情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2010-CG-140 No.7
2010/9/9

粒子法における乱流を考慮した流体表面の改良

三　村　豪[†1]　藤　澤　誠[†1]　天　野　敏　之[†1]
宮　崎　純[†1]　加　藤　博　一[†1]

乱流まで考慮した現実的な液体アニメーションを高速に生成することは非常に困難である．そこで，本論文では，粒子法においてパーティクルをサブパーティクルに分割することによりシミュレーションでは捉えきれない速度場，粒子配置を再現し，乱流を含む流れの流体表面を生成する手法を提案する．また，その実装を行い，提案手法の有効性を検証する．

A Improvement of Fluid Surface using SPH for Turbulent Flow

Go Mimura,[†1] Makoto Fujisawa ,[†1]
Toshiyuki Amano ,[†1] Jun Miyazaki [†1]
and Hirokazu Kato[†1]

It is very difficult to make an animation of liquid with turbulence. In this paper, we propose a method to generate fluid surface including turbulent flow by dividing fluid particles into sub-particles and updating positions and velocity field that usual particle method can not capture. We implement this method and verify it.

1. はじめに

現在，流体シミュレーションは科学計算だけでなく，映画，ゲームなどのエンターテイメント分野などでも盛んに用いられるようになってきた．流れは層流と乱流の二種類に分けることができる．これまで CG の分野で研究されてきたのは層流がほとんどである。層流は乱れがほとんどなく，規則正しく運動する流れである．乱流は乱れが多く，不規則な流れであり，自然現象の流れのほとんどは乱流である．そのため，流体の CG アニメーションを生成するために，乱流を表現するということは重要なテーマである．

最も多く用いられる流体シミュレーション手法であるグリッド法で乱流を再現するためには，グリッドを細かくしなければならず，計算コストは非常に高くなる．これは荒いグリッドでは数値拡散により乱流を構成する渦が消失してしまうからである．この高計算コストにより，インタラクティブなアプリケーションで必要とされるリアルタイムシミュレーションにおいて乱流を再現することは困難である．

そのため，近年，グリッド法において，シミュレーションのみでは捉えきれない，もしくは消えてしまう乱れを別のフレームワークで再現する研究が盛んに行われている．それらによって，計算コストを抑えながら，本物らしい乱流を表現することが可能となった．しかし，それらはリアルタイムでのシミュレーションできるものではなく，またグリッド法以外で同様の研究をした例はほとんどない．

そこで，本論文では，通常の粒子法による流体シミュレーションを行い，そのシミュレーションでの各パーティクルをサブパーティクルに分割することにより，シミュレーションだけでは再現できない乱れを含んだパーティクルの動き，表面を生成する手法を提案する．また，GPU を用いて実装を行うことで高速化する．

2. 関連研究

Stam[12] 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，CIP 法[5] などの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[13] 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[2] 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

†1 奈良先端科学技術大学院大学
Nara Institute of Science and Technology
(630-0192 奈良県生駒市高山町　8916-5)

レーションに乱流を付加することも可能である．Selle ら[10] 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[9] 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[6] 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．

また，パーティクル法においては，Adams ら[1] はシミュレーションにおいてタイムステップごとに一定の条件で粒子を分割，合成することにより粒子の大きさを自由に変更できるようにした．これにより，表面付近，視点などの条件を考慮してより本物らしい流体アニメーションを生成した．また，Zhang ら[14] はそれを GPU で実装した．Solenthar ら[11] は粒子の位置から粒子をアップサンプリングすることで，精巧な表現を悦手法を提案した．ただし，この手法は流体を考慮したものではなく，幾何学的情報のみから粒子をアップサンプリングする手法である．

我々は Kim ら[6] の方法を粒子法に応用し，さらにサブパーティクルを使うことで，追加の計算コストを抑えつつ，よりリアルな液体表面を得られる手法を提案する．また，この手法はリアルタイムでの計算にも適応可能なものである．

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) 各パーティクルに属するサブパーティクルの位置，回転を更新
(4) パーティクルの位置と速度を更新

一般的な SPH 法による計算を行い，その速度場をウェーブレット解析することによりエネルギースペクトル分布を得る．そのエネルギースペクトルの値をもとに，各パーティクルに属する複数のサブパーティクルを移動させる．通常の粒子法で得られた各粒子の位置からの相対的な位置を更新するだけで，サブパーティクルに関する計算は SPH 法の計算には影響を与えることはない．(1),(2) については藤澤らの研究[15] と同様である．以下に，各ステップでの計算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．

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

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

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[8]．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．

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，

$$\hat{e}(k) = \frac{1}{2}|\hat{\boldsymbol{u}}(k)|^2 \tag{3}$$

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，ウェーブレット変換を用いて，実際には，$\hat{e}(k, \boldsymbol{x})$ となる．

我々はパーティクル法を用いているためウェーブレット変換を直接適用することはできない．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．

$$\hat{u}(s, a, b, c) = \frac{1}{\sqrt{s}} \iiint_{-\infty}^{\infty} u(\boldsymbol{x}) \psi\left(\frac{x-a}{s}, \frac{y-b}{s}, \frac{z-c}{s}\right) dxdydz \tag{4}$$

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，

$$\hat{u}_i = \frac{1}{\sqrt{s}\psi_{sum}} \sum_j u_j \psi \left(\frac{x_i - x_j}{s}, \frac{y_i - y_j}{s}, \frac{z_i - z_j}{s} \right). \tag{5}$$

ここで ψ_{sum} は，

$$\psi_{sum} = \sum_j \psi \tag{6}$$

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} で正規化する．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 (3) 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．

3.3 サブパーティクルへの分割

パーティクルより小さなスケールの渦を表現するために本研究ではサブパーティクルを用いる．パーティクルは二つのサブパーティクルに分割できるものとする．また，サブパーティクルについてもさらに二つのサブパーティクルに再帰的に分割できるものとする．つまり，SPH 法での各パーティクルを設定したレベルまでそれぞれ 2^j 個に分割する．このとき，分割していない元のパーティクルをレベル 0，それを 2 つに分割したものをレベル 1，さらに 2 つずつに分割したものをレベル 2 とする．レベル L まで分割すると，1 つのパーティクルが 2^L 個のサブパーティクルへと分割されることとなる．ここで，分割前のサブパーティクルを親，分割後の二つのサブパーティクルを子と呼ぶ．あるサブパーティクルとそれをさらに 2 つに分割したサブパーティクルの位置関係を図 1 に示す．

レベル L のパーティクルの半径 r_L，質量 m_L は

$$r_L = 2^{-L/3} r_0 \tag{7}$$
$$m_L = 2^{-L} m_0 \tag{8}$$

である．ここで，r_0，m_0 は SPH 法でのパーティクルの半径，質量である．サブパーティクルの質量の総和が元 (レベル 0) のパーティクルの質量となる．

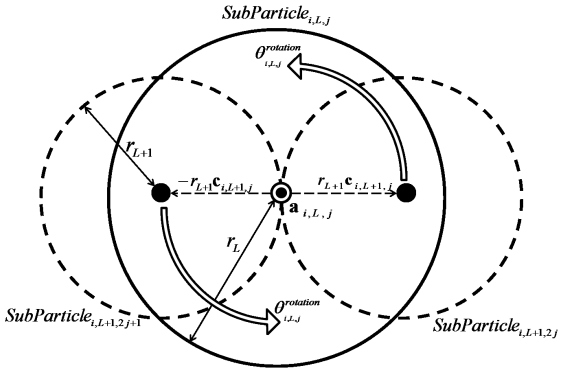

図 1 あるサブパーティクルとそれをさらに 2 つに分割したサブパーティクルの関係

パーティクル i のレベル L のサブパーティクルで j 番目のサブパーティクルを親としたとき，子の位置 $\boldsymbol{x}_{i,L+1,2j}$，$\boldsymbol{x}_{i,L+1,2j+1}$ はそれぞれ

$$\boldsymbol{x}_{i,L+1,2j} = \boldsymbol{x}_{i,L,j} + r_L \boldsymbol{c}_{i,L,j} \tag{9}$$
$$\boldsymbol{x}_{i,L+1,2j+1} = \boldsymbol{x}_{i,L,j} - r_L \boldsymbol{c}_{i,L,j} \tag{10}$$

となる．ここで，$\boldsymbol{c}_{i,L,j}$ はパーティクル i に属するレベル L で j 番目のサブパーティクルからその子への単位ベクトルである．親の位置から子へ位置への相対的なベクトルはそれぞれ子の半径に子方向への単位ベクトルをかけた $\pm(r_L \boldsymbol{c}_{i,L,j})$ となる．

この子への単位ベクトルは Kolmogorov の理論を満たす速度になるよう親の位置を通るベクトル $\boldsymbol{a}_{i,L,j}$ を軸として回転させる．ただし，ベクトル $\boldsymbol{a}_{i,L,j}$，$\boldsymbol{c}_{i,L,j}$ は垂直である．

その角度 $\theta^{rotation}_{i,L,j}$ は以下の式から導かれる．

$$\theta^{rotation}_{i,L,j} = \frac{|\hat{\boldsymbol{u}}(\frac{1}{2r_{L+1}}, \boldsymbol{x}_i)|\Delta t}{r_{L+1}} \tag{11}$$

ベクトル $\mathrm{a}_{i,L,j}$ を中心に角度 $\theta^{rotation}_{i,L,j}$ 回転させる．ただし，三次元の場合，その回転軸は一意に決まらない．そこで，軸であるベクトル $\boldsymbol{a}_{i,L,j}$ を単位ベクトル $\boldsymbol{c}_{i,L,j}$ と垂直の状態を保ったまま，$\boldsymbol{c}_{i,L,j}$ を軸として角度 $\theta^{axis}_{i,L,j}$ 回転させる．

$$\theta^{axis}_{i,L,j} = \alpha\phi\theta^{rotation}_{i,L,j} \tag{12}$$

ϕ は $[-1.0, 1.0]$ の乱数，α は回転軸のぶれの大きさを決定する 0 以上の定数であり，ユーザーが設定する．

以上の計算により，任意に設定したレベルまで粒子を分割することができる．

3.4 サブパーティクルによる表面生成

表面の生成には Marching Cube 法[7] を用いる．本研究では，レベルを考慮したカーネル関数 $W_{sub}(\boldsymbol{x}, L)$ を導入し，以下の関数に関して等値面を生成する．

$$\phi(\boldsymbol{x}) = \sum_{i}^{N} W_{sub}(\boldsymbol{x}_i, L_i) \tag{13}$$

W_{sub} は式 (14) のエネルギースペクトル値に基づき，用いるサブパーティクルを変化させるカーネルである．基準となるエネルギースペクトル値 e_{cri} は Kolmogorov の理論に基づき、以下のようになる．

$$e_{cri} = \hat{e}(\frac{1}{2r_L}, \boldsymbol{x})(2^{-\frac{5}{9}})^{L_i} \tag{14}$$

式 (14) より使用するサブパーティクルレベル L_i を算出する．L_i の値により W_{sub} は以下のように決定される．ここで，分割するレベルの最大値を L_{max} とする．

(1) $L_i \leq 0$ の場合

レベル 0 のサブパーティクルを採用する．

$$W_{sub}(\boldsymbol{x}_i, L_i) = W(\boldsymbol{x}_{i,0,0}, r_0) = W(\boldsymbol{x}_i r_0) \tag{15}$$

(2) $0 < L_i < L_{max}$ の場合

まず床関数を用いて，

$$L_i^{down} = \lfloor L_i \rfloor \tag{16}$$

$$L_i^{up} = \lfloor L_i \rfloor + 1 \tag{17}$$

とする．そして，レベル L_i^{down}，L_i^{up} のサブパーティクルを採用し，Marching Cube 法の陰関数に加えるカラー関数を $L_i^{up} - L_i$,$L_i - L_i^{down}$ という比重とする．

$$\begin{aligned} W_{sub}(\boldsymbol{x}_i, L_i) \quad = \quad & (L_i^{up} - L_i) \sum_{j=0}^{2^{L_i^{down}}} W(\boldsymbol{x}_{i,L_i^{down},j}, r_{L_i^{down}}) \\ & + (L_i - L_i^{down}) \sum_{j=0}^{2^{L_i^{up}}} W(\boldsymbol{x}_{i,L_i^{up},j}, r_{L_i^{up}}) \end{aligned} \tag{18}$$

(3) $L_i \geq L_{max}$ の場合

レベル L_{max} のサブパーティクルを採用する．

$$W_{sub}(\boldsymbol{x}_i, L_i) = \sum_{j=0}^{2^{L_{max}}} W(\boldsymbol{x}_{i,L_{max},j}, r_{L_{max}}) \tag{19}$$

これにより，エネルギースペクトルが大きい，つまり乱れが発生しやすい場所では細かい乱れを再現できる．一方，エネルギースペクトルが小さい，つまり乱れが発生しにくい場所では細かい乱れは現れにくくなる．ここで求めた陰関数に対して，Marching Cube 法を用いることで流体表面を生成する．

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 i7 2.93GHz, GPU:GeForce GTX285 である．SPH 法とサブパーティクルの移動については NVIDIA CUDA を用いて GPU 上で実装されている．メッシュ生成は CPU 上で実装した．また，ウェーブレット関数には Mexican Hat を用いた．

図 2，図 3 はそれぞれサブパーティクルなし，ありの三次元シミュレーションの結果である．立方体形状の液体を穏やかな下り坂に落下させたものである．パーティクル数は 50000, サブパーティクルの分割最大レベル L_{max} は 4，ウェーブレットスケールはパーティクル半径の 2 倍とした．1 ステップ当たりの計算時間は 26 ミリ秒/フレームであり，その内訳は SPH 法 12 ミリ秒/フレーム，エネルギースペクトルの算出 6 ミリ秒/フレーム，サブパーティクルの位置計算 8 ミリ秒/フレームであった．ただし，ここにはメッシュ生成に関する計算時間は含まれない．どちらも上から 1 段目のような乱れの発生しにくい状態では，乱流が見られない．一方，2 段目や 3 段目の下流 (右) 側のような乱れが発生しやすい状態では，サブパーティクルなしでは乱れがほとんど見られないが，提案手法を適用することにより本物らしい乱れが発生している．

5. おわりに

本論文では，粒子法における乱流を考慮したサブパーティクルへの分割とそれを用いた表面の生成手法を提案した．それによって，高速に乱流の液体アニメーションを生成した．しかし，SPH 法に関する時間に対してサブパーティクルに関する計算時間が同じ程度かかってしまっている．これは、液体内部まで分割処理を行っているためであり、表面だけに限定することでさらなる高速化が可能である。また，本手法は並列計算に適しており，リアルタイムな流体アニメーション生成に十分適応可能である．そして，エネルギースペクトルを用

Vol.2010-CG-140 No.7
2010/9/9

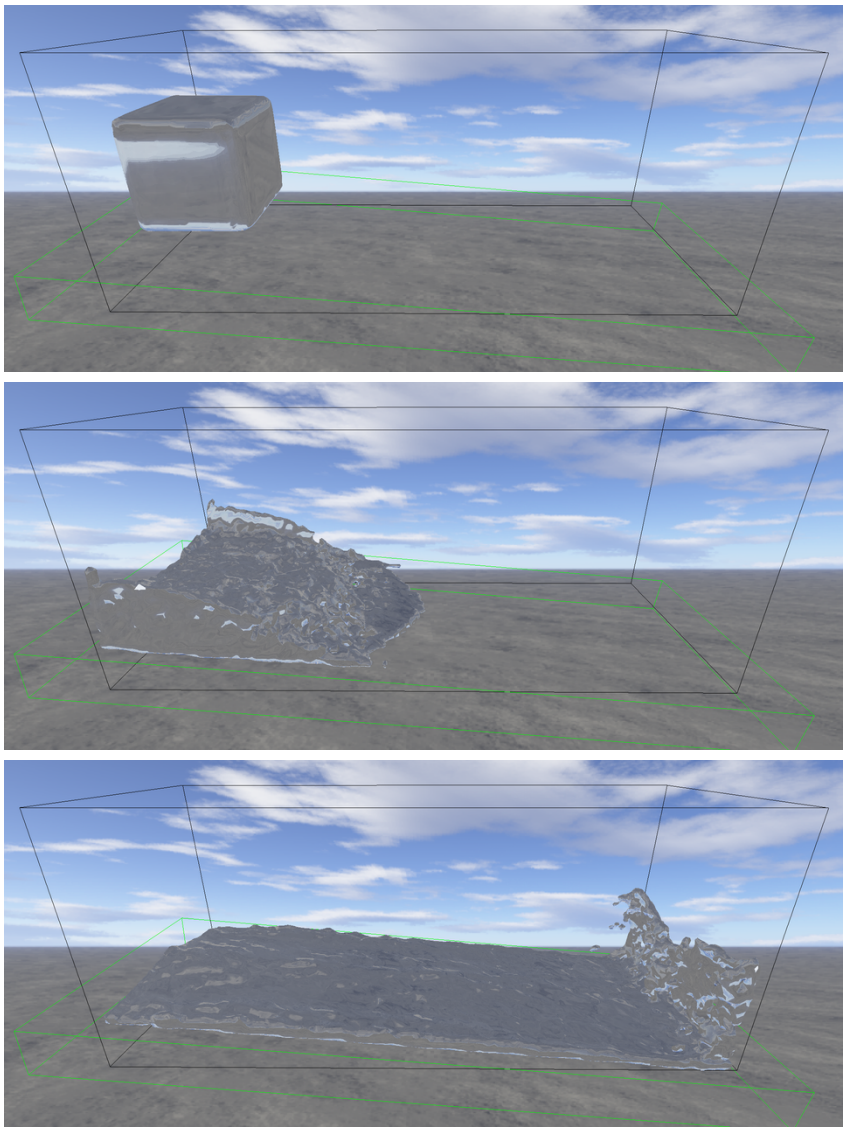

図 2 3 次元シミュレーション結果 (サブパーティクルなし)

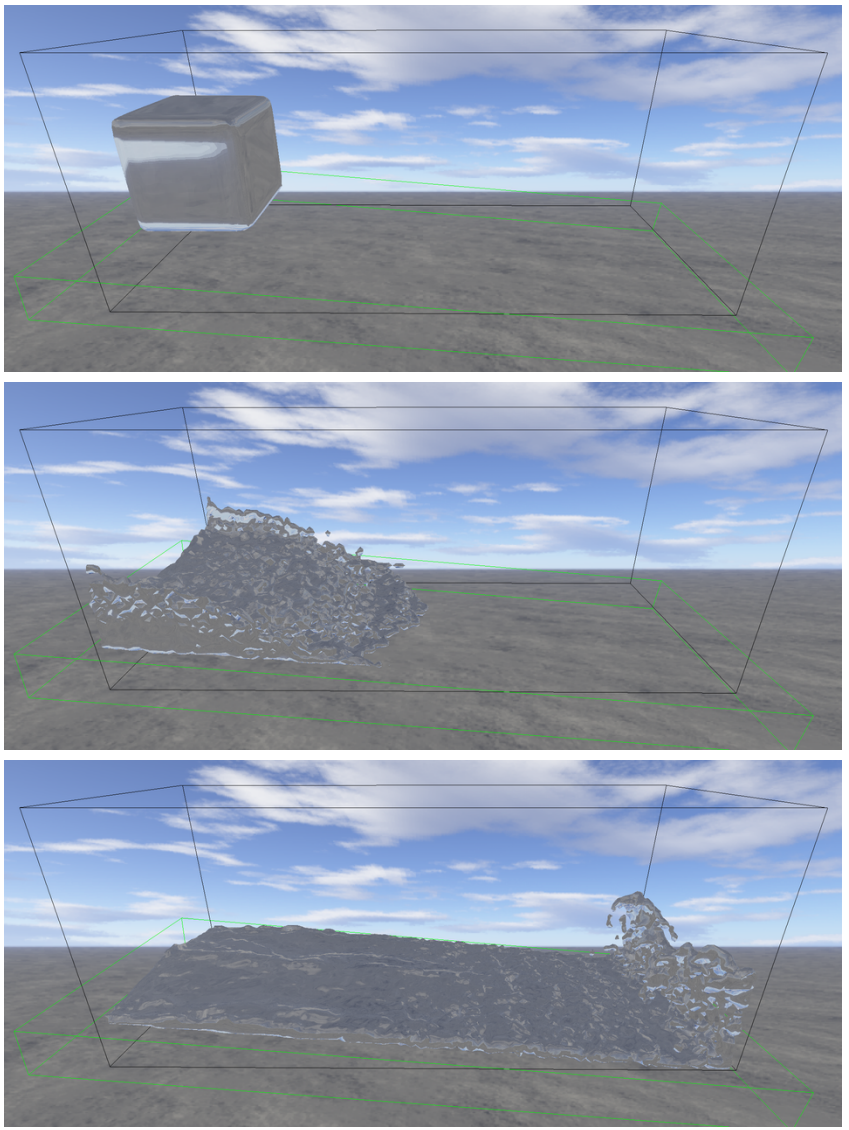

図 3 3 次元シミュレーション結果 (サブパーティクルあり)

いた表面生成により，乱れの発生しやすい場においてのみ本物らしい乱流を加えることができた．

今後の課題としては，サブパーティクルの移動のモデルの改良が挙げられる．本論文で提案したサブパーティクルの動きはエネルギースペクトルのみに基づいたものであり、表面付近では表面張力などの影響などもあるためそれらを考慮する必要がある．例えば，液体が薄い膜となるような場合には，提案手法は有効とは言えない．また、表面生成まで含めて，GPU 実装することで，リアルタイムで乱流を考慮したアニメーションを生成したい．

参 考 文 献

1) Adams, B., Pauly, M., Keiser, R. and Guibas, L.J.: Adaptively sampled particle fluids, *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, New York, NY, USA, ACM Press, p.48 (2007).
2) Bridson, R., Houriham, J. and Nordenstam, M.: Curl-noise for procedural fluid flow, *Proc. SIGGRAPH 2007*, p.46 (2007).
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, 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).
6) Kim, T., Thürey, N., James, D. and Gross, M.: Wavelet turbulence for fluid simulation, *Proc. SIGGRAPH 2008*, pp.1–6 (2008).
7) Lorensen, W.E. and Cline, H.E.: Marching cubes: a high resolution 3D surface construction algorithm, *Computer Graphics (SIGGRAPH '87 Proceedings)*, Vol.21, No.4, pp.163–169 (1987).
8) Müller, M., Charypar, D. and Gross, M.: Particle-Based Fluid Simulation for Interactive Applications, *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation*, pp.154–159 (2003).
9) 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).
10) Selle, A., Rasmussen, N. and Fedkiw, R.: A vortex particle method for smoke, water and explosions, *Proc. SIGGRAPH 2005*, pp.910–914 (2005).
11) Solenthaler, B., Zhang, Y. and Pajarola, R.: Efficient Refinement of Dynamic Point Data, *Proceedings Eurographics/IEEE VGTC Symposium on Point-Based Graphics* (2007).
12) Stam, J.: Stable fluids, *Proc. SIGGRAPH 1999*, pp.121–128 (1999).
13) Stam, J. and Fiume, E.: Turbulent wind fields for gaseous phenomena, *Proc. SIGGRAPH1993*, pp.369–376 (1993).
14) Zhang, Y., Solenthaler, B. and Pajarola, R.: Adaptive Sampling and Rendering of Fluids on the GPU, *IEEE / EG Symposium on Point-Based Graphics* (2008).
15) 藤澤　誠，加藤博一：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，グラフィックスと CAD/Visual Computing 合同シンポジウム 2010 予稿集 (2010).