CGI2012 manuscript No.
(will be inserted by the editor)

Simulating Water Turbulence in Particle Fluids

the date of receipt and acceptance should be inserted later

Abstract Simulating water turbulence is challenging as it requires multi-scale vortical motions to be resolved with an emphasis on reproducing small-scale details. This paper presents a novel particle-based method to simulate water turbulence. To reduce numerical dissipations, which consume small-scale details, we have incorporated the Hermite-interpolation scheme into a particle-advection process. For capturing large- and small-scale vortical motions efficiently, we propose instant-kernels and a small-scale vorticity (SSV) model, respectively. The results of simulations with various examples demonstrate the efficiency and the physical accuracy of our method in reproducing turbulence effects in particle fluids.

Keywords SPH, Water Turbulence, Fluid Simulation

1 Introduction

Turbulence is visually interesting. As observed in everyday surroundings, turbulent flow is unsteady, irregular, and seemingly random. The key challenge in simulating water turbulence lies in resolving small-scale velocities. Due to the numerical diffusion inherent in simulations, small-scale velocities tend to vanish, and simulated flows often exhibit viscous motion. In order to resolve the small-scale velocities correctly, the simulation domain should be discretized into very small spatial and temporal units. However, this incurs a heavy computational cost.

To remedy this situation, in grid-based simulations, high-order advection schemes [13,14,17,26] have been proposed. These methods reduce the numerical errors that occur in the advection process and reproduce high levels of turbulence by broadening the range of resolvable scales to some extent. Recently, in an effort to generate high-frequency velocities far beyond the resolvable scale, a variety of turbulence synthesis methods [16,20,25,22] have been proposed. Instead of resolving the velocities directly, the synthesis methods generate very fine-scale velocities by using noise functions based on the dynamics of the turbulence.

Address(es) of author(s) should be given

Unfortunately, most of the above methods are not directly applicable to particle fluids, because the high-order advection schemes are primarily designed for grid-based simulations, and because existing turbulence synthesis methods are coupled with an Eulerian grid. When they are applied to particle fluids, such coupling entails particles-to-grid or grid-to-particles interpolation procedures, which can introduce additional numerical diffusion to particle-based fluid simulations. Moreover, a high-resolution grid is required to suppress the numerical diffusion, which can undermine the meshless property of particle fluids by limiting the size of the domain to avoid high simulation costs.

In this work we introduce a novel method for particle fluids to simulate water turbulence. In contrast to existing methods [16,20,25,22], our method simulates effects of turbulence without relying on an Eulerian grid. Our Lagrangian approach consists of two components: (1) a particle-based high-order advection process, and (2) a multi-level vorticity-enhancing process. Each component is applied to particle velocities sequentially to reinforce the weakened turbulence effects. The first part, a high-order advection process, focuses on diminishing numerical diffusions that occur when integrating the particle velocities. In particular, we employ the Hermite-interpolation method [5] to particle fluids so that the computed velocities have more small-scale details. In the second part, we develop a new vorticity-enhancing process to capture vorticity forces in multi-levels. Specifically, we introduce the use of instant kernels with a scalable radius for the computation of large-scale vorticity forces. To handle costly small-scale vorticity forces, we propose a novel small-scale vorticity (SSV) model that only requires velocities from neighboring particles. By integrating these two components, we can provide an efficient particle-based technique to simulate water turbulence. Figure 1 shows the overall process.

2 Related Work

We briefly review relevant research on turbulence simulation. As our fluid solver is based on Smoothed Particle Hydrodynamics (SPH), a survey of previous work on SPH turbulence simulations follows next. We also survey the work on SPH-LES turbulence models from the computational fluid dynamics (CFD) field, as those models are closely related to our SSV model.

2.1 Turbulence Simulation

As turbulence has a wide spectrum of velocity scales, applying the direct numerical simulation (DNS) approach to resolving minute details can lead to high computational cost. Recently, as alternatives to DNS, many cost-effective methods have been proposed to simulate turbulence effects. One popular approach is the procedural turbulence-synthesis method, which was pioneered by Stam and Fiume [30]. During simulations, the costly small-scale parts are separately generated by using noise functions that are guided by low-resolution grid-velocities. This scale separation drastically reduces the total simulation cost. Stam and Fiume [30] employed an inverse Fast Fourier Transform (FFT) method to generate a small-scale velocity field. Kim et al. [16] applied a band-limited wavelet method to guide the addition of high-frequency details.

Another approach is based on modeling of the turbulence kinetic energy. This approach computes the distribution and the evolvement of the turbulence energy by solving additional energy-governing equations. Narain et al. [20] generated small-scale noise fields and synthesized them into a resolved velocity field by using an energy-transportation equation. Schechter and Bridson [25] used an extended k-ϵ method combined with a noise function to produce subgrid-scale turbulence. Recently, Pfaff et al. [22] proposed a turbulence particle method that simulates the turbulent energy dynamics by solving a well-known k-ϵ equation in a Lagrangian way.

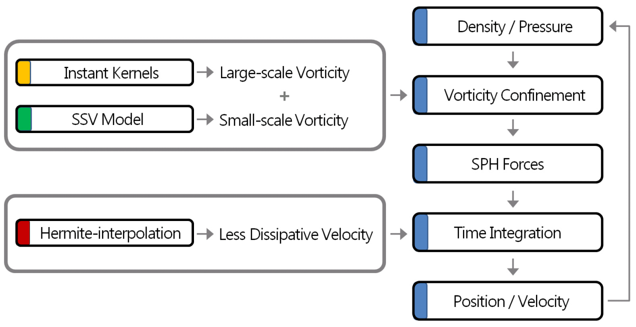

Fig. 1 Our method allows (right) SPH solver to reproduce turbulence effects by integrating (left-up) a multi-scale vorticity enhancing process and (left-down) a high-order advection process. Note that two-components of our method complement two steps in the conventional SPH solver so that they can be easily integrated with WCSPH [2] and with PCISPH [29], which emphasize the enforcement of the incompressibility.

The preservation of subgrid-scale details from numerical diffusion is another important approach for reproducing realistic turbulence effects. This approach utilizes the advantages of the non-dissipative nature of Lagrangian particles. Selle et al. [27] employed the vortex particle method to simulate water and smoke turbulence. Park and Kim [21] simulated the circulating motion of rising smoke by using vortex particles. Pfaff et al. [23] seeded vortex particles into precomputed turbulence regions to reproduce vortices around objects. Narain et al. [20] allowed particles to carry a noise function that generates a high frequency vector potential field according to their sum.

A distinctive feature of our method is that it is designed for particle fluids. As most existing methods [16, 15, 11] rely on grid-based fluid simulations, such methods are not directly applicable to particle-based fluid solvers such as SPH.

2.2 SPH Turbulence Simulation

With recent improvements on SPH solvers [8, 29, 6, 9], a number of researchers have paid attention to simulating turbulence in SPH fluids. For example, Bo et al. [35] proposed a particle-grid hybrid formulation for reproducing vortical details in a SPH fluid simulation. In their work, the creation of vortices is handled by high-resolution grids, which move along the objects. The preservation of the generated vortices is simulated by transferring the vortices from grid cells to SPH particles. Similarly, Jang et al. [12] proposed a hybrid formulation that simulates multi-level vortices in SPH. They employed multiple Eulerian grids to reproduce vorticity fields in multiple scales. The main advantage of their hybrid formulation is that it exploits the regularity of a Eulerian grid in detecting highly-deformable volumes.

While the purpose of previous works [12, 35] is similar to ours, their hybrid formulations are different with our method in two respects. First, in the previous work, the creation of vorticity forces is spatially limited due to the employed Eulerian grid; the vorticity forces are created at encasing grids that move along with objects [35] or at the deformable voxels in uniform grids [12]. Second, as the evolution of vortices is processed separately with the creation process in their formulations, additional governing equations [35] or multiple Eulerian grids are required [12] to evolve the vorticity. In contrast, our method can create and evolve vorticity for the flows in the whole domain without Euerian grids, because the vorticity can be computed and carried by SPH particles. Recently, Yuan et al. [34] proposed a swirling incentive particle (SIP) method. In their method, they introduced random swirling-probability values to efficiently mimic the stochastic nature of the turbulence. In contrast, we strive to compute multi-scale vorticity forces to capture the dynamics of turbulence faithfully.

2.3 SPH-LES Turbulence Model

In the CFD field, there have been several attempts to apply existing turbulence models, which were originally developed for grid-base methods, to SPH simulations. The large eddy simulation (LES) model has been applied to simulate collapsing water columns [10]. Shao and Gotoh [28] proposed a SPH-LES model to simulate propagation of waves close to a coastline. The k-ϵ model was applied to simulate a turbulent channel flow [32]. However, the applications of the above methods are often limited to 2D test-cases with a simple turbulence-closure model due to high computational costs required for more complex models in 3D.

A fundamental difference between the existing SPH-LES turbulence models and our method lies in the manner used to close the subgrid-scale tensor. Typically, the tensor is closed by adopting a turbulence-closure model that uses an eddy-viscosity assumption. In contrast, as we are interested in modeling and evaluating the dynamic effects of small-scale vorticity forces, we directly compute the subgrid-scale tensor term without relying on the eddy-viscosity assumption. In order to close the tensor term, we estimate the unresolved small-scale velocities by exploiting the resolved large-scale velocities. We describe this estimation process in detail in Section 4.5. In addition, as our method is purely Lagrangian, simulations in the 3D domain can easily be accelerated with the aid of the latest GPU via parallel processing.

3 SPH High-order Advection Process

3.1 Numerical Dissipation

The discretization of the momentum equation intrinsically entails numerical dissipations. As the accuracy of a particle-based simulation depends on the sampling resolution of moving interpolants (SPH particles), a very large number of samples is necessary to satisfy the resolution required to resolve turbulence effects, which tends to make numerical simulation infeasible.

In practice, the particle resolution can be adjusted to achieve balance between the visual quality and the speed of simulations. Depending on the employed resolution, SPH simulations surrender small-scale quality to some extent due to the numerical dissipations caused by the finite resolution. This phenomenon acts as an implicit filter when resolving the velocity fields. Reflecting this implicit filtering effect, we can write the momentum equation with filtered quantities as follows:

$$\rho\frac{d\bar{u}_i}{dt} = -\frac{\partial\bar{p}}{\partial x_i} + \mu\frac{\partial^2\bar{u}_i}{\partial x_j^2} + f_i^{ext}. \quad (1)$$

Here, quantities with an overline denote filtered values. i.e. $\bar{p}$ is a filtered pressure. μ is the fluid viscosity and f_i^{ext} denotes the external-force term for gravity, user defined forces, or vorticity confinement forces, where $i, j \in \{0, 1, 2\}$ is the tensor index and $i \neq j$.

Note that, compared to the momentum equation in the Navier–Stokes Equations, the non-linear advection term is omitted in Equation 1. Typically, the term is assumed to be unnecessary because SPH particles carry physical quantities in a Lagrangian manner. In an SPH solver, the integration-step implicitly replaces the removed advection term. However, this replacement is another source of numerical dissipation, because conventional integration schemes are related to a simple linear-interpolation task. To address this problem, we introduce a high-order interpolation scheme for particle advection.

3.2 High-order Particle Advection

A second-order leap-frog method is a typical choice [18, 33] for time integration. This method defines velocities at the mid-points of time-steps. For each time step, the positions of particles are updated by using the velocities of the next mid-point as $x_{t+\Delta t} = x_t + \Delta t u_{t+1/2\Delta t}$. The velocities are computed as $u_{t+1/2\Delta t} = u_{t-1/2\Delta t} + \Delta t a_t$, where a_t denotes the acceleration at time t.

While this method is a second order, it introduces numerical dissipation to a_t. The numerical dissipation results from the utilization of u_t when computing a_t.

Specifically, u_t is required when computing vorticity forces or viscosity forces [19,29], which determines the acceleration a_t. As the estimation of u_t is based on a simple linear-interpolation as $u_t = (u_{t-1/2\Delta t} + u_{t+1/2\Delta t})/2$, the numerical dissipation can be introduced to a_t, which consequently consumes high-frequency momentum at the particle-advection stage.

Adopting a high-order interpolation scheme such as the 4th-order Runge-Kutta or cubic-interpolation method can be a remedy. However, they require several points (or more memories) for evaluations. For example, a simple cubic-interpolation method would require data from 4 different time steps simultaneously for all the particles. Such memory consumption makes most high-order schemes inadequate for simulating scenes that accompany a large number of particles.

As an alternative, we employ the Hermite interpolation method. The interpolation function can be written as follows:

$$Q_H(r) = R^T M_H U^T \tag{2}$$

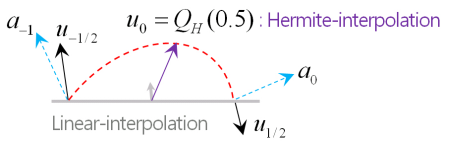

Fig. 2 Hermite-interpolation corrects the velocity at time t by using the accelerations a_{t-1} and a_t, and diminishes numerical diffusion.

where $0 \leq r \leq 1$, R^T is the coefficient vector, M_H is the Hermite basis matrix, and U^T is the Hermite geometry vector. Setting $r = 0.5$ as $u_t = Q_H(0.5)$ estimates the value of u_t. This interpolation method takes velocities $(u_{t-1/2\Delta t}, u_{t+1/2\Delta t})$ and accelerations (a_{t-1}, a_t) from two time steps as input (Figure 2), and corrects u_t to reproduce less dissipative flow motions. Note that, to avoid spurious oscilations during corrections, we employ a condition to exclude velocities that have a magnitude lower than a user-given threshold as $|u_t| > k_u$. Figure 3 shows the results of corrections.

As the requirement of data from only two time steps saves memory, GPU acceleration becomes possible, which is often limited by a small amount of available memory. In particular, compared to the typical SPH implementation, an array for storing a_{t-1} is additionally required. Details on the interpolation function $Q_H(r)$ can be found in the appendix.

4 Multi-level Vorticity Enhancing Process

The goal of the second component is to reinforce the weakened turbulence effects in the flow by enhancing

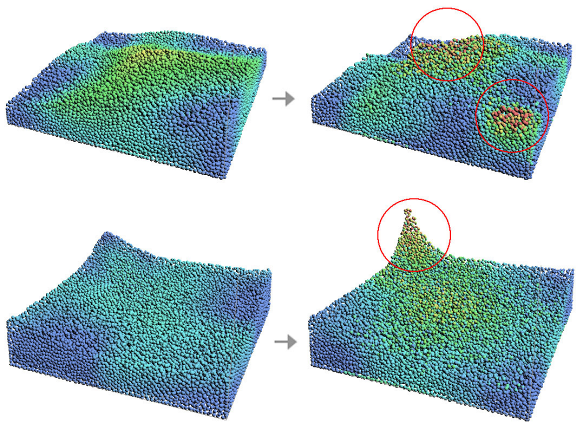

Fig. 3 Images captured from a single-dam breaking scene at frame 455 (top row) and frame 622 (bottom row), respectively. Red circles highlight the effect of our high-order advection in the corrected flow. Less dissipative velocities reproduce sharper features in the flow than the normal SPH flow does (left column). The magnitudes of the velocity for each particles are represented with a color spectrum from low (blue) to high (red).

multi-scale vorticity forces. This component takes the velocities computed from the particle-advection process as input, and computes large- and small-scale vorticity forces for each particle. In this section, we describe how to compute the vorticity forces from each scale in turn.

4.1 Particle Vorticity in SPH

We assume that the total vorticity ω_i^T for each particle is composed of large-, normal-, and small-scale vorticity as $\omega_i^T = \omega_i^L + \omega_i^N + \omega_i^S$. The criterion for the scale-separation is the size of the kernel. One way to compute such vorticity is to apply the curl operation to the particle velocity, delineated as $\omega_i = (\nabla \times u)_i$. In SPH, an approximation of the curl-operation can be written as follows:

$$\omega_i = \sum\nolimits_j ((u_i - u_j)/\rho_i) \nabla \times W_{ij} \tag{3}$$

where i,j denotes the index of particles and $W_{ij} \rightarrow W(x_i - x_j, h)$ is an interpolation kernel [19] that has a smoothing range of h. As the kernel has a compact support range h, Equation 3 measures the rotations of the velocity field locally.

This locality of computation is one of the main properties of SPH kernels that mimic the Dirac delta function. However, it poses a spatial limitation on the measurement process when computing ω_i^L and ω_i^S. In particular, as pointed out by Jang et al. [12], large-scale vorticity ω_i^L that spans beyond the size of kernels is not captured by Equation 3.

4.2 Generating Instant Kernels

We introduce instant kernels to compute such large-scale vorticity. The basic idea is to generate kernels with a radius larger than h at the volumes occupied by SPH particles (fluid-volumes) temporarily, and to utilize those kernels for computing the vorticity in large-scale as follows:

$$\omega_i^L = \sum\nolimits_j ((u_i^L - u_j^L)/\rho_i^L) \nabla \times W_{ij}^L \tag{4}$$

Here, quantities given in upper-case L are defined at each instant kernel W_{ij}^L. A challenge in this step lies in generating such kernels at every simulation frame. For the kernel generation, an important constraint is to maintain the user-given distance between kernels to guarantee the accuracy of the kernel-based operations [18]. A naive approach would be creating large-scale kernels iteratively by examining distances to the existing kernels to fulfill the generation-constraint, which is too slow for practical simulation because it entails $O(N^2)$ complexity.

In order to generate instant kernels efficiently, we employ a fast Poisson-disk sampling algorithm [3] that provides $O(N)$ complexity to the same task. After identifying the fluid-volumes, we apply this algorithm to generate new samples (or SPH kernels) with a radius of h^L at those volumes. For efficiency, we exploit a hash-grid structure when identifying the fluid volumes; if a voxel in the hash-grid has at least one SPH particle, the voxel is marked as a fluid volume. Figure 4 shows the generated instant kernels at the fluid-volumes.

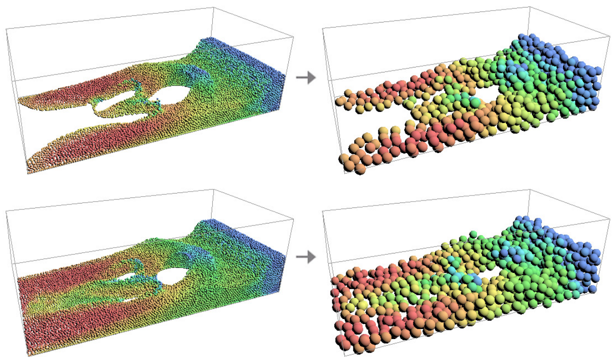
Fig. 4 Images captured from a water-stream scene at frame 77 (top row) and frame 103 (bottom row). A Fast Poisson-Disk sampling method generates large-scale instant-kernels W_{ij}^L at the fluid volumes. The number of particles decreases as 33,881→759 (top row) and 42,634→958 (bottom row), respectively. Through a weighted-averaging process, the velocities of SPH particles are smoothly transferred to nearby instant-kernels. We run this instant-kernel generation process at every simulation frame.

The physical quantities for each instant-kernel are determined by weighted-averaging or by a summation of quantities of nearby SPH particles. For example, $u_i^L = \sum_j (w_j u_j)/\sum_j w_j$, where w_j is a distance-based weight and j denotes the index of SPH particles that lie inside the instant kernel W_{ij}^L. As the distribution of kernels resembles the results of a low-resolution simulation, this process can be regarded as an instant approximation (or a low-resolution snapshot) of the current simulation.

4.3 Computing Large-scale Vorticity Forces

After determining the physical quantities for each kernel, we can compute the large-scale vorticity by using Equation 4. However, Equation 4 does not consider the symmetry of operation. The application of a symmetric formulation similar to [24] can increase the efficiency.

$$\omega_i^L = \rho_i \sum\nolimits_j m_j^L \left[\left(\frac{u_i^L}{\rho_j^{2\,L}} \nabla \times W_{ij}^L\right) + \left(\frac{u_j^L}{\rho_i^{2\,L}} \nabla \times W_{ji}^L\right) \right] \tag{5}$$

The next step is to compute forces for enhancing the vortical motions of SPH fluid by using the resolved vorticity field. For this, we adopted the vorticity confinement method. This method [31] formulates the vorticity force at each particle as

$$f_i^L = \varepsilon (N_i^L \times \omega_i^L) \tag{6}$$

where ε is a user-given scaling-factor, and N_i^L is the normalized vorticity-location vector ($N_i^L = \eta_i^L / \left|\eta_i^L\right|$). Here, η_i^L can be evaluated via SPH interpolation as $\eta_i^L = \left\langle \nabla \left|\omega_i^L\right| \right\rangle = \sum_j (m_j^L/\rho_j^L) \left|\omega_j^L\right| \nabla W_{ij}^L$. By applying this vorticity confinement method to the instant kernels, the large-scale vorticity force at each SPH particle is obtained in the form of an external force as f_i^L.

4.4 Small-scale Vorticity in SPH

The small-scale vorticities are the most difficult to resolve as they are prone to numerical dissipations. This issue can be addressed by incorporating the instant-kernels into the small-scale. In a large-scale scene, however, such application can lead to very high computational cost, as it requires an excessive increase of the number of sampling points (or kernels). Moreover, the locations of tiny instant-kernels inside the SPH particles are especially difficult to determine, as they are chosen randomly [3]. While such randomness can be exploited to mimic the stochastic nature of turbulence, as in [34], it can incur spurious fluctuations in the flow by replacing the dynamics of turbulence.

To pursue phyiscal plausibility, we investigated existing turbulence models including LES. In particular, the LES method avoids computation of high-cost parts by separating them from the total velocity-field, and by utilizing a subgrid-scale turbulence-model based on an assumption that small-scale structures in turbulence are homogeneous in character. The LES governing equation can be written as follows:

$$\frac{\partial \bar{u}_i}{\partial t} + \frac{\partial \bar{u}_i \bar{u}_j}{\partial x_j} = -\frac{1}{\rho}\frac{\partial \bar{p}}{\partial x_i} + \nu \frac{\partial^2 \bar{u}_i}{\partial x_j^2} - \frac{\partial \tau_{ij}^{sgs}}{\partial x_j} \tag{7}$$

The above equation is a filtered version of the momentum equation in the Navier–Stokes equations, and describes temporal and spatial evolution of the large-scale velocities. The non-linearity of the advection term in the Navier–Stokes equations leads to the appearance of the subgrid-scale stress tensor τ_{ij}^{sgs} after filtering. This subgrid-scale stress term represents the dynamic effects of the small-scale velocities on the resolved scales in the form of additional stress. Motivated by this, we developed a new method, called a SSV model, to compute small-scale vorticity forces.

4.5 Small-scale Vorticity (SSV) Model

The main purpose of the SSV model is to use the subgrid-scale tensor to compute small-scale vorticity forces. By applying the LES decomposition rule ($u_i = \bar{u}_i + u'_i$), the tensor $\tau_{ij}^{sgs} = \overline{u_i u_j} - \bar{u}_i \bar{u}_j$ can be written as follows:

$$\tau_{ij}^{sgs} = \overline{(\bar{u}_i + u'_i)(\bar{u}_j + u'_j)} - \bar{u}_i \bar{u}_j. \tag{8}$$

According to the Leonard decomposition [1], the term is composed of three parts; $L_{ij} = \overline{\bar{u}_i \bar{u}_j} - \bar{u}_i \bar{u}_j$ is the Leonard stress, which denotes the interactions among velocities in the large scale; $C_{ij} = \overline{\bar{u}_i u'_j} + \overline{u'_i \bar{u}_j}$ represents the interaction of velocities between the large and small scales; $R_{ij} = \overline{u'_i u'_j}$ is the Reynolds stress that represents the interactions among velocities in the small scale. The generation of small eddies and their dynamics is closely related to τ_{ij}^{sgs}. Note that, as the tensor τ_{ij}^{sgs} contains unresolved components u'_i, an assumption for closure is required. In LES, the eddy-viscosity assumption is employed to posit a linear relationship between the subgrid-scale stress and the large-scale strain-rate [4]. As the LES method focuses on reproducing turbulent large-scale eddies, the LES method approximates the small-scale parts by simplifying their dynamic structures with the turbulence-closure model, and saves computational resources.

In contrast, as our goal is to enhance both large- and small-scale vortices separately by constructing a vorticity-force field for each scale, we compute the tensor term τ_{ij}^{sgs} without relying on the eddy-viscosity assumption. The computation process is composed of four steps. First, we replace the unresolved velocity u'_i with its filtered version $\bar{u}'$. By this substitution, Equation 8 can be rewritten with the filtered velocity $\bar{u}'$ as follows:

$$\tau_{ij}^{sgs} = \overline{(\bar{u}_i + \bar{u}'_i)(\bar{u}_j + \bar{u}'_j)} - \bar{u}_i \bar{u}_j, \tag{9}$$

where $\bar{u}_i$ denotes the resolved SPH velocities, as in Equation 1. Second, in order to compute the first term in the RHS of Equation 9, we estimate the unresolved velocities by utilizing velocities of nearby SPH particles. Unlike u'_i, the value of $\bar{u}'_i$ can be computed. As $\bar{u}$ is known according to Equation 1, $\overline{\bar{u}}$ can be computed by applying a low-pass filter as follows:

$$\overline{\bar{u}_i} = (1-\alpha)\bar{u}_i + \alpha \left(\frac{\sum_j \bar{u}_j S_{ij}(h_e)}{\sum_j S_{ij}(h_e)} \right) \tag{10}$$

In this equation, $S_{ij}(h_e) = (1 - (|x_i - x_j| / h_e)^2)^3$, α is a scaling-weight for filtering and $h_e = 2h$ is the effective radius. Subtracting $\overline{\bar{u}}$ from the flow velocity $\bar{u}$ provides the value of $\bar{u}'$. Third, as the first term in the RHS of Equation 9 is a filtered tensor, we apply the same low-pass filter to $(\bar{u}_i + \bar{u}'_i)(\bar{u}_j + \bar{u}'_j)$. Finally, the tensor τ_{ij}^{sgs} can be computed by subtracting $\bar{u}_i \bar{u}_j$ from the $\overline{(\bar{u}_i + \bar{u}'_i)(\bar{u}_j + \bar{u}'_j)}$. For each SPH particle, the computed tensor is stored. Algorithm 1 shows the computation process in detail.

Algorithm 1 Computing subgrid-scale tensors

```
Input: $[p_i]$ (SPH particles), $0.3 \le \alpha \le 0.5$
 1: // [ ] denotes a resizable array
 2: // step1: compute a filtered small-scale velocity $\bar{u}'_i$
 3: for all $p_i$ in $[p_i]$ do
 4:     $[p_j] \leftarrow$ get neighboring particles within $h_e (= 2h)$
 5:     $[\bar{u}_j] \leftarrow$ get large-scale velocities of $[p_j]$
 6:     $\bar{u}_i \leftarrow$ get large-scale velocity of $p_i$
 7:     $\overline{\bar{u}_i} \leftarrow$ apply low-pass filter($\bar{u}_i$, $[\bar{u}_j]$, $\alpha$)
 8:     $\bar{u}'_i = \bar{u}_i - \overline{\bar{u}_i}$
 9:     $\tau^e_i(l) = (\bar{u}_i + \bar{u}'_i)(\bar{u}_i + \bar{u}'_i)^T$
10:     $\tau^e_i(r) = \bar{u}_i(\bar{u}_i)^T$
11: end for
12: // step2: compute a subgrid-scale tensor $\tau^e_i$
13: for all $p_i$ in $[p_i]$ do
14:     $[p_j] \leftarrow$ get neighboring particles within $r_e$
15:     $[\tau^e_j(l)] \leftarrow$ get left term of the tensors from $[p_j]$
16:     $\tau^e_i(l) \leftarrow$ get left term of the tensor from $p_i$
17:     $\overline{\tau^e_i(l)} \leftarrow$ apply low-pass filter($\tau^e_i(l)$, $[\tau^e_j(l)]$, $\alpha$)
18:     $\tau^e_i = \overline{\tau^e_i(l)} - \tau^e_i(r)$
19: end for
```

To exploit the computed tensor, we evaluate the last term in the RHS of Equation 7. As the divergence of a tensor becomes a vector according to the tensor-index summation rule (See appendix for details), the subgrid-scale force term can be written as $f_i^{sgs} = \nabla \cdot \tau_i^{sgs}$. To compute small-scale vorticity, we apply the same curl

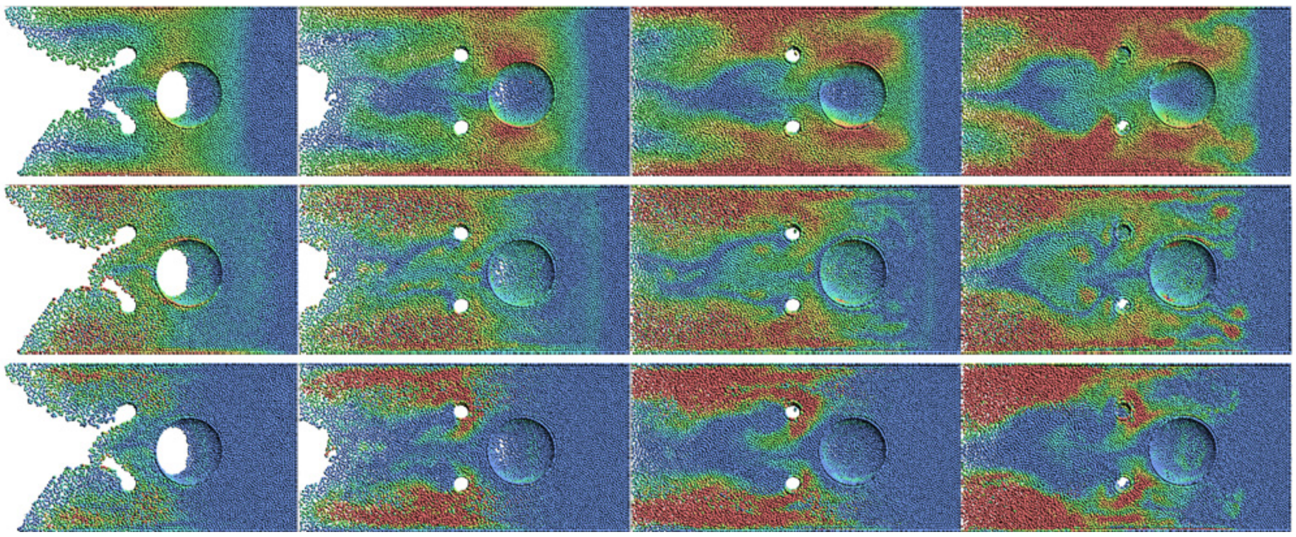

Fig. 5 The visualizations of multi-scale vorticity fields in a water-stream scene. Images in each column are captured from bottom-view of the scene at frame 87, 117, 147 and 177, respectively. Each row shows the vorticity field in the (top row) large-scale, (middle row) normal-scale and (bottom row) small-scale. The particles are colorized according to the magnitudes of the vorticity. The red color represents high magnitude and the blue color represents low magnitude. The scene containes an emitter at the right side of the simulation domain. The obstacles (poles and a sphere) were omitted for better visualization.

operator used in [24] to the estimated small-scale velocity field $\bar{u}'_i$.

$$\omega_i^S = |\nabla \cdot \tau_i^{sgs}| \, (\nabla \times \bar{u}'_i) \tag{11}$$

Note that we multiply the magnitude of the subgrid-scale force term. This process adjusts the magnitude of the small-scale vorticity, and reflects the small-scale dynamics in turbulence. Similar to Section 4.3, we apply the vorticity confinement method to ω_i^S to compute small-scale vorticity-forces as $f_i^S = \varepsilon(N_i^S \times \omega_i^S)$, where N_i^S is a normalized vorticity-location vector. Figure 6 summarizes the process of computing small-scale vorticity forces.

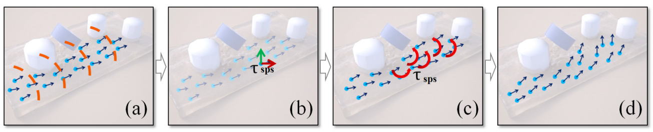

Fig. 6 The small-scale vorticity forces are obtained by a) computing filtered velocities to b) evaluate the subgrid-scale tensor term. c) The small-scale vorticity field is constructed by using curl-operation. d) The vorticity confinement method is employed to compute the small-scale vorticity forces.

4.6 Integrating Turbulence Effects into SPH Solver

The computed vorticity forces from Section 4.3 and Section 4.5 including the normal-scale vorticity force ($f_i = \nabla \times u_i$) are integrated into Equation 1 as a single external force as $f_i^{ext}(= k_L f_i^L + k_N f_i^N + k_S f_i^S)$, where k_L, k_N and k_S are turbulence-scaling parameters given by users. In addition, we solve the vorticity equation [21,27,35,34] for each scale to evolve the vorticity in a Lagrangian manner as follows:

$$\frac{d\omega}{dt} = \nu \nabla^2 \omega + T(u) \tag{12}$$

Note that, we added a vorticity source-term $T(u)$. This source-term is related to the baroclinic contributions occuring from density fluctuations in particle fluids, and represents the creation of vorticity in the flow. In contrast to the existing vorticity seeding process, which uses a random initailization [27] or a manual intervention [34], our method utilizes the computed vorticity $(\omega_i^L, \omega_i^N, \omega_i^S)$ as the source of vorticity for each scale. To avoid exponential increase of the magnitude of vorticity, we employ an attenuation factor similar to [35].

5 Results and Discussion

5.1 Turbulence Effects

Figure 5 visualizes the spatial distribution of large-, normal-, and small-scale vorticity. Images in each row show different complexions on the generation and the evolution of the vorticity. First, the large-scale kernels (instant-kernels) outperform the normal-scale SPH kernels in capturing the rotational motions of the main-stream. Thus, the large-scale vorticity mainly emerges around boundaries such as a spherical obstacle and domain boundaries. Such large vorticity is generated by

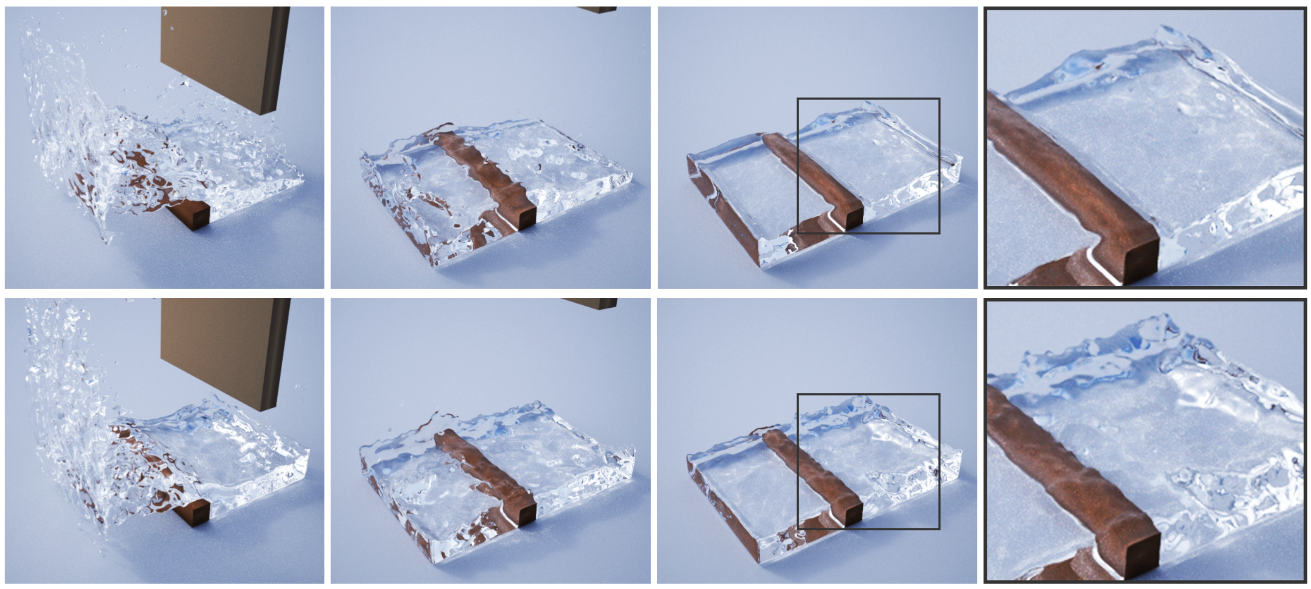

Fig. 7 A dam breaking simulation. The images are sampled at frames 50, 100, 150, and 150 (magnified), respectively. The top row shows the results from a conventional SPH method and the bottom row shows the results from our method.

the collision of the mainstream and the wall, and by strains of the mainstream around the swirling eddies. Second, the normal-scale kernels complement the large-scale kernels by capturing the local swirling motions inside the mainstream. The mainstream is divided into two sections after colliding with a sphere. The normal-scale vorticity emerges inside those sections, and merged and split along the sections. Finally, the small-scale vortices appear mostly in the area where the large-scale vortices do not exist. In addition, small vortices emerge where normal-scale vortices have dissipated. This verifies that our model effectively handles the dynamics of turbulence in a physically plausible manner.

5.2 Comparisons with a Reference Simulation

Figure 7 shows an example of a dam breaking. The overall flows from both methods have similar motions, as the two methods share the same velocity solver and boundary handling mechanism. However, the reference method generates smoother surfaces lacking small-scale ripples while ripples are found in the results obtained via our method. In addition, the swirly motions last longer in our results. Our method preserves small details by employing a high-order advection scheme, and injects additional energies computed from vorticity fields into the flow via the vorticity confinement method to enhance vortical motions.

Figure 8 demonstrates a water-stream scene. The images in the bottom-row show natural deformations on water surfaces with a combination of large- and small-scale vortical motions. The images in the middle column show noticeable differences behind the poles; the same number of employed particles led to different local distributions of particles in those areas. This phenomenon is related to the momentum injected during vorticity confinement, which makes the flow move slightly faster. The right column shows that the effects of various scales of the vortices satisfy the boundary conditions appropriately, which is difficult with the noise-based turbulence synthesis methods [16, 20]. In our results, the collisions from both large- and small-scale velocities were handled correctly. In addition, the vortical motions appropriately deformed the water surfaces in such boundaries. The accompanying video shows that the reproduced turbulence effects vanished and eventually stabilized, which is a very important requirement for the realistic simulation of water turbulence.

5.3 Performance

The simplicity of our approach allows easy implementation on a graphics processor. In every simulation step including a low-pass filtering step, only nearby particles are required for computing physical quantities at each particle. Consequently, every physical quantity can be computed in parallel. We used CUDA to implement both large- and small-scale parts to reduce the computation time in the simulation. Table 1 shows average timings for the simulations.

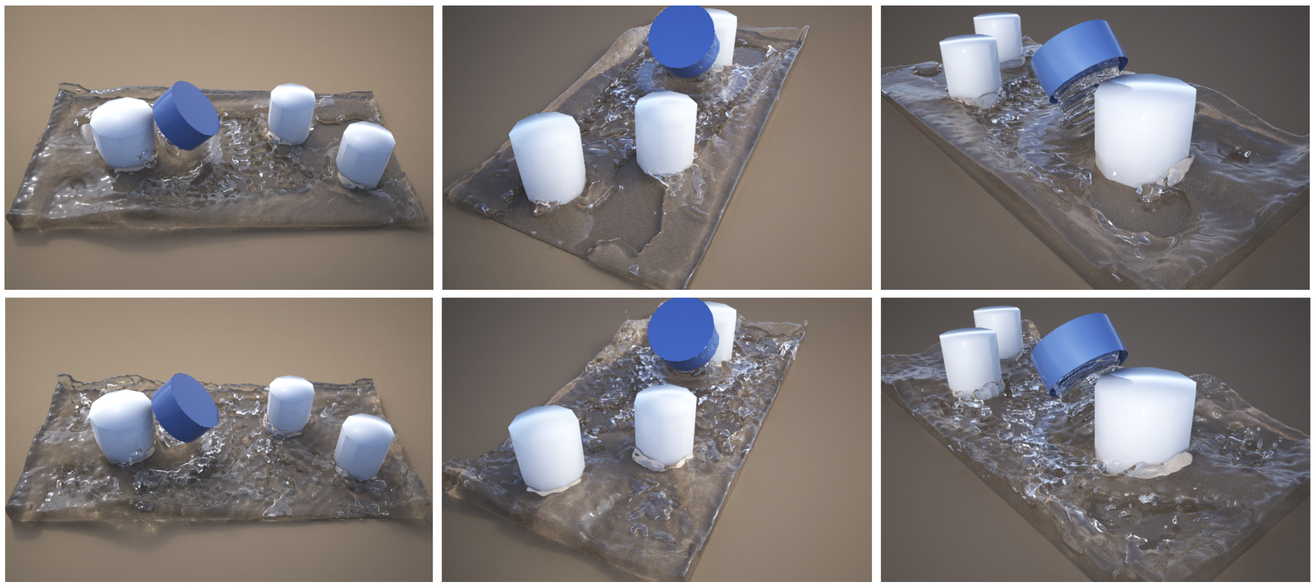

Fig. 8 Comparisons of a water stream scene. Results from the conventional SPH method (top row) and those from our method (bottom row). The blue cylinder-shaped source emits SPH particles for the stream. The white poles represent stationary obstacles. Our results show more exquisite turbulent motions in the flow. The image pair in each column was captured with different camera angles at three simulation frames; frames 50, 100, and 150.

Scene name	# of particles	SPH method	Our method	Time difference
Dam	24,000	1.13s	1.58s	0.45s
Stream	45,000	1.023s	1.26s	0.237s

Table 1 Performance comparison. All simulations were run utilizing a workstation with an Intel CPU, 8GB of RAM and a NVIDIA GTX580 Graphics card. Timing is given per frame. The offline rendering time is not included. The averages were computed by dividing the measured simulation times by the number of frames for each scene. The number of particles of the stream scene denotes the number of emitted particles at the last frame of the simulation.

6 Conclusion

We have presented a new method to enhance vortical motions in SPH fluid simulations. Unlike previous methods that rely on an Eulerian grid, we utilize particles only for the simulation of turbulence effects. Our method successfully reproduced various scales of vortical motions by employing a high-order advection scheme, and by adopting a scale-separation metaphor to derive a SSV model based on the LES method. The absence of Eulerian grids or complex data structures allows easy parallelization in implementation. By using the GPU for accelerations, our method efficiently increased the swirly motions in the flows with low simulation cost. We demonstrated the effectiveness of the proposed method by showing test results with realistic rendering. The turbulence energy cascading process and the decay of the vorticity are properly simulated in those scenes. Our method can accurately handle the effects of collisions at the boundaries.

While the main benefits of our method over previous methods lie in its simplicity and physical correctness, the resulting accuracy may not be very high, as we approximate the small-scale velocities with the normal-scale velocities (Section 4.5). In order to increase the accuracy, the small-scale velocity field has to be simulated with a high sampling resolution to fulfill the required length scale for spatial and temporal discretization. However, such dense sampling can cause the simulation to be as expensive as DNS. As a future work, we plan to extend our method by adopting acceleration algorithms such as Fast Multipole Method [7] to exploit more sampling particles for simulation.

References

1. A, L.: Energy cascade in large-eddy simulations of turbulent fluid flows. Turbulent diffusion in environmental pollution. Proceedings of the Second Symposium **A** (1974)
2. Becker, M., Teschner, M.: Weakly compressible sph for free surface flows. In: Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '07, pp. 209–217 (2007)
3. Bridson, R.: Fast poisson disk sampling in arbitrary dimensions. In: ACM SIGGRAPH 2007 sketches, SIGGRAPH '07 (2007)
4. Dalrymple, R., Rogers, B.: Numerical modeling of water waves with the sph method. Coastal Engineering **53**(2-3), 141 – 147 (2006)

5. Fedkiw, R., Stam, J., Jensen, H.W.: Visual simulation of smoke. In: SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques, pp. 15–22 (2001)
6. Goswami, P., Schlegel, P., Solenthaler, B., Pajarola, R.: Interactive SPH Simulation and Rendering on the GPU. In: M. Otaduy, Z. Popovic (eds.) Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 1–10 (2010)
7. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. Journal of Computational Physics **73**(2), 325 – 348 (1987). DOI 10.1016/0021-9991(87)90140-9
8. Harada, T., Koshizuka, S., Kawaguchi, Y.: Smoothed Particle Hydrodynamics on GPUs. In: Proc. of Computer Graphics International, pp. 63–70 (2007)
9. He, J., Chen, X., Wang, Z., Cao, C., Yan, H., Peng, Q.: Real-time adaptive fluid simulation with complex boundaries. Vis. Comput. **26**, 243–252 (2010)
10. Issa, R., Violeau, D., Laurence, D.: A first attempt to adapt 3d large eddy simulation to the smoothed particle hydrodynamics gridless method. In: Proceedings of the International Conference on Computational and Experimental Engineering and Sciences, pp. 87–94 (2005)
11. Jang, T., Kim, H., Bae, J., Seo, J., Noh, J.: Multilevel vorticity confinement for water turbulence simulation. Vis. Comput. **26**, 873–881 (2010)
12. Jang, T., i Ribera, R.B., Bae, J., Noh, J.: Simulating sph fluid with multi-level vorticity. International Journal of Virtual Reality **1** (2011)
13. Kim, B., Liu, Y., Llamas, I., Rossignac, J.: Advections with significantly reduced dissipation and diffusion. IEEE Transactions on Visualization and Computer Graphics **13**(1), 135–144 (2007)
14. Kim, D., Song, O.Y., Ko, H.S.: A semi-lagrangian cip fluid solver without dimensional splitting. Comput. Graph. Forum **27**(2), 467–475 (2008)
15. Kim, D., Song, O.y., Ko, H.S.: Stretching and wiggling liquids. In: SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers, pp. 1–7 (2009)
16. Kim, T., Thürey, N., James, D., Gross, M.: Wavelet turbulence for fluid simulation. In: SIGGRAPH '08: ACM SIGGRAPH 2008 papers, pp. 1–6 (2008)
17. Molemaker, J., Cohen, J.M., Patel, S., Noh, J.: Low viscosity flow simulations for animation. In: SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 9–18 (2008)
18. Monaghan, J.J.: Smoothed particle hydrodynamics. Reports on Progress in Physics **68**(8), 1703 (2005)
19. Müller, M., Charypar, D., Gross, M.: Particle-based fluid simulation for interactive applications. In: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, SCA '03, pp. 154–159 (2003)
20. Narain, R., Sewall, J., Carlson, M., Lin, M.C.: Fast animation of turbulence using energy transport and procedural synthesis. In: SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers, pp. 1–8 (2008)
21. Park, S.I., Kim, M.J.: Vortex fluid for gaseous phenomena. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 261–270 (2005)
22. Pfaff, T., Thuerey, N., Cohen, J., Tariq, S., Gross, M.: Scalable fluid simulation using anisotropic turbulence particles. In: ACM SIGGRAPH Asia 2010 papers, SIGGRAPH ASIA '10, pp. 174:1–174:8 (2010)
23. Pfaff, T., Thuerey, N., Selle, A., Gross, M.: Synthetic turbulence using artificial boundary layers. In: SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers, pp. 1–10 (2009)
24. Price, D.J.: Smoothed Particle Magnetohydrodynamics IV - Using the Vector Potential (2009). * Brief entry *
25. Schechter, H., Bridson, R.: Evolving sub-grid turbulence for smoke animation. In: SCA '08: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp. 1–7 (2008)
26. Selle, A., Fedkiw, R., Kim, B., Liu, Y., Rossignac, J.: An unconditionally stable maccormack method. J. Sci. Comput. **35**(2-3), 350–371 (2008)
27. Selle, A., Rasmussen, N., Fedkiw, R.: A vortex particle method for smoke, water and explosions. In: SIGGRAPH '05: ACM SIGGRAPH 2005 Papers, pp. 910–914 (2005)
28. Shao, S., Gotoh, H.: Simulating coupled motion of progressive wave and floating curtain wall by sph-les model. Coastal Engineering Journal **46**, 172–202 (2004)
29. Solenthaler, B., Pajarola, R.: Predictive-corrective incompressible sph. In: ACM SIGGRAPH 2009 papers, SIGGRAPH '09, pp. 40:1–40:6 (2009)
30. Stam, J., Fiume, E.: Turbulent wind fields for gaseous phenomena. In: SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 369–376 (1993)
31. Steinhoff, J., Underhill, D.: Modification of the euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. Physics of Fluids **6**(8), 2738–2744 (1994)
32. Violeau, D.: One and two-equations turbulent closures for smoothed particle hydrodynamics. In: Proceedings of the VIth Int. Conf. Hydroinformatics, pp. 87–94 (2004)
33. Yan, H., Wang, Z., He, J., Chen, X., Wang, C., Peng, Q.: Real-time fluid simulation with adaptive sph. Comput. Animat. Virtual Worlds **20**, 417–426 (2009)
34. Yuan, Z., Zhao, Y., Chen, F.: Incorporating stochastic turbulence in particle-based fluid simulation. Vis. Comput. **7**, 1–10 (2011)
35. Zhu, B., Yang, X., Fan, Y.: Creating and preserving vortical details in sph fluid. Comput. Graph. Forum pp. 2207–2214 (2010)

A Divergence of Subgrid-scale Tensor

$$\frac{\partial \tau_{ij}}{\partial x_j} = \begin{pmatrix} \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} \\ \frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y} + \frac{\partial \tau_{yz}}{\partial z} \\ \frac{\partial \tau_{zx}}{\partial x} + \frac{\partial \tau_{zy}}{\partial y} + \frac{\partial \tau_{zz}}{\partial z} \end{pmatrix} \tag{13}$$

B Hermite Interpolation for Velocity

$$Q_H(r) = R^T M_H U^T \tag{14}$$

$$R^T = \left[r^3 \; r^2 \; r \; 1 \right] \tag{15}$$

$$U^T = \left[u_{t-1/2\Delta t} \; u_{t+1/2\Delta t} \; a_{t-1} \; a_t \right] \tag{16}$$

$$M_H = \begin{pmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{17}$$