計算工学講演会論文集 Vol. 28 (2023 年 5 月)
計算工学会

粒子法におけるポアソンソルバのマルチグリッド法を用いたGPU高速計算

Fast GPU computation of the Poisson solver in particle method using multigrid method

忠地涼汰 [1)] Kim Taehwan[1)] 龍野智哉 [1)]
Ryota Tadachi, Taehwan Kim and Tomoya Tatsuno

[1)]電気通信大学大学院情報理工学研究科 情報・ネットワーク工学専攻 (〒 182-8585 東京都調布市調布ヶ丘 1-5-1)

Fast GPU computation of the Poisson solver in the MPS method is investigated using the multigrid method. We constructed a bucket-based multigrid which determines the coarse grid from the particle location. In the conversion between fine and coarse grids, a scale corrected cycle is invoked to accelerate convergence. We also examined mixed-precision and sparse matrix-vector multiplication optimized for GPU calculations. For the benchmark problems of the Poisson equation and dam-break simulations in 2D and 3D, we have confirmed the significant speedup of the proposed method.

Key Words : *MPS method, Pressure Poisson equation, Bucket-based multigrid, Parallel computing, GPU*

1. はじめに

連続体の数値解析手法として、移動する粒子を計算点として用いる粒子法がある。粒子法は格子法に比べて大変形を柔軟に取り扱うことができ、自由表面をともなうような複雑な流れも特別なスキームなしに表現することができる。非圧縮性流体を扱う MPS 法 [1,2] や ISPH 法 [3] では圧力のポアソン方程式を解く必要があり、大規模な疎行列線形連立方程式の求解を必要とする。現在は高速なポアソンソルバとしてマルチグリッド法 [4] が注目を集めており、粒子法へ適用した事例も存在する [5,6]。一方、GPU を用いた粒子法高速計算の研究 [7,8] も行われているが、ポアソンソルバには並列計算が容易な対角スケーリング付き CG 法が用いられることが多い。格子法では赤黒順序付けを用いてマルチグリッド法を並列化することができる [9] が、粒子法では多色順序付けを GPU 上で効率的に適用することが困難であり、CPU 上での逐次計算に留まっている。

本研究では MPS 法におけるポアソンソルバに対して並列計算可能なマルチグリッド法を提案する。さらに粒子法に適した高速化手法を取り入れ、GPU への最適化を行う。

2. MPS 法

ここでは非圧縮性流体を扱う MPS 法 [1]~[3] について説明する。支配方程式はナビエストークス方程式と非圧縮条件

$$\frac{D\boldsymbol{u}}{Dt} = -\frac{1}{\rho_0}\nabla p + \nu\nabla^2\boldsymbol{u} + \boldsymbol{g} \tag{1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \tag{2}$$

である。ここで、$\frac{D}{Dt}$ はラグランジュ微分、$\boldsymbol{u}$ は速度、t は時間、ρ_0 は密度定数、p は圧力、ν は動粘性係数、$\boldsymbol{g}$ は重力加速度である。MPS 法では次式のように粘性項と重力項を陽的、圧力勾配項を陰的に求める。

$$\frac{\boldsymbol{u}_i^{k+1}-\boldsymbol{u}_i^k}{\Delta t} = -\frac{1}{\rho_0}\langle\nabla P\rangle_i^{k+1} + \nu\langle\nabla^2\boldsymbol{u}\rangle_i^k + \boldsymbol{g}^k \tag{3}$$

ここで、$\langle\,\rangle$ は粒子間相互作用モデル、上付き添字 k は時間ステップ、下付き添字 i は粒子番号を表す。次の圧力に関するポアソン方程式を解くことで p^{k+1} を求める。

$$-\langle\nabla^2 p\rangle_i^{k+1} = \frac{\rho_0}{\Delta t^2}\frac{n_i^* - n^0}{n^0} \tag{4}$$

ここで、n は粒子数密度、n^0 は基準粒子数密度、上付き添字 $*$ は仮時刻を示す。MPS 法では粒子間相互作用の影響半径を r_e として、次の重み関数を用いる。

$$w(r) = \begin{cases} \frac{r_e}{r} - 1 & (0 < r < r_e) \\ 0 & (r_e \le r) \end{cases} \tag{5}$$

また、粒子数密度を次式で求める。

$$n_i = \sum_{j\neq i} w(r_{ij}) \tag{6}$$

ここで、r_{ij} は粒子 i と j の粒子間距離である。ラプラシアンの粒子間相互作用モデルは次式である。

$$\langle\nabla^2\phi\rangle_i = \frac{2d}{\lambda^0 n^0}\sum_{j\neq i}(\phi_{ij})w(r_{ij}) \tag{7}$$

$$\lambda^0 = \frac{1}{n^0}\sum_{j\neq i}(r_{ij}^0)^2 w(r_{ij}) \tag{8}$$

ここで、$\phi_{ij} = \phi_i - \phi_j$、$d$ は次元数、上付き添字 0 は基準初期位置を表す。

3. 粒子法におけるマルチグリッド法

(1) マルチグリッド法

本節では粒子法におけるポアソン方程式にマルチグリッド (MG) 法 [10,11] を適用する方法について述べる。粒子法におけるポアソン方程式 (4) を離散化した線形連立方程式

$$Az = \boldsymbol{b} \tag{9}$$

を考える。ここで、A は ∇^2 を離散化した係数行列、$\boldsymbol{z}$ は方程式の解、$\boldsymbol{b}$ は式 (4) の右辺に対応するソース項である。

MG 法はより粗いグリッド上で離散方程式を生成し、細かいグリッド上での残差を粗いグリッド上の方程式で解く。この操作をグリッド数が十分小さくなるまで再帰的に実行することで低周波数成分の残差を効率的に減衰させることができる。MG 法の 1 反復の大まかな流れは次である。

1. **緩和 (pre-smoothing)** 定常反復法などによるスムージングを数回適用する
2. **制限 (restriction)** 残差を粗いグリッドへ落とし込む
3. **コースグリッド修正** 粗いグリッド上で近似解を求める
4. **補間 (prolongation)** 近似解を細かいグリッドへ補間する
5. **緩和 (post-smoothing)** スムージングを数回適用する

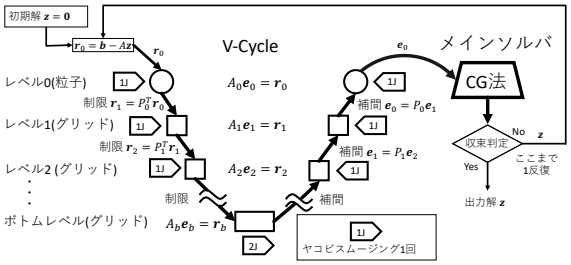
スムージングにはヤコビ法やガウスザイデル (GS) 法 [10]、並列計算では赤黒順序付け SOR 法 [9] などが使用される。

細かいグリッド数を n_f、粗いグリッド数を n_c とすると、補間には $n_f \times n_c$ の補間行列 P が用いられ、制限には P を転置した制限行列 P^{T} を用いる。粗いグリッド上の係数行列 A_c は細かいグリッド上の係数行列 A_f と補間行列からガラーキンコースグリッド近似 [10] を用いて次で求める。

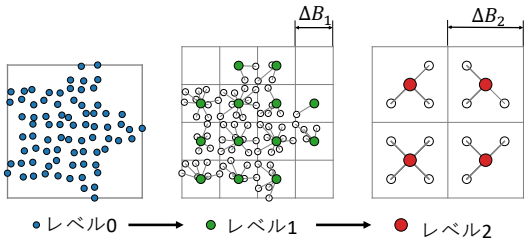
$$A_c = P^{\mathrm{T}} A_f P \tag{10}$$

本研究で使用する MG 法の概要を図-1 に示す。MG 法の深さをレベル ℓ と呼び、粒子に関する方程式 (9) はレベル 0、以降グリッドが粗くなるほどレベル数は大きくなる。最も深いレベルをボトムレベルと呼ぶ。MG 法の 1 反復は cycle と呼ばれ再帰関数で書かれる。再帰部分であるコースグリッド修正を何度行うかで cycle の形が決まり、図-1 のように 1 度行うものを V-cycle、2 度行うものを W-cycle と呼ぶ。cycle に関する詳細な説明については [5,6] を参照していただきたい。

本研究では並列計算のために、全てのスムーザにヤコビスムーザを採用する。ボトムレベルでも数回のスムージングを適用するだけとする。スムージングの回数はプレスムージング 1 回、ポストスムージング 1 回、ボトムレベルのスムージング 2 回とした。$n_c \leq 1024$ となったレベルをボトムレベルとして再帰を終了する。図 1 のように MG 法は前処理 [6,11] として使用し、メインソルバには共役勾配 (CG) 法を使用する MGCG 法を採用する。



図–1 MGCG 法 (V-cycle) の概要図



図–2 BMG 法のイメージ

(2) バケットに基づくマルチグリッド法

粒子法にはグリッドが存在しないため、前節で述べた粗いグリッドをどのように定義するかが問題となる。係数行列のみから代数的に粗いグリッドを決定する代数的 MG 法 [5] と、粒子位置から空間を格子状に分割したバケットを用いて粗いグリッドを決定するバケットに基づく MG(BMG) 法 [6] がある。本研究では並列化が容易である BMG 法を採用する。

BMG 法とは、図-2 で示すように計算領域を格子状のバケットに区切り、同じバケットに属している粒子またはセルを 1 つのセルにすることで粗いグリッドを決定する方法である。粒子法の場合は図-2 のように粒子位置から粒子を適当なサイズのバケットで区切り、レベルが 1 つ深くなるにつれてバケット幅を 2 倍にする。細かいグリッド上の要素 i が粗いグリッド上の要素 k_i に集約されるとすると、補間行列 P は次のようになる。

$$P_{j,i} = \begin{cases} 1 & (j = k_i) \\ 0 & (\text{otherwise}) \end{cases} \quad (1 \leq j \leq n_c, 1 \leq i \leq n_f) \tag{11}$$

(3) バケット幅の制約

レベル 1 の要素を決定するために使用するバケットの幅 ΔB_1 を影響半径 r_e 以上にすることで、隣接する要素にのみ係数行列の非零要素が発生し、非零要素数を抑えることができる。グリッド幅を 2 倍以上大きくすると収束性が若干落ちるが、全体としては高速であり、A_c の計算も容易であるため本研究ではレベル 1 でのバケット幅を影響半径とする。

(4) スケール補正 BMG 法

ポアソン方程式において BMG 法のガラーキンコースグリッド近似から計算された粗いグリッド上での方程式

$$A_c \boldsymbol{e}_c = \boldsymbol{r}_c \tag{12}$$

から導かれる近似解は以下で述べるようにスケールが小さくなる。ここで、$\boldsymbol{r}$ は残差ベクトル、$\boldsymbol{e}$ は修正解ベクトルである。松永ら [5] は K-cycle という自動でスケールを調整する方法を用いてこの問題を回避しているが、K-cycle は非線形前処理であったり並列計算が苦手とする内積計算が多く、GPU に最適化することが難しい。本研究ではよりシンプルで線形前処理であるスケール補正 BMG 法を提案する。

はじめにスケールについて、簡単のために 2 次元ポアソン方程式の差分法で説明する。細かいグリッド上の方程式を

$$A_f \boldsymbol{e}_f = \boldsymbol{r}_f \tag{13}$$

として、グリッド幅を h とした中心差分を用いると領域内部における A_f の i 行目の要素 $A_{f,i}$ は

$$A_{f,i} = \frac{1}{h^2}(\dots, 1, \dots, 1, -4, 1, \dots, 1, \dots) \tag{14}$$

となる。図-2 で示されるように 4 つのグリッドを 1 つにまとめるような補間行列 P を用いると、代数的に求めた粗いグリッド上の方程式 (12) は次のようになる。

$$P^\top A_f P \boldsymbol{e}_c = P^\top \boldsymbol{r}_f \tag{15}$$

ここで、$A_c = P^\top A_f P$ を計算すると、

$$A_{c,i} = \frac{1}{h^2}(\dots, 2, \dots, 2, -8, 2, \dots, 2, \dots) \tag{16}$$

となるが、幾何学的に求めた係数行列 $\bar{A}_c$ はグリッド幅が $2h$ なので、

$$\bar{A}_{c,i} = \frac{1}{(2h)^2}(\dots, 1, \dots, 1, -4, 1, \dots, 1, \dots) \tag{17}$$

となる。よって、$A_c = 8\bar{A}_c$ の関係があり、$P^\top \boldsymbol{r}_f$ によって右辺は 4 倍されているので、式 (13) の解と式 (15) を解くことで導かれた近似解の間には $\boldsymbol{e}_f \approx 2P\boldsymbol{e}_c$ の関係がある。これは 3 次元でも同様に 2 倍の関係が発生する。

そこで、係数行列 A_c を

$$A_c = \frac{1}{2} P^\top A_f P \tag{18}$$

と定義することでスケールが幾何学的な方程式と一致し、収束が加速することが期待できる。しかし、数値実験的にレベル 01 間の粒子からグリッドへの変換部分でのみ最適なスケール補正値にばらつきがあり、場合によっては収束性が著しく悪化することがあった。よって、レベル 01 間ではスケール補正を行わず、それ以降でのみスケール補正を行うこととする。

4. GPU での並列計算

本研究で提案する BMG 法は GPU 上で CUDA[12] を用いて並列に実行する。特に効率的な並列計算が難しい、BMG 法のセットアップに関する並列アルゴリズムと、GPU の特性を活かした MG 法の最適化、高速化について説明する。

なお、MG 法や CG 法で頻出するベクトル和、内積計算については cuBLAS[13] を使用し、疎行列ベクトル積 (SpMV) は本節 (4) で説明する粒子法に最適化した自作の関数を使用する。ベクトルの最大、最小値計算と prefix sum には Thrust[14] を使用した。

(1) BMG 法におけるセットアップの並列化

a) 補間行列 P の並列計算

補間行列 P を生成する並列アルゴリズムについて 2 次元の場合を例に説明する。式 (11) のように P は $n_f \times n_c$ の行列であり、各行に 1 つの非零要素があり、値は 1 である。よって、長さ n_f の 1 次元配列に非零要素の列番号を格納する。初めにレベル 0 において各粒子 i が属するバケット番号を $Q_{0,i}$ にセットする。

$$Q_{0,i} = \left(\left\lfloor \frac{x_i - x_{\min}}{\Delta B_0} \right\rfloor, \left\lfloor \frac{y_i - y_{\min}}{\Delta B_0} \right\rfloor \right) \quad (0 \le i < N) \tag{19}$$

(x_i, y_i) は粒子 i の座標、$\lfloor\ \rfloor$ は floor 関数、ΔB_0 はレベル 0 における仮想的なバケット幅、N は粒子数である。$x_{\min}, y_{\min}$ は各座標の粒子が存在する最小値である。

細かいグリッド上の 2 次元バケット番号 Q_f について各次元の番号を 2 で切り捨て除算することで、制限先である粗いグリッド上の 2 次元バケット番号 Q_c を求めることができる。Q_c に prefix sum を用いて 1 次元の番号順序付けを行い、補間行列 P を生成する。これらの操作はすべて並列に計算することができる。

b) $P^\top AP$ の並列計算

補間行列 P を用いて $A_c = P^\top A_f P$ を並列計算するアルゴリズムについて説明する。$\Delta B_0 \ge 2r_e$ とすれば、レベル 1 以降の係数行列 A_c の各行の非零要素は 2 次元、3 次元の場合でそれぞれ高々 $9, 27$ 個である。$P^\top A_f P$ の並列計算アルゴリズム 1 を示す。粗いグリッドの係数行列 A_c は CSR 形式で生成することにし、値の配列を $A_{c,\mathrm{val}}$、列番号の配列を $A_{c,\mathrm{col}}$ としている。行方向のセル i と列方向のセル j の粗いグリッド上のバケット番号の差を計算することで、値を足し合わせる A_c の対応する列番号を高速に計算することができる。$A_{c,\mathrm{col}}$ が -1 である要素は係数が存在しないので除外する必要がある。

Algorithm 1 $P^\top AP$ の並列計算アルゴリズム (2 次元)

Input: A_f, P, Q_c
Output: A_c

1: $A_{c,\mathrm{val}}[i] \leftarrow 0 \quad (0 \le i < 9n_c)$
2: $A_{c,\mathrm{col}}[i] \leftarrow -1 \quad (0 \le i < 9n_c)$
3: **for** $i = 0$ to $n_f - 1$ **do in parallel**
4: $\quad i_c \leftarrow P[i]$
5: $\quad (i_x, i_y) \leftarrow Q_c[i]$
6: $\quad$ **for** $j = 0$ to $n_f - 1$ **if** $A_{f,ij} \neq 0$ **do**
7: $\quad\quad j_c \leftarrow P[j]$
8: $\quad\quad (j_x, j_y) \leftarrow Q_c[j]$
9: $\quad\quad (dx, dy) \leftarrow (j_x - i_x + 1, j_y - i_y + 1)$
10: $\quad\quad k \leftarrow 3dy + dx$
11: $\quad\quad A_{c,\mathrm{col}}[9i + k] \leftarrow j_c$
12: $\quad\quad A_{c,\mathrm{val}}[9i+k] \leftarrow A_{c,\mathrm{val}}[9i+k] + A_{f,ij}$ (atomicAdd)
13: $\quad$ **end for**
14: **end for**

(2) 前処理における混合精度計算

GPU は単精度演算を得意としており、倍精度演算では性能が出ないことが多い。Shioya ら [15] は ILU 分解前処理付き BiCGSTAB 法において、前処理を単精度、メインソルバを倍精度で計算しても数値実験から収束性の悪化には繋がらないとしている。

本研究でも前処理である MG 法は全て単精度で計算し、メインソルバの CG 法のみ倍精度で計算する混合精度を採用した。MGCG 法では半分以上を前処理計算に費やすため大幅な高速化が期待できる。

(3) 粒子法における SpMV の高速化

CG 法や MG 法では SpMV の計算が大半を占める。SpMV を GPU に実装する場合は 1 スレッドで 1 行を計算することが多いが、1 行を複数スレッドによるリダクション処理によって計算する方法がある。CSR 形式を用いた SpMV では係数行列の値と列番号のメモリアクセスがボトルネックとなるため、リダクション処理を用いることで連続的なメモリアクセスを実現できる。本研究では 1 行を 8 スレッドで計算し、シェアードメモリを用いて 8 スレッドの総和計算を行う。

5. ポアソンソルバの数値実験

(1) 計算条件

次の 2 次元と 3 次元におけるポアソン方程式を用いて数値実験を行った。

$$\Omega = \left\{ \boldsymbol{x} \in \mathbb{R}^d \,|\, 0 \le x_j \le 1, j \in [1, d] \right\} \tag{20}$$

$$-\nabla^2 \phi = \prod_{j=1}^{d} \sin 2\pi x_j \quad \text{in} \quad \Omega \tag{21}$$

$$\phi = 0 \quad \text{on} \quad \partial\Omega \tag{22}$$

$$\phi_{\text{exact}} = \frac{1}{4\pi^2 d} \prod_{j=1}^{d} \sin 2\pi x_j \tag{23}$$

重み関数とラプラシアンモデルには式 (5) と式 (7) を使用した。影響半径は基準粒子間距離を l_0 として 2 次元問題では $r_e = 3.1 l_0$、3 次元問題では $r_e = 2.1 l_0$ とした。Ω 内部の粒子を流体粒子とし、規則正しく格子点上に並べた位置 $\boldsymbol{x}_i^{\text{grid}}$ に次の擾乱を加えた $\boldsymbol{x}_i^{\text{FLD}}$ を粒子 i の位置とする。

$$\boldsymbol{x}_i^{\text{FLD}} = \boldsymbol{x}_i^{\text{grid}} + 0.3 l_0 \boldsymbol{\delta}_i \tag{24}$$

ここで、$\boldsymbol{\delta}_i \in [-1, 1]^d$ は擬似一様乱数ベクトルである。境界における粒子数密度を保つために、Ω 外部に 2 層の境界粒子を格子点上に規則正しく配置し、$\phi_i^{\text{BC}} = \phi_{\text{exact}}$ とするディリクレ境界条件を与える。線形連立方程式では流体粒子のみを解き、境界粒子は式 (9) における係数行列の対角成分とソース項に表れる。ここで、係数行列は対称行列となる。メインソルバの収束判定は ε と相対残差ノルムを用いて

$$\frac{\|\boldsymbol{b} - A\boldsymbol{z}\|}{\|\boldsymbol{b}\|} \le \varepsilon \tag{25}$$

とした。ここで、$\|\|$ は ℓ^2 ノルムを表す。反復の初期値は $\boldsymbol{z}_0 = \boldsymbol{0}$ とする。本実験では CPU に Intel Core i7-11700 メモリ 16GB、GPU に NVIDIA GeForce RTX3060ti 8GB を使用した。

(2) 混合精度の実験結果

混合精度の収束性について、2 次元問題を用いて数値実験を行う。K-cycle とスケール補正 W-cycle (S-W-cycle と表記する) を用いて、それぞれ倍精度と混合精度で収束性と計算時間を比較した。粒子数を $N = 1024^2$ とし、収束判定は $\varepsilon = 10^{-14}$ とした。

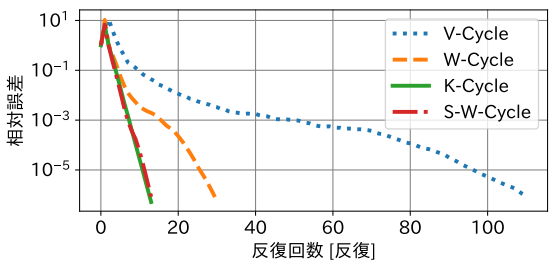
K-cycle の結果では、倍精度を用いた場合は相対誤差 10^{-14} まで収束したが、混合精度を用いた場合は相対誤差が単精度を超えた 10^{-8} あたりで数値発散した。S-W-cycle の結果では、倍精度、混合精度どちらも同じ反復回数で相対誤差 10^{-14} まで収束した。S-W-cycle の計算時間としては、倍精度では 2.15 秒であるのに対して混合精度では 1.72 秒であり、約 1.25 倍の高速化が得られた。

よって、V-cycle や W-cycle などの線形な MG 前処理では混合精度を適用することで、収束性を保ったまま計算時間を減少させることができる。

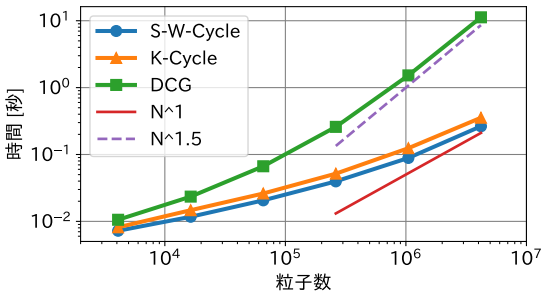
(3) 各種 cycle の比較実験結果

cycle の種類とスケール補正による収束性と計算時間を比較する。初めに 2 次元で粒子数 $N = 1024^2$ とした場合の各種 cycle の収束曲線を比較する。ここでは V-cycle、W-cycle、K-cycle、スケール補正を行う S-W-cycle の 4 つを測定した。結果を図-3 に示すが、ここでの反復回数は図-1 における CG 法の適用回数に等しい。特筆すべきは、W-cycle にスケール補正を行うことで収束までの反復回数が K-cycle と同程度にまで改善されることである。

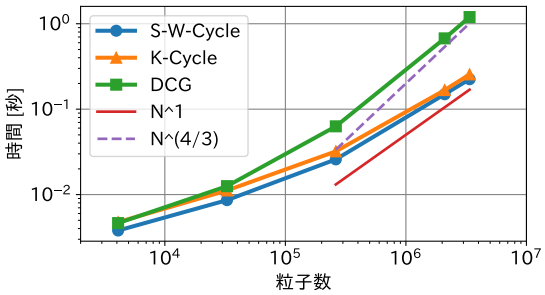
次に粒子数を変化させた場合の、S-W-cycle と K-cycle、参考のために対角スケーリング付き CG (DCG) 法の計算時間を測定した。収束判定は $\varepsilon = 10^{-6}$ とし、MGCG 法については混合精度を適用した。それぞれ 2 次元と 3 次元の計算時間を図-4、図-5 に示す。図-4、図-5 より、MGCG 法の S-W-cycle と K-cycle は粒子数に対して高々線形時間で解いており、その中でも S-W-cycle が最も高速である。さらに S-W-cycle は 2 次元問題で粒子数が 100 万個の場合でも 0.1 秒以下で解いており、DCG 法と比較すると 10 倍以上高速である。



図–3 2 次元問題の収束曲線



図–4 2 次元問題における各種 cycle の計算時間



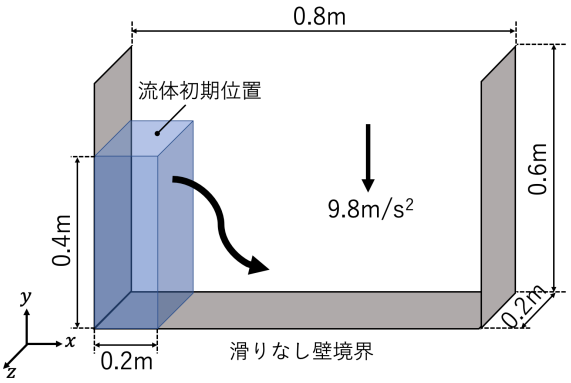
図–5 3 次元問題における各種 cycle の計算時間

6. 流体シミュレーションへの適用

(1) 実験条件

提案手法をポアソンソルバに使用した MPS 法を用いて、2 次元、3 次元のダムブレイク問題のシミュレーションを行い、実問題に対する有効性を検証した。実験条件は図-6 に示すように、壁で囲まれた領域の一部を流体で満たし、時刻 0 秒から流体が重力によって流れ込む挙動をシミュレートする。支配方程式はナビエストークス方程式 (1) と非圧縮条件 (2) を用いる。物理パラメータは重力加速度 $\boldsymbol{g} = (0, -9.8, 0)^{\top} m/s^2$、密度 $\rho_0 = 1000 kg/m^3$、動粘性係数 $\nu = 10^{-6} m^2/s$ とした。式 (4) の右辺に安定化のための緩和係数 0.2 をかけたものを圧力ポアソン方程式とする [2]。ポアソンソルバの収束条件は $\varepsilon = 10^{-12}$ とし、影響半径と MG 法の各パラメータ、計算機は 5(1) 節の実験条件と同一とした。壁境界には壁粒子とダミー粒子を 2 層ずつ配置した。MPS 法における近傍粒子探索には基数ソートを用いた Uniform Grid[8] を使用した。また、近傍粒子リストを生成することで近傍粒子候補を最小限にし、粒子間相互作用を高速に計算している。近傍粒子探索、近傍粒子リスト生成、粒子間相互作用などは全てを並列化し、GPU 計算に十分な最適化を行なっている。

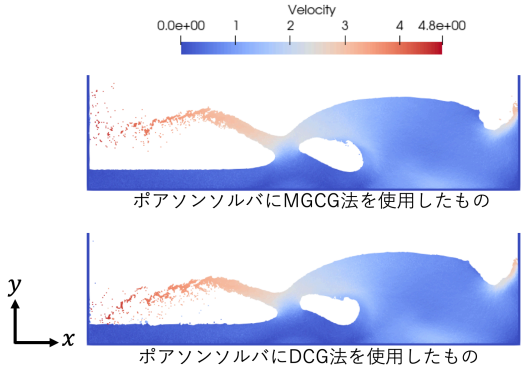
2 次元、3 次元について粒子数の異なる 2 つのケースでシミュレーションを行う (表-1)。ポアソンソルバの解法にはスケール補正 W-cycle BMGCG 法 (MGCG 法と記述する) と DCG 法の 2 つを適用し、シミュレーション結果と計算時間の比較を行う。各ケースのパラメータを表-1 に示す。クーラン数が 0.2 以下となるような一定の時間刻み幅 Δt を設定する。ケース 2A、3A では時刻 1 秒まで、ケース 2B、3B では 0.01 秒までシミュレーションを行なった。



図–6 ダムブレイク問題の実験条件

表–1 ダムブレイク問題のパラメータ

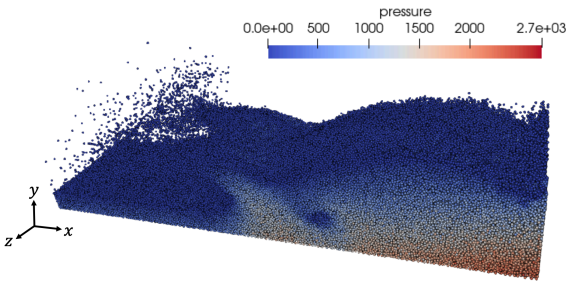
ケース	d	$l_0[m]$	流体粒子数	$\Delta t[s]$
2A	2	1×10^{-3}	80000	2.5×10^{-5}
2B		2.5×10^{-4}	1280000	6.25×10^{-6}
3A	3	4×10^{-3}	250000	1×10^{-4}
3B		2×10^{-3}	2000000	5×10^{-5}



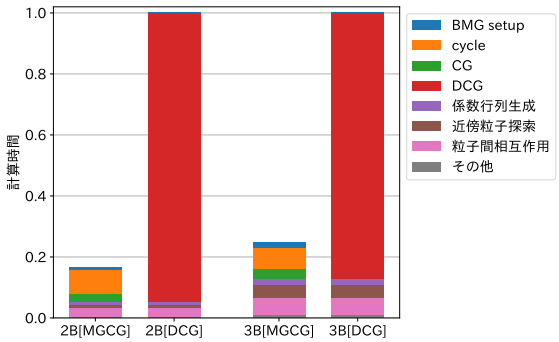
図–7 ケース 2A の時刻 1 秒における結果 (粒子の色は速度の大きさを表す)

(2) 結果

各解法を用いたケース 2A の時刻 1 秒におけるシミュレーション結果を図-7 に示す。ポアソンソルバに MGCG 法と DCG 法を用いた場合の結果は良く一致しており、本研究で提案する BMG 法をポアソンソルバに使用することによる問題は発生しないと考えられる。次に MGCG 法を用いて解析したケース 3A の時刻 1 秒におけるシミュレーション結果を図-8 に示す。図-8 では流体粒子のみを表示しており、粒子の色は圧力値を表している。滑らかな圧力場が見られ、流体の自然な挙動が見られる。 ケース 2B、3B について、時間ステップあたりの計算時間の内訳を図-9 に示す。DCG 法の時間ステップあたりの計算時間を 1 として、MGCG 法と DCG 法のそれぞれの場合についてグラフ化した。全体の計算時間として MGCG 法は DCG 法と比較して 2 次元で 5 倍、3 次元で 4 倍以上高速である。DCG 法を用いた場合は MPS 法においてポアソンソルバの計算時間が多くを占



図–8 MGCG 法を用いたケース 3A の時刻 1 秒における結果 (粒子の色は圧力値を表す)



図–9 ケース 2B、3B における計算時間の内訳

めることがわかる。一方、3 次元問題において MGCG 法を用いた場合は MPS 法全体のうちポアソンソルバが占める割合は半分程度であり、高速に処理されていることがわかる。

以上より、本研究で提案するスケール補正 W-cycle BMGCG 法は実際の流体シミュレーションに適用することが可能であり、GPU 上で従来の DCG 法よりも高速に計算できることが分かった。

7. まとめと今後の展望

本研究では粒子法におけるポアソンソルバのマルチグリッド法を用いた GPU 上での高速計算を検討した。MG 法には粒子位置から粗いレベルの要素を決定する BMG 法を採用した。CG 法の前処理として MG 法を使用し、MG 法を単精度、CG 法を倍精度で計算する混合精度を採用した。また、BMG 法のセットアップに関する並列アルゴリズムや粒子法に最適化した SpMV の実装方法について説明した。粗いグリッド空間で導かれた近似解のスケールを細かいグリッド空間に合わせるためにスケール補正 cycle を提案した。2 次元、3 次元におけるポアソン方程式の数値実験から、スケール補正 W-cycle は先行研究 [6] で最も高速であるとされてきた K-cycle と同程度の収束性を持ち、GPU 上の並列計算では K-cycle よりも高速であった。

実際の流体シミュレーションへの適用として、2 次元、3 次元のダムブレイク問題をスケール補正 BMGCG 法を用いた MPS 法によって GPU 上で解析した。その結果、DCG 法と比較して、提案する MG 法は 2 次元問題では粒子数 128 万の場合に約 5 倍、3 次元問題では粒子数 200 万の場合に約 4 倍の高速化に成功した。本研究で提案するスケール W-cycle を用いた BMG 法は粒子法に止まらず、他のメッシュフリー解析にも容易に適用することができる。

本研究では標準的な MPS 法で導かれる対称行列の疎行列線形連立方程式を対象としており、より高精度な MPS 法における非対称行列線形連立方程式への適用は今後の課題である。スケール補正 W-cycle における各種パラメータの最適な値については検討の余地がある。

参考文献

[1] Koshizuka, S. and Oka, Y.: Moving-Particle Semi-Implicit Method for Fragmentation of Incompressible Fluid. Nuclear Science and Engineering, Vol.123, pp.421-434, 1996.
[2] 越塚誠一, 柴田和也, 室谷浩平: 粒子法入門 流体シミュレーションの基礎から並列計算と可視化まで, 丸善出版, 2014.
[3] 後藤仁志: 粒子法 連続体・混相流・粒状体のための計算科学, 森北出版株式会社, 2018.
[4] Brandt, A.: Multi-Level Adaptive Solutions to Boundary-Value Problems, Mathematics of Computation, Vol.31, pp.333-390, 1977.
[5] 松永拓也, 柴田和也, 室谷浩平, 越塚 誠一: 代数的マルチグリッド法を用いた粒子法における圧力ポアソン方程式の解法, 日本計算工学会論文集, Vol.2016, p.20160012, 2016.
[6] Södersten, A., Matsunaga, T., Koshizuka, S.: Bucket-based Multigrid Preconditioner for Solving Pressure Poisson Equation Using a Particle Method. Computers and Fluids, Vol.191, p.104242, 2019.
[7] 後藤仁志, 堀智恵実, 五十里洋行, KHAYYER, A.: GPU による粒子法半陰解法アルゴリズムの高速化, 土木学会論文集 B, Vol.66-2, pp. 217-222, 2010.
[8] 佐々木卓雅: GPU および領域分割を用いた粒子法による流体シミュレーションの高速化. 修士論文, 電気通信大学, 2015.
[9] 小川慧, 青木尊之: GPU によるマルチグリッド法を用いた 2 次元非圧縮性流体解析の高速計算, 日本計算工学会論文集, Vol.2009, p.20090021, 2009.
[10] 杉原正顯, 室田一雄: 線形計算の数理, 岩波書店, 2009.
[11] 建部修見, 小柳義夫: マルチグリッド前処理付き共役勾配法の並列化, 並列処理シンポジウム論文集, Vol.1993, pp.387-394, 1993.
[12] NVIDIA: CUDA Toolkit Documentation 12.1, https://docs.nvidia.com/cuda/.
[13] NVIDIA: The API Reference Guide for cuBLAS, https://docs.nvidia.com/cuda/cublas/.
[14] NVIDIA: The API Reference Guide for Thrust, https://docs.nvidia.com/cuda/thrust/.
[15] Shioya, A., Yamamoto, Y.: Block Red-Black MILU(0) Preconditioner with Relaxation on GPU, Parallel Computing, Vol.103, p.102760, 2021.