Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2010)
M. Otaduy and Z. Popovic (Editors)

Enhancing Fluid Animation with Adaptive, Controllable and Intermittent Turbulence

Ye Zhao and Zhi Yuan and Fan Chen[†]

Dept. of Computer Science, Kent State University, Ohio, USA

Abstract

This paper proposes a new scheme for enhancing fluid animation with controllable turbulence. An existing fluid simulation from ordinary fluid solvers is fluctuated by turbulent variation modeled as a random process of forcing. The variation is precomputed as a sequence of solenoidal noise vector fields directly in the spectral domain, which is fast and easy to implement. The spectral generation enables flexible vortex scale and spectrum control following a user prescribed energy spectrum, e.g. Kolmogorov's cascade theory, so that the fields provide fluctuations in subgrid scales and/or in preferred large octaves. The vector fields are employed as turbulence forces to agitate the existing flow, where they act as a stimulus of turbulence inside the framework of the Navier-Stokes equations, leading to natural integration and temporal consistency. The scheme also facilitates adaptive turbulent enhancement steered by various physical or user-defined properties, such as strain rate, vorticity, distance to objects and scalar density, in critical local regions. Furthermore, an important feature of turbulent fluid, intermittency, is created by applying turbulence control during randomly selected temporal periods.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.5]: Computational Geometry and Object Modeling —Physically Based Modeling; Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism —Animation

Keywords: Turbulence, Fluid Simulation, Animation Control, Random Forcing, Intermittency, Kolmogorov

1. Introduction

Fluid simulation, mostly based on numerically solving the governing Navier-Stokes (NS) equations, has achieved great success in computer graphics, which has led to astounding appearances in movies and games of streaming water, flaming fire, propagating smoke, and more. Recently, many researchers have endeavored to introduce turbulence for enhancing fluid animations. As stated by Taylor and von Kármán in 1937 (at Royal Aeronautical Society): "Turbulence is an irregular motion which in general makes its appearance in fluids, gaseous or liquid". However, turbulence could also mean "very hard to predict" due to the very large degree of freedom with high Reynolds number (*Re*). Turbulent fluids exhibit intrinsic fluctuations in a wide range of length and time scales, featuring stochastic and intermittent dynamics.

Strategy and Related Work Direct numerical simulation (DNS) cannot directly model turbulent behavior with a very large *Re* due to limited computational resources. Furthermore, fast simulation and interaction are very important for animation design and control in computer graphics. Therefore, graphical animations of turbulent fluids typically involve coupling synthetic small-scale (subgrid) noise, modeling chaotic dynamics, to a coarse-grid NS simulator. This strategy relies on the Reynolds decomposition that breaks the instantaneous velocity field $\mathbf{u}$ into a mean (DNS resolved) field $\mathbf{U}$ and a rapidly fluctuating component $\mathbf{u}'$. Based on this, the methodology can be described as

$$\mathbf{u} \Leftarrow \mathsf{NS}(\mathbf{U}) \oplus \mathsf{ST}(\mathbf{u}'). \tag{1}$$

Follow this strategy, several successful approaches [SF93, RNGF03, KTJG08, NSCL08, SB08, PTSG09] provide various implementations for: a fluid solver NS() simulating the mean flow $\mathbf{U}$, a noise-based procedure ST() synthesizing and evolving the synthetic fluctuation $\mathbf{u}'$, and an integration

[†] {zhao,zyuan,fchen}@cs.kent.edu

DOI: 10.2312/SCA/SCA10/075-084

model $\oplus$ coupling them together. Here, NS() is usually implemented as a stable solver on a coarse grid.

In noise synthesis ST(), these methods generate turbulence $\mathbf{u}'$ with random functions at various spatial scales and frequencies, sometimes referred to as octaves. A chaotic field was modeled in the frequency domain directly [SF93, Sta97, RNGF03]. Recently, the curl operation, following Bridson et al. [BHN07], is used on Perlin noise [NSCL08, SB08] and wavelet vector noise [KTJG08], or alternatively, vortex particles belonging to different wave numbers are randomly seeded according to probabilities precomputed by artificial boundary layers [PTSG09]. During the dynamic noise generation, the energy transport among octaves is modeled by a simple linear model [SB08], an advection-reaction-diffusion PDE [NSCL08], locally assembled wavelets [KTJG08] or decay of particles [PTSG09]. As a common recipe, the celebrated Kolmogorov 1941 theory (K41) of energy cascade is applied [Fri95]. These methods are built upon two *graphical assumptions*: **A**. The K41 inspired energy transport is modeled within the limited grid resolution in NS() and ST(). In fact, the simulation scale is much larger than that of the inertial subrange described in K41 for very high *Re* flows; **B**. The energy cascade happens locally, where the energy is carried by local scalars or particles. However, the theory actually postulates the spectral statistics of global energy distribution that may not be spatially localized. Nevertheless, the assumptions, which we will follow, enable turbulence creation and feedback to the mean flow in a graphical way, leading to great success in improving fluid animation techniques.

In integration operation $\oplus$, there exist two challenges: one is the magnitude relation between $\mathbf{u}'$ and $\mathbf{U}$, and the other is the temporal evolution of $\mathbf{u}'$ with respect to $\mathbf{U}$. Early work [SF93, Sta97, RNGF03] advected gas by $\mathbf{u}'$ and $\mathbf{U}$ together. The velocity magnitude matching is achieved easily with the graphical assumption **B**: at a location the kinetic energy of the smallest resolved scale of $\mathbf{U}$ can be used to derive the kinetic energy of $\mathbf{u}'$ from K41 so that the velocity relation is determined. In different implementations, Schechter et al. [SB08] seeded the resolved energy artificially, Kim et al. [KTJG08] used a locally computed kinetic energy, Narain et al. [NSCL08] adopted a strain rate related viscosity hypothesis, and Pfaff et al. [PTSG09] created the confined vorticity also following the hypothesis. The strain rate based method is physically meaningful but not always suitable for a graphical animator, who for example wants to introduce boundary effects for a small obstacle not solved by the existing coarse simulation. In this case, sufficient strain information from $\mathbf{U}$ cannot be provided. For this reason, Pfaff et al. [PTSG09] sought solution in a high resolution precomputation.

However, it is not very pleasant to address the second challenge, where the generated fields should temporally evolve with the large-scale flow. To handle this, small-scale $\mathbf{u}'$ fields are deliberately managed with texture distortion detection [KTJG08], through an empirical rotation scalar field [SB08] or by special noise particles [NSCL08]. These approaches achieve good results while introducing complexity originating from implementing $\oplus$ as a simple vector combination outside of the governing NS equations. Pfaff et al. [PTSG09] instead coupled a stable solver with a vortex particle system, in which $\oplus$ is realized as particle forces. It requires careful management of particles and the wall-introduced turbulence is the focus.

In this paper, we propose a framework to integrating turbulence to an existing/ongoing flow suitable for graphical controls. In comparison with direct field addition, our framework avoids the artificial and complex coupling by solving integration inside the NS solvers.

Our Solution We model fluid fluctuation by a random process of adaptive turbulence forcing from a sequence of precomputed force fields with scale and spectrum control:

- ST(): In retrospect of K41, Kolmogorov assumed that at small scales the flow will be statistically homogeneous and isotropic. Inspired by this, we spectrally synthesize small-scale homogeneous fields with respect to an energy spectrum distribution, which follows K41's $-\frac{5}{3}$ law or user-prescribed ones. A sequence of synthetic fields are pre-generated and play a role as random forces.
- $\oplus$: Instead of being combined with $\mathbf{U}$ directly, the synthesized fields represent chaotic forcing $\mathbf{f}$ perturbing the resolved mean flow. Thus, $\oplus$ is realized in a forced NS simulator (FNS()), inherently leading to smooth feedback and temporal evolution. Eqn. 1 can be rewritten as

$$\mathbf{u} \Leftarrow \mathsf{FNS}(\mathsf{NS}(\mathbf{U}), \mathsf{ST}(\mathbf{f})). \tag{2}$$

  Moreover, this scheme handles boundaries inside FNS() where many successful methods exist, releasing ST() from special operations of previous endeavors.
- NS(): As an independent process from original simulation, our framework can be combined with a large body of work on NS solvers (e.g. [Sta99, MCP*09, FSJ01, MCG03]).

Using random forcing is a standard method in physics to study and evaluate liberally-developed *homogeneous and isotropic* turbulence [CD96, LDM03]. Here we contribute to explore it in integrating synthetic turbulence with external large-scale flows. This approach is different from simply using a high-resolution fluid simulation. First, the randomness is critical in modeling turbulent dynamics. Second, the random forcing can represent higher frequency effects not limited by a given high-resolution grid. Furthermore, the method receives input from large-scale flows and controls their effect on resultant fields. More important, as a graphical tool, we practically apply this turbulent forcing only in necessary local areas and/or in appropriate temporal periods, which are defined by user-interests, boundaries, strain and vorticity etc. Furthermore, we can apply random

forces with large rotational scales, modeling chaotic fluctuation overlapped with the resolved flow. We therefore not only model small-scale turbulence but also inject manipulative turbulence in large octaves. To make more realistic fluid animation, we further model the temporal intermittency by randomly controlling the turbulence forcing in a heuristic way. Our effort, to the best of our knowledge, is the first attempt in graphics to include this important feature of turbulent fluids.

In summary, we implement an adaptive fluid animation scheme with controllable turbulent behavior, which meets the demand in many interactive applications. Our contributions can be summarized as:

- *Random turbulence forcing* integrates synthetic turbulent fluctuation with large-scale simulation, with respect to spatial and temporal consistency;
- *Controllable turbulence amplitude* includes unresolved subgrid fluctuation, and/or overlapped large scale chaos;
- *Spectral synthesis* of turbulence forces enables easy implementation and direct spectral control, following arbitrary energy spectrum descriptions. A sequence of small-scale force fields is independently pre-computed without extra simulation overhead and can be reused for different animations;
- *Adaptive turbulence* takes effect in local areas and/or in particular time ranges, conditioned by physical or user-defined features;
- *Intermittent turbulence* provides more realistic turbulent fluid animation.

2. Background

A variety of approaches have been published for physically-based modeling of fluid phenomena. A stable fluid solver is devised [Sta99] using semi-Lagrangian advection schemes without time step restrictions, which contributes to the enhanced visual impact of fluid animations [Bri08]. Many approaches are proposed to address the energy loss due to numerical solution of stable fluids, including feeding rotational forces [FSJ01], coupling Lagrangian vortex particles [SRF05], substituting of advection with the Lagrangian fluid-implicit-particles (FLIP) [ZB05]. Furthermore, different numerical schemes are introduced including higher order advection scheme (BFECC) [KLLR07], enforced circulation preservation [ETK*07] and energy preserving scheme [MCP*09]. Alternative paths include adaptive high-resolution simulation (e.g. [LGF04]), particle fluids (e.g. [MCG03]) and precomputation (e.g. [WST09]).

On the other hand, fluid turbulence described as statistical fluctuation of velocities has been modeled in a noise-based way. Flow noise is proposed [PN01] to create fluid-like textures. Turbulent divergence-free fields are generated [BHN07] by applying the curl of a potential field. Divergence-free fields for artistic simulation are calculated by a fast simulation noise [PT05]. Beyond fluids, fractal mountains were created in the frequency domain according to fractal spectrum [Vos85], which can be applied to fluid turbulence. Forces have also been used in animated fluid control [FL04, SY05, TKPR06, BP08]. We also refer interested readers to a good textbook of fluid turbulence [Pop00].

3. Random Forcing

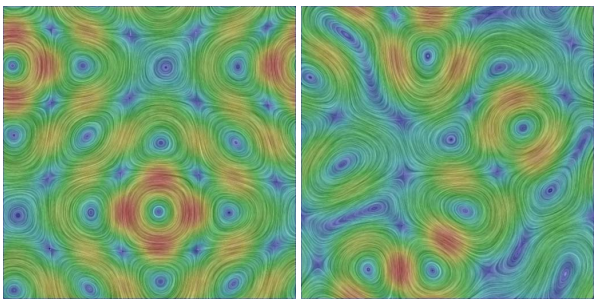
Turbulent flows, which are *unrepeatable* in details and *irregular* in both time and space, confound simple attempts to solve them in the ordinary NS equations. It leads to an extension of the understanding of fluid velocity as a random variable. Based on the Reynolds decomposition , the Reynolds-Average NS (RANS) equations for incompressible fluid are introduced [Pop00]: $\frac{\partial \mathbf{U}}{\partial t} + div(\mathbf{UU}) = -\nabla P + \nu\nabla^2\mathbf{U} - div(\overline{\mathbf{u'u'}})$, where the divergence ($div()$) of an additional Reynolds stress tensor, $\overline{\mathbf{u'u'}}$, describes the underlying stochastic turbulent agitation. As an *unknown* tensor containing the information about the effect of the subgrid scales on the mean flow, it is typically approximated by heuristic models (e.g. under Boussinesq's reasonable hypothesis treating turbulent stress like viscous stress). Although the models capture some of the chaotic nature of real turbulence in small-amplitude disturbances at resolved scales, the models are essentially deterministic. Hence, they miss the stochastic effect of random fluctuations at subgrid scales. More general attempts model the Reynolds stress effects by a random process that manifests as random forcing:

$$\frac{\partial \mathbf{U}}{\partial t} + div(\mathbf{UU}) = -\nabla P + \nu\nabla^2\mathbf{U} + \mathbf{f}. \tag{3}$$

The turbulence forcing term $\mathbf{f}$ is different and does not conflict with typical external forces (e.g. buoyancy). It nonetheless is a stochastic instrument to inject turbulent energy. Typically, it is considered as Gaussian random noises that are white in time [CD96], whose Fourier transform has the property: $\overline{\mathbf{f}(\mathbf{w},t)\mathbf{f}(\mathbf{w},\tau)} = E(\mathbf{w})\delta(t-\tau)$, where $\mathbf{w}$ is the wave number, t and τ are time steps. The overline denotes ensemble averaging, $\delta()$ is the Dirac function, and $E(\mathbf{w})$ represents input energy. Using Eqn. 3, a sequence of random $\mathbf{f}$ will naturally satisfy temporal coherence of the resultant turbulence. In this paper we apply synthetic force fields to drive the velocity fluctuation integrated with the mean flow in FNS(). However, we do not fully provide a physical solution of RANS. To make the turbulent animation following the large-scale flow, we control the mean flow input and force agitation with a special feedback scheme, which will be discussed in Sec. 5. Next, we first describe the generation of solenoidal $\mathbf{f}$ fields with spectral modeling.

4. Turbulence Synthesis

We create a divergence-free vector field, $\mathbf{v}$, completely in the Fourier domain by constructing random functions following



(a) $\mu = 2\sqrt{2}$, $\sigma = 0.2$ (b) $\mu = 2\sqrt{2}$, $\sigma = 0.5$

Figure 1: *Random vector fields generated for a preferred scale with different deviations.*

the frequency domain version of the divergence-free equation. After an inverse Fourier transform, the resultant field is strictly band limited with single or multiple vortex scales following a prescribed energy spectrum flexibly controlled by users. Its strict compliance with a spectrum design is mathematically guaranteed.

4.1. Frequency Domain Generation

The Fourier domain form of the divergence-free equation $div(\mathbf{v}) = 0$ is:

$$\mathbf{w} \cdot \hat{\mathbf{v}}(\mathbf{w}) = 0, \tag{4}$$

where $\hat{\mathbf{v}} = (\hat{v}_x, \hat{v}_y, \hat{v}_z)$ is the Fourier transform of $\mathbf{v}$, and $\mathbf{w} = (w_x, w_y, w_z)$ is the spatial frequency (wave number). We define the vector as

$$\hat{\mathbf{v}}(\mathbf{w}) = R_1(\mathbf{w})\mathbf{v_1}(\mathbf{w}) + R_2(\mathbf{w})\mathbf{v_2}(\mathbf{w}), \tag{5}$$

where $R_1(\mathbf{w})$ and $R_2(\mathbf{w})$ are two random complex numbers. Here two unit vectors $\mathbf{v_1}$ and $\mathbf{v_2}$ are orthogonal to $\mathbf{w}$, and also orthogonal to each other:

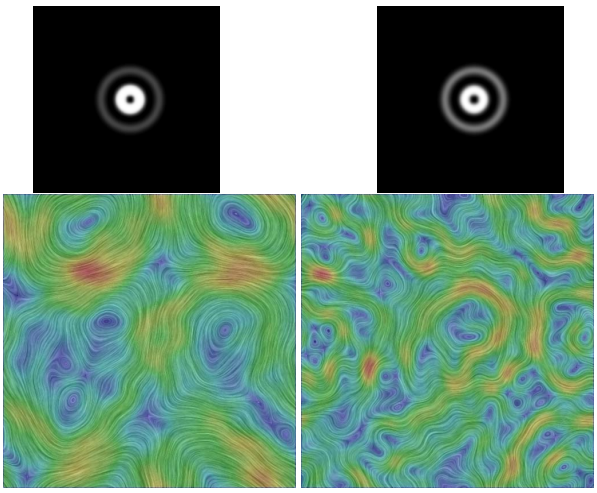
$$\mathbf{v_1}(\mathbf{w}) = \left(\frac{w_y}{\sqrt{w_x^2+w_y^2}}, -\frac{w_x}{\sqrt{w_x^2+w_y^2}}, 0\right), \tag{6}$$

$$\mathbf{v_2}(\mathbf{w}) = \left(\frac{w_x w_z}{|\mathbf{w}|\sqrt{w_x^2+w_y^2}}, \frac{w_y w_z}{|\mathbf{w}|\sqrt{w_x^2+w_y^2}}, -\frac{\sqrt{w_x^2+w_y^2}}{|\mathbf{w}|}\right),$$

where $|\mathbf{w}|$ is the magnitude of vector $\mathbf{w}$ [Alv99]. The two random numbers are generated as

$$\begin{aligned} R_1(\mathbf{w}) &= S_{\mathbf{w}} \cdot e^{i\alpha_1} sin\beta, \\ R_2(\mathbf{w}) &= S_{\mathbf{w}} \cdot e^{i\alpha_2} cos\beta, \end{aligned} \tag{7}$$

where $S_{\mathbf{w}}$ is a spectrum controlling parameter at frequency $\mathbf{w}$. We utilize three scalar random numbers $\alpha_1, \alpha_2, \beta \in [0, 2\pi]$. This solenoidal field generation strategy, based on the Fourier domain orthogonal projection, has been widely used in physics, as well as by Stam [SF93, Sta97]. The method was also applied to create 3D Kolmogorov spectrum fields which are added to 2D simulations for large-scale smoke phenomena [RNGF03]. Our method generates small-scale



(a) Kolmogorov energy. (b) Arbritrary energy.

Figure 2: *Divergence-free vector fields with two scales. Top: Spectrum; Bottom: Vector field.* $\mu_1 = \sqrt{2}$, $\mu_2 = 8$ *and* $\sigma_1 = \sigma_2 = 0.7$.

force fields in a similar way. As described in Sec. 3, we are able to supply turbulent randomness that is white in time, i.e. not necessary to strictly respect temporal continuity and smoothness, which will be implicitly satisfied by forcing in FNS(), so that we no longer need to model the 4D Fourier field as Stam did. This also gives us freedom to explicitly model intermittency (see Sec. 6). Next, we show how to control energy input in spectral bands.

4.2. Energy Spectrum Control

The parameter $S_{\mathbf{w}}$ is related to the energy input in a particular frequency $\mathbf{w}$, which is used to control the total kinetic energy of the resultant vector field, $\frac{1}{2}\langle \mathbf{v}^2 \rangle$. Here, $\langle\ \rangle$ represents statistically averaging over the domain. The kinetic energy can be computed in the Fourier domain by integrating $\frac{1}{2}\langle \hat{\mathbf{v}}\hat{\mathbf{v}}^* \rangle$ in the whole domain Ω, where $*$ denotes complex conjugate. This computation is achieved by integrating on each spherical area, Λ, with a radius $|\mathbf{w}|$: $\frac{1}{2}\langle \hat{\mathbf{v}}\hat{\mathbf{v}}^* \rangle = \frac{1}{2}\int_\Omega \hat{\mathbf{v}}\hat{\mathbf{v}}^* d\Omega = \frac{1}{2}\int_0^{+\infty}(\oint_\Lambda \hat{\mathbf{v}}\hat{\mathbf{v}}^* d\Lambda)d|\mathbf{w}| = \frac{1}{2}\int_0^{+\infty} 4\pi|\mathbf{w}|^2 \hat{\mathbf{v}}\hat{\mathbf{v}}^* d|\mathbf{w}|$. An energy input $E_{\mathbf{w}}$ is thus computed at each $|\mathbf{w}|$ as $E_{\mathbf{w}} = 4\pi|\mathbf{w}|^2\hat{\mathbf{v}}\hat{\mathbf{v}}^*$, which determines the total kinetic energy of the vector field by $\frac{1}{2}\langle \mathbf{v}^2 \rangle = \int_0^{+\infty} E_{\mathbf{w}} d|\mathbf{w}|$. From Eqns. 5, 6 and 7, we get $\hat{\mathbf{v}}\hat{\mathbf{v}}^* = R_1 \cdot R_1^* + R_2 \cdot R_2^* = S^2(sin\beta^2 + cos\beta^2) = S^2$. We thus define $S_{\mathbf{w}}$ by

$$S_{\mathbf{w}}^2 = \frac{E_{\mathbf{w}}}{4\pi|\mathbf{w}|^2}, \tag{8}$$

where $E_{\mathbf{w}}$ is a controllable input for the resultant fields.

Single Scale To provide more flexibility, we generate a single-scale field by

$$E_{\mathbf{w}} = C_{\mathbf{w}} e^{-\frac{(|\mathbf{w}|-\mu)^2}{2\sigma^2}}. \tag{9}$$

The Gaussian function defines an energy spectrum with concentration at frequencies that have a magnitude μ and a corresponding deviation σ determining the degree of concentration. For a field size N, a given magnitude μ approximately models a 3D vortex scale $l = \frac{(N/2)\sqrt{3}}{\mu}$, where $\sqrt{3}$ is the diagonal factor, and $N/2$ comes from the conjugate symmetric implementation in the Fourier domain for achieving inverse transform results as real (non-complex) vectors. Fig. 1a shows 2D results using $\mu = 2\sqrt{2}$ and $\sigma = 0.2$. The nearly regular vortex size and energy distribution are due to the small $\sigma = 0.2$ which plays a significant role in vortex appearance. In Fig. 1b, the variation is made significant when $\sigma = 0.5$, due to a loose concentration. The major energy input (i.e. large velocity magnitude visualized by red/yellow colors) focuses on the vortices with the predefined scale μ. This example illustrates using the Gaussian function to flexibly control the vortex scale and energy distribution, with 2D visualization used for clearer representation and better understanding. However the method works equally well in 3D cases.

Multiple Scales A multiple-scale field, with two concentrations as an example here, is computed by

$$E_{\mathbf{w}} = C1_{\mathbf{w}} e^{-\frac{(|\mathbf{w}|-\mu_1)^2}{2\sigma_1^2}} + C2_{\mathbf{w}} e^{-\frac{(|\mathbf{w}|-\mu_2)^2}{2\sigma_2^2}}. \qquad (10)$$

Fig. 2 shows blended rotational behaviors from the two scales, where large-scale (μ_1) vortices are agitated by the small scale (μ_2). Following K41 that suggests small-scale vortices holding decreasing kinetic energy with the $-\frac{5}{3}$ law, we define $C1_{\mathbf{w}}, C2_{\mathbf{w}} \propto |\mathbf{w}|^{-\frac{5}{3}}$ (Fig. 2a). In comparison, we also use an arbitrary $C1_{\mathbf{w}} = C2_{\mathbf{w}}$ not obeying this physical rule (Fig. 2b). Consequently, it shows more small-scale turbulence than Fig. 2a (see the brighter $\mu_2 = 8$ spectrum ring in Fig. 2b compared to Fig. 2a). Note that the $-\frac{5}{3}$ law in K41 describes a continuous decay in inertial subrange. Here we use the relation between two discrete scales (within a Gaussian kernel range). Though not physically accurate, it leads to easy and meaningful control of chaotic fluid behavior.

4.3. Computation

A sequence of force fields is independently pre-computed with the spectral method. This separation from a mean flow simulation makes it flexible to control and design turbulent effects in a post-processing stage. This differs from the previous methods where they particularly create $\mathbf{u}'$ from $\mathbf{U}$ at each step. Due to the force integration, our method does not need to generate the force field at each step. In fact, the generated fields can be reused in a simulation. In our experiments, only 25 pre-computed force fields are randomly chosen, leading to good turbulent results. Furthermore, the same sequence of the fields can be repeatedly used in different simulations with different incoming flow fields. Since the Fourier domain operations are trivial, the computational complexity is completely bounded by the inverse Fourier transform. Though fast computing is not demanded as a precomputation, the generation still can be completed very fast with Fast Fourier Transform in O(nlogn) and with GPU acceleration. For example, it costs 450ms and 275ms for a 128^3 and 64^3 grid on an nVidia 8800 GT GPU, respectively.

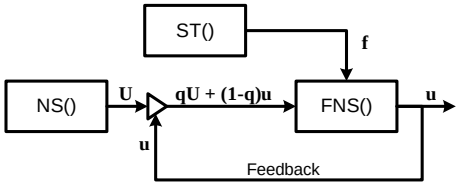


Figure 3: *Data flow of FNS() computation.*

5. Turbulence Integration

Integration Scheme To integrate pre-computed $\mathbf{f}$ to an existing flow, FNS() executes operations at each step as:

1. Load velocity field from an existing mean flow simulation NS() at critical local regions of the whole simulation domain;
2. Apply linear interpolation to generate the high-resolution velocity field $\mathbf{U}$ from the mean flow input;
3. Create initial condition of FNS() as $q\mathbf{U} + (1-q)\mathbf{u}(t)$, where $\mathbf{u}(t)$ is the instantaneous velocity field from last simulation step;
4. Run FNS() fluid solver for one step, with the force coupling from a randomly-selected field $\mathbf{f}$;
5. Use the resultant high-resolution flow field $\mathbf{u}(t+1)$ for density advection and rendering;
6. Goto 1.

Figure 3 shows the data flow of the FNS() computation. At each step, the initial velocity field consists of two components: one is the output instantaneous field ($\mathbf{u}$) from last simulation step, and another one ($\mathbf{U}$) is acquired from the mean flow by spatial interpolation. The two components are added by $q\mathbf{U} + (1-q)\mathbf{u}$ with a control parameter q, and then modified by solving the NS equations with the infusion from a solenoidal force field $\mathbf{f}$. A large q enforces the resultant instantaneous flow strictly regulated by $\mathbf{U}$. On the contrary, a smaller q will make the turbulence become much significant diverging from $\mathbf{U}$. Our scheme can be looked as enabling a feedback control so that the integration provides natural coupling and control flexibility. In comparison, the previous methods directly coupling synthesized noise with $\mathbf{U}$ act as a feed-forward control that requires special handling of ST() and $\oplus$ as we discussed before. Finally, the resultant velocity field contributes in its corresponding regions for fluid rendering.

Note that the interpolation might not be needed if an animator plans to add turbulence to the existing flow without using a high-resolution grid. In this case, $\mathbf{f}$ is directly applied for stimulating synthetic turbulence in existing octaves of $\mathbf{U}$.

Besides the Eulerian solver, our method can also be directly applied to Lagrangian fluid solvers. We conduct an experiment with the state-of-the-art SPH methods (Smoothed Particle Hydrodynamics), which have been widely explored in computer graphics due to its programming simplicity, various simulation scales and easy boundary handling [MCG03]. Together with the typical pressure and viscosity forces, we impose $\mathbf{f}$ to each particle, manifesting the unsolved turbulent behavior. The force coupling strategy has no difference from Eulerian approaches, which is described below. To the best of our knowledge, this is the first time in graphics applying turbulence enhancement to a pure Lagrangian solver (Note that though with introduced vortex particles, [PTSG09] still relies mainly on an Eulerian solver).

Force Coupling A stochastic force $\mathbf{f}$ should perform in the turbulence simulator with respect to the mean flow properties. We match the force $\mathbf{f}$ with $\mathbf{U}$ at each location, using the graphical assumption $\mathbf{B}$ as in previous works (see Sec. 1). At first, we make $\mathbf{f}\cdot\mathbf{U}\leq 0$ by reversing the direction of $\mathbf{f}$ if needed. This guarantees the randomly created turbulent fluctuation will not reverse $\mathbf{U}$ dramatically that leads to unnatural flow variation effects. Second, we follow the magnitude relation $|\mathbf{f}|\propto|\mathbf{u_f}|/\delta t$ [OP98], where δt is the turbulence simulation time step length and $\mathbf{u_f}$ is the introduced velocity variation by $\mathbf{f}$. Finally an amplitude relation between $\mathbf{u_f}$ and $\mathbf{U}$ should be determined: $|\mathbf{u_f}|=p_c|\mathbf{U}|$, where p_c is a coupling parameter. Then we achieve

$$|\mathbf{f}|=|\mathbf{u_f}|/\delta t=p_c|\mathbf{U}|/\delta t. \tag{11}$$

p_c can be found by applying the velocity cascade relation among octaves of $\mathbf{u_f}$ and $\mathbf{U}$ [KTJG08]. This approach is feasible but increases complexity. It indeed may not be necessary due to the approximation already produced by the graphical assumptions. p_c can be more conveniently defined as an empirical control parameter. In general, if a function is defined as $|\mathbf{f}|=\Psi(\mathbf{U})$, Ψ provides a very good tool to control the turbulence integration, and hence the final fluid effects. Besides Eqn. 11, we describe flexible approaches of Ψ in Sec. 6.

6. Conditional and Intermittent Turbulence

Forcing acts as a *stimulus* for inducing chaotic effects rather than adding a *resultant* turbulent field to global flow, leading to easy implementation of (1) adaptive turbulence only in necessary spatial areas and temporal periods; and (2) conditional turbulent effects with physical or artificial conditions. We apply turbulence forcing within critical areas in a large domain running global simulation. The turbulent effects will propagate out of the selected local regions through the motion of scalar densities. This approach is very useful for many applications such as interactive games and emergency training.

Conditional Coupling As discussed in Sec. 5, a function Ψ is to determine the integration conditions of turbulence. We have defined Eqn. 11, which couples turbulence in the whole effective region based on velocity magnitude. Here we provide several other choices for different animation purposes:

- **Strain rate**: At each location $\mathbf{r}$, the local strain rate $S(\mathbf{U})^2=\sum\sum((\partial\mathbf{U}_i/\partial\mathbf{r}_j+\partial\mathbf{U}_j/\partial\mathbf{r}_i)/2)^2$. We define
$$|\mathbf{f}|=\Psi(\mathbf{U})=p_c|\mathbf{w_1}|^{-1}S(\mathbf{U})/\delta t, \tag{12}$$
so that turbulence is initiated at locations with a large rate of change in $\mathbf{U}$.
- **Distance**: Obstacles are prone to introduce high strain rate thus causing boundary-induced turbulence. While small obstacles may not be fully accounted for in a coarse grid simulation of $\mathbf{U}$, a fluid animation can define
$$|\mathbf{f}|=\Psi(\mathbf{U})=p_c Ramp(\frac{D(\mathbf{r})}{D_0})|\mathbf{U}|/\delta t, \tag{13}$$
where $D(\mathbf{r})$ is the shortest distance from $\mathbf{r}$ to the obstacles and D_0 is a cutting length. $Ramp()$ defines a smoothly decreasing function from one to zero for $D(\mathbf{r})<D_0$, and otherwise it equals zero. Here we link $D(\mathbf{r})$ to boundary layer effects, based on an observation that the profile of shear stress, which leads to turbulence, is a decreasing curve of the distance to the boundary surface [Pop00].
- **Vorticity**: Similar to strain rate, turbulence is related to the vorticity by
$$|\mathbf{f}|=\Psi(\mathbf{U})=p_c\frac{|\omega|}{max(|\omega|)}H(|\omega|-|\omega|_0)/\delta t, \tag{14}$$
where the vorticity $\omega=\nabla\times\mathbf{U}$, $H()$ is the Heaviside step function, and $|\omega|_0$ is a threshold used to control the effects together with p_c.
- **Density**: Turbulence can be triggered by a function of the scalar density m of fluid. A simple formula is
$$|\mathbf{f}|=\Psi(\mathbf{U})=p_c\frac{m}{max(m)}H(m-m_0)|\mathbf{U}|/\delta t, \tag{15}$$
where $H()$ is the Heaviside function and m_0 is a threshold.

These examples illustrate that our solution supplies a framework incorporating a variety of turbulence starters, from physical features to an animator's discretion, which can be further improved and extended for controllable and interactive animations.

Intermittency Turbulent fluids show alternations in time between nearly non-turbulent and chaotic behavior, which challenges K41's hypothesis of universality. Many attempts, including by Kolmogorov himself, have been proposed to explain and solve the problem. It is extremely hard to present intermittency physically by DNS. We instead introduce temporal control in forcing integration to animate intermittent fluids. The fluid behaves in non-turbulent or turbulent dynamics alternately with *randomly varied* time intervals, Δt_{turb} and Δt_{non}, respectively. We initiate turbulence coupling in intervals of Δt_{turb}, and otherwise use the large-scale flow only. The two intervals are computed, when each

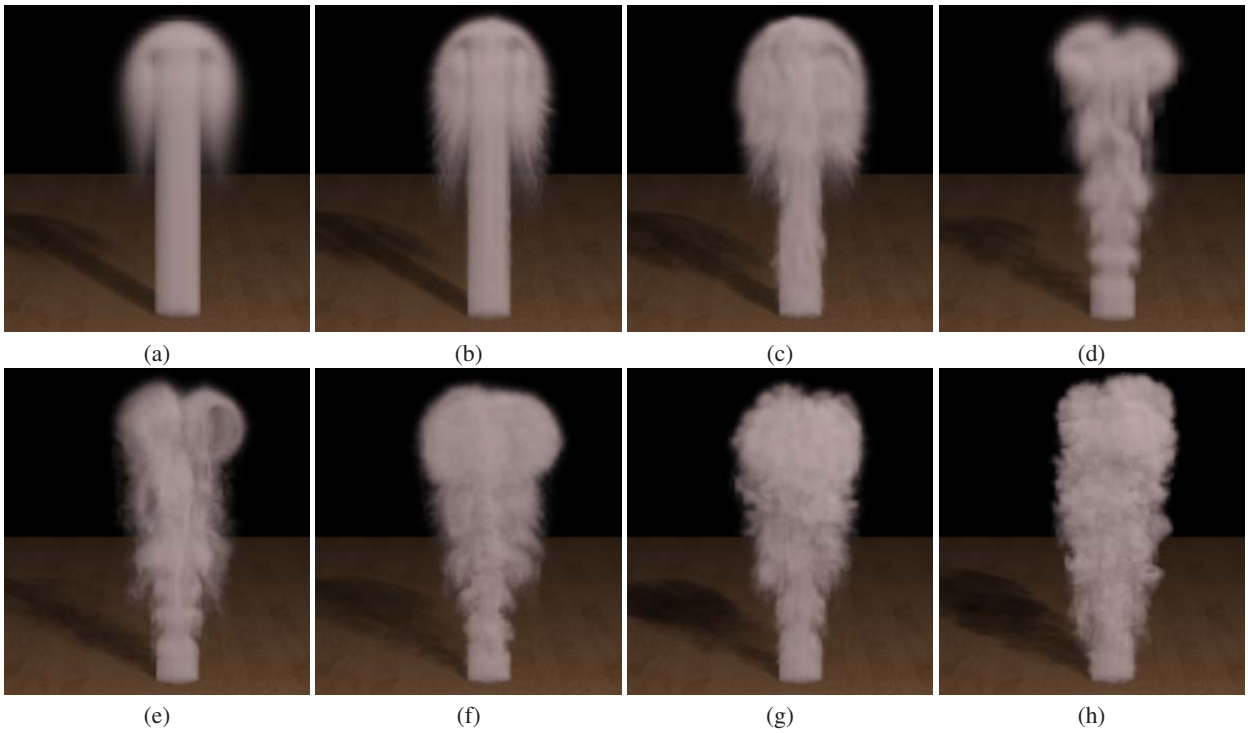


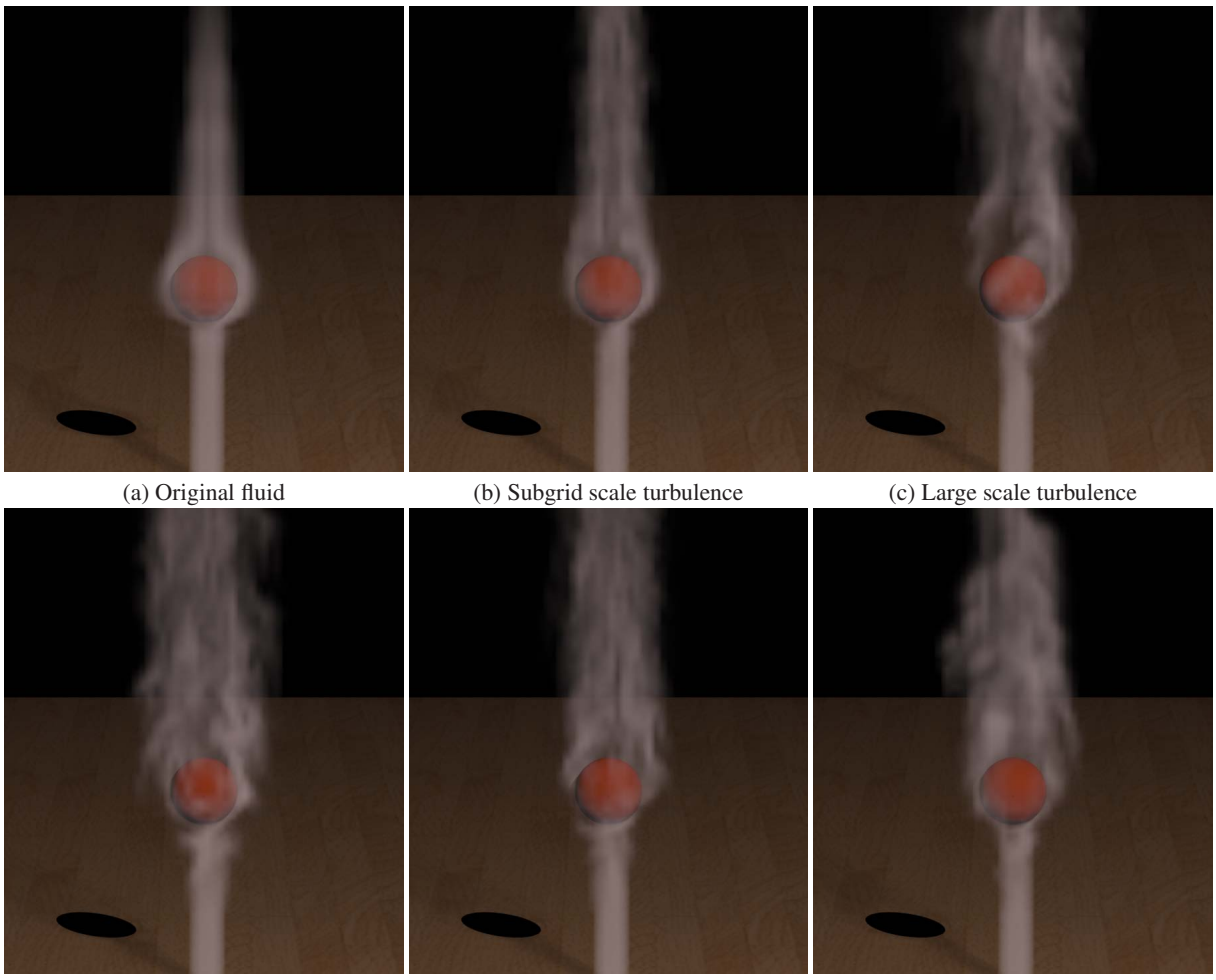
Figure 4: *Snapshots of turbulence enhancement simulations: (a) Original coarse simulation; (b) Wavelet subgrid turbulence; (c) Our subgrid turbulence; (d) Add vorticity confinement to (a); (e) Wavelet turbulence to (d); (f) Our turbulence to (d) with $q = 0.8$; (g) Our turbulence to (d) with $q = 0.2$; (h) Our turbulence to (d) with $q = 0.1$.*

time needed, as scalar random values by $\Delta t_{turb} \in [0, L_{turb}]$ and $\Delta t_{non} \in [0, L_{non}]$, where L_{turb} and L_{non} are used to control the maximum interval length in time steps. However, when sometimes $\Delta t_{non} = 0$, two turbulent periods are concatenated. For the whole animation period, the intermittency factor is $\gamma = \Sigma(\Delta t_{turb})/\Sigma(\Delta t_{turb} + \Delta t_{non})$.

7. Experiments

Exp. 1 First, we describe our method in Fig. 4 in comparison to the successful wavelet turbulence enhancement method [KTJG08]. A basic stable solver [Sta99] generates very static smoke effects with a coarse $48 \times 64 \times 48$ simulation (Fig. 4a). Wavelet turbulence adds subgrid turbulence with a 2x finer grid (Fig. 4b). Our method creates similar subgrid turbulence in Fig. 4c with $\mu = 32\sqrt{3}$ and $p_c = 0.5$. Our solution looks relatively more realistic since the forcing can impose chaotic behavior even for this very regular mean flow, while the wavelet approach's effect is contingent on the original $\mathbf{U}$. We then add the vorticity confinement [FSJ01] to Fig. 4a producing large-scale rotational behavior (Fig. 4d). In this case, the wavelet turbulence provides good small-scale turbulence (Fig. 4e). In comparison, our method can introduce small-scale turbulence in different levels with the control parameter q (see Fig. 3). In Fig. 4f, with $q = 0.8$ mainly supplying the mean flow component to FNS(), the enhanced smoke propagates close to the origin shape of Fig. 4d which is similar to the wavelet result. While we decrease $q = 0.2$, turbulence becomes significant in Fig. 4g since more enhanced component feedbacks to FNS() inducing amplification effects. In Fig. 4h, $q = 0.1$ leads to even stronger turbulent effects. On a PC CPU (Intel Core2 6300 1.86GHz 4GB), our turbulent enhancement runs in 12982 ms per step, and the wavelet method uses 4048 ms per step, respectively. We compare these animations side by side in the supplemental movie to illustrate the dynamic difference.

Exp. 2 Next, we execute our animation using $q = 0.2$ based on a simulated laminar smoke past a sphere with a very coarse grid at $16 \times 32 \times 16$. To better illustrate our approach, no enhancing techniques (e.g. vorticity confinement) are employed. We use a smoke evolving grid with a resolution 4x denser at $64 \times 128 \times 64$. Turbulence integration is implemented on a local region surrounding the sphere with a resolution $38 \times 76 \times 38$. Fig. 5 shows snapshots of the integrated turbulence at different scales: (a) original laminar result; (b) small turbulent variation with a subgrid scale ($\mu_{sub} = 16\sqrt{3}$) that approximates the grid scaling factor 4 ($p_c = 0.3$); (c) strong turbulent dynamics with a larger scale ($\mu_l = \frac{1}{3}\mu_{sub}$, $p_c = 0.5$); (d) turbulent behavior accommodating finer details than (c), with two coalesced scales (μ_l and μ_{sub}) following $-\frac{5}{3}$ law ($p_c = 0.5$); (e) reproducing dynamics of (d) with an arbitrary spectrum law where two octaves



(a) Original fluid (b) Subgrid scale turbulence (c) Large scale turbulence

(d) Multiple scale turbulence (K41) (e) Multiple scale turbulence (Arbitrary) (f) Density based turbulence

Figure 5: *Snapshots of integrating turbulence to a laminar smoke.*

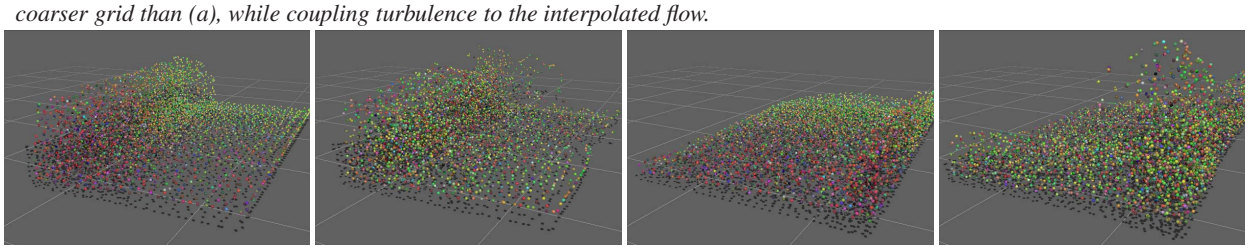
(μ_l and μ_{sub}) having an equalized energy spectrum (non-Kolmogorov), which further reduces the effects of the large scale one. Fig. 5b-e use Eqn. 11 for force integration. Finally in Fig. 5f smoke density (Eqn. 15) is used to trigger turbulence. We use $\sigma = 0.5$ for the simulations. In the supplemental movie, we compare the smoke effects in the different configurations. We also include multiscale animations using the vorticity (Eqn. 14) and strain rate (Eqn. 12) based turbulence integration, where chaotic variations appear around the sphere. On the PC CPU, the experiment uses 48 ms per step for the global simulation and 956 ms per step for the interpolation, integration and forced simulation. The density advection costs 342 ms. In comparison, a direct $64 \times 128 \times 64$ simulation consumes 8715 ms that is 6.5 times slower, which cannot generate the various turbulent results.

Exp. 3 Another experiment is applied with three obstacles using $q = 0.2$, as shown in Fig. 6. Here we utilize the distance based turbulence enhancement condition (Eqn. 13) to approximate boundary induced chaos. Fig. 6a is the original simulation result with a resolution $64 \times 32 \times 50$. Fig. 6b presents a turbulent flow by adding turbulence forces to Fig. 6a without any interpolation or subgrid time steps. We use $\mu = 12\sqrt{3}$, $\sigma = 0.7$ and $p_c = 0.5$. The $64 \times 32 \times 50$ simulation (with or without turbulence) runs in 628 ms per frame on the PC CPU and the added force does not increase noticeable computing overload. In Fig. 6c, we instead run the global simulation on a 2x coarser grid at $32 \times 16 \times 25$ (51 ms per frame), and apply turbulence integration on an interpolated grid at $64 \times 32 \times 50$. It shows finer turbulent features compared with Fig. 6b using a larger $\mu = 16\sqrt{3}$. With this configuration, we also include intermittent turbulent effects in the supplemental movie using $L_{turb} = 20$ and $L_{non} = 40$ steps.

Exp. 4 We also perform SPH based turbulence enhancement. We use 4096 particles to perform a wave simulation. We follow [MCG03] for implementation details. Fig. 7 shows the

(a) (b) (c)

Figure 6: *Snapshots of turbulence enhancement conditioned by the distance to obstacles: (a) a laminar smoke simulation; (b) direct turbulence integration to (a) at the same resolution; (c) Finer turbulent behavior achieved by executing simulation on a coarser grid than (a), while coupling turbulence to the interpolated flow.*



(a) Origin (b) Turbulent (c) Origin (d) Turbulent

Figure 7: *Snapshots of turbulence enhancement with SPH ((b) and (d)), in comparison with original simulation ((a) and (c)).*

comparison of two snapshots between the turbulent and original animations. Each particle acquire the turbulence force from a 128^3 grid with $\mu = 20$ and $p_c = 1$. Realtime simulation (52 frames per second) is achieved since our enhancement only need minimal extra computation of force addition. However, it shows good turbulent dynamics in Fig. 7b and Fig. 7d, comparing with the original results in Fig. 7a and Fig. 7c, respectively.

Discussion In the experiments, our method provides good turbulent behavior even with very coarse simulations, showing that it can be a fast and convenient tool for creating turbulent animations. The simulation performance is totally dependent on the FNS() solvers, which is more complex than using a simple vector field addition of small and large scales. However, in comparison to previous methods, this integration approach originates from the RANS equation and provides natural and controllable coupling for turbulent effects. Furthermore, in most cases FNS() can be computed only on local and adaptive regions. Our method is independent of any numerical solver, where high-order and Lagrangian solvers can be employed. It also makes it possible to utilize pre-computed and reusable synthetic fields separated from original simulations. Moreover, boundaries are easily handled with no special handling needed in noise synthesis.

8. Conclusion

By utilizing random forcing in turbulence integration, we are able to enhance an existing fluid simulation with controllable turbulence. The effects can be designed by applying (1) subgrid and/or large scale fluctuation; (2) prescribed energy spectrum; (3) local effective region; (4) intermittent temporal periods; (5) physical or artificial inducing conditions. It will lead to a variety of future works to provide a powerful and interactive tool for fluid animators. For example, new integration conditions based on geometric features or visibility will be designed for more interesting fluid behaviors. We will also apply random forcing from wavelet noise or other synthetic approaches.

Acknowledgments

This work is partially supported by the US National Science Foundation under grant IIS-0916131. We would like to thank the anonymous reviewers for their valuable comments, Paul Farrel for proofreading, and NVidia for their donation of hardware to support our research. We also thank Theodore Kim and Nils Thuerey for their publicly released source code of wavelet turbulence, and Rama Hoetzlein for his SPH open source code - Fluids v.1.

References

[Alv99] ALVELIUS K.: Random forcing of three-dimensional homogeneous turbulence. *Physics of Fluids 11*, 7 (1999), 1880–1889.

[BHN07] BRIDSON R., HOURIHAN J., NORDENSTAM M.: Curl-noise for procedural fluid flow. In *Proceeding of ACM SIGGRAPH* (New York, NY, USA, 2007), ACM, p. 46.

[BP08] BARBIČ J., POPOVIĆ J.: Real-time control of physically based simulations using gentle forces. *ACM Trans. Graph. 27*, 5 (2008), 1–10.

[Bri08] BRIDSON R.: *Fluid Simulation for Computer Graphics*. A. K. Peters, Ltd., Natick, MA, USA, 2008.

[CD96] CANUTO V., DUBOVIKOV M.: A dynamical model for turbulence. i. general formalism. *Physics of Fluids 8*, 2 (1996).

[ETK*07] ELCOTT S., TONG Y., KANSO E., SCHRÖDER P., DESBRUN M.: Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1 (2007), 4.

[FL04] FATTAL R., LISCHINSKI D.: Target-driven smoke animation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 441–448.

[Fri95] FRISCH U.: *Turbulence: The legacy of A.N. Kolmogorov*. Cambridge University Press, 1995.

[FSJ01] FEDKIW R., STAM J., JENSEN H.: Visual simulation of smoke. *Proceedings of SIGGRAPH* (2001), 15–22.

[KLLR07] KIM B., LIU Y., LLAMAS I., ROSSIGNAC J.: Advections with significantly reduced dissipation and diffusion. *IEEE Transactions on Visualization and Computer Graphics 13*, 1 (2007), 135–144.

[KTJG08] KIM T., THÜREY N., JAMES D., GROSS M.: Wavelet turbulence for fluid simulation. In *Proceeding of ACM SIGGRAPH* (New York, NY, USA, 2008), ACM, pp. 1–6.

[LDM03] LAVAL J., DUBRULLE B., MCWILLIAMS J. C.: Langevin models of turbulence: Renormalization group, distant interaction algorithms or rapid distortion theory? *Physics Of Fluids 15*, 5 (2003).

[LGF04] LOSASSO F., GIBOU F., FEDKIW R.: Simulating water and smoke with an octree data structure. *ACM Trans. Graph. 23*, 3 (2004), 457–462.

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2003), Eurographics Association, pp. 154–159.

[MCP*09] MULLEN P., CRANE K., PAVLOV D., TONG Y., DESBRUN M.: Energy-preserving integrators for fluid animation. *ACM Trans. Graph. 28*, 3 (2009).

[NSCL08] NARAIN R., SEWALL J., CARLSON M., LIN M. C.: Fast animation of turbulence using energy transport and procedural synthesis. In *Proceeding of ACM SIGGRAPH Asia* (New York, NY, USA, 2008), ACM, pp. 1–8.

[OP98] OVERHOLT M. R., POPE S. B.: A deterministic forcing scheme for direct numerical simulations of turbulence. *Comput. Fluids 27*, 1 (1998), 11–28.

[PN01] PERLIN K., NEYRET F.: Flow noise. *ACM SIGGRAPH Technical Sketches and Applications* (2001), 187.

[Pop00] POPE S. B.: *Turbulent Flows*. Cambridge University Press, 2000.

[PT05] PATEL M., TAYLOR N.: Simple divergence-free fields for artistic simulation. *Journal of Graphics Tools 10*, 4 (2005), 49–60.

[PTSG09] PFAFF T., THUEREY N., SELLE A., GROSS M.: Synthetic turbulence using artificial boundary layers. In *SIGGRAPH Asia '09: ACM SIGGRAPH Asia 2009 papers* (New York, NY, USA, 2009), ACM, pp. 1–10.

[RNGF03] RASMUSSEN N., NGUYEN D. Q., GEIGER W., FEDKIW R.: Smoke simulation for large scale phenomena. *ACM Trans. Graph. 22*, 3 (2003), 703–707.

[SB08] SCHECHTER H., BRIDSON R.: Evolving sub-grid turbulence for smoke animation. In *Eurographics/ACM SIGGRAPH Symposium on Computer Animation* (2008), pp. 1–8.

[SF93] STAM J., FIUME E.: Turbulent wind fields for gaseous phenomena. In *Proceeding of ACM SIGGRAPH* (New York, NY, USA, 1993), ACM, pp. 369–376.

[SRF05] SELLE A., RASMUSSEN N., FEDKIW R.: A vortex particle method for smoke, water and explosions. *Proceedings of SIGGRAPH* (2005), 910–914.

[Sta97] STAM J.: A general animation framework for gaseous phenomena. *ERCIM Research Report R047* (1997).

[Sta99] STAM J.: Stable fluids. In *Proceeding of ACM SIGGRAPH* (New York, NY, USA, 1999), ACM Press, pp. 121–128.

[SY05] SHI L., YU Y.: Controllable smoke animation with guiding objects. *ACM Trans. Graph. 24*, 1 (2005), 140–164.

[TKPR06] THÜREY N., KEISER R., PAULY M., RÜDE U.: Detail-preserving fluid control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2006), pp. 7–12.

[Vos85] VOSS R. F.: Random fractal forgeries. In *Fundamental Algorithms in Computer Graphics*, Earnshaw R., (Ed.). Springer, 1985, pp. 805–883.

[WST09] WICKE M., STANTON M., TREUILLE A.: Modular bases for fluid dynamics. *ACM Trans. Graph. 28*, 3 (2009), 1–8.

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. In *Proceedings of ACM SIGGRAPH* (New York, NY, USA, 2005), ACM, pp. 965–972.