粒子法とウェーブレット解析による リアルタイム乱流シミュレーション

A Fast Simulation Method Using SPH and Wavelet for Turbulent Flow

藤澤誠　　加藤博一
Makoto FUJISAWA and Hirokazu KATO
奈良先端科学技術大学院大学
Graduate School of Information Science, NAIST
E-mail: {fujis,kato}@is.naist.jp

1 はじめに

複雑な流体現象 (煙や炎などの気体，波や川の流れのような液体など) の CG アニメーションを作成するために流体シミュレーションが広く用いられている．しかし，これまでの流体シミュレーション手法はそのほとんどが層流と対象としたものである．層流は流体が規則正しく揃って運動するような流れであり，不規則な流れは乱流と呼ばれる．層流はナビエ・ストークス方程式を数値的に解くことでその動きが再現されてきた．一方，我々の身近な流れのほとんどは解析がとても困難な乱流である．例えば，水道の蛇口から流れる水は水の量が少なければきれいに流速が揃って流れる (層流) が，少し蛇口をひねって水量を増やすと急激に流れが乱れる (乱流)．そのほかにも，砂浜に押し寄せる波や川の流れ，爆発や燃焼時の煙，車や船の後方にできる後流も乱流である．乱流は流れの一つの現象であり，どのような種類の流体 (液体，気体) でも，レイノルズ数が十分大きければ乱流になりえる．計算グリッドが十分細かければ，乱流も層流と同様にナビエ・ストークス方程式でその運動を予測することが可能である．しかし，現在においても乱流を完全に計算できるほどの性能を持つコンピュータは存在していない．そのため，流れの中から乱れの部分のみをとりだし，流れを平均流れと乱流モデルに分けて計算する手法 (例えば，レイノルズ平均流れ (RANS)，Large Eddy Simulation(LES) など) が研究されてきた．

CG の分野でもグリッド法を用いた流体シミュレーションに乱流を組み込む研究が近年盛んに行われている．ナビエ･ストークス方程式をグリッドで離散化することでベースとなる流れを生成し，ベースの流れから乱流を生成するためのエネルギーを計算する，もしくは，渦や渦エネルギーを移流させることで乱れの発生場所を算出する．しかし，これらの研究のほとんどはオフラインで実行され，リアルタイムで乱流を含む流体現象をシミュレーションできるものではない．

本論文では，粒子法による液体シミュレーションに GPU により高速化したウェーブレット解析とウェーブレットノイズを用いることで，乱流を高速に計算する手法を提案する．バックグラウンドグリッド等を用いず近傍粒子から直接，エネルギースペクトル値を計算し，ウェーブレットノイズにより乱流速度場の追加する．さらに，GPU を用いたウェーブレット解析，ウェーブレットノイズの実装も行った．

この論文の次節以降の構成は以下である．次章で関連研究を述べ，第 3 章でシミュレーション手法，そして，第 4 章において我々の手法を用いた結果を示し，これらのまとめを第 5 章で述べる．

2 関連研究

Stam[13] はセミラグランジュ移流法により，グリッド法で流れを安定して解く方法を開発した．これにより，グリッドを用いた流体計算が CG 分野で広く用いられるようになった．しかし，この手法は移流による数値拡散という潜在的な問題点をかかえている．数値拡散は乱流エネルギーの散逸を起こし，乱流場を捉えるのを難しくする．この数値拡散を抑えるために，BFFCC[5] や QUICK[8]，CIP 法 [6] といった高次の補間法を移流に用いる手法が提案されている．また，Fedkiw ら [3] は数値拡散で失われる渦を検出し，再注入する手法を提案し，Hong ら [4] はこれを SPH 法へ導入した．

一方，乱流モデルを用いて乱れを構成する渦を計算する研究もなされている．Stam[14] は Kolgomorov の理論に基づき乱流を手続き的に生成する方法を開発した．Bridson ら [1] はノイズ関数を用いて非圧縮乱流速度場を発生させることで，乱流のような流れを高速に生成する手法を提案した．これらの手法は流体力学に基づくものではなく，流れそのものはユーザの入力などに依存している．渦や流れのエネルギーを計算することで，従来の流体シミュレーションに乱流を付加することも可能である．Selle ら [11] は渦方程式に従い渦パーティクルを移流することで，グリッド上では数値拡散により失われる渦を再現し，Pfaff ら [10] は渦パーティクルの散布場所を壁面境界層のせん断流れから計算することで固体とのインタラクションにより発生す

る乱れを計算した．Kim ら [7] は粗いグリッドでの流体シミュレーションとウェーブレットノイズによる渦を組み合わせることで，高解像度の煙の乱流アニメーションを生成した．我々は Kim らの方法をパーティクル法に応用することで，リアルタイムでの乱流液体シミュレーション手法を開発する．

3　シミュレーション手法

全体的なシミュレーションの流れを以下に示す．

1. SPH 法により粘性拡散項，非圧縮項，重力項を計算
2. ウェーブレット解析により乱流エネルギースペクトル分布を算出
3. ウェーブレットノイズによる乱流場を追加
4. パーティクルの位置と速度を更新

SPH 法による各項の計算に加えて，ウェーブレット解析とウェーブレットノイズによる乱流を外力として追加することで，従来の粒子法に容易に乱流を導入可能である．この章の残りの部分でこれらの各ステップについて詳しく述べる．

3.1　SPH 法による流れの計算

流体の流れを計算するためにパーティクル法の一種である SPH 法を用いる．支配方程式である非圧縮のナビエ・ストークス方程式は以下である．

$$\nabla \cdot \boldsymbol{u} = 0, \tag{1}$$

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

ここで，$\boldsymbol{u}$ は流体速度，ν は動粘性係数，ρ は流体の密度，p は圧力，$\boldsymbol{f}$ は外力で重力，表面張力などが含まれる．支配方程式をパーティクルで離散化し，SPH 法を用いて解く．パーティクル自体が液体を表しているため，パーティクル質量が変化しないかぎり質量保存性が常に保持され (質量保存式である式 (1) を解く必要がない)，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追跡の必要性がないことが利点である．SPH 法による物理量 ϕ の離散化式を以下に示す．

$$\phi(\boldsymbol{x}) = \sum_{j \in N} m_j \frac{\phi_j}{\rho_j} W(\boldsymbol{x}_j - \boldsymbol{x}, h) \tag{3}$$

ここで，N は近傍パーティクルの集合，m はパーティクル質量，ρ はパーティクル密度，W はカーネル関数である．物理量の勾配 $\nabla\phi$ はカーネル関数の導関数を用いて表される．流体の密度は以下で計算される．

$$\rho(\boldsymbol{x}) = \sum_{j \in N} m_j W(\boldsymbol{x}_j - \boldsymbol{x}, h) \tag{4}$$

流体の圧力は基準となる密度 ρ_0 と状態方程式より，

$$p = k(\rho - \rho_0) \tag{5}$$

となる．ここで k は非圧縮性を制御するためのパラメータである．我々は Müller らの方法 [9] により式 (2) を解く．式 (2) を数値的に解くことで各パーティクルにかかる力が計算されるため，それに基づきパーティクルの位置，速度を Leap-Frog 法により更新する．

SPH 法における計算のボトルネックは近傍粒子の探索である．パーティクル数を N とすると，全探索では $O(N^2)$ で計算量が増加する．そのため，近傍粒子の探索には空間分割法 ($O(N \log N)$) が一般的に用いられている．空間分割法では計算領域をバケットと呼ばれる直交格子 (ボクセル) で分割し，各ボクセルにパーティクルのインデックスを格納する．そして，周囲のボクセルに格納されたパーティクルのみを探索することで全探索の必要性をなくす．本研究ではこの空間分割法を NVIDIA CUDA を用いて GPU 上で実装し，さらに，式 (2) の各項の計算も GPU 上で並列計算することで高速化を行った．

3.2　パーティクル速度のウェーブレット解析

流れのシミュレーションにおいて数値拡散で失われた，もしくは，離散化の解像度上表現できないような小スケールの渦を再構成することで乱流速度場を構成する．そのためには，シミュレーション上で再現できる大スケールの渦からのエネルギー遷移による，小スケール渦の生成を計算しなければならない．渦のエネルギー値 $\hat{e}$ は速度場 $\boldsymbol{u}$ の周波数変換 $\hat{\boldsymbol{u}}$ より，

$$\hat{e}(k) = \frac{1}{2}|\hat{\boldsymbol{u}}(k)|^2 \tag{6}$$

となる．k は波数である．乱れを必要なところに正確に生成するためには，渦エネルギー値は波数についてのみでなく，空間についてもその変化を計算しなければならないため，実際には，$\hat{e}(k, \boldsymbol{x})$ となる．

ある特定の座標における周波数変換値を求めるために窓フーリエ変換が広く用いられている．しかし，窓フーリエ変換では窓の大きさによる周波数分解能と空間分解能のトレードオフに悩まされることになる．[7] ではウェーブレット変換を用いることでこの問題を解決し，各グリッドでのエネルギースペクトル値を計算した．我々はパーティクル法を用いているためこれをそのまま用いることはできない．ひとつの方法としては空間にグリッドを定義し，パーティ

クルの速度場を投影する方法がある．しかし，グリッドのための余計なメモリや計算時間がかかる上に，投影時の補間による数値拡散も問題となる．これを解決するために，バックグラウンドグリッドを用いるのではなく，近傍パーティクルの速度場から直接，周波数空間の速度場 $\hat{\boldsymbol{u}} = (\hat{u}, \hat{v}, \hat{w})$ およびあるスケール s でのエネルギースペクトル $\hat{e}(1/s, \boldsymbol{x})$ を求める．

x 方向速度場 u の連続ウェーブレット変換式を以下に示す．

$$\hat{u}(s,a,b,c) = \frac{1}{\sqrt{s}} \iiint_{-\infty}^{\infty} u(\boldsymbol{x}) \psi \left(\frac{x-a}{s}, \frac{y-b}{s}, \frac{z-c}{s} \right) dxdydz \quad (7)$$

ここで s はウェーブレットスケール，a, b, c は平行移動量である．ψ はウェーブレット関数である．グリッドを用いた場合，周囲のグリッドの値を畳み込みすることでウェーブレット変換値を求める．これをパーティクル法で離散化する．SPH 法では近傍パーティクル j の重み付き和を用いて物理量を定義する．同様にして，ウェーブレット変換値も近傍 j をウェーブレット関数によって重み付けし，積算して求める．パーティクル i の x 方向速度 u_i に関するウェーブレット変換は，

$$\hat{u}_i = \frac{1}{\sqrt{s}\psi_{sum}} \sum_j u_j \psi \left(\frac{x_i - x_j}{s}, \frac{y_i - y_j}{s}, \frac{z_i - z_j}{s} \right). \quad (8)$$

ここで ψ_{sum} は，

$$\psi_{sum} = \sum_j \psi \quad (9)$$

である．グリッドを用いた場合は周囲のセル数が一定値となるが，パーティクル法ではパーティクルの分布によりばらつきが発生する．特に表面付近でパーティクルが少ないため，最近傍に存在する少数のパーティクルによる影響が大きくなり，結果として表面におけるウェーブレット変換値が大きくなる現象が発生する．それを防ぐために，ψ_{sum} を導入した．また，近傍探索には SPH 法において構築したバケットを用い，各パーティクルごとに GPU で並列に計算する．$\hat{v}, \hat{w}$ についても同様にして計算し，その結果を式 (6) に代入することで各パーティクルの渦のエネルギー値 $e_i(k)$ が計算される．この値を次節で述べるウェーブレット乱流速度場に掛けることで，小スケールの渦が発生すべき場所を指定する．

パーティクル速度をバックグラウンドに定義したグリッドに投影し，式 (7) によりウェーブレット変換を行った結果との比較を図 1 に示す．初期条件として矩形境界内の左下に矩形状の液体を配置した．図 1 はシミュレーション中の 1 フレームである．青 → 赤でエネルギー値が大きくなる．我々のパーティクルによる渦エネルギー値の算出法はグリッドに投影する方法とほぼ同様の結果が得られる．ただし，まだパーティクル密度によって値が変化する現象も見られた．これはパーティクル密度の均一化 [12] や密度による参照半径の修正などで解決できると考えられる．

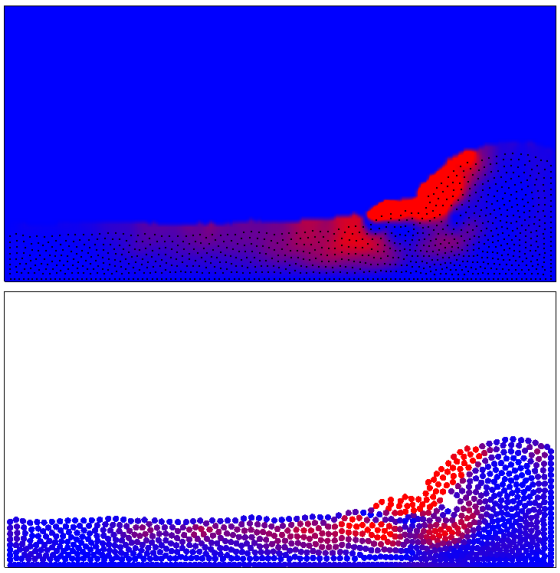

図 1: グリッド (上段) とパーティクル (下段) から算出されたエネルギースペクトル値の比較．値が高いグリッド，パーティクルは赤，低いものは青で描画している．

3.3 ウェーブレット乱流速度場

得られたあるスケールでのエネルギースペクトル値からより小さいスケールの非圧縮乱流速度場を求める．[1] で示されるように非圧縮乱流速度場は，ノイズ関数 ω を用いて以下のように定義される．

$$\boldsymbol{w}(\boldsymbol{x}) = \left(\frac{\partial w_3}{\partial y} - \frac{\partial w_2}{\partial z}, \frac{\partial w_1}{\partial z} - \frac{\partial w_3}{\partial x}, \frac{\partial w_2}{\partial x} - \frac{\partial w_1}{\partial y} \right) \quad (10)$$

ω にはウェーブレットノイズ [2] を用いる．ウェーブレットノイズはノイズタイルの 2 次の B スプライン補間により値を算出しているため，B スプライン重み係数を必要な軸方向に対して変更するだけで，その導関数を求めることができる．ウェーブレットノイズの計算は CPU でも高速に計算できるが，乱流を生成する際には複数帯域のウェーブレットノイズを生成する必要があり，計算量が増大する．よってノイズタイルからの補間部分を GPU を用いて並列化することでウェーブレットノイズを高速に計算した．

乱流を構成する渦はエネルギー遷移によりより細かな渦に分裂する (forward scattering)，もしくは，合流して大きな渦になる (back scattering)．この乱流エネルギー遷移を表現するために Kolgomorov スペクトルを用いる．Kolgomorov の理論に基づき，波数空間における乱流のエネルギースペクトルは以下のようになる．

$$\hat{e}(k) = \alpha \epsilon^{\frac{2}{3}} k^{-\frac{5}{3}} \quad (11)$$

ここで，α は Kolgomorov 定数，ϵ は平均エネルギー散逸量である．式 (11) より，乱流エネルギースペクトルは波数 k の $-5/3$ 乗となる．このことから波数が 2 倍となった場合のエネルギースペクトルは以下のようになる．

$$\hat{e}(2k) = \hat{e}(k)2^{-\frac{5}{3}} \tag{12}$$

ただし，$e(1) = C\epsilon^{\frac{2}{3}}$ である．式 (12) を式 (6) に代入することで，速度場のウェーブレット変換に対する乱流エネルギーの遷移率が $2^{-5/6}$ に比例することが分かる [7]，各周波数帯域での乱流速度場を式 (10) で求めるので，最終的な乱流速度場 $\boldsymbol{y}(\boldsymbol{x})$ は，

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{i=i_{min}}^{i_{max}} \boldsymbol{w}(2^i\boldsymbol{x})2^{-\frac{5}{6}(i-i_{min})} \tag{13}$$

である．$[i_{min}, i_{max}]$ はスペクトルバンドの幅であり，本研究ではパーティクルスケールの乱流を生成するため，i_{max} はパーティクル直径 d を最小スケールとした解像度となるように，i_{min} は式 (8) で用いたウェーブレットスケール値 s に基づき計算する．

式 (13) と前節で計算した $\hat{e}_i(k)$ を用いることで，乱流による力は，

$$\boldsymbol{f}_i^{turb} = A\frac{\rho_i}{\Delta t}\hat{e}_i(k)\boldsymbol{y}(\boldsymbol{x}_i) \tag{14}$$

となる．ここで，A はユーザが乱流の大きさを制御するためのパラメータである．

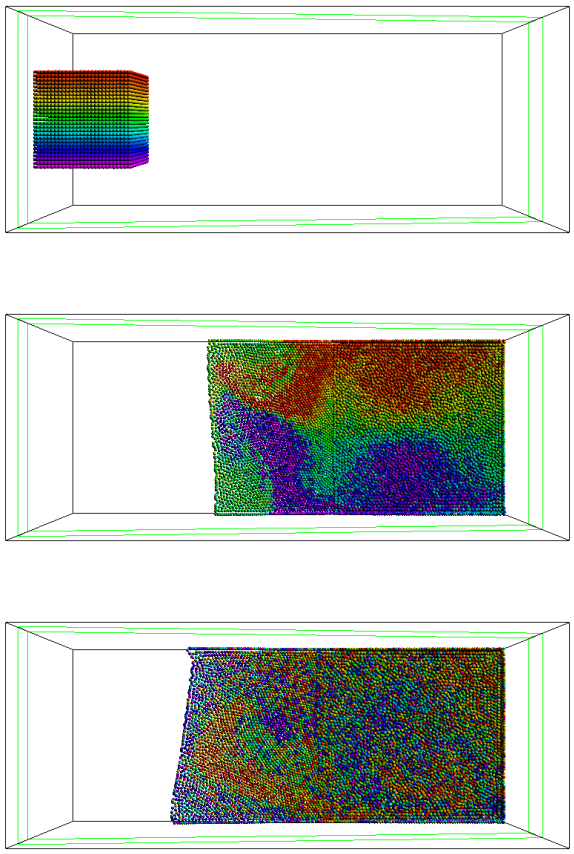

図 2: 乱流による攪拌効果 (上段:初期状態，中段:乱流なし，下段:乱流あり)

4 結果

提案手法を GPU 上で実装した結果を示す．実行環境は，CPU:Core 2 Duo 3.16GHz, GPU:GeForce GTX285 である．手法のほとんどの部分は NVIDIA CUDA を用いて GPU 上で実装されている．また，ウェーブレット関数には Mexican Hat を用いた．

図 4 は 3 次元シミュレーションの結果である．立方体状の液体を緩やかな下り坂に落下させた．左列はウェーブレット乱流なし，右列はウェーブレット乱流有りの結果である．パーティクル数は約 20000，ウェーブレットスケールはパーティクル直径の 3 倍とした．GPU 上で更新された粒子位置とその近傍情報から密度値を算出し，Marching Cube 法により等値面をメッシュ化した．平均シミュレーション時間は乱流なしの場合で約 20 ミリ秒/フレーム，乱流ありで約 40 ミリ秒/フレーム，MC 法によるメッシュ化，レンダリングまで含めた平均計算時間は約 60 ミリ秒/フレームであった．乱流を追加した場合でも 16fps 以上であり，リアルタイムでの実行が可能である．

図 2 は図 4 と同じシーンについて初期配置の深さ方向に対して色をグラデーションさせたポイントスプライトで粒子を描画した計算結果である．視点は計算空間上方に配置し，下の 2 段はシミュレーションが十分進んだ後のフレームの画像を示している．乱流なし (図 2 中段) では流れが均一になるため流れた後もグラデーションが保たれている．我々の方法で乱流効果を加えた場合，図 2 下段のように流体が攪拌される．これは流体宙に浮かぶ固体などをシミュレーションしたときのその動きに大きく影響する．

図 3 は一様な流れの中に球状の固体を配置したシミュレーション結果である．粒子数は 40000 で平均シミュレーション時間は約 60 ミリ秒/フレーム，レンダリングまで含めた平均計算時間は約 150 ミリ秒/フレームである．メッシュ化解像度を図 4 の 2 倍にしたため，レンダリング時間がかかっている．球状固体の後方の流れ (後流) が特に大きく乱れている．

5 まとめと今後の課題

本論文では，粒子法によるリアルタイム乱流シミュレーション手法を提案した．近傍粒子の速度を用いたウェーブレット解析による渦のエネルギースペクトル値の直接計算を行い，その結果から乱流を外力として付加することで，SPH 法による液体のシミュレーションに乱流を加えた．ま

た，Kolgomorov の理論に基づき複数の帯域のウェーブレットノイズを合成することで乱流速度場を生成した．そして，SPH 法，ウェーブレット解析，ウェーブレットノイズの生成の GPU 実装により，数万粒子においてリアルタイムでの乱流計算を達成した．

今後の課題としては，より小さいスケールの渦の再現があげられる．現在はパーティクルスケール以下の渦は表現不可能であるため，より細かなパーティクルの再散布による表面追跡やレンダリング表面生成時の修正などによりサブパーティクルスケールでの乱流アニメーションを生成したい．

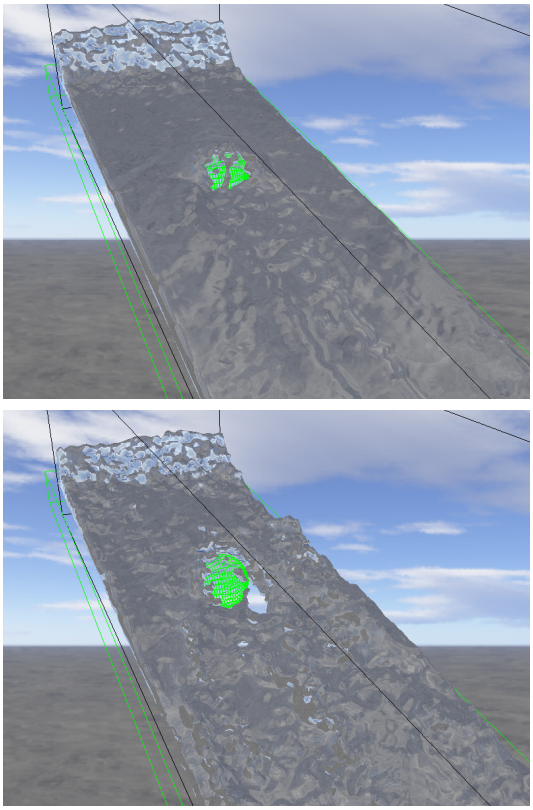

図 3: 流れの中に球状の固体を配置 (上段:乱流なし，下段:乱流あり)

参考文献

[1] Bridson, R., Houriham, J. and Nordenstam, M.: Curl-noise for procedural fluid flow, *Proc. SIGGRAPH 2007*, p. 46 (2007).

[2] Cook, R. L. and DeRose, T.: Wavelet noise, *Proc. SIGGRAPH 2005*, pp. 803–811 (2005).

[3] Fedkiw, R., Stam, J. and Jensen, H.: Visual simulation of smoke, *Proc. SIGGRAPH 2001*, pp. 15–22 (2001).

[4] Hong, J.-M., Lee, H.-Y., Yoon, J.-C. and Kim, C.-H.: Bubbles alive, *Proc. SIGGRAPH 2008*, pp. 1–4 (2008).

[5] Kim, B., Liu, Y., Llamas, I. and Rossignac, J.: FlowFixer: Using BFECC for Fluid Simulation, *Eurographics Workshop on Natural Phenomena* (2005).

[6] Kim, D., young Song, O. and Ko, H.-S.: A Semi-Lagrangian CIP Fluid Solver without Dimensional Splitting, *Computer Graphics Forum (Proc. Eurographics)*, Vol. 27, No. 2, pp. 467–475 (2008).

[7] Kim, T., Thürey, N., James, D. and Gross, M.: Wavelet turbulence for fluid simulation, *Proc. SIGGRAPH 2008*, pp. 1–6 (2008).

[8] Molemaker, J., Cohen, J. M., Patel, S. and yong Noh, J.: Low Viscosity Flow Simulations for Animation, *Proc. ACM/Eurographics Symposium on Computer Animation 2008* (2008).

[9] Müller, M., Charypar, D. and Gross, M.: Particle-Based Fluid Simulation for Interactive Applications, *Proc. ACM/Eurographics Symposium on Computer Animation 2003*, pp. 154–159 (2003).

[10] Pfaff, T., Thuerey, N., Selle, A. and Gross, M.: Synthetic turbulence using artificial boundary layers, *Proc. SIGGRAPH Asia 2009*, New York, NY, USA, ACM, pp. 1–10 (2009).

[11] Selle, A., Rasmussen, N. and Fedkiw, R.: A vortex particle method for smoke, water and explosions, *Proc. SIGGRAPH 2005*, pp. 910–914 (2005).

[12] Solenthaler, B. and Pajarola, R.: Predictive-corrective incompressible SPH, *Proc. SIGGRAPH 2009*, pp. 1–6 (2009).

[13] Stam, J.: Stable fluids, *Proc. SIGGRAPH 1999*, pp. 121–128 (1999).

[14] Stam, J. and Fiume, E.: Turbulent wind fields for gaseous phenomena, *Proc. SIGGRAPH1993*, pp. 369–376 (1993).

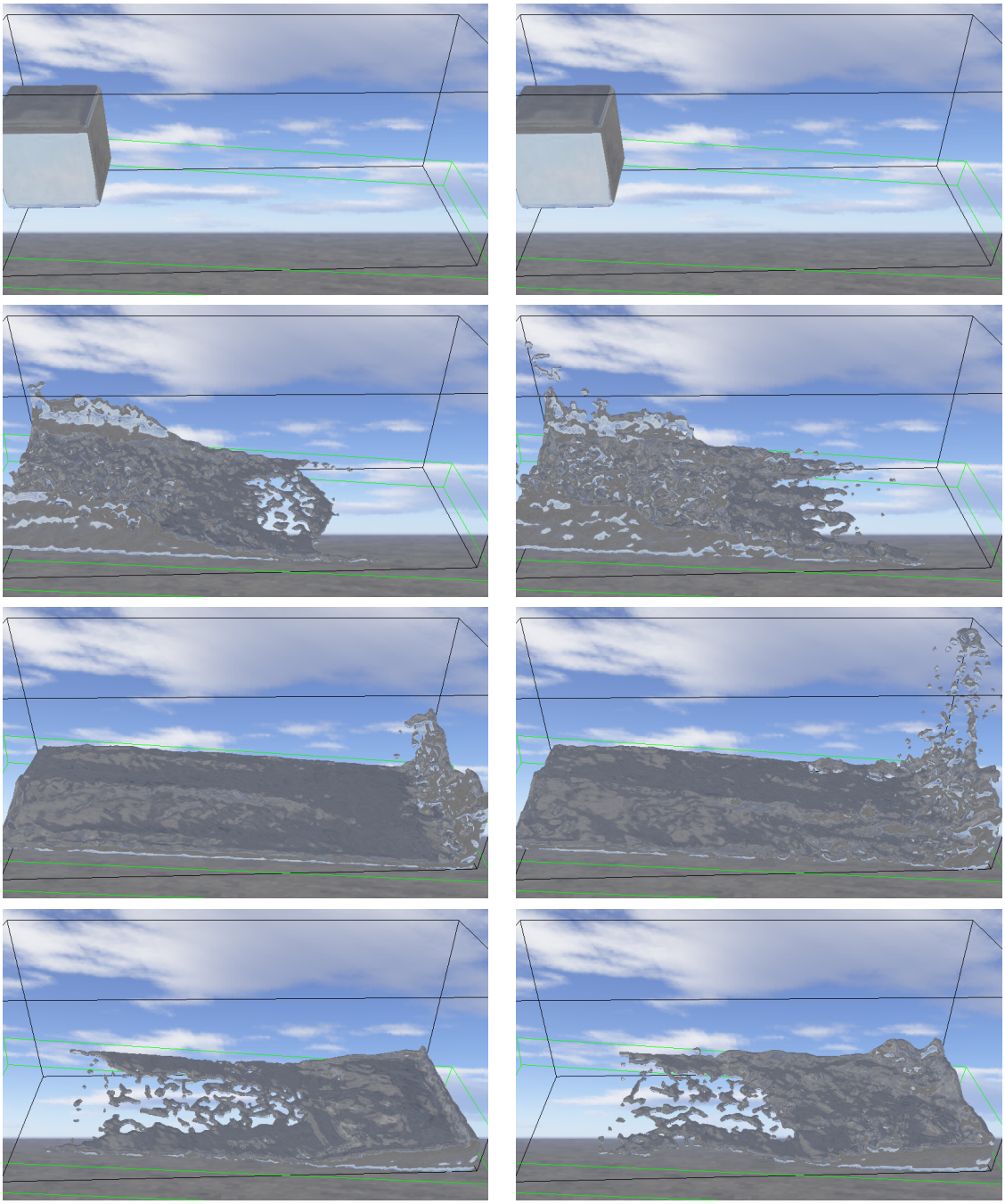

図 4: 3 次元シミュレーション結果 (左列:乱流なし，右列:乱流あり)