情報処理学会論文誌 Vol.53 No.7 1915–1923 (July 2012)

粒子法とウェーブレットを用いたサブパーティクルスケール乱流の高速なシミュレーション

藤澤 誠[1,a)] 三村 豪[2] 天野 敏之[3] 宮崎 純[4] 加藤 博一[4]

受付日 2011年9月28日, 採録日 2012年4月2日

概要：本論文ではパーティクル法とウェーブレット解析を用いた高速な乱流シミュレーション手法を提案する．高速に流体の流れを再現するためにパーティクル法の一種であるSPHを用い，そのパーティクル間の相互関係から直接ウェーブレット解析により乱流が発生する領域を検出する方法を提案する．検出された領域にウェーブレットノイズに基づき渦場を付加することで乱流を再現する．さらに，乱流エネルギーに基づきパーティクルを分割・回転させることで，サブパーティクルスケールでの乱流を実現する．最後にこれらの処理のほとんどはGPU上で容易に実装可能であり，リアルタイムに近い速度で実行できることを示す．

キーワード：流体シミュレーション，SPH，乱流，ウェーブレット

A Fast Simulation Method Using SPH and Wavelet for Sub-particle-scale Turbulent Flow

Makoto Fujisawa[1,a)] Go Mimura[2] Toshiyuki Amano[3]
Jun Miyazaki[4] Hirokazu Kato[4]

Received: September 28, 2011, Accepted: April 2, 2012

Abstract: This paper presents a fast simulation method for turbulent flow which uses a particle method and wavelet analysis. To simulate fluid flow, the method uses smoothed particle hydrodynamics (SPH), which discretizes the fluid into a collection of particles, and detects regions where turbulent flow will occur by using wavelet analysis without a spatial grid. By taking the curl of wavelet noise, the turbulent flow is then appended as a divergence-free turbulence velocity field. Additionally, by using a particle splitting, which characterize the vortex features of turbulence, a sub-particle-scale representation of turbulent flow is proposed. Implementing almost all processes on a graphics processing unit (GPU), simulations are performed in near real time.

Keywords: fluid simulation, SPH, turbulence, wavelet

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

[1] 筑波大学大学院図書館情報メディア系
Faculty of Library, Information and Media Science, University of Tsukuba, Tsukuba, Ibaraki 305–8550, Japan
[2] 任天堂株式会社
Nintendo Co., Ltd., Kyoto 601–8501, Japan
[3] 山形大学工学部システム創成工学科
Department of Systems Innovation Engineering, Yamagata University, Yonezawa, Yamagata 992–8510, Japan
[4] 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma, Nara 630–0192, Japan
[a)] fujis@slis.tsukuba.ac.jp

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．たとえば，水道の蛇口から流れる水は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 1 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うる．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（たとえば，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さかんに行われている．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．近年の GPU の発展により，高速な流体シミュレーションをゲームやインタラクティブアプリケーションに応用する例が増えており，より現実感の高い乱流を含むアニメーションをリアルタイムに生成することはとても重要である．

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など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る乱流速度場を追加する．さらに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，粒子法による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．具体的には粒子法を用いた CG アニメーションで一般的な数万程度の粒子でもリアルタイムで実行することを目指す．この手法は個々のパーティクルまで問題を分割できるため GPU での並列処理に適したものであり，高速に実行できる．

本論文の次章以降の構成は以下である．次章で関連研究を述べ，3 章でシミュレーション手法，そして，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 5 章で述べる．

2. 関連研究

Stam [22] 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，乱流場をとらえるのを難しくする．この数値拡散を抑えるために，BFFCC [8] や CIP 法 [9]，QUICK [12] といった高次の補間法を移流に用いる手法が導入されている．また，Fedkiw ら [5] 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 [7] はこれを SPH 法へ導入した．

Selle ら [20] は，乱流が起こるであろう場所に渦を表現するパーティクルを散布し，渦方程式に従って移流させることで，グリッド上では数値拡散により失われた乱流を再現する渦パーティクル法を提案した．しかし，渦パーティクル法ではユーザが散布場所を直接指定する必要がある．これを解決するために，Pfaff ら [18] 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．Zhu ら [26] は WCSPH [13] と固体周囲に配置したグリッド構造を用いることで，壁面境界層で発生する乱れを粒子法で再現した．

一方，乱流モデルを用いて乱れを構成する渦を手続き的に生成する手法として，Stam ら [23] は Kolmogorov 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．Zhao ら [25] は random forcing により，Chang ら [3] は粗いグリッドと細かいグリッドを組み合わせることで，シミュレーションに細かな乱れを追加した．一方，乱れが発生する場所を検出するために，乱流エネルギーの発生と散逸をモデル化した k-ε モデルがいくつかの研究で用いられている [16], [17], [19]．流れそのものから直接乱流発生領域を特定するために，Kim ら [10] はウェーブレット解析を用いた．粗いグリッドでの流体シミュレーションに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．しかし，これらのグリッドを使った手法は 1 フレームに数秒から数分の計算時間を要し，リアルタイムでの生成には適さない．我々は Kim らの方法をパーティクル法に応用し，さらにパーティクルをサブパーティクルへと分割することで，高速に計算でき，かつ，パーティクルスケール以下の小さな乱れも再現できる手法を提案する．

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は以下のようになる．

(1) SPH により粘性項，圧力項，外力項を計算し，速度，位置を更新（3.1 節）
(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（3.2 節）
(3) 乱流応力を計算し，パーティクルの位置と速度を更新（3.3 節）
(4) 各パーティクルに属するサブパーティクルの位置と速度を更新（3.4 節）
(5) Marching Cube 法により表面生成（3.5 節）

この章では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 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，t は時間，$\boldsymbol{u}$ は流体速度，ν は動粘性係数，ρ は流体の密度，p は圧力，$\boldsymbol{f}$ は外力である．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つねに保持され（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 [14] 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．Harada ら [6] は近傍粒子探索を GPU 上で実装する手法を提案した．本研究では文献 [6] の手法を NVIDIA CUDA を用いて GPU 上で実装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ためには，シミュレーション上で再現できている大スケールの渦からエネルギー遷移により小スケール渦が生成される過程を再現しなければならない．そのために，まず，3.1 節で計算された速度場から渦のエネルギーを算出する．

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，

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

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，$\hat{e}(k, \boldsymbol{x})$ となっている．

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．文献 [10] 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．1 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うえに，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．

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

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，

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

ここで ψ_{sum} は，

$$\psi_{sum} = \sum_j \psi \tag{9}$$

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 $\hat{e}_i(k)$ 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．式 (8) 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．たとえば，図 1 (c)，(d) では液体表面 1 列分のパーティクルのエネルギーが小さくなっている．パーティクルの探索半径は描画している円の半径の 2〜3 倍であるので，表面 1 列のパーティクルのみパーティクル密度が低くなり，その影響が現れている．これはパーティクル密度の均一化 [21] 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．

3.3 ウェーブレット乱流速度場

ウェーブレット解析により算出した乱流発生領域に非圧縮乱流速度場を追加することで，数値拡散で失われた渦を再現する．非圧縮乱流速度場は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でのエネルギーを Kolmogorov の理論に基づき計算することで求められる．文献 [2] 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．

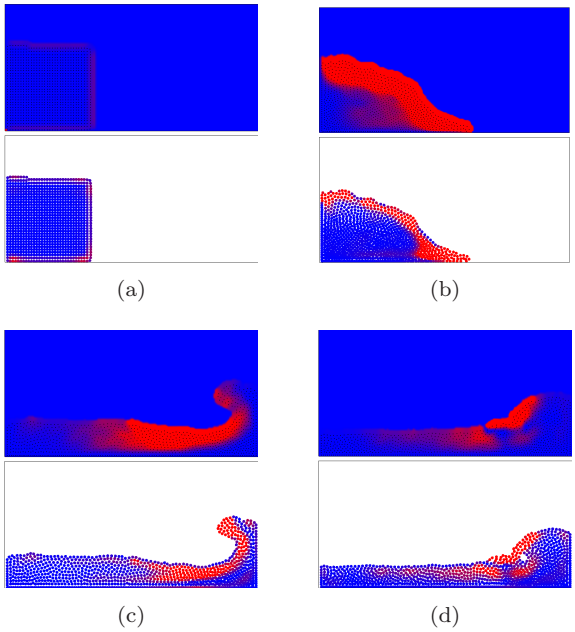


図 1 グリッド（各図の上段）とパーティクル（各図の下段）から算出されたエネルギースペクトル値の比較．値が高いグリッド，パーティクルは赤，低いものは青で描画している

Fig. 1 Energy spectrum calculated using a grid (top) and particles (bottom). High and low energies are shown in red and blue, respectively.

$$\boldsymbol{w}(\boldsymbol{x}) = \left(\frac{\partial \omega_3}{\partial y} - \frac{\partial \omega_2}{\partial z}, \frac{\partial \omega_1}{\partial z} - \frac{\partial \omega_3}{\partial x}, \frac{\partial \omega_2}{\partial x} - \frac{\partial \omega_1}{\partial y} \right) \tag{10}$$

ノイズ関数 ω にはウェーブレットノイズ [4] を用いる．式 (10) は 3 次元のノイズタイルから $\boldsymbol{x}$ におけるノイズ値とオフセットした位置での値（ω_1，ω_2，ω_3）を流速ベクトルとした渦度ベクトルとなっている．

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する．

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（backward scattering）．この乱流エネルギー遷移を表現するために Kolmogorov スペクトルを用いる．Kolmogorov 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．

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

ここで，α は Kolmogorov 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 [10]．

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

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，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，

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

である．$[b_{min}, b_{max}]$ 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，b_{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，b_{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) と前節の式 (6) で計算した $\hat{e}_i(k)$ を用いることで，乱流により各パーティクルにかかる力は，

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

となる．ここで，添え字 i はパーティクルインデックスを表し，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．

$\boldsymbol{f}^{turb}$ は SPH 法へ外力として追加され，パーティクルの位置と速度に影響する．このとき生成される乱流のスケールは SPH パーティクルで表現できるものに限られるため，パーティクルスケール乱流と呼ぶこととする．また，これよりさらに細かな乱流をサブパーティクルスケール乱流と呼ぶ．

3.4 サブパーティクル分割

乱流は様々な大きさの渦の集合と考えることができる．そのため，乱流を完全に表現したいならば，すべての大きさの渦を再現しなければならない．しかし，SPH において再現できる渦の大きさはパーティクルの大きさに依存し，小さな渦をシミュレートするためにはより小さいパーティクルが必要となる．これはシミュレーション全体のパーティクル数を増大させ，同時に計算時間を爆発的に増大させる結果となる．van der Laan ら [24] はスクリーンスペースメッシュに対してノイズ関数を適用することで，パーティクル以外のフレームワークを用いたサブパーティクルスケールでの乱れの表現を可能とする手法を提案した．Adams ら [1] は液体形状に応じてパーティクルを分割，結合させる手法を提案している．3.1 節で述べたように乱流では異なるスケールの渦間でのエネルギー遷移が発生する．

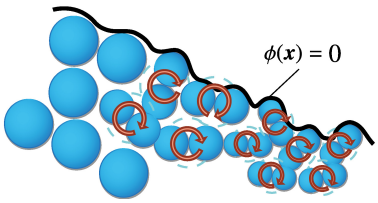


図 2 パーティクルからサブパーティクルへの分割
Fig. 2 Sub-particle splitting.

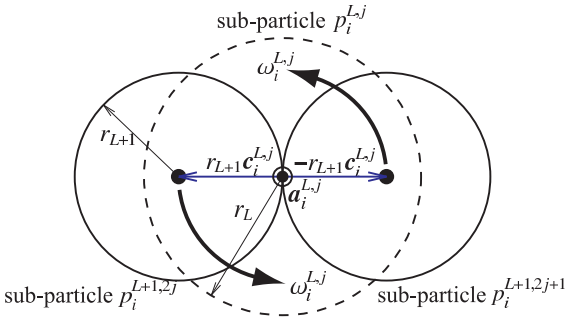


図 3 サブパーティクル p^L と p^{L+1} の位置関係
Fig. 3 Relationship between the particle p^L and p^{L+1}.

我々はこのエネルギー遷移の中でも forward scattering に注目し，ウェーブレット解析で算出したパーティクルスケールでの乱流エネルギーに基づき，SPH のパーティクルを分割し，それを元のパーティクルを中心として回転させることでより細かな渦を再現する．この分割したパーティクルをサブパーティクルと呼ぶ．渦パーティクル法 [20] と似たものであるが，サブパーティクルは単に元のパーティクルの周りを回転するだけであり，計算コストは非常に小さい．また，サブパーティクルはレンダリング時のみに用い，SPH によるシミュレーションには影響しない．

図 2 にパーティクル分割の概要を示す．式 (12) に基づき乱流エネルギー遷移を計算するために，SPH のパーティクル p_i^0 は 2 つのサブパーティクル $p_i^{1,j}$，$j = 0, 1$ に分割できるものとする．また，サブパーティクル p_i^L についてもさらに 2 つのサブパーティクル p_i^{L+1} に再帰的に分割できるものとする．ここで，分割前のサブパーティクルを親，分割後の 2 つのサブパーティクルを子と呼ぶ．これにより，乱流理論における forward scattering を表現する．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．このように分割することで，他のパーティクルと干渉することなしにサブパーティクルを生成できるため，並列処理に適している．

あるサブパーティクル p^L とそれをさらに 2 つに分割したサブパーティクル p^{L+1} の位置関係を**図 3** に示す．$\boldsymbol{c}_i^{L,j}$ はパーティクル $p_i^{L,j}$ からその子 $p_i^{L+1,2j}$ への単位ベクトルである（$j = 0, 1, \ldots, 2^L - 1$）．$p_i^L$ に対する p_i^{L+1} の相対的な位置は $\pm(r_L\boldsymbol{c}_i^{L,j})$ となる．p_i^{L+1} の半径 r_L と質量 m_L は

$$r_L = 2^{-L/3}r_0, \quad m_L = 2^{-L}m_0 \tag{15}$$

である．ここで，r_0，m_0 は SPH 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．

子への単位ベクトル $\boldsymbol{c}_i^{L,j}$ は，各サブパーティクルの回転速度が Kolmogorov の理論を満たすよう親の位置を通るベクトル $\boldsymbol{a}_i^{L,j}$ ($\perp \boldsymbol{c}_i^{L,j}$) を軸として回転させる．角速度 ω は，

$$\omega_i^{L,j} = \frac{|\hat{\boldsymbol{u}}(\frac{1}{2r_{L+1}}, \boldsymbol{x}_i)|\Delta t}{r_{L+1}}. \tag{16}$$

ここで，

$$\begin{aligned} |\hat{\boldsymbol{u}}\left(\frac{1}{2r_{L+1}}, \boldsymbol{x}_i\right)| &= \sqrt{2\hat{e}\left(\frac{1}{2r_{L+1}}, \boldsymbol{x}_i\right)} \\ &= \sqrt{2\hat{e}\left(\frac{1}{2r_0}, \boldsymbol{x}_i\right)(2^{-\frac{5}{9}})^{L_i}} \end{aligned} \tag{17}$$

ウェーブレット解析で求めた各パーティクルの乱流エネルギーから，そのサブパーティクルのスケールでの乱流エネルギーを求めることで，サブパーティクルでの速度を決定する．ベクトル $\boldsymbol{a}_i^{L,j}$ を軸として角速度 $\omega_i^{L,j}$ で回転させる．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，$\boldsymbol{c}_i^{L,j}$ を軸として毎ステップ角度 $\theta_i^{L,j}$ 回転させる．

$$\theta_i^{L,j} = \alpha\phi\omega_i^{L,j} \tag{18}$$

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パーティクルを分割することができる．並列計算における処理の分岐を減らすために，サブパーティクルの回転はすべてのレベルで行われる．そして，レンダリングにおいて，乱流エネルギーに基づいて使用するサブパーティクルのレベルを選択することで，エネルギー分布による渦のスケール変化を表現する．

3.5 サブパーティクルを考慮した表面生成

表面の生成には Marching Cube 法 [11] 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．

$$\phi(\boldsymbol{x}) = \sum_i^N W_{sub}(\boldsymbol{x}_i, L_i) \tag{19}$$

W_{sub} はエネルギースペクトル値に基づき，用いるパーティクルのレベ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} を定数として設定し，以下の式を満たすようなレベル L_i を求める．

$$e_{cri} = \hat{e}\left(\frac{1}{2r_L}, \boldsymbol{x}\right)(2^{-\frac{5}{9}})^{L_i} \tag{20}$$

ここで，$\hat{e}(s, \boldsymbol{x})$ は位置 $\boldsymbol{x}$ でのスケール s でのエネルギースペクトルである．この式は，Kolmogorov の理論に従い，レベル 0 のパーティクルからどのレベルのサブパーティクルまでエネルギー遷移（forward scattering）すれば，基準のエネルギースペクトルとなるかを計算する．式 (20) より使用するレベル 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．

$$W_{sub}(\boldsymbol{x}_i, L_i) = \begin{cases} W(\boldsymbol{x}_i, r_0) \\ \qquad L_i \le 0 \\ \sum_{j=0}^{2^{L_{max}}} W(\boldsymbol{x}_i^{L_{max},j}, r_{L_{max}}) \\ \qquad L_i \ge L_{max} \\ (L_i^{up} - L_i)\sum_{j=0}^{2^{L_i^{down}}} W(\boldsymbol{x}_i^{L_i^{down},j}, r_{L_i^{down}}) \\ \quad +(L_i - L_i^{down})\sum_{j=0}^{2^{L_i^{up}}} W(\boldsymbol{x}_i^{L_i^{up},j}, r_{L_i^{up}}) \\ \qquad \text{otherwise} \end{cases} \tag{21}$$

ここで，$L_i^{down} = \lfloor L_i \rfloor$，$L_i^{up} = \lfloor L_i \rfloor + 1$ である．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大きなレベルまで分割され細かい乱れを再現し，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分割のレベルは小さく，細かい乱れは現れにくくなる．陰関数 ϕ に対して，Marching Cube 法 [11] で流体表面を生成する．

4. 結果

提案手法を実装した結果を示す．実行環境は，CPU: Core i7 2.93 GHz，GPU: GeForce GTX580 である．手法のほとんどの部分は NVIDIA CUDA を用いて GPU 上で実装した．また，ウェーブレット関数には一般的に用いられる Mexican Hat を用いた．

図 4 は谷の中における流れをシミュレーションした結果である．パーティクル数は最大で 40,000，サブパーティクル最大分割レベル L_{max} は 3 とした．図 4 (a) は SPH のみ，(b) はパーティクルスケール乱流，(c) はサブパーティクルスケール乱流を含めた結果である．(c) では (b) に比べて流れが激しいところでより細かな乱れが発生している．乱流を含む流体シミュレーションにかかった計算時間は約 15 ミリ秒/ステップである．また，Marching Cube 法によるメッシュ生成（最大メッシュ数は約 80,000）には，(a)，(b) について 15 ミリ秒/ステップ，サブパーティクルスケール乱流を含めた (c) については 120 ミリ秒/ステップかかった．(b) と (c) の間のシミュレーション時間にはほとんど変化がなく，計算時間の主な違いはメッシュ化のための陰関数 ϕ の計算に現れた．これは，サブパーティクルスケール乱流でのメッシュ化の際にはサブパーティクルを含んでおり，$L_{max} = 3$ の場合，最大でパーティクル数は 8 倍となるためである．

(a) SPH のみ　(b) パーティクルスケール乱流　(c) サブパーティクルスケール乱流

図 4 谷の中における流れ

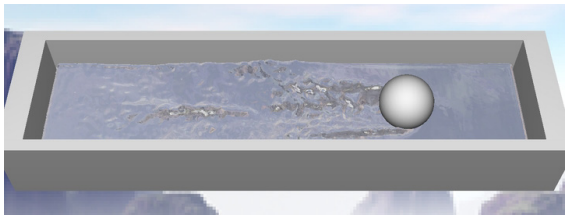
Fig. 4 Flooding in a valley.

サブパーティクルの効果を検証するために，より単純な移動物体の後にできる後流をシミュレーションした結果を**図 5** に示す．図 5 (a) は SPH のみ，(b) はパーティクル乱流，(c) はサブパーティクルスケール乱流，(d) は (b) のパーティクル数を 8 倍にした結果である．(a)，(b)，(c) におけるパーティクル数は約 23,000，(d) は約 184,000 である．サブパーティクル最大分割レベル L_{max} は 3 とし，(c) と (d) のレンダリングにおけるパーティクル数がほぼ同じとなるようにした．提案手法はパーティクル数を増やした場合と同じ程度の細かさの渦を再現できている．(c) の計算時間は約 75 ミリ秒/ステップ，(d) は約 260 ミリ秒/ステップであった．ただし，(d) ではパーティクルを小さくした影響により計算が不安定になるため，時間ステップ幅を他のものの 1/2 にする必要があり，結果として提案手法の計算時間は約 1/7 ですんでいる．しかし，(d) と比べて小さな渦が物体が離れた後も残っており，不自然な結果となった．これは，サブパーティクルにより forward scattering は再現されたものの，その逆のエネルギー遷移である backward scattering が考慮されていないことが原因と考えられる．パーティクルスケール乱流では数値拡散によりこれらの現象が再現されていたのに対して，サブパーティクルでは原理上数値拡散が発生しない．また，粘性による渦エネルギーの拡散も考慮されていない．図 5 のような乱流が発生してすぐに収まるシーンではこれは問題であり，エネルギー遷移において forward scattering を取り入れるなどの対策が必要である．一方，図 4 のような乱流が連続して発生するシーンでは大きな問題とはならない．

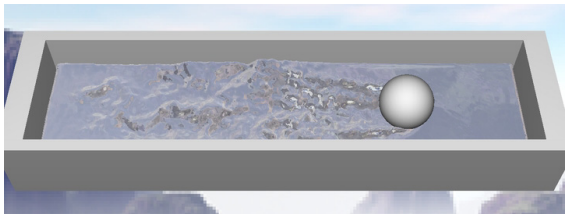
5. まとめと今後の課題

本論文では，SPH 法とウェーブレット解析により乱流を再現する高速な手法を提案した．我々は近傍パーティクルの重み付き合計をとることでグリッド構造を必要とせずにパーティクルから直接ウェーブレット分解を行い，ウェーブレット乱流と Kolmogorov の理論に基づき数値拡散で失われたパーティクルスケールの乱流渦を合成し，外力として追加する方法を提案した．さらに，サブパーティクル分割により計算速度を抑えつつ，パーティクルスケール以下の乱れを再現可能な方法も提案した．そして，これらを GPU 上で実行することによりリアルタイムに近い高速な処理を実現した．

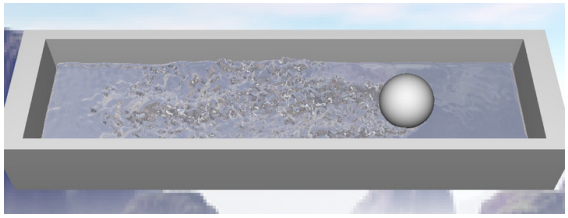
しかしながら，発生した渦の生存時間が長く，流れが不自然に見える現象が観測された．これを解決するために，



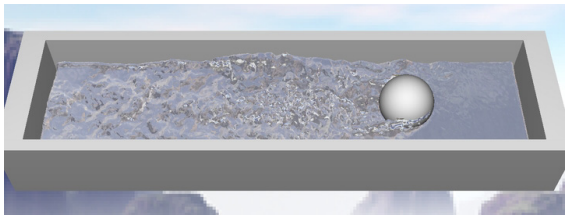
(a) SPH のみ



(b) パーティクルスケール乱流



(c) サブパーティクルスケール乱流



(d) パーティクルスケール乱流（パーティクル数 8 倍）

図 5 移動固体の後流

Fig. 5 Trailing vortex left by a moving ball.

大きな渦へのエネルギー遷移（backward scattering）および，粘性拡散を考慮する必要がある．さらに，液体中に発生する気泡や泡，しぶきを乱流エネルギーに基づき発生させることでよりリアリティを向上させることができるだろう．また，サブパーティクルを使ったシミュレーションそのものの計算コストは小さいものの，そのレンダリングに大きな計算時間のオーバヘッドが生じている．これをより効率的なアルゴリズム（たとえば，スクリーンスペースメッシュ [15]）を用いることで高速化することも今後の課題である．

参考文献

[1] Adams, B., Pauly, M., Keiser, R. and Guibas, L.J.: Adaptively sampled particle fluids, *Proc. SIGGRAPH 2007*, p.48 (2007).

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

[3] Chang, T., Kim, H., Bae, J., Seo, J. and Noh, J.: Multi-level vorticity confinement for water turbulence simulation, *Proc. CGI 2010* (*The Visual Computer*) (2010).

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

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

[6] Harada, T., Koshizuka, S. and Kawaguchi, Y.: Smoothed Particle Hydrodynamics on GPUs, *Proc. Computer Graphics International*, pp.63–70 (2007).

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

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

[9] 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).

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

[11] Lorensen, W.E. and Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm, *Computer Graphics* (*Proc. SIGGRAPH '87*), Vol.21, No.4, pp.163–169 (1987).

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

[13] Monaghan, J.J.: Simulating free surface flows with SPH, *Journal of Computational Physics*, Vol.110, pp.399–406 (1994).

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

[15] Müller, M., Schirm, S. and Duthaler, S.: Screen space meshes, *Proc. 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.9–15 (2007).

[16] Narain, R., Sewall, J., Carlson, M. and Lin, M.C.: Fast animation of turbulence using energy transport and procedural synthesis, *Proc. SIGGRAPH Asia 2008*, pp.1–8 (2008).

[17] Pfaff, T., Thuerey, N., Cohen, J., Tariq, S. and Gross, M.: Scalable fluid simulation using anisotropic turbulence particles, *Proc. SIGGRAPH Asia 2010*, pp.174:1–174:8 (2010).

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

[19] Schechter, H. and Bridson, R.: Evolving Sub-Grid Turbulence for Smoke Animation, *Proc. 2008 ACM/Eurographics Symposium on Computer Animation* (2008).

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

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

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

[23] Stam, J. and Fiume, E.: Turbulent wind fields for gaseous phenomena, *Proc. SIGGRAPH1993*, pp.369–376, ACM, New York, NY, USA (1993).

[24] van der Laan, W.J., Green, S. and Sainz, M.: Screen

space fluid rendering with curvature flow, *Proc. 2009 Symposium on Interactive 3D Graphics and Games*, pp.91–98 (2009).
[25] Zhao, Y., Yuan, Z. and Chen, F.: Enhancing Fluid Animation with Adaptive, Controllable and Intermittent Turbulence, *Proc. 2010 ACM SIGGRAPH/ Eurographics Symposium on Computer Animation* (2010).
[26] Zhu, Y., Sifakis, E., Teran, J. and Brandt, A.: An efficient multigrid method for the simulation of high-resolution elastic solids, *ACM Trans. Graphics*, Vol.29, No.2, pp.1–18 (2010).

藤澤 誠 （正会員）

2003 年静岡大学工学部機械工学科卒業．2005 年同大学大学院理工学研究科修士課程修了．2008 年同博士課程修了．同年奈良先端科学技術大学院大学情報科学研究科助教．2011 年筑波大学大学院図書館情報メディア研究科助教．博士（工学）．CG，物理シミュレーション等の研究に従事．ACM，IEEE，バーチャルリアリティ学会，ヒューマンインタフェース学会各会員．

三村 豪

2009 年神戸大学工学部建設学科卒業．2011 年奈良先端科学技術大学院大学情報科学研究科修士課程修了．修士（工学）．

天野 敏之

2000 年大阪大学大学院基礎工学研究科博士後期課程修了．名古屋工業大学助手，奈良先端科学技術大学院大学助教を経て，2011 年山形大学工学部准教授．2009 年バウハウス大学客員研究員，2011 年ヨハネス・ケプラー大学招聘研究員．博士（工学）．パターン認識，コンピュータビジョン，拡張現実感の研究に従事．電子情報通信学会シニア会員，日本バーチャルリアリティ学会，ヒューマンインタフェース学会，IEEE 各会員．

宮崎 純 （正会員）

奈良先端科学技術大学院大学情報科学研究科准教授．博士（情報科学）．1992 年東京工業大学工学部情報工学科卒業．1997 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．同大学助手を経て，2003 年より現職．2000～2001 年テキサス大学アーリントン校客員研究員．2003～2007 年科学技術振興機構さきがけ研究員．高性能・高機能データベースならびに情報検索の研究に従事．電子情報通信学会，日本データベース学会，ACM，IEEE CS 各員．

加藤 博一 （正会員）

1986 年大阪大学基礎工学部制御工学科卒業．1988 年同大学大学院修士課程修了．1989 年同大学基礎工学部助手．1996 年講師．1998 年ワシントン大学客員研究員．1999 年広島市立大学情報科学部助教授．2003 年大阪大学大学院基礎工学研究科助教授．2007 年より奈良先端科学技術大学院大学情報科学研究科教授．博士（工学）．拡張現実感，ヒューマンインタフェースの研究に従事．電子情報通信学会，ヒューマンインタフェース学会，日本 VR 学会，ACM，IEEE 等各会員．