Vol.2019-CG-175 No.6
2019/9/28

流体の再シミュレーション効率化のための領域の自動調整

加地 佑季[1,a)]　佐藤 周平[2,b)]　櫻井 快勢[3,c)]

概要：流体シミュレーションは映画やゲームなどのエンターテインメント分野において，リアルな映像を作るために用いられる．しかし，その計算コストは非常に大きい．更に，所望の映像を得るためには何度もパラメータ等の調整を行いながらシミュレーションを実行する必要があり，非常に時間がかかる．この問題を解決するため，既存のシミュレーション結果の一部のみを局所的に再シミュレーションする方法が提案されている．この方法により，本来は一部を編集するために全体を再計算しなければならなかったところが，必要な一部領域のみでの計算で済ませることができる．しかしこの従来手法では，計算領域が固定であるため，編集したい領域が時間とともに広がる，あるいは大きく移動するようなシーンでは常に広い計算領域を確保しておかなければならず，あまりコストを削減できない．そこで我々は，従来手法の計算領域の位置と大きさを適応的に変更する手法を提案する．我々の手法では，編集前の流れと編集後の流れの差分を基準に計算領域の境界位置を変化させる．また，移動する物体などを追加していく場合は，その移動に応じて計算領域の中心位置を移動させる．これにより，編集したい箇所が動く場合でもそれを追従して必要最小限の領域を設定することができ，従来法よりも効率的に再計算ができる．本稿では，2 次元と 3 次元の液体シミュレーションについていくつかの結果を提示し，提案手法が有効であることを示す．

キーワード：流体シミュレーション，再シミュレーション，perfectly matched layers，液体の局所編集

1. はじめに

物理ベースの流体シミュレーションは，映画やゲームなどの映像製作において，リアルな水や煙などを表現するために用いられる．しかし，高品質の映像を得るためにはその計算量は膨大になる．また，所望の映像を作成するためにはパラメータ調整に時間をかける必要がある．もし，再シミュレーションを一から全てやり直すことなく既存の流体アニメーションを編集できれば，このような時間のかかる工程のコストを下げることができる．そこで，我々は再シミュレーションと呼ばれる技術に注目する．一般的な再シミュレーションでは，全域でシミュレーションをやり直すが，その場合ユーザが編集したくない領域も含まれてしまう．これに対し，Bojsen-Hansen ら [1] は，既存の流れの一部のみを再度シミュレーションし，元の流れと違和感なく繋げることで，流れの部分的な編集を可能とする手法を提案した．この方法では，再計算領域（編集領域）の境界付近で，Perfectly Matched Layers（PMLs）[2] という技術を用いて既存の流れと再シミュレーション後の流れを滑らかに接続する．しかし，この手法では，編集領域の大きさと位置が固定されているため，例えば海に大きく動く船を追加したい場合のような，編集する部分が大きく移動する場合では，編集領域もあらかじめ大きく設定しておく必要があり，コストの削減率が低くなってしまう．

そこで我々は，既存手法 [1] での編集領域の位置やサイズを毎フレーム適切に調整する手法を提案する．本手法では，既存の流れと再シミュレーション後の流れの差分に応じて，編集領域の境界の位置を移動させる．具体的には，差分が大きいほど領域が広がるように境界が移動し，逆に差分が小さくなると領域が狭まる方向に境界が移動する．この処理とは別に，動く障害物などを追加する場合は，その移動に応じて編集領域の中心も移動させる．本手法では，各フレームにおいて計算を行う領域を最適化するため，従来手法よりも更にコストを削減できる．本稿では，従来手法と同様に液体のシミュレーションを対象とする．いくつかの条件について実験を行い，提案手法が有効であることを示す．

1 東京大学
The University of Tokyo
2 富山大学
University of Toyama
3 株式会社ドワンゴ
DWANGO Co., Ltd.
a) kaji@nae-lab.org
b) ssato@eng.u-toyama.ac.jp
c) kaisei_sakurai@dwango.co.jp

2. 関連研究

Kim ら [3] は，既存のシミュレーション結果に対し，外力やシミュレーションの総フレーム数などの条件を変更した場合の流れを高速で再シミュレーションすることを可能とした．この手法はシミュレーション空間の全体を再シミュレーションする手法であり，一部の領域のみを再シミュレーションすることはできない．これに対し Bojsen-Hansen ら [1] は，ユーザが指定した一部の領域のみを再シミュレーションする方法を提案した．この手法では，元の流れと再計算した流れを滑らかに接続するために，Söderström ら [2] が用いた PML の技術を利用した．しかし前述の通り，編集領域が固定であるため，例えば船が波のある広い海を移動していく状況のように，時間経過で編集する領域が広範囲を移動する場合には，編集領域を初めから十分に大きく設定しておく必要があり，コスト削減の割合が低い．これに対し我々は，編集領域を適応的に変化させ，さらにコストを削減する．

液体の再シミュレーションにおいてシミュレーション領域を移動させる手法としては，Stomakhin ら [4] の手法があげられる．この方法では，広い海などの一部で領域を移動させながら流れを再シミュレーションすることが可能である．この手法は粒子ベースの手法と粒子と格子を組み合わせた手法に適用可能である．この手法はユーザが指定したシミュレーション領域境界の形状と速度の分布から境界条件を取得するもので，境界条件として既存の流れを与えればシミュレーション領域外部との整合性を保って流れを計算することが可能である．しかし，粒子ベースの方法のため格子ベースのシミュレーションにそのまま適用するのは難しい．また我々は既存のデータに加える編集に対して自動で最適な編集領域を計算することを目的としている．

煙のシミュレーションデータを後処理的に編集する手法として，Sato らの方法がある [5]．この手法ではある既存の流れのデータの一部を切り取り，別の流れに滑らかに合成することができる．これは組み合わせる流れのデータの境界の流れを補間することで達成される．また，補間される流れが可能な限り物理法則を満足するように手法がデザインされている．しかし，この手法では流れを合成する位置を時間的に変更することはできない．また，気体のシミュレーションのみを対象としている．

3. 提案手法

この節では，提案手法のアルゴリズムを説明する．まず最初に，提案手法のベースとなっている，Bojsen-Hansen らの提案した再シミュレーション手法について紹介する．その後，提案手法による拡張部分である，適応的に再シミュレーションの領域境界の位置を決定する方法について述べる．図 1 に既存手法及び提案手法の説明に用いる記号の定義を示す．本節ではまず，簡単のために二次元のシミュレーションを対象に手法の基礎の部分を説明する．三次元の場合に追加される部分については本節の最後で述べる．

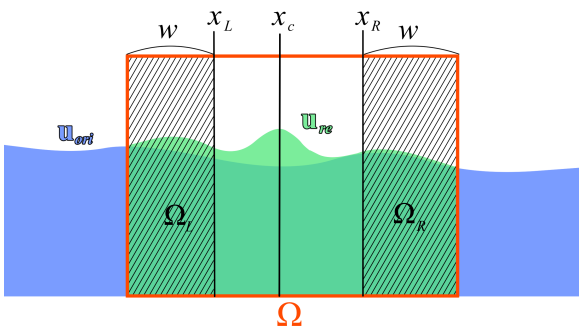

図 1　記号の定義

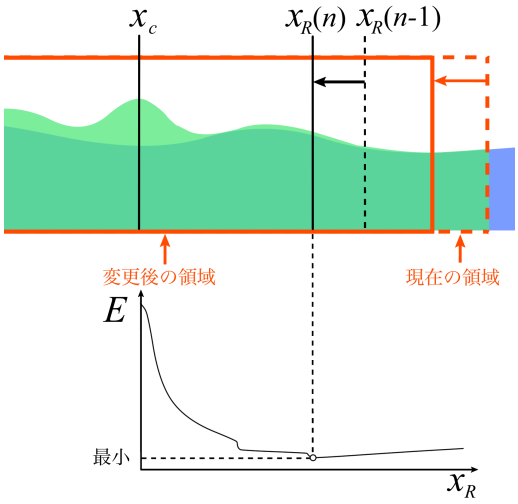

図 2　$E(x_a)$ と変更後の領域の対応

既存手法 [1] では，液体の流れ $\mathbf{u}_{re}$ をユーザが指定した部分領域 Ω においてシミュレーションする．この際，PML を用いることで，結果として得られる Ω 内の流れ $\mathbf{u}_{re}$ と，編集前の流れ $\mathbf{u}_{ori}$ の Ω の外にある部分が滑らかに接続するように計算を行う．PML は図内の Ω_L, Ω_R の領域に適用される．それぞれの領域において，$\mathbf{u}_{re}$ は $\mathbf{u}_{ori}$ との差分が Ω_L, Ω_R それぞれの外側端に近づくにつれて 0 となるように計算される．そして，最終的に Ω の外の流れ $\mathbf{u}_{ori}$ と内部の流れ $\mathbf{u}_{re}$ とを合わせて結果の流れ $\mathbf{u}$ が得られる．計算方法の詳細については [1] を参照いただきたい．

既存手法では編集領域 Ω は固定されていたのに対し，我々は Ω を適応的に変更する．提案手法では，次フレームでの境界 x_L 及び x_R を $\mathbf{u}_{ori}$ と $\mathbf{u}_{re}$ の現在の差分によって移動させる．x_L と x_R の位置を決めるにあたり，我々は以下の式を最小化する．

$$E(x_a) = E_d(x_a) + \alpha|x_a - x_c| \tag{1}$$

$$E_d(x_a) = -\sum_{x=x_c}^{x_a} ||\mathbf{u}_{re}(x,y) - \mathbf{u}_{ori}(x,y)|| \tag{2}$$

ここで，a は L または R を示し，x_c は現在の編集領域 Ω の中心となる位置を表す．また，y は流体シミュレーションに用いられる格子の鉛直方向の座標である．y には各 x について以下の式を最大化するものを選ぶ．

$$y = \arg\max_{y_j} ||\mathbf{u}_{re}(x, y_j) - \mathbf{u}_{ori}(x, y_j)|| \tag{3}$$

x の定義域は中心 x_c から領域境界 $x_L - w$ 及び $x_R + w$ までとなる．ここで，w は PML 幅を表す．我々は式 (1) に記述された最小化問題を各フレームにおいて全探索で解き，領域境界の位置を図 2 に示すように更新する．x_c, x_L, x_R の初期位置と x_c の移動量はユーザが定義する．

式 (1) の第一項は $\mathbf{u}_{re}$ と $\mathbf{u}_{ori}$ の差分の総和が最大になる位置を見つけるように働く．そのため，我々は差分の総和の負をとったものを最小化した．予備実験において，我々は計算領域を可能な限り小さくするため，単純に速度の差分を最小化していたが，これでは Ω が必要以上に小さくなることがわかった．これは，領域境界 $x_a \pm w$ 付近に領域中心 x_c 付近での差分よりも大きな差分がある場合，それを無視してしまうためである．この場合，$\mathbf{u}_{re}$ がたった一フレームの間に大きく異なる速度を持つ $\mathbf{u}_{ori}$ で置き換えられることになるため，大きな差分が無視された領域での速度が時間的に不連続なものとなってしまう．こうした不連続性により結果に視覚的なアーティファクトが生じることを実験で確認した．このため，我々は代わりに速度の差分の総和を用いた．また，式 (1) の第二項は編集領域をできるだけ小さくするための項である．各境界の位置を決定した後，得られた新しい編集領域 Ω 内で Bojsen-Hansen らの手法を用いて液体の流れを再シミュレーションし，提案手法による結果の流れ $\mathbf{u}_{our}$ を得る．

式 (1) を最小化する際，新しい境界の位置を前のフレームの境界位置を基にそれぞれ以下のように制限する．

$$x_L(n) < \min\left(x_c - dx, x_L(n-1) + \frac{w}{2}\right)$$

$$\max\left(x_c + dx, x_R(n-1) - \frac{w}{2}\right) < x_R(n)$$

ここで，n はフレーム番号である．dx はユーザが経験的に与えるオフセットで，Ω が Ω_L と Ω_R のみを含む，あるいは Ω_L と Ω_R の領域が重複するほど近接してしまうことを防ぐ．また，これらの制約には x_a が PML 幅 w を越えて大きく移動することを防ぐ目的がある．このような大きな移動が起こると，計算領域内部の速度差の大きな部分が領域 Ω_R，Ω_L に含まれてしまう場合がある．大きな速度差に対して PML を用いたシミュレーションを適用すると，予期せぬ速度が発生してしまうことを実験により確認した．そのため上記の制約により，なるべく大きな速度差を領域 Ω_R，Ω_L に含まないように手法を設計した．

三次元のシミュレーションでは，x と y の二つの軸に加え，さらに奥行きのために z 軸を追加する．編集領域 Ω の奥行きを考慮するため，式 (1) に PML 領域 Ω_F, Ω_B 及び境界 x_F, x_B が加わる．つまり，三次元の場合には式 (1) を x_L, x_R, x_F, x_B の四つの変数について最小化することになる．

式 (3) に関しては，x_L と x_R が x 軸について動く場合は，y と z を式 (2) の計算のために決定する必要があり，これは Ω 中の各 x について，x を通る yz 平面上で探す．また，x_F と x_B が z 軸について動く場合は，x と y を各 z を通る xy 平面上で探す．

表 1 格子サイズ, 式 1 のパラメータ α，計算時間の表．T_{our} と T_{prev} は提案手法と既存手法の各フレームの計算時間の平均を表す．

図	格子サイズ	α	T_{our}	T_{prev}
4	400×200	0.03	0.117s	0.188s
5	$250 \times 75 \times 150$	0.01	10.2s	21.2s
6	$250 \times 75 \times 150$	0.015	7.6s	17.4s

4. 実験結果

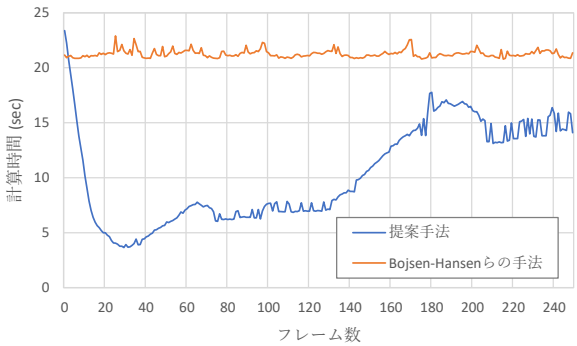

図 3 図 5 における各フレームの計算時間

既存手法と比較した場合の我々のアルゴリズムの有効性を評価するため，我々はシミュレーション領域が広範囲に移動するような実験例を設定した．全ての実験は 2.80GHz の Intel Core i7-7700HQ CPU と 16.0GB の RAM を用いて行った．我々は提案手法の結果を，既存手法において十分に広いシミュレーション領域を取った場合の結果と比較する．体積の誤差を補正するため，Bojsen-Hansen ら [1] は particle level set 法を用いたのに対し，我々は Kim ら [6] の提案した，目標体積を必要とする体積補正手法を提案手法，既存手法両方の実験に用いた．この場合，編集前の流れと編集後の流れの間で同じ部分での体積の差分が大きくなると，二つの流れの表面が滑らかに接続しなくなってしまう．我々は，それぞれの体積を毎フレームブレンドすることでこの効果を抑制した．以降の全ての実験において，PML 幅は編集前の流れのシミュレーション領域の水平方向の幅の 5%とした．表 1 はシミュレーションパラメータと提案，既存それぞれの計算時間を示したものである．

二次元シミュレーション：領域中心 x_c が端から端まで

移動し，球状の液滴が一定の時間間隔で x_c から落下する例を二次元でシミュレーションした．図 4 は提案手法と既存手法の結果の比較である．この例において提案手法は，シミュレーション領域を適応的に変化させることで計算時間を減らしつつ，適応的でない場合とほぼ同じ視覚的クオリティを達成している．

三次元シミュレーション：三次元でも，図 5 のように，二次元の実験例と同じような場合をシミュレーションした．また，我々はそれぞれの手法のパフォーマンスを図 3 の通り比較した．この例では，提案手法は既存手法と比較し，より少ない計算時間でほぼ同じ視覚的クオリティを達成している．上記の例に加え，我々は図 6 に示すような状況をシミュレーションした．この例では，x_c は楕円形の経路上を移動し，前の例と同じように x_c から液滴が落下する．x_c 付近では我々の手法と既存手法の間で視覚的に大きな差はないのに対し，境界付近では提案手法でのみ界面に複数の段差が生じている．既存手法でも似たような問題は起きているが，こちらの場合では段差はシミュレーション領域の最も外側でのみ生じる．これらのアーティファクトについて，以下で更に詳しく述べる．

三次元の実験例ではシミュレーション領域境界付近の界面に段差のアーティファクトが生じる．particle level set 法を使っていないことがこうしたアーティファクトの一つの原因となっている可能性がある．このような段差は既存手法 [1] においても生じているが，編集領域の境界において，編集後と編集前のそれぞれの液体表面を表すメッシュを後処理でブレンドすることでこの問題を解決している．我々もこの処理を加えることでこうしたアーティファクトを回避できる可能性がある．

5. 結論

本稿では，液体の流れの局所的な再シミュレーションを効率化するために，シミュレーション領域を適応的に変更する手法を提案した．我々のアルゴリズムでは，再シミュレーションにより編集された流れと編集前の流れの差分によって境界の位置を自動的に決定する．シミュレーション領域を最適な大きさと位置に調整することによって，特に重要な部分が広範囲を移動するような状況を再シミュレーションする計算時間を減らすことができる．

今後の研究としては，パラメータ α の影響の検証が考えられる．このパラメータは，大きくすると $\mathbf{u}_{re}$ と $\mathbf{u}_{ori}$ の差分の影響が小さくなり計算領域が縮まりやすくなる．一方，小さくするとわずかな速度の差分でも大きく影響するため縮まりにくくなる．そのため，α の違いにより生成される結果が大きく変わってしまう場合がある．この問題に対して，再度全体をシミュレーションした場合となるべく近い結果を得られ，かつコストが削減されるような適切な α を自動で決定する方法の開発が今後の課題としてあげられる．

参考文献

[1] Bojsen-Hansen, M. and Wojtan, C.: Generalized Non-reflecting Boundaries for Fluid Re-simulation, *ACM Trans. Graph.*, Vol. 35, No. 4, pp. 96:1–96:7 (online), DOI: 10.1145/2897824.2925963 (2016).

[2] Söderström, A., Karlsson, M. and Museth, K.: A PML-based nonreflective boundary for free surface fluid animation, *ACM Transactions on Graphics*, Vol. 29, No. 5, p. Article 136 (2010).

[3] Kim, T. and Delaney, J.: Subspace fluid re-simulation, *ACM Transactions on Graphics*, Vol. 32, No. 4, p. Article 62 (2013).

[4] Stomakhin, A. and Selle, A.: Fluxed animated boundary method, *ACM Transactions on Graphics*, Vol. 36, No. 4, p. Article 68 (2017).

[5] Sato, S., Dobashi, Y. and Nishita, T.: Editing Fluid Animation Using Flow Interpolation, *ACM Transactions on Graphics*, Vol. 37, No. 5, p. Article 173 (2018).

[6] Kim, B., Liu, Y., Llamas, I., Jiao, X. and Rossignac, J.: Simulation of bubbles in foam with the volume control method, *ACM Transactions on Graphics*, Vol. 26, No. 3, p. Article 98 (2007).

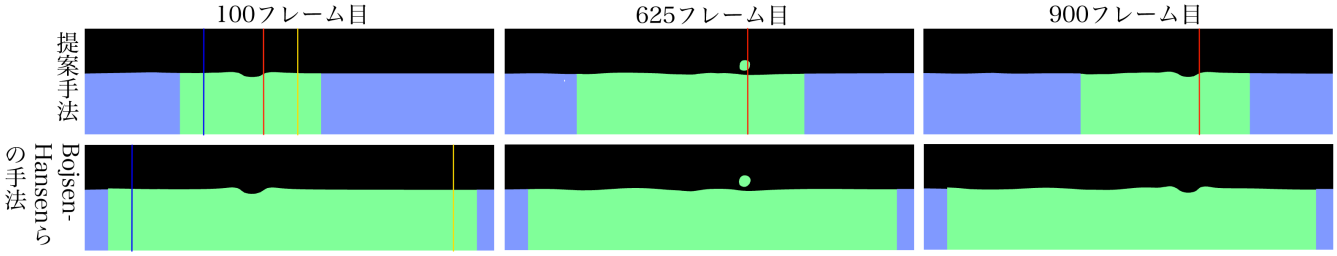

図 4 二次元での提案手法（上）と既存手法（下）の結果の比較．それぞれの線分は x_c（赤），x_L（青），x_R（黄）を表す．

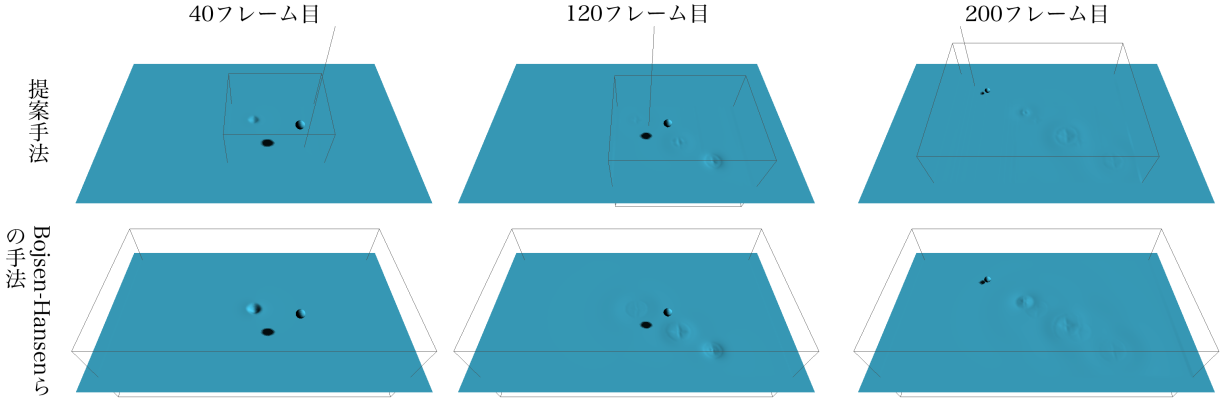

図 5 三次元での結果の比較．黒枠は編集領域を，編集領域の中にある黒い線分は x_c を表す．

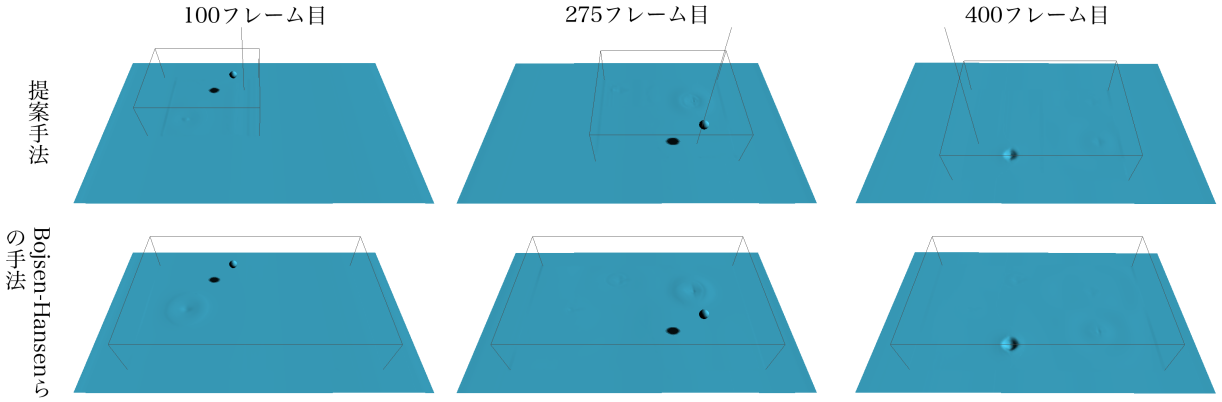

図 6 三次元での結果の比較．この例では，x_c は楕円状に動く．