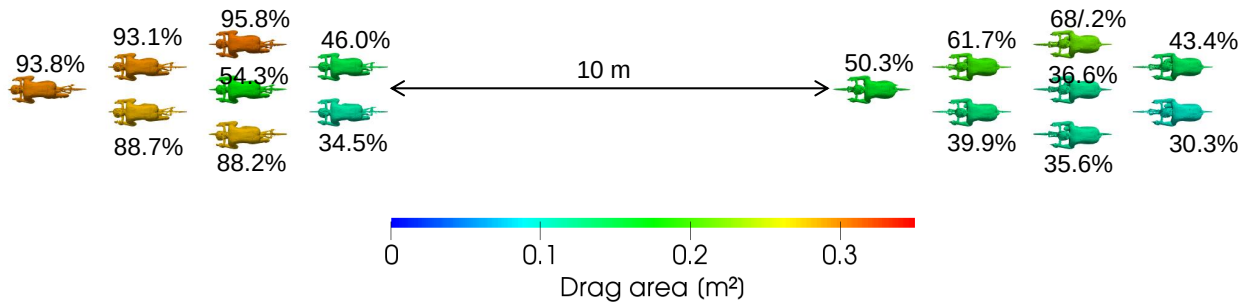


Fig. 6.23: Result of drag area and drag ratio on each cyclist in the two rhomboid groups with 10 m displacement.

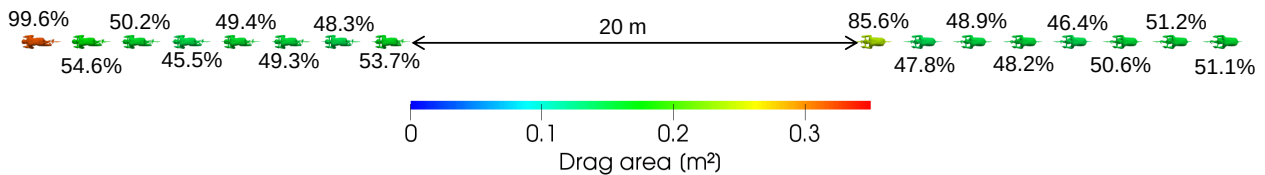


Fig. 6.24: Result of drag area and drag ratio on each cyclist in the two in-line groups with 20 m displacement.

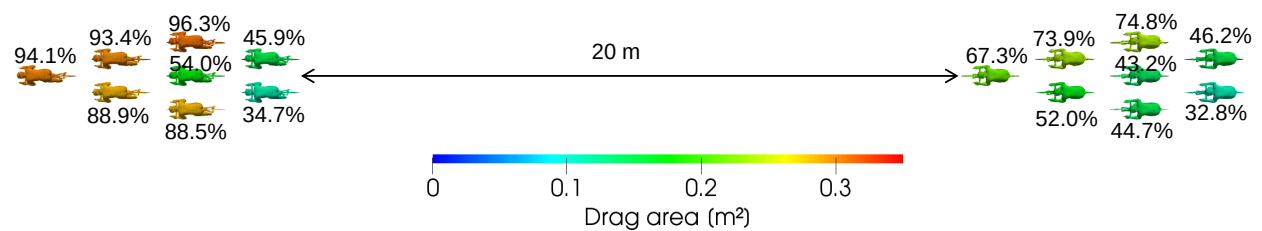


Fig. 6.25: Result of drag area and drag ratio on each cyclist in the two rhomboid groups with 20 m displacement.

Fig. 6.27 に示す．横軸および奥行き方向の軸で各選手の配置を表し，縦軸は集団内で最大抗力を受ける選手（横軸 1，奥行き軸 6 の位置の選手）の抗力を 100% としたときの各選手に加わる抗力の大きさの割合である．最前列の 6 人の選手に高い抗力が掛かるが，その中でも特に，内側よりも左右の端に近い選手ほど高い抗力を受けていることがわかる．本計算では前項までの計算とは異なり，最前列でも後方の列でも等しく横 1 列に 6 人を並べているが，これにより外側の選手に高い抗力が加わったと考えられる．横に広がる配置の場合でも，最前列には 1 人だけを配置し，後ろにいくほど横に広がる人数を増やすような構造を採れば，前方外側の選手が受ける抗力を抑えられると考えられ，単純に縦と横に正方形に広がるような配置よりも，前項で計算したようなひし形配置のほうが有効であることが示唆される．また，後方 36 人の集団での抗力低減が特に大きいことがわかる．後方集団の中心付近の選手に加わる空気抗力は，最前列で最大の抗力を受ける選手に比べて 8.8% にまで低減されており，大人数での集団走行におけ

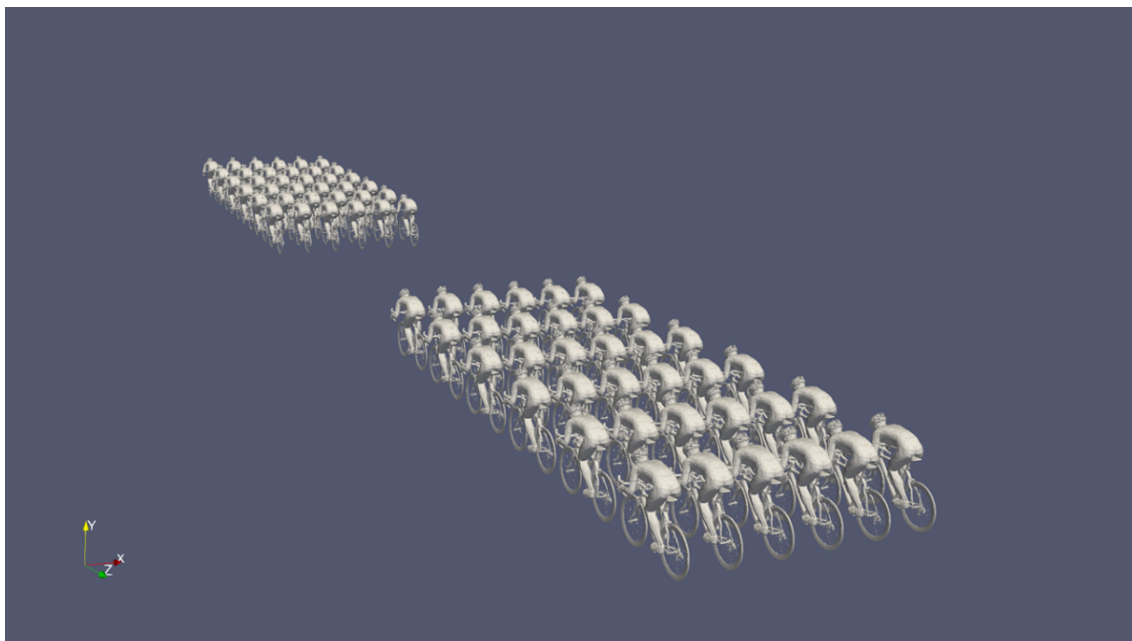


Fig. 6.26: Arrangement of the large-scale running group of 72 cyclists.

る抗力低減の効果が極めて大きいことが確認できた．なお，抗力の分布に左右非対称性が見られるが，これは 8 人の集団走行と同様で，自転車および選手の形状の非対称性から生じたものと考えられる．

6.5 本章のまとめ

細分化格子を導入した格子ボルツマン法に基づく大規模 LES 計算により，自転車競技の単体走行および集団走行の空力解析を実施した．単体走行および 4 人の集団走行の検証計算を行い，実験値と比較して本手法の妥当性を確認した．応用計算として，集団の形状や 2 集団の間の距離などの条件を変えていくつかの計算を行った．8 人の集団走行の計算では，集団全体の抗力は縦 1 列に並ぶ縦列配置がよいが，ある特定の選手の体力を温存させる目的で同選手に掛かる抗力を最小化したい場合にはひし形配置のように横に広がる配置が有効であることを示した．2 集団が前後に並んだ条件の計算では，前方の集団は後方の集団に風除けとして利用されるおそれがあること，前方の集団がひし形配置を採ると風除けの効果が増大し後方の集団に有利になってしまうことなど，いくつかの知見が得られた．上記のすべての計算は 64 GPU 以下かつ 5 日以内と実用上十分な資源で実行できた．また，72 人の集団の大規模計算を，22.3 億格子・192 GPU で行い，約 4 日で計算を完了した．このように大規模並列計算においても実用上十分な性能で計算を行うことができるため，本手法は数十人から百人規模の選手を含むペロトン集団の空力解析への応用も可能である．また，本計算は LES 計算であり本質的に非定常計算である．そのため，今後，時間的な物体の形状や位置の変動を含む計算条件として，ペダルの回転および体の揺動の解析，路面上を走行する条件での空力計算など，これまで風洞実験や RANS 解析では実施できなかったような条件で，実際の自転車競技に近い解析を行える可能性がある．これらにより，これまで主に RANS

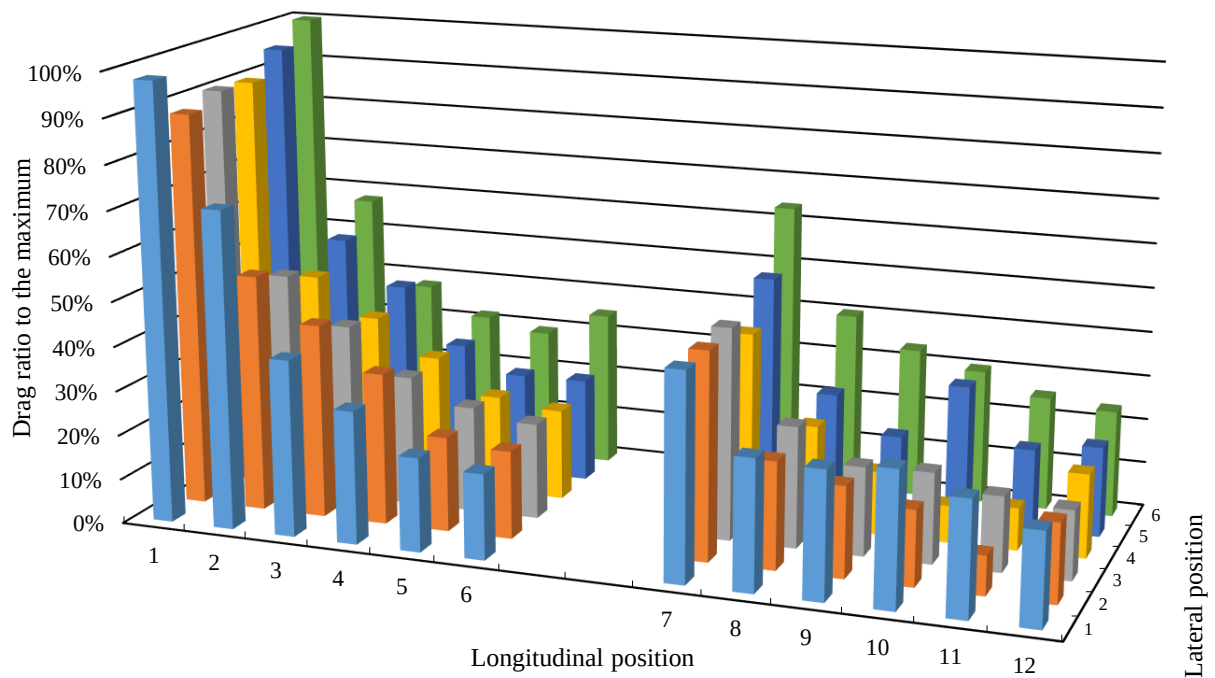


Fig. 6.27: Result of the drag area on each cyclist in the group of 72 cyclists.

で解析が行われてきた自転車競技の集団走行の解析について、LES に基づく計算としては初めてこれを実現し、本手法の実用空力解析への有用性を示した。

第 7 章

結言

本研究では、大規模空力解析アプリケーションを実現することを目的として、格子ボルツマン法および適合細分化格子法の複数 GPU 計算への実装を行った。

第 2 章では、本研究で用いる数値計算手法として適合細分化格子法を導入した格子ボルツマン法について述べるとともに、いくつかの検証計算を実施した。2 次元キャビティ流れ、3 次元球周りの流れおよび平行平板間乱流の検証計算により、本手法が単層非圧縮流れを妥当に計算できていることを確認した。

第 3 章では、格子ボルツマン法に適合細分化格子法を導入した単一 GPU 計算の実装と高速化手法について述べた。適合細分化格子のリーフに割り当てられる格子点数は等間隔直交格子に比べて少ないため、GPU で効率的な並列計算を行うことが難しい。特に、リーフの境界部分にある外殻格子は隣接参照のために他のリーフの格子を参照する必要があり、アクセスパターンが悪く高い実行性能が出ない。外殻格子点数を削減するため、Octree の 1 段上のノードに格子を割り当てる「マザーリーフ法」を提案し、外殻格子計算で 1.96 倍、全体で 1.78 倍の高速化を達成した。さらに、外殻格子計算では隣接格子へのアクセスパターンを適切に分類して実装する必要があるが、D3Q27 格子ボルツマン法による 3 次元計算の場合、リーフの外殻格子のアクセスパターンは 702 通りになり、すべてのパターンを手動で列挙して実装することはプログラムの生産性や保守性の観点から現実的でない。これまで格子ボルツマン法や適合細分化格子法には適用されていない「C++ テンプレートによる GPU カーネル関数生成」の方法により、実行性能の低い外殻格子計算を 2.29 倍に高速化し、全体でも 1.86 倍の高速化を達成した。1 種類のみのテンプレート関数を記述し、テンプレート引数を変化させた複数の関数呼び出しによりコンパイル時にアクセスパターンの判定をすることで、プログラムの生産性を維持しながら計算を高速化することができた。

第 4 章では、適合細分化格子法を導入した格子ボルツマン法の複数 GPU 計算の実装およびその計算性能について述べた。直方体形状の計算領域を取る Forest-of-octrees 構造において、アコーディオン曲線と Hilbert 曲線を組み合わせた独自の空間充填曲線を用いることで、空間局所性の高い領域分割法を導入した。また、袖領域通信の最適化を行い、最適化前に比べて MPI 通信のデータ量を 4.3% まで、および通信時間を 20.9% まで、それぞれ大幅に削減した。複数 GPU 計算の弱スケーリングは、単純な格子配置において 90% 以上の並列化効率を達成したほか、自転車のような複雑形状を含む計算条件においても 8 GPU から 128 GPU のスケーリングにおいて効率 72.8% と、実用上十分な性能であった。ただ

し、1 GPU あたりの格子点数が少なくなる強スケーリングでは十分な性能が得られておらず、本コードは GPU メモリの上限近くまで格子点数を取るような並列化に適していることが示唆された。

第5章では、応用計算のひとつとして球周りの流れにおけるドラッグ・クライシスの再現計算を行った。計算資源の問題から困難とされていた、 $Re > 10^5$ オーダーの高レイノルズ数条件で抗力が急激に低下するドラッグ・クライシスという現象について、本計算で実装した複数 GPU を用いた大規模計算により再現を試みた。格子ボルツマン法の衝突モデルとして MRT モデルおよびキュムラントモデル、ならびにそれらを組み合わせたハイブリッドモデルを用い、それぞれのモデルの計算精度と数値安定性を比較した。計算結果から、以下の知見が得られた。

- MRT モデルは、ドラッグ・クライシスの再現精度が最も良いが、 $Re > 10^6$ の高レイノルズ数条件では格子解像度が不足し、数値的に不安定になった。
- キュムラントモデルは数値安定性が高いが数値粘性が MRT モデルよりも高い傾向にあり、キュムラントモデル単体では細分化格子中の粗い格子において過剰な数値粘性がはたらいたため、現状の格子数ではドラッグ・クライシスを再現できなかった。
- キュムラントモデルと MRT モデルを組み合わせることで、キュムラントモデル単体ほど数値粘性は強くなく、かつ、MRT モデルよりも数値安定性が高いハイブリッドモデルを構成することができる。

また、複数 GPU の大規模計算により、球の直径に対して 1000 格子という極めて細かい解像度の計算を実施でき、ドラッグ・クライシスという高レイノルズ数かつレイノルズ数依存性の高い難問を解くことが可能になったことから、本手法の有用性を示すことができた。なお、本計算を行ったのち、2017 年 7 月にキュムラントモデルの数値粘性の問題を改善した 4 次精度キュムラント格子ボルツマン法が発表され [61]、同モデルによるドラッグ・クライシスの再現計算が示された [28]。4 次精度キュムラントモデルは計算精度が大幅に改善されており、MRT モデルに匹敵する精度を有している。初期のキュムラントモデルに比べると 4 次精度モデルは数値安定性が低下するものの、計算が不安定になりつつある格子点において局所的に計算を安定化させるリミターが新たに導入されており、MRT モデル以上の数値安定性を有している。このため、本研究で用いたキュムラントと MRT のハイブリッドモデルよりも、4 次精度キュムラントモデルのほうがより高い精度で計算を行える可能性がある。追計算を行い両モデルの比較を行う必要があるが、本研究では未だ 4 次精度キュムラントモデルの実装が完了しておらず、今後の課題である。

第6章では、本手法のスポーツ工学への応用事例として、細分化格子を導入した格子ボルツマン法に基づく大規模 LES 計算により、自転車競技の単体走行および集団走行の空力解析を実施した。単体走行および 4 人の集団走行の検証計算を行い、実験値と比較して本手法の妥当性を確認した。応用計算として、集団の形状や 2 集団の間の距離などの条件を変えていくつかの計算を行った。8 人の集団走行の計算では、集団全体の抗力は縦 1 列に並ぶ縦列配置がよいが、ある特定の選手の体力を温存させる目的で同選手に掛かる抗力を最小化したい場合にはひし形配置のように横に広がる配置が有効であることを示した。2 集団が前後に並んだ条件の計算では、前方の集団は後方の集団に風除けとして利用されるおそれがある

こと、前方の集団がひし形配置を採ると風除けの効果が増大し後方の集団に有利になってしまうことなど、いくつかの知見が得られた。上記のすべての計算は 64 GPU 以下かつ 5 日以内と実用上十分な資源で実行できた。また、72 人の集団の大規模計算を、22.3 億格子・192 GPU で行い、約 4 日で計算を完了した。このように大規模並列計算においても実用上十分な性能で計算を行うことができるため、本手法は数十人から百人規模の選手を含むペロトン集団の空力解析への応用も可能である。また、本計算は LES 計算であり本質的に非定常計算である。そのため、今後、時間的な物体の形状や位置の変動を含む計算条件として、ペダルの回転および体の揺動の解析、路面上を走行する条件での空力計算など、これまで風洞実験や RANS 解析では実施できなかったような条件で、実際の自転車競技に近い解析を行える可能性がある。これらにより、これまで主に RANS で解析が行われてきた自転車競技の集団走行の解析について、LES に基づく計算としては初めてこれを実現し、本手法の実用空力解析への有用性を示した。

計算機は日々発展を続けている。高性能計算 (HPC) の分野では、10 年で計算性能が 1000 倍となるムーアの法則の終焉が近いとも言われているが、進展の速度に差はあれ計算性能が今後も向上し続けることはほぼ確実である。そこで、本研究で開発した空力解析コードや計算手法について将来のスパコンにおけるスケーラビリティの予測を示す。本研究で適用した計算手法である格子ボルツマン法およびコヒーレント構造 Smagorinsky モデルに基づく LES は、完全陽解法であり、1 時刻ステップあたりの計算量は線形時間での増大に留まるから、今後の計算機の発展とともにさらなる大規模計算へとスケーリング可能であると考えられる。一方で、適合細分化格子法は木構造に基づく再帰アルゴリズムであるため、格子生成に要する時間が対数線形時間で増大するおそれがある。しかし、静的な細分化において格子生成は初期設定で行われるのみであり、また、第 5 章および第 6 章で行った応用計算において格子生成の時間は数分程度と計算時間に比べてわずかであったため、影響は小さいと予想される。計算の大規模化により、計算可能な問題のレイノルズ数が 10^7 のオーダーに到達し、自動車の空力などへの応用範囲の拡大が期待される。ただし、複数 GPU 計算のスケーラビリティについては懸念が残る。適合細分化格子法を導入した格子ボルツマン法は解像度ごとに異なる時間刻みを設定するマルチタイムステップの実装であるため、本研究のように空間充填曲線に基づいて格子点数を等しく分散した場合、計算負荷バランスが悪化するおそれがある。実際、第 4 章で示したとおり、複雑形状の細分化格子においてスケーリング性能の低下が見られた。並列化効率は、複雑形状の細分化格子において 8 GPU から 128 GPU の疑似弱スケーリングで 72.8% であった。これを単純に外挿すると、並列化効率は、1 000 GPU で 34.3%、10 000 GPU で 3.4% まで低下すると推定される。たとえば、8 GPU で 2 日程度の計算時間が掛かるような計算を大規模化させると、1 000 GPU では 5.8 日、10 000 GPU では 58 日の計算時間が必要になると見積もられる。格子点数の増大を高解像度化に充当する場合には、合わせて時間刻みも小さくなるため、計算時間の増加はさらに深刻になる。このため、計算時間を鑑みれば、本コードによる大規模空力解析はたかだか 1 000 GPU 程度の計算が実用上の限界である。現在の世界トップのスパコンが搭載する GPU 数は既に 10 000 のオーダーに達しており、今後もその数を増やし続けると予想されるため、現状のコードのまま将来のスパコンにおいて性能を得ることは絶望的である。スケーリング性能を改善し、10 000 から 100 000 GPU 級の大規模空力解析に到達するためには、空間充填曲線に替わる、計算負荷バランスの良好な領域分割法の導入が望まれる。

以上、適合細分化格子法を導入した格子ボルツマン法の複数 GPU 計算の、乱流現象たる空力解析に対する有用性、および今後の発展への期待を述べた。とくに、単一 GPU 計算において、これまで格子ボルツマン法や適合細分化格子法には適用されていない「C++ テンプレートによる GPU カーネル関数生成」の方法により実行性能の低い外殻格子計算を 2.29 倍に高速化し、格子計算全体でも 1.86 倍の高速化を達成したこと、および、複数 GPU 計算において、リーフ全体を袖領域に取る実装における袖領域通信の最適化を行い、最適化前に比べて MPI 通信のデータ量を 4.3% まで、および通信時間を 20.9% までそれぞれ大幅に削減し通信を高速化したことは、GPU を多数搭載する現代のスパコンにおいて高性能な空力解析計算を実現するという本研究の目的において特筆すべき成果である。また、 $Re \approx 10^5$ オーダーの流れである球周りの流れのドラッグ・クライシスの再現計算、および $Re \approx 10^6$ オーダーの流れである自転車競技の集団走行の空力解析を実施し、本研究で開発した計算コードが応用計算に対して実用上十分な計算速度を得たことは、実用的な大規模空力解析計算コードを開発するという本研究の目的を達したことを示す重要な計算結果である。とくに、自転車競技の集団走行の計算においては、最大で 72 人の選手の計算を 22.3 億格子という膨大な格子点数を用い、4 日という実用可能な時間で計算が完了することを示した。数十台と多数の自転車を含む集団走行の解析は、すでに RANS では行われているが、LES 計算として実現したのは本研究が初となる。これらにより、実用的な大規模 LES 空力解析が最新の GPU スパコン環境において達成可能となったことを示した。

付録 A

Chapman-Enskog 展開

本章は文献 [6] を参考に作成した．格子ボルツマン法は気体分子運動のアナロジーから導出された微視的手法である一方，流体は連続体としての場の方程式が記述される巨視的な現象であり，両者が一致するかどうかは自明でない．また，BGK モデルにおける衝突緩和係数 τ は，気体分子の微視的特徴を表す係数である一方で，(2.8) のように巨視的特徴量である動粘度を含んでいるが，その理論的根拠については言及していなかった．本章では，これらを代数的に示すため Chapman-Enskog 展開について解説する．なお，本章では式の展開を簡単にするため，空間ベクトルをボールド体 ($\mathbf{u}$) で表すかわりに添字を $i, j, k, \dots$ を用いて $u_i, u_j, u_k, \dots$ と表記する．また，添字は $u_i u_i = \sum_{i \in \{x, y, z\}} u_i u_i$ のように縮約の規則に従うものとする．

まず，クヌーセン数相当の微小パラメータ ϵ を導入し，速度分布関数を以下のように展開する．

$$f_\alpha = f_\alpha^{\text{eq}} + \epsilon f_\alpha^* + \epsilon^2 f_\alpha^{**} \quad (\text{A.1})$$

ϵ^3 以降の項は微小であるため無視する．また，速度分布関数の局所偏差である f_α^* および f_α^{**} は，巨視的量である質量・運動量の変化には寄与しないものと仮定する．すなわち，

$$\sum_{\alpha=0}^{Q-1} f_\alpha^* = \sum_{\alpha=0}^{Q-1} f_\alpha^{**} = 0 \quad (\text{A.2})$$

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} f_\alpha^* = \sum_{\alpha=0}^{Q-1} c_{\alpha i} f_\alpha^{**} = 0 \quad (\text{A.3})$$

とする．さらに，時間，空間も同様に展開する．

$$t = t^{\text{eq}} + \frac{t^*}{\epsilon} + \frac{t^{**}}{\epsilon^2} \quad (\text{A.4})$$

$$x_i = x_i^{\text{eq}} + \frac{x_i^*}{\epsilon} \quad (\text{A.5})$$

ここで， $t^{\text{eq}}, x^{\text{eq}}$ は物理量の変動が無視できる十分に大きな時間・空間長である． t^*, x_i^* は対流・圧力スケールの時間・空間長， t^{**} は拡散スケールの時間長である．拡散過程は局所性の高い現象であるため，拡散スケールの空間長は無視できる．上式の全微分から，時間および空間の微分作用素を以下のように

展開することが出来る．

$$\frac{\partial}{\partial t} = \epsilon \frac{\partial}{\partial t^*} + \epsilon^2 \frac{\partial}{\partial t^{**}} \quad (\text{A.6})$$

$$\frac{\partial}{\partial x_i} = \epsilon \frac{\partial}{\partial x_i^*} \quad (\text{A.7})$$

ただし, $t^{\text{eq}}, x_i^{\text{eq}}$ のスケールでは物理量の変動が無視できるため, $\partial/\partial t^{\text{eq}} = 0, \partial/\partial x_i^{\text{eq}} = 0$ と仮定している．

外力を無視した場合の BGK モデルによる格子ボルツマン方程式は

$$f_\alpha(\mathbf{x} + \mathbf{c}_\alpha \Delta t, t + \Delta t) = f_\alpha(\mathbf{x}, t) - \frac{1}{\tau} (f_\alpha(\mathbf{x}, t) - f_\alpha^{\text{eq}}(\mathbf{x}, t)) \quad (\text{A.8})$$

$$f_\alpha^{\text{eq}} = w_\alpha \rho \left(1 + \frac{c_{\alpha i} u_i}{c_s^2} + \frac{c_{\alpha i} u_i c_{\alpha j} u_j - c_s^2 u_i u_i}{2c_s^4} \right) \quad (\text{A.9})$$

である．(A.8) を時空間 2 次まで Taylor 展開すると, 以下が得られる．

$$\left\{ \Delta t \left(\frac{\partial}{\partial t} + c_{\alpha i} \frac{\partial}{\partial x_i} \right) + \frac{\Delta t^2}{2} \left(\frac{\partial^2}{\partial t^2} + 2c_{\alpha i} \frac{\partial^2}{\partial t \partial x_i} + c_{\alpha i} c_{\alpha j} \frac{\partial^2}{\partial x_i \partial x_j} \right) \right\} f_\alpha(\mathbf{x}, t) + \frac{1}{\tau} \{ f_\alpha(\mathbf{x}, t) - f_\alpha^{\text{eq}}(\mathbf{x}, t) \} = 0 \quad (\text{A.10})$$

ここで, 添字 i, j は空間軸の方向 ($i, j \in x, y, z$) であり, 縮約の規則に従う．以降の議論では座標および時刻が一定であるので, $(\mathbf{x}, t)$ の表記を省略する．式 (A.10) に式 (A.1)–(A.7) を代入すると,

$$\left[\Delta t \left(\epsilon \frac{\partial}{\partial t^*} + \epsilon^2 \frac{\partial}{\partial t^{**}} + \epsilon c_{\alpha i} \frac{\partial}{\partial x_i^*} \right) + \frac{\Delta t^2}{2} \left\{ \epsilon \frac{\partial}{\partial t^*} + \epsilon^2 \frac{\partial}{\partial t^{**}} + 2c_{\alpha i} \left(\epsilon \frac{\partial}{\partial t^*} + \epsilon^2 \frac{\partial}{\partial t^{**}} \right) \epsilon \frac{\partial}{\partial x_i^*} + c_{\alpha i} c_{\alpha j} \epsilon^2 \frac{\partial^2}{\partial x_i^* \partial x_j^*} \right\} \right] (f_\alpha^{\text{eq}} + \epsilon f_\alpha^* + \epsilon^2 f_\alpha^{**}) + \frac{1}{\tau} (\epsilon f_\alpha^* + \epsilon^2 f_\alpha^{**}) = 0 \quad (\text{A.11})$$

となる． ϵ の次数で整理し ϵ^3 以上の項を無視することで, 以下の式を得る．

$$\epsilon E_\alpha^* + \epsilon^2 E_\alpha^{**} = 0, \quad (\text{A.12})$$

$$E_\alpha^* = \Delta t \left(\frac{\partial}{\partial t^*} + c_{\alpha i} \frac{\partial}{\partial x_i^*} \right) f_\alpha^{\text{eq}} + \frac{1}{\tau} f_\alpha^*, \quad (\text{A.13})$$

$$E_\alpha^{**} = \left\{ \frac{\Delta t^2}{2} \left(\frac{\partial^2}{\partial t^{*2}} + 2c_{\alpha i} \frac{\partial^2}{\partial t^* \partial x_i^*} + c_{\alpha i} c_{\alpha j} \frac{\partial^2}{\partial x_i^* \partial x_j^*} \right) + \Delta t \frac{\partial}{\partial t^{**}} \right\} f_\alpha^{\text{eq}} + \Delta t \left(\frac{\partial}{\partial t^*} + c_{\alpha i} \frac{\partial}{\partial x_i^*} \right) f_\alpha^* + \frac{1}{\tau} f_\alpha^{**} \quad (\text{A.14})$$

任意の微小パラメータ ϵ に対して式 (A.12) が成り立つためには, $E_\alpha^* = E_\alpha^{**} = 0$ が要請される．

連続の式の導出

ϵ の 1 次の項から、連続の式を導出する．式 (A.13) において、全速度方向の和を取ると、

$$\sum_{\alpha=0}^{Q-1} E_{\alpha}^* = \Delta t \frac{\partial}{\partial t^*} \sum_{\alpha=0}^{Q-1} f_{\alpha}^{\text{eq}} + \Delta t \frac{\partial}{\partial x_i^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} f_{\alpha}^{\text{eq}} + \frac{1}{\tau} \sum_{\alpha=0}^{Q-1} f_{\alpha}^* = 0 \quad (\text{A.15})$$

が得られる．中辺第 1 項および第 2 項の総和は、格子ボルツマン法における流体の密度および運動量の定義と同一の項になっている．また、中辺第 3 項は (A.2) より 0 に等しい．したがって、式を整理すると、

$$\frac{\partial \rho}{\partial t^*} + \frac{\partial \rho u_i}{\partial x_i^*} = 0 \quad (\text{A.16})$$

となり、連続の式が導かれた．

非圧縮 Euler 方程式の導出

次に、 ϵ の 1 次の項から非圧縮 Euler 方程式、すなわち Navier-Stokes 方程式から粘性項を取り除いた式を導出する．式 (A.13) に速度 $c_{\alpha j}$ を掛け、全速度方向の和を取ると、

$$\sum_{\alpha=0}^{Q-1} c_{\alpha j} E_{\alpha}^* = \Delta t \frac{\partial}{\partial t^*} \sum_{\alpha=0}^{Q-1} c_{\alpha j} f_{\alpha}^{\text{eq}} + \Delta t \frac{\partial}{\partial x_i^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} f_{\alpha}^{\text{eq}} + \frac{1}{\tau} \sum_{\alpha=0}^{Q-1} c_{\alpha j} f_{\alpha}^* = 0 \quad (\text{A.17})$$

が得られる．中辺第 1 項の総和は流体の運動量の定義そのものである．中辺第 2 項の総和は、 f_{α}^{eq} に式 (A.9) を代入することで、以下のように展開できる．

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} f_{\alpha}^{\text{eq}} = \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} w_{\alpha} \rho \left(1 + \frac{c_{\alpha k} u_k}{c_s^2} + \frac{c_{\alpha k} u_k c_{\alpha l} u_l - c_s^2 u_k u_k}{2c_s^4} \right) \quad (\text{A.18})$$

$$= \rho \left(1 - \frac{u_k u_k}{2c_s^2} \right) \sum_{\alpha=0}^{Q-1} w_{\alpha} c_{\alpha i} c_{\alpha j} + \rho \frac{u_k}{c_s^2} \sum_{\alpha=0}^{Q-1} w_{\alpha} c_{\alpha i} c_{\alpha j} c_{\alpha k} + \rho \frac{u_k u_l}{2c_s^4} \sum_{\alpha=0}^{Q-1} w_{\alpha} c_{\alpha i} c_{\alpha j} c_{\alpha k} c_{\alpha l} \quad (\text{A.19})$$

$$= \rho \left(1 - \frac{u_k u_k}{2c_s^2} \right) c_s^2 \delta_{ij} + 0 + \rho \frac{u_k u_l}{2c_s^4} c_s^4 (\delta_{ij} \delta_{kl} + \delta_{ik} \delta_{jl} + \delta_{il} \delta_{jk}) \quad (\text{A.20})$$

$$= \rho u_i u_j + \rho c_s^2 \delta_{ij} \quad (\text{A.21})$$

$$= \rho u_i u_j + p \delta_{ij} \quad (\text{A.22})$$

ただし、 δ_{ij} はクロネッカーのデルタである．式 (A.17) の中辺第 3 項は、式 (A.3) より 0 である．したがって式 (A.17) は以下のように変形できる．

$$\frac{\partial \rho u_j}{\partial t^*} + \frac{\partial \rho u_i u_j}{\partial x_i^*} + \frac{\partial p}{\partial x_j^*} = 0 \quad (\text{A.23})$$

また、上式を更に展開すると、

$$\rho \frac{\partial u_j}{\partial t^*} + u_j \frac{\partial \rho}{\partial t^*} + \rho u_i \frac{\partial u_j}{\partial x_i^*} + u_j \frac{\partial \rho u_i}{\partial x_i^*} + \frac{\partial p}{\partial x_j^*} = 0 \quad (\text{A.24})$$

$$\rho \left(\frac{\partial u_j}{\partial t^*} + u_i \frac{\partial u_j}{\partial x_i^*} + \frac{1}{\rho} \frac{\partial p}{\partial x_j^*} \right) + u_j \left(\frac{\partial \rho}{\partial t^*} + \frac{\partial \rho u_i}{\partial x_i^*} \right) = 0 \quad (\text{A.25})$$

となる．連続の式より左辺第 2 項は 0 に等しいため、式を整理すると、

$$\frac{\partial u_j}{\partial t^*} + u_i \frac{\partial u_j}{\partial x_i^*} = -\frac{1}{\rho} \frac{\partial p}{\partial x_j^*} \quad (\text{A.26})$$

のように、非圧縮 Euler 方程式が得られる．

非圧縮 Navier-Stokes 方程式の導出

さらに、 ϵ の 2 次の項を考慮することで、粘性項を含む Navier-Stokes 方程式を導出することができる．式 (A.14) に速度 $c_{\alpha k}$ を掛け、全速度方向の和を取ると、

$$\begin{aligned} \sum_{\alpha=0}^{Q-1} c_{\alpha k} E_{\alpha}^{**} &= \frac{\Delta t^2}{2} \frac{\partial^2}{\partial t^{*2}} \sum_{\alpha=0}^{Q-1} c_{\alpha k} f_{\alpha}^{\text{eq}} + \Delta t^2 \frac{\partial^2}{\partial t^* \partial x_i^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha k} f_{\alpha}^{\text{eq}} + \frac{\Delta t^2}{2} \frac{\partial^2}{\partial x_i^* \partial x_j^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} c_{\alpha k} f_{\alpha}^{\text{eq}} \\ &\quad + \Delta t \frac{\partial}{\partial t^{**}} \sum_{\alpha=0}^{Q-1} c_{\alpha k} f_{\alpha}^{\text{eq}} + \Delta t \frac{\partial}{\partial t^*} \sum_{\alpha=0}^{Q-1} c_{\alpha k} f_{\alpha}^* + \Delta t \frac{\partial}{\partial x_i^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha k} f_{\alpha}^* + \frac{1}{\tau} \sum_{\alpha=0}^{Q-1} c_{\alpha k} f_{\alpha}^{**} \\ &= 0 \end{aligned} \quad (\text{A.27})$$

が得られる．中辺の項を可能な限り巨視的量に置き換え、式を整理すると、

$$\begin{aligned} \frac{\Delta t}{2} \frac{\partial^2 \rho u_k}{\partial t^{*2}} + \Delta t \frac{\partial^2 (\rho u_i u_k + p \delta_{ik})}{\partial t^* \partial x_i^*} + \frac{\Delta t}{2} \frac{\partial^2 \rho c_s^2 (u_i \delta_{jk} + u_j \delta_{ki} + u_k \delta_{ij})}{\partial x_i^* \partial x_j^*} + \frac{\partial \rho u_k}{\partial t^{**}} \\ + \frac{\partial}{\partial x_i^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha k} f_{\alpha}^* = 0 \end{aligned} \quad (\text{A.28})$$

となる．残りの $\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha k} f_{\alpha}^*$ は、以下の手順により求める．まず、式 (A.13)=0 より、

$$f_{\alpha}^* = -\tau \Delta t \left(\frac{\partial}{\partial t^*} + c_{\alpha j} \frac{\partial}{\partial x_j^*} \right) f_{\alpha}^{\text{eq}} \quad (\text{A.29})$$

である．上式にさらに $c_{\alpha i} c_{\alpha k}$ を掛け、全速度方向の総和を取る．

$$\begin{aligned} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha k} f_{\alpha}^* &= -\tau \Delta t \frac{\partial}{\partial t^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha k} f_{\alpha}^{\text{eq}} - \tau \Delta t \frac{\partial}{\partial x_j^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha k} c_{\alpha j} f_{\alpha}^{\text{eq}} \\ &= -\tau \Delta t \frac{\partial (\rho u_i u_k + p \delta_{ik})}{\partial t^*} - \tau \Delta t \frac{\partial \rho c_s^2 (u_i \delta_{jk} + u_j \delta_{ki} + u_k \delta_{ij})}{\partial x_j^*} \end{aligned} \quad (\text{A.30})$$

式 (A.28) に (A.30) を代入すると,

$$\frac{\Delta t}{2} \frac{\partial^2 \rho u_k}{\partial t^{*2}} + \Delta t (1 - \tau) \frac{\partial^2 (\rho u_i u_k + p \delta_{ik})}{\partial t^* \partial x_i^*} + \Delta t \left(\frac{1}{2} - \tau \right) \frac{\partial^2 \rho c_s^2 (u_i \delta_{jk} + u_j \delta_{ki} + u_k \delta_{ij})}{\partial x_i^* \partial x_j^*} + \frac{\partial \rho u_k}{\partial t^{**}} = 0 \quad (\text{A.31})$$

左辺第 1 項および第 3 項の微分項は以下のように変形できる.

$$\frac{\partial^2 \rho u_k}{\partial t^{*2}} = - \frac{\partial^2 (\rho u_i u_k + p \delta_{ik})}{\partial t^* \partial x_i^*} \quad (\text{A.32})$$

$$\begin{aligned} \frac{\partial^2 \rho c_s^2 (u_i \delta_{jk} + u_j \delta_{ki} + u_k \delta_{ij})}{\partial x_i^* \partial x_j^*} &= \frac{\partial^2 \rho c_s^2 u_i}{\partial x_i^* \partial x_k^*} + \frac{\partial^2 \rho c_s^2 u_j}{\partial x_k^* \partial x_j^*} + \frac{\partial^2 \rho c_s^2 u_k}{\partial x_i^{*2}} \\ &= c_s^2 \left(2 \frac{\partial^2 \rho u_i}{\partial x_i^* \partial x_k^*} + \frac{\partial^2 \rho u_k}{\partial x_i^{*2}} \right) \end{aligned} \quad (\text{A.33})$$

これらを式 (A.31) に代入すると,

$$\Delta t \left(\frac{1}{2} - \tau \right) \frac{\partial^2 (\rho u_i u_k + p \delta_{ik})}{\partial t^* \partial x_i^*} + c_s^2 \Delta t \left(\frac{1}{2} - \tau \right) \left(2 \frac{\partial^2 \rho u_i}{\partial x_i^* \partial x_k^*} + \frac{\partial^2 \rho u_k}{\partial x_i^{*2}} \right) + p d \rho u_k t^{**} = 0 \quad (\text{A.34})$$

が得られる. さらに左辺第 1 項は, $\partial \rho u_i u_k / \partial t^* \approx 0$ と近似し, 式 (2.16), (A.23) を代入すると, 以下のように変形される.

$$\begin{aligned} \Delta t \left(\frac{1}{2} - \tau \right) \frac{\partial^2 (\rho u_i u_k + p \delta_{ik})}{\partial t^* \partial x_i^*} &= \Delta t \left(\frac{1}{2} - \tau \right) \frac{\partial^2 \rho c_s^2}{\partial t^* \partial x_k^*} \\ &= -c_s^2 \Delta t \left(\frac{1}{2} - \tau \right) \frac{\partial^2 \rho u_i}{\partial x_i^* \partial x_k^*} \end{aligned} \quad (\text{A.35})$$

これを式 (A.34) に代入すれば,

$$c_s^2 \Delta t \left(\frac{1}{2} - \tau \right) \left(\frac{\partial^2 \rho u_i}{\partial x_i^* \partial x_k^*} + \frac{\partial^2 \rho u_k}{\partial x_i^{*2}} \right) + \frac{\partial \rho u_k}{\partial t^{**}} = 0 \quad (\text{A.36})$$

となる. 非圧縮近似 ($\rho = \text{const.}$) を適用し, 連続の式を代入すると,

$$\frac{\partial u_k}{\partial t^{**}} = \nu \frac{\partial^2 u_k}{\partial x_i^{*2}}, \quad (\text{A.37})$$

$$\nu \equiv c_s^2 \Delta t \left(\tau - \frac{1}{2} \right) \quad (\text{A.38})$$

が得られる. ここで, ν は流体の動粘度である. 最後に, 式 (A.26) に ϵ を掛けたものと式 (A.37) に ϵ^2 を掛けたものを足し合わせることで,

$$\left(\epsilon \frac{\partial}{\partial t^*} + \epsilon^2 \frac{\partial}{\partial t^{**}} \right) u_k + u_i \epsilon \frac{\partial u_k}{\partial x_i^*} = -\frac{1}{\rho} \left(\epsilon \frac{\partial p}{\partial x_k^*} \right) + \nu \epsilon^2 \frac{\partial^2 u_k}{\partial x_i^{*2}} \quad (\text{A.39})$$

$$\frac{\partial u_k}{\partial t} + u_i \frac{\partial u_k}{\partial x_i} = -\frac{1}{\rho} \frac{\partial p}{\partial x_k} + \nu \frac{\partial^2 u_k}{\partial x_i^2} \quad (\text{A.40})$$

と、非圧縮性 Navier-Stokes 方程式が導出される。以上により、格子ボルツマン法が非圧縮性流体の巨視的方程式を代数的に再現することが示された。

ひずみ速度テンソルの導出

ひずみ速度テンソルとは、次の式で定義される 2 階テンソルである。

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \quad (\text{A.41})$$

Large eddy simulation (LES) の計算では、ひずみ速度テンソルの大きさを数値的に求める必要がある。ひずみ速度テンソルを数値的に求めるには、本来、空間差分の計算が必要であるが、格子ボルツマン法の場合には速度分布関数の値からひずみ速度テンソルを直接求めることができる。Chapman-Enskog 展開を用いて、ひずみ速度テンソルと速度分布関数の関係を導出する。

式 (A.13) より、

$$f_\alpha^* = -\tau \Delta t \left(\frac{\partial f_\alpha^{\text{eq}}}{\partial t^*} + c_{\alpha k} \frac{\partial f_\alpha^{\text{eq}}}{\partial x_k^*} \right) \quad (\text{A.42})$$

である。両辺に $c_{\alpha i} c_{\alpha j}$ を掛けて全速度方向の和を取ると、

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} f_\alpha^* = -\tau \Delta t \left(\frac{\partial}{\partial t^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} f_\alpha^{\text{eq}} + \frac{\partial}{\partial x_k^*} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} c_{\alpha k} f_\alpha^{\text{eq}} \right) \quad (\text{A.43})$$

$$= -\tau \Delta t \left(\frac{\partial}{\partial t^*} (\rho u_i u_j + \rho c_s^2 \delta_{ij}) + \frac{\partial}{\partial x_k^*} \{ \rho c_s^2 (u_k \delta_{ij} + u_j \delta_{ik} + u_i \delta_{jk}) \} \right) \quad (\text{A.44})$$

$$= -\tau \Delta t \left(\frac{\partial}{\partial t^*} (\rho u_i u_j + \rho c_s^2 \delta_{ij}) + c_s^2 \left(\frac{\partial \rho u_k}{\partial x_k^*} \delta_{ij} + \frac{\partial \rho u_j}{\partial x_i^*} + \frac{\partial \rho u_i}{\partial x_j^*} \right) \right) \quad (\text{A.45})$$

$$= -\tau \Delta t \left(\frac{\partial \rho u_i u_j}{\partial t^*} + c_s^2 \left(\frac{\partial \rho}{\partial t^*} + \frac{\partial \rho u_k}{\partial x_k^*} \right) \delta_{ij} + c_s^2 \left(\frac{\partial \rho u_j}{\partial x_i^*} + \frac{\partial \rho u_i}{\partial x_j^*} \right) \right) \quad (\text{A.46})$$

となる。ここで、Navier-Stokes 方程式の導出過程と同様に、 $\partial \rho u_i u_j / \partial t^* \approx 0$ と近似する。また、右辺第 2 項は連続の式により 0 となる。結局右辺には第 3 項のみが残る、以下のような関係が得られる。

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} f_\alpha^* = -c_s^2 \tau \Delta t \left(\frac{\partial \rho u_j}{\partial x_i^*} + \frac{\partial \rho u_i}{\partial x_j^*} \right) \quad (\text{A.47})$$

さらに非圧縮近似 ($\rho = \text{const.}$) を適用することで、

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} f_\alpha^* = -\rho c_s^2 \tau \Delta t \left(\frac{\partial u_j}{\partial x_i^*} + \frac{\partial u_i}{\partial x_j^*} \right) \quad (\text{A.48})$$

を得る。両辺に微小パラメータ ϵ を掛け、(A.1) および (A.7) を代入すると、

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} (f_\alpha - f_\alpha^{\text{eq}} - \epsilon^2 f_\alpha^{**}) = -\rho c_s^2 \tau \Delta t \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \quad (\text{A.49})$$

$$= -2\rho c_s^2 \tau \Delta t S_{ij} \quad (\text{A.50})$$

となる．最後に， $O(\epsilon^2)$ の項を無視することで，

$$\sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} (f_{\alpha} - f_{\alpha}^{\text{eq}}) = -2\rho c_s^2 \tau \Delta t S_{ij} \quad (\text{A.51})$$

または，

$$S_{ij} = -\frac{1}{2\rho c_s^2 \tau \Delta t} \sum_{\alpha=0}^{Q-1} c_{\alpha i} c_{\alpha j} (f_{\alpha} - f_{\alpha}^{\text{eq}}) \quad (\text{A.52})$$

として，速度分布関数とひずみ速度テンソルの関係が得られた．

参考文献

- [1] 総合エネルギー統計 | 各種統計情報 (需給関連) | 資源エネルギー庁, http://www.enecho.meti.go.jp/statistics/total_energy/. (Accessed on Dec. 26, 2018).
- [2] C. R. Kyle and E. Burke: Improving the racing bicycle, *Mechanical Engineering*, Vol. 106, pp. 34–35 (1984).
- [3] ANSYS Fluent Software: CFD Simulation, <https://www.ansys.com/products/fluids/ansys-fluent>. (Accessed on Dec. 25, 2018).
- [4] T. Abe: Derivation of the Lattice Boltzmann Method by Means of the Discrete Ordinate Method for the Boltzmann Equation, *Journal of Computational Physics*, Vol. 131, pp. 241–246 (online), doi:10.1006/jcph.1996.5595 (1997).
- [5] 蔦原道久: 格子ボルツマン法の応用と基礎, 日本機械学会論文集 (B 編), Vol. 77, No. 784, pp. 149–160 (2011).
- [6] D. A. Wolf-Gladrow: *Lattice-gas cellular automata and lattice Boltzmann models: An Introduction*, Springer (2000).
- [7] K. Suga, Y. Kuwata, K. Takashima and R. Chikasue: A D3Q27 multiple-relaxation-time lattice Boltzmann method for turbulent flows, *Computers and Mathematics with Applications*, Vol. 69, No. 6, pp. 518–529 (online), doi:10.1016/j.camwa.2015.01.010 (2015).
- [8] 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, Vol. 2013, pp. 123–131 (2013).
- [9] N. Onodera, Y. Idomura, Y. Ali and T. Shimokawabe: Communication Reduced Multi-time-step Algorithm for Real-time Wind Simulation on GPU-based Supercomputers, *2018 IEEE/ACM 9th Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)*, pp. 9–16 (2018).
- [10] A. Randles, E. W. Draeger, T. Oppelstrup, L. Krauss and J. A. Gunnels: Massively parallel models of the human circulatory system, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15*, pp. 1–11 (online), doi:10.1145/2807591.2807676 (2015).
- [11] PowerFLOW: CFD Software Solution for Aerodynamic Design — Exa Corporation, <https://www.exa.com/powerflow/>.

- [//exa.com/en/product/simulation-tools/powerflow-cfd-simulation](http://exa.com/en/product/simulation-tools/powerflow-cfd-simulation). (Accessed on Dec. 25, 2018).
- [12] XFlow, <https://www.3ds.com/ja/products-services/simulia/products/xflow/>. (Accessed on Dec. 26, 2018).
- [13] ultraFluidX — FluiDyna GmbH, <http://www.fluidyna.com/content/ultrafluidx>. (Accessed on Dec. 27, 2018).
- [14] M. J. Berger and J. Oliger: Adaptive mesh refinement for hyperbolic partial differential equations, *Journal of Computational Physics*, Vol. 53, No. 3, pp. 484–512 (online), doi:10.1016/0021-9991(84)90073-1 (1984).
- [15] M. Berger and P. Colella: Local adaptive mesh refinement for shock hydrodynamics, *Journal of Computational Physics*, Vol. 82, No. 1, pp. 64–84 (online), doi:10.1016/0021-9991(89)90035-1 (1989).
- [16] T. Tu, D. R. O’Hallaron and O. Ghattas: Scalable parallel octree meshing for terascale applications, *Proceedings of the ACM/IEEE 2005 Supercomputing Conference, SC’05*, Vol. 2005, No. 1-59593-061-061-2/05, pp. 1–15 (online), doi:10.1109/SC.2005.61 (2005).
- [17] C. Burstedde, O. Ghattas, M. Gurnis, T. Isaac, G. Stadler, T. Warburton and L. Wilcox: Extreme-Scale AMR, *2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, No. November, pp. 1–12 (online), doi:10.1109/SC.2010.25 (2010).
- [18] CUDA Zone — NVIDIA Developer, <https://developer.nvidia.com/cuda-zone>. (Accessed on Dec. 28, 2018).
- [19] TOP500 Supercomputer Sites, <https://www.top500.org/>. (Accessed on Dec. 27, 2018).
- [20] O. Filippova and D. Hänel: Grid Refinement for Lattice-BGK Models, *Journal of Computational Physics*, Vol. 147, pp. 219–228 (1998).
- [21] D. Yu, R. Mei and W. Shyy: A multi-block lattice Boltzmann method for viscous fluid flows, *International Journal for Numerical Methods in Fluids*, Vol. 39, No. 2, pp. 99–120 (online), doi:10.1002/fld.280 (2002).
- [22] M. Rohde, D. Kandhai, J. J. Derksen and H. E. A. van den Akker: A generic, mass conservative local grid refinement technique for lattice-Boltzmann schemes, *International Journal for Numerical Methods in Fluids*, Vol. 51, No. 4, pp. 439–468 (online), doi:10.1002/fld.1140 (2006).
- [23] Y. Kuwata and K. Suga: Imbalance-correction grid-refinement method for lattice Boltzmann flow simulations, *Journal of Computational Physics*, Vol. 311, pp. 348–362 (online), doi:10.1016/j.jcp.2016.02.008 (2016).
- [24] W. Xian and T. Aoki: Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster, *Parallel Computing*, Vol. 37, No. 9, pp. 521–535 (online), doi:10.1016/j.parco.2011.02.007 (2011).

-
- [25] H.-Y. Schive, Y.-C. Tsai and T. Chiueh: GAMER : A GRAPHIC PROCESSING UNIT ACCELERATED ADAPTIVE-MESH-REFINEMENT CODE FOR ASTROPHYSICS, *The Astrophysical Journal Supplement Series*, Vol. 186, No. 2, pp. 457–484 (online), doi:10.1088/0067-0049/186/2/457 (2010).
 - [26] AMReX-Codes GitHub, <https://github.com/AMReX-Codes>. (Accessed on 01/23/2019).
 - [27] M. N. Farooqi, T. Nguyen, W. Zhang, A. S. Almgren, J. Shalf and D. Unat: Phase Asynchronous AMR Execution for Productive and Performant Astrophysical Flows, *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, SC '18, Piscataway, NJ, USA, IEEE Press, pp. 70:1–70:14 (online), <http://dl.acm.org/citation.cfm?id=3291656.3291750> (2018).
 - [28] M. Geier, A. Pasquali and M. 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 (online), doi:10.1016/j.jcp.2017.07.004 (2017).
 - [29] 渡辺勢也 : 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, 博士学位論文, 東京工業大学 (2019).
 - [30] Y. Kuwata and K. Suga: Anomaly of the lattice Boltzmann methods in three-dimensional cylindrical flows, *Journal of Computational Physics*, Vol. 280, pp. 563–569 (online), doi:10.1016/j.jcp.2014.10.002 (2015).
 - [31] 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 Review*, Vol. 94, No. 3, pp. 511–525 (online), doi:10.1103/PhysRev.94.511 (1954).
 - [32] P. Lallemand and L.-S. Luo: Theory of the lattice Boltzmann Method: Dispersion, Dissipation, Isotropy, Galilean Invariance, and Stability, *Physical Review E*, Vol. 61, No. 6, pp. 6546–6562 (online), doi:10.1103/PhysRevE.61.6546 (2000).
 - [33] D. D’Humières, I. Ginzburg, M. Krafczyk, P. Lallemand and L.-S. Luo: Multiple-relaxation-time lattice Boltzmann models in three dimensions., *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, Vol. 360, No. 1792, pp. 437–451 (online), doi:10.1098/rsta.2001.0955 (2002).
 - [34] M. Geier, M. Schönherr, A. Pasquali and M. Krafczyk: The cumulant lattice Boltzmann equation in three dimensions: Theory and validation, *Computers and Mathematics with Applications*, Vol. 70, No. 4, pp. 507–547 (online), doi:10.1016/j.camwa.2015.05.001 (2015).
 - [35] D. Yu, R. Mei, L.-S. Luo and W. Shyy: Viscous flow computations with the method of Lattice Boltzmann Equation, *Progress in Aerospace Sciences*, Vol. 39, No. 5, pp. 329–338 (online), doi:10.1016/S0376-0421(03)00003-4 (2003).
 - [36] M. Bouzidi, M. Firdaouss and P. Lallemand: Momentum transfer of a Boltzmann-

- lattice fluid with boundaries, *Physics of Fluids*, Vol. 13, No. 11, pp. 3452–3459 (online), doi:10.1063/1.1399290 (2001).
- [37] M. Renwei, Y. Dazhi and W. Shyy: Force Evaluation Curved in the Lattice Geometry Boltzmann Method Involving, *ICASE Report*, No. 2002-22 (2002).
- [38] J. Smagorinsky: General circulation experiments with the primitive equations. Part I: The basic experiment, *Monthly Weather Review*, Vol. 91, No. 3, pp. 99–164 (online), doi:10.1126/science.27.693.594 (1963).
- [39] H. Kobayashi: The subgrid-scale models based on coherent structures for rotating homogeneous turbulence and turbulent channel flow, *Physics of Fluids*, Vol. 17, No. 4, p. 045104 (online), doi:10.1063/1.1874212 (2005).
- [40] C. Burstedde, L. C. Wilcox and O. 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 (online), doi:10.1137/100791634 (2011).
- [41] U. Ghia, K. N. Ghia and C. T. 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 (online), doi:10.1016/0021-9991(82)90058-4 (1982).
- [42] J. Almedeij: Drag coefficient of flow around a sphere: Matching asymptotically the wide trend, *Powder Technology*, Vol. 186, No. 3, pp. 218–223 (online), doi:10.1016/j.powtec.2007.12.006 (2008).
- [43] J. Kim, P. Moin and R. Moser: Turbulence statistics in fully developed channel, *Journal of Fluid Mechanics*, Vol. 177, pp. 133–166 (online), doi:10.1017/S0022112087000892 (1987).
- [44] C. Crassin, F. Neyret, S. Lefebvre, E. Eisemann and I. Sophia-antipolis: GigaVoxels : Ray-Guided Streaming for Efficient and Detailed Voxel Rendering, *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, Vol. 1, No. 212, pp. 15–22 (online), doi:10.1145/1507149.1507152 (2009).
- [45] M. Harris: High Performance Computing with CUDA; Optimizing CUDA, *Tutorial of 2007 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, SC '07 (2007).
- [46] S. Williams, A. Waterman and D. Patterson: Roofline: An Insightful Visual Performance Model for Multicore Architectures, *Commun. ACM*, Vol. 52, No. 4, pp. 65–76 (online), doi:10.1145/1498765.1498785 (2009).
- [47] T. Shimokawabe, T. Aoki, C. Muroi, J. Ishida, K. Kawano, T. Endo, A. Nukada, N. Maruyama and S. Matsuoka: An 80-fold speedup, 15.0 TFlops GPU acceleration of non-hydrostatic weather model ASUCA production code (2010).
- [48] S. Aluru and F. E. Sevilgen: Parallel Domain Decomposition and Load Balancing Using Space-Filling Curves, *Proceedings Fourth International Conference on High-Performance Computing*,

- pp. 230–235 (online), doi:10.1109/HIPC.1997.634498 (1997).
- [49] G. M. Morton: *A computer oriented geodetic data base and a new technique in the file sequencing*, IBM Ltd., Ottawa, Canada (1966).
 - [50] D. Hilbert: Ueber die stetige Abbildung einer Linie auf ein Flaechenstueck, *Mathematischen Annalen*, Vol. 38, No. 3, pp. 459–460 (online), doi:10.1007/BF01199431 (1891).
 - [51] J. Rudi, O. Ghattas, A. C. I. Malossi, T. Isaac, G. Stadler, M. Gurnis, P. W. J. Staar, Y. Ineichen, C. Bekas and A. Curioni: An extreme-scale implicit solver for complex PDEs, *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC '15*, pp. 1–12 (online), doi:10.1145/2807591.2807675 (2015).
 - [52] M. Mirzadeh, A. Guittet, C. Burstedde and F. Gibou: Parallel level-set methods on adaptive tree-based grids, *Journal of Computational Physics*, Vol. 322, pp. 345–364 (online), doi:10.1016/j.jcp.2016.06.017 (2016).
 - [53] H. Sagan: *Space-Filling Curves*, Springer New York (1994).
 - [54] 坂東幸浩, 西修功, 鎌田清一郎 : 3次元擬似ヒルベルト走査法と動画像圧縮への応用, 電子情報通信学会技術研究報告. MVE, マルチメディア・仮想環境基礎, Vol. 98, No. 208, pp. 31–38 (オンライン), <https://ci.nii.ac.jp/naid/110003270685/> (1998).
 - [55] J. Zhang and S.-i. Kamata: A generalized 3-D Hilbert scan using look-up tables, *Journal of Visual Communication and Image Representation*, Vol. 23, No. 3, pp. 418–425 (online), doi:10.1016/j.jvcir.2011.12.005 (2012).
 - [56] Open MPI: Open Source High Performance Computing, <https://www.open-mpi.org/>. (Accessed on Dec. 25, 2018).
 - [57] G. Eiffel: *Nouvelles recherches sur la résistance de l'air et l'aviation faites au laboratoire d'Auteuil*, Dunod & Pinat (1914).
 - [58] M. Muto, M. Tsubokura and N. Oshima: Negative Magnus lift on a rotating sphere at around the critical Reynolds number, *Physics of Fluids*, Vol. 24, No. 1 (online), doi:10.1063/1.3673571 (2012).
 - [59] H. Schlichting: *Boundary-Layer Theory*, McGRAW-HILL BOOK COMPANY (1955).
 - [60] P. R. Spalart: Detached-Eddy Simulation, *Annual Review of Fluid Mechanics*, Vol. 41, No. 1, pp. 181–202 (online), doi:10.1146/annurev.fluid.010908.165130 (2009).
 - [61] M. Geier, A. Pasquali and M. 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 (online), doi:10.1016/j.jcp.2017.05.040 (2017).
 - [62] N. Barry, J. Sheridan, D. Burton and N. A. Brown: The effect of spatial position on the aerodynamic interactions between cyclists, *Procedia Engineering*, Vol. 72, pp. 774–779 (online), doi:10.1016/j.proeng.2014.06.131 (2014).
 - [63] N. Barry, D. Burton, J. Sheridan, M. Thompson and N. A. Brown: Aerodynamic performance

- and riding posture in road cycling and triathlon, *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, Vol. 229, No. 1, pp. 28–38 (online), doi:10.1177/1754337114549876 (2015).
- [64] T. Defraeye, B. Blocken, E. Koninckx, P. Hespel and J. Carmeliet: Computational fluid dynamics analysis of drag and convective heat transfer of individual body segments for different cyclist positions, *Journal of Biomechanics*, Vol. 44, No. 9, pp. 1695–1701 (online), doi:10.1016/j.jbiomech.2011.03.035 (2011).
- [65] M. D. Griffith, T. Crouch, M. C. Thompson, D. Burton, J. Sheridan and N. A. T. Brown: Computational Fluid Dynamics Study of the Effect of Leg Position on Cyclist Aerodynamic Drag, *Journal of Fluids Engineering*, Vol. 136, No. 10, p. 101105 (online), doi:10.1115/1.4027428 (2014).
- [66] T. Defraeye, B. Blocken, E. Koninckx, P. Hespel, P. Verboven, B. Nicolai and J. Carmeliet: Cyclist Drag in Team Pursuit: Influence of Cyclist Sequence, Stature, and Arm Spacing, *Journal of Biomechanical Engineering*, Vol. 136, No. 1, p. 011005 (online), doi:10.1115/1.4025792 (2013).
- [67] B. Blocken, T. van Druenen, Y. Toparlar, F. Malizia, P. Mannion, T. Andrianne, T. Marchal, G.-J. Maas and J. Diepens: Aerodynamic drag in cycling pelotons: New insights by CFD simulation and wind tunnel testing, *Journal of Wind Engineering and Industrial Aerodynamics*, Vol. 179, pp. 319–337 (online), doi:10.1016/J.JWEIA.2018.06.011 (2018).
- [68] T. Defraeye, B. Blocken, E. Koninckx, P. Hespel and J. Carmeliet: Computational fluid dynamics analysis of cyclist aerodynamics: Performance of different turbulence-modelling and boundary-layer modelling approaches, *Journal of Biomechanics*, Vol. 43, No. 12, pp. 2281–2287 (online), doi:10.1016/j.jbiomech.2010.04.038 (2010).
- [69] dm76: Cyclist On Bike, <https://www.turbosquid.com/FullPreview/Index.cfm/ID/722423>. (Accessed on Dec. 25, 2018).
- [70] B. Blocken, T. Defraeye, E. Koninckx, J. Carmeliet and P. Hespel: CFD simulations of the aerodynamic drag of two drafting cyclists, *Computers and Fluids*, Vol. 71, pp. 435–445 (online), doi:10.1016/j.compfluid.2012.11.012 (2013).
- [71] T. N. Crouch, D. Burton, M. C. Thompson, N. A. Brown, J. Sheridan, D. M. Fintelman, H. Hemida, M. Sterling and F. X. Li: Dynamic leg-motion and its effect on the aerodynamic performance of cyclists, *Journal of Fluids and Structures*, Vol. 144, pp. 31–41 (online), doi:10.1016/j.jweia.2015.05.009 (2015).
- [72] T. N. Crouch, D. Burton, Z. A. LaBry and K. B. Blair: Riding against the wind: a review of competition cycling aerodynamics, *Sports Engineering*, Vol. 20, No. 2, pp. 81–110 (online), doi:10.1007/s12283-017-0234-1 (2017).
- [73] K. Shirasaki, K. Yamanobe, K. Akashi and W. Takashima: Variation of the drafting effect on

- the trailing rider for different numbers of riders in a cycling group, *Proceedings of the Institution of Mechanical Engineers, Part P: Journal of Sports Engineering and Technology*, No. Special Issue, pp. 1–8 (online), doi:10.1177/1754337117736617 (2017).
- [74] C. T. Davies: Effect of air resistance on the metabolic cost and performance of cycling, *European Journal of Applied Physiology and Occupational Physiology*, Vol. 45, No. 2-3, pp. 245–254 (online), doi:10.1007/BF00421332 (1980).
- [75] A. E. Jeukendrup and J. Martin: Improving Cycling Performance, *Sports Medicine*, Vol. 31, No. 7, pp. 559–569 (online), doi:10.2165/00007256-200131070-00009 (2001).
- [76] J. García-López, J. A. Rodríguez-Marroyo, C. E. Juneau, J. Peleteiro, A. C. Martínez and J. G. Villa: Reference values and improvement of aerodynamic drag in professional cyclists, *Journal of Sports Sciences*, Vol. 26, No. 3, pp. 277–286 (online), doi:10.1080/02640410701501697 (2008).
- [77] T. Defraeye, B. Blocken, E. Koninckx, P. Hespel and J. Carmeliet: Aerodynamic study of different cyclist positions: CFD analysis and full-scale wind-tunnel tests, *Journal of Biomechanics*, Vol. 43, No. 7, pp. 1262–1268 (online), doi:10.1016/j.jbiomech.2010.01.025 (2010).
- [78] J. P. Broker, C. R. Kyle and E. R. Burke: Racing cyclist power requirements in the 4000-m individual and team pursuits, *Medicine and Science in Sports and Exercise*, Vol. 31, No. 11, pp. 1677–1688 (online), https://journals.lww.com/acsm-msse/Fulltext/1999/11000/Racing{}_cyclist{}_power{}_requirements{}_in{}_the{}_4000{}_m.26.aspx (1999).

研究業績

原著論文

- (1) 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, Vol.9, No.2, pp.34–45, 2016.
- (2) 長谷川雄太, 青木尊之, 小林宏充, 白崎啓太: 格子ボルツマン法による自転車競技の集団走行の大規模 LES 空力解析, 日本機械学会論文集 (採録決定)

国際会議講演

- (3) Yuta Hasegawa, Takayuki Aoki, Hiromichi Kobayashi: Aerodynamics Study using Locally Mesh-refined Lattice Boltzmann Method for a GPU Computing, ECCOMAS Congress 2016, Crete Island, Greece, 2016.
- (4) Yuta Hasegawa, Takayuki Aoki: Lattice Boltzmann Simulations Coupled with Octree-based Locally-refined Mesh using Space-filling Curves, COUPLED PROBLEMS 2017, Rhodes Island, Greece, 2017.
- (5) Yuta Hasegawa, Takayuki Aoki: An Octree-based AMR Lattice Boltzmann Method for Multi-GPU Aerodynamics Simulations, SIAM Conference on Parallel Processing for Scientific Computing (SIAM PP18), Tokyo, Japan, 2018.
- (6) Yuta Hasegawa, Takayuki Aoki: AERODYNAMICS STUDY USING LBM WITH OCTREE-BASED MESH REFINEMENT FOR RACING BICYCLES, Parallel CFD 2018, Indianapolis, United States, 2018.
- (7) Yuta Hasegawa, Takayuki Aoki: Aerodynamics Study for Complex Bodies by Using Locally Refined Mesh and Lattice Boltzmann Method, ECCM-ECFD 2018, Glasgow, UK, 2018.

国内会議講演

- (8) 長谷川雄太, 青木尊之, 小野寺直幸: 格子細分化を導入した D3Q27 格子ボルツマン法の GPU 実装, 第 28 回数値流体力学シンポジウム (CFD28), 東京, 2014.
- (9) 長谷川雄太, 青木尊之, 小野寺直幸: 格子ボルツマン法における適合細分化格子の GPU 実装, 第 20 回計算工学講演会 (JSCES20), 筑波, 2015.
- (10) 長谷川雄太, 青木尊之: ステンシル計算の高速化のための C++ テンプレートによる GPU カーネル生成, 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 (HPC149), 東京, 2015.
- (11) 長谷川雄太, 青木尊之: 適合細分化格子 LBM による複雑形状物体周りの流れの LES 計算, 日本機械学会 第 28 回計算力学講演会 (CMD2015), 横浜, 2015.
- (12) 長谷川雄太, 青木尊之: Octree 細分化格子を導入した格子ボルツマン法による乱流の GPU 計算, 第 29 回数値流体力学シンポジウム (CFD29), 福岡, 2015.
- (13) 長谷川雄太, 青木尊之: 適合格子細分化法を導入した格子ボルツマン法の複数 GPU 計算, 第 21 回計算工学講演会 (JSCES21), 新潟, 2016.
- (14) 長谷川雄太, 青木尊之, 小林宏充: 適合格子細分化法を導入した格子ボルツマン法の GPU 計算による自転車競技の空力解析, 日本機械学会 第 29 回計算力学講演会 (CMD2016), 名古屋, 2016.
- (15) 長谷川雄太, 青木尊之: 適合細分化格子を用いた格子ボルツマン法の複数 GPU による大規模計算, 第 22 回計算工学講演会 (JSCES22), 大宮, 2017.
- (16) 長谷川雄太, 青木尊之: 適合格子細分化法を導入した格子ボルツマン法の複数 GPU 計算による非圧縮流れの空力解析, 日本機械学会 第 30 回計算力学講演会 (CMD2017), 大阪, 2017.
- (17) 長谷川雄太, 青木尊之: 適合格子細分化法を導入した格子ボルツマン法による複雑形状物体周りの流れの複数 GPU 計算, 第 31 回数値流体力学シンポジウム (CFD31), 京都, 2017.
- (18) 長谷川雄太, 青木尊之: Octree 細分化格子による LBM を用いたロードバイクの空力計算, 第 23 回計算工学講演会 (JSCES23), 名古屋, 2018.
- (19) 長谷川雄太, 青木尊之: スパコンによる大規模計算に基づいた自転車の集団走行の空力評価, 日本機械学会 シンポジウム: スポーツ工学・ヒューマンダイナミクス 2018 (SHD2018), 京都, 2018.
- (20) 長谷川雄太, 青木尊之: 細分化格子 LBM による自転車の集団走行の大規模空力解析, 日本機械学会 第 31 回計算力学講演会 (CMD2018), 徳島, 2018.
- (21) 長谷川雄太, 青木尊之: 細分化格子 LBM の複数 GPU 計算に基づく自転車の集団走行の空力計算, 第 32 回数値流体力学シンポジウム (CFD32), 東京, 2018.

受賞等

- (22) 東京工業大学大学院 総合理工学研究科創造エネルギー専攻 修士論文優秀論文賞, 2016.

謝辞

本研究を行うにあたり、5年間多大なるご指導を賜りました青木尊之教授に深く感謝いたします。研究が順調なときも行き詰まっているときも、常に冷静に的確なアドバイスをくださり、迷走せずに研究を進めることができました。原著論文の投稿の際には、本文の修正から、査読後の回答書の作成まで、丁寧にご指導いただきました。また、国内外様々な学会や研究会に参加させていただきました。講演発表ばかりでなく聴講のみの参加でも快く旅費を出してくださり、様々な分野の学会に参加し広く見識を得ることができました。本当にありがとうございました。

本研究に関わる様々なテーマについて活発に議論してくださった研究者のみなさまに感謝致します。慶応大学法学部の小林宏充教授には、LES について色々と教えて頂いたほか、研究業績 (2) の投稿の際には、査読に対応するための追加計算の方針などについて丁寧に議論していただきました。日本原子力研究開発機構システム計算科学センターの小野寺直幸研究員には、修士時代にお世話になり、特に格子ボルツマン法や LES の実装について教えていただきました。また、来年度からは同じ職場で働くことになりますので、よろしくお願いします。東京大学情報基盤センターの下川辺隆史准教授には、修士時代に学振 DC や修士論文の文章の添削をしていただきました。日本自転車競技連盟ハイパフォーマンスセンターの白崎啓太様には、HPCI 予算での自転車競技の空力計算について、何度も議論させていただきました。おかげさまで、スポーツ工学の側面から研究背景を充実させることができたことに加え、自転車競技として計算する意義のある計算条件や計算結果を得ることができました。みなさま本当にありがとうございました。

本論文の審査を引き受けてくださった奥野喜裕教授、末包哲也教授、肖鋒教授、長崎孝夫准教授に謝意を表します。

最後に、精神面および金銭面で生活を支えてくれた家族に感謝します。

研究支援に対する謝辞

本研究の一部は科学研究費補助金・基盤研究 (S) 課題番号 2622002「ものづくり HPC アプリケーションのエクサスケールへの進化」、科学技術振興機構 CREST「ポストペタスケール高性能計算に資するシステム・ソフトウェア技術の創出」、革新的ハイパフォーマンス・コンピューティング・インフラ平成 30 年度 A 期 HPCI システム利用課題 (課題番号: hp180146)、学際大規模情報基盤共用利用・共同研究拠点 (課題番号: jh160042, jh170015, jh180034, jh180035) から支援を頂いた。計算には東京工業大学・学術国際情報センターのスーパーコンピュータ TSUBAME2.5 および TSUBAME3.0, 東京大学・情報

基盤センターの Reedbush-H スーパーコンピュータシステム，ならびに九州大学・情報基盤研究開発センターのスーパーコンピュータシステム ITO・サブシステム B を利用した．記して謝意を表す．