JAERI-Data/Code
2000-029



# 格子ボルツマン法による二相流シミュレーションコードの開発並びに並列化

2000年9月

渡辺　正・海老原　健一・伊藤　豪一*・河野　浩二*

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは、日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは、日本原子力研究所研究情報部研究情報課（〒319-1195　茨城県那珂郡東海村）あて、お申し越し下さい。なお、このほかに財団法人原子力弘済会資料センター（〒319-1195　茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布を行っております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 〒319-1195, Japan.


編集兼発行　日本原子力研究所

JAERI－Data/Code 2000－029

# 格子ボルツマン法による
# 二相流シミュレーションコードの開発並びに並列化

日本原子力研究所計算科学技術推進センター
渡辺　正・海老原　健一・伊藤　豪一*・河野　浩二*

(2000 年 7 月 24 日　受理)

格子ボルツマン法による 3 次元二成分二相流シミュレーションコードを作成し、MPI ライブラリを用いて並列化を行った。サンプル問題として上昇気泡及び気泡の合体のシミュレーションを行い、上昇速度、気泡周囲の流れ場等に関して妥当な結果を得た。異なる機種のワークステーションからなるクラスターを用い、計算速度に応じた領域分割を行うことによりシミュレーションを行った。データ転送量が増えるにつれて計算効率は低下したが、実用上十分な並列化効率が得られた。

日本原子力研究所（東海駐在）：〒 319-1195　茨城県那珂郡東海村白方白根 2-4
* (株) 富士総合研究所

JAERI－Data/Code 2000－029

# Development of a Two-phase Flow Simulation Code Using the Lattice Boltzmann Method and Its Parallelization

Tadashi WATANABE, Ken-ichi EBIHARA, Goichi ITO* and Koji KOHNO*

Center for Promotion of Computational Science and Engineering
(Tokai Site)
Japan Atomic Energy Research Institute
Tokai-mura, Naka-gun, Ibaraki-ken

(Received July 24 , 2000)

A three-dimensional two-component two-phase flow simulation code using the lattice Boltzmann method has been developed and parallelized using the MPI library. Rising bubbles and their coalescence were simulated as sample problems and reasonable results were obtained for the rising velocity and the flow field around the bubble. Numerical simulations were performed on a cluster composed of different workstations by using the domain decomposition according to the calculation speed. Reasonable efficiency of parallelization was obtained though the efficiency was degraded as the amount of data transfer increased.

Keywords : Two-phase Flow, Lattice Boltzmann Method, Parallelization, MPI Library

* Fuji Research Institute Corporation

# 目　　次

1. はじめに . . . . . 1
2. 格子ボルツマン法 . . . . . 3
2.1 格子形状 . . . . . 3
2.2 単相流モデル . . . . . 5
2.3 二成分二相流モデル . . . . . 13
3. シミュレーションコードの並列化 . . . . . 27
3.1 メインプログラム . . . . . 27
3.2 データ転送 ： サブルーチン exchange0 . . . . . 29
3.3 移動：サブルーチン stream . . . . . 32
3.4 密度と速度：サブルーチン cal_rho_v . . . . . 33
3.5 衝突項：サブルーチン collision . . . . . 34
3.6 転送 ： サブルーチン exchange2 . . . . . 36
3.7 表面張力 ： サブルーチン cal_omega . . . . . 41
3.8 粒子の再配置 ： サブルーチン recoloring . . . . . 43
3.9 境界条件 ： サブルーチン set_boundary . . . . . 45
3.10 出力データ集約 ： サブルーチン gather . . . . . 50
3.11 データ出力 ： サブルーチン filtering . . . . . 51
4. 並列計算 . . . . . 54
4.1 上昇気泡のシミュレーション . . . . . 54
4.2 並列化効率 . . . . . 57
4.3 異なるタイプのワークステーションからなるクラスターの場合 . . . . . 58
5. まとめ . . . . . 60
参考文献 . . . . . 61

Contents

1. Introduction . . . . . 1
2. Lattice Boltzmann Method . . . . . 3
2.1 Lattice Configuration . . . . . 3
2.2 Single-phase Flow Model . . . . . 5
2.3 Two-component Two-phase Flow Model . . . . . 13
3. Parallelization of Simulation Code . . . . . 27
3.1 Main Program . . . . . 27
3.2 Data Transfer : Subroutine Exchange0 . . . . . 29
3.3 Traveling : Subroutine Stream . . . . . 32
3.4 Density and Velocity : Subroutine Cal_rho_v . . . . . 33
3.5 Collision Term : Subroutine Collision . . . . . 34
3.6 Data Transfer : Subroutine Exchange2 . . . . . 36
3.7 Surface Tension : Subroutine Cal_omega . . . . . 41
3.8 Redistribution of Particles : Subroutine Recoloring . . . . . 43
3.9 Boundary condition : Subroutine Set_boundary . . . . . 45
3.10 Summation of Output Data : Subroutine Gather . . . . . 50
3.11 Output Data : Subroutine Filtering . . . . . 51
4. Parallel Computation . . . . . 54
4.1 Simulation of Rising Bubble . . . . . 54
4.2 Efficiency of Pallalelization . . . . . 57
4.3 In Case of the Cluster with Different Type of Workstations . . . . . 58
5. Summary . . . . . 60
References . . . . . 61

# 1. はじめに

日本原子力研究所計算科学技術推進センターでは、原子力分野で見られる複雑現象として、代表的な伝熱・流動現象をとりあげ、その基礎過程を計算機シミュレーションを通して解明する研究を進めている。流動現象の解析は、通常、ナビエ ストークス方程式に代表されるマクロな連続流体の微分方程式を数値的に解くことによって行われているが、原子炉の熱流動としてもっとも基本的かつ重要である二相流現象の解析にあたっては、基礎方程式や数値解法に何らかの仮定を施す必要がある。これは、二相流では二相の界面の形状が不規則に変化することや、分散していた相が合体したり、一つの相が分裂することにより自由表面の生成消滅が不規則に起こり、流動様式が変化するためである。解析においては、それぞれの相内部は単相流の扱いで良いが、単相流の境界としての二相界面では、気相と液相、界面形状の変化などを同時に扱う必要がある。なめらかな界面を持つ水平層状流や自由表面流のような場合には界面を境界として気相、液相それぞれを単相流として計算することができるが、気泡や液滴を多数含んだり、混合が激しい流れでは、液相の体積率関数の輸送方程式を用いたり、気相のボイド率により液相と気相の方程式をカップリングさせる方法が用いられる。この場合、界面を通しての運動量やエネルギー輸送の評価には実験相関式や経験的な仮定などを用いることが多い [1]。

熱流動現象は、多数の流体の原子・分子、あるいは流体の微小な塊としての流体粒子（流体要素）による集団的な運動であると考え、粒子の運動を追跡しその統計的な平均値として流れ場の温度、速度などの流動状態を求める粒子法と呼ばれる数値解法がある。粒子法には、ミクロなレベルで原子・分子の運動を扱う直接シミュレーションモンテカルロ法 [2] や、分子動力学法 [3]、メソスコピックないしはマクロなレベルで流体粒子を扱う格子ガス系の手法 [4]、スムースパーティクル系の手法 [5] 等がある。これら代表的な粒子法のうち直接シミュレーションモンテカルロ法は、希薄気体の解析のためにボルツマン方程式を解く手法として発展してきたもので、流れ場の代表長さに対する粒子の平均自由行程の比が 0.01 程度以上の場合に用いられる。それ以下の場合、信頼できる結果を得るためには膨大な数の粒子が必要となり現在のスーパーコンピュータのレベルでも不十分となる。すなわち、直接シミュレーションモンテカルロ法単独で、液相あるいは二相流を扱うことはできない。分子動力学法は、分子間に働く力をポテンシャルから求めて運動方程式を解くため、粒子数が多くなると力の計算に多くの計算時間が必要となる。また、計算できるサイズが分子の大きさのオーダーであるため、二相流の計算を行う際に扱える現象としては、二相界面での蒸発・凝縮に関わる分子運動や分子レベルでのミクロな流体モードといった現象に限られる。格子ガス系の手法は、物理的な空間を格子で置き換え、格子上に限定された仮想的な流体粒子の運動で流体現象を模擬するもので、多くの流体現象の計算が効率的に行えるにもかかわらず、非物理的な制約も多い。しかし、ボルツマン方程式の採用などにより、二相流に限らず、多くの流動現象が現実的な時間と精度で計算できるようになってきており、今後、応用が広がるものと思われる。スムースパーティクル系の手法やその他の粒子法は、流動現象の計算に対しては発展段階であり、仮定や制限も多く定性的な流れのパターンなどが議論されることが多い。

粒子法は、分子運動に起因する統計的な変動を平均的な流れ場と同時に求めたり、複雑な流路の模擬や相界面の形状変化を記述するのに優れているが、変動の少ない定常的な流れ場を再現す

るためには統計平均数を多くする必要があり、一般に多数の粒子が必要となる。近年、計算機やネットワークの進歩と並列計算技術の進展にともなって、多数の粒子を現実的な計算時間で扱うことが可能となってきており、今後、粒子法の様々な流動現象への適用が期待されている[6]。特に、ここ数年、これまで並列型のスーパーコンピュータでなければ計算できなかったような大規模かつ長時間の計算が、ワークステーションやパソコンレベルで計算できるようになってきている。このため、ワークステーションやパソコンをネットワークで接続したクラスターを利用した並列計算の技術が、経済性や取り扱いの容易さといった点で重要性を増してきている。

二相流現象の数値解析への応用が期待される粒子法の一つとして、格子ガス系の手法のうちもっともシンプルなモデルによる2次元二相流シミュレーションコードについては、既に、コードの構造と並列化の概要、ワークステーションクラスターを用いた並列計算等について検討を行った[7, 8]。本報告書では、格子ガス法の欠点を補う手法として発展してきた格子ボルツマン法による3次元二相流のシミュレーションコードについて記述する。格子ボルツマン法は、時間空間に関して離散化を行った格子ボルツマン方程式を基礎方程式とし、従属変数である分布関数の時間発展を計算するものである。これは、格子ガス法において粒子の有無を1と0の整数で表していたのに対し、粒子の存在の期待値を実数で表すことに対応している。すなわち、分布関数は粒子の有無についての時間平均に対応しており、格子ガス法の欠点の一つであった雑音の問題を解消したものとなっている。以下では、まず手法の概要を基本的な単相流モデルにより示し、次に二相流モデルに関して基礎方程式と巨視的流体方程式の対応を示す。シミュレーションコードはFORTRAN言語により作成し、MPIライブラリを用いて並列化を行う。サンプル問題は、上昇気泡のシミュレーションとした。

# 2. 格子ボルツマン法

格子ボルツマン法は、物理空間を格子により離散化し、仮想的な流体粒子の格子上の移動と衝突により巨視的な流体挙動を模擬する計算モデルである。格子ボルツマン法で扱う粒子は、格子ガス法のように粒子の有無を示すブーリアン変数（整数）ではなく、存在の期待値を示す分布関数（実数）により表される。すなわち、分布関数の格子上での時間発展を求めるものであり、これは、分布関数の位相空間での時間発展を表すボルツマン方程式を離散化した格子ボルツマン方程式の数値計算を行うことに対応している。分布関数を用いるため、格子ガス法においてなめらかな流れ場を得るために必要とされた時間方向の粗視化は、格子ボルツマン法では必要がなく、格子ガス法の欠点の一つである解に含まれる雑音の問題は解消されている。ただし、変数が整数から実数になることにより、格子ガス法における計算速度やメモリー容量などの計算効率に関する利点は失われている。

以下では、まず格子ボルツマン法の基本となる単相流のモデルについて格子ボルツマン方程式とナビエ－ストークス方程式の対応を示し、次に2成分2相流モデルの詳細を記述する。

## 2.1 格子形状

### 2.1.1 3次元15速度モデル

3次元の格子モデルは15速度、19速度などの幾つかのモデルが提唱されているが、計算効率の観点から、Fig.2.1に示される3次元の最小格子単位である3次元15速度モデル(3D15V)を用いる。

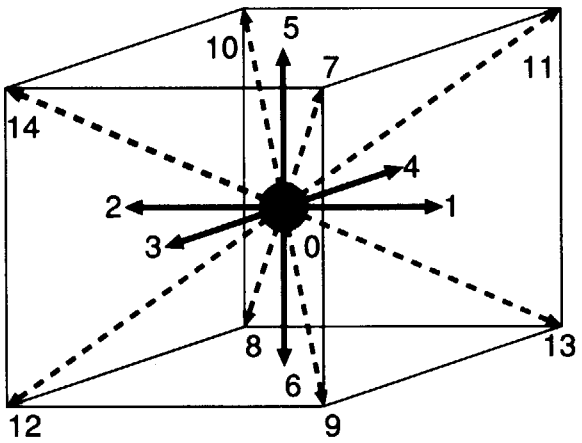



Fig. 2.1 Lattice structure for 3D15V model

3D15Vモデルは、14方向($\boldsymbol{e}_i, i = 1, \cdots 14$)の速度を持つ粒子と一つの静止粒子($\boldsymbol{e}_0$)を仮定したモデルである。単位格子は立方体によって構成され、6つの最近接近傍(面の中心方向)と8つの第2近接近傍(各頂点方向)が存在し、各近傍に格子が継っている。この格子は 面心立方格子(FCC)と呼ばれている。

仮想粒子は、各タイムステップにおいて、ひとつの格子点から継っている近傍の格子点にのみ移動できる。その速さ $|\boldsymbol{e}_i|$ は格子方向により3種類に分類される。各タイプの速度ベクトルは格子点を原点として、

$$\text{Type 0 } (\sigma=0): \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (i=0) \tag{2.1}$$

$$\text{Type 1 } (\sigma=1): \begin{pmatrix} \pm1 \\ 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ \pm1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \pm1 \end{pmatrix} \quad (i=1,\cdots 6) \tag{2.2}$$

$$\text{Type 2 } (\sigma=2): \begin{pmatrix} \pm1 \\ \pm1 \\ \pm1 \end{pmatrix}, \begin{pmatrix} \pm1 \\ \pm1 \\ \mp1 \end{pmatrix}, \begin{pmatrix} \pm1 \\ \mp1 \\ \pm1 \end{pmatrix}, \begin{pmatrix} \pm1 \\ \mp1 \\ \mp1 \end{pmatrix} \quad (i=7,\cdots 14) \tag{2.3}$$

と表される(複号同順)。また、各タイプの速さ $c_i = |\boldsymbol{e}_i^2|$ は

$$\text{Type 0} \quad (i=0) \quad : |\boldsymbol{e}_0|^2 = 0 \tag{2.4}$$

$$\text{Type 1} \quad (i=1\cdots 6) \quad : |\boldsymbol{e}_i|^2 = 1 \tag{2.5}$$

$$\text{Type 2} \quad (i=7\cdots 14) \quad : |\boldsymbol{e}_i|^2 = 3 \tag{2.6}$$

となる。$\sigma$ の値により、$i$ の範囲が異なる事に注意する。

### 2.1.2 格子テンソル

各格子方向のベクトルに関する各階のテンソルは、以下のようになる。ここで、

$$\sum_i{}' = \begin{cases} \sum_{i=1}^{6} & \sigma=1 \\ \sum_{i=7}^{14} & \sigma=2 \end{cases} \tag{2.7}$$

とする。

奇数次数は全て0であり、

$$\sum_i{}' e_{i\alpha} = 0 \tag{2.8}$$

$$\sum_i{}' e_{i\alpha}e_{i\beta}e_{i\gamma} = 0 \tag{2.9}$$

$$\sum_i{}' e_{i\alpha}e_{i\beta}e_{i\gamma}e_{i\theta}e_{i\xi} = 0 \tag{2.10}$$

2階および4階のテンソルは、式(2.11)および(2.12)で表される。

$$\sum_i{}' e_{i\alpha}e_{i\beta} = \frac{b_\sigma C_\sigma^2}{D}\delta_{\alpha\beta} \tag{2.11}$$

$$\sum_i{}' e_{i\alpha}e_{i\beta}e_{i\gamma}e_{i\theta} = \phi_\sigma \Delta_{\alpha\beta\gamma\theta} + \psi_\sigma \delta_{\alpha\beta\gamma\theta} \tag{2.12}$$

ここで、$\delta_{\alpha\beta}$ は Kronecker delata であり、

$$\delta_{\alpha\beta} = \begin{cases} 1 & \alpha = \beta \\ 0 & \alpha \neq \beta \end{cases} \tag{2.13}$$

$D$ は格子の次元数 ($D = 3$)、$b_\sigma$ および $C_\alpha$ は 各々タイプ $\sigma$ の格子数 ($b_0 = 1$、$b_1 = 6$ 、$b_2 = 8$) と格子長さ ($C_0 = 0$、$C_1 = 1$、$C_2 = \sqrt{3}$) である。また、

$$\Delta_{\alpha\beta\gamma\theta} = \delta_{\alpha\beta}\delta_{\gamma\theta} + \delta_{\alpha\gamma}\delta_{\beta\theta} + \delta_{\alpha\theta}\delta_{\beta\gamma} \tag{2.14}$$

$$\delta_{\alpha\beta\gamma\theta} = \delta_{\alpha\beta}\delta_{\beta\gamma}\delta_{\gamma\theta} \tag{2.15}$$

であり、$\phi$、$\psi$ はモデルに依存する係数である。3D15V モデルでは

| $\sigma$ | $\phi$ | $\psi$ |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 2 |
| 2 | 8 | -16 |

となる。ギリシャ文字のサブスクリプトは、インデックス $i$ 格子方向における、座標方向のコンポーネントを表す。また、以下の表記において、

$$\sum_{i\sigma} = \sum_{\sigma}\sum_{i}' \tag{2.16}$$

とする。

## 2.2 単相流モデル

### 2.2.1 概要

単相流に対する 格子ボルツマン方程式は以下の様に表わされる。

$$f_i(\boldsymbol{x} + \boldsymbol{e}_i, t+1) - f_i(\boldsymbol{x}, t) = \Omega_i(f_i(\boldsymbol{x}, t)) \tag{2.17}$$

ここで、$f_i(\boldsymbol{x}, t)$ は格子方向 $i$ に沿った空間 $\boldsymbol{x}$ および 時刻 $t$ に対して離散的な分布関数である。また、$\boldsymbol{e}_i$ は $i$ 番目の格子方向の粒子の速度ベクトルを表す。さらに、$\Omega_i(f_i(\boldsymbol{x}, t)) = \Omega_i$ は衝突項であり、各タイムステップにおいて、分布関数 $f_i(\boldsymbol{x}, t)$ を更新させるものである。

衝突項のモデルで最も広く用いられているものは、BGK 近似と呼ばれるモデルであり、緩和時間 $\tau$ を用いて 式 (2.18) のように表される。

$$\Omega_i^{(BGK)} = -\frac{1}{\tau}[f_i(\boldsymbol{x}, t) - f_i^{(eq)}(\boldsymbol{x}, t)] \tag{2.18}$$

また、この項は BGK オペレータ とも呼ばれている。

この BGK オペレータは位置 $\boldsymbol{x}$ および時刻 $t$ の局所平衡分布 $f_i^{(eq)}(\boldsymbol{x}, t)$ を用いて、$f_i(\boldsymbol{x}, t) = f_i$ が平衡状態に緩和して行く過程を示している。巨視的な物理量である密度 $\rho$ および運動量 $\rho\boldsymbol{u}$

は分布関数 $f_i$ を用いて式 (2.19) 、 (2.20) により定義される。また、局所的な質量および運動量保存から平衡分布関数 $f_i^{(eq)}(\boldsymbol{x},t)$ は、式 (2.21) 、 (2.22) の関係を満たしている。

$$\rho = \sum_{i\sigma} f_i \tag{2.19}$$

$$\rho \boldsymbol{u} = \sum_{i\sigma} f_i \boldsymbol{e}_i \tag{2.20}$$

$$\rho = \sum_{i\sigma} f_i^{(eq)} \tag{2.21}$$

$$\rho \boldsymbol{u} = \sum_{i\sigma} f_i^{(eq)} \boldsymbol{e}_i \tag{2.22}$$

格子ボルツマン方程式は、流体運動に対する連続の式 (2.23) および運動量保存 (2.24) の巨視的方程式を満たす。

$$\partial_t \rho + \partial_\alpha \rho u_\alpha = 0 \tag{2.23}$$

$$\partial_t \rho u_\alpha + \partial_\beta \rho u_\alpha u_\beta = \partial_\beta P_{\alpha\beta} + \partial_\alpha [\lambda \partial_\gamma u_\gamma] + \partial_\beta [\mu [\, \partial_\alpha u_\beta + \partial_\beta u_\alpha]] + \rho K_\alpha \tag{2.24}$$

ここで、$\rho$ は密度、$\boldsymbol{u}$ は局所速度 $P_{\alpha\beta}$ は圧力テンソル、$K_\alpha$ は外力であり、$\lambda$ および $\mu$ は各々第二粘性係数およびバルクな粘性係数である。ギリシャ文字 $\alpha\beta\cdots$ は空間に関するカーティシアンであり、縮約をとる。

圧力テンソルは、

$$P_{\alpha\beta} = -p\delta_{\alpha\beta} \tag{2.25}$$

であたえられ、$p$ はバルク圧力であり、局所密度および音速 $c_s$ に依存する。

$$p = C_s^2 \rho \tag{2.26}$$

以下には巨視的方程式の導入の詳細を示す。

### 2.2.2 平衡分布関数と巨視的方程式

#### 2.2.2.1 平衡分布

格子ボルツマン法では、仮想粒子の平衡密度分布関数として、以下に示す Maxwell-Boltzmann 分布を仮定する。

$$f_i^{(eq)} = \frac{1}{\exp\left(h(\rho,\boldsymbol{u}) + \boldsymbol{q}(\rho,\boldsymbol{u})\cdot\boldsymbol{e}_i\right)} \tag{2.27}$$

ここで、係数 $h$ および $\boldsymbol{q}$ を $\boldsymbol{u}$ まわりで展開し、パリティ不変を考慮すると、

$$f_i^{(eq)} = f_i^{(eq)}(1 - h_1 u_\alpha^2 - q_1 u_\alpha e_{i\alpha} + \frac{1}{2} q_1^2 (u_\alpha e_{i\alpha})^2) + O(u^3) \tag{2.28}$$

となる。ここで、$u_\alpha$ は速度、$e_{i\alpha}$ は格子方向、$h_1$、$h_2$ 、$q_1$ は $\rho$ に依存する係数である。また、$f_i^{(eq)}$ は平衡分布関数である。

密度 $\rho$ 運動量 $\rho u_\alpha$ は平衡分布関数 $f_i^{(eq)}$ からのみ決まり、

$$\sum_{i\sigma} f_i^{(eq)} \begin{pmatrix} 1 \\ e_{i\alpha} \end{pmatrix} = \begin{pmatrix} \rho \\ \rho u_\alpha \end{pmatrix} \tag{2.29}$$

となる。

$\boldsymbol{u} = 0$ を仮定し、$f_i^{(eq)} = d_\sigma$ とすると、$f_i^{(eq)}$ は

$$\sum_{i\sigma} f_i^{(eq)} = \sum_{i\sigma} d_\sigma \tag{2.30}$$

$$= \rho \tag{2.31}$$

を満たす。

式 (2.28) は

$$f_i^{(eq)} = d_\sigma(1 - h_1 u_\alpha^2 - q_1 u_\alpha e_{i\alpha} + \frac{1}{2} q_1^2 u_\alpha u_\beta e_{i\alpha} e_{i\beta}) \tag{2.32}$$

となる。ここで、$O(u^3)$ の項は無視した。

式 (2.32) に $e_{i\alpha}$ を乗じたものと 式 (2.32) に対しそれぞれ、$i$ および $\sigma$ で和をとると、

$$\sum_{i\sigma} e_{i\alpha} f_i^{(eq)} = -q_1 u_\alpha \sum_\sigma \frac{b_\sigma C_\sigma^2}{D} d_\sigma \tag{2.33}$$

$$= \rho u_\alpha \tag{2.34}$$

$$\sum_{i\sigma} f_i^{(eq)} = \rho - h_1 u_\alpha^2 \rho + \frac{1}{2} \frac{\rho^2 D u_\alpha^2}{\sum_\sigma b_\sigma C_\sigma^2 d_\sigma} \tag{2.35}$$

$$= \rho \tag{2.36}$$

となり、

$$q_1 = -\frac{\rho D}{\sum_\sigma b_\sigma C_\sigma^2 d_\sigma} \tag{2.37}$$

および

$$h_1 = \frac{\rho D}{2(\sum_\sigma b_\sigma C_\sigma^2 d_\sigma)} \tag{2.38}$$

が得られる。

ここで、$f_i^{(eq)}$ を

$$f_i^{(eq)} = A_\sigma(1 + B_\sigma(e_{i\alpha} u_\alpha) + C_\sigma(e_{i\alpha} u_\alpha)^2 + D_\sigma u_\alpha^2) \tag{2.39}$$

と仮定すると、係数 $A_\sigma$, $B_\sigma$, $C_\sigma$ $B_\sigma$ は、

$$\sum_\sigma A_\sigma = \rho \tag{2.40}$$

$$C_\sigma = \frac{1}{2} B_\sigma^2 \tag{2.41}$$

$$D_\sigma = -\frac{1}{2} B_\sigma \tag{2.42}$$

を満たさなければならない。

2.2.2.2　マルチスケール展開

巨視的方程式を導くために、格子ボルツマン方程式に対し時間および空間に関する Chapman-Enskog マルチスケール展開を行う。

BGK 近似を用いた格子ボルツマン方程式 (2.17) に対して Taylor 展開を行い 3 次以降を無視すると、

$$\partial_t f_i + e_{i\alpha}\partial_\alpha f_i + \frac{1}{2}\partial_t^2 f_i + e_{i\alpha}\partial_t\partial_\alpha f_i + \frac{1}{2}e_{i\alpha}e_{i\beta}\partial_\alpha\partial_\beta f_i = -\frac{1}{\tau}(f_i - f_i^{(eq)}) \tag{2.43}$$

となる。

各演算子はマルチスケール展開により、

$$\partial_t = \varepsilon\partial_{t_1} + \varepsilon^2\partial_{t_2} \tag{2.44}$$

$$\partial_\alpha = \varepsilon\partial_\alpha \tag{2.45}$$

となる。ここで $\varepsilon$ は Knudesn 数と関係する微小係数である。

また、$f_i$ を $\varepsilon$ で展開し、2 次のオーダーまでとると、

$$f_i = \sum_{n=0}^{\infty}\varepsilon^n f_i^{(n)} = f_i^{(0)} + \varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)} \tag{2.46}$$

となる。ここで、$f_i^{(0)}$ は平衡分布関数であるから、

$$\sum_{i\sigma} f_i^{(n)} \begin{pmatrix} 1 \\ e_{i\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, n \geq 1. \tag{2.47}$$

が満たされている。

さらに BGK オペレータは、

$$\Omega_i = -\frac{1}{\tau}[\varepsilon f_i^{(1)} + \varepsilon^2 f_i^{(2)} + \cdots] \tag{2.48}$$

となる。

式 (2.46) を 式 (2.43) に代入すれば、

$\varepsilon^1$ オーダーに対して、

$$\partial_{t_1} f_i^{(0)} + e_{i\alpha}\partial_\alpha f_i^{(0)} = -\frac{1}{\tau}f_i^{(1)} \tag{2.49}$$

$\varepsilon^2$ オーダーに対して、

$$\partial_{t_1} f_i^{(1)} + \partial_{t_2} f_i^{(0)} + e_{i\alpha}\partial_\alpha f_i^{(1)} + \frac{1}{2}(\partial_{t_1} + e_{i\alpha}\partial_\alpha)^2 f_i^{(0)} = -\frac{1}{\tau}f_i^{(2)} \tag{2.50}$$

からなる方程式が導出される。

2.2.2.3　密度および運動量

式 (2.29) より、係数 $A$、$B$、$C$、$D$ の関係を求める。式 (2.29) の密度の定義は、

$$\sum_{i\sigma} f_i^{(eq)} = \rho \tag{2.51}$$

である。この左辺に、式 (2.39) を代入することによって、

$$\sum_{i\sigma} f_i^{(eq)} = \sum_{i\sigma} A_\sigma + \sum_{\sigma}(A_\sigma C_\sigma \frac{b_\sigma C_\sigma^2}{D})u_\alpha^2 + \sum_{i\sigma}(A_\sigma D_\sigma)u_\alpha^2 \tag{2.52}$$

$$= \rho \tag{2.53}$$

がえられる。これにより、以下の関係が導かれる。

$$A_0 + 6A_1 + 8A_2 = \rho \tag{2.54}$$

$$2A_1C_1 + 8A_2C_2 + A_0D_0 + 6A_1D_1 + 8A_2D_2 = 0 \tag{2.55}$$

運動量の定義は、

$$\sum_{i\sigma} e_{i\alpha} f_i^{(eq)} = \rho u_\alpha \tag{2.56}$$

である。式 (2.56) は密度と同様の手法により、

$$\sum_{i\sigma} e_{i\alpha} f_i^{(0)} = \sum_{\sigma} A_\sigma B_\sigma \frac{b_\sigma C_\sigma^2}{D} u_\alpha \tag{2.57}$$

$$= \rho u_\alpha \tag{2.58}$$

と変形できる。ここで、$b_\sigma$、$C_\sigma$、$D$ は各々 Type $\sigma$ の方向ベクトルの数、音速、格子の次元数である。これより、

$$2A_1B_1 + 8A_2B_2 = \rho \tag{2.59}$$

の関係が導かれる。

2.2.2.4 $\varepsilon^1$ オーダーの連続の式の導出

式 (2.49) に対して、$\sigma$ および $i$ で和をとると、

$$\partial_{t_1}\sum_{i\sigma} f_i^{(0)} + \partial_\alpha \sum_{i\sigma} e_{i\alpha} f_i^{(0)} = -\frac{1}{\tau}\sum_{i\sigma} f_i^{(1)} \tag{2.60}$$

となり、式 (2.29) 、(2.47) を式 (2.60) に代入する事により、$\varepsilon^1$ オーダーの連続の式が得られる。

$$\partial_{t_1}\rho + \partial_\alpha \rho u_\alpha = 0 \tag{2.61}$$

2.2.2.5 $\varepsilon^2$ オーダーの連続の式の導出

$\varepsilon^2$ オーダーの 格子ボルツマン方程式 (2.50) は

$$\partial_{t_2} f_i^{(0)} + (1 - \frac{1}{2\tau})(e_{i\alpha}\partial_\alpha + \partial_{t_1}) f_i^{(1)} = -\frac{1}{\tau} f_i^{(2)} \tag{2.62}$$

と書き換えられる。式 (2.62) に対し $\varepsilon^1$ の場合と同様に和をとると、

$$\partial_{t_2}\sum_{i\sigma} f_i^{(0)} + (1 - \frac{1}{2\tau})\sum_{i\sigma}\partial_{t_1} f_i^{(1)} + (1 - \frac{1}{2\tau})\sum_{i\sigma} e_{i\alpha}\partial_\alpha f_i^{(1)} = -\frac{1}{\tau}\sum_{i\sigma} f_i^{(2)} \tag{2.63}$$

となり、式 (2.29) 、(2.47) を用いると、$\varepsilon^2$ の連続の式

$$\partial_{t_2}\rho = 0 \tag{2.64}$$

が得られる。

2.2.2.6　$\varepsilon^1$ オーダーの運動方程式の導出

式（2.49）に $e_{i\alpha}$ を掛けて、連続の式と同様に和をとると、

$$\partial_{t_1}\sum_{i\sigma} f_i^{(0)} e_{i\alpha} + \sum_{i\sigma} \partial_\beta e_{i\beta} f_i^{(0)} e_{i\alpha} = -\frac{1}{\tau}\sum_{i\sigma} f_i^{(1)} e_{i\alpha} \tag{2.65}$$

となる。式（2.29）、（2.47）を用いて 式（2.65）を整理すると、

$$\partial_{t_1}(\rho u_\alpha) + \partial_\beta \sum_{i\sigma}(e_{i\alpha} e_{i\beta} f_i^{(0)}) = 0 \tag{2.66}$$

となり、運動量フラックステンソルを

$$\Pi_{\alpha\beta} = \sum_{i\sigma} e_{i\alpha} e_{i\beta} f_i \tag{2.67}$$

$$= \sum_n \sum_{i\sigma} e_{i\alpha} e_{i\beta} f_i^{(n)} \tag{2.68}$$

$$= \sum_n \Pi_{\alpha\beta}^{(n)} \tag{2.69}$$

と定義すると、式（2.66）は、

$$\partial_{t_1} \rho e_{i\alpha} + \partial_\beta \Pi_{\alpha\beta}^{(0)} = 0 \tag{2.70}$$

と表せる。

一方、$\varepsilon^1$ オーダーの巨視的方程式は

$$\partial_{t_1} \rho u_\alpha + \partial_\beta \rho u_\alpha u_\beta = -\partial_\beta p \delta_{\alpha\beta} \tag{2.71}$$

であるので、両者を比較する事により、

$$\Pi_{\alpha\beta}^{(0)} = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta \tag{2.72}$$

という関係が得られる。

ここで、$\Pi_{\alpha\beta}^{(0)}$ を式（2.39）を用いて展開すると、

$$\Pi_{\alpha\beta}^{(0)} = \sum_{i\sigma} e_{i\alpha} e_{i\beta} f_i^{(0)} \tag{2.73}$$

$$= (2A_1 + 8A_2)\delta_{\alpha\beta} + (2A_1 C_1 - 16A_2 C_2)\delta_{\alpha\beta} u_\alpha u_\beta + 16A_2 C_2 u_\alpha u_\beta \tag{2.74}$$
$$+(8A_2 C_2 + 2A_1 D_1 + 8A_2 D_2) u_\alpha^2 \delta_{\alpha\beta}$$

が得られる。式（2.72）を満たすためには式（2.74）の係数に対して、

$$2A_1 + 8A_2 = p \tag{2.75}$$

$$16A_2 C_2 = \rho \tag{2.76}$$

$$2A_1 C_1 - 16A_2 C_2 = 0 \tag{2.77}$$

$$8A_2 C_2 + 2A_1 D_1 + 8A_2 D_2 = 0 \tag{2.78}$$

が満たされなければならない。

2.2.2.7　$\varepsilon^2$ オーダーの運動方程式の導出

$e_{i\alpha}$ を式（2.50）に掛け、$\sigma$ および $i$ で和をとると、

$$\partial_{t_2}\sum_{i\sigma}e_{i\alpha}f_i^{(0)}+(1-\frac{1}{2\tau})\partial_{t_1}\sum_{i\sigma}e_{i\alpha}f_i^{(1)}+(1-\frac{1}{2\tau})\partial_\beta\sum_{i\sigma}e_{i\alpha}e_{i\beta}f_i^{(1)}+=-\frac{1}{\tau}\sum_{i\sigma}e_{i\alpha}f_i^{(2)} \quad (2.79)$$

となる。式（2.29）、（2.47）および（2.67）を用いると、式（2.79）は、

$$\partial f_{t_2}\rho u_{i\alpha}+(1-\frac{1}{2\tau})\partial f_\beta\Pi_{\alpha\beta}^{(1)}=0 \quad (2.80)$$

と計算できる。

一方、$\varepsilon^2$ オーダーの巨視的方程式は、

$$\partial_{t_2}\rho u_\alpha-\partial_\beta[\mu(\partial_\alpha u_\beta+\partial_\beta u_\alpha)]-\partial_\alpha(\lambda\partial f_\gamma u_\gamma)=0 \quad (2.81)$$

と記述される。ここで、$\lambda$ および $\mu$ は第二粘性係数、バルクな粘性係数である。

ここで、式（2.81）と（2.80）を比較する。

$$\Pi_{\alpha\beta}^{(1)}=-[\mu(\partial_\alpha u_\beta+\partial_\beta u_\alpha)]-(\lambda\partial_\gamma u_\gamma)\delta_{\alpha\beta} \quad (2.82)$$

が満たされれば、$\varepsilon^2$ オーダーの巨視的方程式を満足する事となる。

ここで、式（2.49）、（2.72）を用いると、$\Pi_{\alpha\beta}^{(1)}$ は

$$\Pi_{\alpha\beta}^{(1)} = \sum_{i\sigma}e_{i\alpha}e_{i\beta}f_i^{(1)} \quad (2.83)$$

$$= (-\tau)\partial_{t1}\{\rho u_\alpha u_\beta+p\delta_{\alpha\beta}\}+(-\tau)\partial_\gamma\sum_{i\sigma}e_{i\alpha}e_{i\beta}e_{i\gamma}f_i^{(0)} \quad (2.84)$$

と計算でき、さらに、式（2.26）、（2.39）、（2.61）および（2.72）を用いて、

$$\Pi_{\alpha\beta}^{(1)} = (-\tau)\{(a_2-C_s^2)u_\beta\partial_\alpha\rho+(a_2-C_s^2)u_\alpha\partial_\beta\rho+(a_2-C_s^2)\partial_\gamma(\rho u_\gamma) \quad (2.85)$$
$$+(a_1-2a_2)\partial_\gamma(\rho u_\gamma\delta_{\alpha\beta\gamma})+a_2\rho\partial_\alpha u_\beta+a_2\rho\partial_\beta u_\alpha\}$$

が得られる。ここで、

$$2A_1B_1 = a_1\rho \quad (2.86)$$

$$8A_2B_2 = a_2\rho \quad (2.87)$$

とした。式（2.82）を満たすためには、式（2.85）の係数は、

$$a_1 = 2a_2 \quad (2.88)$$

$$a_2 = C_s^2 \quad (2.89)$$

すなわち、

$$A_1B_1 = 8A_2B_2 \quad (2.90)$$

$$8A_2B_2 = C_s^2\rho \quad (2.91)$$

が満足されなくてはならない。

これにより、$\varepsilon^2$ オーダーの 格子ボルツマン方程式は、

$$\partial_{t_2}\rho u_\alpha + (\frac{1}{2}-\tau)\partial_\beta(C_s^2\rho)\{\partial_\beta u_\alpha + \partial_\alpha u_\beta\} = 0 \tag{2.92}$$

となる。

ここで、$\varepsilon^2$ オーダーの巨視的方程式 (2.81) と比較すると、バルクおよび第二粘性係数 $\mu$、$\lambda$ は

$$\mu = -(\frac{1}{2}-\tau)C_s^2\rho \tag{2.93}$$

$$\lambda = 0 \tag{2.94}$$

となる。

2.2.2.8　平衡分布関数の決定

前節までに得られた、平衡分布関数中の係数が満たす関係を示す。

$$\sum_\sigma A_\sigma = \rho \tag{2.95}$$

$$C_\sigma = \frac{1}{2}B_\sigma^2 \tag{2.96}$$

$$D_\sigma = -\frac{1}{2}B_\sigma \tag{2.97}$$

$$A_0 + 6A_1 + 8A_2 = \rho \tag{2.98}$$

$$2A_1C_1 + 8A_2C_2 + A_0D_0 + 6A_1D_1 + 8A_2D_2 = 0 \tag{2.99}$$

$$2A_1B_1 + 8A_2B_2 = \rho \tag{2.100}$$

$$2A_1 + 8A_2 = p \tag{2.101}$$

$$16A_2C_2 = \rho \tag{2.102}$$

$$2A_1C_1 - 16A_2C_2 = 0 \tag{2.103}$$

$$8A_2C_2 + 2A_1D_1 + 8A_2D_2 = 0 \tag{2.104}$$

$$A_1B_1 = 8A_2B_2 \tag{2.105}$$

$$8A_2B_2 = C_s^2\rho \tag{2.106}$$

ここで、式 (2.100)、(2.105)、(2.106) を用いる事により、音速は、

$$C_s^2 = \frac{1}{3} \tag{2.107}$$

と決定される。

さらに、式 (2.95) ～ (2.107) を用いて、

$$A_1B_1 = \frac{1}{3}\rho \tag{2.108}$$

$$A_2B_2 = \frac{1}{24}\rho \tag{2.109}$$

$$A_1 C_1 = \frac{1}{2}\rho \tag{2.110}$$

$$A_2 C_2 = \frac{1}{16}\rho \tag{2.111}$$

$$A_0 D_0 = -\frac{1}{3}\rho \tag{2.112}$$

$$A_1 D_1 = -\frac{1}{6}\rho \tag{2.113}$$

$$A_2 D_2 = -\frac{1}{48}\rho \tag{2.114}$$

が決定される。ここで、係数 $A_\sigma$ の値は、式（2.96）（2.108）～（2.114）を用ることにより得られ、

$$A_0 = \frac{2}{9}\rho \tag{2.115}$$

$$A_1 = \frac{1}{9}\rho \tag{2.116}$$

$$A_2 = \frac{1}{72}\rho \tag{2.117}$$

となる。

最終的な単相流 3D15V BGK モデルの平衡分布関数は以下のように決定される。

$$f_0^{(eq)} = \rho(\frac{2}{9} - \frac{1}{3}u^2) \quad , \quad \text{(type 0)} \tag{2.118}$$

$$f_i^{(eq)} = \rho[\frac{1}{9} + \frac{1}{3}(\boldsymbol{e}_i \cdot \boldsymbol{u}) + \frac{1}{2}(\boldsymbol{e}_i \cdot \boldsymbol{u})^2 - \frac{1}{6}u^2] \quad , i = 1, \cdots 6 \quad \text{(type 1)} \tag{2.119}$$

$$f_i^{(eq)} = \frac{\rho}{8}[\frac{1}{9} + \frac{1}{3}(\boldsymbol{e}_i \cdot \boldsymbol{u}) + \frac{1}{2}(\boldsymbol{e}_i \cdot \boldsymbol{u})^2 - \frac{1}{6}u^2] \quad , i = 7, \cdots 14 \quad \text{(type 2)} \tag{2.120}$$

## 2.3 二成分二相流モデル

### 2.3.1 概要

二成分二相流 に対する格子ボルツマン方程式は単相流モデル同様以下の式で表現される。

$$f_i^k(\boldsymbol{x} + \boldsymbol{e}_i, t+1) - f_i^k(\boldsymbol{x}, t) = \Omega_i^k(f_i(\boldsymbol{x}, t)) \tag{2.121}$$

ここで、$f_i^k(\boldsymbol{x}, t)$ 、$\Omega_i^k(f_i(\boldsymbol{x}, t))$ は $k$ 成分粒子（以下 $k$ は $r : red$ または $b : blue$ とする）の分布関数および衝突項である。全粒子の分布関数は

$$f_i = \sum_k f_i^k \tag{2.122}$$

で表される。

各流体の密度は

$$\rho_r = \sum_{i\sigma} f_i^r = \sum_{i\sigma} f_i^{r(eq)} \tag{2.123}$$

$$\rho_b = \sum_{i\sigma} f_i^b = \sum_{i\sigma} f_i^{b(eq)} \tag{2.124}$$

で定義され、全粒子密度、運動量は

$$\rho = \sum_{i\sigma k} f_i^k = \sum_{i\sigma k} f_i^{k(eq)} \tag{2.125}$$

$$\rho \boldsymbol{u} = \sum_k \sum_{i\sigma} f_i^k \boldsymbol{e}_i = \sum_k \sum_{i\sigma} f_i^{k(eq)} \boldsymbol{e}_i \tag{2.126}$$

である。ここで、$\rho_r$ 、$\rho_b$ は赤および青粒子の密度であり、全粒子密度は $\rho = \rho_r + \rho_b$ となる。また、$\boldsymbol{u}$ は流速である。

衝突項 $\Omega_i^k$ は 2 つの項に分けてモデル化される。

$$\Omega_i^k = \Omega_i^{k(BGK)} + \Omega_i^{k(SF)} \tag{2.127}$$

第 1 項目 $\Omega_i^{k(BGK)}$ は、単相流に対する格子ボルツマン方程式で用いた BGK オペレータ (2.18) であり、

$$\Omega_i^{k(BGK)} = -\frac{1}{\tau_k}[f_i^k(\boldsymbol{x},t) - f_i^{k(eq)}(\boldsymbol{x},t)] \tag{2.128}$$

と表される。

ここで、$f_i^{k(eq)}$ は $k$ 流体の平衡分布関数、$\tau_k$ は $k$ 流体の緩和時間である。

第 2 項目 $\Omega_i^{k(SF)}$、は 表面張力項であり、Grunau [9] により提唱されたモデルを用いる。

$$\Omega_i^{k(SF)} = A_k \mid \boldsymbol{F} \mid [\frac{(\boldsymbol{e}_i \cdot \boldsymbol{F})^2}{\mid \boldsymbol{e}_i \mid^2 \mid \boldsymbol{F} \mid^2} - G] \tag{2.129}$$

ここで、$A_k$ は $k$ 流体の表面張力の強さを表すパラメータであり、$G$ は粒子数保存、運動量保存を示す格子に依存する係数である。また、$\boldsymbol{F}$ は局所的な色の傾き (local color gradient) であり、

$$\boldsymbol{F} = \sum_i \boldsymbol{e}_i[\rho_r(\boldsymbol{x}+\boldsymbol{e}_i,t) - \rho_b(\boldsymbol{x}+\boldsymbol{e}_i,t)] \tag{2.130}$$

と定義されている。

これらをまとめると、二成分二相流に対する格子ボルツマン方程式は以下のように表現される。

$$f_i^k(\boldsymbol{x}+\boldsymbol{e}_i,t+1) - f_i^k(\boldsymbol{x},t) = -\frac{1}{\tau_k}[f_i^k(\boldsymbol{x},t) - f_i^{k(eq)}(\boldsymbol{x},t)] + A_k \mid \boldsymbol{F} \mid [\frac{(\boldsymbol{e}_i \cdot \boldsymbol{F})^2}{\mid \boldsymbol{e}_i \mid^2 \mid \boldsymbol{F} \mid^2} - G] \tag{2.131}$$

### 2.3.2 平衡分布関数と巨視的方程式

#### 2.3.2.1 平衡分布

二相流における $k$ 流体の平衡分布関数は、単相流同様 Maxwell-Boltzmann 分布を仮定する。単相流の場合の式 (2.39) と同様に、

$$f_i^{k(eq)} = A_\sigma^k(1 + B_\sigma^k(e_{i\alpha}u_\alpha) + C_\sigma^k(e_{i\alpha}u_\alpha)^2 + D_\sigma^k u_\alpha^2) \tag{2.132}$$

と表すと、$A_\sigma^k$、$B_\sigma^k$、$C_\sigma^k$、$D_\sigma^k$ は

$$\sum_\sigma A_\sigma^k = \rho_k \tag{2.133}$$

$$C_\sigma^k = \frac{1}{2}B_\sigma^{k2} \tag{2.134}$$

$$D_\sigma^k = -\frac{1}{2}B_\sigma^k \tag{2.135}$$

の関係をみたす。ここで、$f_i^{k(eq)}$ は

$$\sum_{i\sigma} f_i^{k(eq)} = \rho_k \tag{2.136}$$

及び

$$\sum_{i\sigma k} f_i^{k(eq)} \begin{pmatrix} 1 \\ e_{i\alpha} \end{pmatrix} = \begin{pmatrix} \rho \\ \rho u_\alpha \end{pmatrix} \tag{2.137}$$

を満たしている。$\rho u_\alpha$ 、$\rho_k$、$\rho$ は、各々運動量、$k$ 粒子および全粒子の密度である。

### 2.3.3 マルチスケール展開

単相流で用いたマルチスケール展開の手法を用いて、式（2.131）から出発して、巨視的方程式の導出を行う。

各微分演算子は、微小パラメータ $\varepsilon$ を用いて

$$\partial_t = \varepsilon\partial_{t_1} + \varepsilon^2\partial_{t_2} \tag{2.138}$$

$$\partial_\alpha = \varepsilon\partial_\alpha \tag{2.139}$$

と展開でき、さらに 式（2.131）を Taylor 展開すると、

$$\begin{aligned} &\partial_t f_i^k + e_{i\alpha}\partial_\alpha f_i^k + \tfrac{1}{2}\partial_t^2 f_i^k + e_{i\alpha}\partial_t\partial_\alpha f_i^k + \tfrac{1}{2}e_{i\alpha}e_{i\beta}\partial_\alpha\partial_\beta f_i^k \\ = \quad &-\tfrac{1}{\tau_k}(f_i^k - f_i^{k(eq)}) + A_k \mid \boldsymbol{F} \mid [\tfrac{(\boldsymbol{e}_i\cdot\boldsymbol{F})^2}{|\boldsymbol{e}_i|^2|\boldsymbol{F}|^2} - G] \end{aligned} \tag{2.140}$$

となる。また、$f_i^k$ は

$$f_i^k = \sum_{n=0}^{\infty} \varepsilon^n f_i^{k(n)} = f_i^{k(0)} + \varepsilon f_i^{k(1)} + \varepsilon^2 f_i^{k(2)} + \cdots, \tag{2.141}$$

となり、

$$\sum_{i\sigma} f_i^{k(n)} \begin{pmatrix} 1 \\ e_{i\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, n \geq 1. \tag{2.142}$$

を満たす。

衝突項は

$$\Omega_i^k = -\frac{1}{\tau_k}[\varepsilon f_i^{k(1)} + \varepsilon^2 f_i^{k(2)} + \cdots] + \Omega_i^{k(SF)} \tag{2.143}$$

と表される。

式（2.121）、（2.127）、（2.141）を用いて $\varepsilon^1$ および $\varepsilon^2$ オーダーの 二成分二相流格子ボルツマン方程式が得られる。

$$\partial_{t_1} f_i^{k(0)} + e_{i\alpha}\partial_\alpha f_i^{k(0)} = -\frac{1}{\tau_k} f_i^{k(1)} + \Omega_i^{k(SF)} \tag{2.144}$$

$$\partial_{t_1} f_i^{k(1)} + \partial_{t_2} f_i^{k(0)} + e_{i\alpha}\partial_\alpha f_i^{k(1)} + \frac{1}{2}(\partial_{t_1} + e_{i\alpha}\partial_\alpha)^2 f_i^{k(0)} = -\frac{1}{\tau_k} f_i^{k(2)} \tag{2.145}$$

### 2.3.4 密度および運動量

式（2.136）および（2.137）から、二相流における平衡分布関数の係数および、表面張力項の係数（$A^k$, $B^k$, $C^k$, $D^k$, $G$）の関係を求める。

$k$ 流体および全体の密度は、式（2.136）によって定義されている。

$$\sum_{i\sigma} f_i^{k(0)} = \rho_k \tag{2.146}$$

$$\sum_{k} \rho_k = \sum_{i\sigma k} f_i^{k(0)} = \rho \tag{2.147}$$

式（2.132）を式（2.146）の左辺に代入し、式（2.8），（2.9），（2.11）（2.12）を用いると、

$$\sum_{i\sigma} f_i^{k(0)} = \sum_{i\sigma} A_\sigma^k + \sum_{\sigma}(A_\sigma^k C_\sigma^k \frac{b_\sigma C_\sigma^2}{D})u_\alpha^2 + \sum_{i\sigma}(A_\sigma^k D_\sigma^k)u_\alpha^2 \tag{2.148}$$

$$= \rho_k \tag{2.149}$$

の関係が得られ、これにより、

$$A_0^k + 6A_1^k + 8A_2^k = \rho_k \tag{2.150}$$

$$\sum_{k}(2A_1^k C_1^k + 8A_2^k C_2^k + A_0^k D_0^k + 6A_1^k D_1^k + 8A_2^k D_2^k) = 0 \tag{2.151}$$

および

$$\sum_{k}(A_0^k + 6A_1^k + 8A_2^k) = \rho \tag{2.152}$$

の関係が導かれる。

運動量の定義は、

$$\sum_{i\sigma k} e_{i\alpha} f_i^{k(0)} = \rho u_\alpha \tag{2.153}$$

であり、密度と同様の手法により、

$$\sum_{i\sigma k} e_{i\alpha} f_i^{k(0)} = \sum_{\sigma k} A_\sigma^k B_\sigma^k \frac{b_\sigma C_\sigma^2}{D} u_\alpha \tag{2.154}$$

$$= \rho u_\alpha \tag{2.155}$$

となる。これより、

$$\sum_{k}(2A_1^k B_1^k + 8A_2^k B_2^k) = \rho \tag{2.156}$$

の関係が導かれる。

2.3.5 $\varepsilon^1$ オーダーの連続の式の導出

$\varepsilon^1$ オーダーの 密度 $\rho$ および 運動量 $\rho u_\alpha$ の関係は、式 (2.144) を $\sigma, i$ および $k$ で和をとる事により得られる。

$$\partial_{t_1}\sum_{i\sigma k} f_i^{k(0)} + \partial_\alpha \sum_{i\sigma k} e_{i\alpha} f_i^{k(0)} = -\sum_k (\frac{1}{\tau_k}\sum_{i\sigma} f_i^{k(1)}) + \sum_k \Omega_i^{k(SF)} \tag{2.157}$$

式 (2.157) に式 (2.137) および (2.142) を代入する事により、

$$\partial_{t_1}\rho + \partial_\alpha \rho u_\alpha = \sum_{i\sigma k} \Omega_i^{k(SF)} \tag{2.158}$$

となる。

$\varepsilon^1$ オーダーの連続の式は

$$\partial_{t_1}\rho + \partial_\alpha \rho u_\alpha = 0 \tag{2.159}$$

であるから、質量保存を満たすためには、$\Omega_i^{k(SF)}$ について、

$$\sum_{i\sigma k} \Omega_i^{k(SF)} = 0 \tag{2.160}$$

が満たされなければならない。

ここで、$\Omega_i^{k(SF)}$ は

$$\sum_{i\sigma k} \Omega_i^{k(SF)} = \sum_{i\sigma k} A_k \mid \boldsymbol{F} \mid [\frac{(\boldsymbol{e}_i \cdot \boldsymbol{F})^2}{\mid \boldsymbol{e}_i \mid^2 \mid \boldsymbol{F} \mid^2} - G] \tag{2.161}$$

$$= \sum_k \{A_k \mid \boldsymbol{F} \mid (\frac{15}{3} - 15G)\} \tag{2.162}$$

と計算され、式 (2.162) は 任意の $k$ において成り立たなければならないから、係数 $G$ は、

$$G = \frac{1}{3} \tag{2.163}$$

と決定される。

$\varepsilon^1$ オーダーの格子ボルツマン方程式の質量保存は、

$$\partial_{t_1}\rho + \partial_\alpha \rho u_\alpha = 0 \tag{2.164}$$

を満たす事となる。

2.3.6 $\varepsilon^2$ オーダーの連続の式の導出

格子ボルツマン方程式の $\varepsilon^2$ オーダーは

$$\partial_{t_1} f_i^{k(1)} + \partial_{t_2} f_i^{k(0)} + e_{i\alpha}\partial_\alpha f_i^{k(1)} + \frac{1}{2}(\partial_{t_0} + e_{i\alpha}\partial_\alpha)^2 f_i^{k(0)} = -\frac{1}{\tau} f_i^{k(2)} \tag{2.165}$$

である。

式 (2.165) に対して 和をとり、式 (2.136) 、 (2.142) 、 (2.144) 、 (2.160) を利用して変形すると、$\varepsilon^2$ オーダーの式

$$\partial_{t_2}\rho + \frac{1}{2}\partial_\alpha \sum_{i\sigma k} e_{i\alpha}\Omega_i^{k(SF)} = 0 \tag{2.166}$$

が得られる。ここで、第2項は

$$\begin{aligned}
\sum_{i\sigma} e_{i\alpha}\Omega_i^{k(SF)} &= \sum_{i\sigma} A_k \mid \boldsymbol{F} \mid e_{i\alpha}[\frac{e_{i\beta}e_{i\gamma}F_\beta F_\gamma}{\mid \boldsymbol{e}_i \mid^2 F^2} - G] & (2.167)\\
&= \frac{A_k}{\mid \boldsymbol{F} \mid}\sum_{i\sigma}\{\frac{1}{C_\sigma^2}(e_{i\alpha}e_{i\beta}e_{i\gamma}F_\beta F_\gamma)\} - A \mid \boldsymbol{F} \mid \sum_{i\sigma} e_{i\alpha}G & (2.168)\\
&= 0 & (2.169)
\end{aligned}$$

であるから、係数 $G$ によらず 0 となる。

以上より、$\varepsilon^2$ オーダーの連続の式

$$\partial_{t_2}\rho = 0 \tag{2.170}$$

が得られる。

また、$\Omega_i^{k(SF)}$ は

$$\sum_{i\sigma k}\Omega_i^{k(SF)}\begin{pmatrix} 1 \\ e_{i\alpha} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{2.171}$$

を満足する。言い換えると、表面張力項のモデル化は、式 (2.171) を満足するようにモデル化されていると考える事が出来る。

### 2.3.7 $\varepsilon^1$ オーダーの運動方程式の導出

式 (2.144) に $e_{i\alpha}$ を乗じて和をとり、式 (2.136) , (2.142) 、 (2.171) を利用して、整理すると

$$\partial_{t_1}\rho u_\alpha + \partial_\beta(\sum_k \Pi_{\alpha\beta}^{(0)}) = 0 \tag{2.172}$$

が得られる。ここで、$\Pi_{\alpha\beta}^{(n)} = \sum_{i\sigma} e_{i\alpha}e_{i\beta}f_i^{k(n)}$ であり、運動量テンソルを表す。

$\varepsilon^1$ オーダーの巨視的方程式は、

$$\partial_{t_1}\rho u_\alpha + \partial_\beta \rho u_\alpha u_\beta = -\partial_\beta p\delta_{\alpha\beta} \tag{2.173}$$

である。

式 (2.173) と (2.172) を比較する事により、$\varepsilon^1$ オーダーの運動方程式は、

$$\sum_k(\Pi_{\alpha\beta}^{(0)}) = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta \tag{2.174}$$

を満たす事が導かれる。

$\Pi_{\alpha\beta}^{(0)}$ は、

$$\begin{aligned}
\Pi_{\alpha\beta}^{(0)} &= \sum_{i\sigma} e_{i\alpha}e_{i\beta}f_i^{k(0)} & (2.175)\\
&= (2A_1^k + 8A_2^k)\delta_{\alpha\beta} + (2A_1^kC_1^k - 16A_2^kC_2)\delta_{\alpha\beta}u_\alpha u_\beta + 16A_2^kC_2^k u_\alpha u_\beta & (2.176)\\
&\quad +(8A_2^kC_2^k + 2A_1^kD_1^k + 8A_2^kD_2^k)u^2\delta_{\alpha\beta}
\end{aligned}$$

となり、式（2.174）の条件から、式（2.175）の係数は以下の関係を満たさなければならない。

$$\sum_k (2A_1^k + 8A_2^k) = p \tag{2.177}$$

$$\sum_k 16A_2^k C_2^k = \rho \tag{2.178}$$

$$\sum_k (2A_1^k C_1^k - 16A_2^k C_2^k) = 0 \tag{2.179}$$

$$\sum_k (8A_2^k C_2^k + 2A_1^k D_1^k + 8A_2^k D_2^k) = 0 \tag{2.180}$$

2.3.8 $\varepsilon^2$ オーダーの運動方程式の導出

$e_{i\alpha}$ を式（2.145）に乗じて、$\sigma$、$i$ および $k$ で和をとり、式（2.136），（2.142），（2.160）および、（2.172）を用いて変形すると、

$$\sum_k [(1 - \frac{1}{2\tau_k})\partial_\beta \Pi_{\alpha\beta}^{(1)} + \frac{1}{2} T_{\alpha\beta}] + \partial_{t_2} \rho u_\alpha = 0 \tag{2.181}$$

となる。$T_{\alpha\beta}$ は次の様に定義される。

$$T_{\alpha\beta} = \sum_{i\sigma} e_{i\alpha} e_{i\beta} \Omega_i^{k(SF)} \tag{2.182}$$

一方、$\varepsilon^2$ オーダーの巨視的方程式は、

$$\partial_{t_2} \rho u_\alpha - \partial_\beta [\mu(\partial_\alpha u_\beta + \partial_\beta u_\alpha)] - \partial_\alpha (\lambda \partial_\gamma u_\gamma) + \partial_\alpha P_{\alpha\beta} = 0 \tag{2.183}$$

である。ここで、$\lambda$、$\mu$ は 第二粘性係数およびバルクな粘性係数であり、また、$P_{\alpha\beta}$ は圧力テンソルである。

式（2.183） と （2.181） を比較する事により、$\varepsilon^2$ オーダーに関する以下の関係が得られる。

$$\sum_k \Pi_{\alpha\beta}^{(1)} = -[\mu(\partial_\alpha u_\beta + \partial_\beta u_\alpha)] - (\lambda \partial_\gamma u_\gamma)\delta_{\alpha\beta} - P_{\alpha\beta} \tag{2.184}$$

式（2.144）を用いて、$\Pi_{\alpha\beta}^{(1)}$ を変形すると、

$$\Pi_{\alpha\beta}^{(1)} = \sum_{i\sigma} e_{i\alpha} e_{i\beta} f_i^{k(1)} \tag{2.185}$$

$$= (-\tau_k)[\partial_\gamma \sum_{i\sigma} e_{i\alpha} e_{i\beta} e_{i\gamma} f_i^{k(0)} + \partial_{t_1} \Pi_{\alpha\beta}^{(0)} - T_{\alpha\beta}] \tag{2.186}$$

となり、さらに、式（2.132） を用いて、

$$\Pi_{\alpha\beta}^{(1)} = (-\tau_k)[\partial_\beta (2A_1^k B_1^k - 16A_2^k B_2^k)\delta_{\alpha\beta} u_\beta + \partial_\gamma 8A_2^k B_2^k \delta_{\alpha\beta} u_\gamma + \partial_\beta 8A_2^k B_2^k u_\alpha + \partial_\alpha 8A_2^k B_2^k u_\beta + \partial_{t_1} \Pi_{\alpha\beta}^{(0)} - T_{\alpha\beta}] \tag{2.187}$$

を得る。ここで、式（2.187）に関して $k$ で和をとるわけであるが、$k$ 流体に関する緩和時間パラメータは後述するように、ある時刻、ある位置おいては、赤粒子、青粒子とも密度に依存する

$\tau(r,b)$ で表される同一の時間パラメータを用いる。このため、$\tau_k = \tau(r,b)$ とでき、$k$ の和の外に出すことができる。

式 (2.187) を $k$ で和をとり、式 (2.164) 、(2.170) 、(2.172) 、(2.174) を用いると、$\Pi^{(1)}_{\alpha\beta}$ は、

$$\begin{aligned}(\frac{1}{2}-\tau(r,b))\partial_\beta\{\partial_\gamma(8a_2-{C_s}^2)\rho u_\gamma\delta_{\alpha\beta}+\partial_\beta(2a_1-16a_2)\rho u_\beta\delta_{\alpha\beta}+u_\alpha\partial_\beta(8a_2-{C_s}^2)\rho \quad (2.188)\\ +u_\beta\partial_\alpha(8a_2-{C_s}^2)\rho+8a_2\rho\partial_\beta u_\alpha+8a_2\rho\partial_\alpha u_\alpha\}+\partial_{t_1}\rho u_\alpha+\tau(r,b)\partial_\beta T_{\alpha\beta} \quad =0\end{aligned}$$

となる。ここで、

$$\sum_k A_1 B_1 = a_1\rho \quad (2.189)$$

$$\sum_k A_2 B_2 = a_2\rho \quad (2.190)$$

とした。式 (2.183) を満足するためには、式 (2.188) の係数は、

$$a_1 = 8a_2 \quad (2.191)$$

$$8a_2 = {C_s}^2 \quad (2.192)$$

すなわち、

$$\sum_k A_1 B_1 = \sum_k 8A_2 B_2 \quad (2.193)$$

$$\sum_k 8A_2 B_2 = {C_s}^2\rho \quad (2.194)$$

を満たさなければならない。

以上より、$\varepsilon^2$ オーダーの巨視的方程式

$$\partial_{t_2}\rho u_\alpha+\partial_\alpha\tau(r,b)T_{\alpha\beta}+(\frac{1}{2}-\tau(r,b))\partial_\beta(C_s^2\rho)\{\partial_\beta u_\alpha+\partial_\alpha u_\beta\}=0 \quad (2.195)$$

が導出された。

バルク粘性係数 $\mu$ および第二粘性係数 $\lambda$ は、

$$\mu = -(\frac{1}{2}-\tau(r,b))C_s^2\rho \quad (2.196)$$

$$\lambda = 0 \quad (2.197)$$

となる。

### 2.3.9 平衡分布関数の決定

全節までに得られた係数の関係をまとめると、

$$\sum_\sigma A^k_\sigma = \rho_k \quad (2.198)$$

$$C^k_\sigma = \frac{1}{2}{B^k_\sigma}^2 \quad (2.199)$$

$$D_\sigma^k = -\frac{1}{2}B_\sigma^k \quad (2.200)$$

$$A_0^k + 6A_1^k + 8A_2^k = \rho_k \quad (2.201)$$

$$\sum_k (2A_1^k C_1^k + 8A_2^k C_2^k + A_0^k D_0^k + 6A_1^k D_1^k + 8A_2^k D_2^k) = 0 \quad (2.202)$$

$$\sum_k (A_0^k + 6A_1^k + 8A_2^k) = \rho \quad (2.203)$$

$$\sum_k (2A_1^k B_1^k + 8A_2^k B_2^k) = \rho \quad (2.204)$$

$$G = \frac{1}{3} \quad (2.205)$$

$$\sum_k (2A_1^k + 8A_2^k) = p \quad (2.206)$$

$$\sum_k 16A_2^k C_2^k = \rho \quad (2.207)$$

$$\sum_k (2A_1^k C_1^k - 16A_2^k C_2^k) = 0 \quad (2.208)$$

$$\sum_k (8A_2^k C_2^k + 2A_1^k D_1^k + 8A_2^k D_2^k) = 0 \quad (2.209)$$

$$\sum_k A_1 B_1 = 8A_2 B_2 \quad (2.210)$$

$$\sum_k 8A_2 B_2 = C_s^2 \rho \quad (2.211)$$

となる。

ここで、二相流動現象において一方の流体のみが存在する領域においても、上記の関係が成り立っていなければならない。そのため、式（2.198）～（2.211）の関係式は、以下の関係も満たしている。

$$\sum_\sigma A_\sigma^k = \rho_k \quad (2.212)$$

$$C_\sigma^k = \frac{1}{2}B_\sigma^{k2} \quad (2.213)$$

$$D_\sigma^k = -\frac{1}{2}B_\sigma^k \quad (2.214)$$

$$A_0^k + 6A_1^k + 8A_2^k = \rho_k \quad (2.215)$$

$$(2A_1^k C_1^k + 8A_2^k C_2^k + A_0^k D_0^k + 6A_1^k D_1^k + 8A_2^k D_2^k) = 0 \quad (2.216)$$

$$(2A_1^k B_1^k + 8A_2^k B_2^k) = \rho_k \quad (2.217)$$

$$G = \frac{1}{3} \quad (2.218)$$

$$(2A_1^k + 8A_2^k) = p_k \quad (2.219)$$

$$16A_2^k C_2^k = \rho_k \quad (2.220)$$

$$(2A_1^k C_1^k - 16A_2^k C_2^k) = 0 \quad (2.221)$$

$$(8A_2^k C_2^k + 2A_1^k D_1^k + 8A_2^k D_2^k) = 0 \quad (2.222)$$

$$A_1 B_1 = 8A_2 B_2 \quad (2.223)$$

$$8A_2 B_2 = C_s^2 \rho_k \quad (2.224)$$

2.212 ～ 2.224 より、

$$A_1^k B_1^k = \frac{1}{3}\rho_k \tag{2.225}$$

$$A_2^k B_2^k = \frac{1}{24}\rho_k \tag{2.226}$$

$$A_1^k C_1^k = \frac{1}{2}\rho_k \tag{2.227}$$

$$A_2^k C_2^k = \frac{1}{16}\rho_k \tag{2.228}$$

$$A_0^k D_0^k = -\frac{1}{3}\rho_k \tag{2.229}$$

$$A_1^k D_1^k = -\frac{1}{6}\rho_k \tag{2.230}$$

$$A_2^k D_2^k = -\frac{1}{48}\rho_k \tag{2.231}$$

の関係を得る。

ここで、係数 $A_\sigma$ は 式 (2.212) の関係を満たすように決定すれば良い。本モデルでは、$k$ 流体の静止粒子の割合を示す実数パラメータ $\lambda_k$ を導入し、

$$A_0 = \frac{\lambda_k}{7+\lambda_k}\rho_k \tag{2.232}$$

$$A_1 = \frac{1}{7+\lambda_k}\rho_k \tag{2.233}$$

$$A_2 = \frac{1}{8}\left(\frac{1}{7+\lambda_k}\rho_k\right) \tag{2.234}$$

と仮定する。

これにより、音速は

$$(C_s^k)^2 = \frac{3}{7+\lambda_k} \tag{2.235}$$

と決定される。

以上より、二相流格子ボルツマン方程式の 平衡分布関数は

$$f_0^{k(eq)} = \rho_k\left(\frac{\lambda_k}{7+\lambda_k} - \frac{1}{3}u^2\right) \quad , \quad \text{(type 0)} \tag{2.236}$$

$$f_i^{k(eq)} = \rho_k\left[\frac{1}{7+\lambda_k} + \frac{1}{3}(\boldsymbol{e}_i\cdot\boldsymbol{u}) + \frac{1}{2}(\boldsymbol{e}_i\cdot\boldsymbol{u})^2 - \frac{1}{6}u^2\right] \quad , i=1,\cdots 6 \quad \text{(type 1)} \tag{2.237}$$

$$f_i^{k(eq)} = \frac{\rho_k}{8}\left[\frac{1}{7+\lambda_k} + \frac{1}{3}(\boldsymbol{e}_i\cdot\boldsymbol{u}) + \frac{1}{2}(\boldsymbol{e}_i\cdot\boldsymbol{u})^2 - \frac{1}{6}u^2\right] \quad , i=7,\cdots 14 \quad \text{(type 2)} \tag{2.238}$$

と決定される。

### 2.3.10 巨視的方程式との対応

#### 2.3.10.1 連続の式

格子ボルツマン方程式における $\varepsilon^1$ および $\varepsilon^2$ の連続の式 (2.164)、(2.170) の和をとると、

$$\varepsilon\partial_{t_1}\rho + \varepsilon\partial_\alpha \rho u_\alpha + \varepsilon^2\partial_{t_2}\rho = 0 \tag{2.239}$$

となる。マルチスケール展開の演算子は、式（2.138）、（2.139）で定義されており、

$$\partial_t = \varepsilon\partial_{t_1} + \varepsilon^2\partial_{t_2} \tag{2.240}$$

$$\partial_\alpha = \varepsilon\partial_\alpha \tag{2.241}$$

であるから、式（2.239）は、

$$\partial_t\rho + \partial_\alpha\rho u_\alpha = 0 \tag{2.242}$$

となり、巨視的な連続の式と一致する。

2.3.10.2　運動方程式

$\varepsilon^1$ オーダー と $\varepsilon^2$ オーダーの 運動量に関する格子ボルツマン方程式（2.172）、（2.195）の和をとり、マルチスケール展開の演算子の定義、式(2.138)、(2.139)を用いると、

$$\partial_t\rho u_\alpha + \partial_\beta\{\sum_k \Pi^{(0)}_{\alpha\beta} + (\frac{1}{2} - \tau(r,b))C_s^2\rho(\partial_\alpha u_\beta + \partial_\beta u_\alpha) + \tau(r,b)T_{\alpha\beta}\} = 0 \tag{2.243}$$

を得る。

二相流に対応する巨視的な運動方程式は

$$\partial_t\rho u_\alpha + \partial_\beta\rho u_\alpha u_\beta = \partial_\beta(-P_{\alpha\beta}) + \partial_\beta[\mu(\partial_\alpha u_\beta + \partial_\beta u_\alpha)] \tag{2.244}$$

であるから、関係式（2.174）

$$\sum_k(\Pi^{(0)}_{\alpha\beta}) = p\delta_{\alpha\beta} + \rho u_\alpha u_\beta \tag{2.245}$$

および（2.196）

$$\mu = -(\frac{1}{2} - \tau(r,b))C_s^2\rho \tag{2.246}$$

$$\lambda = 0 \tag{2.247}$$

を考慮し、（2.243）と（2.244）の両者を比較することにより、圧力テンソル $P_{\alpha\beta}$ の関係

$$P_{\alpha\beta} = -p\delta_{\alpha\beta} + (-\tau(r,b))T_{\alpha\beta} \tag{2.248}$$

が得られる。

2.3.11　表面張力

格子ボルツマン方程式から求められる二相流動現象に対応する方程式は

$$\partial_t\rho u_\alpha + \partial_\beta\{\sum_k \Pi^{(0)}_{\alpha\beta} + (\frac{1}{2} - \tau(r,b))C_s^2\rho(\partial_\alpha u_\beta + \partial_\beta u_\alpha) + \sum_k \tau(r,b)T_{\alpha\beta}\} = 0 \tag{2.249}$$

であり、$\Pi^{(0)}_{\alpha\beta}$ は、

$$\Pi^{(0)}_{\alpha\beta} = -p\delta_{\alpha\beta} + \rho_\alpha u_\alpha u_\beta \tag{2.250}$$

である。

また、圧力テンソル $P_{\alpha\beta}$ は、

$$P_{\alpha\beta} = -p\delta_{\alpha\beta} + \sum_k (-\tau(r,b))T_{\alpha\beta} \tag{2.251}$$

と定義され、$T_{\alpha\beta} = \sum_{i\sigma} e_{i\alpha}e_{i\beta}\Omega_i^{k(SF)}$. である。

さらに、表面張力項 $\Omega_i^{k(SF)}$ は

$$\Omega_i^{k(SF)} = A_k \mid \boldsymbol{F} \mid [\frac{(\boldsymbol{e}_i \cdot \boldsymbol{F})^2}{\mid \boldsymbol{e}_i \mid^2 \mid \boldsymbol{F} \mid^2} - \frac{1}{3}] \tag{2.252}$$

となる。

ここで、緩和時間 $\tau_k = \tau(r,b)$ は、赤および青流体の混合状態から決定される。一般には界面の厚さは、平均緩和時間と粒子密度に依存しており、青粒子領域から赤粒子領域へと、あるは赤粒子領域から青粒子領域へ移るにしたがって、その界面で粘性がスムーズに変化して行くモデルが幾つか提唱されている。

これを実現するために各々の密度に依存する 粒子オーダーパラメータ $\psi$ （Atwood 数）を導入する。

$$\psi = \frac{\rho_r - \rho_b}{\rho_r + \rho_b} \tag{2.253}$$

通常は、$\mid \psi \mid \leq 1$ を仮定し、赤粒子のみ、あるいは 青粒子のみの状態では、$\psi = 1$ または $\psi = -1$ となる。緩和時間係数を滑らかに接続するために、以下のような手法を用いる。

$$\tau = \begin{cases} \tau_r, & \psi > \delta \\ g_r(\psi), & \delta \geq \psi > 0 \\ g_b(\psi), & 0 \geq \psi \geq -\delta \\ \tau_b, & -\delta > \psi \end{cases} \tag{2.254}$$

ここで、$g_r(\psi)$、$g_b(\psi)$ は $\psi$ の 2 次オーダーの関数であり、

$$g_r(\psi) = \alpha + \beta\psi + \kappa\psi^2 \tag{2.255}$$

$$g_b(\psi) = \alpha + \eta\psi + \zeta\psi^2 \tag{2.256}$$

と定義される。$\mid \psi \mid = \delta$ における境界条件を $g_r(\delta) = \tau_r$、$g_b(-\delta) = \tau_b$、$\frac{\partial\tau}{\partial\psi} = 0$ および、$g_r(0) = g_b(0) = \langle\tau\rangle$ とすると、

$$\alpha = 2\frac{\tau_r\tau_b}{\tau_r + \tau_b} \tag{2.257}$$

$$\beta = 2\frac{\tau_r - \alpha}{\delta} \tag{2.258}$$

$$\kappa = -\frac{\beta}{2\delta} \tag{2.259}$$

$$\eta = 2\frac{\alpha - \tau_b}{\delta} \tag{2.260}$$

$$\zeta = \frac{\eta}{2\delta} \tag{2.261}$$

となる。ここで、$\delta \leq 1$ は界面の厚さに関するフリーパラメータ、$\langle\tau\rangle$ は平均緩和時間 $\langle\tau\rangle = 2\tau_r\tau_b/(\tau_r + \tau_b)$ である。

いま、圧力テンソル $P_{\alpha\beta}$ は、式 (2.251) および (2.252) をもちいて、

$$P_{\alpha\beta} = p\delta_{\alpha\beta} + \tau \sum_k \sum_{i\sigma} e_{i\alpha} e_{i\beta} A_k \mid \boldsymbol{F} \mid \{ \frac{(e_{i\alpha} F_\alpha)^2}{e_{i\alpha}^2 F^2} - \frac{1}{3} \} \tag{2.262}$$

となる。ここで、$p$ はバルク圧力である。

$T_{\alpha\beta}$ の計算は、

$$\begin{aligned}
T_{\alpha\beta} &= \sum_{i\sigma} e_{i\alpha} e_{i\beta} \Omega_i^{k(SF)} && (2.263)\\
&= \sum_{i\sigma} e_{i\alpha} e_{i\beta} A \mid \boldsymbol{F} \mid \{ \frac{(e_{i\gamma} F_\gamma)^2}{e_{i\gamma}^2 F^2} - \frac{1}{3} \} && (2.264)\\
&= \sum_{i\sigma} \frac{A}{\mid \boldsymbol{F} \mid C_\sigma^2} [e_{i\alpha} e_{i\beta} e_{i\gamma} e_{i\theta} F_\gamma F_\theta] - \frac{1}{3} A \mid \boldsymbol{F} \mid \sum_{i\sigma} e_{i\alpha} e_{i\beta} && (2.265)\\
&= \frac{A}{\mid \boldsymbol{F} \mid} [(2 - \frac{16}{3}) F_\gamma F_\theta \delta_{\alpha\beta\gamma\theta} + \frac{8}{3} \Delta_{\alpha\beta\gamma\theta} F_\gamma F_\theta] - \frac{1}{3} A \mid \boldsymbol{F} \mid \frac{b_\sigma C_\sigma^2}{D} \delta_{\alpha\beta} && (2.266)\\
&= \frac{A}{\mid \boldsymbol{F} \mid} [(-\frac{10}{3}) F_\beta F_\beta \delta_{\alpha\beta} + \frac{8}{3} \delta_{\alpha\beta} F^2 + \frac{8}{3} F_\alpha F_\beta + \frac{8}{3} F_\alpha F_\beta] - \frac{10}{3} A \mid \boldsymbol{F} \mid \delta_{\alpha\beta} && (2.267)\\
&= \frac{A}{\mid \boldsymbol{F} \mid} [\frac{16}{3} F_\alpha F_\beta - \frac{10}{3} F_\beta F_\beta \delta_{\alpha\beta}] - \frac{2}{3} A \mid \boldsymbol{F} \mid \delta_{\alpha\beta} && (2.268)
\end{aligned}$$

となり、圧力テンソルは、

$$P_{\alpha\beta} = p\delta_{\alpha\beta} + \tau \sum_k A \mid \boldsymbol{F} \mid [\frac{F_\alpha F_\beta}{\mid \boldsymbol{F} \mid^2} (\frac{16}{3} - \frac{10}{3} \delta_{\alpha\beta}) - \frac{2}{3} \delta_{\alpha\beta}] \tag{2.269}$$

となる。ただし、パラメータ $A_k$ については、$A_r = A_b = \frac{1}{2} A$ とした事に注意する。

$\boldsymbol{F}$ は local color gradient であり、

$$\boldsymbol{F} = \sum_{i\sigma} \boldsymbol{e}_i (\rho_r(\boldsymbol{x} + \boldsymbol{e}_i, t) - \rho_b(\boldsymbol{x} + \boldsymbol{e}_i, t)) \tag{2.270}$$

と定義されている。

式 (2.270) を Taylor 展開し 3 次以上の項を無視すると、

$$\begin{aligned}
F_\alpha &= \sum_{i\sigma} e_{i\gamma} \{ \rho_r + \partial_\alpha e_{i\alpha} \rho_r + \frac{1}{2} \partial_\beta \partial_\alpha e_{i\alpha} e_{i\beta} \rho_r - (\rho_b + \partial_\alpha e_{i\alpha} \rho_b + \frac{1}{2} \partial_\beta \partial_\alpha e_{i\alpha} e_{i\beta} \rho_b) \} && (2.271)\\
&= \partial_\alpha 10 (\rho_r - \rho_b) && (2.272)
\end{aligned}$$

が得られる。

一方、表面張力の定義は、

$$\sigma = \int (P_n - P_t) dx \tag{2.273}$$

で与えられる。ここで、$P_n$ は圧力テンソルの界面に平行な方向の成分 $P_t$ は垂直な方向の成分である。

今、$yz$ 平面上に界面があると仮定する。local color grarient $\boldsymbol{F}$ は、

$$\begin{aligned}
F_x &= F && (2.274)\\
F_y &= 0 && (2.275)\\
F_z &= 0 && (2.276)
\end{aligned}$$

となり、圧力テンソルは、

$$P_{xx} = p - \frac{2}{3}\tau A \mid \boldsymbol{F} \mid + 2\tau A \frac{F^2}{\mid \boldsymbol{F} \mid} \tag{2.277}$$

$$P_{yy} = p - \frac{2}{3}\tau A \mid \boldsymbol{F} \mid \tag{2.278}$$

$$P_{zz} = p - \frac{2}{3}\tau A \mid \boldsymbol{F} \mid \tag{2.279}$$

$$\tag{2.280}$$

となる。よって、表面張力は、

$$\sigma = \int (P_{xx} - P_{yy}) dx \tag{2.281}$$

$$= \int 2\tau A \mid \boldsymbol{F} \mid dx \tag{2.282}$$

となり、式（2.272）、（2.277）、（2.278）を用ると、表面張力として、

$$\sigma = \int 2\tau A 10 \partial_x (\rho_r - \rho_b) dx \tag{2.283}$$

が得られる。

密度差 $\rho_r - \rho_b$ の関数を、$\theta$ 関数 $\theta(x)$ を用いて

$$f(x) = \rho_r - \rho_b \tag{2.284}$$

$$= (\rho_r + \rho_b)\theta(x_0) - \rho_b \tag{2.285}$$

と仮定すると、

$$\partial_x f(x) = (\rho_r + \rho_b)\delta(x_0) \tag{2.286}$$

となり、最終的な表面張力として、

$$\sigma = \int 20\tau A(\rho_r + \rho_b)\delta(x_0) dx \tag{2.287}$$

$$= 20\tau A(\rho_r + \rho_b) \tag{2.288}$$

が得られる。

# 3. シミュレーションコードの並列化

## 3.1 メインプログラム

メインプログラムにおける処理の概略を以下に示す。

```
      call read_param
      call set_param
      call read_init
      call read_obsdat
      call set_init_data
      call read_restart
      call set_stream_param
      call set_boundary

      do 10 iot1 = 1,notmax

         do 20 iot2 = 1,ntmax

            call exchange0
            call stream
            call cal_rho_v
            call collision
            call exchange2
            call cal_omega
            call recoloring
            call set_boundary

 20      continue

         call cal_rho_v
         call open_out_file
         call output_fld1
         call gather
         call filtering
         call restart
         call close_out_file

 10   continue
```

まず、サブルーチン read_param で格子ボルツマン方程式中のパラメータの入力を行い、サブルーチン set_param で各種パラメータの設定、及び 3D15V 格子座標の設定を行う。サブルーチン read_init では、計算のステップ数、計算の種類の選択などの計算条件の入力を行う。計算の種類は現在、均質な系からの相分離、気泡の上昇、気泡流が選択可能となっており、密度、入口流速などを入力する。計算格子数は、common 文設定用のファイルにおいて parameter 文で指定している。サブルーチン read_obsdat では、計算領域内の障害物の座標、境界条件の入力を行う。固定壁境界条件の設定もここで行う。サブルーチン set_init_data では、計算に使用する変数の初期値設定を行う。リスタート計算の場合は、変数の初期値はサブルーチン read_restart により、リスタートファイルから読み込む。サブルーチン set_stream_param では、粒子の移動計算について各 PE が受け持つ計算領域の指定を行う。ここでは、計算領域を Fig.3.1に示すように 4 つの小領域に分割し並列処理を行っている。4 台以上を用いた並列計算も可能であるが、ここでは 4 台を使用する場合について示している。図左側に示すのが計算領域全体であり、右側の小領域を PE0 から PE3 までの 4 台のワークステーションが担当する。各 PE は小領域のデータだけを持ちその処理を行う。小領域には上下に一列ずつ境界データ用の配列を置き、適宜データ転送を行っている。例えば、PE1 の計算領域（白い部分）の底辺の部分のデータを PE0 の計算領域の上面のさらに上部に設けた境界用の配列（灰色の部分）に転送し、PE0 の計算で使用する。

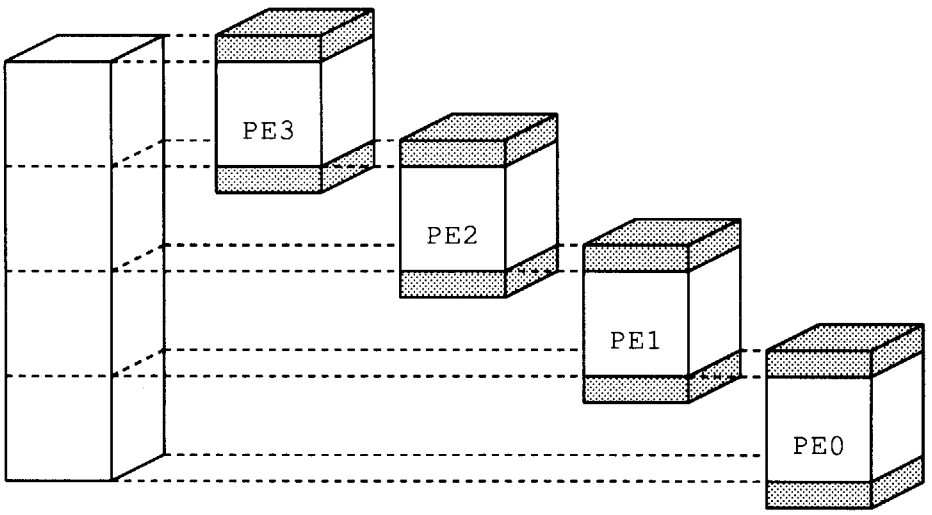



Fig. 3.1 Domain decomposition of the simulation region

サブルーチン set_boundary では、流入、流出、周期などの流れの境界条件の設定を行う。以上により、計算の初期状態の設定が終了する。パラメータや計算条件は、一つの入力用データファイルにまとめることも可能であるが、現在は、処理毎に分けて入力している。

do 10 のループで格子ボルツマン法による流れ場の時間発展の計算を行う。notmax は流れ場のデータを出力する回数である。内側の do 20 のループは計算の 1 タイムステップに相当する処理であり、ntmax はタイムステップ何回毎に流れ場のデータを出力する処理を行うかを示すものである。notmax 及び ntmax は、サブルーチン read_init において入力するものである。

do 20 のループでは、まず、サブルーチン exchange0 において計算に使用する変数配列の転送を行う。ただし、計算領域内部の値のみとし、上下の境界については転送処理は行わない。サブルーチン stream は分布関数の格子上の移動を PE 毎に計算する。サブルーチン cal_rho_v では、格子点での分布関数からその点での密度と速度を計算する。ここで求めた密度と速度は、以下の計算で平衡分布関数を求める際に使用する。サブルーチン collision では、格子ボルツマン方程式の衝突項、すなわち平衡分布への緩和過程の計算を行う。サブルーチン exchange2 では、格子点の密度について領域境界の値を転送する。この密度情報は、粒子配置の空間変化を計算する際に使用する。サブルーチン cal_omega では、格子点近傍の粒子配置の空間変化から、衝突項のうちの表面張力に関わる項を計算し、サブルーチン recoloring では、格子点での同色の分布関数の再配置を計算する。サブルーチン set_boundary では、流入、流出、壁などの境界条件の設定を行う。do 20 のループの内側が、計算の 1 タイムステップの処理に相当し、計算を ntmax ステップ進めてループを出ると、流れ場の処量を 1 回出力する。

流れ場の出力にあたっては、まず、サブルーチン cal_rho_v で格子点での密度と速度を計算する。次にサブルーチン open_out_file で、出力用ファイルのオープンを行い、サブルーチン output_fld1 で AVS でデータを読み込むためのフィールドファイルを出力する。サブルーチン gather では、各 PE が持つ格子点の密度と速度を PE0 に集めている。サブルーチン filtering では、計算のパラメータと密度と速度から圧力や体積率などの出力に必要な処量を計算し、出力する。サブルーチン restart では、計算に必要なデータをリスタートファイルに出力し、サブルーチン close_out_file でファイルのクローズを行い一つの流れ場を求める処理を終了する。リスタート用データは、プロセッサ毎に別々のファイル（ディスク）に書きだし、リスタート計算ではそれを読み込むようにしている。

以下では、格子ボルツマン法により流れ場を求めるための計算の主要なサブルーチンとして exchange0、stream、cal_rho_v、collision、exchange2、cal_omega、recoloring、set_boundary、gather、filtering の処理について記述する。

### 3.2 データ転送 ： サブルーチン exchange0

サブルーチン exchange0 では、以下に示すように各 PE で計算に使用する分布関数の配列のうち、計算領域内部の境界の値についての転送を行う。転送量を減らすため、境界上の格子点のデータ全てではなく、計算に関係する方向の値のみを転送する。すなわち、PE0 の上部の境界用配列には下向きの方向のデータだけを PE1 の計算領域の底面から転送する。これは、PE1 の底面のデータのうち下向きの速度を持つものしか PE0 の上部に関わってこないためである。なお、ここでは、計算領域上下の境界については転送処理は行わない。以下に処理の主要な部分を示す。

```
do 270 ij=1,npe-1
i=ij+1
```

```
      if(i.eq.npe) i=1

      if((iam.eq.i).or.(iam.eq.i-1)) then

      ideso=i-iam+i-1

      if(iam.eq.i) then
        jz=0
        do 250 jy=-1,lymax
        do 250 jx=-1,lxmax
        wsend(1,jx,jy) = red(6,jx,jy,jz)
        wsend(2,jx,jy) = red(8,jx,jy,jz)
        wsend(3,jx,jy) = red(9,jx,jy,jz)
        wsend(4,jx,jy) = red(12,jx,jy,jz)
        wsend(5,jx,jy) = red(13,jx,jy,jz)
        wsend(6,jx,jy) = blue(6,jx,jy,jz)
        wsend(7,jx,jy) = blue(8,jx,jy,jz)
        wsend(8,jx,jy) = blue(9,jx,jy,jz)
        wsend(9,jx,jy) = blue(12,jx,jy,jz)
        wsend(10,jx,jy)= blue(13,jx,jy,jz)
250     continue
        else
        jz=nlzmax-1
        do 260 jy=-1,lymax
        do 260 jx=-1,lxmax
        wsend(1,jx,jy) = red(5,jx,jy,jz)
        wsend(2,jx,jy) = red(7,jx,jy,jz)
        wsend(3,jx,jy) = red(10,jx,jy,jz)
        wsend(4,jx,jy) = red(11,jx,jy,jz)
        wsend(5,jx,jy) = red(14,jx,jy,jz)
        wsend(6,jx,jy) = blue(5,jx,jy,jz)
        wsend(7,jx,jy) = blue(7,jx,jy,jz)
        wsend(8,jx,jy) = blue(10,jx,jy,jz)
        wsend(9,jx,jy) = blue(11,jx,jy,jz)
        wsend(10,jx,jy) = blue(14,jx,jy,jz)
260     continue
      endif

        nsend=10*(lxmax+2)*(lymax+2)
```

```
      nrecv=nsend

     call mpi_sendrecv(wsend(1,-1,-1),nsend,
    c                  mpi_double_precision,ideso,1,
    c                  wrecv(1,-1,-1),nrecv,
    c                  mpi_double_precision,ideso,1,
    c                  mpi_comm_world,istatus,ierr)

     if(iam.eq.i) then
       jz=-1
       do 240 jy=-1,lymax
       do 240 jx=-1,lxmax
       red(5,jx,jy,jz)  = wrecv(1,jx,jy)
       red(7,jx,jy,jz)  = wrecv(2,jx,jy)
       red(10,jx,jy,jz) = wrecv(3,jx,jy)
       red(11,jx,jy,jz) = wrecv(4,jx,jy)
       red(14,jx,jy,jz) = wrecv(5,jx,jy)
       blue(5,jx,jy,jz)  = wrecv(6,jx,jy)
       blue(7,jx,jy,jz)  = wrecv(7,jx,jy)
       blue(10,jx,jy,jz) = wrecv(8,jx,jy)
       blue(11,jx,jy,jz) = wrecv(9,jx,jy)
       blue(14,jx,jy,jz) = wrecv(10,jx,jy)
240    continue
       else
       jz=nlzmax
       do 230 jy=-1,lymax
       do 230 jx=-1,lxmax
       red(6,jx,jy,jz)  = wrecv(1,jx,jy)
       red(8,jx,jy,jz)  = wrecv(2,jx,jy)
       red(9,jx,jy,jz)  = wrecv(3,jx,jy)
       red(12,jx,jy,jz) = wrecv(4,jx,jy)
       red(13,jx,jy,jz) = wrecv(5,jx,jy)
       blue(6,jx,jy,jz)  = wrecv(6,jx,jy)
       blue(8,jx,jy,jz)  = wrecv(7,jx,jy)
       blue(9,jx,jy,jz)  = wrecv(8,jx,jy)
       blue(12,jx,jy,jz) = wrecv(9,jx,jy)
       blue(13,jx,jy,jz) = wrecv(10,jx,jy)
230    continue
     endif
```

```
      endif

270   continue
```

ここで、npeは計算に使用するPEの数、あるいは計算機の台数である。iamはPEの番号であり、4台の場合、0、1、2、3となる。隣合うPEの場合にデータを転送するため、この処理は、0-1、1-2、2-3のあいだでのみ行われる。idesoは転送の相手先の番号となる。red(i,jx,jy,jz)、blue(i,jx,jy,jz)はそれぞれ赤粒子、青粒子の格子点(jx,jy,jz)におけるi方向の分布関数の値である。計算領域はjz方向に分割してあり、PEの受け持つ計算領域の範囲は、jx方向が0～lxmax-1、jy方向が0～lymax-1、jz方向が0～nlzmax-1であり、外側に境界を1格子おいている。wsend(i,jx,jy)、wrecv(i,jx,jy)はそれぞれ送信用、受信用の作業用配列である。プログラム中では3D15Vの格子の方向を5、7、10、11、14が上向き、6、8、9、12、13が下向きとしており、上のPEからは領域底面の下向きの方向のデータ、下のPEからは領域上面の上向きの方向のデータを転送することになる。転送では赤青粒子のデータを送信用配列にまとめ、mpi_sendrecvを用い送信と受信を同時に行い、受信したデータを境界の格子点のデータとしてに配列に格納している。

### 3.3 移動：サブルーチンstream

サブルーチンstreamでは、格子点の分布関数の隣の格子点への移動を計算する。

```
     do 11 id = 1,14
        do 20 iz = iz_start(id),iz_last(id),iz_step(id)
           do 30 iy = iy_start(id),iy_last(id),iy_step(id)
             do 40 ix = ix_start(id),ix_last(id),ix_step(id)
                 red(id,ix+ix_add(id),iy+iy_add(id),iz+iz_add(id))
    &                 = red(id,ix,iy,iz)
                 blue(id,ix+ix_add(id),iy+iy_add(id),iz+iz_add(id))
    &                 = blue(id,ix,iy,iz)
40             continue
30           continue
20         continue
11    continue
```

ここでは、3D15V格子の中心を除く14方向についてx、y、z方向にstartからlastまでstepずつデータを移動させ上書きしていく。上書きのため、移動は格子の方向のもっとも下流

側から行う。また、PE内での移動の計算だけなので転送は行わない。start、last、stepの値は、PE毎にサブルーチンset_stream_paramにおいて設定したものである。

### 3.4 密度と速度：サブルーチンcal_rho_v

ここでは格子点の分布関数の値からマクロな流れの変数である密度と速度を計算する。計算の主要な部分を以下に示す。

```
      do 40 iz = iam*nlzmax,(iam+1)*nlzmax-1
            izz=iz-iam*nlzmax
         do 30 iy = -1,lymax
            do 20 ix = -1,lxmax

               rho_red(ix,iy,iz) = 0.d0
               rho_blue(ix,iy,iz) = 0.d0

               do 10 id = 14, 0, -1
                  rho_red(ix,iy,iz)  = red(id,ix,iy,izz)
     &                 + rho_red(ix,iy,iz)
                  rho_blue(ix,iy,iz) = blue(id,ix,iy,izz)
     &                 + rho_blue(ix,iy,iz)
 10            continue

               rho = rho_red(ix,iy,iz) + rho_blue(ix,iy,iz)

               vx(ix,iy,iz) = 0.d0
               vy(ix,iy,iz) = 0.d0
               vz(ix,iy,iz) = 0.d0

               if (rho .gt. 0.d0) then

                  djx_red = 0.d0
                  djy_red = 0.d0
                  djz_red = 0.d0
                  djx_blue = 0.d0
                  djy_blue = 0.d0
                  djz_blue = 0.d0

                  do 50 id = 14, 0, -1
```

```
                djx_red = dex(id)*red(id,ix,iy,izz) + djx_red
                djy_red = dey(id)*red(id,ix,iy,izz) + djy_red
                djz_red = dez(id)*red(id,ix,iy,izz) + djz_red

                djx_blue = dex(id)*blue(id,ix,iy,izz)
     &                 + djx_blue
                djy_blue = dey(id)*blue(id,ix,iy,izz)
     &                 + djy_blue
                djz_blue = dez(id)*blue(id,ix,iy,izz)
     &                 + djz_blue

 50           continue

              vx(ix,iy,iz) = (djx_red + djx_blue)/rho
              vy(ix,iy,iz) = (djy_red + djy_blue)/rho
              vz(ix,iy,iz) = (djz_red + djz_blue)/rho
```

マクロな変数である密度、rho_red(ix,iy,iz) と rho_blue(ix,iy,iz)、及び速度、vx(ix,iy,iz)、vy(ix,iy,iz)、vz(ix,iy,iz)、については、PE 毎のデータ分割は行わず、処理の分割だけを行っている。このため、z 方向のループの開始と終了を示すインデックスは、PE の番号に応じて変化している。

### 3.5 衝突項：サブルーチン collision

サブルーチン collision では、以下に示すようにまず平衡分布関数と緩和時間を計算し、衝突項の計算を行う。一つの格子点上での計算であるため、データ転送は行わない。

```
      do 10 iz = 0,nlzmax-1
            iw = iz+iam*nlzmax
         do 20 iy = 0,lymax-1
            do 30 ix = 0,lxmax-1

                  r_r = rho_red(ix,iy,iw)
                  r_b = rho_blue(ix,iy,iw)

                  vvx = vx(ix,iy,iw)
                  vvy = vy(ix,iy,iw)
                  vvz = vz(ix,iy,iw)
```

```
                do 60 id = 0,14

                   call cal_feq
     &                  (
     &                  vvx,vvy,vvz,
     &                  r_r,r_b,
     &                  id,
     &                  feq_red,feq_blue
     &                  )

                   tau_red  = tau_red_org
                   tau_blue = tau_blue_org

                   psi = (rho_red(ix,iy,iw) - rho_blue(ix,iy,iw))/
     &                    (rho_red(ix,iy,iw) + rho_blue(ix,iy,iw))

                   if (psi .gt. delta) then
                      tu = tau_red_org
                   elseif ((psi .gt. 0.0).and.(psi .le. delta)) then
                      tu = alpha + beta*psi + dkappa*psi*psi
                   elseif ((psi .ge. -1.0*delta).and.(psi .le. 0.d0)) then
                      tu = alpha + eta*psi + xi*psi*psi
                   elseif (psi .lt. -1.0*delta) then
                      tu = tau_blue_org
                   endif

                   tau_red  = tu
                   tau_blue = tu

                   red(id,ix,iy,iz) = red(id,ix,iy,iz) -
     &                  (red(id,ix,iy,iz) - feq_red)/tau_red

                   blue(id,ix,iy,iz) = blue(id,ix,iy,iz) -
     &                  (blue(id,ix,iy,iz) - feq_blue)/tau_blue

60              continue
30           continue
20        continue
10     continue
```

do 60 のループで呼ぶサブルーチン cal_feq は格子点の密度と速度から平衡分布関数を求めるものである。緩和過程のパラメータである緩和時間 (tu) は、それぞれの相の値の間の内挿により求めている。

### 3.6 転送 ： サブルーチン exchange2

サブルーチン exchange2 では、各 PE が受け持つ領域の境界の密度データを隣の PE に転送する。上下方向が周期境界の場合、一番上の PE の上部境界のデータを一番下の PE の下部へ転送する。流出境界や流入境界の場合は転送を行う必要はない。転送用の作業配列として wsend(i,jx,jy)、 wrecv(i,jx,jy) を使用している。転送終了後、境界データの矩形領域の 4 つの辺にあたるデータ、及び 4 つのコーナーにあたるデータを、周期境界条件として処理している。

```
      if(iam.eq.0.or.iam.eq.npe-1) then

      ideso=npe-1-iam

       if(iam.eq.0) then
        jz= 0
        else
        jz= lzmax-1
      endif
        do 350 jy=-1,lymax
        do 350 jx=-1,lxmax
        wsend(1,jx,jy) = rho_red(jx,jy,jz)
        wsend(2,jx,jy) = rho_blue(jx,jy,jz)
350     continue

        nsend=2*(lxmax+2)*(lymax+2)
        nrecv=nsend

       call mpi_sendrecv(wsend(1,-1,-1),nsend,
     c                   mpi_double_precision,ideso,1,
     c                   wrecv(1,-1,-1),nrecv,
     c                   mpi_double_precision,ideso,1,
     c                   mpi_comm_world,istatus,ierr)

       if(iam.eq.0) then
        jz=-1
        else
```

```
      jz= lzmax
    endif

      do 340 jy=-1,lymax
      do 340 jx=-1,lxmax
      rho_red(jx,jy,jz)  = wrecv(1,jx,jy)
      rho_blue(jx,jy,jz) = wrecv(2,jx,jy)
340   continue

       rho_red(lxmax,lymax,jz) = rho_red(0,0,jz)
       rho_red(lxmax,-1,jz)    = rho_red(0,lymax-1,jz)
       rho_red(-1,lymax,jz)    = rho_red(lxmax-1,0,jz)
       rho_red(-1,-1,jz)       = rho_red(lxmax-1,lymax-1,jz)

       rho_blue(lxmax,lymax,jz) = rho_blue(0,0,jz)
       rho_blue(lxmax,-1,jz)    = rho_blue(0,lymax-1,jz)
       rho_blue(-1,lymax,jz)    = rho_blue(lxmax-1,0,jz)
       rho_blue(-1,-1,jz)       = rho_blue(lxmax-1,lymax-1,jz)

       do 100 ix = 0,lxmax-1

          rho_red(ix,lymax,jz) = rho_red(ix,0,jz)
          rho_red(ix,-1,jz)    = rho_red(ix,lymax-1,jz)

          rho_blue(ix,lymax,jz) = rho_blue(ix,0,jz)
          rho_blue(ix,-1,jz)    = rho_blue(ix,lymax-1,jz)

100    continue

       do 160 iy = 0,lymax-1

          rho_red(lxmax,iy,jz) = rho_red(0,iy,jz)
          rho_red(-1,iy,jz)    = rho_red(lxmax-1,iy,jz)

          rho_blue(lxmax,iy,jz) = rho_blue(0,iy,jz)
          rho_blue(-1,iy,jz)    = rho_blue(lxmax-1,iy,jz)

160    continue
```

```
      endif

    do 270 ij=1,npe-1
     i=ij+1
     if(i.eq.npe) i=1

    if((iam.eq.i).or.(iam.eq.i-1)) then

    ideso=i-iam+i-1

    if(iam.eq.i) then
      jz=nlzmax*iam
      do 250 jy=-1,lymax
      do 250 jx=-1,lxmax
      wsend(1,jx,jy) = rho_red(jx,jy,jz)
      wsend(2,jx,jy) = rho_blue(jx,jy,jz)
250   continue
      else
      jz=nlzmax*(iam+1)-1
      do 260 jy=-1,lymax
      do 260 jx=-1,lxmax
      wsend(1,jx,jy) = rho_red(jx,jy,jz)
      wsend(2,jx,jy) = rho_blue(jx,jy,jz)
260   continue
     endif

      nsend=2*(lxmax+2)*(lymax+2)
      nrecv=nsend

     call mpi_sendrecv(wsend(1,-1,-1),nsend,
    c                  mpi_double_precision,ideso,1,
    c                  wrecv(1,-1,-1),nrecv,
    c                  mpi_double_precision,ideso,1,
    c                  mpi_comm_world,istatus,ierr)

     if(iam.eq.i) then
       jz=nlzmax*iam-1
       do 240 jy=-1,lymax
       do 240 jx=-1,lxmax
```

```
      rho_red(jx,jy,jz)  = wrecv(1,jx,jy)
      rho_blue(jx,jy,jz) = wrecv(2,jx,jy)
240   continue
      else
      jz=nlzmax*(iam+1)
      do 230 jy=-1,lymax
      do 230 jx=-1,lxmax
      rho_red(jx,jy,jz)  = wrecv(1,jx,jy)
      rho_blue(jx,jy,jz) = wrecv(2,jx,jy)
230   continue
    endif

    endif

270   continue

      else

       rho_red(lxmax,lymax,-1) = rho_red(0,0,lzmax-1)
       rho_red(lxmax,-1,-1)    = rho_red(0,lymax-1,lzmax-1)
       rho_red(-1,lymax,-1)    = rho_red(lxmax-1,0,lzmax-1)
       rho_red(-1,-1,-1)       = rho_red(lxmax-1,lymax-1,lzmax-1)
       rho_red(lxmax,lymax,nlzmax) = rho_red(0,0,nlzmax)
       rho_red(lxmax,-1,nlzmax)    = rho_red(0,lymax-1,nlzmax)
       rho_red(-1,lymax,nlzmax)    = rho_red(lxmax-1,0,nlzmax)
       rho_red(-1,-1,nlzmax)       = rho_red(lxmax-1,lymax-1,nlzmax)

       rho_blue(lxmax,lymax,-1) = rho_blue(0,0,lzmax-1)
       rho_blue(lxmax,-1,-1)    = rho_blue(0,lymax-1,lzmax-1)
       rho_blue(-1,lymax,-1)    = rho_blue(lxmax-1,0,lzmax-1)
       rho_blue(-1,-1,-1)       = rho_blue(lxmax-1,lymax-1,lzmax-1)
       rho_blue(lxmax,lymax,nlzmax) = rho_blue(0,0,nlzmax)
       rho_blue(lxmax,-1,nlzmax)    = rho_blue(0,lymax-1,nlzmax)
       rho_blue(-1,lymax,nlzmax)    = rho_blue(lxmax-1,0,nlzmax)
       rho_blue(-1,-1,nlzmax)       = rho_blue(lxmax-1,lymax-1,nlzmax)

       do 800 ix = 0,lxmax-1

          rho_red(ix,lymax,-1) = rho_red(ix,0,lzmax-1)
```

```
          rho_red(ix,-1,-1)     = rho_red(ix,lymax-1,lzmax-1)
          rho_red(ix,lymax,nlzmax) = rho_red(ix,0,nlzmax)
          rho_red(ix,-1,nlzmax)     = rho_red(ix,lymax-1,nlzmax)

          rho_blue(ix,lymax,-1) = rho_blue(ix,0,lzmax-1)
          rho_blue(ix,-1,-1)     = rho_blue(ix,lymax-1,lzmax-1)
          rho_blue(ix,lymax,nlzmax) = rho_blue(ix,0,nlzmax)
          rho_blue(ix,-1,nlzmax)     = rho_blue(ix,lymax-1,nlzmax)

800       continue

          do 880 iy = 0,lymax-1
        do 880 ix = 0,lxmax-1

             rho_red(ix,iy,-1) = rho_red(ix,iy,lzmax-1)
             rho_red(ix,iy,lzmax) = rho_red(ix,iy,0)

             rho_blue(ix,iy,-1) = rho_blue(ix,iy,lzmax-1)
             rho_blue(ix,iy,lzmax) = rho_blue(ix,iy,0)

880          continue

        do 860 iy = 0,lymax-1

          rho_red(lxmax,iy,-1) = rho_red(0,iy,lzmax-1)
          rho_red(-1,iy,-1)     = rho_red(lxmax-1,iy,lzmax-1)
          rho_red(lxmax,iy,lzmax) = rho_red(0,iy,0)
          rho_red(-1,iy,lzmax)     = rho_red(lxmax-1,iy,0)

          rho_blue(lxmax,iy,-1) = rho_blue(0,iy,lzmax-1)
          rho_blue(-1,iy,-1)     = rho_blue(lxmax-1,iy,lzmax-1)
          rho_blue(lxmax,iy,lzmax) = rho_blue(0,iy,0)
          rho_blue(-1,iy,lzmax)     = rho_blue(lxmax-1,iy,0)

860       continue

 80       continue

        endif
```

```
      do 120 iz = 0,nlzmax-1

         rho_red(lxmax,lymax,iz) = rho_red(0,0,iz)
         rho_red(lxmax,-1,iz)    = rho_red(0,lymax-1,iz)
         rho_red(-1,lymax,iz)    = rho_red(lxmax-1,0,iz)
         rho_red(-1,-1,iz)       = rho_red(lxmax-1,lymax-1,iz)

         rho_blue(lxmax,lymax,iz) = rho_blue(0,0,iz)
         rho_blue(lxmax,-1,iz)    = rho_blue(0,lymax-1,iz)
         rho_blue(-1,lymax,iz)    = rho_blue(lxmax-1,0,iz)
         rho_blue(-1,-1,iz)       = rho_blue(lxmax-1,lymax-1,iz)

         do 130 ix = 0,lxmax-1
            rho_red(ix,lymax,iz) = rho_red(ix,0,iz)
            rho_red(ix,-1,iz)    = rho_red(ix,lymax-1,iz)

            rho_blue(ix,lymax,iz) = rho_blue(ix,0,iz)
            rho_blue(ix,-1,iz)    = rho_blue(ix,lymax-1,iz)

130         continue

      do 140 iy = 0,lymax-1

            rho_red(lxmax,iy,iz) = rho_red(0,iy,iz)
            rho_red(-1,iy,iz)    = rho_red(lxmax-1,iy,iz)

            rho_blue(lxmax,iy,iz) = rho_blue(0,iy,iz)
            rho_blue(-1,iy,iz)    = rho_blue(lxmax-1,iy,iz)

140   continue

120   continue
```

### 3.7 表面張力 ： サブルーチン cal_omega

サブルーチン cal_omega では、二相の界面でのみ作用する表面張力の項を計算する。

```
      call cal_f

      do 10 iz = 0,nlzmax-1
            iw=iz+iam*nlzmax
         do 20 iy = 0,lymax-1
            do 30 ix = 0,lxmax-1

                  F2 = vx(ix,iy,iw)*vx(ix,iy,iw)
     &                 + vy(ix,iy,iw)*vy(ix,iy,iw)
     &                 + vz(ix,iy,iw)*vz(ix,iy,iw)
                  F = dsqrt(F2)

                     fi(0,ix,iy,iz) = A*F*(1.d0/3.d0 - H)
     &                    + fi(0,ix,iy,iz)

                     do 60 id = 1,14

                        ef = vx(ix,iy,iw)*dex(id)
     &                       + vy(ix,iy,iw)*dey(id)
     &                       + vz(ix,iy,iw)*dez(id)

                        cs2 = dex(id)*dex(id)
     &                       + dey(id)*dey(id)
     &                       + dez(id)*dez(id)

                        fi(id,ix,iy,iz) =
     &                       A*F*(ef*ef/(F2*cs2) - H)
     &                       + fi(id,ix,iy,iz)

60                   continue
30             continue
20          continue
10       continue
```

サブルーチン cal_f は格子点での x,y,z 各方向における二相の密度差、すなわち色の勾配を計算し配列 vx(ix,iy,iw)、vy(ix,iy,iw)、vz(ix,iy,iw) に格納するものである。したがってここでは vx、vy、vz は速度成分を表す変数ではない。F は色の勾配の絶対値、dex(id)、dey(id)、dez(id) はそれぞれ、格子方向 id の単位ベクトルの x、y、z 方向の大きさであり、cs2 は大き

さの絶対値の二乗である。色の勾配を用いて表面張力の効果を計算し、分布関数に作用させる。fi(id,ix,iy,iz) は赤粒子と青粒子の分布関数の和である。

### 3.8 粒子の再配置 ： サブルーチン recoloring

サブルーチン recoloring では、二相界面近傍の格子点での粒子の再配置を行う。再配置は運動量の方向が粒子の色の勾配の方向と一致するように行う。この時、格子点におけるそれぞれの色の粒子密度及び全運動量は保存される。

```
      do 10 iz = 0,nlzmax-1
            iw = iz+iam*nlzmax
         do 20 iy = 0,lymax-1
            do 30 ix = 0,lxmax-1

                     rho_r = 0.d0

                     do 40 id = 0,14

                        cs2 = dex(id)*dex(id)
     &                       +dey(id)*dey(id)
     &                       +dez(id)*dez(id)
                        cs = dsqrt(cs2)

                        if (id .eq. 0) cs = 1.d0

                        edf(id) = (vx(ix,iy,iw)*dex(id)
     &                       + vy(ix,iy,iw)*dey(id)
     &                       + vz(ix,iy,iw)*dez(id))/cs

                        ib(id) = id

 40                  continue

                     do 200 iroop = 13,0,-1
                        do 210 id = 0,iroop

                           if (edf(ib(id+1)) - edf(ib(id))
     &                          .gt. cri) then
                              itmp = ib(id+1)
```

```
                        ib(id+1) = ib(id)
                        ib(id) = itmp
                      end if
210                 continue
200               continue

                  do 150 id = 0,14
                     red(id,ix,iy,iz) = 0.d0
                     blue(id,ix,iy,iz) = 0.d0
150               continue

                  rho_r = 0.d0
                  rho_check = 0.d0

                  do 70 id = 0,14

                     rho_check = rho_r + fi(ib(id),ix,iy,iz)

                     if ( rho_red(ix,iy,iw) - rho_check
     &                    .ge. 0.d0) then
                        red(ib(id),ix,iy,iz)
     &                       = fi(ib(id),ix,iy,iz)
                        rho_r = red(ib(id),ix,iy,iz) + rho_r
                     else
                        if (rho_red(ix,iy,iw) - rho_r
     &                       .ge. 0.d0) then
                           red(ib(id),ix,iy,iz)
     &                          = rho_red(ix,iy,iw) - rho_r
                           rho_r = rho_r + red(ib(id),ix,iy,iz)
                        else if (dabs(rho_r - rho_red(ix,iy,iw))
     &                          .lt. cri) then
                           red(ib(id),ix,iy,iz)
     &                          = 0.d0
                        else
                           red(ib(id),ix,iy,iz)
     &                          = 0.d0
                        endif
                     end if
```

```
                    blue(ib(id),ix,iy,iz) =
     &                   fi(ib(id),ix,iy,iz)
     &                   - red(ib(id),ix,iy,iz)

70                  continue

              endif
30          continue
20        continue
10    continue
```

ここで、edf(id) は id 方向の色の勾配の大きさであり、do 200 のループで、格子の方向を色の勾配の大きい方から順に並べている。do 70 のループでは、色の勾配の大きい方向から順に、全分布関数を越えない範囲で赤粒子の分布関数の再配置を行い、残りを青粒子の分布関数の値としている。

### 3.9 境界条件 ： サブルーチン set_boundary

ここでは、PE 毎に計算領域側面の境界条件を周期境界として変数の設定を行う。全計算領域上下の境界条件が、周期境界条件である場合は、側面の処理の前に上下の境界のデータを転送し合う必要がある。上下面の境界条件には、流出、流入なども設定可能であるが、以下には周期境界条件の場合のプログラムを示す。サブルーチン exchange0、exchange2 と同様、転送時には作業用配列 wsend、wrecv を使用している。

```
      if(npe.ne.1) then

        if(iam.eq.0.or.iam.eq.npe-1) then

        ideso=npe-1-iam

        if(iam.eq.0) then
          jz=0
          else
          jz=nlzmax-1
        endif
```

```
      do 350 jy=-1,lymax
      do 350 jx=-1,lxmax
      do 330 id=1,15
      wsend(id,jx,jy) = red(id-1,jx,jy,jz)
330   continue
      do 340 id=16,30
      wsend(id,jx,jy) = blue(id-16,jx,jy,jz)
340   continue
350   continue

      nsend=30*(lxmax+2)*(lymax+2)
      nrecv=nsend

      call mpi_sendrecv(wsend(1,-1,-1),nsend,
     c                  mpi_double_precision,ideso,1,
     c                  wrecv(1,-1,-1),nrecv,
     c                  mpi_double_precision,ideso,1,
     c                  mpi_comm_world,istatus,ierr)

      if(iam.eq.0) then
        jz=-1
        else
        jz=nlzmax
      endif

      do 450 jy=-1,lymax
      do 450 jx=-1,lxmax
      do 440 id=1,15
      red(id-1,jx,jy,jz)  = wrecv(id,jx,jy)
440   continue
      do 430 id=16,30
      blue(id-16,jx,jy,jz)  = wrecv(id,jx,jy)
430   continue
450   continue


      do 10 id = 0,14

       red(id,lxmax,lymax,jz) = red(id,0,0,jz)
```

```
        red(id,lxmax,-1,jz) = red(id,0,lymax-1,jz)
        red(id,-1,lymax,jz) = red(id,lxmax-1,0,jz)
        red(id,-1,-1,jz) = red(id,lxmax-1,lymax-1,jz)

        blue(id,lxmax,lymax,jz) = blue(id,0,0,jz)
        blue(id,lxmax,-1,jz) = blue(id,0,lymax-1,jz)
        blue(id,-1,lymax,jz) = blue(id,lxmax-1,0,jz)
        blue(id,-1,-1,jz) = blue(id,lxmax-1,lymax-1,jz)

10   continue

       do 100 ix = 0,lxmax-1
       do 100 id = 0,14

           red(id,ix,lymax,jz) = red(id,ix,0,jz)
           red(id,ix,-1,jz) = red(id,ix,lymax-1,jz)

           blue(id,ix,lymax,jz) = blue(id,ix,0,jz)
           blue(id,ix,-1,jz) = blue(id,ix,lymax-1,jz)

100      continue

       do 160 iy = 0,lymax-1
       do 160 id = 0,14

           red(id,lxmax,iy,jz) = red(id,0,iy,jz)
           red(id,-1,iy,jz) = red(id,lxmax-1,iy,jz)

           blue(id,lxmax,iy,jz) = blue(id,0,iy,jz)
           blue(id,-1,iy,jz) = blue(id,lxmax-1,iy,jz)

160      continue

       endif

     else

     do 80 id = 0,14
```

```
      red(id,lxmax,lymax,-1) = red(id,0,0,lzmax-1)
      red(id,lxmax,-1,-1) = red(id,0,lymax-1,lzmax-1)
      red(id,-1,lymax,-1) = red(id,lxmax-1,0,lzmax-1)
      red(id,-1,-1,-1) = red(id,lxmax-1,lymax-1,lzmax-1)
      red(id,lxmax,lymax,lzmax) = red(id,0,0,0)
      red(id,lxmax,-1,lzmax) = red(id,0,lymax-1,0)
      red(id,-1,lymax,lzmax) = red(id,lxmax-1,0,0)
      red(id,-1,-1,lzmax) = red(id,lxmax-1,lymax-1,0)

      blue(id,lxmax,lymax,-1) = blue(id,0,0,lzmax-1)
      blue(id,lxmax,-1,-1) = blue(id,0,lymax-1,lzmax-1)
      blue(id,-1,lymax,-1) = blue(id,lxmax-1,0,lzmax-1)
      blue(id,-1,-1,-1) = blue(id,lxmax-1,lymax-1,lzmax-1)
      blue(id,lxmax,lymax,lzmax) = blue(id,0,0,0)
      blue(id,lxmax,-1,lzmax) = blue(id,0,lymax-1,0)
      blue(id,-1,lymax,lzmax) = blue(id,lxmax-1,0,0)
      blue(id,-1,-1,lzmax) = blue(id,lxmax-1,lymax-1,0)

80    continue

      do 800 ix = 0,lxmax-1
      do 800 id = 0,14

         red(id,ix,lymax,-1) = red(id,ix,0,lzmax-1)
         red(id,ix,-1,-1) = red(id,ix,lymax-1,lzmax-1)
         red(id,ix,lymax,lzmax) = red(id,ix,0,0)
         red(id,ix,-1,lzmax) = red(id,ix,lymax-1,0)

         blue(id,ix,lymax,-1) = blue(id,ix,0,lzmax-1)
         blue(id,ix,-1,-1) = blue(id,ix,lymax-1,lzmax-1)
         blue(id,ix,lymax,lzmax) = blue(id,ix,0,0)
         blue(id,ix,-1,lzmax) = blue(id,ix,lymax-1,0)
800   continue

         do 880 iy = 0,lymax-1
         do 880 ix = 0,lxmax-1
         do 880 id = 0,14

            red(id,ix,iy,-1) = red(id,ix,iy,lzmax-1)
```

```
            red(id,ix,iy,lzmax) = red(id,ix,iy,0)

            blue(id,ix,iy,-1) = blue(id,ix,iy,lzmax-1)
            blue(id,ix,iy,lzmax) = blue(id,ix,iy,0)

880         continue

          do 860 iy = 0,lymax-1
          do 860 id = 0,14

            red(id,lxmax,iy,-1) = red(id,0,iy,lzmax-1)
            red(id,-1,iy,-1) = red(id,lxmax-1,iy,lzmax-1)
            red(id,lxmax,iy,lzmax) = red(id,0,iy,0)
            red(id,-1,iy,lzmax) = red(id,lxmax-1,iy,0)

            blue(id,lxmax,iy,-1) = blue(id,0,iy,lzmax-1)
            blue(id,-1,iy,-1) = blue(id,lxmax-1,iy,lzmax-1)
            blue(id,lxmax,iy,lzmax) = blue(id,0,iy,0)
            blue(id,-1,iy,lzmax) = blue(id,lxmax-1,iy,0)

860       continue

        endif

          do 120 iz = 0,nlzmax-1
          do 30 id=0,14

            red(id,lxmax,lymax,iz) = red(id,0,0,iz)
            red(id,lxmax,-1,iz) = red(id,0,lymax-1,iz)
            red(id,-1,lymax,iz) = red(id,lxmax-1,0,iz)
            red(id,-1,-1,iz) = red(id,lxmax-1,lymax-1,iz)

            blue(id,lxmax,lymax,iz) = blue(id,0,0,iz)
            blue(id,lxmax,-1,iz) = blue(id,0,lymax-1,iz)
            blue(id,-1,lymax,iz) = blue(id,lxmax-1,0,iz)
            blue(id,-1,-1,iz) = blue(id,lxmax-1,lymax-1,iz)
 30       continue
120       continue
```

```
      do 130 iz = 0,nlzmax-1
      do 130 ix = 0,lxmax-1
      do 130 id=0,14
            red(id,ix,lymax,iz) = red(id,ix,0,iz)
            red(id,ix,-1,iz) = red(id,ix,lymax-1,iz)

            blue(id,ix,lymax,iz) = blue(id,ix,0,iz)
            blue(id,ix,-1,iz) = blue(id,ix,lymax-1,iz)

130         continue

      do 140 iz = 0,nlzmax-1
      do 140 iy = 0,lymax-1
      do 140 id=0,14

            red(id,lxmax,iy,iz) = red(id,0,iy,iz)
            red(id,-1,iy,iz) = red(id,lxmax-1,iy,iz)

            blue(id,lxmax,iy,iz) = blue(id,0,iy,iz)
            blue(id,-1,iy,iz) = blue(id,lxmax-1,iy,iz)

140     continue
```

### 3.10 出力データ集約 ： サブルーチン gather

ここでは、各 PE で計算した密度及び速度データをルートプロセッサに集約し、出力の準備をする。

```
      call mpi_gather(rho_red(-1,-1,nlzmax*iam),
     c                (lxmax+2)*(lymax+2)*nlzmax,
     c                mpi_double_precision,
     c                rho_red(-1,-1,nlzmax*iam),
     c                (lxmax+2)*(lymax+2)*nlzmax,
     c                mpi_double_precision,
     c                0,mpi_comm_world,ierr)
      call mpi_gather(rho_blue(-1,-1,nlzmax*iam),
     c                (lxmax+2)*(lymax+2)*nlzmax,
```

```
c                    mpi_double_precision,
c                    rho_blue(-1,-1,nlzmax*iam),
c                    (lxmax+2)*(lymax+2)*nlzmax,
c                    mpi_double_precision,
c                    0,mpi_comm_world,ierr)
      call mpi_gather(vx(-1,-1,nlzmax*iam),
c                    (lxmax+2)*(lymax+2)*nlzmax,
c                    mpi_double_precision,
c                    vx(-1,-1,nlzmax*iam),
c                    (lxmax+2)*(lymax+2)*nlzmax,
c                    mpi_double_precision,
c                    0,mpi_comm_world,ierr)
      call mpi_gather(vy(-1,-1,nlzmax*iam),
c                    (lxmax+2)*(lymax+2)*nlzmax,
c                    mpi_double_precision,
c                    vy(-1,-1,nlzmax*iam),
c                    (lxmax+2)*(lymax+2)*nlzmax,
c                    mpi_double_precision,
c                    0,mpi_comm_world,ierr)
      call mpi_gather(vz(-1,-1,nlzmax*iam),
c                    (lxmax+2)*(lymax+2)*nlzmax,
c                    mpi_double_precision,
c                    vz(-1,-1,nlzmax*iam),
c                    (lxmax+2)*(lymax+2)*nlzmax,
c                    mpi_double_precision,
c                    0,mpi_comm_world,ierr)
```

rho_red、rho_blue は、それぞれ赤粒子、青粒子の密度であり、vx、vy、vz はそれぞれ x、y、z 方向の速度である。これらの出力用変数の配列は PE 毎に分割していないため、ルートプロセッサに集約し出力している。これは、大きな気泡や液滴の計算において、全計算領域のデータから表面張力などを求める処理を行っているためである。通常の流れ場を計算するだけであれば、計算中にデータの集約をする必要はなく、各 PE 毎に出力する方が転送時間は節約できる。

### 3.11 データ出力 ： サブルーチン filtering

ここでは、サブルーチン gather で集約したデータを出力する。出力にあたっては、指定した格子点数だけの空間平均を行っている。このサブルーチンでは密度と速度から、適宜、圧力、界面積濃度、体積率などを計算し出力することが可能である。以下では、速度と密度についてのみの処理を示す。

```
      do 10 iiz= 0, lzmax/num_zave -1
        do 20 iiy= 0, lymax/num_yave -1
           do 30 iix= 0, lxmax/num_xave -1

              ncount = 0

              rho_red_sum = 0.d0
              rho_blue_sum = 0.d0

              vx_sum = 0.d0
              vy_sum = 0.d0
              vz_sum = 0.d0

              do 40 ix = num_xave*iix, num_xave*(iix+1)-1
                 do 50 iy= num_yave*iiy, num_yave*(iiy+1)-1
                    do 60 iz= num_zave*iiz, num_zave*(iiz+1)-1

                       rho_red_sum
     &                      = rho_red(ix,iy,iz) + rho_red_sum
                       rho_blue_sum
     &                      = rho_blue(ix,iy,iz) + rho_blue_sum

                       vx_sum
     &                      = vx(ix,iy,iz) + vx_sum
                       vy_sum
     &                      = vy(ix,iy,iz) + vy_sum
                       vz_sum
     &                      = vz(ix,iy,iz) + vz_sum

                       ncount = ncount + 1

 60                 continue
 50              continue
 40           continue

              rho_red_ave = rho_red_sum/ncount
              rho_blue_ave = rho_blue_sum/ncount

              vx_ave = vx_sum/ncount
```

```
            vy_ave = vy_sum/ncount
            vz_ave = vz_sum/ncount

            atwood = (rho_red_ave - rho_blue_ave)/
     c                (rho_red_ave + rho_blue_ave)

          write(ioutfile,1008)
     c      vx_ave,vy_ave,vz_ave,atwood
1008 format(4(f7.4,1x))

30         continue
20       continue
10     continue
```

ここで、lxmax、lymax、lzmax はそれぞれ x、y、z 方向の格子数であり、num_xave、num_yave、num_zave は空間平均を行う格子数である。赤青の密度 rho_red(ix,iy,iz)、rho_blue(ix,iy,iz) と x、y、z 方向の速度 vx(ix,iy,iz)、vy(ix,iy,iz)、vz(ix,iy,iz) を空間平均し、rho_red_ave、rho_blue_ave、vx_ave、vy_ave、vz_ave とし、平均密度から Atwood 数を算出し、それらを出力している。

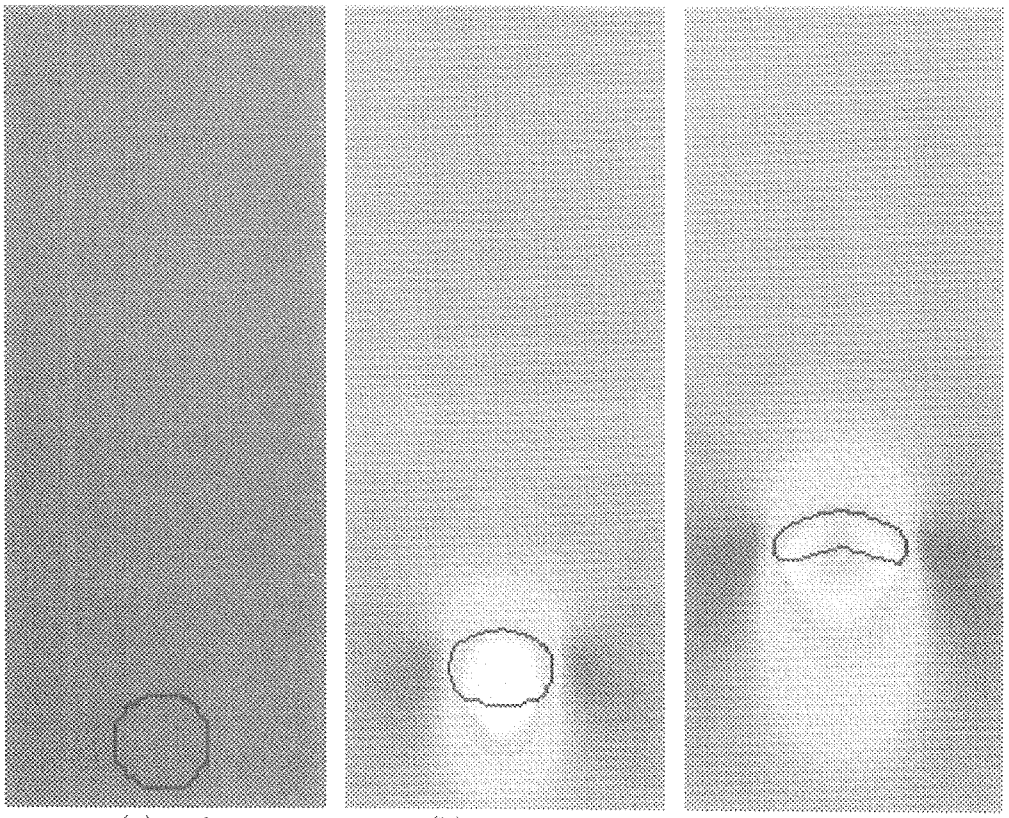
# 4. 並列計算

## 4.1 上昇気泡のシミュレーション

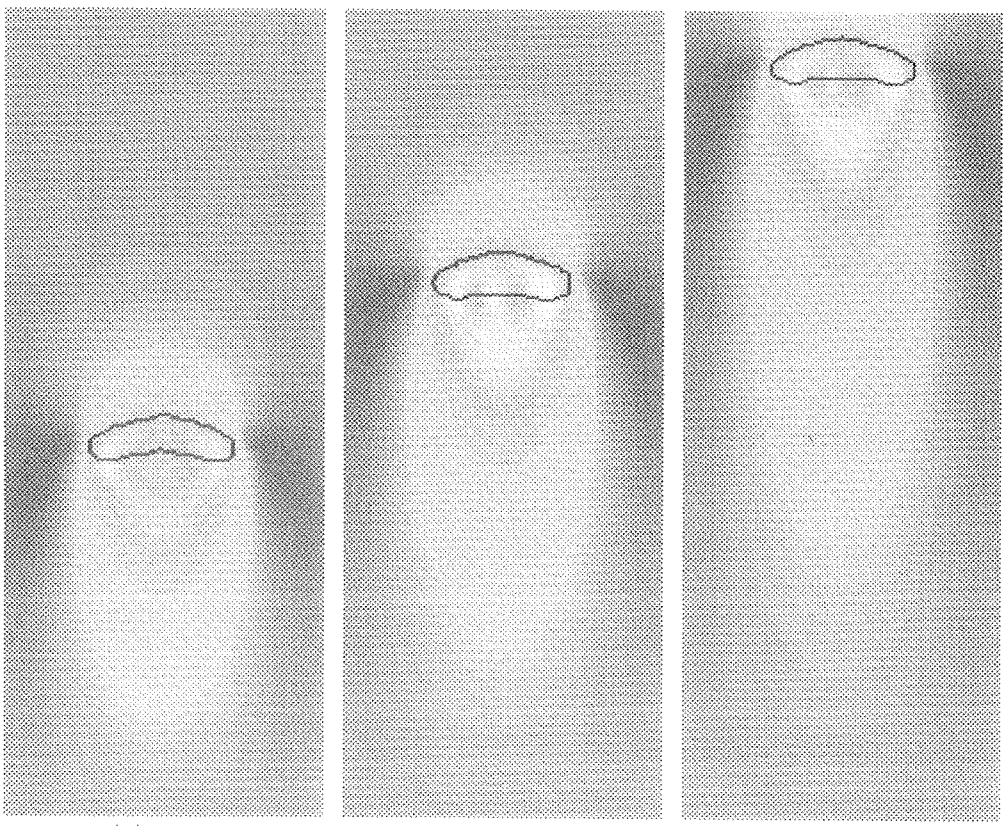
並列計算の実行例として外力の影響のもとに上昇する単一気泡及び二つの気泡のシミュレーションを示す。計算領域は単一気泡の場合は $x \times y \times z = 80 \times 80 \times 200$ であり、二つの気泡の場合、$x \times y \times z = 80 \times 80 \times 480$ である。初期条件として静止した青相を領域内に満たしておき、半径 10 の赤相の球を計算領域下方におく。密度は赤、青とも 2.6 とする。境界条件は、側面は周期境界条件とし、上面は流出境界、底面は滑べりなし壁の境界条件とした。

Fig.4.1に単一気泡の場合、4.2 に二つの気泡の場合の流れ場の様子を示す。図の実線は界面の位置を赤相と青相の密度が等しくなる点とし、計算領域の高さ方向の中心軸を通る平面内で等高線を描いたものである。背景の濃淡は流れ場の流速に対応しており、気泡左右に見られる濃い部分は下降流が生じている領域、気泡上下の明るい部分は上昇流が生じている領域である。ただし、Fig.4.2 において気泡下部に見られる暗い部分は明るい部分より高速の上昇流領域を示している。

Fig.4.1では、気泡が変形しながら上昇し、気泡両側に渦が発生する様子が見られるが、これは 2 次元の格子ガス法を用いて行った計算 [7, 8] と同様の結果となっている。気泡径に基づく Reynolds 数は 18.6、Etovos 数は 11.5、Morton 数は 0.00396 と求められるが、これらの無次元数の範囲は回転楕円体状（上下に潰れた球形）の気泡に対応することが知られており [10]、妥当な結果となっている。Fig.4.2 では、下方の気泡が上方の気泡より早く上昇し、上方の気泡に追い付くことにより合体する様子がわかる。合体前後も含めて、これらの上昇気泡の挙動は、同一の条件ではないものの実験によって観察されているものと定性的に一致しており [11]、計算プログラム及び並列化は問題なく行われていると考えることができる。

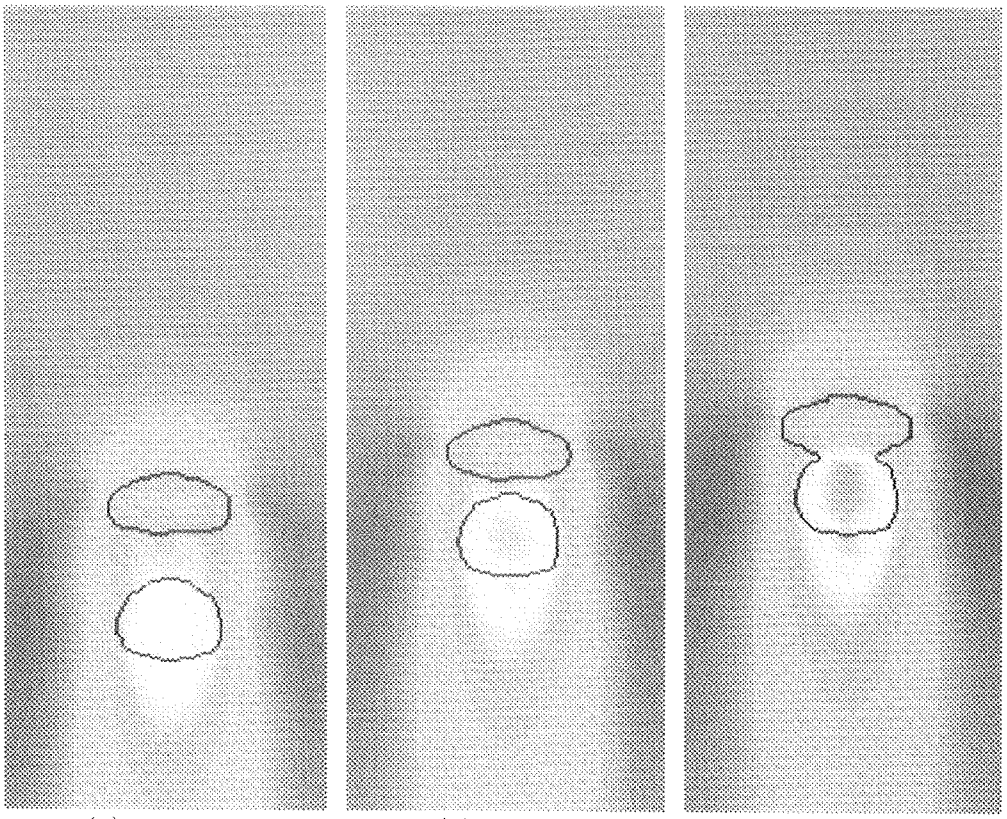


(a) t=0　(b) t=140　(c) t=300

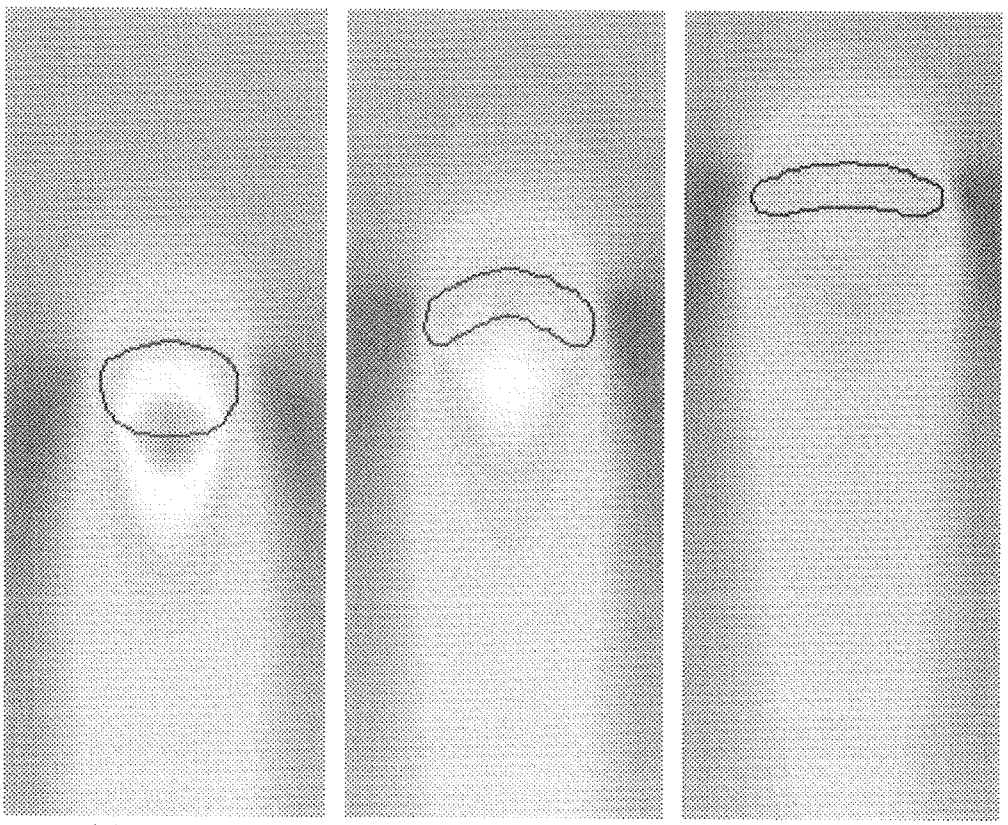


(d) t=440　(e) t=660　(f) t=940

Fig. 4.1　Simulation of single rising bubble



(a) t=1100　(b) t=1180　(c) t=1220



(d) t=1300　(e) t=1400　(f) t=1600

Fig. 4.2　Simulation of two rising bubbles

### 4.2 並列化効率

ワークステーションクラスターでの計算時間の測定結果の例を以下に示す。計算時間は、上昇気泡のシミュレーションにおいて、計算格子数を $20\times20\times120$ としたものと $80\times80\times120$ としたものについて、20 ステップ毎に流れ場を出力することを 5 回繰り返して計算全体の経過時間として測定したものである。流れ場の計算では、$2\times2\times2$ の格子毎に空間平均を行っている。使用したワークステーションの機種は DEC au600 (alpha 21164: 600MHz)、または COMPAQ XP1000 (alpha 21264: 500MHz) である。ワークステーションの台数はそれぞれの機種について 1～4 台であり、ルートワークステーションでの測定時間とした。いずれの計算も、ワークステーション上には OS 関係以外のプロセスがない状態で行った。領域分割は全て均等な分割とした。コンパイルリンクにはそれぞれ純正の Unix 上で Digital Fortran、及び並列計算環境パッケージ Digital PSE の MPI を使用した。

Table 4.1 Calculation time on workstation cluster

| 計算機 | DEC au600 | | | | COMPAQ XP1000 | | | |
|---|---|---|---|---|---|---|---|---|
| 計算格子 | 20 x 20 x 120 | | 80 x 80 x 120 | | 20 x 20 x 120 | | 80 x 80 x 120 | |
| | (秒) | (倍) | (秒) | (倍) | (秒) | (倍) | (秒) | (倍) |
| 1 台 | 217.41 | 1.000 | 3450.95 | 1.000 | 115.34 | 1.000 | 1870.77 | 1.000 |
| 2 台 | 109.09 | 1.993 | 1795.13 | 1.922 | 59.05 | 1.953 | 976.40 | 1.916 |
| 3 台 | 73.62 | 2.953 | 1226.48 | 2.814 | 39.72 | 2.904 | 691.25 | 2.706 |
| 4 台 | 54.48 | 3.991 | 964.04 | 3.580 | 30.64 | 3.765 | 538.25 | 3.476 |

まず、1 台の計算では、DEC au600 よりも、COMPAQ XP1000 が 1.8 倍ほど早いことがわかる。これは、本シミュレーションに対しての、ワークステーション単体の性能である。並列計算による速度向上率は、いずれの機種も計算規模の小さいケースの方が大きくなっており、データ転送の影響が大きいことがわかる。並列化は 1 方向の領域分割を行っており、本シミュレーションでは格子数 120 の方向が分割されている。データ転送は $20\times20$、あるいは $80\times80$ の平面に対して行われるため、ここで比較している 2 つのケースでは転送量は 16 倍異なることになる。データ転送は $100Mb/s$ のイーサネットを通して行われるため転送量が並列化効率に及ぼす影響は大きく、計算規模が大きいケースでは転送の影響が健在化したものと考えられる。並列化効率は計算規模により変わるものの、数台からなるワークステーションクラスターを用いる場合においては、実用上十分な速度向上とみなすことができる。

同じ計算規模、同じ台数において、機種の違いを比べると、早い機種の方が速度向上率は低下している。これは、計算処理が高速に行われることにより、ネットワークの速度に規定されるデータ転送の影響が相対的に大きく現れたものと考えられる。ワークステーションクラスター

は、スーパーコンピュータに比べ安価であり、小さな研究グループで使う場合、取り扱いが容易な点で便利ではあるが、プロセッサーの性能が高くなるほどネットワークの性能も考慮した構成を考える必要があることがわかる。

### 4.3 異なるタイプのワークステーションからなるクラスターの場合

前節ではDEC au600、あるいはCOMPAQ XP1000を4台まで用いた計算結果を示したが、ここでは、それらを同時に使用した8台のワークステーションからなるクラスターによる並列計算を行う。まず、領域分割は等分割としたまま2台から8台まで台数を増やした場合の計算時間を以下に示す。

Table 4.2 Calculation time with different workstations

| 計算格子 | 20 x 20 x 120 | | 80 x 80 x 120 | |
|---|---|---|---|---|
| | (秒) | (倍) | (秒) | (倍) |
| 2台 | 112.48 | 1.000 | 1751.94 | 1.000 |
| 4台 | 54.60 | 2.060 | 959.94 | 1.825 |
| 6台 | 35.86 | 3.137 | 662.04 | 2.646 |
| 8台 | 28.31 | 3.973 | 518.43 | 3.379 |

ここで2台とはDEC au600とCOMPAQ XP1000を1台ずつ使用することであり、4台とは2台ずつ、6台とは3台ずつ、8台とはそれぞれ4台ずつを使用するということである。まず、2台、4台のケースで見ると、いずれも、DEC au600の機種を2台、4台用いた場合の計算時間と、ほぼ同じとなっており、早い機種を混在させた効果は現れていない。ここでは、領域分割を均等に行っているため、早い機種が所定の計算を先に終えてもデータ転送の段階で遅い機種の計算が終了するのを待っている必要があるためである。このため、計算時間は遅い機種のものとなっている。したがって6台、8台を用いた場合の計算時間はDEC au600を6台、8台用いたものと同じと考えられる。

そこで、計算領域の大きさをDEC au600とCOMPAQ XP1000とで変えた計算を行った結果が、次の表に示すものである。ここでは、DEC au600に割り当てる計算領域の大きさに対して、COMPAQ XP1000に対してはfactorをかけた大きさを割り当てるものとする。すなわち、factorが2であれば、XP 1000はDEC au600の2倍の大きさの計算領域を計算することになる。それぞれの機種を4台ずつ、全て使用している。

Table 4.3 Effect of the difference in domain size

| 計算格子 | 20 x 20 x 120 | | 80 x 80 x 120 | |
|---|---|---|---|---|
| | (秒) | (倍) | (秒) | (倍) |
| factor=1.00 | 28.07 | 1.000 | 513.11 | 1.000 |
| factor=1.14 | 26.60 | 0.948 | 499.10 | 0.973 |
| factor=1.31 | 25.23 | 0.899 | 461.70 | 0.900 |
| factor=1.50 | 23.16 | 0.825 | 438.43 | 0.854 |
| factor=1.73 | 22.57 | 0.804 | 405.36 | 0.790 |
| factor=2.00 | 22.85 | 0.814 | 388.47 | 0.757 |
| factor=2.33 | 23.45 | 0.835 | 396.07 | 0.772 |
| factor=2.75 | 24.04 | 0.856 | 407.11 | 0.793 |
| factor=3.29 | 24.84 | 0.885 | 419.02 | 0.817 |

計算速度の倍率はfactor=1の計算時間を基準としたものである。単体性能が1.8倍程度であったため、計算領域の大きさも1.7～2倍程度にとると最も計算時間が短くなることがわかる。ただし、最適なfactorの値は計算領域全体の大きさに依存しているため、本プログラムにおいては、指定された計算領域の大きさを基に計算の始めに最適なfactorの値を計算することとした。

ここでは、factor=1.00として粒子の移動に関わる計算を数ステップ分実行し、その平均の経過時間をDEC au600とCOMPAQ XP1000とで測り、比をとることにより最適なfactorの値とした。この方法により求めたfactorの値と計算時間は、小さい規模の計算に対して2.27、23.19秒、大きい規模の計算に対して1.91、393.55秒であった。これらは、何回かの測定のうちの最小の計算時間のケースであるが、ばらつきは1%以下であった。これは小さい規模の計算に対しては、DEC au600の1台の場合を基準とすると速度向上率は9.375倍、COMPAQ XP1000を基準とすると4.974倍となる。大きい規模の計算に対してはDEC au600を基準とすると8.769倍、COMPAQ XP1000を基準とすると4.754倍となる。計算効率は必ずしも満足な値ではないが、既存のワークステーションを利用して並列計算を行い、ある程度の速度向上をはかり、かつ多くのメモリーを使用するような場合、本手法は簡便な手段の一つであると考えられる。

より計算効率の高い動的負荷分散を目指すことも可能であるが、本プログラムではそこまでは行っていない。それは、本プログラムが多数のプロダクションランを行うためではなく、今後のモデル開発、あるいは多様な流れ場の計算のためのベースとして開発されているためであり、プログラムの主要な部分の変更が見込まれるためである。

# 5. まとめ

本報告書では、格子ボルツマン法による3次元二成分二相流シミュレーションコードの開発と、MPIライブラリを用いた並列化について記述した。ワークステーションクラスターは4台のDEC au600 及び 4台の COMPAQ XP1000を100Baseイーサネットによりスイッチングハブを介して接続したものである。ネットワークが遅いため、計算規模が大きくなりデータ転送量が増えるにつれて並列計算効率は低下したが、性能、価格、使い勝手等を考慮すると、ワークステーションクラスターによる並列計算は十分実用に耐えるものと思われる。また、本報告書では、研究室などにある様々なワークステーションを利用することを想定して、異なるワークステーションからなるクラスターを用いて、計算速度の違いに応じた領域分割を含む並列計算を試みた。簡便で効率が良く、かつプログラム変更にも柔軟に対応できる負荷分散の手法を検討することは、今後の重要な課題である。

なお、本報告書で使用した格子ボルツマン二相流シミュレーションコードは、(財)電力中央研究所との共同研究「気液二相流のミクロ／マクロ・モデルによる解析に関する研究」及び筑波大学との協力研究「超音波による二相流制御に関する研究」において使用するものである。

# 参考文献

[1] 数値流体力学編集委員会 編：”数値流体力学シリーズ５”、東京大学出版会、東京（1995）.

[2] G. A. Bird: “Molecular Gas Dynamics and the Direct Simulation of Gas Flows”, (Clarendon, Oxford, 1994).

[3] 上田 顕：”コンピュータシミュレーション”、朝倉書店、東京 (1990).

[4] D. H. Rothman and S. Zaleski: ”Lattice-gas models of phase separation: interfaces, phase transitions, and multiphase flow,” Rev. Mod. Phys., **66**, 1417(1994).

[5] 矢部 孝、観山 正見、椛島 成治：“パソコンによるシミュレーション物理”、朝倉書店、東京（1992）.

[6] 矢部 孝、川田 重夫、福田 昌宏：”シミュレーション物理入門”、朝倉書店、東京（1989）.

[7] 渡辺 正、海老原 健一、蕪木 英雄：”非浸透格子ガスモデルによる二相流シミュレーションコードの開発並びに並列化”、JAERI-Data/Code 97-056(1998).

[8] 渡辺 正、海老原 健一、加藤 克海：”ワークステーションクラスターによる格子ガス二相流シミュレーションコードの並列計算”、JAERI-Data/Code 99-029(1999).

[9] D. Grunau, S. Chen, and K.Eggert: ”A lattice Boltzmann model for multiphase fluid flows,” Phys. Fluids A **5** (10), 2557 (1993) .

[10] R.Clift, et al.: ”Bubbles, Drops, and Particles”, (Academic Press, New York, 1978).

[11] J. R. Crabtree and J. Bridgwater: ”Bubble coalescence in viscous liquids,” Chem. Eng. Sci. **26**, 839 (1971).

This is a blank page.

# 国際単位系 (SI) と換算表

**表 1　SI 基本単位および補助単位**

| 量 | 名称 | 記号 |
|---|---|---|
| 長さ | メートル | m |
| 質量 | キログラム | kg |
| 時間 | 秒 | s |
| 電流 | アンペア | A |
| 熱力学温度 | ケルビン | K |
| 物質量 | モル | mol |
| 光度 | カンデラ | cd |
| 平面角 | ラジアン | rad |
| 立体角 | ステラジアン | sr |

**表 2　SI と併用される単位**

| 名称 | 記号 |
|---|---|
| 分，時，日 | min，h，d |
| 度，分，秒 | °，′，″ |
| リットル | l，L |
| トン | t |
| 電子ボルト | eV |
| 原子質量単位 | u |

1 eV = 1.60218 × $10^{-19}$ J

1 u = 1.66054 × $10^{-27}$ kg

**表 5　SI 接頭語**

| 倍数 | 接頭語 | 記号 |
|---|---|---|
| $10^{18}$ | エクサ | E |
| $10^{15}$ | ペタ | P |
| $10^{12}$ | テラ | T |
| $10^{9}$ | ギガ | G |
| $10^{6}$ | メガ | M |
| $10^{3}$ | キロ | k |
| $10^{2}$ | ヘクト | h |
| $10^{1}$ | デカ | da |
| $10^{-1}$ | デシ | d |
| $10^{-2}$ | センチ | c |
| $10^{-3}$ | ミリ | m |
| $10^{-6}$ | マイクロ | μ |
| $10^{-9}$ | ナノ | n |
| $10^{-12}$ | ピコ | p |
| $10^{-15}$ | フェムト | f |
| $10^{-18}$ | アト | a |

**表 3　固有の名称をもつ SI 組立単位**

| 量 | 名称 | 記号 | 他の SI 単位による表現 |
|---|---|---|---|
| 周波数 | ヘルツ | Hz | $s^{-1}$ |
| 力 | ニュートン | N | m·kg/$s^2$ |
| 圧力，応力 | パスカル | Pa | N/$m^2$ |
| エネルギー，仕事，熱量 | ジュール | J | N·m |
| 工率，放射束 | ワット | W | J/s |
| 電気量，電荷 | クーロン | C | A·s |
| 電位，電圧，起電力 | ボルト | V | W/A |
| 静電容量 | ファラド | F | C/V |
| 電気抵抗 | オーム | Ω | V/A |
| コンダクタンス | ジーメンス | S | A/V |
| 磁束 | ウェーバ | Wb | V·s |
| 磁束密度 | テスラ | T | Wb/$m^2$ |
| インダクタンス | ヘンリー | H | Wb/A |
| セルシウス温度 | セルシウス度 | ℃ | |
| 光束 | ルーメン | lm | cd·sr |
| 照度 | ルクス | lx | lm/$m^2$ |
| 放射能 | ベクレル | Bq | $s^{-1}$ |
| 吸収線量 | グレイ | Gy | J/kg |
| 線量当量 | シーベルト | Sv | J/kg |

**表 4　SI と共に暫定的に維持される単位**

| 名称 | 記号 |
|---|---|
| オングストローム | Å |
| バーン | b |
| バール | bar |
| ガル | Gal |
| キュリー | Ci |
| レントゲン | R |
| ラド | rad |
| レム | rem |

1 Å = 0.1 nm = $10^{-10}$ m

1 b = 100 $fm^2$ = $10^{-28}$ $m^2$

1 bar = 0.1 MPa = $10^5$ Pa

1 Gal = 1 cm/$s^2$ = $10^{-2}$ m/$s^2$

1 Ci = 3.7 × $10^{10}$ Bq

1 R = 2.58 × $10^{-4}$ C/kg

1 rad = 1 cGy = $10^{-2}$ Gy

1 rem = 1 cSv = $10^{-2}$ Sv

（注）

1. 表 1 − 5 は「国際単位系」第 5 版，国際度量衡局 1985 年刊行による。ただし，1 eV および 1 u の値は CODATA の 1986 年推奨値によった。
2. 表 4 には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。
3. bar は，JIS では流体の圧力を表わす場合に限り表 2 のカテゴリーに分類されている。
4. EC 閣僚理事会指令では bar，barn および「血圧の単位」mmHg を表 2 のカテゴリーに入れている。

## 換算表

| 力 | N(=$10^5$ dyn) | kgf | lbf |
|---|---|---|---|
| | 1 | 0.101972 | 0.224809 |
| | 9.80665 | 1 | 2.20462 |
| | 4.44822 | 0.453592 | 1 |

粘度　1 Pa·s(N·s/$m^2$) = 10 P(ポアズ)(g/(cm·s))

動粘度　1 $m^2$/s = $10^4$ St(ストークス)($cm^2$/s)

| 圧力 | MPa(=10 bar) | kgf/$cm^2$ | atm | mmHg(Torr) | lbf/$in^2$(psi) |
|---|---|---|---|---|---|
| | 1 | 10.1972 | 9.86923 | 7.50062 × $10^3$ | 145.038 |
| | 0.0980665 | 1 | 0.967841 | 735.559 | 14.2233 |
| | 0.101325 | 1.03323 | 1 | 760 | 14.6959 |
| | 1.33322 × $10^{-4}$ | 1.35951 × $10^{-3}$ | 1.31579 × $10^{-3}$ | 1 | 1.93368 × $10^{-2}$ |
| | 6.89476 × $10^{-3}$ | 7.03070 × $10^{-2}$ | 6.80460 × $10^{-2}$ | 51.7149 | 1 |

| エネルギー・仕事・熱量 | J(=$10^7$ erg) | kgf·m | kW·h | cal(計量法) | Btu | ft·lbf | eV |
|---|---|---|---|---|---|---|---|
| | 1 | 0.101972 | 2.77778 × $10^{-7}$ | 0.238889 | 9.47813 × $10^{-4}$ | 0.737562 | 6.24150 × $10^{18}$ |
| | 9.80665 | 1 | 2.72407 × $10^{-6}$ | 2.34270 | 9.29487 × $10^{-3}$ | 7.23301 | 6.12082 × $10^{19}$ |
| | 3.6 × $10^6$ | 3.67098 × $10^5$ | 1 | 8.59999 × $10^5$ | 3412.13 | 2.65522 × $10^6$ | 2.24694 × $10^{25}$ |
| | 4.18605 | 0.426858 | 1.16279 × $10^{-6}$ | 1 | 3.96759 × $10^{-3}$ | 3.08747 | 2.61272 × $10^{19}$ |
| | 1055.06 | 107.586 | 2.93072 × $10^{-4}$ | 252.042 | 1 | 778.172 | 6.58515 × $10^{21}$ |
| | 1.35582 | 0.138255 | 3.76616 × $10^{-7}$ | 0.323890 | 1.28506 × $10^{-3}$ | 1 | 8.46233 × $10^{18}$ |
| | 1.60218 × $10^{-19}$ | 1.63377 × $10^{-20}$ | 4.45050 × $10^{-26}$ | 3.82743 × $10^{-20}$ | 1.51857 × $10^{-22}$ | 1.18171 × $10^{-19}$ | 1 |

1 cal = 4.18605 J(計量法)
= 4.184 J　(熱化学)
= 4.1855 J　(15 °C)
= 4.1868 J(国際蒸気表)

仕事率　1 PS(仏馬力)
= 75 kgf·m/s
= 735.499 W

| 放射能 | Bq | Ci |
|---|---|---|
| | 1 | 2.70270 × $10^{-11}$ |
| | 3.7 × $10^{10}$ | 1 |

| 吸収線量 | Gy | rad |
|---|---|---|
| | 1 | 100 |
| | 0.01 | 1 |

| 照射線量 | C/kg | R |
|---|---|---|
| | 1 | 3876 |
| | 2.58 × $10^{-4}$ | 1 |

| 線量当量 | Sv | rem |
|---|---|---|
| | 1 | 100 |
| | 0.01 | 1 |

(86 年 12 月 26 日現在)

# 格子ボルツマン法による二相流シミュレーションコードの開発並びに並列化