位置ベース粒子法を用いた
高速な融解シミュレーション

筑波大学

図書館情報メディア研究科

2016年3月

仲宗根 良

目 次

第 1 章　序論 **1**
1.1　背景と目的 1
1.2　関連研究 2
1.2.1　融解シミュレーションに関する先行研究 2
1.2.2　弾性体シミュレーションに関する先行研究 3

第 2 章　提案手法 **4**
2.1　全体の流れ 4
2.2　液体の運動計算 4
2.3　弾性体の運動計算 6
2.3.1　クラスタによる領域分割 6
2.3.2　弾性体を計算するための制約条件 7
2.3.3　反復法ソルバー 12
2.3.4　クラスタのサンプリング 16
2.3.5　姿勢情報の補間 18
2.4　粘弾性体の運動計算 19
2.5　陰関数場の生成とレンダリング 20

第 3 章　実験と考察 **21**
3.1　実験環境 21
3.2　剛性調整に関する実験 22
3.2.1　実験設定 22
3.2.2　実験結果と考察 23
3.3　計算時間を固定した実験 26
3.3.1　実験設定 26
3.3.2　実験結果と考察 28
3.4　融解シミュレーション 30

第 4 章　結論 **32**
4.1　まとめ 32
4.2　今後の課題 32

謝辞 **33**

参考文献 **34**

図 目 次

2.1 クラスタを重ね合わせて物体を構成 7
2.2 粒子 x_k^0 が x_k へ移動 7
2.3 並行移動と回転を推定して得た目標位置 g_k へ x_k を移動 8
2.4 SM 法 (左) と OP 法 (右) の比較．SM 法は反復 20 回，OP 法は反復 10 回，サンプリング数は共通して 832 個で計算時間は約 30fps．耳の部分に差が出ている． 11
2.5 単純な平均と重み付き和の比較 15
2.6 許容変形量が小さい場合の反復回数を比較 16
2.7 反復処理による剛性の調整．反復 1 回目は移動した粒子の属するクラスタのみが移動の影響を受ける．反復 2 回目では隣接するクラスタに影響が伝播する． 17
2.8 サンプリングの例．白い粒子は選択されたクラスタに対応する粒子である．このモデルでは約 800 個がサンプリング数の下限となっている． 17
2.9 反復回数とサンプリング数を調整して実験．計算時間は全て約 30fps． 18
2.10 補間の有無によるアニメーションの違いの比較．Stanford bunny を落下させる実験を行った． 19

3.1 変形状態から元の形状への回復を確認するための初期状態と目標形状 23
3.2 制約条件とソルバーを組み合わせた場合の反復回数と変形量を比較 24
3.3 制約条件とソルバーの組み合わせ．左から反復 100 回目,400,700,1000 回目 . 25
3.4 制約条件とソルバーを組み合わせた場合の反復回数と変形量を比較 26
3.5 ソルバーの改良結果．左から反復 100 回目,200,300,1000 回目 27
3.6 計算時間を固定した実験結果．左から初期状態,15,30,45,60 フレーム目 29
3.7 固体から液体への融解 31

表 目 次

3.1 実行環境 21
3.2 シミュレーションパラメータ 21
3.3 各組み合わせの計算時間と反復回数 28
3.4 融解シミュレーションの平均計算時間と反復回数 30

第 1 章

序論

1.1 背景と目的

CG を用いて自然現象を再現することは，エンターテイメント分野において重要なテーマとなっている．特に，水，煙，布，人体といった我々の身近に存在する現象のアニメーションは，映画，広告，ゲームなどにおいて多用される演出であるため，多くの需要が存在する．従来，自然現象を対象にしたアニメーションの制作には多くのノウハウが不可欠であり，クリエータの熟練した技術と経験が必要であった．しかし，近年，自然現象・物理現象を物理法則に基づいたシミュレーション (以下，物理シミュレーション) により，人の手による高度な技術が無くとも高品質なアニメーションを作成することが可能となっている．

エンターテイメントの中でも，映画やテレビなどの映像分野では高品質で現実に近いアニメーションが求められている．この場合では，決められた視点や状況から最終的なアニメーションを出力することが目的なので，シミュレーションやレンダリングに十分な計算時間を用いることができる．一方，ゲームなどのインタラクティブなアプリケーションでは入力に対する高速な反応が求められる．そのため，大きな計算コストを要する厳密な物理式を用いることはできず，計算式を近似したり表現を簡易化して効率よく計算を行う必要がある．このように，エンターテイメントの分野によっては求められる品質と計算時間の優先順位に差があるため，ある自然現象のアニメーション作成に用いられる物理シミュレーション手法は複数存在する事が多い．

初期の CG における物理シミュレーションは，単純なシーンや単一の現象を扱うものが多かった．これらのシミュレーションに関する研究はこれまで数多く行われており，オフラインでもリアルタイムでも様々な CG アニメーションを作成できるようになっている．一方で，より複雑なシーンや様々な現象の相互作用を対象としたものはまだまだ研究が始まったばかりである．例えば，水，煙，砂，布，紐といった様々な物体のアニメーションを，リアルタイムに統一的なフレームワークで作成するための研究 [3] や，衝撃によって 1 つの物体が複数の破片へ分かれる破砕現象をリアルタイムに実現したもの [4] が存在する．これらは，現実世界で起こるような複数の物体が干渉し合う複雑な状況であったとしても，単純なパラメータを操作するだけで物体の性質や動作を調整でき，容易かつ自動的にアニメーションを作成することができる手法である．

本論文では，様々な自然現象の中でも特に物体の融解現象をリアルタイムに表現するための手法について述べる．温度変化によって物体が固体から液体へ状態変化する，という融解現象は様々な物体に見られる．例えば，氷は物体温度の上昇に伴い固体から水という液体へ変化する．このとき，物体の形状や熱の与え方によって物体の分離などの構造変化が発生す

る．この融解現象を CG で表現するには，液体と固体の両方の特徴を考慮して運動計算すればよい．しかし，ゼリーのような物体では固体と液体に加えてその中間状態である弾性体，塑性体，粘弾性体と複雑に変化する．そのため，これら複数の物体を同時に扱い，かつそれらの相互作用も考慮してシミュレーションしなくてはならない．そこで本論文では，物体の融解現象に伴う物性変化の過程を CG アニメーションのために単純化し，物体が変化し得る状態を固体，弾性体，粘弾性体，液体という 4 つに限定する．ただしここでの固体とは連続体の分類上の呼び名ではなく，変形が小さい弾性体を表すために便宜上用いる．提案手法では，弾性体の硬度を調整することで硬い物体を表現するため，非変形が前提の剛体を表現することは原理上困難である．そのため，上記のような固体という表記を硬い物体を表すために用いる．この単純なモデルを元に，リアルタイムでの融解アニメーションを作成する．

固体は，一定の形と体積を持った物体で変形しにくい特殊な弾性体であり，液体は，形は不定だが体積が一定である物体の状態を示し，外力に応じてその形状を流動的に変える．弾性体は，外力により変形すると一時的に形は不定であるが，しばらくすると元の形に戻ろうとする物体を示し，粘弾性体は，液体と弾性体の両方の性質を併せ持つ物体である．粘弾性体は，緩やかに変形するような粘性挙動と，力が加わると瞬時に伸びて加重から開放されると即座に元の状態に戻るという弾性挙動が混在した物体である．粘弾性挙動が見られる物体は様々で，例としては粘土，ろう，溶岩などがあり，これらの物体は荷重，時間，温度変化に伴い変形する．

本論文では，固体，弾性体，粘弾性体，液体という 4 つの状態を統一的に扱いながら，融解現象により引き起こされる構造変化への高速な対応を目的とする．弾性体から粘弾性体，粘弾性体から液体までの運動は既存手法をうまく組み合わせて計算し，さらに本論文では特に固体から弾性体の間の運動を効率的に計算する手法を提案する．提案手法は，従来手法の長所をそのままにより幅広い物性を扱うことを目指す．

1.2 関連研究

1.2.1 融解シミュレーションに関する先行研究

Fujisawa ら [13] は，熱力学に基づいた氷の融解シミュレーション手法を提案したが，グリッドベースの手法であるため計算時間が非常に大きかった．Solenthaler ら [14] は，粒子法を用いた統一的な融解・凝固・結合・分離の手法を提案し，固体・液体・弾性体など幅広い物体をシミュレーションした．しかし，この手法では固体・液体・弾性体の運動計算方法が異なるため温度で物性が変化するような物体を表現できない．内田ら [15] は，融解後の水と氷のインタラクションを考慮した高速なシミュレーション手法を GPU を用いて実現した．この手法は非常に有用だが，物体の分離判定が必要で，また表現内容が固体と液体間に限定されている．仲宗根ら [16] は粘弾性体を対象とした高速な融解シミュレーションを提案した．固体粒子の属する四面体ごとに運動計算を行い，その個々の計算結果の総和を固体の運動として扱うことで，ボトムアップなアプローチから運動計算を行う．これにより，固体粒子が四面体から外れるだけで位置計算が行われなくなるため，文献 [15] のような粒子が所属している固体の判定が必要なくなる．また，四面体ごとに固体粒子を扱うというボトムアップアプローチのおかげで粒子間の関連性が減り，並列処理が可能となるなどのメリットが得られる．しかし，この手法では硬い物体の運動計算には計算時間がかかるため，作成可

能なアニメーションの表現の幅に限界があった．本論文では，仲宗根ら [16] の研究を拡張して変形の小さな硬い物体まで扱える融解シミュレーション手法を提案する．

1.2.2 弾性体シミュレーションに関する先行研究

弾性体の研究には，バネ—質点系と呼ばれる変形物体の弾性エネルギーを物理法則に基づいて計算する方法がある．この手法はとても実装しやすいため，布 [5] などの弾性体シミュレーションでは多くの場面で用いられるが，安定性に問題がある．これに対して Müller ら [6] の提案した Shape Matching 法（以下 SM 法）では，擬似的な弾性体の表現ではあるが安定かつ高速に挙動を計算することができる．この手法では物体の変形を点群で近似して表し，変形後の点群を元に戻るべき参照形状と照合して変形位置を求める．このような位置を基準とした運動計算のフレームワークを位置ベース法 [1] と呼ぶ．位置ベース法は従来のバネ—質点系や有限要素法と異なり複雑な方程式を解く必要が無く，安定して高速に計算できるためリアルタイムなアプリケーションに採用しやすい．一方で，軟らかい物体の表現は用意だが，硬い物体を表現しようとすると多くの反復計算が必要となり，計算コストが大きくなってしまう．

SM 法を拡張してより幅広い表現と高速なシミュレーションを可能とした手法が多く存在する．Rivers ら [7] は，SM 法における制約条件の計算単位（クラスタ）を重ね合わせたグリッド構造を導入し，運動計算を高速化し動きの質を向上させた．また Steinemann ら [8] は，階層構造を利用して計算点を統合することで処理の高速化と変形自由度の制限を提案している．Dizol ら [9] は，任意の三角形メッシュの頂点を計算点として SM 法を適用し，前処理として全頂点を網羅するパスを導入することで高速な総和計算を実現した．しかし，これらの手法は破断や分裂・結合といった幾何構造の変化を考慮しておらず，構造変化に伴う物体情報の更新に大きな計算コストがかかる．

Takamatsu ら [12] は，弾性体応力を正確に計算しない高速な粘弾性体アニメーション生成手法を提案した．基本となるアイディアは，液体の挙動と弾性体の挙動をそれぞれ計算しその結果を線形補間することにある．彼らの手法では，液体，粘弾性体，弾性体までを単純なパラメータで安定かつ統一的に制御することができる．また，物体の分裂，結合までを考慮したアニメーションをリアルタイムに作成できた．しかし，粒子の影響半径を大きくして制約条件に含める粒子数を増やすことで弾性体の硬さを調節しているため，アニメーションの表現内容にはまだ制限がある．井尻ら [10] は筋繊維の方向を考慮した重みの付け方や，クラスタのサイズと構造を最小にしつつ剛性を調整する方法として反復処理を提案している．しかし，単純な反復処理では平均計算時間が反復回数によって線形に増加するため，計算負荷の大きさが問題になる．また，SM 法の制約条件を修正した Oriented Particle 法 [11] は，粒子に姿勢の情報を加えることでより安定した運動計算が可能となっている．OP 法では紐，布，立体といった様々な構造の物体を統一的な計算手法で扱えるため，融解や破砕などによる物体の構造変化に左右されることなく運動計算できる．本論文では，クラスタのサイズが小さいままでも変形の伝播速度を向上させるために，OP 法に加えて計算次元の削減とソルバーの改良を行う．

第 2 章

提案手法

2.1 全体の流れ

提案手法の全体的な流れは以下のようになる．

1. 物体の領域分割：１つの物体に複数の制約条件を適用するための前処理（2.3.1 節)
2. 液体の運動計算：SPH 法を用いて液体粒子の速度・位置を更新（2.2 節）
3. 弾性体の運動計算：SM 法，OP 法を用いて弾性体粒子位置を更新（2.3 節）
4. 液体運動と固体運動の線形補間：最終的な粒子位置を決定（2.4 節）
5. 陰関数場の生成とレンダリング：シミュレーション結果の可視化 (2.5 節)

１は前処理としてオフラインで処理され，２〜５を各ステップごとにオンラインで行う．

2.2 液体の運動計算

液体の支配方程式として，以下の非圧縮粘性流体のナビエ・ストークス方程式を用いる．

$$\nabla \cdot \mathbf{u} = 0 \tag{2.1}$$

$$\rho(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}) = -\nabla p + \mu \nabla^2 \mathbf{u} + \mathbf{f} \tag{2.2}$$

ここで，t は時間，$\mathbf{u}$ は液体速度，μ は粘性係数，ρ は液体の密度，p は圧力，$\mathbf{f}$ は外力である．式 (2.1) は質量保存則である．実際の液体では，厳密には非圧縮ではないため，右辺は 0 にはならないが，一般的には液体が音速に比べて十分小さい場合は非圧縮とみなして 0 とする．式 (2.2) は運動量保存則である．左辺第１項は時間項であり，加速度を表す．左辺第２項は移流項であり，液体微小領域が自身の速度によって流れることを表す．ラグランジュ的方法の場合，この項は無視できる．右辺第１項は圧力項であり，流体が圧力の大きい位置から，小さい位置に流れることを表す．右辺第２項は粘性項であり，周りの速度との相互影響を表す．右辺第３項は外力項であり，重力や剛体からの作用力などが含まれる．

本論文では，液体のシミュレーションにラグランジュ的方法である粒子法を用いている．粒子法は粒子自体が液体を表しているため，粒子質量が変化しないかぎり質量保存性が常に保持される (質量保存式である式 (2.1) を解く必要がない)．そのため．グリッド法において

計算時間のかかる処理である液体表面の追跡の必要性がないことが利点である．また，粒子法では粒子の移動が液体の移流を表しているため，グリッド法のように移流項の計算を行う必要がない．

粒子法の代表的なものとして，MPS 法と SPH 法がある．MPS 法は半陰解法であり，非圧縮性を再現できるが計算コストが大きい．SPH は陽解法なので，非圧縮性を厳密に再現できないが，計算コストは小さい．そこで本論文では，リアルタイムシミュレーションに適した SPH を用いることにする．式 (2.2) を SPH を用いて離散化し，解く [21].

SPH では，座標 $\mathbf{x}$ での物理量 ϕ は式 (2.3) で示すように周囲の粒子 j のもつ物理量 ϕ_j の和で与えられる．

$$\phi(\mathbf{x}) = \sum_j m_j \frac{\phi_j}{\rho_j} W(\mathbf{x} - \mathbf{x}_j, h) \tag{2.3}$$

ここで，m_j，ρ_j，$\mathbf{x}_j$ はそれぞれ粒子 j の質量，密度，位置である．関数 W はカーネル関数で，基本的に距離 $||\mathbf{x} - \mathbf{x}_j||$ が小さくなるほど大きくなる関数が用いられる．また，探索範囲 h 外で 0 となる関数を用いることで，処理が局所的になるようにする．式（2.2）の右辺第 2 項で用いる物理量 ϕ の勾配は以下の式で表される．

$$\nabla\phi(\mathbf{x}) = \sum_j m_j \frac{\phi_j}{\rho_j} \nabla W((\mathbf{x} - \mathbf{x}_j), h) \tag{2.4}$$

また右辺第 2 項で用いられる物理量 ϕ のラプラシアンは以下の式で表される．

$$\nabla^2\phi(\mathbf{x}) = \sum_j m_j \frac{\phi_j}{\rho_j} \nabla^2 W((\mathbf{x} - \mathbf{x}_j), h) \tag{2.5}$$

外力項を $\mathbf{f}_i^{ext}$ とすると，式（2.2）から粒子の加速度は以下のように求まる．

$$\frac{d\mathbf{u_i}}{dt} = \frac{1}{\rho_i}(\mathbf{f}_i^{press} + \mathbf{f}_i^{vis} + \mathbf{f}_j^{ext}) \tag{2.6}$$

作用，反作用の法則に従うよう，粒子間の力を対称化することで圧力項 $\mathbf{f}_i^{press}$，粘性項 $\mathbf{f}_i^{vis}$ はそれぞれ次式のように求められる．

$$\mathbf{f}_i^{press} = -\sum_j m_j (\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}) \nabla W_{press}(\mathbf{x}_i - \mathbf{x}_j) \tag{2.7}$$

$$\mathbf{f}_i^{vis} = \mu \sum_j m_j \frac{\mathbf{v}_i + \mathbf{v}_j}{\rho_j} \nabla W_{vis}(\mathbf{x}_i - \mathbf{x}_j) \tag{2.8}$$

W には，Müller ら [21] が提案したカーネルを採用する．密度，圧力項，粘性項について，それぞれ Poly6 カーネル，Spiky カーネル，Viscosity カーネルである．圧力 p は，その液体の静止状態における密度を ρ_0 として以下の式で求める．

$$p = k_{press}(\rho - \rho_0) \tag{2.9}$$

k_{press} はガス定数で，これ大きくすることで，ほぼ非圧縮な液体として計算できるが，計算が不安定になるのに注意しなくてはならない．

2.3 弾性体の運動計算

固体や弾性体の運動計算の流れは以下のようになる．

1. クラスタによる領域分割：１つの物体に複数の制約条件を適用するための前処理（2.3.1 節)
2. 弾性体を計算するための制約条件：弾性体をシミュレーションするための制約条件の解を求める（2.3.2 節）
3. 反復法ソルバー：各制約条件の解を元に粒子位置を算出（2.3.3 節）

１は前処理としてオフラインで処理され，２, ３を各ステップごとにオンラインで行う．また，２, ３を各ステップ内で反復処理することで物体の硬さを調節する．以下では各項目についての詳細を説明する．

2.3.1 クラスタによる領域分割

粒子法を使ってシミュレーションを行うためには，粒子数，シミュレーション領域，各粒子の初期位置・速度・温度・熱量，液体か固体かという状態，といった情報を設定する必要がある．これらの情報は主にシミュレーションを行うユーザーが数値として設定する．例えばシミュレーション領域の大きさ（x,y,z 軸の幅），空気温度，タイムステップ幅，など単一あるいは数個のシミュレーション全体に関連するものは数値を指定するだけで設定できる．しかし手動で数値を指定するのが難しい情報もある．例えば，数千〜数十万個の粒子の配置場所を１つ１つ手作業で指定することは実質的に不可能である．そのため，位置指定の自動化は必須となる．本論文では，立方体の幅を数値で指定するとその立方体内部に敷き詰められた粒子の集合を自動で得られるようにしている．

以下では，特に弾性体の運動計算を行うために必要な前処理とその方法について説明する．

位置ベース粒子法によるシミュレーションでは，オブジェクトを粒子という要素で離散化する．弾性体の運動計算においては，2.3.2 節で説明する制約条件を物体全体に１つだけ適用すると線形変換で表される単純な変形アニメーションとなってしまう．そこで，１つの物体を複数の領域に分けてその領域ごとに制約条件を適用することで複雑な変形をシミュレーションする．それぞれの制約条件は，オブジェクトを構成する粒子を領域分割した小さな集合に適用され，この集合をクラスタと呼ぶ．クラスタの情報をユーザーが手作業で指定すると非常に手間がかかるため，本論文では粒子間の距離を用いて機械的にクラスタ分割を行う．

弾性体の運動計算のための前処理として，オブジェクトを構成する全粒子集合 P の要素それぞれに対してクラスタ C を作成する．以下の式 (2.10) で表すように，ある粒子 $p_i \in P$ を中心とした半径 d の球内にある全ての粒子集合 P_{C_i} を用いてクラスタ C_i を作成する．

$$P_{C_i} = \{p_j \mid ||\mathbf{x}_i - \mathbf{x}_j|| \leq d\} \subseteq P \tag{2.10}$$

最終的に全ての粒子に対してクラスタが１つずつ対応付けられ，弾性体の制約条件はそれぞれのクラスタで適用される．物体は，図 2.1 に示すように複数のクラスタを重ね合わせることで構成される．このクラスタ間の重複を大きくすることでも物体の硬さを調整することが可能だが，物体の構造が変化した際の分離判定にかかる計算コストが大きくなる．本論文では，クラスタの重なりは小さいままに反復回数を増やすことで硬さを調整する．

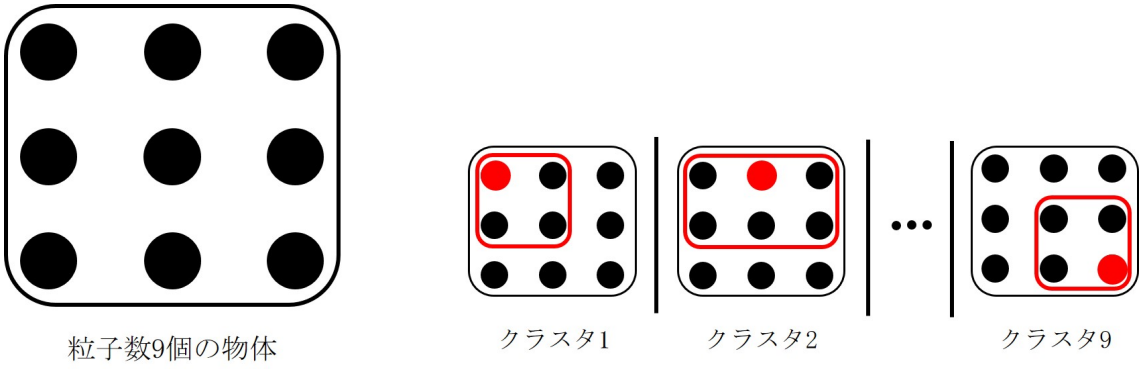


図 2.1: クラスタを重ね合わせて物体を構成

2.3.2 弾性体を計算するための制約条件

Shape Matching 法

SM 法 [6] は位置ベースな弾性体変形の計算手法である．物理法則に基づいた計算方法ではないためシミュレーションは擬似的なものになるが，高速に安定して計算できるのが特徴である．SM 法では，ある粒子で定義された形状の変形を考える際に，各粒子の位置と速度を元の形状に戻ろうとするように決定することで弾性体への運動を近似する．以下では線形変換を用いた SM 法の計算手順について解説する．

ある形状を構成する粒子の座標を $\mathbf{x}_k$,（$k = 1 \sim N$）とし，元の形状における粒子座標を $\mathbf{x}_k^0$ として表す．この初期形状を参照形状と呼ぶ．参照形状が外力により変形して，形状を構成する粒子の位置が図 2.2 右の白点で表した位置に移動したとする．

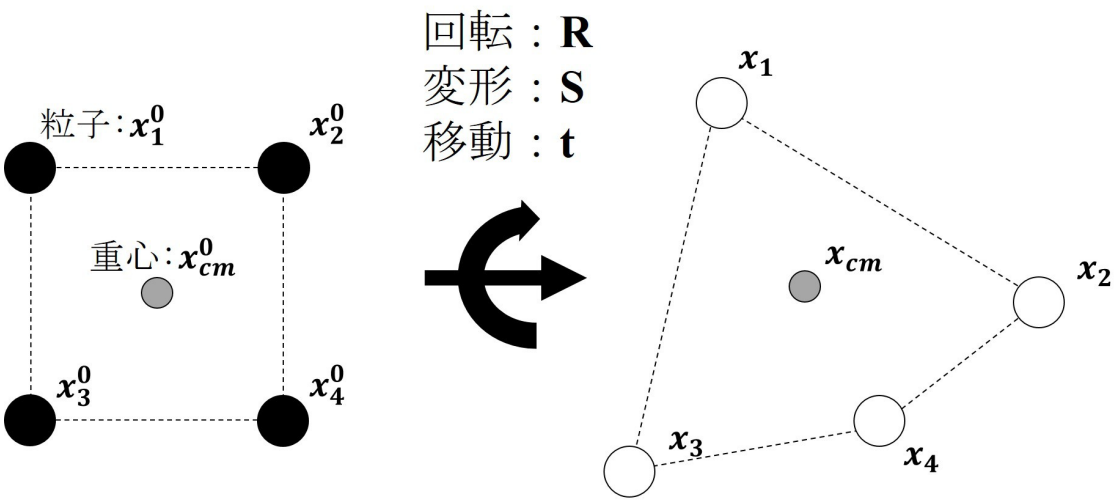


図 2.2: 粒子 x_k^0 が x_k へ移動

このとき，各粒子の移動は形状全体あるいはクラスタ全体の回転，平行移動，せん断（ひずみ）の 3 つの成分で構成され，以下の式（2.11）で表すことができる．

$$\mathbf{x}_k = \mathbf{RS}(\mathbf{x}_k^0 - \mathbf{x}_{cm}^0) + \mathbf{t} \tag{2.11}$$

ここで $\mathbf{R}$ は回転行列，$\mathbf{S}$ はひずみを表す行列，$\mathbf{t}$ は平行移動ベクトル，$\mathbf{x}_{cm}^0$ は参照形状の重心であり，右辺第 1 項の括弧内は初期座標の重心を原点とした相対座標の変換を表している．ゴムのような弾性体は，変形しても元の形状をに戻ろうとするが物体全体の平行移動や回転は発生する．そこで，有限要素法などの物理法則に基づく手法のように変形の行列 $\mathbf{S}$ を直接求めて変形を適用していくのではなく，変形がなかった場合，つまり回転と平行移動のみが発生した場合の粒子位置を基準として変形させることで弾性体変形を近似する．

手順としてはまず，式（2.11）の $\mathbf{R}$ と $\mathbf{t}$ を $\mathbf{x}_k^0$ と $\mathbf{x}_{cm}$ を用いて推定する．そうすることで変形が行われなかった場合の粒子位置を得られる．そして変形 $\mathbf{S}$ や弾性体的な挙動は別の方法で実装することで，移動や回転の推定と変形の処理を分けて行う．この推定や処理の分割によって物理的な正確さは低くなるが，高速に安定して処理を行うことができるようになる．推定して得られた $\mathbf{R}$ と $\mathbf{t}$ を参照形状 $\mathbf{x}_k^0$ へ適用すると，式（2.11）は以下の式（2.12）のように表せる．

$$\mathbf{g}_k = \mathbf{R}(\mathbf{x}_k^0 - \mathbf{x}_{cm}^0) + \mathbf{t} \tag{2.12}$$

$\mathbf{g}_k$ は参照形状 $\mathbf{x}_k^0$ へ回転と移動が適用されて得られる目標位置であり，更新位置ではない．図 2.3 のように $\mathbf{g}_k$ を目標位置として $\mathbf{x}_k$ を移動させて得られるのが更新位置となる．以上が SM 法の全体像となる．

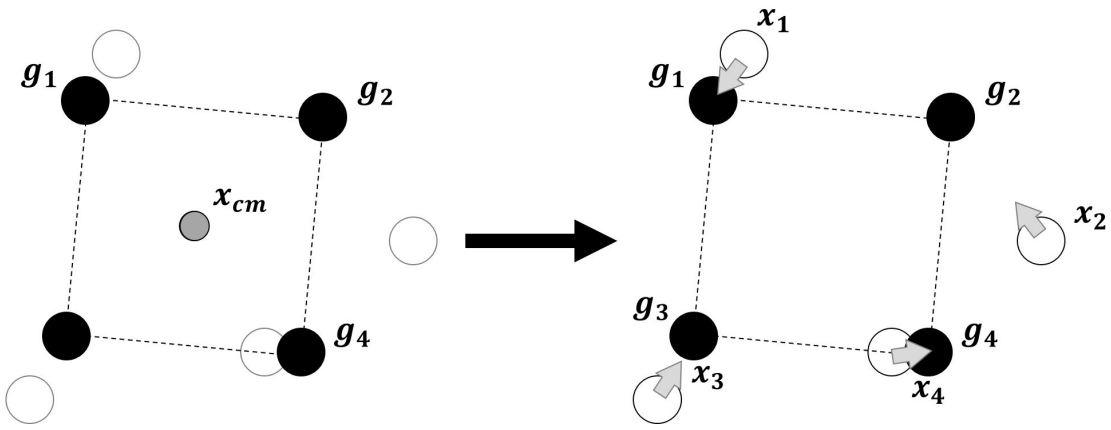


図 2.3: 並行移動と回転を推定して得た目標位置 g_k へ x_k を移動

必要となる $\mathbf{R}$ と $\mathbf{t}$ の推定問題は，以下の式（2.13）の最小化問題として解くことができる．

$$e = \sum_{k=1}^{N} m_k |\mathbf{g}_k - \mathbf{x}_k|^2 = \sum_{k}^{N} m_k |\mathbf{R}(\mathbf{x}_k^0 - \mathbf{x}_{cm}) + \mathbf{t} - \mathbf{x}_k|^2 \tag{2.13}$$

最小化問題を解くために，ここでは最小二乗法を用いる．式（2.13）は，目標位置 $\mathbf{g}_k$ と変形後の位置 $\mathbf{x}_k$ の差の重み m_k 付き最小二乗誤差を表しており，最小二乗法により e を最小

とする $\mathbf{R}$ と $\mathbf{t}$ を求める．この重み m_k を大きくした点は移動しにくくなるため，固定点や質量の大きい点を表すために用いる．平行移動ベクトル $\mathbf{t}$ に関しては，クラスタの重心位置が移動したと考えればよく，以下の式（2.14）で求められる．

$$\mathbf{t} = \frac{\sum_{k=1}^{N} m_k \mathbf{x}_k}{\sum_{k=1}^{N} m_k} = \mathbf{x}_{cm} \tag{2.14}$$

それぞれの粒子を式（2.14）で重心を原点とした相対座標へ変換し，変形後の座標をそれぞれ $\mathbf{p}_k = \mathbf{x}_k - \mathbf{x}_{cm}$，$\mathbf{q}_k = \mathbf{x}_k^0 - \mathbf{x}_{cm}^0$ とする．また，$\mathbf{A} = \mathbf{RS}$ と回転・変形を考慮したモーメント行列 $\mathbf{A}$ を仮定すると，式（2.13）は式（2.15）のように変形できる．

$$e = \sum_{k=1}^{N} m_k |\mathbf{A}\mathbf{q}_k - \mathbf{p}_k|^2 \tag{2.15}$$

最小二乗法で行列 $\mathbf{A}$ を求めるには，e を行列 $\mathbf{A}$ の各要素 a_{ij} で微分したものが 0 となるとして解けばよい．よって以下の式（2.16）が求められる．

$$\frac{\partial e}{\partial a_{ij}} = \sum_{k=1}^{N} 2m_k(\mathbf{A}\mathbf{q}_k - \mathbf{p}_k)\mathbf{q}_k^T = 0 \tag{2.16}$$

式（2.16）を $\mathbf{A}$ について解くと，以下の式（2.17）が得られる．

$$\mathbf{A} = (\sum_{k=1}^{N} m_k \mathbf{p_k} \mathbf{q_k^T})(\sum_{k=1}^{N} m_k \mathbf{q_k} \mathbf{q_k^T})^{-1} = \mathbf{A}_{pq}\mathbf{A}_{qq}^{-1} \tag{2.17}$$

$\mathbf{A}qq$ は $\mathbf{q}_k\mathbf{q}_k^T$ より対称行列で，スケーリング（拡大縮小）を表している．$\mathbf{A}pq$ は変形 $\mathbf{S}$ と回転 $\mathbf{R}$ が含まれた変形勾配テンソルとなっている．変形勾配テンソルは極分解により回転成分とその他に分けることができる．極分解には右極分解と左極分解の二つがあるが，ここでは右極分解を行う．極分解では $\mathbf{A}_{pq} = \mathbf{RS}$ として表し，以下の式（2.17）,（2.18）で $\mathbf{R}$ と $\mathbf{S}$ に分解する．

$$\mathbf{S} = (\mathbf{A}^T\mathbf{A})^{\frac{1}{2}} \tag{2.18}$$

$$\mathbf{R} = \mathbf{A}\mathbf{S}^{-1} \tag{2.19}$$

行列の 1/2 乗を求めるために，まずは以下の式（2.20）のように行列を対角化した後に 2 乗する．

$$\mathbf{S}^2 = \mathbf{A}^T\mathbf{A} = (\mathbf{U}^{-1}\Lambda\mathbf{U})(\mathbf{U}^{-1}\Lambda\mathbf{U}) = \mathbf{U}^{-1}\Lambda^2\mathbf{U} \tag{2.20}$$

ここで $\mathbf{U}$ は $\mathbf{S}^2$ の固有ベクトルを並べた直交行列，Λ は $\mathbf{S}^2$ の固有値を対角要素に持つ対角行列である．Λ は対角行列であるため，対角要素を 2 乗すれば Λ^2 が求められる．また U は直行行列なので $\mathbf{U}^{-1} = \mathbf{U}^T$ となる．これより式（2.20）を変形して以下の式（2.21）が得られ，$\mathbf{S}$ が求められる．

$$\mathbf{S} = (\mathbf{A}^T\mathbf{A})^{\frac{1}{2}} = \mathbf{U}^{-1}\sqrt{\Lambda}\mathbf{U} = \mathbf{U}^T \begin{pmatrix} \sqrt{\Lambda} & & & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & & & \sqrt{\Lambda} \end{pmatrix} \mathbf{U} \tag{2.21}$$

以上の極分解の式（2.18）〜（2.21）を $\mathbf{A}_{pq}$ に適用する手順は以下になる．

1. $\mathbf{S}^2 = (\mathbf{A}_{pq}^T \mathbf{A}_{pq})$ の固有値 λ_1〜λ_n，固有ベクトル μ_1〜μ_n を求める
2. 固有ベクトルから直交行列 $\mathbf{U}$ を，固有値の平方根から行列 Λ を作成
3. 式（2.21）を計算
4. 式（2.19）を計算

$\mathbf{S}$ は既に対角化しているため，$\mathbf{S}^{-1}$ は以下の式で求められる．

$$\mathbf{S}^{-1} = \mathbf{U}^{\mathbf{T}} \begin{pmatrix} \sqrt{\frac{1}{\Lambda}} & & & 0 \\ & \ddots & & \\ & & \ddots & \\ 0 & & & \sqrt{\frac{1}{\Lambda}} \end{pmatrix} \mathbf{U} \tag{2.22}$$

今回は $\mathbf{A}_{qq}$ を計算していないが，体積が保存された変形を行う場合は $det(\mathbf{A}) = 1$ を満たす必要があるため，$\mathbf{A}_{qq}$ も求める必要がある．なお，$\mathbf{A}_{qq}$ は参照形状の情報のみで計算できるので，計算は初期化が行われた際の 1 度でよい．また，大規模な変形が原因で参照形状が反転した場合，$det(\mathbf{A}_{pq}) < 0$ となる．反転した状態でも形状は保たれるが，不都合がある場合は反転を許可しないために $det(\mathbf{A}_{pq}) > 0$ を満たすよう行列の要素入れ替えればよい．

最終的に，各粒子の速度と位置は以下の式で更新する．

$$\mathbf{v}_i^{n+1} = \mathbf{v}_i^n + \alpha \frac{\mathbf{g}_i^n - \mathbf{x}_i^n}{\Delta t} + \frac{\mathbf{F}_{external}}{m_i} \Delta t \tag{2.23}$$

$$\mathbf{x}_i^{n+1} = \mathbf{x}_i^n + \mathbf{v}_i^{n+1} \Delta t \tag{2.24}$$

式（2.23）における変数 $\alpha \in [0, 1]$ は，物体の剛性を調整するパラメータである．$\alpha \geq 1$ の場合，$\alpha \frac{\mathbf{g}_i^n - \mathbf{x}_i^n}{\Delta t}$ によって現在の粒子位置によって速度が変化する．そのため外力が存在すれば変形が見られ，外力がなくなれば元の形に戻る，という弾性体の振る舞いをする．$\alpha = 1$ の場合，$\mathbf{v}_i^n = \frac{\mathbf{x}_i^n}{\Delta t}$ であるため常に元の形状である $\mathbf{g}_i^n$ が保たれ，剛体の振る舞いをする．

Oriented Particle 法

SM 法を用いた運動計算では，細長い物体などで望んだ弾性体表現を得られない場合がある．これは，物体を構成するクラスタ間で重複している粒子が少ないために起こる．粒子の重複数が少ないと，クラスタ間における変形が伝播しにくくなるため物体の構造によっては図 2.4 左のように元の形状に戻らず変形したままになってしまう．

変形の伝播速度を向上させるため Müller ら [11] の手法を用いて制約条件を修正する．Müller らの手法では，球ではなく姿勢情報を考慮した楕円体粒子 (Oriented Particle) を用いて運動計算を行う（以下 OP 法）．SM 法ではクラスタに含まれる粒子が同一平面上や同一直線上に近い場合，式 (2.17) で得られる $\mathbf{A}$ が特異行列になりやすいため極分解が不安定になる．この問題を解決するために，位置情報だけでなく姿勢情報を加えた粒子を用いてモーメント行列 $\mathbf{A}$ を再定義することで，紐・布・立体のような 1 次元から 3 次元の弾性体までを安定かつ統一的に扱うことができる．これに加えて，姿勢と角速度から粒子の向きや回転方向が

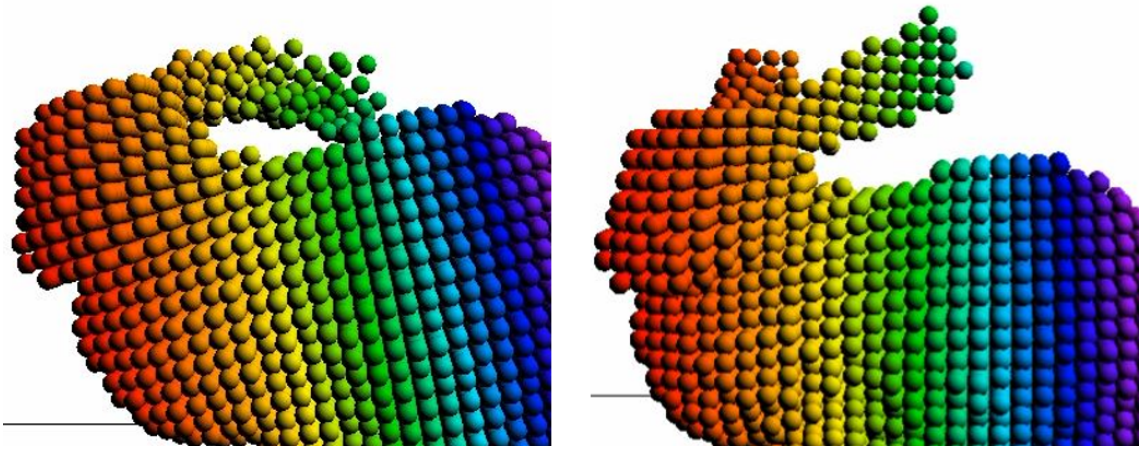


図 2.4: SM 法 (左) と OP 法 (右) の比較．SM 法は反復 20 回，OP 法は反復 10 回，サンプリング数は共通して 832 個で計算時間は約 30fps．耳の部分に差が出ている．

分かるため，より正確に物体の形状を近似可能かつ高精度な衝突応答が可能な楕円体粒子を用いることができる．以下では，OP 法による制約条件の修正方法を具体的に説明する．

SM 法ではクラスタに含まれる粒子の位置情報 $\mathbf{x}_k, \mathbf{x}_k^0$ からモーメント行列 $\mathbf{A}$ を定義するが，OP 法では粒子の回転成分も考慮してこのモーメント行列 $\mathbf{A}$ を定義する．本論文で用いる粒子ベース弾性体の運動計算では，粒子それぞれにクラスタが 1 対 1 で定義されているのでモーメント行列 $\mathbf{A}$ は粒子の数だけ存在する．しかし式 (2.17) からわかるように，クラスタのモーメント行列はそのクラスタの重心を用いて定義されるため，単純に近傍のクラスタの $\mathbf{A}_{pq}$ を足し合わせて求めることはできない．そこで，Rivers ら [7] の提案したように以下の式で $\mathbf{A}_{pq}$ を修正する．

$$\begin{aligned}\mathbf{A}_{pq} &= \sum_{k=1}^{N} m_k(\mathbf{p_k q_k^T}) = \sum_{k=1}^{N} m_k(\mathbf{x}_k - \mathbf{x}_{cm})(\mathbf{x}_k^0 - \mathbf{x}_{cm}^0)^T \\ &= \sum_{k=1}^{N} m_k \mathbf{x}_k \mathbf{x}_k^{0T} - M\mathbf{x}_{cm}\mathbf{x}_{cm}^{0\ T}, \quad M = \sum_{k=1}^{N} m_k \end{aligned} \tag{2.25}$$

クラスタの重心を総和計算から取り除けたので，$\mathbf{A}_{pq}$ に粒子のモーメント行列 $\mathbf{A}_{elipsoid}$ を加え以下の式（2.26）で再定義する．

$$\mathbf{A}_{pq} = \sum_{k=1}^{N}(\mathbf{A}_{elipsoid_k} + m_k \mathbf{x}_k \mathbf{x}_k^{0T}) - M\mathbf{x}_{cm}\mathbf{x}_{cm}^{0\ T} \tag{2.26}$$

また楕円体粒子のモーメント行列は粒子の回転行列 $\mathbf{L}$，楕円の各軸の長さ a, b, c を用いて以下の式（2.27）で表す．

$$\mathbf{A}_{elipsoid} = \frac{1}{5}m \begin{pmatrix} a^2 & 0 & 0 \\ 0 & b^2 & 0 \\ 0 & 0 & c^2 \end{pmatrix} \mathbf{L} \tag{2.27}$$

ただし，本論文では楕円体粒子ではなく球体粒子を用いて運動計算を行うため，$a = b = c = 1$ とする．

次に，各ステップにおける粒子の姿勢情報の更新方法について説明する．粒子の姿勢情報をクォータニオン $\mathbf{q}$ で表し，また角速度を $\boldsymbol{\omega}$ とする．各シミュレーションステップの初めには，角速度 $\boldsymbol{\omega}$ を用いて以下の式（2.29）で姿勢情報 $\mathbf{q}_{predict}$ を推定する．

$$\mathbf{q}_{predict}^{n+1} \leftarrow \left[\frac{\boldsymbol{\omega}}{|\boldsymbol{\omega}|}\sin\left(\frac{|\boldsymbol{\omega}|\Delta t}{2}\right), \cos\left(\frac{|\boldsymbol{\omega}|\Delta t}{2}\right)\right]\mathbf{q}^n \tag{2.28}$$

安定性のために，$|\boldsymbol{\omega}| < \epsilon$ の場合は更新せずにそのまま $\mathbf{q}^n$ を用いる．この推定姿勢 $\mathbf{q}_{predict}^{n+1}$ とステップ最初の姿勢 $\mathbf{q}^n$ から，$\mathbf{r}_p = \mathbf{q}_{predict}^{n+1}(\mathbf{q}^n)^{-1}$ を想定することができる．このクォータニオン $\mathbf{r}_p$ を回転行列 $\mathbf{L}$ に変換し，式（2.27）からモーメント行列 $\mathbf{A}_{elipsoid}$ を算出する．最終的に，各粒子で得られたモーメント行列 $\mathbf{A}_{elipsoid}$ を用いて，各クラスタにて極分解を行う．

クラスタでの運動計算後に粒子の姿勢情報を更新する．本論文では，$\mathbf{q}_R$ をクラスタの回転行列 $\mathbf{R}$ をクォータニオンに変換したもの，$\mathbf{q}_0$ を初期の姿勢を表すクォータニオン，$\mathrm{axis}(\mathbf{q})$ をクォータニオンの方向，$\mathrm{angle}(\mathbf{q})$ をクォータニオンの角度を得る関数として，角速度と姿勢を以下の式で更新する．

$$\mathbf{q}^{n+1} \leftarrow \mathbf{q}_R\mathbf{q}_0 \tag{2.29}$$

$$\boldsymbol{\omega}^{n+1} \leftarrow \mathrm{axis}(\mathbf{q}^{\mathrm{n+1}}(\mathbf{q}^{\mathrm{n}})^{-1}) \cdot \mathrm{angle}(\mathbf{q}^{\mathrm{n+1}}(\mathbf{q}^{\mathrm{n}})^{-1})/\Delta\mathrm{t} \tag{2.30}$$

安定性のために，$|\mathrm{angle}(\mathbf{q}^{\mathrm{n+1}}(\mathbf{q}^{\mathrm{n}})^{-1})| < \epsilon$ の場合は $\mathrm{angle}(\mathbf{q}^{\mathrm{n+1}}(\mathbf{q}^{\mathrm{n}})^{-1}) = 0$ とする．また，回転成分には $\mathbf{r} = \mathbf{q}^{n+1}(\mathbf{q}^n)^{-1}$ とその共役クォータニオン $-\mathbf{r}$ の二種類が存在する．この 2 つは同一の回転を示すが，計算には常に短い道筋を表すクォータニオンを使う必要があるため $\mathbf{r}.w < 0$ の場合は $-\mathbf{r}$ を用いる．

2.3.3 反復法ソルバー

粒子を用いた位置ベース法 [6]（以下，位置ベース粒子法）では，物体は粒子によって離散化され粒子間それぞれの関係は幾何学的な制約条件で表される．複数の粒子に対して 1 つの制約条件が課され，この制約条件を一次式と見立てることで物体全体の運動計算を連立一次方程式のようにして解く．

弾性体の運動計算においては，2.3.2 節で説明した式（2.13）が制約条件にあたり，この制約条件は 2.3.1 節で定義した各クラスタに適用される．クラスタは粒子の集合であり近傍関係で定義されるため，近傍のクラスタ間には少なくとも 1 つの粒子が重複して所属する．そのため 1 つの粒子に対して同じ種類の制約条件が複数個適用され，制約条件ごとに仮の解が得られる．この複数の制約条件を解いた結果から最終的な結果を算出するために，位置ベース粒子法では反復法を用いたいくつかのソルバーが提案されている．

連立一次方程式の解法のひとつである反復法では，まず連立一次方程式の初期解を適当に選択し，これを出発点とする．次に，方程式を満たすように解を修正することで新たな近似解を作成する．この解を修正し近似解を得るという操作を何度も繰り返し，あるステップにおいて得られた近似解が真の解に十分近づいたら処理を終了する．反復計算の各ステップで

は，それまでに得られた近似解を駆使してより良い近似解を生成する工夫が施されている．近似解の生成方法は様々であり，いくつかの手法が提案されている．

位置ベース粒子法における反復法とは，それぞれの制約条件を解いて仮の位置ベクトルを得る，という操作を 1 フレーム内に何度も繰り返す方法である．位置ベース粒子法における反復法は，連立一次方程式の反復法とは異なり解が収束する条件を定義するのは困難である．しかし，大抵の状況において解は収束し，振動する場合は微小な移動を無視することで安定したアニメーションが得られる．以下では，位置ベース粒子法における反復法ソルバーについて説明する．

Jacobi ソルバー

反復法のひとつである Jacobi ソルバーでは，新たな近似解を作成するために前ステップの近似解を用いる．Jacobi ソルバーを位置ベース粒子法に適用すると，前フレームの位置ベクトルが初期解に該当する．また新たな位置座標は，反復計算内における前ステップの位置ベクトルを用いて制約条件を解くことで得られる．

弾性体の運動計算における Jacobi ソルバーでは，反復処理 $m+1$ 回目における粒子 p_i の新たな位置ベクトル $\mathbf{x}_i^{m+1}$ を以下の式（2.31）で計算する．

$$\mathbf{x}_i^{m+1} = \frac{1}{|C_i|} \sum_{k \in C_i} \mathbf{R}_k^{m+1} (\mathbf{x}_k^0 - \mathbf{x}_{k,cm}^0) + \mathbf{x}_{k,cm}^m \tag{2.31}$$

ここで粒子 p_i を含むクラスタの集合を C_i，その集合に含まれるクラスタ数を $|C_i|$ とした．同様に，反復処理 $m+2$ 回目の位置ベクトルは $\mathbf{x}_i^{m+1}$ を用いて求める．Jacobi ソルバーでは，各制約条件は変形後の位置ベクトルを用いて独立して運動計算できるため，最大限に並列処理できるというメリットがある．

Gauss-Seidel ソルバー

Gauss-Seidel ソルバー（以下 GS ソルバー）では，t 番目の制約条件を解く際に $1, ..., t-1$ 番目の制約条件を解いて得られた最新の解を用いて計算を行う．これは，ある制約条件を解いて得られた仮の解が収束に近づくなら部分的に精度のよい解が得られるため，全体の解の収束が早まるという仮定に基づいている．位置ベース粒子法において GS ソルバーを用いる場合，それぞれの制約条件を 1 つずつ連続して解き，制約条件を解く過程で得られた最新の位置座標を用いて運動計算を行う．

弾性体の運動計算における GS ソルバーについて説明する．まず，クラスタの回転行列 $\mathbf{R}$ を求める関数を Rotate(x) と置く．関数 Rotate(x) の引数はクラスタに含まれる粒子それぞれの位置ベクトルであり，$\mathrm{Rotate}(\mathrm{x_1^m}, \mathrm{x_2^m}, ..., \mathrm{x_k^m})$ のように表すとする．上記の Jacobi ソルバーでは，クラスタの回転行列を $\mathbf{R}^{m+1} = \mathrm{Rotate}(\mathrm{x_1^m}, \mathrm{x_2^m}, ..., \mathrm{x_k^m})$ と表せる．これに対して GS ソルバーでは，既に得られた m+1 フレーム目の最新の解である粒子位置を用いて $\mathbf{R}^{m+1} = \mathrm{Rotate}(\mathrm{x_1^{m+1}}, ..., \mathrm{x_t^{m+1}}, \mathrm{x_{t+1}^m}, ..., \mathrm{x_k^m})$ と表せる．同様に，クラスタの重心 $\mathbf{x}_{k,cm}^m$ を求める関数を Center(x) として，クラスタに含まれる位置ベクトルを引数に取り $\mathrm{Center}(\mathrm{x_1^m}, \mathrm{x_2^m}, \cdots \mathrm{x_k^m})$

と表す．以上を踏まえて，GS ソルバーにおける位置ベクトルは以下の式（2.32）で表される．

$$\mathbf{x}_i^{m+1} = \frac{1}{|C_i|}\sum_{k\in C_i} \mathrm{Rotate}(\mathrm{x}_1^{\mathrm{m+1}}, ..., \mathrm{x}_{\mathrm{t}}^{\mathrm{m+1}}, \mathrm{x}_{\mathrm{t+1}}^{\mathrm{m}}, ..., \mathrm{x}_{\mathrm{k}}^{\mathrm{m}})(\mathbf{x}_{\mathrm{k}}^{0} - \mathbf{x}_{\mathrm{k,cm}}^{0}) + \mathrm{Center}(\mathrm{x}_1^{\mathrm{m+1}}, ..., \mathrm{x}_{\mathrm{t}}^{\mathrm{m+1}}, \mathrm{x}_{\mathrm{t+1}}^{\mathrm{m}}, ..., \mathrm{x}_{\mathrm{k}}^{\mathrm{m}}) \tag{2.32}$$

上記の説明は，1 番目の制約条件から n 番目の制約条件までを順次解いた場合の表現であり，実際のシミュレーションにおいては制約条件を解く順番による運動計算結果の偏りを防ぐためランダムな順番で運動計算を行う．

GS ソルバーは Jacobi ソルバーに比べて解の収束が早く，反復回数を抑えることができる．しかしそれぞれの制約条件を直列に計算する必要があるため，並列処理による高速化が困難である．そのため，GS ソルバーは計算要素が少ない場合にしか用いることができない．

Hybrid ソルバー

コンピュータグラフィックスを用いたアニメーション作成において，製作期間の短縮のために GPU を用いた並列処理は必要不可欠になっている．位置ベース粒子法を GS ソルバーで並列実装し，各スレッドで制約条件を解く場合について考える．仮に一つの粒子に異なる二つの制約条件が適用される場合，あるスレッドで制約条件を解いて得られた位置情報を即座に反映してはならない．このとき他のスレッドが位置情報のデータを読み込んでいるとすると，データの読み込みの途中で内容の変更が起こり情報の一貫性が失われてしまうためである．そのため位置情報へのアクセスを Atomic 操作で実装しなくてはならないが，Atomic 操作を多様すると著しく性能低下が起こってしまう．

この問題を解決するために，制約条件をグループまたはフェーズに分けて制約条件を解く．制約条件の求解結果を粒子に反映する際に衝突が起こらないように，それぞれの制約条件の間で共有する粒子が存在しないようグループに分ける．このグループごとに制約条件を並列に解くことを繰り返し，最終的に全ての制約条件を解く．Fratarcangeli ら [2] は，Position Based Dynamics[6] の並列計算手法として Jacobi ソルバーと GS ソルバーを組み合わせた Hybrid ソルバーを提案した．彼らの手法では，物体の構造変化をリアルタイムに扱うことを考慮していないものの弾性体の運動計算において高い並列性と計算精度を実現している．以下では，構造変化を考慮しつつ制約条件を並列処理するためのグループ分けと計算手法について説明する．

Hybrid ソルバーの基本的なアイディアは，Jacobi ソルバーの並列性と GS ソルバーの逐次性を組み合わせてそれぞれの長所を生かすことにある．はじめに，制約条件をそのラベル番号の偶奇性でグループ分けする．このとき，グループ内の制約条件の間で共有する粒子の有無を考慮していないため，並列 GS ソルバーのように独立した制約条件を並列計算できない．そこで，ある粒子を共有する複数の制約条件の解から最終的なひとつの解を得るために，Jacobi ソルバーを用いる．まず偶数グループの制約条件を Jacobi ソルバーで解き，得られた結果を最新の結果として粒子位置を更新し，続いて奇数グループの制約条件を解く．このように，並列性と逐次性のバランスを取りながら運動計算を行うのが Hybrid ソルバーである．上記のようなラベルの偶奇性を用いてグループに分ける手法は，数値計算の分野において Red-Black Gauss-Seidel ソルバーという名前で知られている．

重み係数を用いた Jacobi 法

Macklin ら [3] は運動計算の収束を早めるために，Jacobi 法に加速緩和係数 w を導入した SOR 法をソルバーに用いている．Macklin ら [3] らの方法ではパラメータ w を $1 < w < 2$ の範囲で定数として指定していた．井尻ら [10] は，筋肉の異方性硬さを表現するために方向ベクトルを用いて重み係数を計算した．本論文では，クラスタベースオブジェクトにおける変形の伝播を高速化するための新たな加速緩和係数 w を提案する．提案手法では Lattice Shape Matching[7] のように各クラスタにおける計算結果の単純な平均ではなく，変形の収束を速めるために重みを付ける．以下では，粒子 p_i の最終的な位置 $\mathbf{x_i}$ を重み係数 w_i を用いた重み付き和で求める提案手法について説明する．まず，粒子 p_i に対応するクラスタ C_i の変形量 d_i を，クラスタ C_i 内の各粒子 p_j の計算結果 $\mathbf{g}_j^i$ を用いて以下の式 (3.1) で定義する．

$$d_i = \sum_{j \in C_i} |\mathbf{g}_j^i - \mathbf{x}_j^i| \tag{2.33}$$

そして変形量 d_i と粒子 p_i を含む各クラスタの変形量の総和によって粒子 p_i の重み係数 w_i を求める．

$$w_i = \mathbf{W}(d_i) = \frac{d_i}{\sum_j d_j} \tag{2.34}$$

重み係数を用いて粒子 p_i を含む各クラスタ C_j における各粒子 p_i の計算結果 $\mathbf{g}_i^j$ を足し合わせることで，全ての頂点位置を決定する．(LSM では $1/|C_i| \sum_j \mathbf{g}_i^j$)

$$\mathbf{x}_i = \sum_j w_j \mathbf{g}_i^j \tag{2.35}$$

この方法では，大きな変形が起こったクラスタの重みを大きくすることで，クラスタ間の変形の伝搬速度を向上させている．頂点数 4913 個の立方体を用いて，反復回数を固定して単純な平均と重み付き和による弾性体の落下アニメーションの比較を行った．図 2.5 の実験結果からわかるように，重み付き和を用いた方が変形が少なくなっていることがわかる．ま

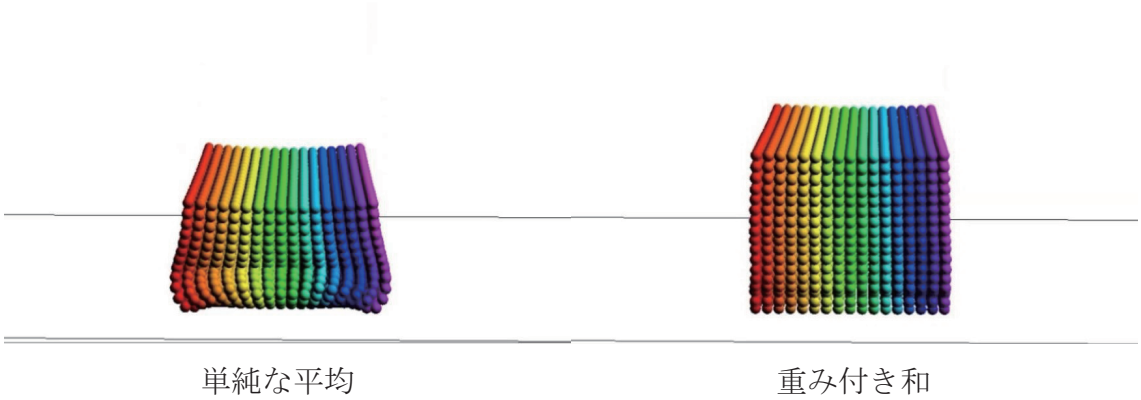


図 2.5: 単純な平均と重み付き和の比較

た，1 ステップにおける変形量が許容量以下になるまで運動計算を反復し，その反復回数をそれぞれの手法で比較する実験を行った．図 2.6 は実験結果で，横軸が時間，縦軸が反復回

数を示している．LSM[7] と提案手法で実験を行ったところ，同じ結果を得るまでに反復した回数は提案手法の方が少ない．これより重み付き和を用いることで変形収束が高速化できたことがわかる．

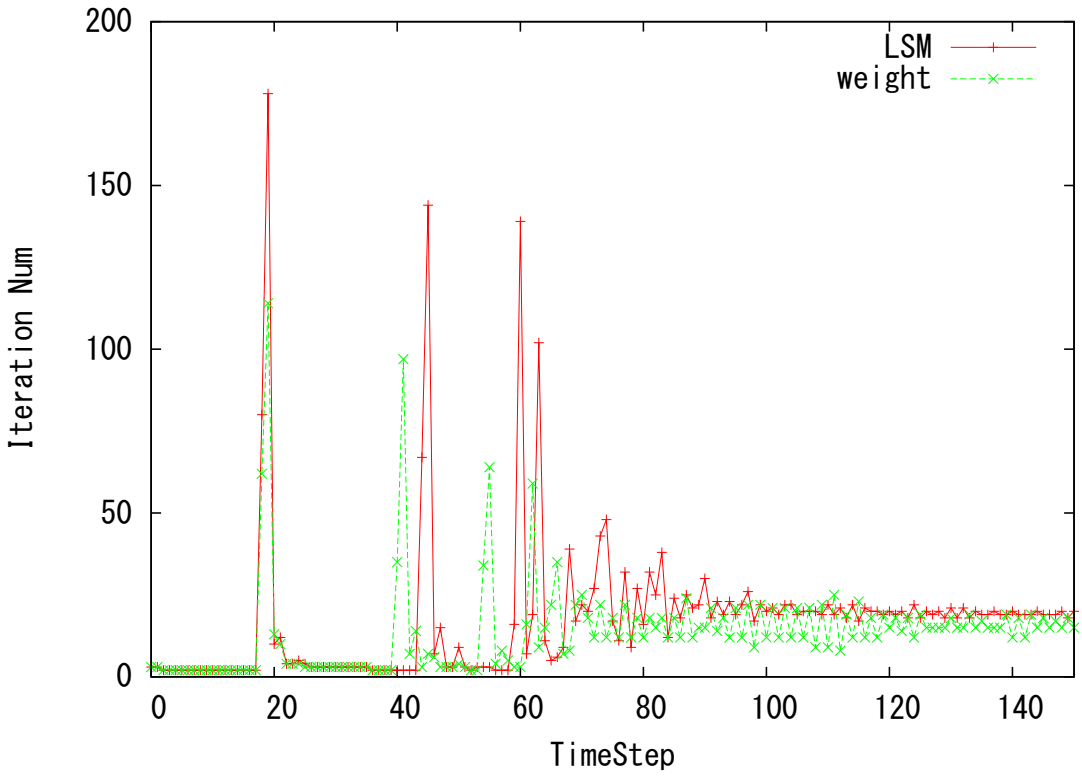


図 2.6: 許容変形量が小さい場合の反復回数を比較

ただし，この重み付き和を用いた反復処理では反復回数が少ないと境界付近で物体が振動してしまう．これは重み係数を変形量とその総和のみで単純に決めていることが原因で，重み付き和は軟らかい物体の運動計算に用いるべきではない．また，変形収束を速めて反復回数を減らすことができても重み係数の計算に時間がかかっては意味が無いため，提案手法では式 (2.34) のようなコストの低い単純な重み関数を用いた．式（3.1）の変形量 d_i は，式（2.23）内で確認できるように弾性体の運動計算過程で使用する値なので小さな計算コストで求められる．

2.3.4　クラスタのサンプリング

本論文では，物体の硬さの調整（剛性調整）を運動計算処理の反復によって行う．図 2.7 に反復処理による剛性調整の概要を示す．移動した粒子 (図 2.7 左の白い点) を含むクラスタ (図 2.7 右上の赤い四角形) の変形は，運動計算を反復することで隣接したクラスタ (図 2.7 右下の赤い四角形) に伝播する．提案手法ではこのような変形の伝播を繰り返すことで局所的な運動を全体へ伝え，最終的に変形が収束することで硬い物体が表現される．

井尻ら [10] は運動計算を反復することで剛性を調節していたが，計算量の問題から反復回数に制限があった．提案手法では，ランダムに選択されたクラスタのみで運動計算することで 1 回毎の運動計算を高速化し，反復回数が増加した場合でも計算負荷を抑える．これにより，1 ステップあたりの反復回数の上限が広がるため，より硬い物体の表現が可能となる．選択するクラスタに偏りが発生しないように Poisson Disk Sampling[20] を用いてサンプリ

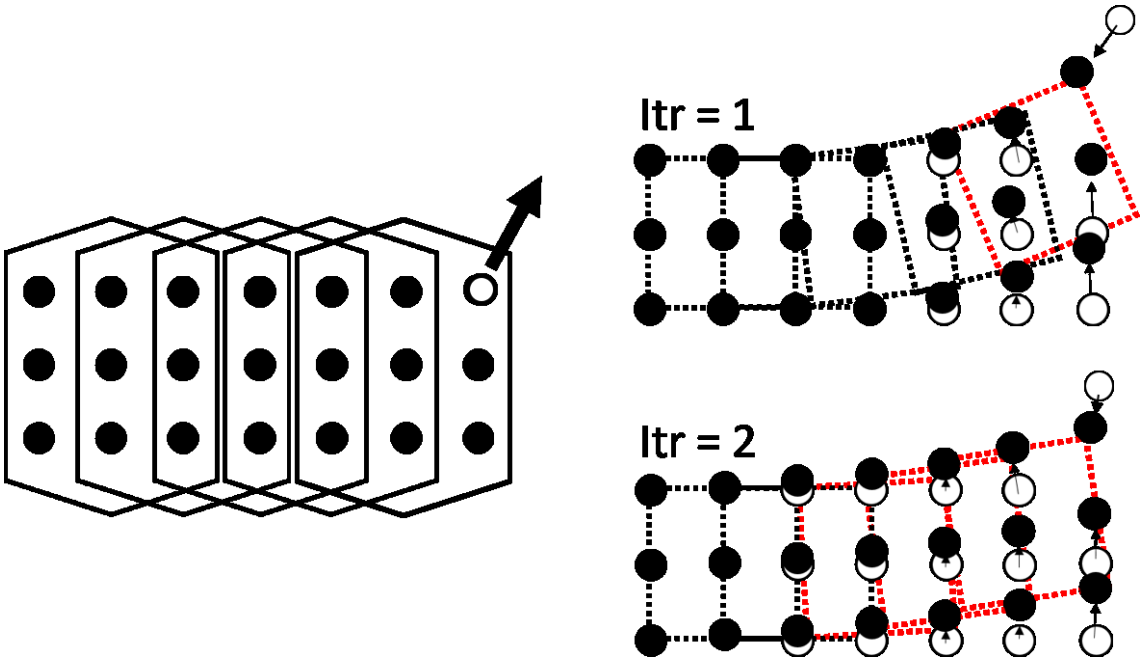


図 2.7: 反復処理による剛性の調整．反復 1 回目は移動した粒子の属するクラスタのみが移動の影響を受ける．反復 2 回目では隣接するクラスタに影響が伝播する．

ングを行う．またこのとき，全ての粒子が選択クラスタのいずれかに含まれなくてはならない．粒子数 6525 個のオブジェクトに対してサンプリングした例を図 2.8 に示す．

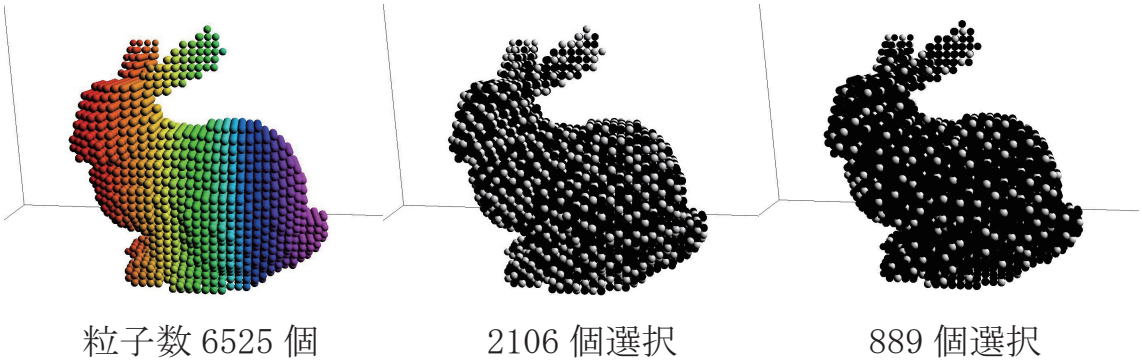


図 2.8: サンプリングの例．白い粒子は選択されたクラスタに対応する粒子である．このモデルでは約 800 個がサンプリング数の下限となっている．

全クラスタ数を n，反復回数を m，サンプリング数を s とすると，従来手法の計算量のオーダーは $O(nm)$，提案手法は $O(sm)$ となる．全体の計算量を一定に保つ場合，n に比べ

て s が小さいほど反復回数 m を大きくすることができる．ただし s が小さいほど計算の誤差は大きくなり，変形の滑らかさは失われてしまう．しかし s を小さくするのは硬い物体を表現する場合であり，処理を繰り返す度に誤差は収束するため，見た目上の問題はほとんどなかった．図 2.9 は Stanford bunny を落下させる実験の結果である．それぞれサンプリング数と反復回数を調整しており，ほぼ同一の計算時間で軟らかい物体と硬い物体を表現できている．

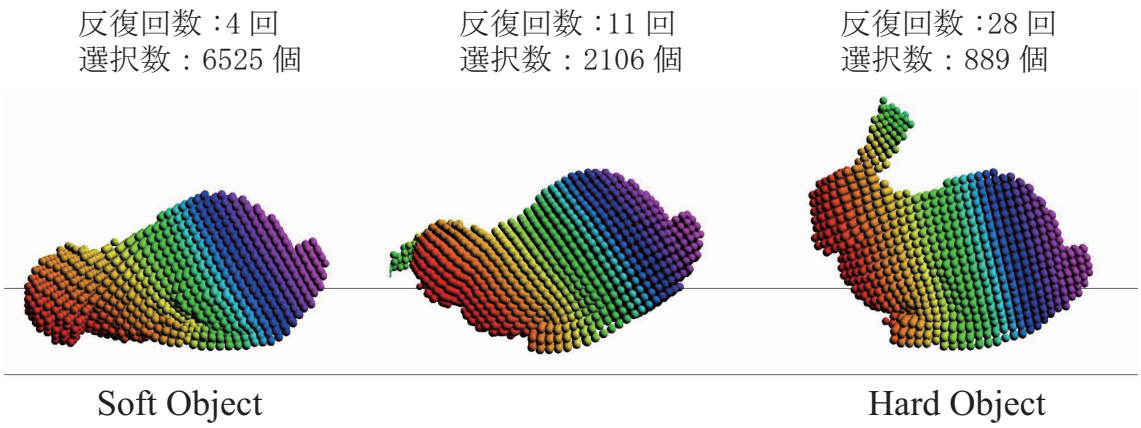


図 2.9: 反復回数とサンプリング数を調整して実験．計算時間は全て約 30fps.

2.3.5 姿勢情報の補間

サンプリングによって運動計算するクラスタを限定すると，限定しない場合に比べて運動計算の精度が落ちてしまう．それを防ぐために，サンプリングクラスタの情報を補間することで非サンプリングクラスタの情報を更新しなくてはならない．特に OP 法では各粒子の姿勢情報を考慮して運動計算を行うため，ここではクラスタに対応する粒子の姿勢情報の補間方法について説明する．まず前提として，非サンプリング粒子に対応するクラスタには必ずいくつかのサンプリング粒子が含まれているとする．この複数のサンプリング粒子を用いて，クォータニオン $\mathbf{q}$ で表される姿勢情報の補間を行う．複数のクォータニオン $\mathbf{q}_i$ を補間して一つのクォータニオン $\mathbf{q}_{blend}$ を求める手法として，QLERP[17] や ExpMap[18] が存在する．QLERP ではクォータニオンを単なる四次元ベクトルとして扱い，式 (2.36) のような４次元ベクトル間で単純に線形補間を行う．

$$\mathbf{q}_{blend} = \sum_{j \in C_i} (-1)^{s_j} w_{q_j} \mathbf{q}_j \tag{2.36}$$

ここで，s_j は最短経路による補間を行うための整数であり，以下の式で求める．

$$s_j = \begin{cases} 1 & (\langle \mathbf{q}^n, \mathbf{q}_j \rangle < 0) \\ 0 & (\langle \mathbf{q}^n, \mathbf{q}_j \rangle \geq 0) \end{cases} \tag{2.37}$$

$\mathbf{q}^n$ は非サンプリング粒子の前フレームの姿勢を表すクォータニオン，$\langle \mathbf{q}_p, \mathbf{q}_j \rangle$ はクォータニオンの内積を表す．

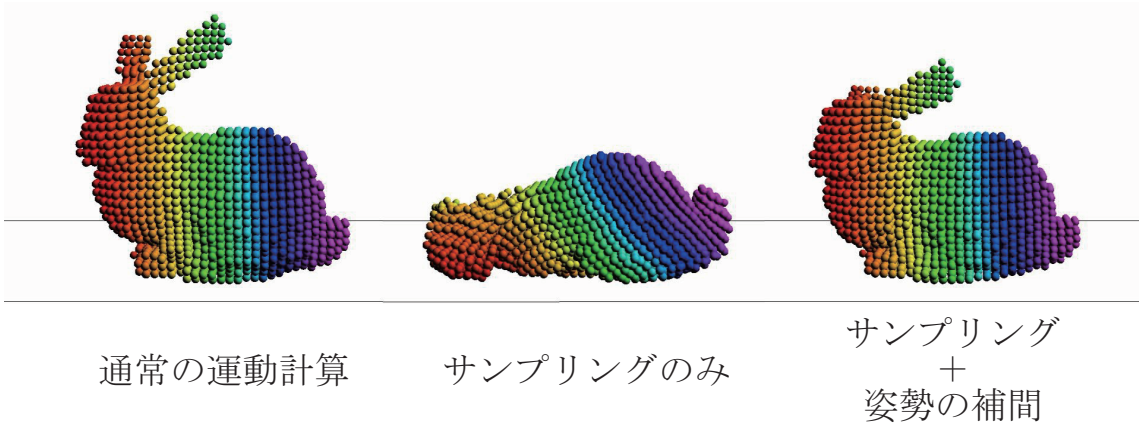


図 2.10: 補間の有無によるアニメーションの違いの比較．Stanford bunny を落下させる実験を行った．

ExpMap を用いた方法では，式 (2.38) で表されるようにクォータニオンを対数空間に写像した後に線形補間し，それを指数演算によってクォータニオンに逆写像することで補間結果を得る．

$$\mathbf{q}_{blend} = \exp(\sum_{j \in C_i} w_{q_j} \ln \mathbf{q}_j) \tag{2.38}$$

両者とも近似処理なので SLERP[19] のような 2 つのクォータニオン間の補間に比べて精度は落ちるが，実験では処理を反復して得られたアニメーションの見た目に問題はなかった．また QLERP と ExpMap で得られたアニメーションの見た目に大きな差が見られなかったため，今回は高速に計算できる QLERP を用いた．また補間に用いる重み係数 w_{q_j} は，補間に用いるサンプリング粒子の総数の逆数を用いた．より補間の精度を上げるには，補間される非サンプリング粒子と補間するサンプリング粒子の距離を基にガウス関数などで重み係数を求めることが考えられるが，計算コストを抑えるために今回は平均値を用いる．

姿勢情報を補間した運動計算の実験結果を図 2.10 に示す．サンプリングしない場合とサンプリングした場合を比べるとアニメーションの内容が大きく異なっているが，補間処理した場合はサンプリングしない場合にある程度近い結果が得られた．

2.4 粘弾性体の運動計算

最終的な粒子の位置･速度は，文献 [16][12] と同様に 2.2 節の液体の運動計算結果 $\mathbf{v}^{fluid}, \mathbf{x}^{fluid}$ と 2.3 節の固体の運動計算結果 $\mathbf{v}^{solid}, \mathbf{x}^{solid}$ を線形補間することで求める．補間はパラメータ $\beta (0 \leq \beta \leq 1)$ を用いて以下の式（2.39）（2.40）で表せる．

$$\mathbf{v}_i^{t+\Delta t} = \beta \mathbf{v}_i^{fluid} + (1-\beta)\mathbf{v}_i^{solid} \tag{2.39}$$

$$\mathbf{x}_i^{t+\Delta t} = \beta \mathbf{x}_i^{fluid} + (1-\beta)\mathbf{x}_i^{solid} \tag{2.40}$$

パラメータ β を変えることで物性を固体，粘弾性体，液体と変化させられる．パラメータ β が 0 に近い場合は，固体としての動きが強く見られ，パラメータが β が 1 に近い場合は液体

に近い振る舞いをする．パラメータ β が 0.5 に近い場合は，液体と固体の中間状態である粘弾性体の振る舞いをする．ロウやマグマなどの物体の中間状態としては粘弾性体が見られるが，水と氷のような物体の場合は液体と固体で状態がはっきり分かれている．提案手法では，パラメータ β の変動方法を 2 つ用意することで中間状態が存在する物体と中間状態が存在しない物体の両方を表現している．中間状態が存在する物体の場合は温度によってパラメータ β を変動させるが，存在しない物体の場合はパラメータ β を用いずに融解処理を行う．

2.5 陰関数場の生成とレンダリング

粒子法において，表面を可視化するために様々な手法が提案されている．Müller ら [21] は，カーネル関数を用いたメタボールで液体表面形状を表現し，マーチングキューブ法 [22] により液体表面を抽出することで，インタラクティブな流体アニメーションを可能にした．Yu ら [23] は，マーチングキューブに用いる陰関数を計算する際に異方性カーネルを用いることで滑らかな表面を生成した．本論文ではリアルタイムな処理を目標とする場合は Müller ら [21] の手法を，高品質な処理を目標とする場合は Yu ら [23] の手法を用いてレンダリングを行う．

第3章

実験と考察

3.1 実験環境

本章では，第2章で述べた弾性体の運動計算手法を実装したプログラムを用いて実験を行う．その後，提案手法の有用性について考察を行う．

実験を行った環境を表3.1に示す．

表 3.1: 実行環境

CPU	Intel Core i5-3470 3.20GHz
Main Memory	8GB
GPU	NVIDIA GeForce GTX 770
Video Memory	4GB
OS	Microsoft Windows 7
開発言語	C++, C for CUDA, OpenMP
グラフィックス API	OpenGL

ただし，上記実験環境の GPU を用いたのは液体の動きを計算する部分と熱処理とレンダリング部分のみである．それ以外の処理は全て CPU を用いて実装し，OpenMP を用いて部分的に並列処理している．

表3.2はシミュレーションで共通して用いたパラメータである．

表 3.2: シミュレーションパラメータ

タイムステップ $dt[sec]$	0.01
静止密度 $\rho[kg/m^3]$	998.29
粘性係数 $\mu[kg/m \cdot sec]$	0.04
ガス定数 k_{press}	3.0
熱拡散係数 h	0.1
熱伝達係数 c_d	0.01

3.2 剛性調整に関する実験

3.2.1 実験設定

硬い物体の表現について，提案手法の有用性を確かめるために各制約条件とソルバーを組み合わせた運動計算方法を用意して実験を行う．

- 従来手法の制約条件とソルバーの組み合わせ
  1. SM 法 + Jacobi ソルバー
  2. SM 法 + Gauss-Seidel ソルバー
  3. OP 法 + Jacobi ソルバー
  4. OP 法 + Gauss-Seidel ソルバー
- 従来手法と提案手法の比較のための組み合わせ
  1. OP 法 + 重み付き Jacobi ソルバー
  2. OP 法 + Hybrid ソルバー
  3. OP 法 + 重み付き Hybrid ソルバー

まず，従来手法の組み合わせて実験する．運動計算結果の評価には，新たに提案する指標である変形量 d_{sum} を用いる．変形量 d_{sum} は以下の式（3.1）で求める．

$$d_{sum} = \sum_{i}^{N} \sum_{j \in C_i} ||\mathbf{R}_i^*(\mathbf{x}_i^0 - \mathbf{x}_{i,cm}^0) - (\mathbf{x}_{i,solid} - \mathbf{x}_{i,cm}^*)|| \tag{3.1}$$

あるフレームにおける運動計算において，クラスタ i で最終的に得られた回転行列を $\mathbf{R}_i^*$ とする．また，あるフレームにおける最終的な固体粒子 i の位置を $\mathbf{x}_{i,solid}$ とする．この最終的な固体粒子位置を用いて，再度クラスタ i で求めた重心座標を $\mathbf{x}_{i,cm}^*$ とする．上記の式は，物体を構成するクラスタでの局所的な最終位置と，ソルバーによって得られた物体全体における最終位置の差分の和を表している．物体の変形が小さいほど，各制約条件を解いて得られたローカルな解とソルバーによって得られたグローバルな解の差は小さくなる．そのため，非変形の場合は $d_{sum} = 0$ となる．

制約条件とソルバーによって，変形状態からの元の形状に戻るまでの収束の速さ（以下では弾性回復力と呼ぶ）がどのように変化するかを数値によって示す．続いて，従来手法と提案手法を比較する実験を行う．提案手法によってどのように弾性回復力が向上しているかを示す．

実験には粒子数 6525 個の Stanford bunny を用いる．変形状態から元の形状への回復力を確認するため，シミュレーションには図 3.1 のように平面状に潰れた状態を初期状態，元の Stanford bunny を目標形状としたオブジェクトを用いる．各種法を組み合わせて運動計算を行うことで，潰れた状態からどのくらい速く元の形状へと戻るのかを実験によって確かめる．

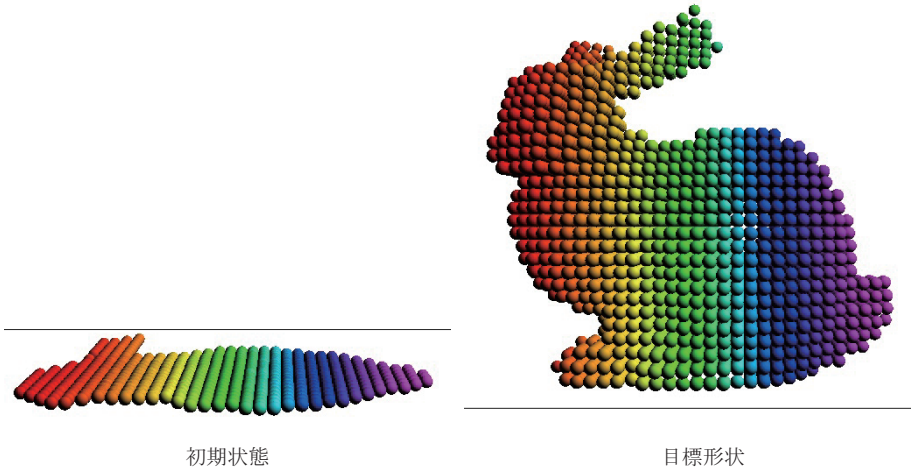


図 3.1: 変形状態から元の形状への回復を確認するための初期状態と目標形状

3.2.2 実験結果と考察

制約条件とソルバーの組み合わせ

図 3.2 は，それぞれの制約条件とソルバーを組み合わせた場合の変形量と反復回数のグラフである．横軸が反復回数，縦軸が変形量となっている．それぞれの結果で共通しているのは，反復回数が増えるにつれて変形量が減少している点である．これは運動計算を反復処理することによって，目標形状と現状の粒子位置との間の差分が小さくなっていることを示している．制約条件に SM 法を用いたグラフの動きが単調減少でないのは，Stanford bunny の耳の部分や頭の部分などでクラスタ間での重複粒子数が少ないことが原因だと考えられる．

図 3.2 のグラフから，制約条件においては SM 法より OP 法を用いるほうがより高速に変形量が減少することが確認できる．また，ソルバーにおいては Jacobi ソルバーよりも GS ソルバーのほうが変形量の減少に寄与していることが分かる．

図 3.3 の各画像はそれぞれのシミュレーション結果を可視化したものである．潰れた状態から元形状へと復帰していく過程を確認できる．SM 法と GS ソルバーを組み合わせた結果に注目すると，物体の変形は小さくなっているが元姿勢から回転してしまっている．これは，SM 法では粒子間の位置情報しか考慮しせず姿勢を保存しないためであると考えられる．

この実験結果から，硬い物体を表現するには制約条件には SM 法ではなく OP 法を，ソルバーには Jacobi ソルバーよりも GS ソルバーを用いるべきであることが確認できた．また，図 3.2 より GS ソルバーを用いると制約条件に SM 法を用いても弾性回復力が大きいことから，制約条件とソルバーではソルバーのほうが寄与が大きいことが分かる．

従来手法と提案手法の比較

次に，提案手法である重み付きのソルバーと従来手法を比較した実験結果を示す．図 3.4 から，重み付き和を用いるとより早く変形量が減少することがわかる．Jacobi ソルバーと

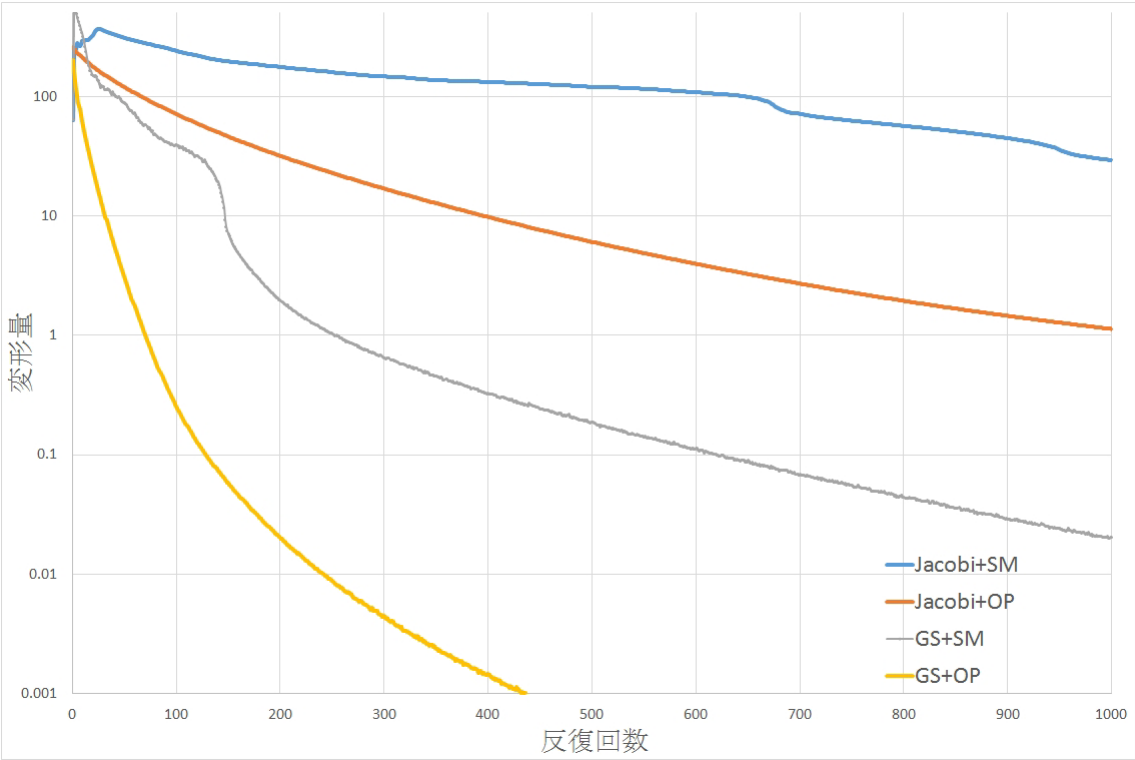


図 3.2: 制約条件とソルバーを組み合わせた場合の反復回数と変形量を比較

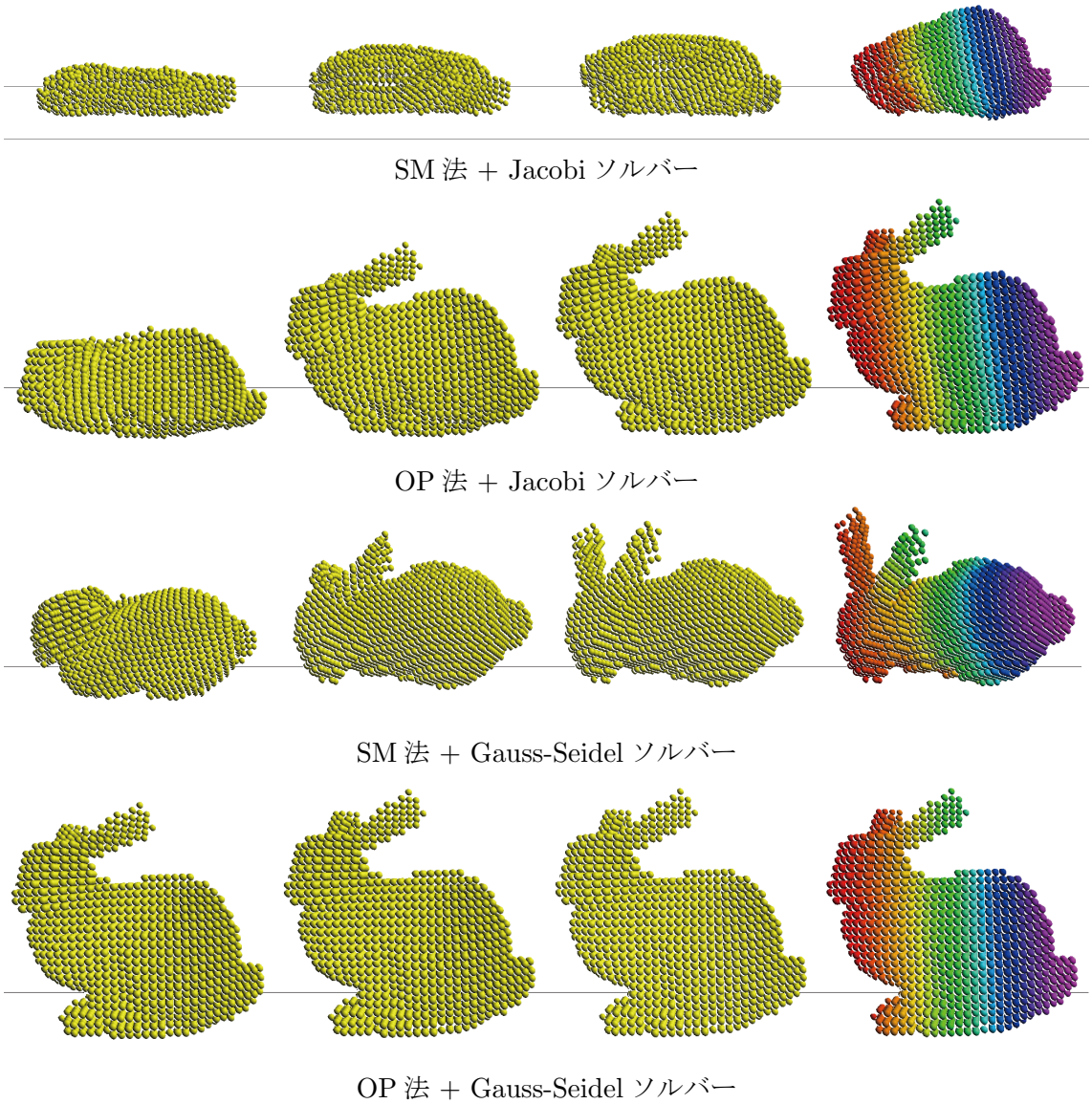


図 3.3: 制約条件とソルバーの組み合わせ．左から反復 100 回目,400,700,1000 回目

Hybrid ソルバーの両方で効果があることを確認できた．重み付き和を用いた場合とそうでない場合でグラフの数値を比較すると，重み付き和には運動計算数十回分の削減効果があることがわかる．

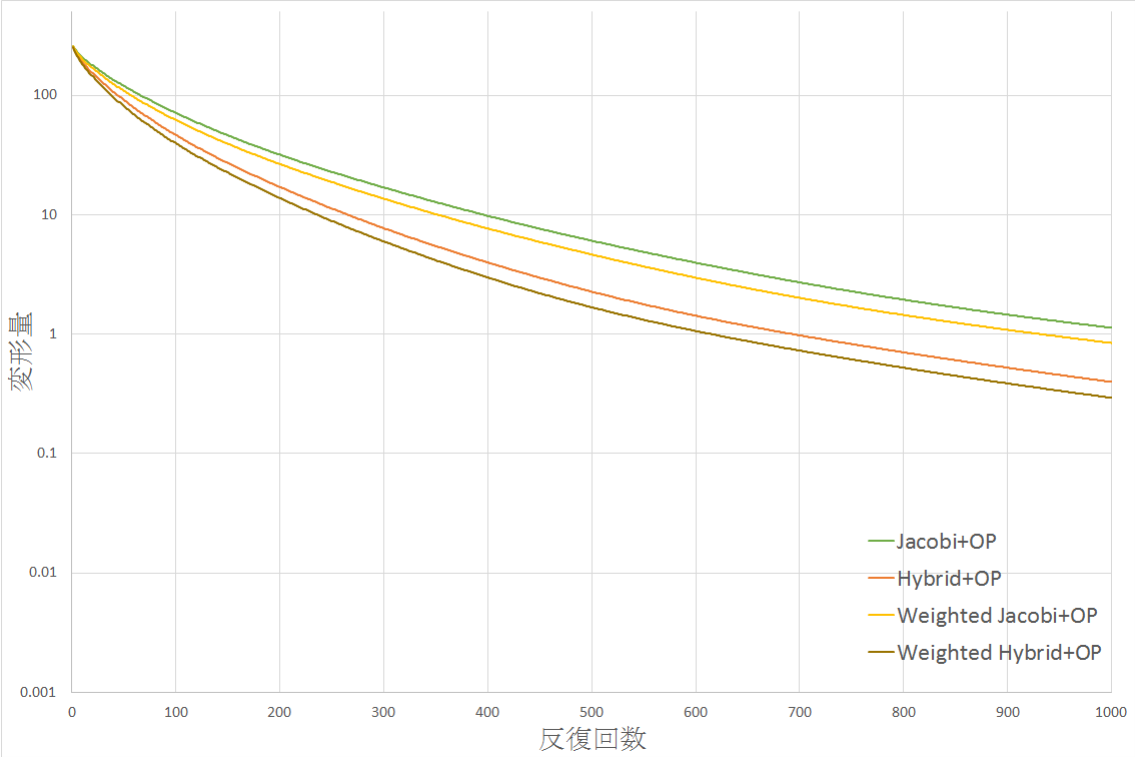


図 3.4: 制約条件とソルバーを組み合わせた場合の反復回数と変形量を比較

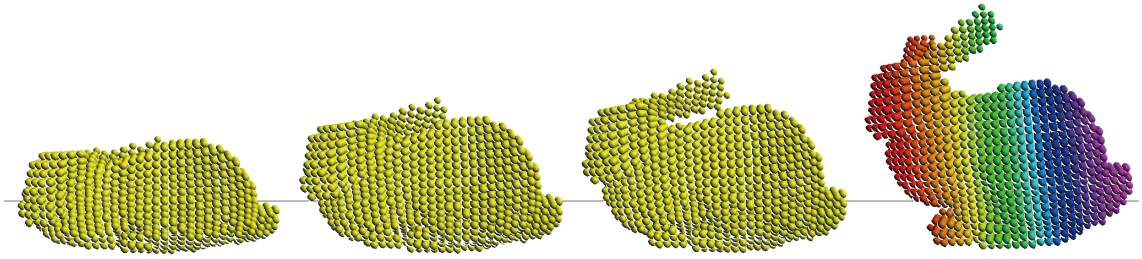
図 3.5 の各画像はそれぞれのシミュレーション結果を可視化したものである．重み付き Jacobi ソルバーと重み付き Hybrid ソルバーの結果を比べると，僅かに耳の部分に差があることが分かる．見た目には僅かな差だが，反復回数を比較すると重み付き Jacobi ソルバーの反復処理 300 回目と重み付き Hybrid ソルバーの反復処理 200 回目がほとんど同じ結果となっている．これは，提案手法により運動計算 100 回分ほど変形の伝播を高速化できたことを表しており，グラフの数値からも同様の事実を確認できる．

この実験結果から，重み付き和は変形の伝播を高速化することがわかった．しかし見た目上の変化は僅かであり，またリアルタイムアプリケーションにおいて有用かどうかはわからない．そこで，次節ではリアルタイムアプリケーションを想定した実験を行った．

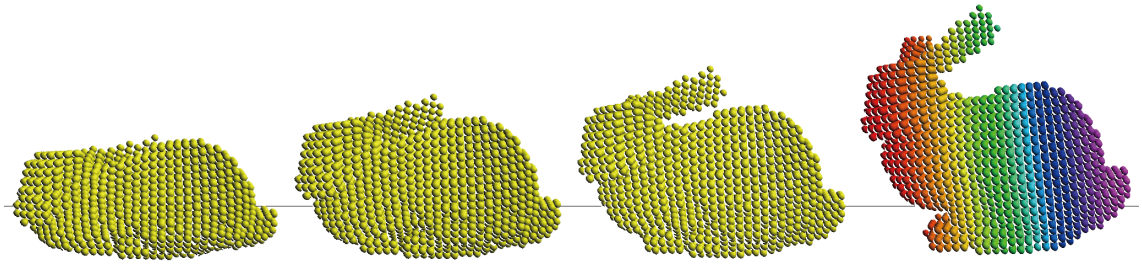
3.3 計算時間を固定した実験

3.3.1 実験設定

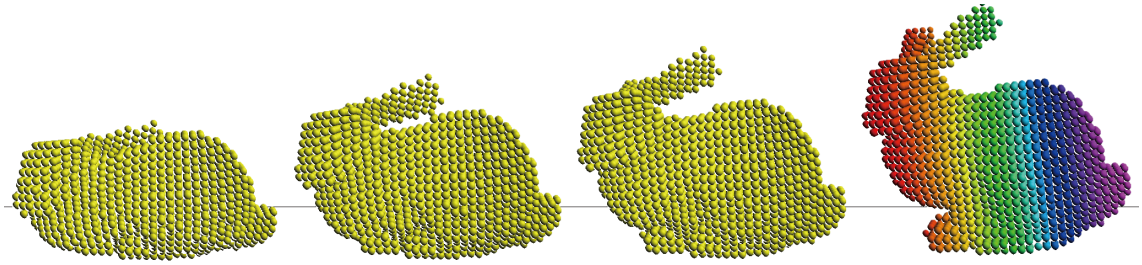
1 フレーム内の計算時間を約 33ms（30fps）に固定したときに，どこまで硬い物体をシミュレーションできるかの実験を行う．実験には以下の制約条件とソルバーの組み合わせを



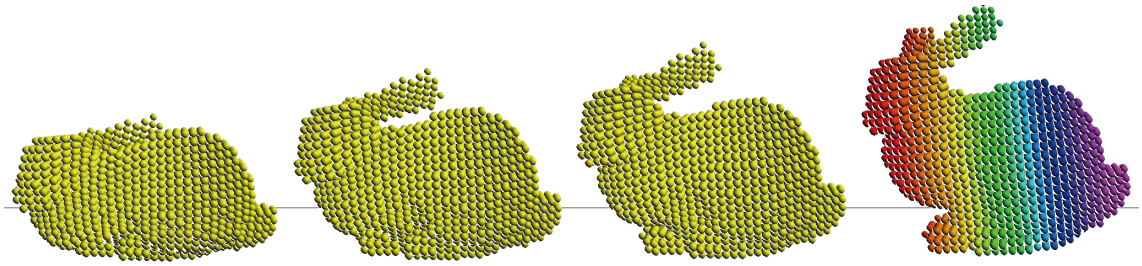
OP 法 + Jacobi ソルバー



OP 法 + 重み付き Jacobi ソルバー



OP 法 + Hybrid ソルバー



OP 法 + 重み付き Hybrid ソルバー

図 3.5: ソルバーの改良結果．左から反復 100 回目,200,300,1000 回目

用いた．

1. SM 法 + 重み付き Jacobi ソルバー
2. OP 法 + 重み付き Jacobi ソルバー
3. OP 法 + GS ソルバー（サンプリングあり，なし）
4. OP 法 + 重み付き Hybrid ソルバー

出来るだけ反復回数を増やすためにクラスタのサンプリングを行っており，サンプリング数は共通して 841 個である．ただし，OP 法と GS ソルバーを組み合わせたものはサンプリングすると軟らかい物体となってしまったので，サンプリングせずに全てのクラスタを用いてシミュレーションした結果も示す．

実験には粒子数 6525 個の Stanford bunny を用いる．リアルタイムなアニメーションでどのくらい硬い物体を表現できるかを確かめるため，Stanford bunny を落下させ地面と衝突させた様子を観察する．

3.3.2 実験結果と考察

図 3.6 の各画像はそれぞれの組み合わせの結果であり，シミュレーション結果を可視化したものである．また，それぞれのシミュレーションの反復回数と，運動計算一回あたりの計算時間（反復処理一回あたりの計算時間）を表 3.3 に示す． 制約条件に SM 法を用いると，

表 3.3: 各組み合わせの計算時間と反復回数

	反復回数	運動計算時間
SM + 重み付き Jacobi	40	0.8ms
OP + 重み付き Jacobi	17	1.9ms
OP + GS（サンプリングあり）	4	8.5ms
OP + GS（サンプリングなし）	1	32.3ms
OP + 重み付き Hybrid	4	7.5ms

耳の部分のようなクラスタ間の重複が少ない部分が硬くならない．また，地面と接触する部分の変形が大きくなっている．OP 法と重み付き Jacobi ソルバーを組み合わせると，耳の部分や地面との接触部分でも硬い弾性体を表現をできた．GS ソルバーを用いると，サンプリングしている場合は反復処理しているにも関わらず変形と地面との接触部分での振動が目立った．サンプリングしていない場合は，変形と地面との接触部分で粒子が振動するが，物体の変形は小さかった．重み付き Hybrid ソルバーを用いると，GS ソルバーでサンプリングした場合より少し硬い物体となった．しかし，重み付き Jacobi ソルバーを用いた場合と比べて変形の目立ったアニメーションとなった．これは，3.2.2 節の図 3.4 のグラフの形とは異なる結果である．原因としては，重み付き Hybrid ソルバーの反復回数が 4 回と少ないことが考えられる．3.2.2 節の実験結果の変形量の値を確認すると，重み付き Jacobi ソルバーで反復回数 17 回の場合の変形量は，重み付き Hybrid ソルバーの反復回数が 13 回の場合に

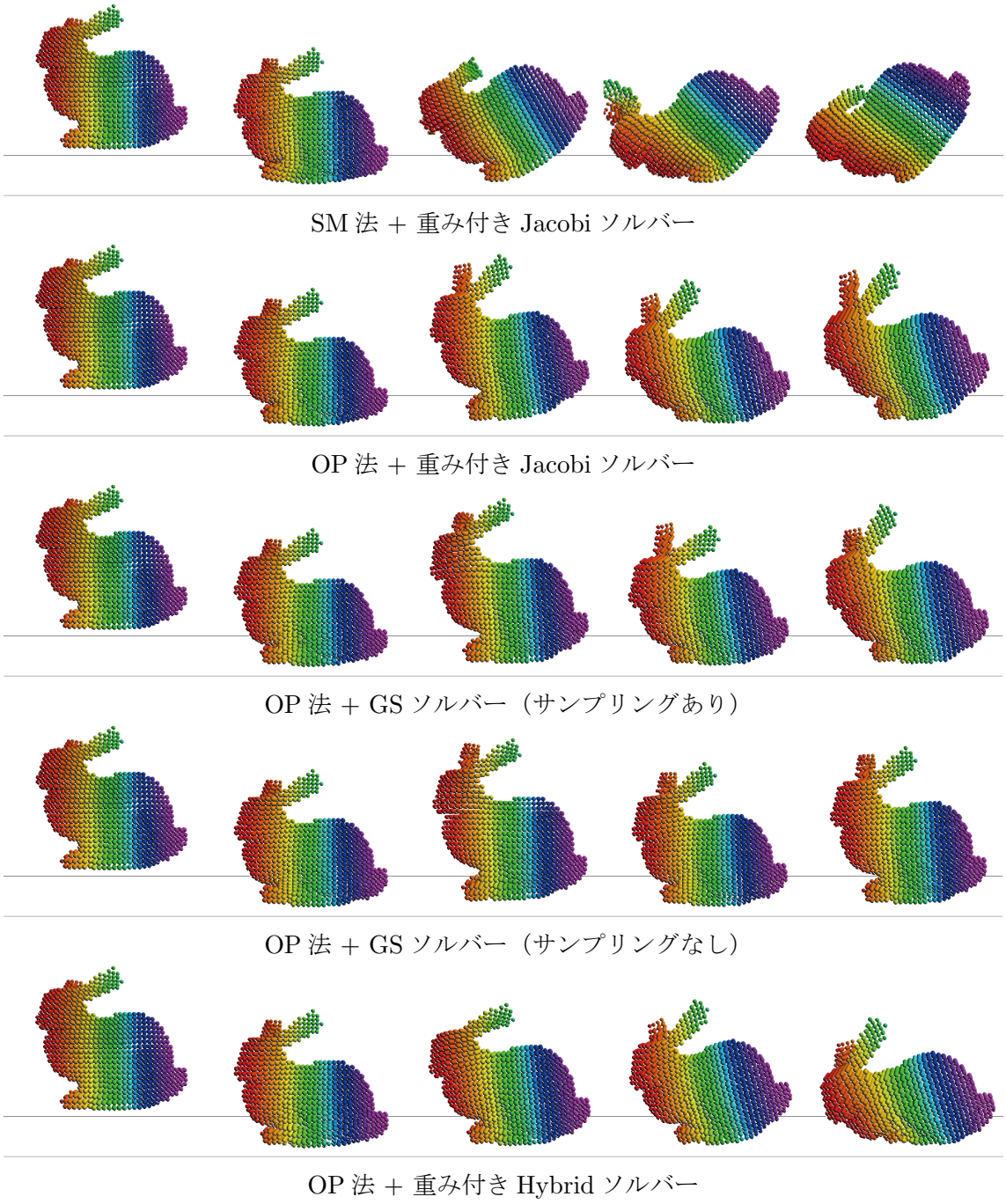


図 3.6: 計算時間を固定した実験結果．左から初期状態,15,30,45,60 フレーム目

ほぼ等しくなる．そのため，今回の実験結果のように重み付き Jacobi ソルバーのほうが硬い物体を表現できたと予想できる．

制約条件に SM 法を用いると，運動計算一回あたりの計算時間が小さいため反復回数を多くすることができる．OP 法を用いると，運動計算一回あたりの計算時間が多いため反復回数は少なくなるが，床との接触部分や耳のような細長い部分の変形を修正できる．GS ソルバーを用いると，一回の運動計算に時間がかかりすぎて反復回数が少なくなっている．また，重み付き Jacobi ソルバーを用いて反復回数を増やした場合は，変形が物体全体にうまく伝播しているためオブジェクト全体が回転している．これに対して GS ソルバーを用いた場合は，反復回数が少ないため地面との接触部分で粒子が振動している．重み付き Hybrid ソルバーの反復回数も 4 回だが，Jacobi ソルバーが入っている分だけシミュレーション結果が安定したと考えられる．

3.4 融解シミュレーション

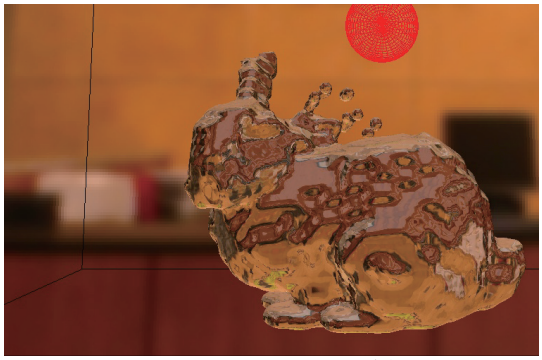
提案手法を用いて物体が固体から液体へと融解するアニメーションを作成した．図 3.7 中の上方にある小さな赤い球体は熱源であり，この球体から近くにある粒子へと熱を与える．弾性体のシミュレーションには OP 法と重み付き Jacobi ソルバーを用いている．図 3.7 中の (a) は初期状態を表している．熱源をオブジェクトへ近づけると，図 3.7 中の (b) のように熱源の近くにある耳の部分が融解している様子が確認できる．(c), (d), (e) では熱源をオブエジェクトへより近づけており，融解に伴い液体粒子が物体表面を伝って床の上へと流れているのがわかる．(f) から (h) の間では物体は崩れ落ち，最終的に複数の部分へと分離している．

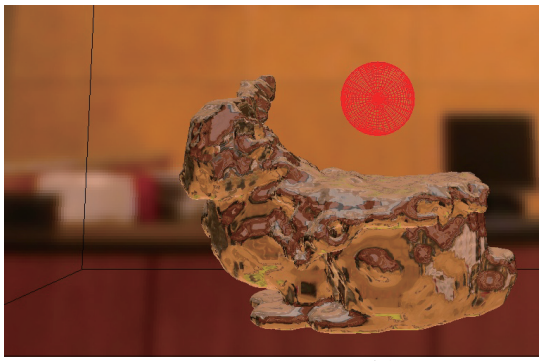
平均計算時間は約 44ms で，リアルタイムにシミュレーションすることができた．また，粒子の融解処理に伴う計算時間の増減はほとんどなく安定していた．以下の表 3.4 では，シミュレーションにかかった計算時間の内訳と反復回数を示している．

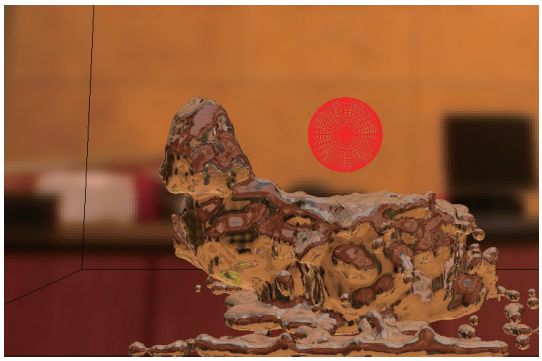
表 3.4: 融解シミュレーションの平均計算時間と反復回数

	運動計算	レンダリング	伝熱・融解処理	反復回数
OP 法+重み付き Jacobi ソルバー	33ms	9ms	1ms	17 回

(a) 17 フレーム目


(b) 35 フレーム目


(c) 70 フレーム目


(d) 125 フレーム

(e) 200 フレーム

(f) 375 フレーム

(g) 400 フレーム

(h) 425 フレーム

図 3.7: 固体から液体への融解

第4章

結論

4.1 まとめ

本論文では，固体，弾性体，粘弾性体，液体という4つの状態を統一的に扱いながら，リアルタイムに融解アニメーションを作成するための方法を提案した．弾性体の運動計算をクラスタ間の重複粒子数を出来る限り少なくして行うことで，物体の構造変化へ高速に対応した．また，従来手法のようにクラスタの構造を修正するのではなく，制約条件とソルバーを改良することで変形の伝播速度を向上させた．クラスタの構造ではなく運動計算の反復処理によって物体の硬さを調整するために，クラスタのサンプリングをすることでリアルタイムアプリケーションにおける反復回数の上限を広げた．

実験によって，提案手法は硬い弾性体を高速にシミュレーション可能であることがわかった．制約条件とソルバーをうまく組み合わせることで，軟らかい物体から硬い物体までの物性を計算時間によらずに扱うことができた．また，リアルタイムな融解アニメーションを作成することで提案手法の構造変化への高速な対応を確認した．

4.2 今後の課題

今後の課題としては，さらに硬い物体の表現を実現して幅広い物性を扱うことが挙げられる．現状では表現可能な物体の硬さに限界があるため，作成可能なアニメーションの表現に制限がある．提案手法のソルバーは制約条件の解の空間的な性質に着目していたが，Chebyshev準反復法のような解の時間的な推移に着目したソルバーを導入することでより変形の伝播速度を向上させられるだろう．加えて，提案手法ではHybridソルバーを用いる際に粒子の偶奇性で制約条件を分類しているが，粒子の位置情報を考慮した分類方法を開発することで高速に変形を伝搬できるだろう．

シミュレーション結果のレンダリングに関しては，現状では物体の表面に球体の丸みが目立ち不自然な見た目となっているため，弾性体のシミュレーション結果である粒子の姿勢情報を利用することでより高精度なメッシュを作成することが挙げられる．また，ユーザーが直感的に硬さを制御できるような単一なパラメータによるサンプリング数と反復回数の決定方法も課題である．

謝辞

本論文を締めくくるにあたり，様々な面から懇切丁寧に研究をご指導していただきました藤澤誠助教授に心から感謝の意を表します．また，合同ゼミにおいて数多くの助言をいただきました三河正彦准教授に深く感謝いたします．そして研究活動や日常生活において多くのご協力をいただきました物理ベースコンピュータグラフィックス研究室の皆様，知能ロボット研究室の皆様，諸先輩方に深く御礼申し上げます．最後に，博士前期課程に進学することを許してもらい，学生生活を暖かく支援して下さった両親に深く感謝いたします．

参考文献

[1] M. Müller, B. Heidelberger, M. Hennix and J. Ratcliff, “Position Based Dynamics”, J. Vis. Comun. Image Represent., 18(2), pp.109-118, 2007.

[2] M. Fratarcangeli and F. Pellacini, “Scalable Partitioning for Parallel Position Based Dynamics”, Computer Graphics Forum, 34, pp.405-413, 2015.

[3] M. Macklin, M. Müller, N. Chentanez and T.-Y. Kim, “Unified Particle Physics for Real-Time Applications”, ACM ToG, 33(4), pp.153:1-153:12, 2014.

[4] M. Müller, N. Chentanez and T.-Y. Kim, “Real Time Dynamic Fracture with Volumetric Approximate Convex Decompositions”, ACM ToG, 32(4), pp.115:1-115:10, 2013.

[5] D. Baraff and A. Wiltkin, “Large Steps in Cloth Simulation”, In Proc. ACM SIGGRAPH ’98, pp. 43-54, 1998.

[6] M. Müller, B. Heidelberger, M. Teschner and M. Gross, “Meshless Deformations Based on Shape Matching”, ACM ToG, 24(3), pp.471-478, 2005.

[7] A. R. Rivers and D. L. James, “FastLSM: Fast Lattice Shape Matching for Robust Real-Time Deformation”, ACM ToG, 26(3), pp.82:1-82:6, 2007.

[8] D. Steinemann, M. A. Otaduy and M. Gross, “Fast Adaptive Shape Matching Deformations”, In Proc.SCA2008, pp.87-94, 2008.

[9] R. Diziol, J. Bender, and D. Bayer, “Robust Real-Time Deformation of Incompressible Surface Meshes”, In Proc. SCA2011, pp.237-246, 2011.

[10] 井尻 敬，芦原 貴司，梅谷 信行，小山 裕己，五十嵐 健夫，原口 亮，横田 秀夫，中沢 一雄, “Shape Matching 法による心臓拍動のビジュアルシミュレーション”, 生体医工学, 53(3), pp.130-137, 2015.

[11] M. Müller and N. Chentanez, “Solid Simulation with Oriented Particles”, ACM ToG, 30(4), pp.92:1-92:10, 2011.

[12] K. Takamatsu and T. Kanai, “A Fast and Practical Method for Animating Particle-Based Viscoelastic Fluid”, The International Journal of Virtual Reality, 10(1), pp.29-35, 2011.

[13] M. Fujisawa and K. Miura, “Animation of Ice Melting Phenomenon Based on Thermodynamics with Thermal Radiation”, In Proc. the 5th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (GRAPHITE 2007), pp 249-256, 2007.

[14] B. Solenthaler, J. Schläfli and R. Pajarola, “A Unified Particle Model for Fluid-Solid Interactions”, Computer Animation and Virtual Worlds, 18(1):69-82, 2007.

[15] 内田 英行，岩崎 慶，土橋 宜典，西田 友是, “融解後の水を考慮した粒子ベース氷塊融解シミュレーション”, Visual Computing/グラフィクスと CAD 合同シンポジウム 2010 予稿集, pp.5:1-5:6, 2010.

[16] 仲宗根 良，藤澤 誠，三河 正彦, “SPH 法と Shape Matching 法を用いた相変化シミュレーション”, Visual Computing/グラフィクスと CAD 合同シンポジウム 2014 予稿集, pp.49:1-49:4, 2014.

[17] F. S. Grassia, “Practical Parameterization of Rotations Using the Exponential Map”, J. Graph. Tools, 3(3), pp.29-48, 1998.

[18] L. Kavan and J. Žára, “Spherical Blend Skinning: A Real-Time Deformation of Articulated Models”, In Proc, I3D ’05, ACM, pp.9-16, 2005.

[19] K. Shoemake, “Animating rotation with quaternion curves”, Comput. Graph. 19(3), pp.245-254, 1985.

[20] R. Bridson, “Fast Poisson Disk Sampling in Arbitrary Dimensions”, ACM SIGGRAPH sketches, Article22, 2007.

[21] M. Müller, D. Charypar and M. Gross, “Particle-based Fluid Simulation for Interactive Applications”, In Proc. SCA2003, pp.154-159, 2003.

[22] W. Lorensen and H. Cline, “Marching Cubes: a High Resolution 3D Surface Construction Algorithm”, Computer Graphics (Proc. SIGGRAPH ’87), 21, pp.163-169, 1987.

[23] J. Yu and G. Turk, “Reconstructing surface of particle-based fluids using anisotropic kernels”, In Proc. SCA2010, pp.217-225, 2010.