Vol.2014-CG-155 No.13
2014/6/28

# 極座標補間による異なる流体流れ場の補間手法

佐藤 周平[1,a)]　土橋 宜典[1,2,3]　山本 強[1]　西田 友是[3,4]

**概要**：近年，CG 技術の発達により写実的な流体映像が映画やゲームなどに多く用いられている．これら流体映像は，流体シミュレーションを用いて作成される場合が多いが，リアルな映像の作成には非常に高い計算コストがかかる．そのため，作成したアニメーションを再度調整したい場合，もう一度シミュレーションを実行しなければならず，微調整であっても多くの時間が必要となってしまう．そこで，本研究では，ある 2 つの流体の速度場が与えられたときにその間のデータを補間することで，再調整を低コストで行うための手法を提案する．提案法では，2 つの速度場に対して，その特徴が類似している箇所に制御点を配置し，対応する制御点同士の中間を極座標補間により算出する．そして，補間した制御点に従って速度場を変形することで，2 つの速度場の中間にあたる尤もらしい速度場を補間する．

**キーワード**：流体シミュレーション，流れ場の補間，極座標補間

# An Interpolation Method of Different Flow Fields using Polar Interpolation

Syuhei Sato[1,a)]　Yoshinori Dobashi[1,2,3]　Tsuyoshi Yamamoto[1]　Tomoyuki Nishita[3,4]

***Abstract:*** Recently, realistic fluid animations are often used in movies or video games. These fluid animations are usually generated by using fluid simulation, but computational costs are too expensive for creating realistic animations. Therefore, if the user would like to obtain results different from simulated results, the user must execute fluid simulation repeatedly and must take a long time. To address this problem, this paper proposes a method for interpolating between two different velocity fields of fluid. By using our system, the user can create various fluid animations from the two input velocity fields, without executing fluid simulation. In our method, control points are placed at the positions where features of these two velocity fields are similar, we interpolate between these control points by using polar interpolation. Then, by deforming the two velocity fields following the interpolated control points, plausible velocity fields are interpolated.

***Keywords:*** fluid simulation, interpolation of flow fields, polar decomposition interpolation

## 1. はじめに

近年，コンピュータグラフィックス (CG) 技術の発達により，様々な自然現象の写実的なアニメーションが，映画やゲームなどで多く見られるようになってきた．その中でも流体現象は，屋内外問わずシーンを構成する重要な要素の 1 つであり，様々なシーンで利用されている．流体現象は，物理方程式に基づいたシミュレーションを行うことにより写実的な表現が可能である．このような方法により，煙や水，炎などをシミュレーションするための手法が数多く提案されている [1][2][4][5]．しかし，流体解析を利用したシミュレーションは計算コストが非常に高い点が問題となる．また，所望の映像を作成するためには，一般に，数多くのパラメータを試行錯誤的に決定するといった煩雑な作業を必要とする．そのため，所望の映像を得るまでには，高コストのシミュレーションを繰り返し実行しなければな

[1] 北海道大学
Hokkaido University, Sapporo, Hokkaido 060–0814, Japan
[2] JST CREST
[3] UEI Research
[4] 広島修道大学
Hiroshima shudo University, Hiroshima, Hiroshima 731–3195, Japan
[a)] sato@ime.ist.hokudai.ac.jp

らず，膨大な時間がかかってしまう．

上記の問題を解決するために，事前に計算した流体速度場のデータベースを用いて実行時のシミュレーションの高速化を可能とする手法が提案されている [6][7]．これらの手法により，流体シミュレーションを高速に実行できる．しかし，前計算データに存在しない流れ場は作成できず，様々な流れを作成可能とするためには，膨大な前計算データとその作成のための計算時間が必要である．また，上記のアプローチに基づき，高速に再シミュレーションを行う手法が提案されている [3]．この手法により，シミュレーションにより作成された流体の速度場に対し，そのパラメータを変えた際の流れ場を再度高速にシミュレーションすることができる．しかしこの手法では，単一のデータを対象としており，例えば流れの大域的な方向を変えるなど，大きな変更を行うことは出来ない．

そこで本研究では，複数の流れ場間においてその中間を補間することで，シミュレーションを再度実行せずに大域的な変更を可能とする手法を提案する．これにより，ユーザは再度シミュレーションを行うことなく，大域的な流れを調整することができる．提案法では，2 つの流れ場において特徴の類似する箇所に対応点を配置し，その中間点を極座標補間により算出する．そして，その補間情報を用いて，速度場を変換することで，補間速度場を生成する．本稿では，実験として 2 次元の煙のシミュレーションを対象とする．提案法により，流体解析により生成した速度場を再利用して大域的な流れを変化させたアニメーションを生成することができる．

## 2. 関連研究

Stam は，タイムステップを大きくとった場合でも，Navier-Stokes 方程式を安定的に解く方法を提案し，CG において実用的な流体シミュレーション手法を確立した [5]．Stam の手法以降，様々な流体現象を対象とした数多くの解析手法が提案されている．それらの手法の詳細は [1] にまとめられている．しかし，流体シミュレーションは計算コストが非常に高く，結果を生成するまでに多大な時間がかかってしまう．また，所望の流体の動きを作成するためにはシミュレーションパラメータを試行錯誤的に調整しなければならない．

シミュレーション結果から前計算においてデータベースを作成し，それを用いて実行時のシミュレーション時間を大幅に削減するための手法が提案されている [6][7]．これらの手法では，様々な初期条件，パラメータにおいてシミュレーションを行ったデータを用意し，そのデータ全体に対して主成分分析を適用する．これにより得られた主成分を基底関数とし，基底空間で Navier-Stokes 方程式を計算することで，高速に流れ場を計算することができる．しかし，主成分分析により基底を作成しているため，前計算データに含まれていないような流れ場については作成することができない．また，様々な流れ場を作成可能とするには，膨大なシミュレーションデータと，その計算に係る時間，さらに主成分分析にも多くの計算時間が必要である．

上記のアプローチに基づき，シミュレーションのパラメータを変更し，高速に再シミュレーションする手法が開発されている [3]．この手法は，シミュレーションにより作成された単一のデータセットに対し主成分分析を適用して求めた基底空間で計算を行うことで，パラメータを変化させた場合の再計算が可能である．ただし，非線形項である移流項の計算は格子空間で行うが，基底空間から格子空間に変換し，再度基底空間へ戻すには，解析空間全体の積分が必要であるため，大きな計算コストがかかる．そのため，求積法を用いることで，領域全体を積分せずとも近似的な解を算出可能とし，高速な再シミュレーションを実現している．しかし，この手法では，単一のデータのみで基底空間を作成するため，元のデータに含まれていないような流れを作成することは出来ない．本手法では，複数のデータからその中間を補間することで，元のデータにない流れ場の作成を可能とする．

## 3. 流体シミュレーション

本稿では，非圧縮性の流体シミュレーションを行う．流体の動きは，次の Navier-Stokes 方程式（以下，NS 方程式）を解くことで計算される．

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u}\cdot\nabla)\mathbf{u} - \frac{1}{\rho}\nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f} \qquad (1)$$

$$\nabla\cdot\mathbf{u} = 0 \qquad (2)$$

$\mathbf{u}$ は流体の速度場，$\rho$ は流体の密度，$p$ は圧力，$\nu$ は動粘性係数，$\mathbf{f}$ は重力や風などの外力である．式 (1) は速度場の時間発展を表す方程式であり，右辺第 1 項から，移流項，圧力項，拡散項，外力項と呼ばれる．また，非圧縮性流体を扱う場合のみ式 (2) が成り立ち，この式は連続の式と呼ばれる．また，Fedkiw らが提案した手法 [2] を用いて乱流を付加する．

以下では上記の方程式を用いて煙をシミュレーションする方法を説明する．NS 方程式により得られた速度場にしたがって煙の密度を以下の式によって移流させる．

$$\frac{\partial D}{\partial t} = -(\mathbf{u}\cdot\nabla)D + D_s \qquad (3)$$

$D$ は煙の密度，$D_s$ は煙の発生源から追加される密度量を表す．提案手法は，速度場のみを補間し，煙の密度は補間された速度場にしたがって移流させることで動きを解析する．そのため，式 (1)，(2) による速度場の解析は，補間元となる速度場を生成する際の 2 次元流体シミュレーションでのみ行う．

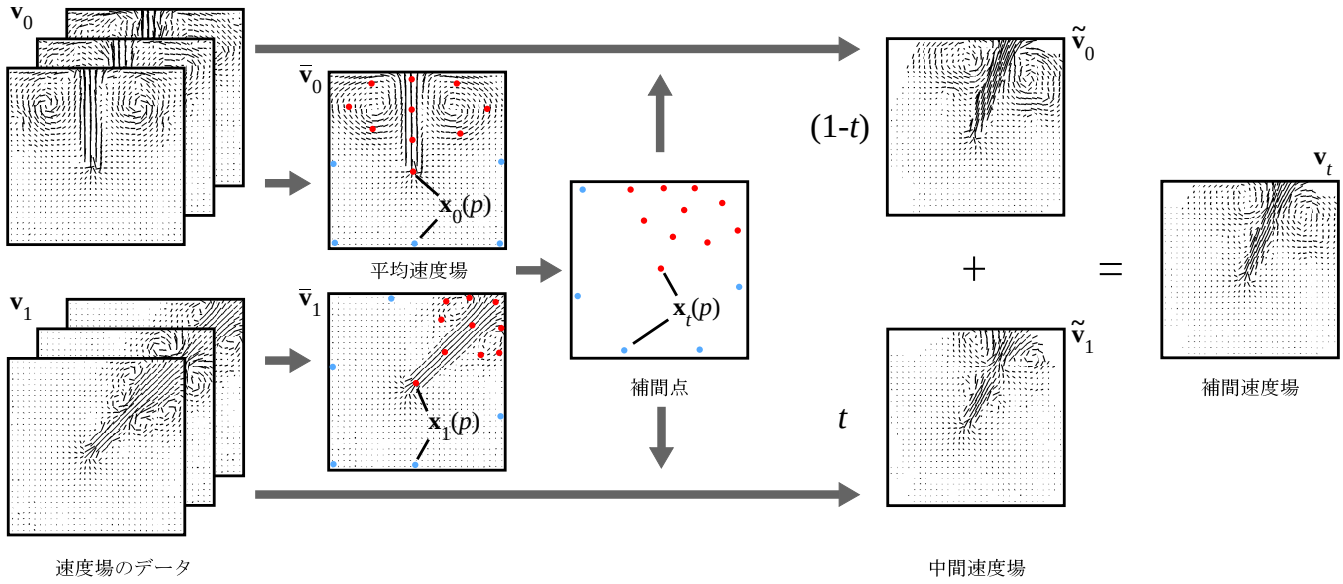



**図 1** 提案手法の概要

## 4. 提案手法の概要

提案法の概要を図 1 に示す．本手法では，まず前節の流体シミュレーションを用いて，2 つの異なる流体の速度場を生成する．2 つの流れ場それぞれについて，時間方向の平均をとり，平均速度場を算出する（図 1 の平均速度場）．この 2 つの平均速度場間において，特徴の類似している箇所に対応点を配置する（図 1 の赤点および青点）．本稿では，この対応点の配置は手動で行う．なお，ここでは点の対応関係を分かりやすくするために，主な流れの部分と解析空間の境界部分とで点の色を分けている．ここで，本稿では，速度の大きさが一定値以上の箇所を主な流れの部分として定義している．次に配置した対応点それぞれに対して，極座標補間により，中間の状態 $t(0 < t < 1)$ を算出する（図 1 の補間点）．この時，対応点の位置における速度の方向から，中間の状態での速度の方向も合わせて算出しておく．そして，補間された位置および速度の方向に基づいて，格子を変形させ，2 つの入力についてそれぞれ中間速度場を生成する（図 1 の中間速度場）．最後に $t$ の値に応じて 2 つの中間速度場をブレンドすることで，補間速度場を合成する．そして，合成された速度場に従って煙の密度を移流させ，その値を可視化する．以下，提案法の各処理について詳しく説明する．

## 5. 対応点の配置

中間の状態の補間を行うために 2 つの速度場に対応点を配置する．まず，流体シミュレーションにより作成された 2 つの補間元の速度場 $\mathbf{v}_0(n)$ および $\mathbf{v}_1(n)$ それぞれについて，時間方向に平均をとった速度場（平均速度場）$\bar{\mathbf{v}}_0$ および $\bar{\mathbf{v}}_1$ を算出する．ここで，$n$ は，$n = 0, 1, 2, ..., N-1$ であり，$N$ は速度場データの全フレーム数を表わす．速度場の平均をとることで，流れ場データの大域的な特徴を得ることができる．

続いて，平均速度場 $\bar{\mathbf{v}}_0$ および $\bar{\mathbf{v}}_1$ において，手動で対応点 $\mathbf{x}_0(p)$ および $\mathbf{x}_1(p)$ を配置する．ここで，$p$ は，$p = 0, 1, 2, \cdots, P-1$ であり，$P$ は対応点の組みの総数である．この時，図 1 の平均速度場の部分に示すように，2 つの流れ場の特徴が類似している箇所が対応するよう点を配置する．この対応点は，平均の速度が一定以上大きい箇所や解析空間の境界に主に配置する．次節では，これらの対応点群を基に中間の状態を補間する．

## 6. 極座標補間による中間状態の生成

前節で配置した対応点 $\mathbf{x}_0(p)$ および $\mathbf{x}_1(p)$ から，極座標補間を適用することで，2 つの点の中間の状態 $\mathbf{x}_t(p)$ を生成する．まず，$\mathbf{x}_0(p)$ および $\mathbf{x}_1(p)$ を極座標形式で表現すると以下のようになる．

$$\mathbf{x}_0(p) = (r_0(p)\cos(\theta_0(p)), r_0(p)\sin(\theta_0(p)))$$
$$\mathbf{x}_1(p) = (r_1(p)\cos(\theta_1(p)), r_1(p)\sin(\theta_1(p)))$$

ここで，$r_0$，$r_1$ は原点からの半径，$\theta_0$，$\theta_1$ は単位ベクトル $(1, 0)$ を $\theta = 0$ とした時の角度である．本稿では，原点は解析空間の左下に設定している．上式の $r_0(p)$ および $r_1(p)$，$\theta_0(p)$ および $\theta_1(p)$ を用いて，$\mathbf{x}_t(p)$ を極座標で表現した場合の $r_t(p)$ および $\theta_t(p)$ を以下の式のように，線形補間により算出する．

$$r_t(p) = (1-t)r_0(p) + tr_1(p)$$
$$\theta_t(p) = (1-t)\theta_0(p) + t\theta_1(p)$$

ここで，$\mathbf{x}_0(p)$ および $\mathbf{x}_1(p)$ を単純に線形補間した場合と極座標へ変換した後に補間した場合との違いを図 2 に示す．

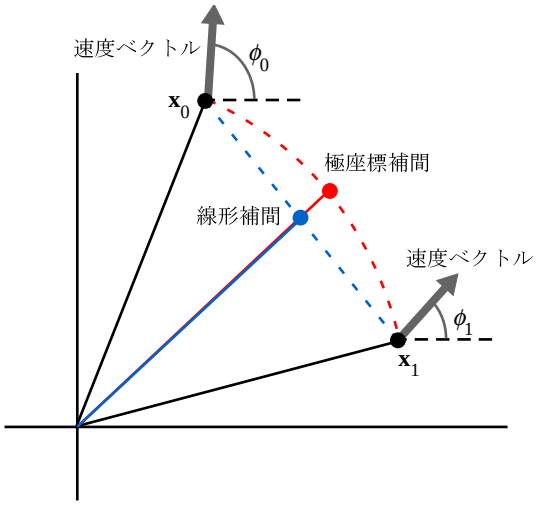



**図 2** $\phi$ の定義および補間方法の比較

図 2 の例のように，単純な線形補間（青点）では，中間の状態において補間点までの距離が変わってしまうが，極座標に変換した後に補間（赤点）することで，補間点までの距離を変化させることなく中間の状態を生成することができる．

また，中間の状態において，流れの方向が変わる場合，速度の方向も変化させなければならない．そこで，対応点の位置 $\mathbf{x}_0(p)$ および $\mathbf{x}_1(p)$ における速度ベクトルの方向 $\phi_0(p)$ および $\phi_1(p)$(定義については図 2 参照) から $\phi_t(p)$ を以下のように算出する．

$$\phi_t(p) = (1-t)\phi_0(p) + t\phi_1(p)$$

この補間値を用いて，次節にて $\mathbf{v}_0(n)$ および $\mathbf{v}_1(n)$ の中間の速度場 $\mathbf{v}_t(n)$ を合成する．

## 7. 補間速度場の合成

前節で求めた $r_t(p)$ および $\theta_t(p)$，$\phi_t(p)$ を用いて，補間速度場 $\mathbf{v}_t(n)$ を合成する．本稿では，$\mathbf{v}_0(n)$ を変換した速度場 $\tilde{\mathbf{v}}_0(n)$ および $\mathbf{v}_1(n)$ を変換した速度場 $\tilde{\mathbf{v}}_1(n)$ から以下の式のように補間速度場 $\mathbf{v}_t(n)$ を算出する．

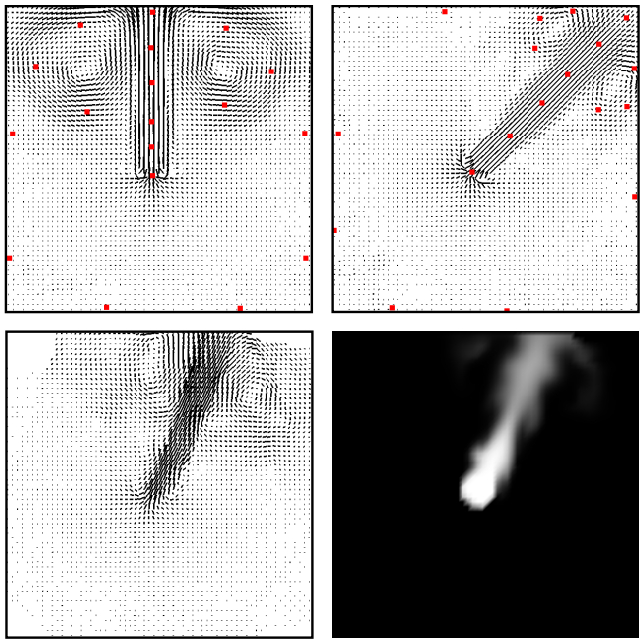
$$\mathbf{v}_t(n) = (1-t)\tilde{\mathbf{v}}_0(n) + t\tilde{\mathbf{v}}_1(n) \qquad (4)$$

以下では，$\tilde{\mathbf{v}}_0(n)$ および $\tilde{\mathbf{v}}_1(n)$ の算出方法を説明する．

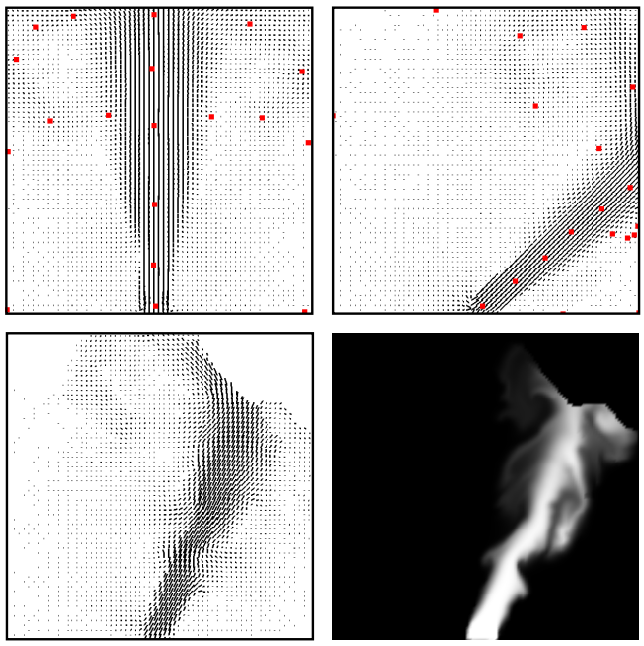
まず，$r_t(p)$ および $\theta_t(p)$ を用いて，以下の式により元データの $\mathbf{x}$ の位置に存在する速度を $\tilde{\mathbf{x}}_0$ および $\tilde{\mathbf{x}}_1$ へ移動する．

$$\tilde{\mathbf{x}}_0 = ((r + r_{\mathbf{x},0})\cos(\theta + \theta_{\mathbf{x},0}), (r + r_{\mathbf{x},0})\sin(\theta + \theta_{\mathbf{x},0}))$$
$$\tilde{\mathbf{x}}_1 = ((r + r_{\mathbf{x},1})\cos(\theta + \theta_{\mathbf{x},1}), (r + r_{\mathbf{x},1})\sin(\theta + \theta_{\mathbf{x},1}))$$

ここで，$r$ および $\theta$ は $\mathbf{x} = (r\cos\theta, r\sin\theta)$ から得られる．また，$r_{\mathbf{x},a}$ および $\theta_{\mathbf{x},a}(a = 0\text{or}1)$ は以下の式により算出する．



**(a)**



**(b)**

**図 3** 適用例

$$r_{\mathbf{x},a} = \sum_p \lambda_{\mathbf{x},a}(p) r_a(p)$$
$$\theta_{\mathbf{x},a} = \sum_p \lambda_{\mathbf{x},a}(p) \theta_a(p)$$
$$\lambda_{\mathbf{x},a}(p) = \frac{w_{\mathbf{x},a}(p)}{\sum_q w_{\mathbf{x},a}(q)}$$
$$w_{\mathbf{x},a}(p) = \frac{1}{\| \mathbf{x}_a(p) - \mathbf{x} \|}$$

ここで，$q$ は $q = 0, 1, 2, \cdots, P-1$ である．加えて元データの位置 $\mathbf{x}$ から中間速度場の位置 $\tilde{\mathbf{x}}$ へ変換する際の速度の回転角 $\phi_{\mathbf{x},a}$ も上記の $\lambda_{\mathbf{x},a}$ を用いて以下のように算出する．

$$\phi_{\mathbf{x},a} = \sum_p \lambda_{\mathbf{x},a}(p)\phi_a(p)$$

この回転角 $\phi_{\mathbf{x},a}$ により速度の方向を回転させ，$\tilde{\mathbf{x}}_a$ の位置に速度を移動することで，$\tilde{\mathbf{v}}_0(n)$ および $\tilde{\mathbf{v}}_1(n)$ を算出し，速度場 $\mathbf{v}_t(n)$ を合成する．

## 8. 実験結果

提案手法を適用して生成した結果を図 3 に示す．実験環境は，CPU が Intel Core i7 2600K（メモリ 16GB），GPU が NVIDIA GeForce GTX 680 となっている．図 3(a) ではシミュレーションの格子数が $64 \times 64$，図 3(b) では $128 \times 128$ である．図 3(a)，(b) どちらの例についても，上段の速度場が補間元の 2 つの速度場の平均をとったものであり，指定した対応点を赤色の点により示している．下段は，左が提案法により生成した補間速度場のある 1 フレームであり，右が補間速度場を用いて煙の密度を移流させその値を可視化したものである．どちらの例についても中間の状態は，$t = 0.5$ の場合の結果を作成した．提案手法により，上段の補間元の速度場の中間の状態が補間出来ているのがわかる．しかし，図 3(b) の結果の右上の部分において，煙が解析空間の境界に到達する前に途切れてしまっている．これは，$r_t(p)$ を線形補間により求めているため，両補間元の流れの主軸の長さよりも，補間速度場の主軸の長さを長くすることができないことが原因である．この問題については，スプライン関数などにより $r_t(p)$ の取り得る値の範囲を線形補間の直線上以外に設定可能とすることで，解決できる可能性がある．

## 9. まとめと今後の課題

本稿では，2 つの流体速度場の中間を極座標補間により算出することで，2 つの流れの間を補間する手法を提案した．2 つの流れ場の時間方向での平均的特徴が類似している箇所に対応点を配置し，その対応点の関係から極座標補間により中間の状態を生成した．そして，実験において，2 次元のシミュレーションについて，提案法の有効性が確認できた．

今後の課題としては，対応点配置の自動化と物理法則の考慮が挙げられる．本稿では，極座標補間により中間を生成するために必要な対応点を手動で配置した．今回は，2 次元かつシンプルな例を用いたため，手動での配置も可能であったが，複雑な流れや 3 次元への拡張を行った際に，手動で対応点を配置するのは不可能である．この問題に対し，ベクトル場をその特徴から領域分割する手法を応用することで，自動化が可能ではないかと考えられるため，今後実験を行う予定である．また，本手法では物理法則を考慮せずに，速度場の変換を行っているため，合成される補間速度場は物理法則（式 (1) および (2)）を満たさない場合が多い．そのため，物理法則を保った速度場の変換方法を今後開発していく予定である．

**謝辞** この研究は独立行政法人科学技術振興機構，CREST によりサポートされています．

### 参考文献

[1] R. Bridson : *Fluid Simulation for Computer Graphics*, AK Peters (2008).
[2] R. Fedkiw, J. Stam, and H.W. Jensen : Visual simulation of smoke, In *Proc. SIGGRAPH 2001*, 15-22 (2001).
[3] T. Kim, and J. Delaney : Subspace fluid re-simulation, *ACM Transactions on Graphics 32*, 4, Article 62 (2013).
[4] D.Q. Nguyen, R. Fedkiw, H.W. Jensen : Physically Based Modeling and Animation of Fire, In *Proceeding of ACM SIGGRAPH 2002*, 721-728, (2002)
[5] J. Stam : Stable fluids, In *Proceedings of ACM SIGGRAPH 1999, Annual Conference Series*, 121-128 (1999).
[6] A. Treuille and A. Lewis and Z. Popovic : Model reduction for real-time fluids, *ACM Transactions on Graphics 25*, 3, 826-834 (2006).
[7] M. Wicke and M. Stanton and A. Treuille : Modular bases for fluid dynamics, *ACM Transactions on Graphics 28*, 3, Article 39 (2009).