情報処理学会論文誌　コンピューティングシステム　Vol.8　No.4　50–60 (Nov. 2015)

接触による粒子間相互作用のGPU計算での近傍探索手法

渡辺 勢也[1,a)]　青木 尊之[1,b)]　都築 怜理[1,c)]　下川辺 隆史[1,d)]

受付日 2015年4月2日, 採録日 2015年8月4日

概要：接触相互作用に基づく粒子法の 1 つである個別要素法の GPU での計算において，その実行性能とメモリ使用量は，近傍粒子探索手法の実装に大きく依存する．本論文では，近傍粒子探索手法として，連結リスト法，ハッシュ法，粒子登録法と連結リスト法のハイブリット手法，粒子登録法とハッシュ法のハイブリット手法を実装し，各手法の違いによる計算時間およびメモリ使用量の詳細な比較を行う．近傍探索手法を比較する例題として，512 個から 8,388,608 個の粒子を用いた 3 次元個別要素法によるダム崩壊シミュレーションをそれぞれの近傍探索手法で計算し，計算時間の測定を行った．粒子登録法と連結リスト法のハイブリット手法が実装した 4 つの近傍探索手法のなかで最も計算を高速に行えることが分かった．粒子登録法での近傍リスト作成時に，ハッシュ法よりも速い連結リスト法を用いたことと，粒子登録法により相互作用計算で影響半径外の粒子との無駄な接触判定を極力減らしたためである．メモリ使用量はハッシュ法が最も少なく，使用できるメモリ量が限られている場合はハッシュ法を用いればよいことも分かった．最も計算が速い粒子登録法と連結リスト法のハイブリット手法のメモリ使用量は，ハッシュ法の 1.18 倍程度であり，この組合せの優位性を確認した．どの近傍探索手法においても，粒子数が 10 万個以下の小規模な計算では計算性能が低下し，1 スレッドが 1 粒子を計算する実装方法では，GPU による計算の高速化が効果的ではないことが分かった．

キーワード：GPU，個別要素法，連結リスト法，ハッシュ法，粒子登録法

Neighbor-particle Searching Method for Particle Simulation Based on Contact Interaction Model for GPU Computing

Seiya Watanabe[1,a)]　Takayuki Aoki[1,b)]　Satori Tsuzuki[1,c)]　Takashi Shimokawabe[1,d)]

Received: April 2, 2015, Accepted: August 4, 2015

Abstract: In particle simulations based on Distinct Element Method (DEM) modeled by contact interaction forces, the computational performance on GPU strongly depends on the implementation of the neighbor-particle searching method. We study four kinds of methods: linked-list method, hash method, book-keeping method with a linked-list method, book-keeping method with a hash method. A benchmark test of 3-dimentional dam breaking problem is carried out by changing the particle number from 512 to 8,388,608 particles to examine the performances and the memory usages. The book-keeping method with a linked-list method is the fastest among the four methods, because it is possible to make the neighbor-particle list without non-contacting particles by using the book-keeping method and results in reducing the number of distance calculations. The amount of memory used in the hash method is the lowest and we choose it when the size of GPU device memory is small. However the book-keeping method with a linked-list method uses only 1.18 times the memory of the hash method. When the number of particles is lower than 100,000 particles, all the neighbor-particle searching methods show low performances and it is found that particle simulations based on DEM has advantages for large-scale problems by using GPU computing.

Keywords: GPU, Distinct Element Method, linked-list method, hash method, book-keeping method

[1] 東京工業大学
Tokyo Institute of Technology, Meguro, Tokyo 152–8550, Japan
[a)] watanabe@sim.gsic.titech.ac.jp
[b)] taoki@gsic.titech.ac.jp
[c)] tsuzuki@sim.gsic.titech.ac.jp
[d)] shimokawabe@sim.gsic.titech.ac.jp

1. 緒言

粉体シミュレーションでは，接触による粒子間相互作用に基づく粒子法である個別要素法（DEM: Distinct Element Method）[1] や不連続変形法（DDA: Discontinuous Deformation Analysis）[2] などが用いられている．実際の現象を粒子法でより正確に解析するためには多くの粒子を用いた大規模な計算が必要であるが，粒子数を増やすと計算負荷が膨大になる．すべて同じ大きさの粒子を用いた場合，接触による相互作用計算では，1 つの粒子が相互作用する粒子はせいぜい十数個であり，計算はメモリアクセスに律速される．

接触相互作用による粒子計算を高速化するには，全体の粒子から接触する十数個の粒子を効率良く探索する必要がある．全粒子数 NP に対して着目している粒子と接触しているかの判定を行う場合，接触判定の回数は NP の 2 乗のオーダとなり，粒子数が増えると接触する粒子を探索する時間の方が相互作用の力の計算時間などよりずっと長くなってしまう．効率良く近傍粒子を探索する手法として，粒子登録法 [3], [4] やセル分割法 [4], [5] がある．粒子登録法は，各粒子が自身と接触する可能性のある近傍の粒子のインデックス（粒子番号）を記憶しておき，粒子の動く距離の短いある一定ステップの間はリスト内の粒子とのみ接触判定と相互作用計算を行う手法である．セル分割法は，計算領域を一様な格子（セル）に分割し，自身の所属するセルおよび周囲のセル内に含まれる粒子に対してのみ，接触判定と相互作用計算を行う方法である．各セルに所属する粒子番号をすべて登録させる通常のセル分割法では，セル内に入りうる最大の粒子数を想定した粒子のインデックスを保持するメモリが必要となり，粒子の移動範囲が広く計算領域が大きくなるとメモリ不足を引き起こす場合がある．セル分割法でのメモリ使用量を削減するために，連結リストを用いたセル分割法 [3], [6]（以下，連結リスト法と記す）と，ハッシュを用いたセル分割法 [7]（以下，ハッシュ法と記す）が用いられている．また，粒子登録法での近傍粒子のリスト作成時に，連結リスト法やハッシュ法による近傍探索を行い，近傍リストの作成を高速化する手法が提案されている [8], [9], [10]．DEM は計算時間のかかる手法であり，このような近傍粒子探索の高速化を行ったとしても，CPU の 1 core で計算できる粒子数は数十万個程度である．

DEM のさらなる高速化および大規模化を行うために，画像処理に特化した演算加速器である GPU（Graphics Processing Unit）を利用した研究がさかんに行われている [8], [11], [12], [13], [14], [15], [16]．GPU は浮動小数点演算性能やメモリバンド幅が CPU に比べて優れており，計算時間がメモリ律速である DEM などの接触相互作用に基づく粒子計算では，GPU の広いメモリバンド幅を活かして高速化が可能である．ただし，高速にアクセスが可能な GPU のオンボードメモリ容量は数 GB と少ないため，近傍粒子探索で使用するメモリ量は抑える必要がある．

GPU 上での DEM 計算の近傍探索手法として，先行研究 [11], [13] では連結リスト法，先行研究 [12], [14], [15] ではハッシュ法，先行研究 [8] では粒子登録法とハッシュ法のハイブリット手法，先行研究 [16] では粒子登録法と通常のセル分割による方法が用いられている．このように，GPU の DEM シミュレーションでは様々な近傍探索手法が用いられており，どの近傍探索手法を実装すればよいかの定量的な比較は行われていない．本研究の目的は，GPU 計算における接触相互作用に基づく粒子計算の高速化に適した近傍探索手法を明らかにすることである．3 次元 DEM の GPU での計算に，連結リスト法と，ハッシュ法と，粒子登録法と連結リスト法のハイブリット手法と，粒子登録法とハッシュ法のハイブリット手法の 4 つの近傍探索手法を実装し，計算の実行時間およびメモリ使用量の詳細な比較を行う．

本論文では，計算の実行環境として，東京工業大学学術国際情報センターの TSUBAME2.5 を用いる．使用した GPU は NVIDIA 社の Tesla K20X であり，CUDA の Version は 6.0 である．

2. DEM（Distinct Element Method）

本論文で用いる DEM のモデルを述べる．DEM では，粉体を構成する 1 つ 1 つの粒子に作用する外力と周囲の粒子との接触力を求め，ニュートンの第二法則に基づき粒子運動を計算することで，粉体全体の挙動を解析する．球形要素の粒子による粉体のモデル化は，粒子どうしの接触判定が容易であるため多くの DEM 計算で用いられており，球形粒子を用いた DEM では以下に記すようにして，粒子間接触力や粒子に作用する合力とモーメントを計算する．

図 1 は DEM の粒子間相互作用モデルの略図である．粒子 i は接触している粒子 j と k から接触力を受け，接触していない粒子 l とは粒子間相互作用が働かない．粒子の衝突は粒子 i-j 間のように，法線方向のバネとダッシュポットでモデル化される．バネは粒子の食い込み深さに比例し

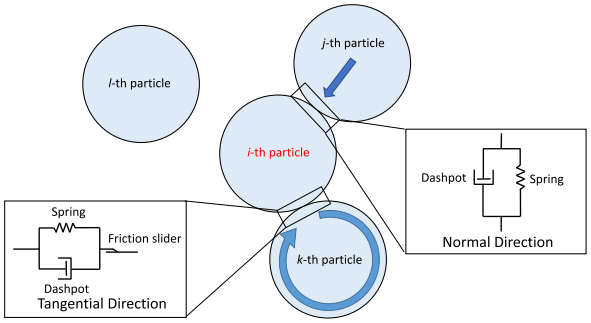

図 1　個別要素法の粒子間相互作用モデル
Fig. 1　DEM interaction model.

た反発力を粒子に与え，ダッシュポットは相対速度に比例した減衰力を与える．粒子 i-k 間のように，粒子の回転により粒子間の摩擦が作用する場合は，バネとダッシュポットに加えて，摩擦係数を考慮するための摩擦スライダが挿入されたモデルで接線方向の力を計算する．

粒子 i が粒子 j から受ける法線方向と接線方向の接触力は以下で表される．

$$\boldsymbol{F}_{ij}^{\mathrm{N}} = k^{\mathrm{N}}\boldsymbol{L}_{ij}^{\mathrm{N}} + c^{\mathrm{N}}\frac{\Delta\boldsymbol{L}_{ij}^{\mathrm{N}}}{\Delta t} \tag{1}$$

$$\boldsymbol{F}_{ij}^{\mathrm{T}} = min\left\{k^{\mathrm{T}}\boldsymbol{L}_{ij}^{\mathrm{T}} + c^{\mathrm{T}}\frac{\Delta\boldsymbol{L}_{ij}^{\mathrm{T}}}{\Delta t}, \mu\boldsymbol{F}_{ij}^{\mathrm{N}}\right\} \tag{2}$$

ここで，添字 N と T はそれぞれ法線方向と接線方向を表し，$\boldsymbol{L}$ はバネの圧縮量ベクトル，$\Delta\boldsymbol{L}$ は相対変位増分ベクトル，Δt は時間刻み，k はバネの弾性係数，c はダッシュポットにおける粘性減衰係数，μ は摩擦係数であり，min は絶対値が小さい方の値をとる関数である．粒子 i の半径を R_i，粒子 i と j の位置ベクトルを $\boldsymbol{x}_i$，$\boldsymbol{x}_j$ とすると，接線方向の力が粒子に与えるモーメントのベクトル $\boldsymbol{M}_{ij}$ は次式で表される．

$$\boldsymbol{M}_{ij} = \frac{R_i}{|\boldsymbol{x}_j - \boldsymbol{x}_i|}(\boldsymbol{x}_j - \boldsymbol{x}_i) \times \boldsymbol{F}_{ij}^{\mathrm{T}} \tag{3}$$

粉体の 1 つの粒子は一度に複数個の粒子と接触する．粒子 i と接触しているすべての粒子 j から受ける力とモーメントのベクトルの総和を計算し，粒子 i に作用する力とモーメントの合力ベクトルを求める．計算した粒子 i に作用する合力ベクトルから粒子の並進と回転の運動方程式を立てる．

$$m_i\frac{d^2\boldsymbol{x}_i}{dt^2} = \sum_{i\neq j}(\boldsymbol{F}_{ij}^{\mathrm{N}} + \boldsymbol{F}_{ij}^{\mathrm{T}}) + m_i\boldsymbol{g} \tag{4}$$

$$I_i\frac{d^2\boldsymbol{\theta}_i}{dt^2} = \sum_{i\neq j}\boldsymbol{M}_{ij} \tag{5}$$

ここで，$\boldsymbol{x}_i$ と $\boldsymbol{\theta}_i$ は粒子 i の位置および回転角ベクトル，m_i と I_i は粒子 i の質量および慣性モーメント，$\boldsymbol{g}$ は重力加速度のベクトルである．各粒子に対して並進および回転の運動方程式を数値積分して，1 タイムステップ後の粒子の位置，速度，回転角および角速度を求める．粒子間接触力の計算と運動方程式の時間積分を繰り返し行い，所望の時間まで時間発展させる．

3. 近傍探索手法

本章ではすでに提案されている粒子法における近傍探索手法について簡単に述べる．

3.1 セル分割法

DEM などの接触相互作用に基づく粒子法では，接触している粒子との相互作用を毎ステップ計算する．その時刻で接触する可能性がない粒子との距離計算および相互作用計算は行う必要がない．そのため，接触している粒子を効率良く探索すれば，近傍粒子探索にかかる時間を大幅に削減でき，計算時間の短縮が可能となる．

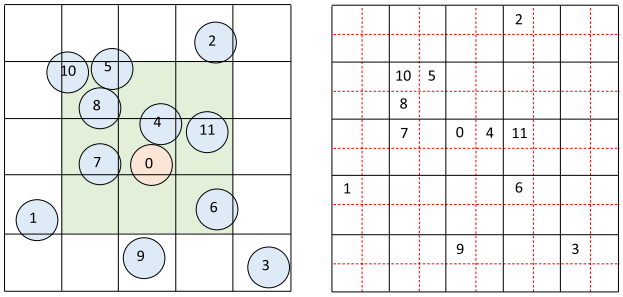

図 2　セル分割による近傍探索
Fig. 2　Neighboring cell list.

セル分割法では，**図 2** の左図のように計算領域を一様な格子で分割し，ある着目している粒子において，その粒子自身が所属するセルおよび周囲のセルに所属する粒子とのみ距離の計算を行うことで効率的に近傍粒子を探索する [4], [5]．各粒子がどのセルに含まれるかを判定し，図 2 の右図のように各セルが所属する粒子を記憶する．セルの大きさを相互作用半径よりも大きく設定することで，着目している粒子と現在のタイムステップで相互作用する可能性のある粒子は，自身が所属しているセルと隣接しているセル（2 次元計算で 8 個，3 次元計算で 26 個）に含まれる粒子に限定される．

DEM では，全粒子の大きさが同じ場合は，先行研究 [5] などではセルの大きさを粒子直径と等しく設定している．簡単のために全粒子の大きさが同じでセルの大きさを粒子直径と等しくした場合を仮定すると，1 つのセルに含まれる最大の粒子数は 2 次元計算で 4 個，3 次元計算で 8 個である．図 2 の右図のように各セルはセル内に粒子が存在していない場合でも，粒子のインデックスを格納するメモリ領域を持っておく必要がある．そのため，セル分割法を行うためには，2 次元計算で格子数の 4 倍，3 次元計算で 8 倍の数の粒子番号を登録しておくメモリが必要となり，計算領域を広くとると DEM 計算で用いる粒子数と無関係にメモリが枯渇する可能性がある．大きさの異なる粒子を扱う場合は，1 つのセルに対してさらに多くの粒子が入る可能性があり，各セルが確保しておくメモリ量が深刻な問題となる．

セル分割法でのメモリ使用量を抑える方法として，3.2 節，3.3 節で示す，連結リスト法 [3], [6] とハッシュ法 [7] がある．

3.2 連結リスト法

連結リスト法による近傍探索手法の概念図を**図 3** に示す．各粒子は同一セル内の他の粒子のインデックスを 1 つ記憶することで 1 方向の連結リストを作成する．最後尾の粒子には，リストが終了することを表す −1 を記憶させる．

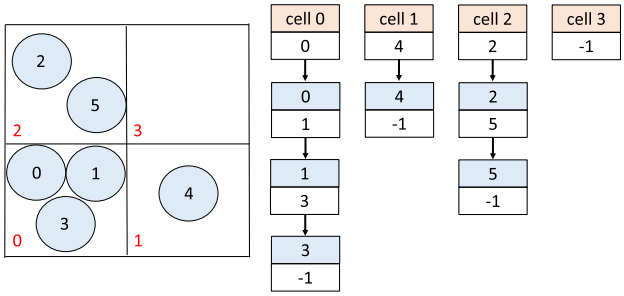

図 3　連結リスト法による近傍探索
Fig. 3　Neighboring cell list using linked-list method.

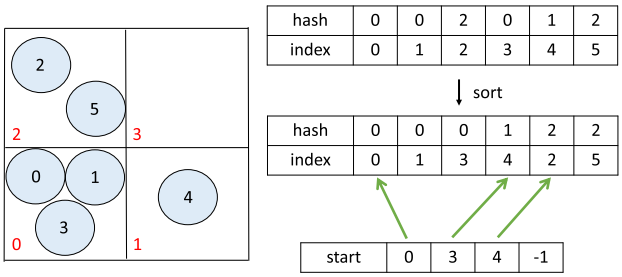

図 4　ハッシュ法による近傍探索
Fig. 4　Neighboring cell list using hash method.

各セルはセル内の粒子のインデックスをすべて持つのではなく，連結リストの先頭の粒子インデックスのみを格納する．セル内に粒子が存在しない場合は，粒子が存在しないことを表す -1 を格納する．計算しようとする粒子の近傍粒子探索の際は，その粒子の属するセルと周囲のセルのリストをたどっていくことにより，接触する粒子を判定する．

3.3　ハッシュ法

ハッシュ法の概念図を**図 4** に示す．各粒子のインデックスを *index* 配列に，各粒子の所属するセルの番号を粒子のインデックスに対応した *hash* 配列に格納する．*hash* 配列の値のソートを行い，ソートされたハッシュ配列を用いて *index* 配列を再配置して，粒子インデックスをセル番号が小さい順に整列する．次に，セル数を要素に持つ *start* 配列に，セル内の粒子の *index* 配列での始点の位置を記憶させる．セル内に粒子が含まれていない場合は，粒子が存在しないことを表す -1 を記憶させる．近傍探索の際は，周囲のセルの番号とセル内粒子の始点を取得して，所属しているセルの番号が変わるまで粒子のインデックスを読み込む．

3.4　粒子登録法

粒子登録法 [3], [4] は，相互作用を計算する可能性のある近傍粒子のインデックスを各粒子が近傍リストとして記憶し，粒子の移動距離が短いタイムステップ中は近傍リスト内の粒子とのみ距離判定および相互作用計算を行う．粒子登録法の概念図を**図 5** に示す．注目している粒子に対

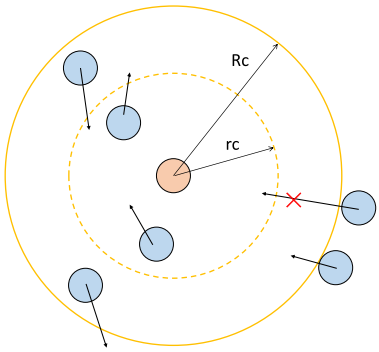

図 5　粒子登録法による近傍探索
Fig. 5　Book-keeping method.

して，影響半径 rc よりも大きい半径 Rc を設定し，Rc 内に含まれる粒子を近傍リストに格納する．DEM の場合は，影響半径 rc は粒子 i の半径 R_i と粒子 j の半径 R_j の和であり，Rc の大きさは

$$Rc = (R_i + R_j) + \alpha D_{\min} \tag{6}$$

で設定する．ここで，$D_{\min}$ は最小の粒子直径，α は近傍リストの更新頻度を調整するパラメータである．ある粒子は，その粒子が持つ近傍リストに含まれる粒子とのみ接触判定をすることで，遠方の粒子との無駄な計算が必要なくなり，近傍探索にかかる時間を削減できる．図 5 の赤いバツ印がついた粒子のように，近傍リストに含まれていない粒子が影響半径 rc 内に入った場合は正確な近傍探索が行えないため，近傍リストを更新する必要がある．そのため，以下に記すようにして近傍リストの更新が必要であるかの判定を行う．現在のタイムステップで全粒子の中で最大の速さ $v_{\max}$ を求めて，1 タイムステップでの移動距離を粒子移動距離の積載値 x_{book} に加算していく．

$$x_{\text{book}} \mathrel{+}= v_{\max}\Delta t \tag{7}$$

x_{book} が $(Rc - rc)/2$ を上回った場合に全粒子の近傍リストの更新を行うことで，近傍リスト外の粒子が影響半径内に入ることを防ぐことができる．近傍リストの更新が行われたら，粒子移動距離の積載値 x_{book} を 0 に初期化する．

3.5　粒子登録法とセル分割法のハイブリット手法

粒子登録法とセル分割による近傍探索手法を組み合わせた手法の概念図を**図 6** に示す．粒子登録法の近傍リストの作成で，全粒子との距離の計算をすると膨大な時間がかかるため，セル分割法による近傍探索を用いて近傍リストの作成にかかる時間を大幅に短縮する．空間を分割するセルの大きさは，近傍リストを作成する半径 Rc よりも大きくする．これにより，粒子が所属するセルと周囲のセルに含まれる粒子との距離を計算するだけで，近傍リストの作成が可能となる．図 6 のように粒子登録法により，周囲のセルに含まれる粒子からさらに近傍の粒子を絞り込むため，セ

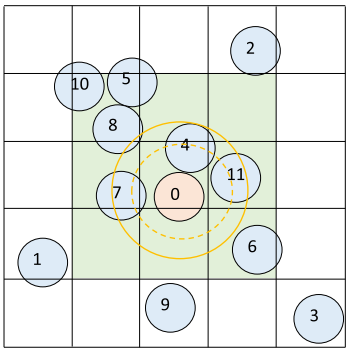

図 6　粒子登録法とセル分割法のハイブリット手法による近傍探索
Fig. 6　Combination method of book-keeping method with neighboring cell list.

ル分割による近傍探索に比べて相互作用計算で参照する粒子数が少なくなり，相互作用計算の高速化が可能である．また，メモリ使用量を抑えるために，粒子登録法とハッシュ法のハイブリット手法 [8] や粒子登録法と連結リスト法のハイブリット手法 [9] が提案されている．

4. DEM の GPU 計算の実装方法

4.1 DEM の相互作用計算と時間積分

GPU による DEM 計算では，メモリ量やデータアクセスの観点から各粒子が持つ位置や速度などの物理量は GPU ボード上の Device メモリに保持する．GPU で DEM の相互作用計算のスレッド並列計算を行う方法として，1 スレッドが 1 粒子を担当する方法 [5], [11], [16] や，1 スレッドが 1 つの接触点を計算する方法 [8] などがある．本論文では，多くの先行研究で行われているように，1 スレッドが 1 粒子を担当して，担当する粒子が接触しているすべての粒子との相互作用を計算する．また，相互作用計算では作用反作用の関係を利用せず，粒子 *i*-*j* の接触力と粒子 *j*-*i* の接触力を重複して計算している．粒子の運動方程式の時間積分は，1 スレッドが 1 粒子を計算する．

4.2 連結リストの作成方法

連結リストの作成では，1 スレッドが 1 粒子を担当して，粒子が所属しているセル番号の計算と他のセル内粒子とのリンクを作成する．粒子どうしのリンクを作成する際に逐次処理が必要となるため，GPU 上で連結リストを作成する際は，CUDA により提供されている atomic 関数の `atomicExch` を利用して行う．

4.3 ハッシュ法の実装

GPU 上での *hash* 配列のソートと *index* 配列の再配置には，Thrust ライブラリの `sort_by_key` を使用する．このとき，位置や速度などの粒子の物理量を *hash* 配列を用いてセル番号順に再配置することで，粒子物理量へのメモリアクセスが高速化される [7]．粒子物理量の再配置を毎ステップ行うことは，しばしばオーバヘッドになるので，本論文では 5.2 節で示すように粒子物理量の再配置を毎ステップ行うのではなく，数百ステップに一度行うように頻度を調整した．

表 1　物理条件
Table 1　Physical condition.

Particle diameter	[m]	2.0×10^{-3}
Particle mass	[kg]	1.0×10^{-5}
Spring constant (Normal)	[N/m]	4.0×10^{3}
Spring constant (Tangential)	[N/m]	1.6×10^{3}
Damping coefficient (Normal)	[Ns/m]	0.4
Damping coefficient (Tangential)	[Ns/m]	0.25
Coefficient of friction	[-]	0.3
Discrete time	[s]	5.0×10^{-6}

4.4 粒子登録法とセル分割法のハイブリット手法の実装

本研究では，粒子登録法と連結リスト法を組み合わせたハイブリット手法と，粒子登録法とハッシュ法のハイブリット手法を実装する．近傍リストの作成は，1 スレッドが 1 粒子を計算するスレッド並列化を行う．

5. 各近傍探索手法の性能比較

5.1 性能比較に用いる問題の設定

各近傍探索手法を実装した DEM 計算のプログラムで，512 個から 8,388,608 個の大きさの等しい球形の粒子を用いたダム崩壊問題に対するシミュレーションを行い，計算時間を比較する．詳しい物理条件と計算条件を**表 1**，**表 2** に示す．

粒子の物理量は単精度であり，時間発展には Leap-frog 法を用いる．粒子登録法の近傍リストの更新頻度を調整するパラメータ α は 0.05 に設定し，近傍リストを作成する半径は粒子直径の 1.05 倍である．近傍リストの配列の大きさは，各粒子が 12 個の粒子のインデックスを格納できる分確保している．格子サイズは，粒子登録法を用いない場合は粒子直径と等しく，粒子登録法を用いる場合は近傍リストを作成する半径（粒子直径の 1.05 倍）と等しく設定した．

図 7 に示すように計算領域の隅にダム崩壊前の初期粒子配置を設定する．ダムの初期粒子配置は，計算領域を分割する格子から擬似乱数を用いて少しずらして粒子を配置し，表 2 のダムの大きさの容器に粒子を自由落下させて，安定するまで計算を行ったものである．計算領域の x，y 方向のサイズは，初期粒子配置の x，y 方向のサイズの 4 倍に設定する．**図 8** は初期配置から 10 万ステップ後のダム崩壊中の様子であり，粒子の色は，粒子の速さが大きいほど赤く，小さいほど青くしている．図 7 の初期配置から図 8 までの 10 万ステップの計算を行うのに要した時間を測定し，各近傍探索手法の性能比較を行う．粒子数を変化させ

表 2　計算条件
Table 2　Calculational condition.

Number of particles	Size of dam [m^3]	Number of cells (without book-keeping method)	Number of cells (using book-keeping method)
512	$0.004 \times 0.004 \times 0.256$	$9 \times 9 \times 151$	$8 \times 8 \times 143$
2,048	$0.008 \times 0.008 \times 0.256$	$17 \times 17 \times 151$	$16 \times 16 \times 143$
8,192	$0.016 \times 0.016 \times 0.256$	$33 \times 33 \times 151$	$31 \times 31 \times 143$
32,768	$0.032 \times 0.032 \times 0.256$	$65 \times 65 \times 151$	$61 \times 61 \times 143$
131,072	$0.064 \times 0.064 \times 0.256$	$129 \times 129 \times 151$	$122 \times 122 \times 143$
524,288	$0.128 \times 0.128 \times 0.256$	$257 \times 257 \times 151$	$244 \times 244 \times 143$
2,097,152	$0.256 \times 0.256 \times 0.256$	$513 \times 513 \times 151$	$488 \times 488 \times 143$
8,388,608	$0.512 \times 0.512 \times 0.256$	$1{,}025 \times 1{,}025 \times 151$	$976 \times 976 \times 143$

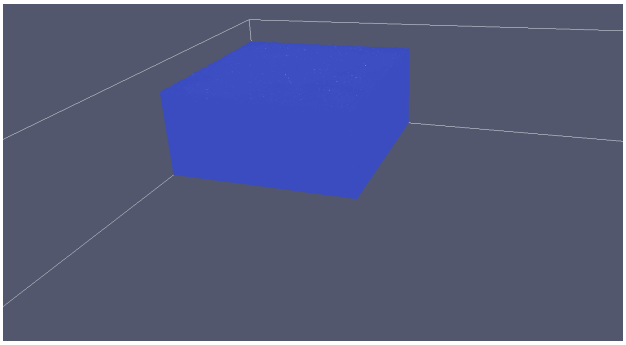

図 7　ダム崩壊問題初期配置（粒子数 8,388,608）
Fig. 7　A initial condition of braking dam for a performance test (Number of particles 8,388,608).

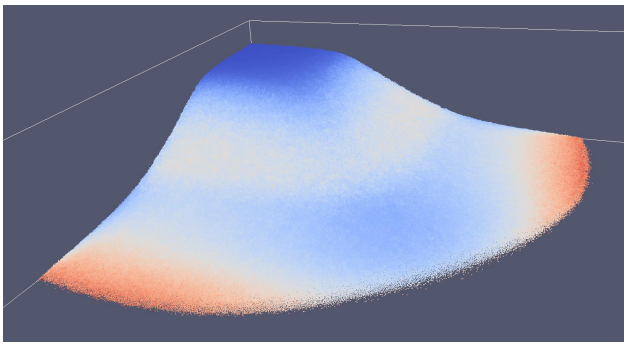

図 8　10 万ステップ計算後のダムの様子（粒子数 8,388,608）
Fig. 8　A simulation result of breaking dam simulation at 100,000 steps (Number of particles 8,388,608).

た場合の計算効率を比較するために，1 秒間に 1 ステップ更新できる粒子数を表す $Cundall\,Number$ を求める．

$$Cundall\,Number = \frac{NP}{TIME/STEP} \tag{8}$$

ここで，NP は計算に使用した粒子数，$TIME$ は全計算ステップの実行時間，$STEP$ は実行ステップ数である．

5.2　粒子物理量の再配置

ハッシュ法で，粒子のインデックスだけでなく，ソートされた hash 値を用いて位置や速度などの粒子の物理量もセル番号順に再配置することで，相互作用計算で接触粒子の物理量へのメモリ参照がコアレスアクセスとなり，キャッシュにも載りやすくなる [7]．粒子物理量の再配置は以下のようにして行う．まず，セル番号順に並べ替えられた粒子インデックスの配列 $index$ を用いて，粒子物理量の変数 $Variable$ を再配置のための配列 tmp に代入する．tid はスレッドの番号である．

$$tmp[tid] = Variable[index[tid]] \tag{9}$$

tmp 配列への代入が終わり次第，元の変数の配列に値を戻す．

$$Variable[tid] = tmp[tid] \tag{10}$$

この操作を粒子の位置や速度などの変数 1 つ 1 つに対して行っていく．しかし，DEM では各粒子は位置や速度のほかにも，接触している粒子との接線方向のバネの圧縮量を持つため，1 つの粒子に対して再配置を行う変数が多い．粒子物理量の再配置には時間がかかるため，毎ステップ行うのではなく，再配置の頻度を調整する必要がある．そこで，表 2 の粒子数が 131,072 の条件で，近傍粒子探索をハッシュ法とし，粒子物理量の再配置を行わない場合，再配置の頻度を 1 ステップに一度，100 ステップに一度，300 ステップに一度，1,000 ステップに一度とした場合について計算時間を測定し，再配置による相互作用の計算時間への影響と再配置のオーバヘッドを調べた．粒子物理量の再配置は，近傍探索でのハッシュ法の $index$ 配列の再配置とは別のカーネル関数で行う．

表 3 に計算時間の内訳を示す．表の左側から順に，粒子再配置の頻度と，相互作用計算，粒子物理量の再配置，ハッシュ法，時間積分および計算全体にかかった時間を示す．粒子物理量の再配置を毎ステップ行うことで，粒子間相互作用の計算にかかる時間を 1,129.66 秒から 887.84 秒に短縮できているが，毎ステップの再配置にかかる時間は計算全体の 53.3%で大きなオーバヘッドとなり，計算全体は粒子物理量の再配置を行わないほうが速い．再配置の頻度を減らすと，相互作用計算の時間は再配置を毎ステップ行った場合よりもわずかに増えるが，再配置にかかる時間が短

表 3　計算時間の粒子物理量の再配置頻度依存性
Table 3　Dependence of computation time on frequency of particle data relocation.

frequency of data relocation	interaction [s]	data relocation [s]	hash [s]	time integration [s]	total [s]
without data relocation	1,129.66	—	151.17	14.05	1,298.22
every 1 time step	887.84	1,201.50	147.06	14.01	2,254.97
every 100 time steps	900.42	12.24	150.62	14.12	1,080.91
every 300 time steps	909.03	4.10	146.78	14.06	1,077.30
every 1,000 time steps	921.40	1.24	147.00	14.03	1,086.89

表 4　連結リスト法とハッシュ法の計算時間
Table 4　Computation time of linked-list method and hash method.

	linked-list method			hash method		
number of particles	total [s]	linked-list [s]	ratio	total [s]	hash [s]	ratio
512	69.72	6.80	0.098	105.42	43.90	0.416
2,048	78.44	7.38	0.094	143.15	62.99	0.440
8,192	81.05	7.13	0.088	156.36	78.18	0.500
32,768	236.38	8.70	0.037	321.11	81.69	0.254
131,072	812.81	15.53	0.019	986.84	150.28	0.152
524,288	3,230.36	42.68	0.013	3,637.28	292.37	0.080
2,097,152	13,220.62	151.42	0.011	14,368.74	665.74	0.046
8,388,608	53,383.79	583.70	0.011	57,463.59	2,205.29	0.038

くなるため，再配置を行わない場合よりも計算全体の時間を短縮できている．再配置の頻度を 300 ステップに一度とした場合が最も計算時間が短く，再配置を行わない場合に比べ，全体の計算時間を 83.0%に短縮することができた．

5.3　実行性能の比較

5.2 節で適切な頻度で粒子物理量をセル番号順に再配置することで計算の高速化を行えることが確認できたので，ハッシュ法だけでなく連結リスト法，粒子登録法とハッシュ法のハイブリット手法，粒子登録法と連結リスト法のハイブリット手法にも粒子物理量の再配置を実装し，性能比較を行う．ハッシュ法と連結リスト法は 300 ステップごとに粒子物理量の再配置を行う．粒子登録法とハッシュ法のハイブリット手法，粒子登録法と連結リスト法のハイブリット手法では，粒子登録法の近傍粒子リストの更新のときに粒子物理量の再配置を行う．粒子登録法を用いた場合は，粒子物理量の再配置の頻度はステップ数で設定するのではなく，近傍粒子リストの更新回数で指定する．近傍リストの更新頻度は，粒子群の速度に依存するが，平均して 10 から 20 ステップに一度なので，近傍リストが 30 回更新された場合に粒子物理量の再配置を行う．

各近傍探索手法でダム崩壊シミュレーションを行った場合の実行時間を**図 9** に示す．横軸に粒子数を，縦軸に実行時間を示す．連結リスト法とハッシュ法を比較すると，各粒子数の計算条件において連結リスト法を用いた場合のほうが計算時間が短く，粒子数が少なくなるほど相対的な差（ハッシュ法の計算時間/連結リスト法の計算時間）は大きくなる．近傍探索手法に連結リスト法，ハッシュ法を

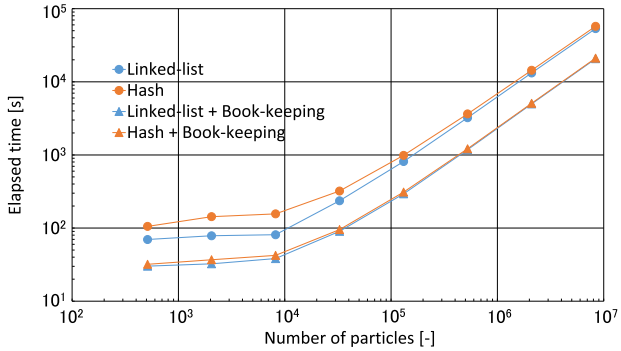

図 9　各近傍探索手法での計算時間
Fig. 9　Computation time of each neighbor-particle searching method.

用いた場合の計算時間，連結リストの作成に要した時間およびハッシュ法での *hash* 配列のソートと *index* 配列の再配置，*start* 配列の作成に要した時間，それらの全計算時間に対する割合を**表 4** に示す．ハッシュ法のソートは，連結リストの作成時間の 3.8 倍から 11.0 倍の時間がかかり，この差により計算全体でも連結リスト法のほうが速い．全体の計算時間に対するハッシュ法と連結リスト法の時間の割合は，粒子数が多い 8,388,608 粒子の場合は，それぞれ 3.8%，1.1%であるが，粒子数が最も少ない 512 粒子では，それぞれ 41.6%，9.8%であり，粒子数が少なくなるにつれて連結リストの作成やハッシュ法に費やす時間の割合が増えるため，全体の計算時間への影響が大きくなる．そのため，粒子数が少ないほど連結リスト法とハッシュ法の計算時間に差が開いたと考えられる．

図 9 の三角形のマーカの粒子登録法と連結リスト法あ

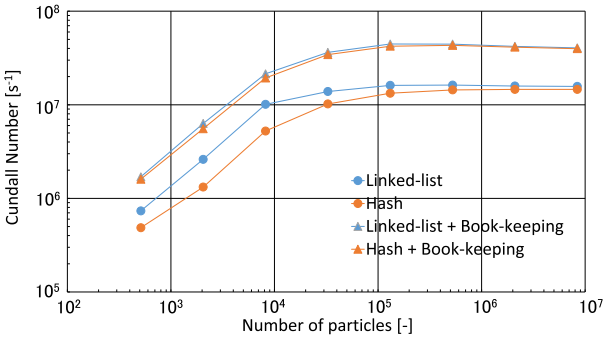

図 10　各近傍探索手法でのパフォーマンス測定結果
Fig. 10　Performance of each neighbor-particle searching method.

るいはハッシュ法を併用したハイブリット手法を用いた場合では，連結リスト法，ハッシュ法のみの場合に比べて大幅に計算時間を削減できている．連結リスト法やハッシュ法では，相互作用計算のときに参照する領域は，1 辺が影響半径の 3 倍の大きさの立方体であり，接触していない粒子が多く含まれる．一方，粒子登録法を併用することにより，相互作用計算で参照する領域は，半径が影響半径よりもわずかに大きい球形なので，接触していない粒子の参照が少なくなる．そのため，接触していない粒子との無駄な接触判定の計算が少なくなり，相互作用に要する時間が短縮された．

各近傍探索手法での *Cundall Number* を図 **10** に示す．横軸が粒子数，縦軸が *Cundall Number* である．最も計算性能の高い近傍探索手法は粒子登録法と連結リスト法のハイブリット手法であり，性能が低いのはハッシュ法である．粒子数が 10 万個以上では，どの近傍探索手法でも *Cundall Number* はほぼ一定の値をとっていて，粒子数により計算性能は変わらなくなる．粒子数が 10 万個以下の少ない条件では，粒子数が少なくなるにつれて計算効率も低下している．特に粒子数が 1 万個以下になると性能が急激に低下する．計算時間の最もかかる粒子間相互作用を計算するカーネル関数に対して，nvcc のコンパイルオプション「`--ptxas-options=-v`」でその関数内で使用されるレジスタ数を求めると，連結リスト法とハッシュ法の計算コードでは，その関数内で使用されたレジスタ数は 67，粒子登録法とハッシュ法のハイブリット手法および粒子登録法と連結リスト法のハイブリット手法では，その関数内で使用されたレジスタ数は 65 である．本研究では相互作用計算のカーネル関数では 1 ブロックあたり 256 スレッド（8 warp）で設定した．相互作用計算のカーネル関数では共有メモリをほとんど利用していないため，占有率は Streaming Multiprocessor（SMX）あたりのレジスタの上限（Tesla K20X では SMX あたり 65,536 レジスタ）で決定される．レジスタ数から SMX あたりのブロック数を計算すると，どの近傍探索手法の計算コードであっても SMX あたり 3 ブロック，24 warp が実行される．Tesla K20X は 14 個の SMX が搭載されているので，合計で 336 warp，10,752 スレッドが実行される．本研究の実装方法では，1 スレッドに 1 つの粒子を割り当てるので，10,752 個の粒子が一度に並列して計算される．そのため粒子数が 10,752 以下では GPU の性能を十分に使うことができず，計算性能が急激に低下したと考えられる．図 9 の計算時間に関しても，粒子数が 10,752 以下では粒子数を少なくしても計算時間はほぼ変わらない．以上のことから，GPU の DEM 計算で，本論文で行ったように 1 スレッドが 1 粒子を担当するスレッド並列計算の場合は，粒子数が 10 万個以下の小規模な問題では，GPU の性能を十分に利用できないため，GPU による高速化の効果は低いと考えられる．

5.4　メモリ使用量の比較

粒子数を NP，1 つの粒子に接触する可能性のある最大の粒子数を NP_{contact}，セルの数を $CELL$，粒子登録法で各粒子が持つ近傍リストに登録される可能性のある最大の粒子数を NP_{neighbor} として，各手法を実装するのに必要なデータ数（メモリ使用量 = データ数 × 4 byte または 8 byte）を示す．

連結リスト法では，各セルが持つ先頭粒子のインデックスと，各粒子の持つ次の粒子のインデックスが必要であり，整数型のデータ数は次式で示される．

$$Data_{\mathrm{linked\text{-}list}} = CELL + NP \tag{11}$$

ハッシュ法では，要素数が粒子数と等しい *hash* 配列と *index* 配列，セル内粒子の始点を格納した要素数がセル数と等しい *start* 配列であり，整数型のデータ数は次式で示される．

$$Data_{\mathrm{hash}} = CELL + 2 \times NP \tag{12}$$

粒子登録法では，近傍リストの配列は大きさが $NP \times NP_{\mathrm{neighbor}}$ の 2 次元配列であり，各粒子は自身の近傍リストに登録されている粒子数を持つため，整数型のデータ数は以下のようになる．

$$Data_{\mathrm{book\text{-}keeping}} = (1 + NP_{\mathrm{neighbor}}) \times NP \tag{13}$$

粒子物理量の再配置では，ハッシュ法で用いたセル番号を格納する *hash* 配列と粒子インデックスを格納する *index* 配列が必要である．また，式 (9) と式 (10) のようにして再配置を行うには，一時的に並べ替えられる値を保持する配列が必要となる．位置や速度などの各粒子が個別に持つ物理量の再配置には，粒子数を要素数とした整数型の配列や浮動小数点数型の配列を用いる．DEM では，各粒子が，その粒子が接触している粒子番号と接線方向のバネ圧縮量を記憶している．これらは，$NP \times NP_{\mathrm{contact}}$ の大きさを持つ 2 次元配列であり，これらを再配置するときには，式

表 5　各近傍粒子探索手法で必要なメモリ量
Table 5　Size of memory usage in each neighbor-particle searching method.

	particle data	data relocation	hash	linked-list	book-keeping	total
linked-list	2,516.6	503.3	—	668.1	—	2,516.6 + 1,171.4
hash	2,516.6	436.2	701.7	—	—	2,516.6 + 1,137.9
book-keeping + linked-list	2,516.6	503.3	—	578.4	436.2	2,516.6 + 1,518.0
book-keeping + hash	2,516.6	436.2	612.0	—	436.2	2,516.6 + 1,484.4

unit: MB

(9) と式 (10) の tmp 配列を $NP \times NP_{\mathrm{contact}}$ の 2 次元配列としている．粒子が再配置されることで接触している粒子番号が変わるため，再配置された新しい粒子の並びに合わせて，接触している粒子番号を修正する必要があり，そのための整数型の配列（要素数 NP）も必要となる．再配置に必要な整数型のデータ数と浮動小数点数型のデータ数はそれぞれ式 (14) と式 (15) となる．

$$Data_{\mathrm{sort(integer)}} = 2 \times NP + (2 + NP_{\mathrm{contact}}) \times NP \quad (14)$$

$$Data_{\mathrm{sort(floating\,point)}} = (1 + NP_{\mathrm{contact}}) \times NP \quad (15)$$

連結リスト法とハッシュ法のデータ数を式 (11) と式 (12) から比較すると，近傍探索に必要なデータ数は，連結リスト法のほうがハッシュ法よりも NP 少ない．ハッシュ法では，$hash$ 配列と $index$ 配列は近傍探索でも用いる配列であるため，粒子物理量の再配置で $2 \times NP$ の配列の分（式 (14) の右辺第 1 項）だけ連結リスト法よりもデータ数が少なくなる．粒子データの容量は連結リスト法とハッシュ法で変わらないため，全体のデータの数はハッシュ法が連結リスト法よりも NP だけ少なくなる．

粒子登録法を用いる場合は，式 (13) の分だけ必要なデータ数が増加するが，格子サイズが大きくなり格子数が少なくなるため連結リスト法とハッシュ法に必要なデータ数は減少する．

一例として，粒子数 8,388,608 のダム崩壊シミュレーションを行ったときの，各近傍探索手法で確保したメモリ量を**表 5** に示す．左から粒子データ，粒子再配置，ハッシュ法，連結リスト法，粒子登録法で確保したメモリ量である．今回の計算では，位置や速度などの粒子物理量は単精度で確保し，1 つの粒子は 58 個の浮動小数点型の変数と 17 個の整数型の変数を持ち，粒子 1 つあたりのメモリ使用量は 300 byte である．格子数 $CELL$ は表 2 のとおりであり，接触する可能性のある最大の粒子数 NP_{contact} は 12，近傍リストに登録される可能性のある最大の粒子数 NP_{neighbor} は 12 である．

4 つの近傍探索手法ではハッシュ法が最も全体のメモリ使用量が少ない．最も計算が速い粒子登録法と連結リスト法のハイブリット手法では，全体のメモリ使用量は 4,034.5 MB であり，メモリ使用量が最も少ないハッシュ法の 3,654.5 MB と比べると 1.18 倍である．粒子登録法と連結リスト法のハイブリット手法のメモリ使用量は膨大ではなく，使用できるメモリ量が制限される GPU の場合でも，大規模計算に適用可能といえる．1 台の GPU で粒子登録法と連結リスト法のハイブリット手法よりも多くの粒子を用いて計算を行いたい場合は，メモリ使用量が少ないハッシュ法を用いればよいといえる．たとえば，粒子登録法と連結リスト法のハイブリット手法が必要とするメモリ量は 4,034.5 MB であり，一方でハッシュ法が必要とするメモリ量は 3,654.5 MB である．その差は 380.0 MB であり，これは 127 万個の粒子データに相当する．

6. 得られた結果の有用性

今回の近傍探索手法の比較では，ダムの初期配置は図 7 のように計算領域の隅に配置し，図 8 のように扇状にダムが崩壊するシミュレーションを行い，計算時間を測定した．たとえば，ダムの初期配置のアスペクト比を変更したり，ダムの位置を計算領域の中央に配置したりすると，ダム崩壊にかかる時間や崩壊中のダムの形状が変化する．連結リストの作成やハッシュ法のソート，粒子データの再配置などの処理にかかる計算時間は粒子数や格子数に依存し，粒子分布や粒子の速度による影響はほとんどない．粒子登録法では，近傍リストの更新頻度は粒子の速度に影響を受け，粒子が速いほど近傍リストの更新頻度は多くなる．しかし，セル分割法を用いて近傍リストの作成を高速に行っているため，更新頻度が増加したとしても計算全体の実行時間に与える影響は小さい．そのため，ダムの初期配置を変更して近傍探索手法の比較を行っても，同様の結果になると考えられる．

ダム崩壊問題以外のシミュレーションの場合でも，得られた知見が有効であるかを考える．近傍探索は粒子分布や粒子速度の影響が小さいため，ダム崩壊問題のように各粒子がつねに周囲の複数の粒子と接触している高濃度系の粉体現象のシミュレーションでは，同様の傾向が得られる．粒子の接触が少ない低濃度系の粉体現象の場合は，高濃度系に比べて粒子間相互作用の計算時間が短くなるため，全体の計算時間に対して，連結リストの作成やハッシュ法のソートに要する時間の割合が大きくなる．そのため低濃度系の粉体シミュレーションでは，連結リスト法とハッシュ法の実行性能の差がダム崩壊よりも大きくなると考えられる．

本論文の DEM 計算では，粒子間相互作用で粒子間の

摩擦が考慮され，粒子は回転自由度を持っており，計算に用いた粒子はすべて同じ大きさの球形である．しかし，DEM 計算には，図 1 の接線方向の摩擦がなく粒子が回転しない計算 [16] や球形ではなく非球形の粒子を用いた計算 [12], [14], [17], [18] など，本論文で行った DEM とは異なるモデルを用いた計算も行われている．

接線方向の摩擦力が働かず回転自由度のない粒子を用いた場合は，近傍探索は本論文と同様な方法で行われるが，接線方向の力を考慮しないため接触力の計算が高速である．そのため，近傍探索手法の実行性能の差は本論文の結果よりも大きくなると考えられる．せん断方向のバネの圧縮量を保持する必要がないため，粒子データと粒子再配置に必要なメモリ量は少なくなる．

非球形の粒子を用いる方法には，球形粒子を連結して表現するモデル [17], [18] とポリゴンを用いるモデル [12], [14] がある．球形粒子を連結した非球形粒子を用いた計算の場合は，接触判定と相互作用計算は球形粒子単位で行われるため，本研究で得られた結果が有効である．ポリゴンを用いた非球形粒子の衝突判定は，接触する可能性がある粒子を判定するブロードフェーズと，ブロードフェーズで検出された粒子とポリゴンの点と点や点と面などの接触判定をするナローフェーズで構成される．ブロードフェーズでは本研究で比較した近傍探索が利用されることもあるため，本研究で得られた知見を活用できる．

7. 結言

接触相互作用に基づく粒子法である DEM の GPU での計算において，連結リスト法と，ハッシュ法と，粒子登録法と連結リスト法のハイブリット手法と，粒子登録法とハッシュ法のハイブリット手法の 4 つの近傍探索手法を実装し，各手法の違いによる計算時間およびメモリ使用量の比較を行った．近傍探索手法を比較する例題として，512 個から 8,388,608 個の粒子を用いたダム崩壊シミュレーションをそれぞれの近傍探索手法で計算し，計算時間の測定を行った．数百ステップに 1 回の頻度で粒子物理量をセル番号順に再整列することによって相互作用計算におけるメモリアクセスが改善され，計算時間を短縮できることを確認した．連結リスト法とハッシュ法を比較した結果，GPU 計算においても連結リスト法のほうが計算時間が短く，粒子数が少ないほど計算時間の差が大きくなることが分かった．近傍探索単体を実装するために必要なメモリ量は連結リスト法のほうが少ないが，ハッシュ法では粒子物理量の再配置に近傍探索で用いる配列を使いまわせるため，全体ではハッシュ法のほうが連結リスト法よりもメモリ使用量は少ない．粒子登録法と連結リスト法のハイブリット手法が 4 つの近傍探索手法のなかで最も計算時間が短く，そのメモリ使用量はメモリ使用量が最も少ないハッシュ法の 1.18 倍で抑えられている．以上のことより，メモリに余裕がある場合は近傍探索手法に粒子登録法と連結リスト法のハイブリット手法を，メモリ使用量を極力抑えたい場合はハッシュ法を実装するのが適切であることが分かった．すべての近傍探索手法で，粒子数が 10 万個以上の場合は計算性能を示す *Cundall Number* は粒子数に影響されずにほぼ一定であるが，粒子数が少ない 10 万個以下の小規模な計算の場合は，粒子数が少なくなるほど *Cundall Number* が低下する傾向が見られた．1 スレッドが 1 粒子を計算する実装方法では，小規模な DEM 計算においては，GPU の性能を十分に利用できないため，GPU による高速化は効果的でないことが分かった．

DEM を使った粉体シミュレーションの GPU 化の研究は多数行われているが，計算の主要部分を占める近傍探索において，代表的な手法である連結リスト法，ハッシュ法，粒子登録法を連結リスト法やハッシュ法と組み合わせたハイブリット手法を GPU に実装し，それらを詳細に比較した報告はなく有用性が高い．

謝辞　本研究の一部は，科学研究費補助金・基盤研究 (S) 課題番号 26220002「ものづくり HPC アプリケーションのエクサスケールへの進化」，科学技術振興機構 CREST「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」，学際大規模情報基盤共同利用・共同研究拠点，および革新的ハイパフォーマンス・コンピューティング・インフラから支援をいただいた．記して謝意を表す．

参考文献

[1] Cundall, P.A. and Strack, O.D.: A discrete numerical model for granular assemblies, *Geotechnique*, Vol.29, No.1, pp.47–65 (1979).
[2] MacLaughlin, M. and Doolin, D.: Review of validation of the discontinuous deformation analysis (DDA) method, *International Journal for Numerical and Analytical Methods in Geomechanics*, Vol.30, No.4, pp.271–305 (2006).
[3] Allen, M.P. and Tildesley, D.J.: *Computer simulation of liquids*, Oxford University Press (1989).
[4] 小口かなえ，澁田　靖，鈴木俊夫：GPU を用いた分子動力学法解析の高速化，日本金属学会誌，Vol.76, No.7, pp.462–467 (2012).
[5] 茂渡悠介，酒井幹夫，越塚誠一，山田祥徳：GPU による離散要素法シミュレーションの高速化，粉体工学会誌，Vol.45, No.11, pp.758–765 (2008).
[6] 平林久義，佐藤雅弘：線形リストを用いた粒子法の近傍粒子探索，日本計算工学会論文集，Vol.2010, No.20100001 (2010).
[7] Green, S.: Particle simulation using cuda, *NVIDIA Whitepaper, December 2010* (2010).
[8] 西浦泰介，阪口　秀：GPU を用いた DEM の高速化アルゴリズム，日本計算工学会論文集，Vol.2010, No.20100007 (2010).
[9] Viccione, G., Bovolin, V. and Carratelli, E.P.: Defining and optimizing algorithms for neighbouring particle identification in SPH fluid simulations, *International Journal for Numerical Methods in Fluids*, Vol.58, No.6, pp.625–638 (2008).

[10] Yao, Z., Wang, J.-S., Liu, G.-R. and Cheng, M.: Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method, *Computer Physics Communications*, Vol.161, No.1, pp.27–35 (2004).
[11] Tsuzuki, S. and Aoki, T.: Large-scale granular simulations using dynamic load balance on a GPU supercomputer, *Poster at the 26th IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis* (*SC*) (2014).
[12] Govender, N., Wilke, D.N., Kok, S. and Els, R.: Development of a convex polyhedral discrete element simulation framework for NVIDIA Kepler based GPUs, *Journal of Computational and Applied Mathematics*, Vol.270, pp.386–400 (2014).
[13] Tian, Y., Qi, J., Lai, J., Zhou, Q. and Yang, L.: A heterogeneous CPU-GPU implementation for discrete elements simulation with multiple GPUs, *2013 International Joint Conference on Awareness Science and Technology and Ubi-Media Computing* (*iCAST-UMEDIA*), pp.547–552, IEEE (2013).
[14] Govender, N., Wilke, D.N. and Kok, S.: Collision detection of convex polyhedra on the NVIDIA GPU architecture for the discrete element method, *Applied Mathematics and Computation*, Vol.267, pp.810–829 (2014).
[15] Washizawa, T. and Nakahara, Y.: Parallel Computing of Discrete Element Method on GPU, *arXiv preprint arXiv:1301.1714* (2013).
[16] Xu, J., Qi, H., Fang, X., Lu, L., Ge, W., Wang, X., Xu, M., Chen, F., He, X. and Li, J.: Quasi-real-time simulation of rotating drum using discrete element method with parallel GPU computing, *Particuology*, Vol.9, No.4, pp.446–450 (2011).
[17] Ono, I., Nakashima, H., Shimizu, H., Miyasaka, J. and Ohdoi, K.: Investigation of elemental shape for 3D DEM modeling of interaction between soil and a narrow cutting tool, *Journal of Terramechanics*, Vol.50, No.4, pp.265–276 (2013).
[18] Matsushima, T., Katagiri, J., Uesugi, K., Tsuchiyama, A. and Nakano, T.: 3D shape characterization and image-based DEM simulation of the lunar soil simulant FJS-1, *Journal of Aerospace Engineering*, Vol.22, No.1, pp.15–23 (2009).

渡辺 勢也

1991 年生．2014 年群馬工業高等専門学校専攻科生産システム工学専攻修了．2014 年東京工業大学大学院総合理工学研究科創造エネルギー専攻修士課程進学．日本計算工学会，粉体工学会各会員．

青木 尊之　（正会員）

1960 年生．1983 年東京工業大学理学部応用物理学科卒業．1985 年同大学大学院総合理工学研究科エネルギー科学専攻修了．1985 年富士通研究所厚木研究所入社．1986 年東京工業大学大学院総合理工学研究科助手．1997 年同大学原子炉工学研究所助教授．2001 年同大学学術国際情報センター教授．数値流体力学，計算力学，GPU コンピューティングの研究に従事．文部科学大臣表彰，日本応用数理学会業績賞，ACM ゴードンベル賞ほか．日本機械学会フェロー，日本応用数理学会等各会員．

都築 怜理　（学生会員）

1988 年生．2011 年東京工業大学理学部物理学科卒業．2013 年同大学大学院総合理工学研究科創造エネルギー専攻修士課程修了（理学）．2013 年同大学院総合理工学研究科創造エネルギー専攻博士課程進学．HPCS2015 最優秀論文賞，IEEE Computer Society Japan Chapter 優秀若手研究賞ほか．日本学術振興会特別研究員（DC2）．IEEE，日本応用数理学会等各会員．

下川辺 隆史　（正会員）

1983 年生．2005 年東京工業大学理学部物理学科卒業．2007 年同大学大学院理工学研究科基礎物理学専攻修士課程修了．2012 年同大学院総合理工学研究科創造エネルギー専攻博士課程修了．博士（理学）．2012 年同大学学術国際情報センター特任助教．2013 年同大学同センター助教．ペタスケールの大規模物理計算の研究に従事．2011 年 ACM ゴードンベル賞・特別賞受賞．日本計算工学会，日本応用数理学会，IEEE-CS，ACM 各会員．