煙シミュレーションのための部分空間法の高速化

Accelerated Subspace Method for 3D Smoke Simulation

情報工学専攻　須之内　俊樹
Toshiki SUNOUCHI

概要

流体シミュレーションに部分空間法を適用することで，各時刻のシミュレーションを大幅に高速化することができるが，前処理に膨大な計算時間がかかることが課題である．そこで本研究は，前処理をシミュレーション時間に対して分割し，前処理全体の計算時間を削減する手法を提案する．計算機実験の結果，前処理のボトルネックとなっている基底計算のための特異値分解の高速化を確認し，前処理の高速化に有効であると示すことができた．また，計算結果の精度について，基底の累積寄与率や相対誤差を用いて評価をし，分割数によるシミュレーションのレンダリング結果の変化を確認した．

キーワード: 流体シミュレーション，部分空間法，スナップショット固有直交分解，立体求積法．

1　はじめに

流体シミュレーションは，工学およびコンピュータグラフィックスの分野において広く利用されている．コンピュータグラフィックスの分野では，現実的な水や煙のアニメーションを生成し，映像作品やゲームの演出に活用されている．コンピュータグラフィックスにおいては，物理的に厳密なシミュレーションよりも，視覚的に自然な動きを高速に計算することが求められる．

近年，より高解像度の流体シミュレーションが求められる中で，計算負荷の増大が課題となっている．そのため，高速化手法の研究が活発に進められており，特に大規模並列計算技術の適用が重要視されている．多くの高速化手法が GPU 上での実装を前提として検討されている．

計算負荷の削減手法の一つとして，前処理によりシミュレーション計算に使用する行列の次元を削減する部分空間法が存在する．部分空間法を適用することで，シミュレーション計算そのものは高速化されるが，前処理に要する計算負荷やメモリ使用量が大きくなるという課題がある．本研究では，部分空間法の計算負荷を削減することにより，前処理に要する時間の短縮を目的とする．

2　関連研究

2.1　煙の流体シミュレーション

下記の式 1 で表される式を，ナビエ・ストークス方程式とよび，これは流体力学の支配方程式である．式 1 の右辺の第一項を移流項，第二項を圧力項，第三項を粘性項，第四項を外力項と呼ぶ．コンピュータグラフィックスにおける気体の外力には重力のほか，温度差による対流を発生させる力や．気体の渦運動を生成する力などが含まれ，これらを適用することで，気体の自然な挙動を再現することが可能となる．式 2 は流体の非圧縮性条件であり，煙のシミュレーションにおいて流速が著しく大きい場合を除き，利用することが可能である．

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

$$\nabla\cdot\boldsymbol{u}(t) = 0 \quad (2)$$

コンピュータグラフィックスにおいては，ナビエ・ストークス方程式の数値解法の中でも，全ての項を個別に分割して解く手法としてフラクショナルステップ法 [1] が一般的に用いられている．フラクショナルステップ法は，中間子 $\boldsymbol{u_0}$ から $\boldsymbol{u_3}$ を用いて，ナビエ・ストークス方程式を以下のように分割する．

$$\boldsymbol{u_0} = \boldsymbol{u}(t) - \Delta t\boldsymbol{f}$$

$$\boldsymbol{u_1}(\boldsymbol{x}) = \boldsymbol{u_0}(\boldsymbol{x} - \Delta t\boldsymbol{u_0})$$

$$\boldsymbol{u_2} = \boldsymbol{V}\boldsymbol{u_1}$$

$$\boldsymbol{b} = \boldsymbol{W}\boldsymbol{u_2}$$

$$\boldsymbol{p} = \boldsymbol{A^{-1}}\boldsymbol{b}$$

$$\boldsymbol{u}(t+1) = \boldsymbol{u_3} = \boldsymbol{u_2} - \boldsymbol{Y}\boldsymbol{p}$$

ここで，$\boldsymbol{V}$，$\boldsymbol{A}$ は離散ラプラシアン行列を用いて ∇^2 を離散化し，それぞれ定数の係数を乗算したものであり，$\boldsymbol{W}$，$\boldsymbol{Y}$ は勾配演算子 ∇ を離散化したものである．流速に従って煙の密度を計算し，ボリュームレンダリングによって可視化することで，煙のアニメーションを作成で

きる．ボリュームデータを三次元的な広がりの様子を把握できるようにレンダリングする手法をボリュームレンダリングという．描画対象の表面のみをレンダリングするサーフェスレンダリングとは異なり，光の散乱や吸収などの光学的性質を利用して，描画対象の内側のデータも出力画像に反映させる．

2.2 部分空間法

部分空間法は，ベクトル空間をより低次元の部分空間に射影し，数値シミュレーションにおける計算負荷を軽減する手法である．流体シミュレーションにおける部分空間法は，モデル縮約やモデル縮退とも呼ばれている．扱うベクトルデータやシミュレーションの計算を低次元の部分空間に射影して行い計算負荷を軽減する．

まず，線形項の計算手法 [2] について説明する．シミュレーションの前処理として，n 次元ベクトル $\boldsymbol{x}$ を r 次元ベクトル $\widetilde{\boldsymbol{x}}$ に変換する，$n \times r$ 直交行列 $\boldsymbol{A}$ を考える．直交行列の性質から，以下の 2 式が成り立つ．

$$\widetilde{\boldsymbol{x}} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{x} \qquad\qquad \boldsymbol{x} = \boldsymbol{A}\widetilde{\boldsymbol{x}}$$

また，$\widetilde{\boldsymbol{x'}} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{x'}$ を満たす n 次元ベクトル $\boldsymbol{x'}$，r 次元ベクトル $\widetilde{\boldsymbol{x'}}$ について，以下の 2 式が成り立つような，任意の $m \times n$ 行列 $\boldsymbol{F}$，任意の $m \times r$ 行列 $\widetilde{\boldsymbol{F}}$ を考える．

$$\boldsymbol{x'} = \boldsymbol{F}\boldsymbol{x} \qquad\qquad \widetilde{\boldsymbol{x'}} = \widetilde{\boldsymbol{F}}\widetilde{\boldsymbol{x}}$$

以上の 4 式を用いて，行列 $\boldsymbol{F}$ と $\widetilde{\boldsymbol{F}}$ の間の以下の関係式が得られる．

$$\widetilde{\boldsymbol{F}} = \boldsymbol{A}^{\mathsf{T}}\boldsymbol{F}\boldsymbol{A}$$

次に，非線形項について説明する．非線形項については，立体求積法（Cubature Method）を用いた積分計算の計算量削減手法 [4] を用いて次元削減を行う．非線形関数 $\mathcal{F}$ について，$\boldsymbol{f} = \mathcal{F}(\boldsymbol{x})$ とする．ここで，$\boldsymbol{f}$，$(\boldsymbol{x})$ は n 次元ベクトルである．削減前の $\boldsymbol{f}$ は，$\mathcal{F}$ をシミュレーション空間 Ω の全ての計算点 $\boldsymbol{x_p}$ において積分することで求められる．

$$\boldsymbol{f} = \int_{\Omega} \mathcal{F}_p(\boldsymbol{x_p})$$

この積分計算を立体求積法を用いて次元削減する．立体求積法はシミュレーション空間全域から適切に計算点を有限個サンプリングし，重み付き和をとることで積分計算を離散化する．サンプリングした点集合を P，サンプリング点 p に対応する重みを w_p とすると，立体求積法は以下のように表せる．

$$\boldsymbol{f} = \sum_{p=1}^{P} w_p \mathcal{F}_p(\boldsymbol{x_p})$$

この式は速度に関する基底を用いて以下のように部分空間へ射影することが可能である．

$$\boldsymbol{U_1 f} = \sum_{p=1}^{P} w_p (\boldsymbol{U^p})^{\mathsf{T}} \mathcal{F}_p(\boldsymbol{U^p x})$$

適切なサンプリング点の集合と重みの計算にかかる計算量は，文献 [4] の NNLS（Non Negative Linear Solve）ソルバによる手法を用いると $O(P \times rTP^2) = O(rTP^3)$ である．

3 スナップショット分割による高速化

3.1 フラクショナルステップ法への部分空間法の適用

フラクショナルステップ法では，中間子 $\boldsymbol{u_0}$ から $\boldsymbol{u_3}$ までの 4 種類の速度ベクトルを計算する．それぞれの速度ベクトルを部分空間に射影するため，対応したスナップショット行列が 1 つずつ必要である．中間子 $\boldsymbol{u_i}$ から得た基底を $\boldsymbol{U_i}$，$\boldsymbol{p}$ から得られた基底を $\boldsymbol{P}$ とする．フラクショナルステップ法の線形項の計算を部分空間に射影すると，以下のようになる．

$$\begin{aligned}
\boldsymbol{U_2^{\mathsf{T}} u_2} &= (\boldsymbol{U_2^{\mathsf{T}} V U_1}) \boldsymbol{U_1^{\mathsf{T}} u_1} & \widetilde{\boldsymbol{u_2}} &= \widetilde{\boldsymbol{V}}\widetilde{\boldsymbol{u_1}} \\
\boldsymbol{P^{\mathsf{T}} b} &= (\boldsymbol{P^{\mathsf{T}} W U_2}) \boldsymbol{U_2^{\mathsf{T}} u_2} & \widetilde{\boldsymbol{b}} &= \widetilde{\boldsymbol{W}}\widetilde{\boldsymbol{u_2}} \\
\boldsymbol{P^{\mathsf{T}} p} &= (\boldsymbol{P^{\mathsf{T}} A^{-1} P}) \boldsymbol{P^{\mathsf{T}} b} & \widetilde{\boldsymbol{p}} &= \widetilde{\boldsymbol{A}}\widetilde{\boldsymbol{b}} \\
\boldsymbol{U_3^{\mathsf{T}} u_3} &= \boldsymbol{U_2^{\mathsf{T}} u_2} - (\boldsymbol{U_3^{\mathsf{T}} Y P}) \boldsymbol{P^{\mathsf{T}} p} & \widetilde{\boldsymbol{u_3}} &= \widetilde{\boldsymbol{u_2}} - \widetilde{\boldsymbol{Y}}\widetilde{\boldsymbol{p}}
\end{aligned}$$

部分空間法によるシミュレーション計算は，前処理として基底と部分空間に射影した行列を計算する．基底を計算するステップの計算量のボトルネックは，QR 分解の $O(nm^2)$ である．部分空間に射影した行列を計算するステップについては，射影前のそれぞれの行列が非常に疎であるため，基底の計算ほど時間はかからない．

非線形項は立体求積法を用いて計算する．立体求積法に用いる基底は，外力項は $\boldsymbol{U}_0$ であり，移流項は $\boldsymbol{U}_1$ である．例えば，移流項を最終的な部分空間に射影した移流項計算は以下のようになる．

$$\widetilde{\boldsymbol{u_2}} = \sum_{p=1}^{P} w_p \widetilde{\boldsymbol{f}}_p = \sum_{p=1}^{P} w_p (\boldsymbol{U_1^p})^{\mathsf{T}} \mathcal{F}_p(\widetilde{\boldsymbol{u_1}})$$

3.2 部分空間法の高速化手法の提案

部分空間法の前処理において，基底の計算と行列の射影の計算量はスナップショットの 2 乗に比例し，前処理の大多数の時間を占めている．本節では部分空間法の前処理の高速化を達成するため，スナップショットを d 個に分割する手法を提案する．

部分空間法の基底の計算に用いるスナップショットを d 分割することによって，基底 1 つの計算負荷 $O(nm^2)$

が $\frac{1}{d^2}$ に削減される．アルゴリズム全体では d 回計算するため，計算負荷は $\frac{1}{d}$ になることが期待できる．行列の射影計算も分割した基底ごとに行う．空間計算量のボトルネックは分割前のスナップショット行列であり，分割前と変わらない．しかし，特異値分解する行列のサイズが $\frac{1}{d}$ になることで，GPU を用いた大規模高速計算と併用する場合，GPU メモリの削減が期待できる．空間計算量を GPU が一度に扱うことができる値まで削減することで，並列計算による高速化が期待できる一方で，特異値分解の回数が d 倍になることで，CPU-GPU 間のデータ通信の時間が増加することに注意が必要である．

4 計算機実験

本研究では，C++ と線形代数ライブラリ Eigen を用いて，文献 [3] の煙の流体シミュレーションを行うソフトウェアを実装した．立方体形状のシミュレーション空間の底面中心から，煙が立ち上る様子をシミュレーションし，そのデータを元に部分空間法による次元削減を行なった．また，OpenGL を用いてスライスベースのボリュームレンダリングを行い，実験結果を可視化するソフトウェアを実装した．これによって得られたシミュレーション結果と，提案手法の計算負荷と削減後のデータの精度について調査を行った．計算負荷の評価は，部分空間法の基底を計算する前処理と行列の射影について実行時間を計測した．データの精度は相対誤差を用いて評価を行なった．ここで相対誤差 L_2 は次元削減前後のベクトル $\boldsymbol{u}$ と $\widetilde{\boldsymbol{u}}$ について，以下のように表される．また，基底の精度を累積誤差を用いて評価した．

$$L_2 = \frac{||\boldsymbol{u} - \widetilde{\boldsymbol{u}}||_2}{||\boldsymbol{u}||_2}$$

表 1 は基底の計算にかかった実行時間，表 2 は行列の射影にかかった実行時間を解像度と分割数ごとに計測した結果である．基底の計算については分割数が大きくなるほど高速化に成功している．一方で，行列の射影に関しては，低解像度では分割数が大きいほど高速になっているが，高解像度における目立った高速化は見られなかった．

表 3 は，基底の累積寄与率を解像度と分割数ごとに計測した結果である．分割数が大きい基底の累積寄与率は，最大値最小値ともに大きくなっているのがわかる．

また，図 1 は，解像度 128^3 における，流速の L_2 誤差の時間推移を表したグラフである．スナップショットの分割の直後に L_2 誤差が減少している．また，流れの時間変化が少ないシミュレーションの後半では，分割なしと比べて精度が向上している．図 2 は，シミュレーション結果のレンダリング画像のうち，誤差が大きく変化した 99 フレーム目と 100 フレーム目のもので，上段が元のシミュレーション結果，中段が分割なし，下段が分割数 2 のものである．また，密度の誤差によって誤差が小さい部分は青く，誤差が大きい部分は赤く色をつけている．流速の誤差に基づいて，煙の密度分布に誤差が発生しているのがわかる．

非線形項については追加の実験を行ったところ，サンプリングされる点によって重みの計算が発散する場合がある．計算時間はスナップショット数については線形時間であるが，サンプリングする点の数が減ることにより NNLS ソルバの反復回数が減ることで，実質的に高速化が見込めることもあった．

表 1 基底計算の解像度と分割数ごとの実行時間 (秒)

解像度	分割なし	分割数 2	分割数 4	分割数 10
64^3	23.43	14.56	8.98	3.78
128^3	190.54	118.49	69.62	33.43

表 2 行列の射影の解像度と分割数ごとの実行時間 (秒)

解像度	分割なし	分割数 2	分割数 4	分割数 10
64^3	3.51	2.82	0.85	0.67
128^3	5.21	3.06	3.05	3.98

表 3 基底の累積寄与率の最小値と最大値

解像度	分割なし	分割数 2	分割数 4	分割数 10
64^3	0.969886	0.972114	0.979512	0.992557
		0.985917	0.990411	0.996893
128^3	0.940446	0.954735	0.971454	0.988967
		0.96489	0.976306	0.996132

5 考察

部分空間法において，基底による近似誤差が常に発生する．特に急激な流れの変動があった場合，元のデータを十分に近似するには多数のモードを用いる必要があり，用いる基底が固定されていると元の流れを再現できず，誤差が発生する．これに対して，基底を切り替えたとき精度が誤差が大きく減少する．これはスナップショットの範囲を限定した基底を用いることで，流れの特徴をよりよく再現することができたためと考えられる．一方で，流速の基底の切り替えに伴う精度向上によ

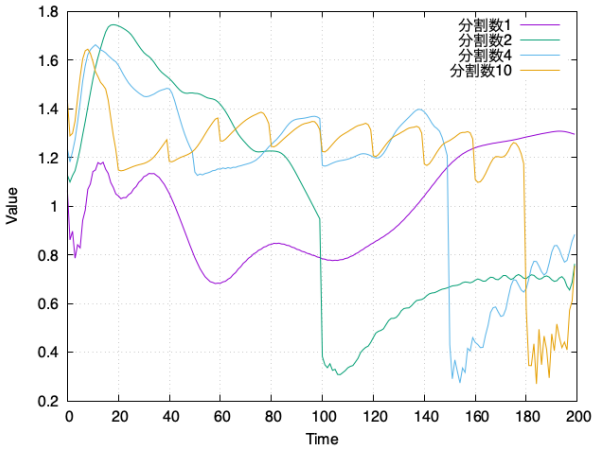
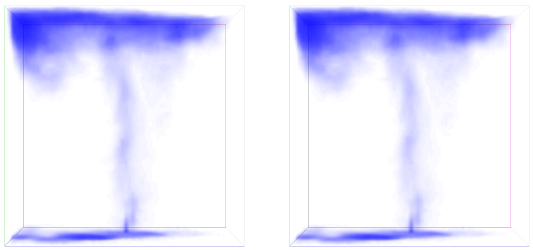
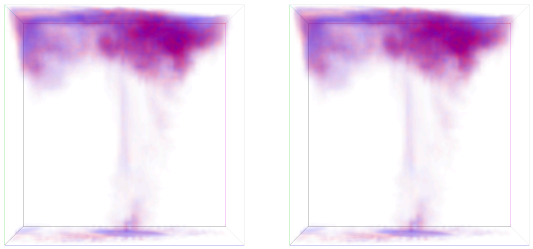


図 1　解像度 128^3 における L_2 誤差

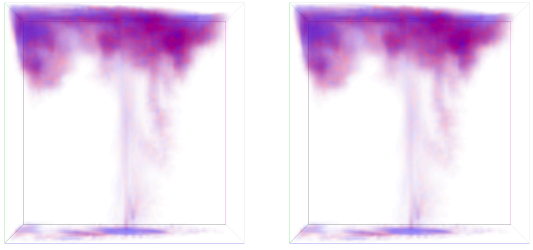
図 2　解像度 128^3 のレンダリング結果



(a) gruound truth 99 フレーム目　**(b)** gruound truth 100 フレーム目



(c) 分割数 1，99 フレーム目　**(d)** 分割数 1，100 フレーム目



(e) 分割数 2，99 フレーム目　**(f)** 分割数 2，100 フレーム目

るシミュレーションの精度は，密度分布を観察すると影響が小さい．これは，部分空間法による誤差は蓄積していくものであり，局所的に誤差が小さくなることによる影響が小さいためと考えられる．

分割後の累積寄与率は分割前よりも大きいのに対して，シミュレーションの相対誤差は分割後の方が大きい場合がある．これは，累積寄与率は直接シミュレーションの誤差を測る指標ではないためである．特に抽出するモードが細部の流れを再現できない場合に誤差が大きくなる可能性がある．

6　終わりに

流体シミュレーションにおける部分空間法の前処理にかかる計算時間に対し，スナップショットを分割することによって高速化する手法を提案した．特に基底計算については分割数に応じた高速化を達成し，前処理全体の高速化を達成した．スナップショット分割によるシミュレーションの結果に関して，L_2 誤差では差が認められたものの，見た目の上では問題ない範囲であった．

今後の研究として，GPU を用いた大規模高速計算の併用などを用いた高速化が挙げられる．部分空間法の高速化について，シミュレーション結果を離散コサイン変換 (DCT) を用いて圧縮する手法が研究されている．この手法はシミュレーション結果を離散コサイン変換を用いて圧縮し，必要に応じて展開することで，GPU を用いた大規模高速計算を改善する手法である．提案した手法を組み込むことで，より高速な基底計算を達成できるかについて検討する．

参考文献

[1] A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, 22(104): 745–762, 1968.

[2] A. Treuille, A. Lewis, and Z. Popovic. Model Reduction for Real-time Fluids. *ACM Transactions on Graphics*,25(3)：826–834, 2006.

[3] R. Fedkiew, J. stam, H. Jensen. Visual simulation of smoke. In *Proceedings of SIGGRAPH 01*, 15–22, 2001.

[4] T. Kim, J. Delaney. Subspace Fluid Re-Simulation. *ACM Transactions on Graphics*,32(4): 62：1–62：9, 2013.