格子ボルツマン法乱流シミュレーションの並列計算による高速化

Isogai, Ryosuke / 磯貝, 亮介

(出版者 / Publisher)
法政大学大学院理工学研究科
(雑誌名 / Journal or Publication Title)
法政大学大学院紀要. 理工学研究科編
(巻 / Volume)
63
(開始ページ / Start Page)
1
(終了ページ / End Page)
5
(発行年 / Year)
2022-03-24
(URL)
https://doi.org/10.15002/00025390

格子ボルツマン法乱流シミュレーションの並列計算による高速化

SIMULATION OF LATTICE BOLTZMANN METHOD TURBULENCE BY PARALLEL COMPUTATION

磯貝亮介
Ryosuke Isogai
指導教員　堀端康善　教授

法政大学大学院理工学研究科システム理工学専攻修士課程

In In this study, we simulate three-dimensional turbulent flows using the lattice Boltzmann method. Most of the flows around us are turbulent, and in turbulent flows, the Reynolds number, which evaluates the properties of the fluid, becomes higher. The Reynolds number is higher in turbulent flows. The problem with turbulence simulation is that the calculation process is often complicated and the results are not stable due to the large and small vortices generated, and it takes a lot of time to complete the simulation. To reduce the computation time for turbulence simulations, we use a method called parallelization, in which multiple CPUs or cores are used to process the program in parallel. In addition, by shortening the calculation time, it becomes easier to perform calculations under multiple conditions and compare the results. Therefore, the boundary conditions of the inlet, outlet, and parallel plates are changed and the calculation results are compared to determine the conditions with high calculation accuracy.

***Key Words** : Lattice Boltzmann Methed, OpenMP, turbulent flows, boundary conditions,*

1. はじめに

(1) 研究背景と目的

数値流体力学(computational fluid dynamics, CFD)とは流体の運動に関する方程式をコンピュータで数値的に解き流体の流れを解析、シミュレーションを行う手法である。航空機、自動車、船舶等の流体中を移動する機械や建設物の設計を行う際に重要な存在である。本研究ではCFD の中でも McNamara ら[1]が開発した格子ボルツマン法(Lattice Boltzmann Method, LBM)を用いる。

また我々の身の回りの流れのほとんどは乱流と呼ばれるものであり、乱流においては流体の性質を評価するレイノルズ数と呼ばれる数値が高くなる。流体のシミュレーションを行うときに計算されるのはほとんどが乱流についてである。この乱流シミュレーションを行う上で課題となるのが乱流の計算は複雑な計算過程となることが多く、また発生する大小の渦によって計算結果が安定せず、シミュレーションを終えるために膨大な計算時間を要することである。

本研究では複数の CPU もしくはコアを利用してプログラムを並列処理する並列化と呼ばれる手法を用いることで乱流シミュレーションの計算時間の短縮を図る。

(2) 並列化を行うプログラムについて

本研究において使用したプログラムは３次元流体のシミュレーションを行うものでありその概要図を図 1 に示す。このプログラムはいくつかの工程を指定した step 数で繰り返し計算する(ループ)プログラムであり、プログラムの実行時間の大部分を占めるのもこのループ部分となる。

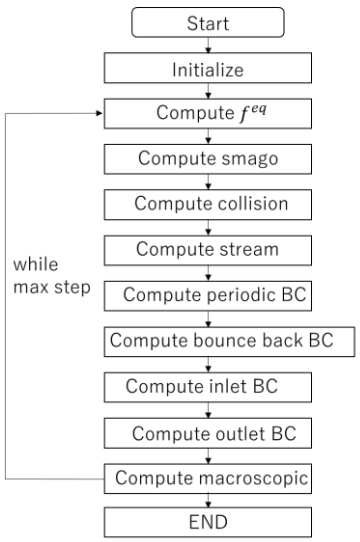

図１　３次元乱流シミュレーションプログラムの概要図

また本研究において用いたコンピュータ(ワークステーション)のスペックを以下に示す。

・hpz8 ワークステーション
Intel Xeon Gold 6234 (1 台)
3.3GHz,8 コア,48GB

2. 並列化について

本研究で用いる並列化とはプログラムを実行するコンピュータ(ワークステーションにおいて)複数の CPU もしくは CPU のコアをスレッドとして指定し、プログラム内の指定した箇所をスレッドごとで並列処理を行う手法である。本研究においては 1 台のワークステーションでメモリを共有し、そのワークステーションの CPU のコアをスレッドとして指定する Open MP[2]と呼ばれる手法を用いる。

Open MP を用いることにより単一プロセッサで動かすことを想定して作られた逐次計算プログラムを並列計算プログラムに書き変えることが出来る。Open MP の記述は文頭が「!$ OMP」となっている指示文によって行われ、Open MP に対応していないコンパイラを使用したときコメント文として無視され、逐次計算と並列計算を容易に切り替えることが出来る。Open MP では指定したスレッド数と同じ回数、同じ動作を繰り返すことが出来る同一演算のほかに、プログラム内のループ数を等分し、ループ内の動作をスレッドごとに分割することでループの分割処理を可能にし、プログラムの実行時間の短縮を行うことが出来るワークシェアリングなどがある。本研究において使用するのは並列化の中でもワークシェアリングだけを用いるため今後の並列化を行う際はワークシェアリングを行うものとする。同一演算とワークステーションについての概要図を図２、図３示す。

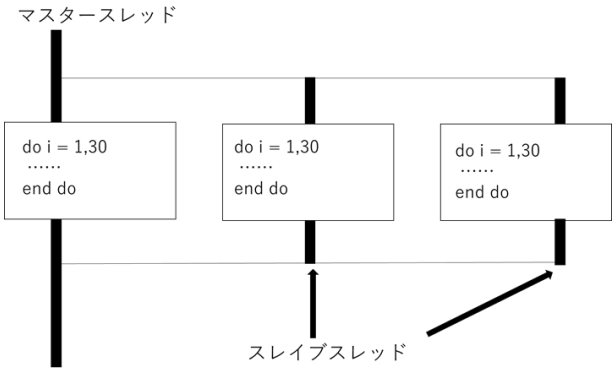

図２ 同一演算の並列実行について

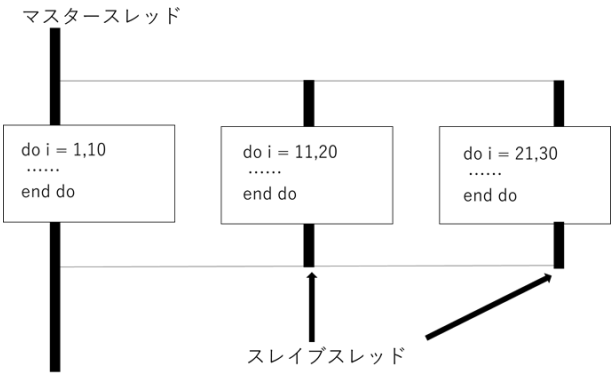

図３ ワークシェアリングについて

3. 問題提起

本研究において使用する３次元流体のシミュレーションプログラムにおいてどの部分を並列化したときに実行時間を効果的に短縮できるか検証するためにプログラムを max step(80000step)動かしたときの各サブルーチンの経過時間を計測した。計測したサブルーチンは図１のループ部分である comfeq、smago、collision、stream、periodicbc 、 bouncebackbc 、 inletbc 、 outletbc 、 macroscopic であり、プログラムを 80000step 動かしたときの各サブルーチンの経過時間を以下の表１にまとめる。

表１ 逐次計算時においてプログラムを 80000step 動かしたときのサブルーチンごとの経過時間のまとめ

	経過時間(s)
comfeq	4291
smago	25482
collision	11508
stream	16557
periodicbc	145
bouncebackbc	354
inletbc	22
outletbc	5
macroscopic	68750

以上の表を検討してサブルーチン comfeq、smago、collision、stream、macroscopic で並列化を行うことにした。プログラムの実行時間の大部分を占めるサブルーチンで並列化を行うことでプログラム全体の実行時間の短縮を図る。

また並列化によって変化した経過時間を比較するためにスピードアップという数値を用いる。スピードアップは逐次計算時の経過時間と比べて並列化時の経過時間が何倍速くなったかを数値化したものであり、この数値が大きいほどより早くプログラムが実行できていると評価する値となる。理論値としては使用したスレッド数と同じ値になる。スピードアップは以下の式１で計算する。

$$\text{スピードアップ} = \frac{\textit{逐次計算時の経過時間}}{\textit{並列化時の経過時間}} \qquad (1)$$

また並列化を行った際に指定したスレッドがどの程度の効率で計算しているかを評価する値として並列化効率(%)という値を用いる。この値が 100(%)に近いほど各スレッドが最大限の性能で計算を行っていると評価できる。並列化効率は以下の式 2 で計算する。

$$\text{並列化効率}(\%) = \frac{\textit{スピードアップ}}{\textit{スレッド数}} \times 100 \qquad (2)$$

4. 並列化の効果を検証

各サブルーチンで並列化を行う際にデータ局所性と呼ばれるものを意識してプログラムの書き換えを行った。本研究で用いるプログラムはプログラミング言語 Fortran を用いている。Fortran において 2 次元配列 x(i,j) は x(1,1) ,x(2,1)…という順序でメモリ上に 1 次元配置されているのでループの順序が変わると局所性が低下し、場合によってはプログラムの実行時間が増大する可能性がある。このデータ局所性を考慮して、配列を用いる多重ループ計算においてはループの順番の入れ替えを行うことで逐次計算時でもプログラムの実行時間を短くすることが出来る場合もある。またプログラムの実行において処理に時間がかかる if 文をなるべく使用しないようにプログラムの書き換えを行った。プログラムの書き換えと並列化の処理を行ったプログラムにおいて、プログラムを 80000step 動作させたときの逐次計算時とスレッド数 1,2,,6,8 と指定して並列化を行ったときで経過時間の測定を行い、逐次計算時を基準としてスピードアップと並列化効率を求めた。また比較として CPU time を計測した。経過時間を実際にプログラムが動作している時間であり、CPU time は CPU が動作したときの時間を合計したものになる。表 2 に結果を示す。

表 2 逐次計算時と各スレッド数で並列化したときの経過時間の比較(80000step)

状態	スレッド数	CPU time (h)	経過時間(h)	スピードアップ	並列化効率(%)
逐次	1	16.67	16.72	1.00	100
並列化	1	16.75	16.79	1.00	100
	2	19.60	9.82	1.70	85
	4	25.56	6.41	2.61	65
	8	41.83	5.31	3.15	39

以上のようにスレッド数を増やしていくごとに経過時間が短縮していくことが確認できた。スレッド数を増やしていくごとに並列化効率は低下しているがこれは計測した経過時間には並列化していない部分や並列化を行う際に発生する処理の時間も含まれているためだと考えられる。スピードアップはスレッド数と同じ値になることが理論値であり、そのスピードアップの理論値と実測値の関係を以下の図 4 に示す。

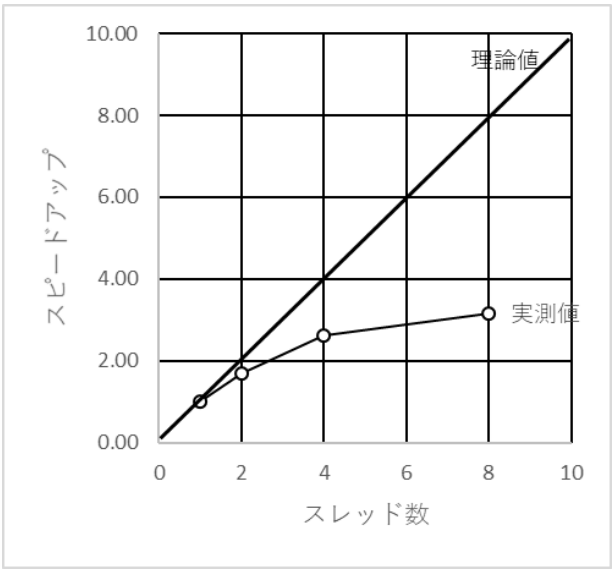

図 4 プログラムを 80000step 動かしたときにおけるスピードアップの理論値と実測値の比較

5. 境界条件の比較

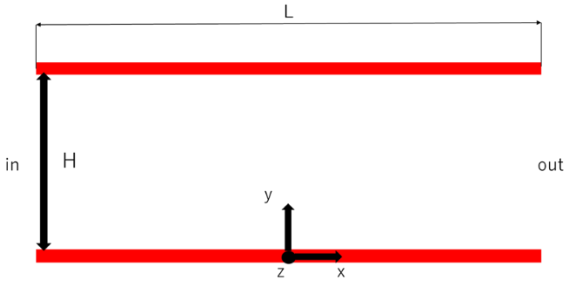

図 5 ポアズイユ流れと x,y,z 方向について

前項までで乱流をシミュレーションするプログラムの計算時間を短縮することに成功した。計算時間が短縮したことで計算結果の比較を行いやすくなったので、次に境界条件を変更して計算結果がどのように変化するかを検証する。計算場としては図 5 のような上下に平行板を設置し、流入部から粒子を流入させることで流れを発生させるポアズイユ流れを採用する。計算結果の比較として流速の x 成分最大値の理論値u_{max}を用いる。u_{max}は平行板間距離 H、流入部密度ρ_{in}と流出部密度ρ_{out}の密度差Δρ、格子点における密度ρ、動粘性係数ν、流れ場の長さ L を用いて以下の式(3)によって計算される。

$$u_{max} = -\frac{H^2 \Delta\rho}{24\rho\nu L} \qquad (3)$$

また計算場の中心となる格子点において理論値u_{max}と計算値 u の差を以下の式で計算し、誤差を求める。

$$\frac{|u_{max}-u|}{u_{max}}\times 100 = \text{誤差}(\%) \tag{4}$$

比較対象として流入部と流出部の境界条件密度指定平衡スキームと平行板上では速度指定平衡スキームを用いる。例として流入部における計算について示す。
平衡スキームは境界上の格子点において計算場の外から向かってくる未知の粒子分布関数に対して計算によって算出できる既知の粒子分布関数を用いて流速の x 成分 u を算出し、その u と措定した密度を用いて境界上の格子点において平衡分布関数を計算し、それを粒子の分布関数とするものである。以下に流入部における平衡スキームの計算法を示す。

ρ：境界上における格子点の粒子密度
ρ_-：未知の粒子分布関数の和
ρ_+：既知の粒子分布関数の和
ρ_0：格子速度が境界面上にある粒子分布関数と
　格子速度 0 の粒子分布間の和

$$\rho = \rho_- + \rho_+ + \rho_0 \tag{5}$$
$$\rho \cdot u = -\rho_+ + \rho_- \tag{6}$$
$$\therefore u = 1 - \frac{2\rho_+ + \rho_0}{\rho} \tag{7}$$
$$\rho_0 = f_0 + f_3 + f_4 + f_5 + f_6 + f_{15} + f_{16} + f_{17} + f_{18} \tag{8}$$
$$\rho_+ = f_2 + f_9 + f_{10} + f_{13} + f_{14} + f_{23} + f_{24} + f_{25} + f_{26} \tag{9}$$
$$\rho_- = f_1 + f_7 + f_8 + f_{11} + f_{12} + f_{19} + f_{20} + f_{21} + f_{22} \tag{10}$$

流入部における ρ は$\rho = \rho_{in}$として流入部の密度ρ_{in}を数値設定して計算を行う。
以上の u とρ_{in}を用いて格子点における i 方向の粒子の分布関数f_iを粒子の平衡分布関数$f_i^{(eq)}$によって求める。

$$f_i = f_i^{(eq)}(\rho = \rho_{in}, u) \tag{11}$$

同様の条件を流出部と平行板上の条件として発展させたものを用いる。この時に以下の条件で計算を行った時の計算値 u と理論値u_{max}から算出した理論値を図 6 にまとめる。

壁面の境界条件：速度指定平衡スキーム
流入出口の境界条件：密度指定平衡スキーム
計算場の大きさ(x × y × z)：$10 \times 2 \times 2$,
平行板間格子数 = 80 個,　$\rho = 1.0$,
$\nu = 1.0 \times 10^{-2}$,　$\rho_{in} = 1.0015$,　$\rho_{out} = 0.9985$,
$\Delta\rho = \rho_{in} - \rho_{out} = 0.003$,　H = 2.0,　L = 10,
Re = 1.0

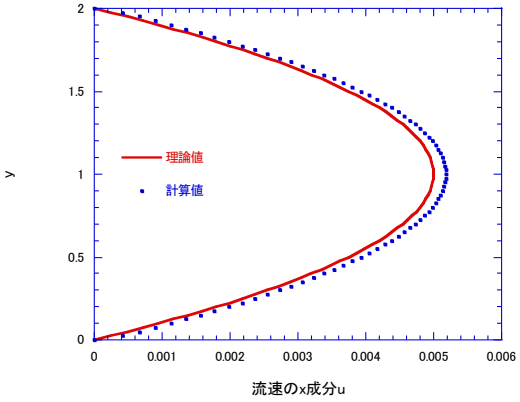

図 6　x=5,z=1 における流速の x 成分 u の
y 軸方向分布（$u_{max} = 0.005$）

また(x,y,z)=(5,1,1)における流速の x 成分 u の理論値u_{max}と計算値 u の誤差を式(4)で計算し、その結果を表 4 にまとめる。

表 4　(x,y,z)=(5,1,1)における流速の x 成分 u の理論値
u_{max}と計算値 u の誤差

理論値	計算値	誤差(%)
0.005	0.00518	3.588

この結果から誤差が小さくなることを目的に境界条件をいくつか変更して計算を行った結果、流入出部の境界条件は密度指定平衡スキーム、平行板上の境界条件として bounceback BC と呼ばれる境界条件を用いたときに計算誤差が一番小さくなる結果を得られた。bounce back BC は壁にぶつかった仮想粒子が反対方向に返ってくるような動きを与える条件である。つまりベクトルとして反対の向きになっている分布関数を与える条件である。この bounce back BC を平行板上の境界条件として用い、流入出部の境界条件は密度指定平衡スキームを用いたときの計算値 u(bounce back BC)と理論値u_{max}から算出した理論値を図 7 にまとめる。また先ほどの図 6 の結果と比較のため図 6 の計算値 u を u(密度指定平衡スキーム)として図 7 に示す。

壁面の境界条件：bounce back BC
流入出口の境界条件：密度指定平衡スキーム
計算場の大きさ(x × y × z)：$10 \times 2 \times 2$,
平行板間格子数 = 80 個,　$\rho = 1.0$,
$\nu = 1.0 \times 10^{-2}$,　$\rho_{in} = 1.0015$,　$\rho_{out} = 0.9985$,
$\Delta\rho = \rho_{in} - \rho_{out} = 0.003$,　H = 2.0,　L = 10,

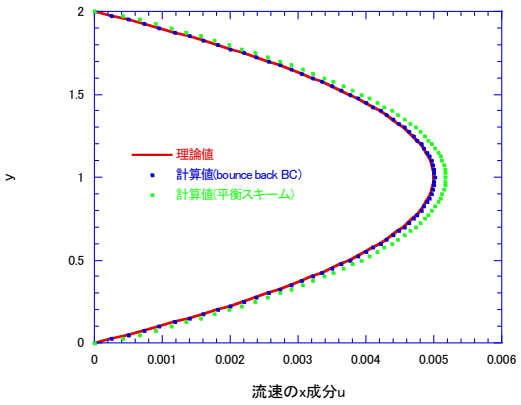

図 42　x=5,z=1 における流速の x 成分 u の y 軸方向分布($u_{max} = 0.005$)

(x,y,z)=(5,1,1)における流速の x 成分 u の理論値u_{max}と計算値 u の誤差を式(4)で計算し、その結果を bounce back BC として、また比較対象として上記の条件の壁面の境界条件を平衡スキームに変えたときにおける同様の比較を平衡スキームとして表 21 にまとめる。

表 21　(x,y,z)= (5,1,1)における流速の x 成分 u の理論値u_{max}と計算値 u の誤差

	理論値	計算値	誤差(%)
bounce back BC	0.005	0.00501	0.208
平衡スキーム	0.005	0.00518	3.588

以上のように境界条件を変更することで計算誤差を約 3.6%から約 0.2%まで減少させた。

参考文献

[1] Guy R Mcnamara and Gianluigi Zanetti.:Use of the Boltzmann equation to simulate lattice-gas automata. Physical review lettees, Vol.61, No.2, p.2332, 1998.

[2]牛島省：OpenMP による並列プログラミングと数値計算法，丸善出版株式会社, 2006.

[3]斉木善唯，瀬田剛：格子ボルツマン法による非圧縮性流体解析，日本機械学会論文集(B 編)74 巻 46 号，p58-p65

[4]Chih-Fung Ho，Cheng Chang，Kuen-Hau Lin and Chao-an Lin：Consistent Boundary Conditions for 2D and 3D Lattice Boltzmann Simulations． Computer Modeling in Enginieering and Science (CMES)，Vol.44, No2, p. 137-p155, 2009.

[5]森西洋平，小林敏雄：人工的壁面境界条件を用いた LES の構成およびその評価．日本機械学会論文集(B 編), 57 巻, 540 号, p115-p121.