中央大学大学院理工学研究科情報工学専攻

修士論文

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

須之内　俊樹
Toshiki SUNOUCHI
学籍番号　23N8100018B

指導教員　　森口　昌樹 准教授

2025年3月

概　要

本研究では部分空間法の前処理に対して，計算時間を高速化する手法を提案する．部分空間法を適用することで，シミュレーションは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 d 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．

キーワード: 流体シミュレーション，部分空間法，Snapshot 固有直交分解，Cubature

目 次

第1章　序論

流体シミュレーションとは，コンピュータを用いて流体の位置．速度．圧力などの物理量を計算する手法である．シミュレーション対象となる流体には，空気やガスなどの気体．水や粘性を有する液体．さらには砂粒のような固体粒子を含むものまで多岐にわたる．また，二次元および三次元の流体現象を対象とすることができる．二次元では水面に広がる波紋や異なる流体が作り出すマーブル模様のシミュレーションが可能であり，三次元では日常的な流体現象を広範に再現することができる．計算結果を視覚化することで，流体力学の理論だけでは把握が困難な空間的な流体の振る舞いを直感的に理解することが可能となる．

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．工学分野では，流体に接する製品の設計・開発において，シミュレーションによって流体と製品の相互作用を事前に評価し，実験コストの削減などに寄与している．一方，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．特に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．流体シミュレーションの計算はGPUによる並列処理と相性が良く，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．

また，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図ることを目的とする．

第2章　関連研究

2.1　流体シミュレーション

まず，流体シミュレーションの分野で一般的に使われている表記の説明をする．位置 $\boldsymbol{x}$ や時刻 t を指定した物理量を表現する際は，$\boldsymbol{u}\left(\boldsymbol{x}, t\right)$ のように表記する．

- Δx,Δt: 離散化する計算格子の格子幅，次の時刻までの時間幅.
- $\boldsymbol{u}$：シミュレーション空間全体の流体の速度ベクトル．$\boldsymbol{u}\left(\boldsymbol{x}, t\right)$ は三次元ベクトルとなる．
- $\boldsymbol{p}$：流体の圧力.
- ρ,ν: それぞれ，流体の密度，流体の粘性．ここでは位置や時刻によらない定数とする．
- $\boldsymbol{f}$: 位置 $\boldsymbol{x}$，時刻 t での流体にかかる外力ベクトル．重力などはここに含める．シミュレーションの手法によっては外力を位置 $\boldsymbol{x}$，時刻 t によって変化する外力を扱うことができるが，今回は簡単のため，重力のみを扱い，どの位置でも，どの時刻でも一定の値として考える．
- ∇: 空間微分演算子ナブラ．スカラー場に作用させると勾配を表し，ベクトル場に作用させると発散を表す．３次元空間なので，$\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z})$
- $\frac{\mathrm{D}}{\mathrm{Dt}} = \frac{\partial}{\partial t} + \boldsymbol{u}\left(\boldsymbol{x}, t\right) \cdot \nabla$ ：ラグランジュ微分，または物質微分．連続体力学において主に用いられており，流れに沿って移動する観測者から見た，物理量の時間変化率を表している．

下記の式 2.1 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．非線形二階微分方程式となっており，代数的に一般解を求める事ができない．下記の式 2.2 は流体の連続の式と呼ばれる，質量保存や流量保存を表す式である．水や砂粒のように圧縮されない流体を扱うときは，流体の密度が常に一定，つまり密度の時間微分が 0 になり，それに伴い式 2.2 の左辺の第二項も 0 になる．これにより，式 2.3 と，式 2.4 を得る．式 2.1 と式 2.4 または式 2.3 を連立することで非圧縮性流体の速度を求め

ることができる．ナビエ・ストークス方程式を扱う際は，コンピューターで近似解を解析的に求める手法が用いられる．

$$\frac{\partial}{\partial t}\boldsymbol{u}\left(t\right)=-\left(\boldsymbol{u}\left(t\right)\cdot\nabla\right)\boldsymbol{u}\left(t\right)-\frac{1}{\rho}\nabla\boldsymbol{p}\left(t\right)+\nu\nabla^{2}\boldsymbol{u}\left(t\right)+\boldsymbol{f} \tag{2.1}$$

$$\frac{\mathrm{D}}{\mathrm{Dt}}\rho+\nabla\cdot\boldsymbol{u}\left(t\right)=0 \tag{2.2}$$

$$\frac{\mathrm{D}}{\mathrm{Dt}}\rho=0 \tag{2.3}$$

$$\nabla\cdot\boldsymbol{u}\left(t\right)=0 \tag{2.4}$$

一般の非線形微分方程式は厳密な解析的解法が存在せず，様々な条件を設定した上で，数値解法により離散化し，近似解を求めるのが一般的である．ナビエ・ストークス方程式の数値解法は，分離解法と連成解法が存在する．分離解法は微分方程式の各項ごとに計算を分割する手法であり，線形問題の収束性が良い利点がある．しかし，分割した項同士の相互作用を計算できないため，厳密な現象の再現が困難であるという欠点がある．一方，連成解法は各項を分離せずに解く手法であり，より正確に現象を再現できる利点がある．しかし，計算負荷が高く，収束性が悪いという欠点がある．分離解放の中でも圧力項の計算を分離する手法 [2, 3] によって，収束性や精度が問題であった圧力項計算の安定性が劇的に向上した．特に Marker And Cell 法 （MAC 法） [2] は現在では圧力を分離する解法のことを広く指す．また， [2] において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が提案されている．フラクショナルステップ法の分割において，式 2.1 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．分離解法は，コンピュータグラフィックスや流体解析の分野で広く用いられており，選択する手法によって詳細に計算したい相互作用を調整することが可能である．

以降はフラクショナルステップ法によるナビエ・ストークス方程式の分割方法を説明する．まず，上記の式 2.1 の時間微分に前進差分を適用し，以下を得る．

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

この式を，中間子 $\boldsymbol{u_0}$ から $\boldsymbol{u_3}$ を用いて，以下のように各項ごとに分割して計算する．

$$\boldsymbol{u_0}=\boldsymbol{u}\left(t\right)-\Delta t\boldsymbol{f} \tag{2.5}$$

$$\boldsymbol{u_1}(\boldsymbol{x})=\boldsymbol{u_0}(\boldsymbol{x})-\Delta t\left(\boldsymbol{u_0}(\boldsymbol{x})\cdot\nabla\right)\boldsymbol{u_0}(\boldsymbol{x}) \tag{2.6}$$

$$\boldsymbol{u_2}=\boldsymbol{u_1}-\Delta t\nu\nabla^{2}\boldsymbol{u_1} \tag{2.7}$$

$$\boldsymbol{u}\left(t+\Delta t\right)=\boldsymbol{u_3}=\boldsymbol{u_2}-\Delta t\frac{1}{\rho}\nabla\boldsymbol{p} \tag{2.8}$$

ここで，式 2.5 は外力項，式 2.6 は移流項，式 2.7 は粘性項，式 2.8 は圧力項を計算する式である．

ナビエ・ストークス方程式の空間の離散化については，大きく分けて格子法（Eulerian）と粒子法（Lagrangian）がある．格子法とは，流体を扱う空間を正方形や立方体の格子に区切り，格子の中心や辺，頂点などに物理量を配置して計算する方法である．区切る格子の数が多いほど，流体の詳細な動きが計算できるが，計算負荷は増加する．格子法の利点として，規則的に配置された格子により空間を離散化することで，微分演算を差分法などで近似できる点や，境界条件の設定が容易である点が挙げられる．格子の大きさよりも細かい流体の振る舞いを正確に表現できないことや，格子内での流体の運動が平均化されるため，個々の流体粒子の詳細な運動を追跡できないという欠点がある．これにより，薄く広がる流体の挙動や水飛沫の表現が困難になる．

粒子法は，流体の個々の粒子の運動を追跡する手法であり，薄く広がる流体や水飛沫の表現が容易である利点を持つ．一方で，計算精度の保証が難しいことや，境界条件の設定が煩雑になるといった欠点がある．

格子法および粒子法は，固体にも適用することが可能であり，流体とは異なる支配方程式を用いて変形や移動を計算することができる．流体と固体の相互作用をシミュレーションする際には，格子法と粒子法をそれぞれ個別に適用することが可能である [1]．また，流体の移流項のみを粒子法で計算し，その他の項を格子法で計算するなど，各項ごとに異なる手法を適用する手法 [4, 5] も存在する．

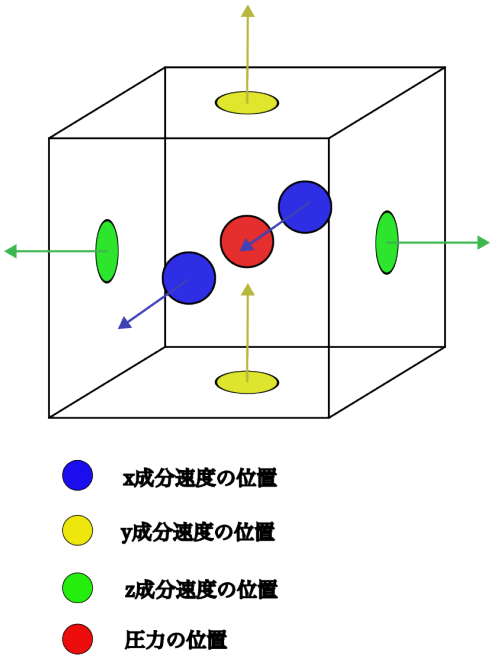


図 2.1: スタッガード格子

格子法による空間の離散化は，扱う格子によって様々な手法が存在する．例えば物理量を格子の中心に配置するコロケート格子や，速度成分を格子の面の中心に配置するスタッガード格子が存在する．コロケート格子は流体の格子の形状によらず適用できるが，境界条件の設定が煩雑である．スタッガード格子は適用が立方体格子などに限定される一方，境界条件の設定が容易である．スタッガード格子は上の図 2.1 のように，圧力の定義位置を格子の中心とし，格子面の中心にその格子面と垂直な流速の成分の位置を定義する．例えば xy 平面に並行な格子面には，その地点での流速の z 成分を配置する．

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．部分空間上におけるシミュレーションでは，元のシミュレーションよりも Δt を小さい値に設定することで，シミュレーションの時間変化をより詳細に追跡することが可能である．以下では，部分空間法の理論の概要を説明する．

2.2.1 部分空間への射影

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 $\tilde{\boldsymbol{x}}$ に変換する，$n \times r$ 直交行列 $\mathbf{A}$ を考える．直交行列の性質から，以下の 2 式が成り立つ．

$$\tilde{\boldsymbol{x}} = \mathbf{A}^{\mathbf{T}}\boldsymbol{x}$$

$$\boldsymbol{x} = \mathbf{A}\tilde{\boldsymbol{x}}$$

また，$\tilde{\boldsymbol{x'}} = \mathbf{A}^{\mathbf{T}}\boldsymbol{x'}$ を満たす n 次元ベクトル $\boldsymbol{x'}$，r 次元ベクトル $\tilde{\boldsymbol{x'}}$ 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 $\mathbf{F}$，任意の $m \times r$ 行列 $\tilde{\mathbf{F}}$ を考える．

$$\boldsymbol{x'} = \mathbf{F}\boldsymbol{x}$$

$$\tilde{\boldsymbol{x'}} = \tilde{\mathbf{F}}\tilde{\boldsymbol{x}}$$

$\tilde{\boldsymbol{x'}} = \mathbf{A}^{\mathbf{T}}\boldsymbol{x'}$ に，$\boldsymbol{x'} = \mathbf{F}\boldsymbol{x}$ と，$\boldsymbol{x} = \mathbf{A}\tilde{\boldsymbol{x}}$ を代入することで，

$$\tilde{\boldsymbol{x'}} = (\mathbf{A}^{\mathbf{T}}\mathbf{F}\mathbf{A})\tilde{\boldsymbol{x}}$$

が得られる．$\tilde{\boldsymbol{x'}} = \tilde{\mathbf{F}}\tilde{\boldsymbol{x}}$ と比較すると，

$$\tilde{\mathbf{F}} = \mathbf{A}^{\mathbf{T}}\mathbf{F}\mathbf{A}$$

が得られる．

以上により，任意のベクトルに対する行列積を，$n \times r$ 直交行列 $\mathbf{A}$ を用いて部分空間に射影することができる．

2.2.2 Snapshot 固有直交分解

流体シミュレーションにおいて，部分空間に射影したベクトルを用いてシミュレーション計算をするためには，射影後のベクトルも流体の非圧縮性条件等の性質を満たさなければならない．そのような射影を達成する行列 $\mathbf{A}$ を得る手法として，Snapshot 固有直交分解がある．行列 $\mathbf{A}$ を既存の流体のデータを主成分分析を用いて計算することで，流体の性質を満たした射影が可能である．以降は Snapshot 固有直交分解の具体的な計算方法について説明する．

時刻 $t\{t \in N, 0 \leq t \leq m\}$ の n 次元ベクトルデータ $\boldsymbol{u_t}$ を用いて，$n \times m$ 行列 $\mathbf{S}$ を定義する．

$$\mathbf{S} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_0 & \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_{m-1} \\ | & | & & | \end{bmatrix}$$

次に，行列 $\mathbf{S}$ 主成分分析を行うため，$n \times r$ 行列 $\mathbf{A}$ について以下の最小化問題を考える．ここで r は抽出したい主成分の数である．

$$min||\mathbf{S} - \mathbf{A}\mathbf{A}^{\mathbf{T}}\mathbf{S}||_F^2$$

この最小化問題の解となる行列 $\mathbf{A}$ は，$\mathbf{S}\mathbf{S}^{\mathbf{T}}$ の固有ベクトル $\boldsymbol{v_i}(0 \leq i \leq r-1)$ を用いて，以下の行列になる．

$$\mathbf{A} = \begin{bmatrix} | & | & & | \\ \boldsymbol{v}_0 & \boldsymbol{v}_1 & \cdots & \boldsymbol{v}_{r-1} \\ | & | & & | \end{bmatrix}$$

ここで，固有ベクトル $\boldsymbol{v_i}$ に対応する固有値を λ_i は，$\lambda_0 \geq \lambda_1 \cdots \geq \lambda_i \geq \lambda_{n-1}$ を満たすとする．$\mathbf{S}\mathbf{S}^{\mathbf{T}}$ は対称行列になっているため，固有ベクトルは互いに直交し，$\mathbf{A}^{-\mathbf{1}} = \mathbf{A}^{\mathbf{T}}$ となる．

次に，行列 $\mathbf{A}$ を用いて $\boldsymbol{u}$ を $\tilde{\boldsymbol{u}}$ に射影するとき，$\tilde{\boldsymbol{u}}$ が流体の非圧縮性条件 $\nabla \cdot \tilde{\boldsymbol{u}} = 0$ を満たしていることを確かめる．行列 $\mathbf{S}$ の各列ベクトル $\boldsymbol{u_i}$ は既存のシミュレーションの速度データであるため，流体の非圧縮性条件 $\nabla \cdot \boldsymbol{u_i} = 0$ を満たす．このことから，

$$\nabla \cdot \mathbf{S} \ = \sum_{0 \leq i \leq n-1} \nabla \cdot \boldsymbol{u_i} = 0$$

が成り立つ．

また，行列 $\mathbf{S}\mathbf{S}^{\mathbf{T}}$ の固有ベクトル $\boldsymbol{v_i}$ と，それに対応する固有値を λ_i とすると，固有値と固有ベクトルの定義から以下が成り立つ．

$$\mathbf{S}\mathbf{S}^{\mathbf{T}}\boldsymbol{v_i} = \lambda_i \boldsymbol{v_i}$$

両辺に左から $\nabla\cdot$ を取ることにより，

$$(\nabla \cdot \mathbf{S})\, \mathbf{S}^{\mathbf{T}}\boldsymbol{v_i} = \lambda_i \nabla \cdot \boldsymbol{v_i}$$

$\nabla \cdot \mathbf{S} \ = 0$ より，0 でない λ_i について，$\nabla \cdot \boldsymbol{v_i} = 0$ が成り立つことがわかる．$\tilde{\boldsymbol{u}} = \mathbf{A}^{\mathbf{T}}\boldsymbol{u}$ の第 i 成分は $\boldsymbol{v_i} \cdot \boldsymbol{u}$ であり，$\nabla \cdot \tilde{\boldsymbol{u}} = 0$ が成り立つことがわかる．

$n \times n$ 行列 $\mathbf{S}\mathbf{S}^{\mathbf{T}}$ の主成分分析を行う代わりに，$\mathbf{S}^{\mathbf{T}}\mathbf{S}$ の主成分分析を行い，$\mathbf{S}\mathbf{S}^{\mathbf{T}}$ の固有値固有ベクトルを求めることができる．$\mathbf{S}\mathbf{S}^{\mathbf{T}}\boldsymbol{v_i} = \lambda_i \boldsymbol{v_i}$ の両辺に左から $\mathbf{S}^{\mathbf{T}}$ をかけることにより，

$$(\mathbf{S}^{\mathbf{T}}\mathbf{S})\, \mathbf{S}^{\mathbf{T}}\boldsymbol{v_i} = \lambda_i \mathbf{S}^{\mathbf{T}}\boldsymbol{v_i}$$

を得る．行列 $\mathbf{S}^{\mathbf{T}}\mathbf{S}$ の固有値は，$\mathbf{S}\mathbf{S}^{\mathbf{T}}$ の固有値と一致し，固有値 λ_i 対応する固有ベクトル $\boldsymbol{u_i}$ について，$\boldsymbol{u_i} = \mathbf{S}^{\mathbf{T}}\boldsymbol{v_i}$ であることがわかる．よって，$\mathbf{S}^{\mathbf{T}}\mathbf{S}$ の主成分分析により得られた固有ベクトルと $\mathbf{S}^{\mathbf{T}}$ の逆行列を用いて，$\mathbf{S}\mathbf{S}^{\mathbf{T}}$ の主成分分析を行うことができる．

2.2.3 特異値分解を用いる方法

$n \times r$ 行列 $\mathbf{S}$ の特異値分解は，以下のように定義される．

$$\mathbf{S} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

ここで，$\mathbf{U}$ は $n \times n$ のユニタリ行列であり，左特異ベクトルを列ベクトルとして持つ．$\mathbf{V}$ は $r \times r$ のユニタリ行列であり，右特異ベクトルを列ベクトルとして持つ．$\mathbf{\Sigma}$ は $n \times r$ の対角行列であり，対角成分には特異値 $\sigma_1, \sigma_2, \ldots, \sigma_r$ が非負の降順で並ぶ．

以下に示す，行列 $\mathbf{S^T S}$ の固有値や固有ベクトルと行列 $\mathbf{S}$ の特異値分解の関係により，$\mathbf{SS^T}$ の主成分分析を行う代わりに，行列 $\mathbf{S}$ の特異値分解を用いることができる．行列 $\mathbf{S^T S}$ の対角化を，以下のように表す．

$$\mathbf{S^T S} = \mathbf{P A P^{-1}} \tag{2.9}$$

ここで，$\mathbf{A}$ は $n \times n$ の対角行列であり，対角成分には行列 $\mathbf{S^T S}$ の固有値が降順に並ぶ．$\mathbf{P}$ の i 列は固有値 $\mathbf{A_{i,\,i}}$ に対応する固有ベクトルである．行列 $\mathbf{S}$ の特異値分解より，以下が成り立つ．

$$\mathbf{S^T S} = (\mathbf{V}\mathbf{\Sigma}\mathbf{U}^T)(\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)$$

行列 $\mathbf{U}$ はユニタリ行列，$\mathbf{\Sigma}$ は対角行列であるから，

$$\mathbf{S^T S} = \mathbf{V}\mathbf{\Sigma^2}\mathbf{V}^T$$

式 2.9 と右辺を比較することによって，$\mathbf{P} = \mathbf{V}$，$\mathbf{A} = \mathbf{\Sigma^2}$ であることがわかる．

2.2.4 非線形項の計算

非線形項については，Cubature 法を用いた積分計算の計算量削減手法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 Cubature 法を用いて次元削減する．Cubature 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，Cubature 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．

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

ここで，$\mathbf{U^p}$ は $3 \times n$ 基底行列 $\mathbf{U}$ のサンプリング点 p に関する行を抜粋して作成した，$3 \times r$ 行列である．ここで，$\tilde{\boldsymbol{f}}_p = (\mathbf{U^p})^T \mathcal{F}_p(\tilde{\boldsymbol{u}})$ とする．

以降はサンプリング点と重みを評価する手法について説明する．サンプリングした点集合を $\boldsymbol{P}$ とし，点集合の要素数を P とする．また，Snapshot 数を T とする．以下の式を解の値が全て非負になるような線形ソルバである NNLS（Non Negative Linear Solve）ソルバを用いることで，非負の $\boldsymbol{w}$ が得られる．

$$\begin{bmatrix} \tilde{\boldsymbol{f}}_0^0 & \cdots & \tilde{\boldsymbol{f}}_0^p & \cdots & \tilde{\boldsymbol{f}}_0^P \\ \vdots & \ddots & \vdots & & \vdots \\ \tilde{\boldsymbol{f}}_t^0 & \cdots & \tilde{\boldsymbol{f}}_t^p & \cdots & \tilde{\boldsymbol{f}}_t^P \\ \vdots & & \vdots & \ddots & \vdots \\ \tilde{\boldsymbol{f}}_T^0 & \cdots & \tilde{\boldsymbol{f}}_T^p & \cdots & \tilde{\boldsymbol{f}}_T^P \end{bmatrix} \begin{bmatrix} w_0 \\ \vdots \\ w_p \\ \vdots \\ w_P \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{f}}_0 \\ \vdots \\ \tilde{\boldsymbol{f}}_t \\ \vdots \\ \tilde{\boldsymbol{f}}_T \end{bmatrix} \tag{2.10}$$

ここで，$\tilde{\boldsymbol{f}}_t$ をシミュレーションの Snapshot 行列 $\mathbf{S}$ から抜粋した t 番目の速度 $\boldsymbol{u}_t$ を用いて $\tilde{\boldsymbol{f}}_t = \mathbf{U}\boldsymbol{u}_t$ とする．$\tilde{\boldsymbol{f}}_t^p$ はサンプリング点に関する値 $\mathbf{U^p}\boldsymbol{u}_t^{\boldsymbol{p}}$ である．それぞれ次元削減後の速度ベクトルを表す．式 2.10 を線形方程式 $\mathbf{A}\boldsymbol{w} = \boldsymbol{b}$ とすると，$\mathbf{A}$ は $rT \times P$ 行列である．空集合の状態からランダムに点集合を選び，残差のノルム $||\boldsymbol{r}||_2 = ||\boldsymbol{b} - \mathbf{A}\boldsymbol{w}||_2$ が閾値以下になるまでサンプリング点を追加する．サンプリングされた点を点集合に追加する際に，以下の式によって求められる確率関数を用いて追加するモンテカルロ法により，更なる効率化を達成できる．

$$f\left(\boldsymbol{x_p}\right) = R\left(\frac{\boldsymbol{a_p} \cdot \boldsymbol{r}}{\boldsymbol{r} \cdot \boldsymbol{r}}\right)$$

上記の行列はアルゴリズムの概略を以下に示す．

Algorithm 1 Cubature random point sampling

1: $P = \phi$, $\boldsymbol{r} = \boldsymbol{b}$ と初期化する.
2: **while** $||\boldsymbol{r}||_2 \leq \epsilon$ **do**
3: ランダムに 1 つ x_p をサンプリングし，確率 $p\left(\boldsymbol{x_p}\right)$ で x_p を P に追加する.
4: 2.10 に従って，$\mathbf{A}\boldsymbol{w} = \boldsymbol{b}$ を解く.
5: 残差を次のように更新する．$\boldsymbol{r} = \boldsymbol{b} - \mathbf{A}\boldsymbol{w}$
6: 重みが 0 となった点を P から削除する.
7: **end while**

サンプリング点の集合と重みの計算にかかる計算量は，式 2.10 を P 回解くため，$O\left(P \times rTP^2\right) = O\left(rTP^3\right)$ である.

第3章　煙の流体シミュレーション

流体シミュレーションを実施する際には，境界条件や外力項を適切に設定することで流体の挙動を制御することが可能である．特に，煙のような流体のシミュレーションにおいて，適切な外力を与えることで現実的な動きを再現できる．Fedkiw らによる手法 [6] は，高速かつ安定したシミュレーションを実現し，映像分野への応用が実用的になった最初の手法である．本章では，文献 [6] を参考に，フラクショナルステップ法を格子法により離散化した煙の流体シミュレーションの実装方法について述べる．さらに，スライスベースのボリュームレンダリングを用いた可視化手法についても説明する．

3.1　格子法によるフラクショナルステップ法

シミュレーション空間の x, y, z 軸方向の分割数を nx, ny, nz とする．このとき空間解像度は nx · ny · nz となり，速度の x, y, z 成分の空間解像度はそれぞれ $(nx+1)\cdot ny\cdot nz$, $nx\cdot(ny+1)\cdot nz$，$nx\cdot ny\cdot(nz+1)$ となる．本論文では，$nx=ny=nz$ とし，速度 $\boldsymbol{v}$ と圧力 $\boldsymbol{p}$ は以下のベクトルに離散化する．

- $\boldsymbol{v_x}$：$(nx+1)\times ny\times nz$ 次元ベクトル．
- $\boldsymbol{v_y}$：$nx\times(ny+1)\times nz$ 次元ベクトル．
- $\boldsymbol{v_z}$：$nx\times ny\times(nz+1)$ 次元ベクトル．
- $\boldsymbol{v}$：$3\times(nx+1)\cdot ny\cdot nz$ 次元ベクトル．
- $\boldsymbol{p}$：$nx\cdot ny\cdot nz$ 次元ベクトル．

ここで $\boldsymbol{v}$ の 0 番目 から $(nx+1)\cdot ny\cdot nz-1$ 番目までの成分には速度の x 成分，$(nx+1)\cdot ny\cdot nz$ から $2\times(nx+1)\cdot ny\cdot nz-1$ 番目の成分には速度の y 成分，$2\times(nx+1)\dot{n}y\cdot nz-1$ から $3\times(nx+1)\cdot ny\cdot nz-1$ 番目の成分 には速度の z 成分を保存する．また，

位置 $\boldsymbol{x} = (i, j, k)$ における速度 $\boldsymbol{v}(\boldsymbol{x})$ を，以下のように表す．

$$\boldsymbol{v}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{v_x}(i, j, k) \\ \boldsymbol{v_y}(i, j, k) \\ \boldsymbol{v_z}(i, j, k) \end{bmatrix}$$

3.1.1 ディリクレ境界条件

流体シミュレーションにおいて，流体と流体以外の境界面における物理量の条件を適切に設定する必要があり，これを境界条件と呼ぶ．文献 [6] では，境界における物理量を事前に一定の値に設定するディリクレ境界条件を採用しており，境界における物理量は 0 と設定される．ディリクレ境界条件は速度成分を逐次初期化することで適用可能であり，また，境界に対応する成分を 0 とした単位行列を用いて表現することもできる．この行列表現は通常の流体シミュレーションでは不要であるが，部分空間法を用いた流体計算において，境界条件を部分空間に射影するため，境界条件を行列積によって表現する手法が用いられている．ディリクレ境界条件を表現する行列を $\mathbf{D}$ とすると，$\mathbf{D}$ の i, j 成分は以下のように定義される．

$$\mathbf{D}_{i,\,j} = \begin{cases} 1 & i = j \text{ かつ } i \text{ は境界成分でない} \\ 0 & \text{その他} \end{cases} \tag{3.1}$$

3.1.2 外力項計算

外力項においては．再現したい現象に適した外力を与える必要がある．ユーザーが設定可能なパラメータによって外力を調整できる計算手法が望ましい．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．さらに，より現実的な挙動を再現するために，他の計算では定数として扱われる密度や温度を，本手法では位置および時刻に依存するスカラー量として計算する．

まず，気体に働く重力と，対流を発生させる力について説明する．位置 $\boldsymbol{x}$ における気体の密度，温度をそれぞれ $\rho(\boldsymbol{x})$，$T(\boldsymbol{x})$，環境温度を $T_{amb}(\boldsymbol{x})$，重力の方向ベクトルを $\boldsymbol{d}$ とすると，気体に働く重力と，対流を発生させる力の合力 $\boldsymbol{f}_{\text{buoy}}$ は以下のように定義される．

$$\boldsymbol{f}_{\text{buoy}}(\boldsymbol{x}) = \alpha\rho(\boldsymbol{x})\,\boldsymbol{d} + \beta(T(\boldsymbol{x}) - T_{amb}(\boldsymbol{x}))\,\boldsymbol{d} \tag{3.2}$$

ここで，α，β はユーザーが設定するパラメータであり，それぞれ重力と対流の力の大きさを調整する．

次に，気体の渦運動を生成する力を説明する．気体の渦運動を強めるために，渦度 $\boldsymbol{\omega}$ を利用した外力を考える．渦度は速度場 $\mathbf{u}$ の回転として以下のように定義される

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}$$

渦度の大きい領域では流体が強く回転しており，この回転運動を強調するために以下のような外力項 $\boldsymbol{f}_{\text{vortex}}$ を定義する

$$\boldsymbol{f}_{\text{vortex}} = \lambda(\boldsymbol{\omega} \times \mathbf{u})$$

ここで，λ は渦運動を強調する強度を制御するパラメータである．この外力項は，渦度ベクトルと速度ベクトルの外積を利用することで，流れの回転を増幅する効果を持つ．

スタッガード格子における外力の配置位置は格子の中心であるため，流速の位置によって外力を補間して計算する．最終的な外力項計算は以下のようになる．

$$\boldsymbol{u_0} = \boldsymbol{u} - \Delta t \left(\boldsymbol{f}_{\text{buoy}} + \boldsymbol{f}_{\text{vortex}} \right) \tag{3.3}$$

3.1.3 移流項計算

移流項は流体の物理量が流体の速度場によって移動する様を計算することができる．移流項は非線形方程式であり，様々な計算方法が存在する．本節ではコンピュータグラフィックスの分野において広く用いられている，文献 [7] の Semi-Lagrangian 法を紹介する．Semi-Lagrangian 法は計算負荷が小さく，陰解法により安定した計算結果を得ることができる．

Semi-Lagrangian 法は，格子に配置されている計算点上の物理量を計算する格子法（Eulerian）の特徴と，移動する計算点を追跡し計算を行う粒子法（Lagrangian）の特徴を両方持っている計算手法である．移流項の計算

$$\boldsymbol{u_1} = \boldsymbol{u_0} - \Delta t \left(\boldsymbol{u_0} \cdot \nabla \right) \boldsymbol{u_0}$$

を変形し，時刻の偏微分に対し前進差分 $\frac{\partial \boldsymbol{u}}{\partial t} = \frac{\boldsymbol{u_1} - \boldsymbol{u_0}}{\Delta t}$ すると，以下の式が得られる．

$$\frac{\partial \boldsymbol{u}}{\partial t} + \left(\boldsymbol{u_0} \cdot \nabla \right) \boldsymbol{u_0} = 0$$

この式は以下の輸送方程式と呼ばれる式の特殊な場合であり，輸送方程式は一般の物理量 ϕ に関して適用可能である．

$$\frac{\partial \phi}{\partial t} + (\boldsymbol{u} \cdot \nabla)\phi = 0 \tag{3.4}$$

この輸送方程式が成り立つとき，位置 $\boldsymbol{x}$，時刻 t における物理量 $\phi(\boldsymbol{x}, t)$ について，以下の式が成り立つ．

$$\phi(\boldsymbol{x}, t) = \phi(\boldsymbol{x} - \Delta t \boldsymbol{u}, t) \tag{3.5}$$

Semi-Lagrangian 法を以下のように流速に適用することで移流項の計算を行う．また，密度や温度などに対して適用することで，空間の速度ベクトル場に沿って気体が運動する様子を再現することができる．

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

以下では，輸送方程式 3.4 が成り立つとき，式 3.5 が成り立つことを確認する．$\boldsymbol{x}$ にある物理量は $\boldsymbol{u}(\boldsymbol{x})$ に沿って $\Delta t \boldsymbol{u}(\boldsymbol{x})$ 移動すると仮定すると，移動前の物理量は $\phi(\boldsymbol{x} - \Delta t \boldsymbol{u}(\boldsymbol{x}))$ となる．この間の物理量の変化 $\Delta\phi$ は以下のようになる．

$$\Delta\phi = \phi(\boldsymbol{u}(\boldsymbol{x}), t) - \phi(\boldsymbol{x} - \Delta t \boldsymbol{u}(\boldsymbol{x}), t)$$

これを一次のテイラー展開を用いて表すと,

$$\Delta\phi = \frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial \boldsymbol{x}} \phi(\boldsymbol{x}, t) \Delta t + \frac{\partial \phi(\boldsymbol{x}, t)}{\partial t} \Delta t$$

$\frac{\partial \boldsymbol{u}(\boldsymbol{x})}{\partial \boldsymbol{x}} \phi(\boldsymbol{x}, t) = (\boldsymbol{u}(\boldsymbol{x}) \cdot \nabla)\phi(\boldsymbol{x}, t)$ より,

$$\frac{\Delta\phi}{\Delta t} = (\boldsymbol{u}(\boldsymbol{x}) \cdot \nabla)\phi(\boldsymbol{x}, t) + \frac{\partial \phi(\boldsymbol{x}, t)}{\partial t}$$

右辺が輸送方程式 3.4 になっていることから，$\Delta\phi = 0$ が成り立つ．$\Delta\phi = \phi(\boldsymbol{u}(\boldsymbol{x})) - \phi(\boldsymbol{x} - \Delta t \boldsymbol{u}(\boldsymbol{x})) = 0$ より，式 3.5 が成り立つことが確認できる．

3.1.4 粘性項計算

粘性項は以下の式を用いて，流体が徐々に周囲に広がっていく様を計算することができる．

$$\boldsymbol{u_2} = \boldsymbol{u_1} - \Delta t \nu \nabla^2 \boldsymbol{u_1}$$

∇^2 を離散ラプラシアン行列 $\mathbf{L}$ を用いて離散化することができるため，粘性項計算は以下のように表すことができる．

$$\boldsymbol{u_2} = \boldsymbol{u_1} - \Delta t \nu \nabla^2 \boldsymbol{u_1} = (\mathbf{I} - \frac{\Delta t \nu}{(\Delta x)^2} \mathbf{L})\, \boldsymbol{u_1} \tag{3.6}$$

ここで，v_0 から v_{n-1} の n 個の頂点を持つ無向グラフに対して $n \times n$ 行列 $\mathbf{L}$ の成分 $\mathbf{L}_{i,j}$ は以下のように定義される．ただし，無向グラフは自己ループを持たないとする．

$$\mathbf{L}_{i,\,j} = \begin{cases} deg\,(v_i) & i = j \\ -1 & v_i \text{と } v_j \text{が接続している} \\ 0 & \text{その他} \end{cases}$$

スタッガード格子を用いた流体シミュレーションにおいて，無向グラフは流体のそれぞれの速度成分ごとの連結成分を持つとする．本論文のシミュレーションにおいては連結成分数は 3 であり，それぞれの頂点数は $(nx+1) \times ny \times nz$ である．また，シミュレーション境界を除いて，速度に対応する次数は 6 である．シミュレーション境界に対応する離散ラプラシアン行列の成分については，ディリクレ境界条件によって速度成分が 0 になるため，任意の値でよい．

粘性項計算を $\boldsymbol{u_2} = \mathbf{V}\boldsymbol{u_1}$ と表現する行列を $\mathbf{V}$ とすると，の成分 $\mathbf{V}_{i,\,j}$ は以下のように定義される．

$$\mathbf{V}_{i,\,j} = \begin{cases} 1 - 6\frac{\Delta t\nu}{(\Delta x)^2} & i = j \\ \frac{\Delta t\nu}{(\Delta x)^2} & v_i \text{と } v_j \text{が接続している} \\ 0 & \text{その他} \end{cases}$$

3.1.5 圧力項計算

圧力項計算は，流体の非圧縮性を満たすように流体に圧力勾配による外力を与える計算である．本節では，代表的な圧力項計算手法であるコリンの射影法 [3] を紹介する．コリンの射影法は以下の式を満たす圧力 $\boldsymbol{p}$ を求める．

$$\boldsymbol{u}\,(t + \Delta t) = \boldsymbol{u_2} - \Delta t\frac{1}{\rho}\nabla p$$

上記の式の両辺の発散をとると，

$$\nabla \cdot \boldsymbol{u}\,(t + \Delta t) = \nabla \cdot \boldsymbol{u_2} - \Delta t\frac{1}{\rho}\nabla^2 p$$

が得られる．流体の非圧縮性条件，$\nabla \cdot \boldsymbol{u}\,(t + \Delta t) = 0$ により，左辺は 0 になるため，

$$\nabla \cdot \boldsymbol{u_2} = \frac{\Delta t}{\rho}\nabla^2 p \tag{3.7}$$

が得られる．この式を圧力に関するポアソン方程式として解き，$\boldsymbol{u}\,(t+\Delta t) = \boldsymbol{u_2} - \Delta t\frac{1}{\rho}\nabla p$ に代入することで，圧力勾配による速度変化を計算する．ポアソンの境界条件にはディリクレ境界条件を適用する．

圧力項は圧力を変数とした，上記の式を $\boldsymbol{Ax} = \boldsymbol{b}$ に対応させたものを解く．空間解像度を n とすると，$\boldsymbol{A}$ は $n \times n$ 行列であり，$\boldsymbol{b}$ は n 次元ベクトルである．以下では上記の式の離散化方法について説明する．行列 $\mathbf{A}$ は式 3.6 と同様に離散ラプラシアンを用いて $\mathbf{A} = \frac{\Delta t}{\rho \Delta x}\mathbf{L}$ と計算できる．よって，$\mathbf{A}$ の (i, j) 成分 $\mathbf{A_{i,\,j}}$ の各成分は以下のように定義される．

$$\mathbf{A}_{i,\,j} = \begin{cases} 6\frac{\Delta t}{\rho \Delta x} & i = j \\ -\frac{\Delta t}{\rho \Delta x} & v_i \text{と } v_j \text{が接続している} \\ 0 & \text{その他} \end{cases}$$

次に，ベクトル $\boldsymbol{b}$ の計算方法について説明する．3 次元のスタッガード格子における流体シミュレーションでは，速度場 $\mathbf{u} = (u_x, u_y, u_z)$ の発散 $\nabla \cdot \mathbf{u}$ を行列 $\mathbf{W}$ を用いて

$$\nabla \cdot \mathbf{u} = \mathbf{W}\mathbf{u}$$

と離散化する．3 次元のスタッガード格子では，発散は以下の中央差分で近似される．

$$\nabla \cdot \mathbf{u} \approx \frac{u_x(i+1, j, k) - u_(\, i, j, k)}{\Delta x} + \frac{u_y(i, j+1, k) - u_y(i, j, k)}{\Delta y} + \frac{u_z(i, j, k+1) - u_z(i, j, k)}{\Delta z}$$

ただし，本実験では $\Delta x = \Delta y = \Delta z$ である．これを行列形式で表現するために，行列 $\mathbf{W}$ の各成分 $W_{i,\,j}$ を次のように定義する．

$$W_{i,\,j} = \begin{cases} \frac{1}{\Delta x}, & u_x(i+1, j, k)\,, u_y(i, j+1, k)\,, u_z(i, j, k+1) \text{ に対応する成分} \\ -\frac{1}{\Delta x}, & u_x(i, j, k)\,, u_y(i, j, k)\,, u_z(i, j, k) \text{ に対応する成分} \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

以上により，ベクトル $\boldsymbol{b}$ は速度ベクトル $\boldsymbol{u_2}$ と行列 $\mathbf{W}$ を用いて，$\boldsymbol{b} = \mathbf{W}\boldsymbol{u_2}$ と表せる．したがって，実際に解く線形方程式は $\mathbf{A}\boldsymbol{p} = \mathbf{W}\boldsymbol{u_2}$ となる．

行列 $\mathbf{A}$ は規模が大きいため，反復法を用いて計算する．$\mathbf{A}$ のほとんどの成分は 0 であるため，疎行列の反復法ソルバーを用いることで空間計算量や時間計算量を削減できる．反復法では，必要な精度に解が収束したら計算を打ち切る．ここでの精度は反復回数や視覚的に影響を与えるほか，低い精度では流体の非圧縮性条件を満たさなくなってしまう恐れがある．行列 $\mathbf{A}$ が正定値対称行列になるため，前処理付き共役勾配法が広く用いられている．文献 [6] のフラクショナルステップ法における計算時間のボトルネックは圧力項計算であり，解の収束が早いほど反復回数が少なくなり，計算負荷が少なくなる．

次に，圧力の勾配ベクトル $\nabla \boldsymbol{p}$ が速度に与える影響を考える．圧力勾配を行列 $\mathbf{Y}$ を用いて離散化し，以下のように表せることを説明する．

$$\nabla p = \mathbf{Y} p \tag{3.9}$$

3 次元格子では，圧力勾配 $\nabla p = (\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z})$ は，各方向において以下のように離散化される：

$$\frac{\partial p}{\partial x} \approx \frac{p_{i+1,j,k} - p_{i,j,k}}{\Delta x}, \quad \frac{\partial p}{\partial y} \approx \frac{p_{i,j+1,k} - p_{i,j,k}}{\Delta y}, \quad \frac{\partial p}{\partial z} \approx \frac{p_{i,j,k+1} - p_{i,j,k}}{\Delta z} \tag{3.10}$$

ただし，本実験では $\Delta x = \Delta y = \Delta z$ である．これを行列形式で表現するために，行列 $\mathbf{Y}$ の各成分 $Y_{m,n}$ を次のように定義する．

$$Y_{m,n} = \begin{cases} \frac{1}{\Delta x}, & \nabla p_x(i+1, j, k)\ , \nabla p_y(i, j+1, k)\ , \nabla p_z(i, j, k+1) \text{ に対応する成分} \\ -\frac{1}{\Delta x}, & \nabla p_x(i, j, k)\ , \nabla p_y(i, j, k)\ , \nabla p_z(i, j, k) \text{ に対応する成分} \\ 0, & \text{otherwise} \end{cases}$$

この行列 $\mathbf{Y}$ を用いることで，速度場の更新に必要な圧力勾配を効率的に計算することが可能となる．

以上のことから，圧力項の計算をまとめると，以下のようになる．

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

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

$$\boldsymbol{u_3} = \boldsymbol{u_2} - \mathbf{Y}\boldsymbol{p}$$

3.2 ボリュームデータの可視化手法

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．代表的な手法として，サンプルデータをスライスごとに投影するスライスベースの手法が存在する．スライスベースの手法は計算負荷が低くリアルタイム処理に適している．本章では，スライスベースのボリュームレンダリングのアルゴリズムと，各スライスのテクスチャの計算方法を紹介する．

スライスベースのボリュームレンダリングは，シミュレーション空間がボクセル空間として捉えられることを利用し，気体の密度を用いてボクセルの透明度や色を計算する．その後，シミュレーション空間をボクセルの透明度や色をテクスチャとしてマッピングした複数の平面（スライス）に分割し，それらを順次描画することで最終的な画像を生成する．以下では，ボクセルの透明度と，最終的な画像の画素値を計算する方法を説明する．

視線方向に沿ったスライス間の距離を Δs とし，i 番目のスライスと視線の交点の位置を $i\Delta s$ とする．ボクセルの透明度 $t(i\Delta s)$ と不透明度 $\alpha(i\Delta)$ は，以下の式で計算する．

$$t(i\Delta s) = \exp(-\rho(\mathrm{i}\Delta\mathrm{s})\Delta\mathrm{s})$$

$$\alpha(i\Delta s) = 1 - t(i\Delta s)$$

$\rho(i\Delta s)$ は，流体シミュレーションによって計算した，位置 $i\Delta s$ の密度である．また，Δs は視線とスライスの法線がなす角 θ とすると以下のように変化するため，Δs はテクスチャ座標によって異なる．

$$\Delta s' = \Delta s/\cos\theta \tag{3.11}$$

透明度を考慮するレンダリング手法の代表的なものとして，アルファブレンディングが存在する．アルファブレンディングは以下の式で計算する．

$$I_n(\boldsymbol{x}) = \alpha_n(\boldsymbol{x})c_n(\boldsymbol{x}) + (1 - \alpha_n(\boldsymbol{x}))I_{n-1}(\boldsymbol{x}) \tag{3.12}$$

- $I_n(\boldsymbol{x})$: n 番目のスライスの位置 $\boldsymbol{x}$ における累積輝度
- $c_n(\boldsymbol{x})$: n 番目のスライスの位置 $\boldsymbol{x}$ における輝度
- $\alpha_n(\boldsymbol{x})$: n 番目のスライスの位置 $\boldsymbol{x}$ における不透明度

式 3.12 の不透明度に $\alpha(i\Delta s')$ を用いることで，アルファブレンディングによってシミュレーション空間がボリュームレンダリングできる．

ここで，シミュレーション空間をスライスによって分割する方向は，視線方向に垂直な方向と，x, y, z 軸の中で視線になるべく垂直な方向が考えられる．一般的に視線とスライスの交点がボクセルの計算点上にないため，$\rho(i\Delta s)$ を補間して得る必要がある．前者は式 3.11 による Δs の変化量が透視投影の影響しか受けない．一方，後者は視線とスライスの法線がなす角は視点に依存し，より式 3.11 による Δs の変化量が大きい．また，x, y, z 軸の方向による分割は視線の方向によって視線と交差するスライスが少なくなり，視覚的に不自然なスライスの形状が現れることがある．CG の分野においては後者の手法でも見た目上問題ないことが多く，文献 [6] では後者の手法を用いている．

第4章　部分空間法

本章では，スタッガード格子を用いた煙の流体シミュレーション [6] に対して，線形項と非線形項の計算を部分空間上に射影して計算する手法 [9] を適用する方法を説明する．

4.1　基底空間構築手法

スタッガード格子を用いた流体シミュレーションの，時刻 $t\{t \in N, 0 \leq t \leq m\}$ の流速データ $\boldsymbol{u_t}$ と圧力データ $\boldsymbol{p}$ を用いて，以下の Snapshot 行列 $\mathbf{S_u}$ と $\mathbf{S_p}$ を定義する．

$$\mathbf{S_u} = \begin{bmatrix} | & | & & | \\ \boldsymbol{u}_0 & \boldsymbol{u}_1 & \cdots & \boldsymbol{u}_{m-1} \\ | & | & & | \end{bmatrix}$$

$$\mathbf{S_p} = \begin{bmatrix} | & | & & | \\ \boldsymbol{p}_0 & \boldsymbol{p}_1 & \cdots & \boldsymbol{p}_{m-1} \\ | & | & & | \end{bmatrix}$$

ここで，x, y, z 軸方向の分割数を nx, ny, nz($nx = ny = nz$) とすると，$\boldsymbol{u_t}$ と $\boldsymbol{p}$ の次元はそれぞれ $3\,(nx+1) \times nx \times nx$，$nx \times nx \times nx$ である．Snapshot 行列は空間計算量が大きく，直接特異値分解を適用すると空間計算量の制約を満たさない場合がある．これを避けるため，QR 分解を利用する手法を説明する．また，特異値分解を用いた次元削減はの条件数は，行列を構成する列ベクトル同士の相関が強いほど悪くなることが知られている．Snapshot 行列はデータ間に相関があるため，直接特異値分解を適応すると条件数が大きくなり，計算が不安定になる問題を緩和することができる．

まず，$n \times m$ 行列 $\mathbf{S}$ を以下のように QR 分解する．

$$\mathbf{S} = \mathbf{QR}$$

ここで，行列 $\mathbf{Q}$ は $n \times m$ 直交行列であり，行列 $\mathbf{R}$ は $m \times m$ 上三角行列である．その後，行列 $\mathbf{R}$ を以下のように特異値分解する．

$$\mathbf{R} = \mathbf{U_R \Sigma_R V_R}^T$$

ここで，$\mathbf{S^T S}$ を計算する．

$$\mathbf{S^T S} = \mathbf{V_R \Sigma_R}^T \mathbf{U_R}^T \mathbf{Q}^T (\mathbf{Q U_R \Sigma_R V_R}^T)$$

$\mathbf{Q}$ と $\mathbf{U_R}$ は直交行列であることから，以下が得られる．

$$\mathbf{S^T S} = \mathbf{V_R \Sigma_R^2 V_R}^T$$

2.2 にて $\mathbf{S^T S} = \mathbf{V \Sigma^2 V}^T$ となることを利用して基底を求めることができることを紹介した．同様にして QR 分解後の特異値分解を利用して基底を求めることができる．この手法により，特異値分解する行列は各列に相関がある $n \times m$ 行列から QR 分解後の $m \times m$ 上三角行列になり，空間計算量と行列の条件数が改善される．

4.2 フラクショナルステップ法への部分空間法の適用

本節では，2.1 にて得たフラクショナルステップ法による以下のシミュレーション計算に，部分空間法を適用する方法を説明する．

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

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

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

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

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

$$\boldsymbol{u_3} = \boldsymbol{u_2} - \mathbf{Y} \boldsymbol{p}$$

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 Snapshot 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 $\mathbf{U_i}$，$\boldsymbol{p}$ から得られた基底を $\mathbf{P}$ 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．

$$\mathbf{U_2^T} \boldsymbol{u_2} = (\mathbf{U_2^T V U_1})\, \mathbf{U_1^T} \boldsymbol{u_1} \qquad \tilde{\boldsymbol{u_2}} = \tilde{\mathbf{V}} \tilde{\boldsymbol{u_1}}$$

$$\mathbf{P^T} \boldsymbol{b} = (\mathbf{P^T W U_2})\, \mathbf{U_2^T} \boldsymbol{u_2} \qquad \tilde{\boldsymbol{b}} = \tilde{\mathbf{W}} \tilde{\boldsymbol{u_2}}$$

$$\mathbf{P^T} \boldsymbol{p} = (\mathbf{P^T A^{-1} P})\, \mathbf{P^T} \boldsymbol{b} \qquad \tilde{\boldsymbol{p}} = \tilde{\mathbf{A}} \tilde{\boldsymbol{b}}$$

$$\mathbf{U_3^T} \boldsymbol{u_3} = \mathbf{U_2^T} \boldsymbol{u_2} - (\mathbf{U_3^T Y P})\, \mathbf{P^T} \boldsymbol{p} \qquad \tilde{\boldsymbol{u_3}} = \tilde{\boldsymbol{u_2}} - \tilde{\mathbf{Y}} \tilde{\boldsymbol{p}}$$

ただし，$\mathbf{P^T A^{-1} P} = (\mathbf{P^T A P})^{-1}$ であり，圧力ベクトルの次元数を n として $\mathbf{A}$ は $n \times n$ 疎行列であり，$(\mathbf{P^T A P})$ は $m \times m$ 密行列である．密行列であることに注意が必要だが，逆行列を高速に求められるため，$\tilde{\mathbf{A}} = (\mathbf{P^T A P})^{-1}$ とする．

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 2.2.4 にて紹介した Cubature 法を用いて計算する．Cubature 法に用いる基底は，外力項は $\mathbf{U}_0$ であり，移流項は $\mathbf{U}_1$ 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．

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

4.3 部分空間法の課題

部分空間法はシミュレーションの次元を n 次元から r 次元に削減するため，各ステップの計算負荷を大幅に減らすことができる手法である．その一方で，前処理として基底の計算や Cubature 法のランダムサンプリングと重み計算，シミュレーションに用いる行列の部分空間への射影などを行う必要がある．これらの計算負荷は Snapshot の数に依存しているが，Snapshot の数以下の次元にしか次元削減できず，流体シミュレーションとして意味のある結果を得るためには Snapshot の数が十分でなければならない．また，Snapshot 行列の空間解像度が大きいという問題点がある．例えば空間解像度 $64 \times 64 \times 64$，Snapshot 数 50 とすると，流速の Snapshot 行列の空間解像度は 10GB 程度になる．GPU による高速化を併用する際，GPU が 1 度のデータ通信によって処理し切ることができる空間解像度の制限を超えてしまうことで計算時間が増加することが考えられる．

他にも，事前計算したシミュレーションと異なるシミュレーションを行うことが困難であり，境界条件や流体の初期化，与える外力を大幅に変更することはできず，別途シミュレーションデータを用意する必要がある．

第5章　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Snapshotの2乗に比例し，前処理の大多数の時間を占めている．この章では部分空間法の前処理の高速化を達成するため，Snapshotをd個に分割する手法を提案する．

5.1　Snapshotの分割による高速化

部分空間法の基底の計算に用いるSnapshot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Snapshot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．

5.2　計算機実験

本研究では，コンピュータグラフィックスライブラリOpenGLとc++，線形代数ライブラリEigenを用いて，文献[6]の煙の流体シミュレーションとスライスベースのボリュームレンダリング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．Cubature法のNNLSソルバにはEigenのUnsupportedライブラリのソルバを用いて実装した．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．データの精度は非線形項の部分空間への射影手法[8]を参考に，L2誤差を用いて評価を行なった．ここでL2誤差は次元削減前後のベクトル$\boldsymbol{u}$と$\tilde{\boldsymbol{u}}$について，以下のように表される．

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

表 5.1: 基底計算における解像度と分割数ごとの実行時間（ms）

解像度	分割なし	分割数 2	分割数 4	分割数 10
64^3	23,428	14,564	8,981	3,780
128^3	190,541	118,489	69,615	33,427

表 5.2: 行列の射影における解像度と分割数ごとの実行時間（ms）

解像度	分割なし	分割数 2	分割数 4	分割数 10
64^3	3,514	2,823	846	666
128^3	5,206	3,061	3,050	3,975

表 5.1 は基底の計算にかかった実行時間，表 5.2 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．

また，図 5.1，図 5.2 は，解像度 64^3，128^3 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．Snapshot 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 5.3，5.4 は，L_2 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．L_2 誤差によって煙の密度分布に変化が見られる一方で，部分空間の適用前後の変化と比べると変化量は小さい．

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 Snapshot 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．

5.3 考察

分割なしと比べて精度が向上している点については，分割なしでは抽出されなかったモードを抽出することができていることによって，シミュレーション後半の流れを詳細に

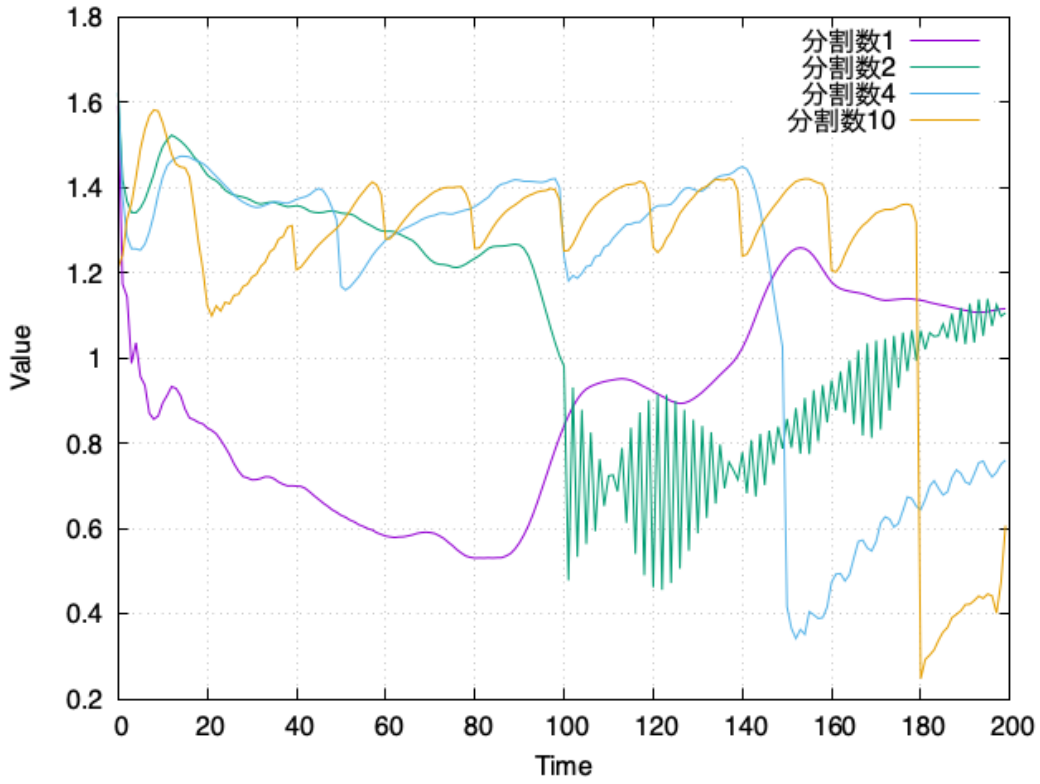


図 5.1: 解像度 64^3における L_2誤差

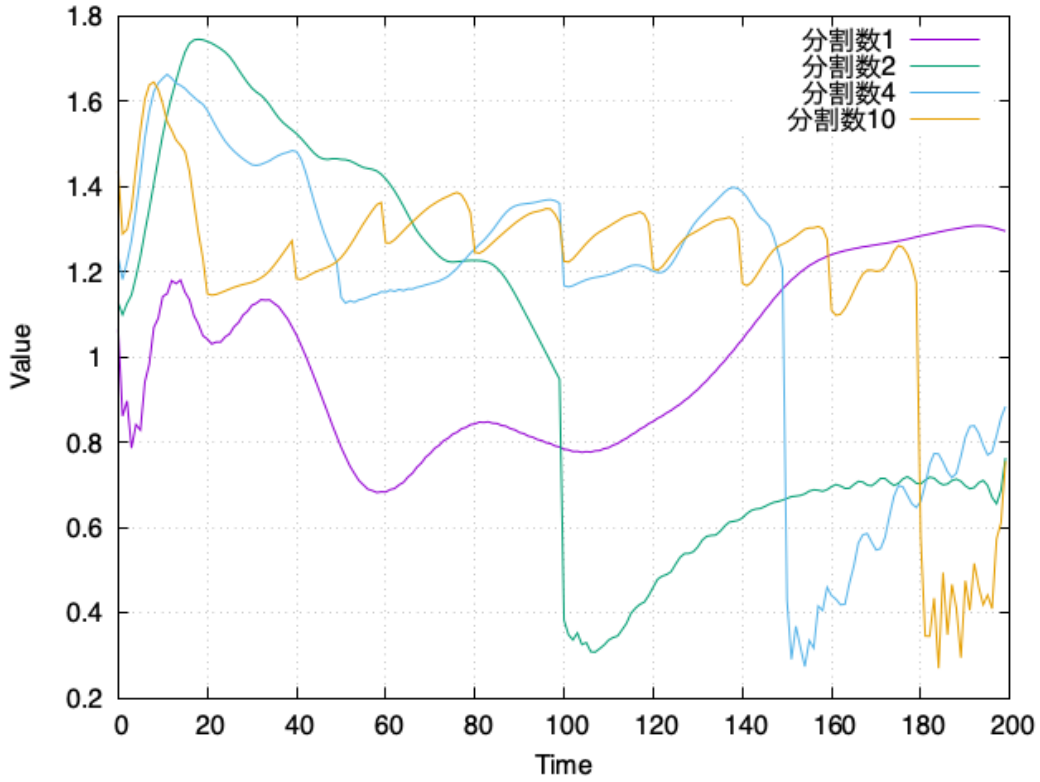
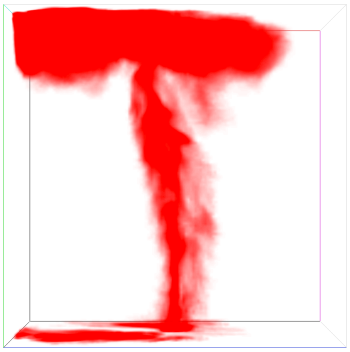
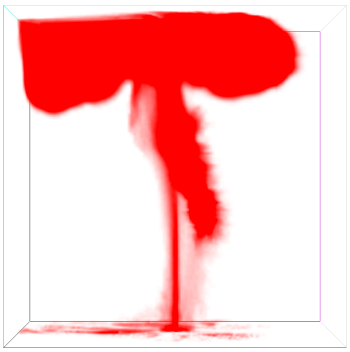


図 5.2: 解像度 128^3における L_2誤差

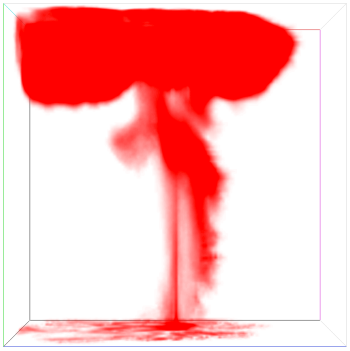
図 5.3: 解像度 128^3, 99 フレーム目



(a) gruound truth

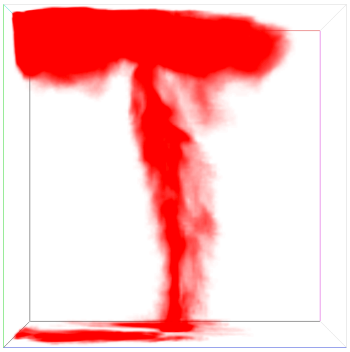


(b) 分割数 1

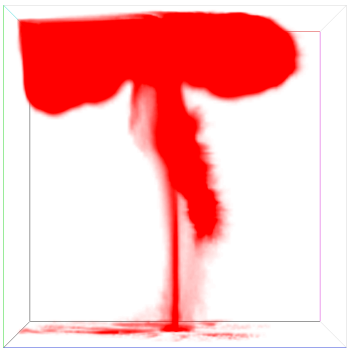


(c) 分割数 2

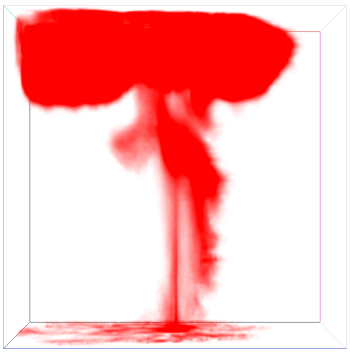
図 5.4: 解像度 128^3, 100 フレーム目



(a) gruound truth



(b) 分割数 1



(c) 分割数 2

シミュレーションできていると考えられる．特異値分解を用いた次元削減はの条件数は，行列を構成する列ベクトル同士の相関が強いほど悪くなることが知られている．本実験のシミュレーションの開始直後においては，タイムステップごとの流速や圧力の変化が大きくSnapshot間の相関関係が弱い．一方でシミュレーション終了付近においては，タイムステップごとの流速や圧力の変化が大きくSnapshot間の相関関係が強い．Snapshotの分割によって各基底の最大特異値が小さくなることによって，分割前と比較してシミュレーション後半に対応する特異値に関する条件数が改善され，詳細にシミュレーションできていると考えられる．

第6章　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，Snapshot 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．Snapshot 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 [10] が研究されている．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．

謝辞

本研究を進めるにあたり，大変多くのご指導，ご助言を頂いた中央大学理工学部情報工学科の森口昌樹 准教授に深く感謝いたします．3年間に渡り，研究の進め方や，開発方法について大変多くのご指導をいただきました．また，多大なるご助言，ご協力を頂いた形状情報処理研究室の皆様には大変お世話になりました．心から感謝いたします．最後に，研究生活を支えてくれた家族にも感謝いたします．

参考文献

[1] Y. Teng, David I.W. Levin, T. Kim.Eulerian Solid-Fluid Coupling. *ACM Transactions on Graphics*, 35, (6): 200:1–200:8, 2016.

[2] F. H. Harlow and J. E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with a free surface. *Physics of Fluids*, 8（12）: 2182–2189, 1965.

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

[4] J.U.Brackbill. D.B.KotheH.M.Ruppel. Flip: A low-dissipation, particle-in-cell method for fluid flow, *Computer Physics Communications*, 48（1）,25–38, January 1988.

[5] K. Iwama, A. Kawachi, and S. Yamashita, The affine particle-in-cell method, IPSJ Digital Courier, *ACM Transactions on Graphics*, 82,（4）, 51∶1–51∶10 August 2015.

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

[7] J. Stam. Stable Fluids. In *SIGGRAPH 99 Conference Proceedings, Annual Conference Series*, pages 121–128, 1999.

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

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

[10] A. Jones, P. Sen, and T. Kim. Compressing fluid subspaces. *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 77–84, 2016.