［共同研究成果］

コンパクトな計算機によるリアルタイム流体解析の実現に向けて

— 中間報告 —

松岡　浩
東北大学電気通信研究所(客員)
一般財団法人高度情報科学技術研究機構
菊池　範子
株式会社サイエンス・サービス

1. はじめに

筆者らは、東北大学サイバーサイエンスセンターとの共同研究公募制度により、平成 27 年度から「連続感度解析の実現を目指した整数型格子ボルツマン法流体解析手法の開発」を行ってきた。本研究は、平成 29 年度まで実施しているが、ここでは、平成 28 年度までの成果として、特に“コンパクトな計算機によるリアルタイム流体解析の実現可能性”について中間報告を行う。

筆者らは、これまでの研究を通じて、“リアルタイム流体解析”を実現するには、４つの技術課題を解決する必要があると考えている。本原稿では、このうちの２つ、①“超並列格子点演算”の実現 と ②“連続動画可視化”の実現に関する考察を述べる。このほか、③実測データ等をシミュレーション過程にリアルタイムでフィードバックする“計測融合機能”の実現、④幅広いレイノルズ数領域で格子ガス法によるシミュレーションを可能にする“粘性制御機能”の実現、が重要な技術課題として挙げられる。後者の２課題は、さらに研究が進んだ時点でご報告したい。

なお、本稿の本文は松岡が執筆し、掲載したシミュレーション事例に関する計算及び可視化は菊池が分担した。

本研究の最終目標は、「流体中を運動する物体が係わる工学システムのものづくり設計において、斬新な設計アイデアの探索に役立つ“リアルタイム流体シミュレーション設計ツール”を実現すること」である。より具体的には、

「ものづくり設計のために必要とされる流体シミュレーションの解像度を維持しながら、
①流体中または流体周辺に存在する物体や構造物の位置や形状（“設計上の境界条件”)、
②工学的に制御できるある特定の場所の流体運動の速さや向き（“運転上の環境条件”)、
などを連続的かつ気ままに変化させながら、その効果を実際の物理現象の実時間に近い応答としてビジュアルに観察できる、コンパクトな規模の計算機による超高速流体シミュレーションシステムの構築」を目指している。

この最終目標への道のりは長いが、本稿では、中間報告として、第２章で、筆者が用いている流体解析手法である“多速さ格子ガス法”の特徴を述べ、第３章で「円柱後流解析に基づく実時間計算の実現可能性」について、第４章で「１格子点１ビット幅演算の二相流解析への適用性」について考察し、最後に、“超並列格子点演算”と“連続動画可視化”の実現性を総括する。

2.“多速さ格子ガス法”の特徴

前章で述べたような“リアルタイム流体解析”を実現していく場合、筆者は、ナビエ･ストークス方程式を差分化して解く従来の王道的な CFD 手法よりも、よりミクロな視点でボルツマン方程式を離散化した格子ボルツマン方程式に従い、整数型の確率変数を用いて時間発展を追跡していく“多速さ格子ガス法（整数型格子ボルツマン法)”[1]を用いた方が有利であると考えている。

本手法は、流体が存在する空間中に格子を張り、多数の仮想粒子が、その格子点上のみで他の

仮想粒子と衝突して進行の向きを変えながら格子点間を走行していく様子を平均化(疎視化)して流体の挙動を模擬する方法である。このとき、仮想粒子がもつ質量・運動量・エネルギーが衝突の前後で保存されるような粒子衝突を想定する限り、その挙動は、自然界におけるある条件下の流体挙動とかなり似かよったものになる。しかしながら、仮想粒子は、格子点が存在する位置の間しか移動できないので、速度の大きさも向きも離散的な値をとり、自然界の流体分子のように連続的な値をとることができない。この制約によって、格子ガス法が導くマクロな挙動は、連続流体を仮定している CFD が導く流体挙動とは多少異なったものになる。しかしながら、テシャラが考案した方法 [2] に従い、異なる速さをもつ仮想粒子どうしの衝突頻度をうまく調節すれば、この“多少の差異”を解消することができる。これにより、“多速さ格子ガス法”は、CFD を代替できる精度をもったシミュレーション手法となりえる。この詳細は、SENAC の 2016 年 10 月号[3]に記載しているので、そちらを参照されたい。

筆者が、この“多速さ格子ガス法”を“リアルタイム流体解析”を実現するツールの最有力候補と考える理由は、本手法が次の特徴をもつからである。

①ひとつの格子点に関する衝突等の状態変化演算を１ビット幅で超並列に実行することができる。
②複雑な形状で時間変化する境界条件でも比較的容易にシミュレーション計算過程に取り込める。
③時間発展計算をビット演算で行えるため打切り誤差の蓄積もなくどんな流れでも計算はできる。

３．円柱後流解析に基づく実時間計算の実現性可能性

ものづくり産業において“設計アイデアの探索”という目的からみた場合、これまで開発してきた“多速さ格子ガス法流体解析コード”が、コンパクトな計算機規模で必要な解像度のシミュレーション結果を実時間で連続的に提供できるのか？という課題が重要である。

この実現性を評価するため、本共同研究を開始した時点では、「小型冷蔵庫の大きさで 100V 電源駆動が可能な SX-ACE Lite」を“コンパクトなものづくり設計用計算機”の代表例と考え、それと同一規模の SX-ACE32 ノード(32CPU=128 コア)による計算を実行して性能評価を試みた。

はじめに、３次元空間中に約 5000 万個の格子点をＸＹＺの各方向に並べ、直方体形状の格子点配列を作る。各格子点は、その内部に４次元目の座標として R＝0, 1, 2, 3 の位置を識別できる自由度をもつとする。これが**図１**であり、４次元面心超立方体格子を３次元空間へ投影した姿である。

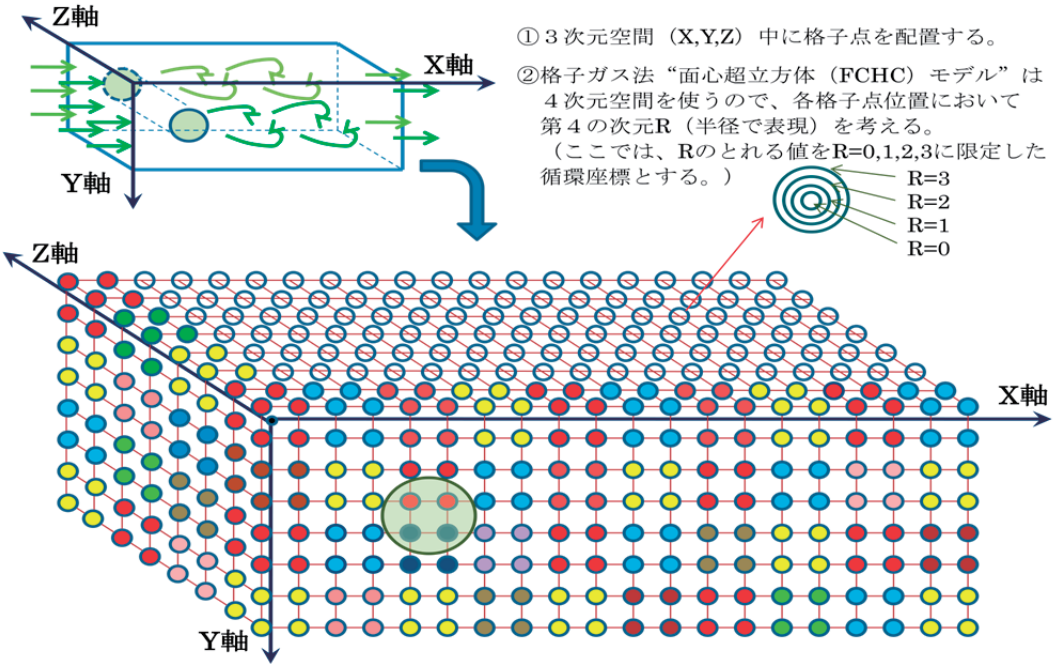

図１．円柱後流をシミュレーションするための４次元面心超立方体格子（３次元投影）

具体的には、３次元縮退格子として、Ｘ方向に 768 個、Ｙ方向に 256 個、Ｚ方向に 256 個の格子点を配置した。また、シミュレーション計算を開始する時刻ステップ 0 の時点で、各格子点には、そこに存在できる仮想粒子の最大数の 20％の数の仮想粒子がランダムな向きに配置されていると仮定した。この結果、疎視化して得られるマクロな流速はゼロであり、流体は、直方体形状の中で静止している。次に、時刻ステップ１の時点から、＋Ｘ向きの速度をもつ仮想粒子をＸ＝0 の位置から注入していく。すると、時刻ステップが進むにつれて、流体全体が＋Ｘ向きのマクロな速度をもつようになる。このとき、＋Ｘ側の先にある直方体出口においては、出口直前に存在する格子点上の仮想粒子配置を、出口直後に存在する格子点の仮想粒子配置にコピーして、出口におけるマクロな流速の勾配がゼロになるという境界条件を近似的に実現した。また、±Ｙ方向と±Ｚ方向には、周期的境界条件を適用した。そして、この流れの中の入り口に近い位置に、“Ｚ方向の中心軸をもつ無限大の長さの円柱”を置き、その後流に生じる流体挙動を計算した。

図２は、３次元的に生じている円柱後流の流速変化のうち、Ｚ軸に垂直なある平断面上の流速分布を示している。はじめ静止していた流体中におかれた円柱の後流には、流体の速度が増すにつれて、最初は双子渦が生成し、次にその長さが伸びて尾が振れ出し、長い尾が切れて、最後は、短い尾が振れ続ける“カルマン渦”になった。これは現実の過渡変化挙動をよく模擬している。

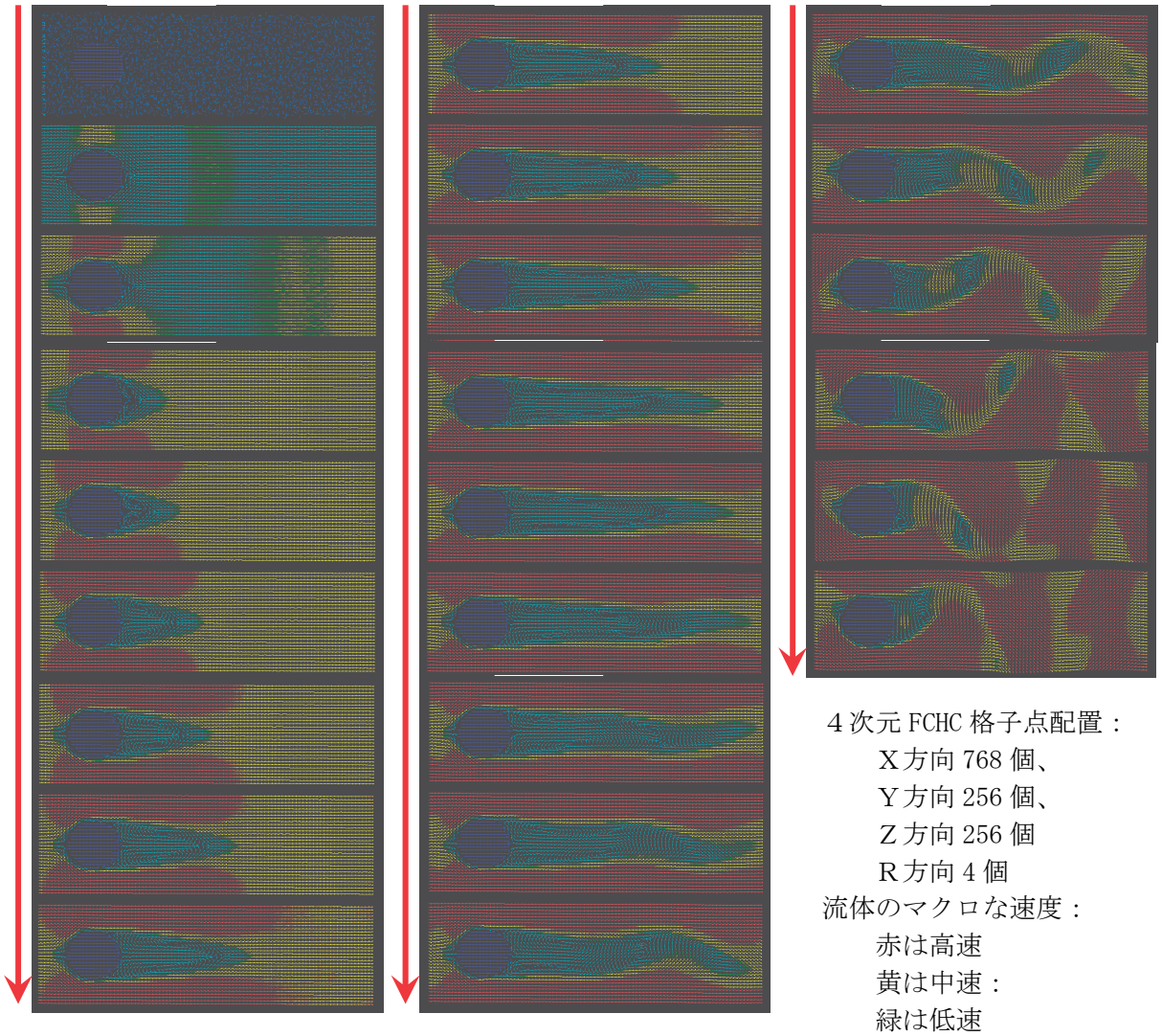

図２．加速される流体中に置かれた円柱後流に発生する渦の過渡変化（Z 軸に垂直なある面上）

本計算は、東北大学サイバーサイエンスセンターの SX-ACE32 ノード(128 ベクトルコア)を利用し、11,776 時刻ステップの計算を実行するのに 716 秒という計算時間を要した。**図２**では、24 個のスナップショットの時系列画像を示しているが、スナップショット画像が得られる時間間隔は、716 秒÷24≒30 秒である。従って、格子点規模が約 5000 万個ではなく約 1000 万個であれば、スナップショット画像が得られる時間間隔は約 6 秒と推測される。ここで、本年(2018 年)発売される SX-Aurora TSUBASA を利用した場合を考えると、SX-ACE に比べて CPU 性能が 10 倍程度になっているため、スナップショット画像は 0.5 秒程度の時間間隔で得られるはずである。

他方、**図２**の最後の方のスナップショット画像の数枚を見ると、カルマン渦の尾の振れる周期とスナップショット画像の時間間隔が同程度であることがわかる。そして、カルマン渦の周期は、一般的に D/(St･U)で与えられ、ここで、D 代表長さ(円柱の直径)、U 代表速さ(マクロな流速)、St ストローハル数(0.2 程度)であることが知られているので、U が毎秒 D の 10 倍程度の流れでは、0.5 秒程度になる。これは、上述した SX-Aurora TSUBASA によるシミュレーションで得られると予想されるスナップショット画像の計算時間間隔と同じである。

以上のことから、SX-Aurora TSUBASA の数 10 ノード用いれば、1000 万格子点規模の単相流体挙動解析については、実時間に近いシミュレーションを実現できそうなレベルにあることがわかる。

４．″１格子点１ビット幅演算″の二相流解析への適用性

4.1　はじめに

第３章で、単相流体挙動のリアルタイムシミュレーションの実現性について考察した。しかしながら、現実のものづくり産業では、現実世界におけるもっと複雑な事象に関心がある。たとえば、実船の船舶設計では、海水と空気の二相流中で船舶が進行する際の抵抗力の評価が必要であり、プロペラ部分ではキャビテーションの評価も重要である。また、原子力発電プラントの原子炉冷却系等の設計では、水と水蒸気の気液二相流の評価が必要になる。ところが、二相流の流体解析計算は、同じ格子点数規模の単相流の流体解析計算に比べて、計算負荷が格段に大きくなり、実時間計算の実現は相当困難であるように思われる。

このような背景を踏まえ、ここでは、“多速さ格子ガス法流体解析コード”において超高速計算性能を生み出す源であった“１格子点１ビット幅演算”が、単相流解析だけではなく、二相流解析にも適用できることを確認することとした。このため、まずは、二相流過渡変化の中で一番単純な“無重力下での相分離現象”のシミュレーションを試みた。

4.2　仮想粒子の存否情報を表す整数型配列と超並列ビット演算による衝突計算の実現

ここで用いる“多速さ格子ガス法流体解析コード”は、第３章で述べたように“４次元面心超立方体(FCHC)格子”を採用している。多数の仮想粒子が、４次元空間(X,Y,Z,R)中に張られた FCHC 格子上を動きまわるが、仮想粒子どうしの“衝突”は、格子点上においてのみ生じると仮定する。この結果、エネルギーが 0 でない仮想粒子は、次の時刻ステップで、今いる格子点の近傍にある格子点にだけに飛び移るので、有限個の離散的な４次元速度だけをもちうる。特にここで用いる“非熱流体 FCHC モデル”では、エネルギーが 0 でない仮想粒子については 48 種類の異なる速度をもちうるが、各格子点において、これらの速度をもつ仮想粒子は、高々１個しか存在できないものとする。なお、エネルギーが 0 の静止仮想粒子は、各格子点に複数個（α）存在してもよい。

以上のことから、各格子点における物理状態は、エネルギーが 0 の静止仮想粒子の識別を表すα個の種類と、エネルギーが 0 でない仮想粒子の速度による識別を表す 48 種類のそれぞれについて、その仮想粒子が存在[１]するか？否［0］か？という１ビット情報を与えることで表現できる。本流体解析コードでは、“１格子点１ビット幅の演算”を効率よく実行させるため、ひとつの格子点ごとに（48＋α）種類の速度をもつ仮想粒子の存否情報をまとめて配列に記憶するのではなく、ひとつの速度ごとに 32 個の４次元格子点における仮想粒子の存否情報をまとめて配列に記憶する。

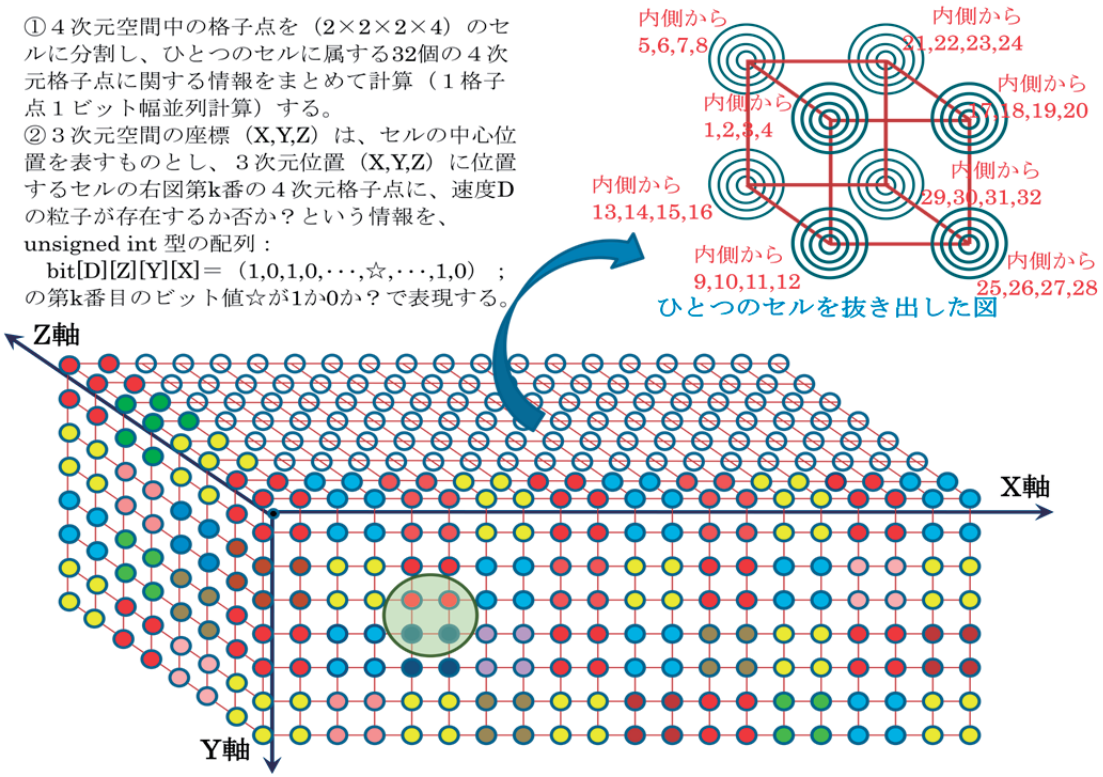

図３．位置(X, Y, Z)に存在する(2×2×2×4)セルとその中に存在する 32 個の格子点(右上)

具体的には、４次元空間中に広がる４次元 FCHC 格子を、2(X 方向)×2(Y 方向)×2(Z 方向)×4(R 方向)の４次元直方体領域の“セル”に分割する。ひとつのセルに存在する 2×2×2×4＝32 個の４次元格子点における仮想粒子の存否情報をひとまとめにして、ある特定の速度 D をもつ仮想粒子ごとに、符号なしの 32 ビット整数型配列(unsigned int 型配列)に記憶する。**図３**は、３次元空間の位置(X,Y,Z)に存在する“セル”の様子と、そのセルに含まれる 32 個の４次元格子点を右上の図で表現している。この 32 個の４次元格子点のそれぞれに速度 D をもつ仮想粒子が存在するか否か？の存否情報が、**図３**の中の説明文に記載された配列 bit[D][Z][Y][X]の各ビットに記憶される。

さて、各格子点では、シミュレーション計算の時刻ステップの刻みごとに、隣接格子点から飛来する仮想粒子が自分の格子点に“到着”し、到着した仮想粒子は、衝突によって向きを変えた後、別の隣接格子点に向かって“出発”していく。このとき、仮想粒子の進行向きの変更は、衝突によって瞬時に起こると仮定されているので、同じ時刻ステップでも、速度 D をもつ仮想粒子が位置（X,Y,Z）の“セル”に存在するか否か？を表現する配列としては、“到着分布用”の bitarv[D][Z][Y][X]と“出発分布用”の bitstr[D][Z][Y][X]の２種類を考える必要がある。

具体的には、例えば、以下のように、符号なし整数型配列に、仮想粒子の存否情報を記憶する。まずは、“到着粒子の存否情報を記憶した配列”として、次の例を考える。

bitarv[D][Z][Y][X]＝10011001111011011110011101111011;

ここで、一番右から n 番目のビットが「1」であれば、位置(X,Y,Z)にあるセルの第 n 番目の４次元格子点には、速度 D をもった到着粒子が「存在する」ことを示している。もし、一番右から m 番目のビットが「0」であれば、位置(X,Y,Z)にあるセルの第 m 番目の４次元格子点には、速度 D をもった到着粒子が「存在しない」ことを示している。(1<=n,m<=32)

出発粒子の場合も、同様であり、例えば、次のようになる。

“出発粒子の存否情報を記憶した配列”

bitstr[D][Z][Y][X]＝01001001111010111011101101111001;

速度 a と速度 b をもつ２つの仮想粒子が同じ位置の格子点に到着して、運動量とエネルギーを保存しながら、速度 c と速度 d の２つの仮想粒子になって出発していく衝突の過程は、bitarv[a][Z][Y][X]と bitarv[b][Z][Y][X]の対応するビットがともに「1」で、かつ、bitarv[c][Z][Y][X]と bitarv[d][Z][Y][X]の対応するビットがともに「0」の場合に生じる。そして、衝突した結果は、bitstr[a][Z][Y][X]と bitstr[b][Z][Y][X]の対応するビットがともに「0」で、かつ、bitstr[c][Z][Y][X]と bitstr[d][Z][Y][X]の対応するビットがともに「1」になることで表現できる。また、この逆の過程も、運動量とエネルギーを保存するはずであるから、「速度 c と速度 d をもつ２つの仮想粒子が同じ位置の格子点に到着して、速度 a と速度 b の２つの仮想粒子になって出発していく衝突の過程」も同様に生じるであろう。このような衝突過程における仮想粒子の存在状態の変化は、次のビット演算で計算できる。ただし、「&」は“論理積”、「｜」は“論理和”、「~」は“論理否定”、「＾」は“排他的論理和”のビット演算を表している。

```
mk
＝(bitarv[a][Z][Y][X] & bitarv[b][Z][Y][X] & ~bitarv[c][Z][Y][X] & ~bitarv[d][Z][Y][X])
 |(~bitarv[a][Z][Y][X] & ~bitarv[b][Z][Y][X] & bitarv[c][Z][Y][X] & bitarv[d][Z][Y][X]);
bitstr[a][Z][Y][X] = bitarv[a][Z][Y][X] ^ mk;
bitstr[b][Z][Y][X] = bitarv[b][Z][Y][X] ^ mk;
bitstr[c][Z][Y][X] = bitarv[c][Z][Y][X] ^ mk;
bitstr[d][Z][Y][X] = bitarv[d][Z][Y][X] ^ mk;
```

以上のようなビット演算により、１ワードの計算をしながら、１セル 32 個の格子点に関する並列計算を実行できることになる。特に、ベクトル計算機でベクトル長 256 ワードの計算を行えば、１回のベクトル命令で、32×256＝8192 個の格子点に係る並列計算を実行できる。

4.3　二相流を扱う場合に仮想粒子の存否情報を表す配列の表現

第３章で述べたように、“１格子点１ビット幅演算”は、単相流の実時間計算をも実現しうる強い威力をもっている。それでは、二相流のシミュレーションを行う場合にも、“１格子点１ビット幅演算”で全ての計算過程をうまく実現できるであろうか？ここでは、“相分離現象のシミュレーション”について、そのことを検証してみる。

２相を取扱うため、２種類の仮想粒子（“赤粒子”と“青粒子”）を考える。各４次元格子点において、エネルギーが 0 でない仮想粒子がもつことを許されるそれぞれの速度は、高々１個の“赤粒子”かあるいは高々１個の“青粒子”がその速度をもつことができるとする。これは、格子ガス法で二相流を模擬する場合によく使われるモデルである。このモデルを使用する場合、各格子点における仮想粒子の存在状態を表現するのに、仮想粒子の存否情報だけではなく、その粒子が“赤粒子”なのか？“青粒子”なのか？という情報も何かの方法で識別できるようにする必要がある。ここでは、上記配列を 32 ビット幅の「unsigned int 型」から 64 ビット幅の「unsigned long int 型」に変更して、右側 32 ビットには、これまでと同じで、仮想粒子の存否情報を記憶させ、左側 32 ビットには、粒子が赤粒子であるか？否か？を「1」or「0」で記憶させることにした。このとき、左側 32 ビットにおける“赤粒子ではない「0」の場合”というのは、“仮想粒子が存在していてそれが「青粒子」の場合”と“そもそも仮想粒子が存在しない空の場合”があることに注意してほしい。

具体的な例を示すと、配列の要素は、次のような 64 ビットの「1」「0」の並びであり、

```
    bitarv[D][Z][Y][X]
  =1000110010110110110100011011110111001100111101101111001110111101;
```

例えば、一番右からｎ番目のビットが「1」であり、かつ、一番右からｎ+32 番目のビットが「1」

であれば、位置(X,Y,Z)にあるセルの第ｎ番目の４次元格子点には、速度Ｄをもつ到着粒子が存在して、それは「赤粒子」である。(1<=n<=32) ここで、一番右からｎ+32 番目のビットが「1」であれば、仮想粒子は必ず存在するはずなので、一番右からｎ番目のビットも必ず「1」になっていなければならない。他方、一番右からｎ+32 番目のビットが「0」であれば、一番右からｎ番目のビットも調べて、このビットが「1」ならば「青粒子」が存在し、このビットが「0」ならば仮想粒子が存在しない空の状態であることがわかる。

4.4　“相分離”を引き起こす計算過程の“１格子点１ビット幅演算”による実現

“赤青粒子モデル”によって“相分離”のシミュレーションを行う場合、赤粒子どうし及び青粒子どうしの間では“引力”を働かせ、赤粒子と青粒子の間では“斥力”を働かせるというモデルが通常採用される。しかしながら、今回のシミュレーションの目的は、“１格子点１ビット幅演算”が二相流解析にも適用できることを確認することにあるので、もっと簡略化したモデル計算を行うことにした。これは、物理的には不適切な部分もあるが、着色された仮想粒子の粒子数をカウントし、その大小を比較する計算を含む点で、“１格子点１ビット幅演算”の適用性を確認するには十分なモデルである。このモデルでは、以下の手順で“相分離”を実現させている。

ある格子点に到着した仮想粒子について、それが“赤粒子”であろうが“青粒子”であろうが気にしないで仮想粒子の衝突による状態変化を計算する。このため、右側 32 ビットに注目して **4.2** 節に述べた衝突過程の計算を行う。左側 32 ビットは、右側 32 ビットにおける仮想粒子の入れ替えの動きに追従させて移動させる。次に、各格子点から出発粒子をいろいろな向きに放出する際に、隣接格子点に存在する「“赤粒子”の数」の値を調べ、その値が大きな隣接格子点の向きに“赤粒子”を多く放出し、その値が小さな隣接格子点の向きに“青粒子”を多く放出させることにする。ただし、“赤粒子”、“青粒子”を問わない出発粒子の速度分布配置は変更せずに、その色だけを変更することとする。また、このとき、“赤粒子”の個数と“青粒子”の個数は保存させるものとする。ここでは、この操作をさらに簡略化して、「仮想粒子の出発分布において、互いに同じ速さで向きが正反対に出発しようとしている“赤粒子”と“青粒子”のペアがたまたま存在し、かつ、“赤粒子の行き先格子点に存在する赤粒子の数”と“青粒子の行き先格子点に存在する赤粒子の数”をカウントして前者が後者よりも小さい場合に、今注目している出発粒子ペアの色を入れ替える。」という操作を行った。

ここで重要なことは、「着色された仮想粒子の粒子数をカウントし、その大小を比較する計算」についても“１格子点１ビット幅演算”を適用できることである。

出発粒子については、配列 bitstr[D][Z][Y][X]の右側 32 ビット中のひとつのビット位置に注目して、すべての速度の種類Ｄについて「1」の数を足し上げれば、そのビット位置に対応した４次元格子点に存在する全出発粒子数(赤粒子数＋青粒子数)を知ることができる。また、左側 32 ビット中のひとつのビット位置に注目して、すべてのＤについて「1」の数を足し上げれば、そのビット位置－32 に対応した４次元格子点に存在する赤粒子の出発粒子数を求めることができる。このとき、すべての D について粒子数を足し上げる計算は、以下のようにして行った。これは、“論理積”と“排他的論理和”によって“半加算器”を６組構成し、それらを“桁上げ”でつないだ形になっている。

```
wa1 = wa2 = wa3 = wa4 = wa5 = wa6 = 0X0000000000000000U;
in1 = bitstr[1][Z][Y][X];
in2 = wa1 & in1;  wa1 = wa1 ^ in1;
in3 = wa2 & in2;  wa2 = wa2 ^ in2;
in4 = wa3 & in3;  wa3 = wa3 ^ in3;
in5 = wa4 & in4;  wa4 = wa4 ^ in4;
in6 = wa5 & in5;  wa5 = wa5 ^ in5;
wa6 = wa6 ^ in6;
```

```
in1 = bitstr[2][Z][Y][X];
in2 = wa1 & in1;  wa1 = wa1 ^ in1;
in3 = wa2 & in2;  wa2 = wa2 ^ in2;
in4 = wa3 & in3;  wa3 = wa3 ^ in3;
in5 = wa4 & in4;  wa4 = wa4 ^ in4;
in6 = wa5 & in5;  wa5 = wa5 ^ in5;
wa6 = wa6 ^ in6;
    ・・・・・
in1 = bitstr[48+α][Z][Y][X];
in2 = wa1 & in1;  wa1 = wa1 ^ in1;
in3 = wa2 & in2;  wa2 = wa2 ^ in2;
in4 = wa3 & in3;  wa3 = wa3 ^ in3;
in5 = wa4 & in4;  wa4 = wa4 ^ in4;
in6 = wa5 & in5;  wa5 = wa5 ^ in5;
wa6 = wa6 ^ in6;

bitnum[1][Z][Y][X] = wa1;
bitnum[2][Z][Y][X] = wa2;
bitnum[3][Z][Y][X] = wa3;
bitnum[4][Z][Y][X] = wa4;
bitnum[5][Z][Y][X] = wa5;
bitnum[6][Z][Y][X] = wa6;
```

以上の計算により、位置(X,Y,Z)にあるセルの第ｎ番目の４次元格子点に存在する全粒子数を２進数で表示したとき、その第１ビットの値がbitnum[1][Z][Y][X]の第ｎビットに、第２ビットがbitnum[2][Z][Y][X]の第ｎビットに、・・・、第６ビットがbitnum[6][Z][Y][X]の第ｎビットに表示されることになる。また同様に、位置(X,Y,Z)にあるセルの第ｎ番目の４次元格子点に存在する赤粒子数を２進数で表示したとき、その第１ビットの値がbitnum[1][Z][Y][X]の第ｎ+32ビットに、第２ビットがbitnum[2][Z][Y][X]の第ｎ+32ビットに、・・・、第６ビットがbitnum[6][Z][Y][X]の第ｎ+32ビットに表示されることになる。

また、粒子数カウントの大小比較は、カウント数の２進法表示の各桁が１格子点１ビット幅の配列で得られているので、第６ビット側から順番に比較していけばよい。まず、bitnum[6][Z][Y][X]の排他的論理和をとり、これが「1」になるビットについては、そのビットが「1」になっている格子点の方に数多くの粒子が存在している。排他的論理和が「0」になるビットについては、第５ビットの比較を行う。すなわち、bitnum[5][Z][Y][X]の排他的論理和をとり、これが「1」になるビットについては、そのビットが「1」になっている格子点の方に数多くの粒子が存在している。排他的論理和が「0」になるビットについては、第４ビットの比較を行う、と続けていく。

以上のことから、“相分離”を引き起こす計算過程は、“１格子点１ビット幅演算”ですべて実現できることがわかる。

4.5　相分離シミュレーションの実行内容

“相分離”のシミュレーションについても、“コンパクトなものづくり設計用計算機”の規模を想定するという意味で、東北大学サイバーサイエンスセンターにあるSX-ACEの32ノード(32CPU=128コア)を用い、128mpiによる並列計算を実行した。

流体シミュレーションを行う空間中には、３次元格子点を

X 方向に XX×S×2＝32×4×2＝256 個
Y 方向に YY×Nyranks×S×2＝2×16×4×2＝256 個
Z 方向に ZZ×Nzranks×S×2＝2×8×4×2＝128 個
配置し、総格子点数は、256×256×128＝8388608≒800 万個とした。

スパコンによるシミュレーション計算では、これら全ての格子点上で、３種類のエネルギー(0,1,2)をもちうる赤色か青色の仮想粒子が、互いに衝突と並進をくりかえしながら移動していく。また、境界条件は、X 方向、Y 方向、Z 方向のすべてを周期的境界条件とした。初期条件は、最初の時刻ステップ 0 で、個々の４次元格子点に赤粒子が存在する確率と青粒子が存在する確率は同じで 25%とし、速度の向きをランダムにとって静止流体を表現した。なお、重力等は印加しない。

また、可視化については、公開アプリケーションソフトウェアである“ParaView”（ParaView 5.0.0）を用いることとした。具体的には、相分離の時間発展計算が進行していく過程で、仮想粒子の粒子数密度がどのように変化していくか？を知るため、計算の時刻ステップがある一定の数進むたびに、そのときのスナップショット画像を、ParaView 可視化用の vtk ファイルフォーマットで作成し、合計 24 画面のファイル出力を実行した。

4.6　“相分離”シミュレーションの評価

SX-ACE の 32CPU による 128mpi 並列計算により、約 800 万格子点規模の計算を約 5900 ステップ行い、233 秒間を要した。ベクトル化率は、99.3%であり、全体としては、二相流の流体解析に対しても“１格子点１ビット幅演算”を適用できることを確認できた。

シミュレーションの結果得られた赤粒子密度の時間発展動画を**図４**に示す。時刻ステップが進むとともに、赤粒子どうしの引力によって静止流体中に発生した“赤粒子の高密度領域”が次第に大きくなっていく様子が見てとれる。なお、実際のシミュレーション計算では、24 画面の動画スナップショットを得ているが、変化がそれほど急速ではないので、**図４**のスナップショット画像の掲載では、24 画面のうち、第 1,5,9,23 番目に得られた４つの画像データのみを示した。

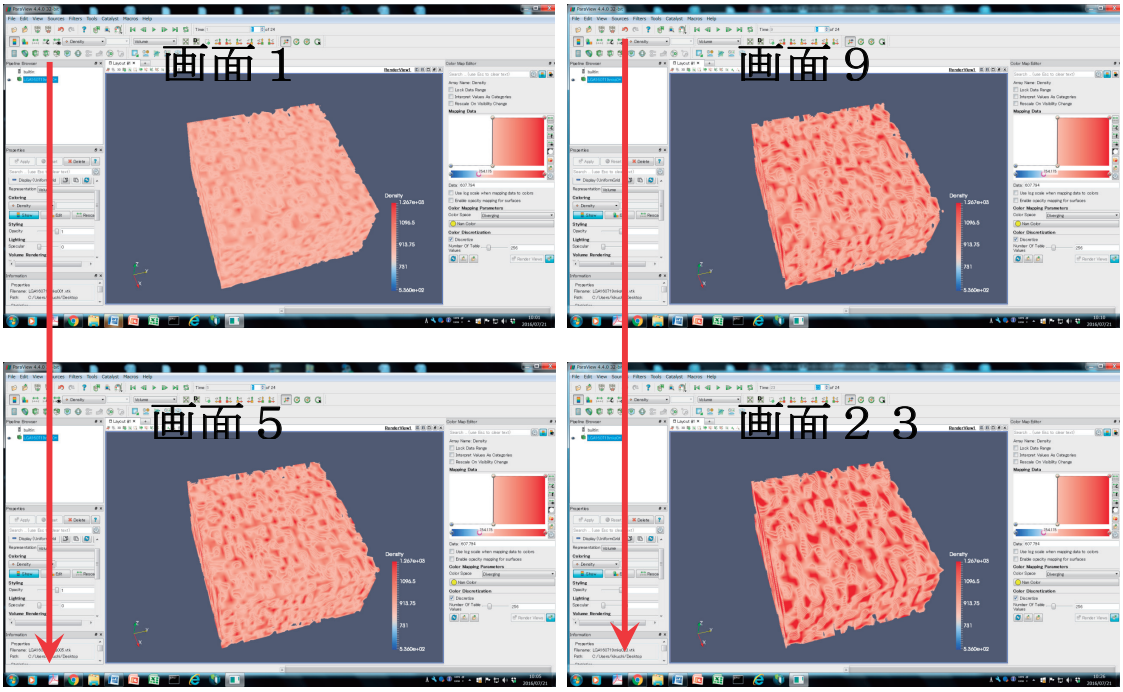

図４．　ParaView による相分離シミュレーションの連続可視化の結果（４画面を抜粋）

なお、格子ガス法では、２相の界面を境界として捉えることはせず、特定の色がついた仮想粒子の粒子数密度が急激に変化する部分として捉える。これは、界面を追跡するとか、界面での境界条件を適用するとかいった特別な取り扱いを必要としない点で大きなメリットとなる。また、ParaView の機能を利用すれば、粒子数密度に比例した着色も、あるいは、あるしきい値以上とそれ未満で極端に濃淡の異なる非線形な着色も選択することができる。可視化画像を見ながらリアルタイムでこのような着色方法を微調整することで、可視化画像中の２相界面構造を詳細に観察・把握することもできた。

ただし、**4.4** 節でも述べたとおり、ここで示した相分離シミュレーションの計算は、物理的に不適切な部分もある非常に簡略化された計算モデルを使用しているため、その結果から実現象について何かの物理的考察をできるレベルのものではない。

５．おわりに

最後に、“多速さ格子ガス法”による流体解析について、“超並列格子点演算”と“連続動画可視化”の実現性を総括する。以下のことが言えると思われる。

（１）超並列格子点演算

①1000 万格子点規模の単相流体挙動解析については、コンパクトな計算機（例：SX-Aurora TSUBASA の数 10 ノード）用いても、実時間に近いシミュレーションを実現できそうなレベルにある。

②二相流のシミュレーションについては、“相分離”を引きおこす計算過程に“１格子点１ビット幅演算”を適用することが可能である。このため、超並列計算による高速化を期待できるが、実時間シミュレーションの実現性については、実現象の種類によって異なると思われる。

（２）連続動画可視化

連続可視化については、例えば、オープンソフトウェアである ParaView を用いて、その機能（例：非線形な着色機能）を活用することにより、かなり実用的な可視化を行うことができる。

謝辞

本稿で述べた共同研究の中間報告は、これまで長期にわたり東北大学サイバーサイエンスセンターのスーパーコンピュータ（特に、SX シリーズのベクトルコンピュータ）を利用することによってはじめて得られたものである。

利用にあたっては同センター関係各位のご親切なご指導とご協力をいただき、心から深く感謝する次第である。

また、今後ますます向上するスパコン性能をすばやく産業展開していくためには、現時点で将来のスパコン利用における新しい可能性を先取りし、関連する技術をしっかり確立しておくことが重要である。この意味で、サイバーサイエンスセンターによる先進的なスパコン利用環境の提供は極めて重要であり、本研究においても、将来のスパコン環境を先取りしたコード開発を試行錯誤できることの有難さを実感している。今後とも、同センターの有意義な活動を発展させつつ継続して頂きたい。

参考文献

[1] Uriel Frisch, Dominique d'Humières, Brosl Hasslacher, Pierre Lallemand, Yves Pomeau, Jean-Pierre Rivet, “Lattice Gas Hydrodynamics in Two and Three Dimensions”, Complex Systems, 1 (1987), pp. 649-707, 1987

[2] Chistopher M. Teixeira, “Continuum Limit of Lattice Gas Fluid Dynamics”, MIT, 1993

[3]松岡, 菊池, “多速さ格子ガス法実用化展開への手がかり”, pp. 1-15, SENAC Vol. 49 No. 4 (2016-10)