Restoring the Missing Vorticity in Advection-Projection Fluid Solvers

Xinxin Zhang*
UBC Computer Science

Robert Bridson†
UBC Computer Science, Autodesk Canada

Chen Greif‡
UBC Computer Science

Figure 1: *Rising smoke simulations with and without IVOCK (Integrated Vorticity of Convective Kinematics). From left to right: Stable Fluids, Stable Fluids with IVOCK; BFECC, BFECC with IVOCK; MacCormack, MacCormack with IVOCK; FLIP, FLIP with IVOCK.*

Abstract

Most visual effects fluid solvers use a time-splitting approach where velocity is first advected in the flow, then projected to be incompressible with pressure. Even if a highly accurate advection scheme is used, the self-advection step typically transfers some kinetic energy from divergence-free modes into divergent modes, which are then projected out by pressure, losing energy noticeably for large time steps. Instead of taking smaller time steps or using significantly more complex time integration, we propose a new scheme called IVOCK (Integrated Vorticity of Convective Kinematics) which cheaply captures much of what is lost in self-advection by identifying it as a violation of the vorticity equation. We measure vorticity on the grid before and after advection, taking into account vortex stretching, and use a cheap multigrid V-cycle approximation to a vector potential whose curl will correct the vorticity error. IVOCK works independently of the advection scheme (we present examples with various semi-Lagrangian methods and FLIP), works independently of how boundary conditions are applied (it just corrects error in advection, leaving pressure etc. to take care of boundaries and other forces), and other solver parameters (we provide smoke, fire, and water examples). For $10 \sim 25\%$ extra computation time per step much larger steps can be used, while producing detailed vorticial structures and convincing turbulence that are lost without correction.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

Keywords: fluid simulation, vorticity, advection

*e-mail:zhxx@cs.ubc.ca
†e-mail:rbridson@cs.ubc.ca
‡e-mail:greif@cs.ubc.ca

Table 1: *Algorithm abbreviations used through out this paper.*

IVOCK	The computational routine (Alg.1) correcting vorticity for advection
SF	Classic Stable Fluids advection [Stam 1999]
SF-IVOCK	IVOCK with SF advection
SL3	Semi-Lagrangian with RK3 path tracing and clamped cubic interpolation
BFECC	Kim et al.'s scheme [2005], with extrema clamping([Selle et al. 2008])
BFECC-IVOCK	IVOCK with BFECC advection
MC	Selle et al.'s MacCormack method [2008]
MC-IVOCK	IVOCK with MacCormack
FLIP	Zhu and Bridson's incompressible variant of FLIP [2005]
FLIP-IVOCK	FLIP advection of velocity and density, SL3 for vorticity in IVOCK.

1 Introduction

In computer graphics, incompressible fluid dynamics are often solved with fluid variables stored on a fixed Cartesian grid, known as an Eulerian approach [Stam 1999]. The advantages of pressure projection on a regular grid and the ease of treating smooth quantities undergoing large deformations help explain its success in a wide range of phenomena, such as liquids [Foster and Fedkiw 2001], smoke [Fedkiw et al. 2001] and fire [Nguyen et al. 2002]. Bridson's text provides a good background [2008].

In Eulerian simulations, the fluid state is often solved with a time splitting method: given the incompressible Euler equation,

$$\begin{gathered}\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \boldsymbol{f} \\ \nabla \cdot \boldsymbol{u} = 0\end{gathered} \tag{1}$$

fluid states are advanced by self-advection and incompressible projection. First one solves the advection equation

$$\frac{D\boldsymbol{u}}{Dt} = 0 \tag{2}$$

to obtain an intermediate velocity field $\tilde{\boldsymbol{u}}$, which is then projected to be divergence-free by subtracting ∇p, derived by a Poisson solve from $\tilde{\boldsymbol{u}}$ itself. We formally denote this as $\boldsymbol{u}^{n+1} = \mathbf{Proj}\,(\tilde{\boldsymbol{u}})$.

Self-advection disregards the divergence-free constraint, allowing some of the kinetic energy of the flow to be transferred into diver-

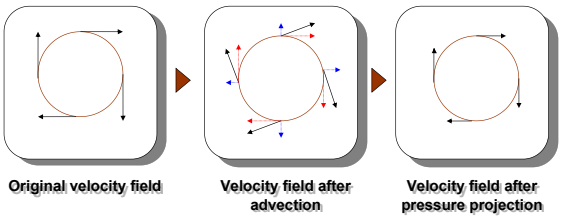

Figure 2: *Self-advection maps the original velocity field into a rotational part and a divergent part, indicated by red and blue arrows respectively. Pressure projection removes the blue arrows, leaving the rotational part, and illustrating how angular momentum has already been lost.*

gent modes which are lost in pressure projection. We focus in particular on rotational motion: self-advection can sometimes cause a noticeable violation of conservation of angular momentum, as illustrated in Fig.2.

Despite many published solutions in improving the accuracy of advection scheme in an Eulerian framework (e.g. [Selle et al. 2008; Lentine et al. 2011]), or reducing the numerical diffusion with hybrid particle-in-cell solvers [Zhu and Bridson 2005], even with higher-order integration [Edwards and Bridson 2014], only a few papers have addressed this particular type of numerical dissipation.

Retaining the velocity-pressure approach but changing the time integration to a symplectic scheme, Mullen et al. [2009] were able to discretely conserve energy when simulating fluids. However, the expense of their non-linear solver for Crank-Nicolson-style integration, and the potential for spurious oscillations arising from non-upwinded advection, raise questions about its practicality in general.

Another branch of fluid solvers take the vorticity-velocity form of the Navier-Stokes equation, exploiting the fact that the velocity field induced from vorticity is always divergence free. These solvers advect the vorticity instead of the velocity of the fluid, preserving circulation during the simulation. Vorticity is usually tracked by Lagrangian elements such as vortex particles [Park and Kim 2005], vortex filaments [Weißmann and Pinkall 2010] or vortex sheets [Brochu et al. 2012]. Apart from the fact that the computational cost of these methods can be a lot more expensive per-time-step than a grid-based solver (three accurate Poisson solves are required instead of just one), there remain difficulties in handling solid boundaries and free surfaces (for liquids), and users may find it less intuitive working with vorticity rather than velocity in crafting controls for art direction. However, researchers have observed that vortex methods can better capture important visual details when running with the same time step and advection scheme [Yaeger et al. 1986].

In this paper, we combine our observation from vortex methods with Eulerian grid-based velocity-pressure solvers to arrive at a novel vorticity error compensation scheme. Our contributions include:

- A new scheme we dub IVOCK (Integrated Vorticity of Convective Kinematics), that tracks and approximately restores the dissipated vorticity during advection, independent of the advection method or other fluid dynamics in play (boundary conditions, forcing terms) being used.
- A set of novel techniques to store and advance vortex dynamics on a fixed spatial grid, maximizing the accuracy while minimizing the computational effort of IVOCK.

Figure 3: *Vorticity confinement (VC) vs. IVOCK. Top row: frame 54 of a rising smoke scenario. From left to right, VC parameter $\varepsilon_1 = 0.125$, $\varepsilon_2 = 0.25$, $\varepsilon_3 = 0.5$, and SL3-IVOCK. Bottom row: frame 162. Vorticity confinement tends to create high-frequency noise everywhere, while IVOCK produces a natural transition from laminar to turbulent flow with realistic vortex structures along the way.*

- Upgraded classic fluid solvers with IVOCK scheme, documenting the combination of IVOCK with different advection methods and different types of fluids.
- A new vortex stretching model in a non-divergence-free environment for compressible vortex flows occuring in volumetric combustion.

2 Related work

Stam's seminal Stable Fluids method [1999] introduced a backward velocity tracing scheme to solve advection equation, making physically based fluid animation highly practical for computer graphics. With its unconditional stability, trading accuracy for larger time step, it provided a basis for most subsequent grid-based fluid solvers in graphics, for fluid phenomena such as smoke [Fedkiw et al. 2001], water [Foster and Fedkiw 2001], thin flames [Nguyen et al. 2002], and volumetric combustion [Feldman et al. 2003].

However, the first order accuracy in both time and space of Stable Fluids manifests in strong numerical diffusion/dissipation. Many researchers have taken it as a basic routine to build higher-order advection schemes. Kim et al. [2005] proposed the BFECC scheme to achieve higher-order approximation by advecting the field back and forth, measuring and correcting errors. Selle et al. [2008] eliminated the last advection step of BFECC to arrive at a cheaper MacCormack-type method, and also introduced extrema clamping in BFECC and MacCormack to attain unconditional stability, at the cost of discontinuities in velocity which can sometimes cause visual artifacts. In this paper, we use these advection schemes as examples to which we apply IVOCK, and show improvements with the IVOCK scheme for smoke and volumetric combustion simulations.

Hybrid particle-grid methods have been introduced to further reduce numerical diffusion in adection, notably Zhu and Bridson's adaptation of FLIP to incompressible flow [Zhu and Bridson 2005]. Although FLIP is almost non-diffusive for advection, the velocity-pressure solver nevertheless may dissipate rotational motion as shown in figure 2, regardless of the accuracy of advection. FLIP cannot address this issue; in this paper we show our FLIP-IVOCK scheme outperforms FLIP in enhancing rotational motions for smoke.

Resolution plays an important role in fluid animations. McAdams et al. [2010] proposed multigrid methods to efficiently solve the Poisson equation at a uniform high resolution. Losasso et al. [2004] introduced a fluid solver running on an octree data structure to adaptively increase resolution where desired, while Setaluri et al. [2014] adopted a sparse data approach. In our work, multigrid is employed for the vorticity-stream function solver while an octree code is used to impose domain-boundary values using Barnes-Hut summation.

Vortex dynamics has proved a powerful approach to simulating turbulence. Lagrangian vortex elements such as vortex particles [Park and Kim 2005], vortex filaments [Weißmann and Pinkall 2010], or vortex sheets ([Brochu et al. 2012], [Pfaff et al. 2012]) have been used to effectively model the underlying vorticity field. Recently, Zhang and Bridson [2014] proposed a fast summation method to reduce the computational burden for Biot-Savart summation, in an attempt to make these methods more practical for production. Unfortunately, these methods are less intuitive for artists (velocity can't be modified directly), tend to be more expensive (finding velocity from vorticity requires the equivalent of three Poisson solves), and continue to pose problems in formulating good boundary conditions. We were nevertheless inspired by the mechanism to induce velocity from vorticity, and constructed our IVOCK scheme as a way to bring some of the advantages of vortex methods to more practical velocity-pressure solvers.

To balance the trade-off between pure grid-based methods and pure Lagrangian vortex methods, researchers have incorporated vorticity-derived forces in Eulerian simulations. Foremost among these is the vorticity confinement approach [Steinhoff and Underhill 1994], where a force field is derived from the current vorticity field to boost it. Selle at al. [2005] refined this by tracing "vortex particles", source terms for vorticity confinement tracked with the flow which provides better artistic control over the added turbulence. Unfortunately, vorticity confinement relies on a non-physical parameter which must be tuned by the artist. As we can clearly see in figure 3, too small a parameter doesn't produce interesting motion for the early frames of a smoke animation, while still turning the smoke into incoherent noise in later frames. IVOCK on the other hand is built in a more principled way upon vortex dynamics, partially correcting the truncation error in velocity-pressure time-splitting, and produces natural swirling motions without any parameters to tune. In addition, it is orthogonal to vorticity confinement (viewed as an art direction tool) and can hence be used together (figure 12).

Turbulence can also be added to the flow as a post-simulation process, such as with wavelet turbulence [Kim et al. 2008]. In particular, the wavelet up-sampling scheme relies on a good original velocity field to produce visually pleasing animations; IVOCK is again orthogonal to this and could be adopted to enhance the basic simulation.

3 The IVOCK scheme

When solving Navier-Stokes, the fluid velocity is usually advanced to an intermediate state ignoring the incompressibility constraint:

$$\tilde{\boldsymbol{u}} = \text{Advect}\left(\boldsymbol{u}^n\right), \tag{3}$$

$$\tilde{\tilde{\boldsymbol{u}}} = \tilde{\boldsymbol{u}} + \Delta t \boldsymbol{f}, \tag{4}$$

where $\boldsymbol{u}^n$ is the divergence-free velocity field from the previous time-step and $\boldsymbol{f}$ is a given force field which may include buoyancy, diffusion, vorticity confinement, and artistically controlled wind or motion objects.

From this intermediate velocity field $\tilde{\tilde{\boldsymbol{u}}}$ one can construct the final divergence-free velocity $\boldsymbol{u}^{n+1}$ with pressure projection, which is usually the place where boundary conditions are also handled:

$$\boldsymbol{u}^{n+1} = \mathbf{Proj}\left(\tilde{\tilde{\boldsymbol{u}}}\right). \tag{5}$$

We observe that in vortex dynamics, the intermediate velocity field $\tilde{\boldsymbol{u}}$ of equation 3 is analogously solved using the velocity-vorticity $(\boldsymbol{u}-\boldsymbol{\omega})$ formula:

1. given $\boldsymbol{\omega}^n$, solve
$$\frac{D\boldsymbol{\omega}}{Dt} = \boldsymbol{\omega}\cdot\nabla\boldsymbol{u} \tag{6}$$
to get $\tilde{\boldsymbol{\omega}}$;
2. deduce the intermediate velocity field $\tilde{\boldsymbol{u}}$ from $\tilde{\boldsymbol{\omega}}$.

Equation 6 is the advection-stretching equation for vorticity in 3D, and step 2 of this $(\boldsymbol{u}-\boldsymbol{\omega})$ formula is usually solved using a Biot-Savart summation, or equivalently by finding a streamfunction Ψ (vector-valued in 3D), which satisfies $\nabla^2\Psi = -\boldsymbol{\omega}$, and readily obtaining the velocity $\tilde{\boldsymbol{u}}$ from $\tilde{\boldsymbol{u}} = \nabla\times\Psi$.

Equation 6 suggests the post-advection vorticity field $\boldsymbol{\omega}^* = \nabla\times\tilde{\boldsymbol{u}}$ should ideally equal the stretched and advected vorticity field $\tilde{\boldsymbol{\omega}}$, but due to the simple nature of self-advection of velocity ignoring pressure, there will be an error related to the time step size.

We therefore define a vorticity correction $\delta\boldsymbol{\omega} = \tilde{\boldsymbol{\omega}} - \boldsymbol{\omega}^*$, from which we deduce the IVOCK velocity correction $\delta\boldsymbol{u}$ and add this amount to the intermediate velocity $\tilde{\boldsymbol{u}}$. Algorithm 1 provides an outline of the IVOCK computation before we discuss the details.

Algorithm 1 IVOCKAdvection(Δt,$\boldsymbol{u}^n$, $\tilde{\boldsymbol{u}}$)

$\boldsymbol{\omega}^n \leftarrow \nabla\times\boldsymbol{u}^n$
$\tilde{\boldsymbol{\omega}} \leftarrow$ stretch($\boldsymbol{\omega}^n$)
$\tilde{\boldsymbol{\omega}} \leftarrow$ advect(Δt, $\tilde{\boldsymbol{\omega}}$)
$\tilde{\boldsymbol{u}} \leftarrow$ advect(Δt, $\boldsymbol{u}^n$)
$\boldsymbol{\omega}^* \leftarrow \nabla\times\tilde{\boldsymbol{u}}$
$\delta\boldsymbol{\omega} \leftarrow \tilde{\boldsymbol{\omega}} - \boldsymbol{\omega}^*$
$\delta\boldsymbol{u} \leftarrow$ VelocityFromVorticity($\delta\boldsymbol{\omega}$)
$\tilde{\boldsymbol{u}} \leftarrow \tilde{\boldsymbol{u}} + \delta\boldsymbol{u}$

In essence, IVOCK upgrades the self-advection of velocity to match self-advection of vorticity, yet retains most of the efficiency and all of the flexibility of a velocity-pressure simulator. In particular, IVOCK doesn't change the pressure computation (as the divergence of the curl of the correcting streamfunction is identically zero, hence the right-hand-side of the pressure projection is unchanged by IVOCK), but simply improves the resolution of rotational motion for large time steps, which we earlier saw was hurt by velocity self-advection.

3.1 Vortex dynamics on grids

3.1.1 Data storage

Extending the classic MAC grid [Harlow and Welch 1965] widely adopted by computer graphics researchers for fluid simulation, we store vorticity and streamfunction components on cell edges in a staggered fashion, so that the curl operator can be implemented more naturally, as illustrated in figure 4. For example, if the grid cell size is h, the z-component of vorticity on the z-aligned edge centred at $(ih, jh, (k+\frac{1}{2})h)$ is approximated as

$$\omega_z(i,j,k+\frac{1}{2}) = \frac{1}{h}\left[\left(v\left(i+0.5,j,k\right) - v\left(i-0.5,j,k\right)\right) - \left(u\left(i,j+0.5,k\right) - u\left(i,j-0.5,k\right)\right)\right]. \tag{7}$$

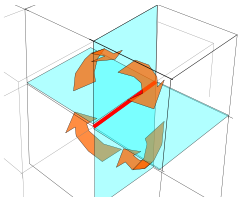

Figure 4: *Vorticity and streamfunction components are stored in a staggered fashion on cell edges in 3D (red line), while velocity components are stored on face centers. This permits a natural curl finite difference stencil, as indicated by the orange arrows.*

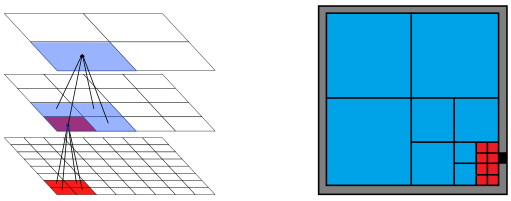

Figure 5: *Barnes-Hut summation [Barnes and Hut 1986] for the boundary values demonstrated in 2D: we construct the monopoles of tree nodes from bottom-up (left), and then for an evaluation voxel (black) in the ghost boundary region (grey), we travel down the tree, accumulating the far-field influence by the monopoles (blue nodes), and do direct summation only with close enough red cells (right).*

3.1.2 Vortex stretching and advection on grid

In 3D flow, when a vortex element is advected by the velocity field, it is also stretched, changing the vorticity field.

We solve equation 6 on an Eulerian grid using splitting; we solve

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \boldsymbol{\omega} \cdot \nabla \boldsymbol{u}, \tag{8}$$

to arrive at an intermediate vorticity field $\boldsymbol{\omega}^{n+\frac{1}{2}}$, which is then advected by the velocity field as

$$\frac{D\boldsymbol{\omega}}{Dt} = 0. \tag{9}$$

When discretizing equation 8 on an Eulerian grid, a geometrically-based choice would be to compute the update with the Jacobian matrix of $\boldsymbol{u}$:

$$\boldsymbol{\omega}^{n+\frac{1}{2}} = \boldsymbol{\omega}^n + \Delta t \mathcal{J}(\boldsymbol{u}) \boldsymbol{\omega}^n. \tag{10}$$

Constructing the Jacobian matrix and computing the matrix vector multiplication involves a lot of arithmetic, so we simplified this computation by using

$$\boldsymbol{\omega} \cdot \nabla \boldsymbol{u} = \frac{\partial \boldsymbol{u}}{\partial \boldsymbol{\omega}} = \lim_{\varepsilon \to 0} \frac{\boldsymbol{u}(x + 0.5\varepsilon\boldsymbol{\omega}) - \boldsymbol{u}(x - 0.5\varepsilon\boldsymbol{\omega})}{\varepsilon}. \tag{11}$$

In our computation, with h the grid cell width, we choose $\varepsilon = \frac{h}{\|\boldsymbol{\omega}\|_2}$, which can be seen as sampling the velocity at the two ends of a grid-cell-sized vortex segment and evaluating how this segment is stretched by the velocity field.

Once the vorticity field is stretched, it is advected by any chosen scheme introduced in §2 to get $\tilde{\boldsymbol{\omega}}$.

3.1.3 Obtaining $\delta \boldsymbol{u}$ from $\delta \boldsymbol{\omega}$

To deduce the velocity correction from the vorticity difference, we solve the Poisson equation for the stream function:

$$\nabla^2 \boldsymbol{\Psi} = -\delta \boldsymbol{\omega} \tag{12}$$

We solve each component of the vector stream function separately: Let subscript x indicate the x-component of the vector function, we have

$$\nabla^2 \Psi_x = -\delta \omega_x, \tag{13}$$

The equations for y and z components can be obtained in a similar fashion.

In vortex dynamics, it is important for this Poisson equation to be solved in open space to get natural motion, without artifacts from the edges of the finite grid. The open space solution of Poisson's equation can be found by a kernel summation with the corresponding fundamental solution. Consider the scalar Poisson problem,

$$\begin{aligned} \nabla^2 \Psi_x &= -\delta \omega_x \\ \Psi_x(p) &= 0, p \to \infty, \end{aligned} \tag{14}$$

Its solution, in a discrete sense, is given by the N-body summation

$$\Psi_x(p_i) \approx \sum_{\text{all} j, j \neq i} \frac{\delta \omega_{x_j} v_j}{4\pi r_{i,j}}, \tag{15}$$

where $\delta \omega_{x_j}$, v_j are the vortex source strength (component-wise) and volume of each sample point (voxel), respectively, p_i is the evaluation position, and r_{ij} is the distance between the i^{th} and j^{th} sample positions. With M evaluation points (on the six faces of the domain boundary) and N source points (the number of voxels), the direct N-body summation has an $\mathcal{O}(MN)$ cost. However, once an evaluation position p_i is far from a cloud of s sources, the summation of the influence of each individual source can be accurately approximated by the influence of the monopole of those sources:

$$\sum_{j=1}^{s} \frac{\delta \omega_{x_j} v_j}{4\pi \|p_i - p_j\|} \approx \frac{1}{4\pi \|p_i - p_c\|} \sum_{j=1}^{s} \delta \omega_{x_j} v_j, \tag{16}$$

where p_c is the center of mass of the cloud, and $\sum_{j=1}^{s} \delta \omega_{x_j} v_j$ is the so-called monopole. Defining an octree on the voxels, akin to multigrid, we construct from bottom-up the monopoles of clusters of voxels, and then for each evaluation position, we traverse the tree top-down as far as needed to accurately and efficiently approximate the N-body summation, as illustrated in Fig.5. Similarly to Liu and Doorly [2000], we apply this summation scheme at the $M \propto N^{\frac{2}{3}}$ boundary cells to approximate the open-space boundary values at an $\mathcal{O}(N^{\frac{2}{3}} \log(N))$ cost, which we then use as boundary conditions on a more conventional Poisson solve.

We discretize the Poisson equation with a standard 7-point finite difference stencil. Because the vorticity difference is small (one can view it as a truncation error proportional to the time step), we can get away with applying a single multigrid V-cycle to solve for each component of the stream function. Recall this is not the stream function for the complete velocity field of the flow, just the much smaller velocity correction to the intermediate self-advected velocity! One V-cycle provides adequate accuracy and global coupling across the grid at a computational cost of only about 20 basic red-black Gauss-Seidel iterations. This is cheaper than the pressure solve, which requires more than three V-cycles for the required accuracy, and is also substantially cheaper than a vorticity-velocity solver which requires three accurate Poisson solves.

Once the approximate streamfunction is determined, the velocity correction is computed as $\delta \boldsymbol{u} = \nabla \times \boldsymbol{\Psi}$.

Figure 6: *Two sequences from 3D rising smoke simulations. Top row: MC-IVOCK with vortex stretching. Bottom row: MC-IVOCK with vortex stretching switched off. With vortex stretching, vortex rings change their radius under the influence of other vortex rings, these process can easily perturb the shape of vortex rings, breaking them to form new vortex structures, which brings rich turbulence into the flow field.*

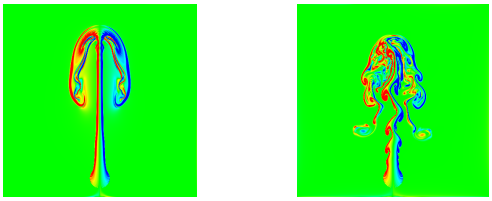

Figure 7: *Left: 2D buoyancy flow simulated with SF. Right: the same with SF-IVOCK. The resolution and time step were the same; the IVOCK scheme produces a more richly detailed result.*

3.2 Discussion

3.2.1 Vortex stretching

Before proceeding to the 3D applications of the IVOCK scheme for smoke §4.1, water §4.2 and combustion §4.3, we present a few examples illustrating the effect of vortex stretching.

In 2D flows, vortex stretching doesn't take place, hence one need only advect the vorticity field. Figure 7 compares SF and SF-IVOCK in 2D.

In 3D flows, vortex stretching plays an important role. Rising vortex rings leapfrogging each other is a physically unstable structure: the vortex rings break up under small perturbations and form new vortex structures. Figure 6 illustrates that IVOCK without vortex stretching produces, at large time steps, visibly wrong results. In figure 8, we show that the ability of the IVOCK method to capture vortex stretching can bring more turbulence to the flow, such as wrinkles on the smoke front, again with relatively large time steps.

3.2.2 Additional forces

IVOCK is only a correction to the self-advection part of a standard velocity-pressure solver. As such, we do not need to take into account how additional force terms such as buoyancy, viscosity, and artist controls will interact with vorticity. These forces are incorporated into the velocity directly in a different step.

Figure 8: *Vortex stretching enhances vortical motion, captured more accurately with IVOCK. Transparent renderings illustrate the internal structures. Left and middle-lfet: FLIP. Middle-right and right most: FLIP-IVOCK.*

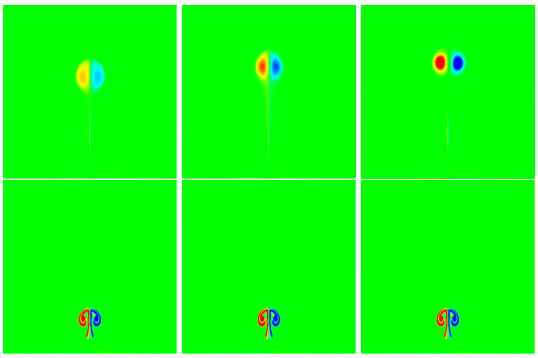

Figure 9: *Rising vortex pair initialized by a heat source. Left column: SF with $\Delta t = 0.01$. Middle column: SF with $\Delta t = 0.0025$. Right column: SF-IVOCK with $\Delta t = 0.01$. Notice IVOCK preserves vorticity best and produces the highest final position among the three.*

3.2.3 Boundary conditions

While solving IVOCK dynamics, we don't take boundaries into account at all: we pose the vorticity-velocity equation in open space, and allow the subsequent pressure projection to handle boundaries. However, in some simulations with a strong shear layer along a viscous boundary, we found IVOCK could cause instability; this was cured easily by zeroing out $\delta\omega$ within a few grid cells of the boundary.

It is worth noting that velocity-pressure solvers in general, and IVOCK in particular, rely on the numerical diffusion of the advection to produce vortex shedding along boundaries (which are otherwise not predicted by the fully inviscid equations). Figure 12 shows an example where rising smoke collides with an obstacle; the velocity boundary condition is handled by the pressure solver, and no special treatment is needed for the IVOCK streamfunction computation.

4 Applications and Results

4.1 Smoke

The clearest application of IVOCK is in enriching smoke simulations, where vortex features are of crucial importance visually.

In 2D, we initialized a raising vortex pair from a heat source. With SF, this vortex pair all but vanishes after 100 time steps; SF-IVOCK preserves the vorticity much better, as shown in figure 9. We recorded the total vortex strength (enstrophy) at each time step, plotted in figure 10.

In 3D, we applied IVOCK to different advection schemes. Figure 1 illustrates the qualitative improvements to all of them. For large

time steps, IVOCK significantly enhances the rotational motion and structures before turbulence fully develops.

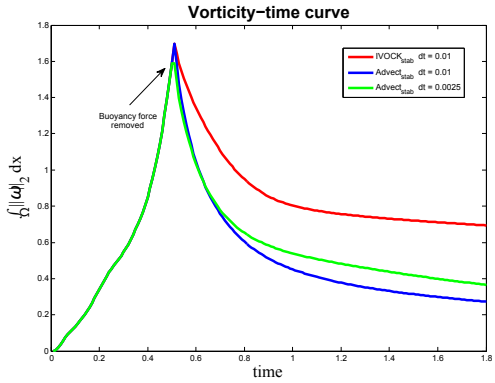

Figure 10: *Vorticity vs. time curve of the 2D vortex pair simulation. While IVOCK does not conserve vorticity exactly due to approximations in advection and the streamfunction computation, it still preserves significantly more.*

Computationally speaking, the overhead for IVOCK includes the stretching computation, vorticity evaluation, vorticity advection, and three V-cycles to approximate the vorticity streamfunction. These operations add about 10% to 25% extra runtime to the original method as a whole: see table 2 for sample performance numbers. Alternatively put, the runtime overhead of IVOCK could be equivalent to improving the original method with about $0.94 \times \Delta x$ in spatial resolution, or taking about $0.83\times$ smaller time steps. Figure 11 illustrates that running IVOCK produces better results than even a simulation with half the time step size, despite running much faster.

Table 2: *Performance comparison of IVOCK augmenting different schemes, for a smoke simulation at 128x256x128 grid resolution, running on an Intel(R) Core(TM) i7-3630QM 2.40GHz CPU.*

Method	sec /time step	% overhead
SF	26.6	
SF-IVOCK	30.5	14%
SL3	27.7	
SL3-IVOCK	32.6	17%
BFECC	31.8	
BFECC-IVOCK	39.9	28%
MC	27.9	
MC-IVOCK	35.4	26%
FLIP	45.5	
FLIP-IVOCK	51.4	12%

Figure 11: *IVOCK can be cheaper and higher quality then taking small time steps. Left and mid-left: BFECC. Mid-right and right: BFECC-IVOCK with twice as large a time step, computed in much less time.*

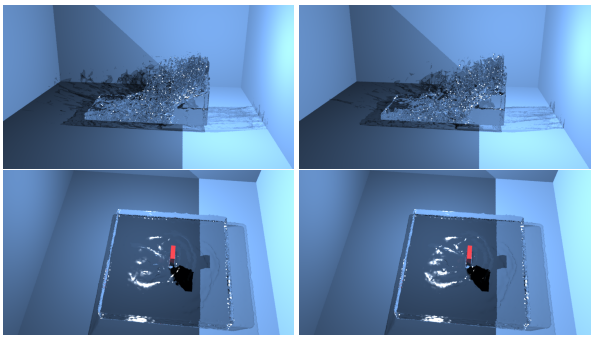

Figure 13: *IVOCK applied to liquids. Top row: dam break simulations obtained with FLIP (left) and FLIP-IVOCK (right). Bottom row: moving paddle in a water tank, simulated with FLIP (left) and FLIP-IVOCK (right). In these and several other cases we tested, FLIP-IVOCK is not a significant improvement, presumably because interior vorticity is either not present (irrotional flow) or not visually important.*

4.2 Liquids

We also implemented IVOCK for liquids solved as free surface flow (see figure 13), with a solver based on Batty et al.'s [2007]. In this case we only compute the vorticity correction $\delta\omega$ inside the liquid, sufficiently below the free surface so that extrapolated velocities are not involved in the vorticity stencil, as otherwise we found stability issues. We solve for the velocity correction with a global solve as before, disregarding boundaries.

Interestingly, we found IVOCK made little difference to the common water scenarios we tested, even ones where we made sure to generate visible vortex structures (e.g. trailing from a paddle pushing through water). We note first that typically the only visible part of a liquid simulation is the free surface itself, unlike smoke and fire, so interesting vortex motions under the surface are relatively unimportant. We also hypothesize that most visually interesting liquid phenomena has to do with largely ballistic motion (splashes) or irrotional motion (as in ocean waves), chiefly determined by gravity, so internal vorticity is of lesser importance. While potential flow can be modeled with a vortex sheet along the free surface, we leave capturing this stably in IVOCK to future work.

4.3 Fire

For combustion, the velocity field is not always divergence-free due to expansions from chemical reactions and intense heating. We subsequently modify the vortex stretching term, noting that $\boldsymbol{\omega}/\rho$ is the appropriate quantity to track [Tabak 2002]:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\boldsymbol{\omega}}{\rho}\right) = \left(\frac{\boldsymbol{\omega}}{\rho}\right)\cdot\nabla\boldsymbol{u}. \tag{17}$$

We discretize equation 17 with Forward Euler,

$$\frac{1}{\Delta t}\left(\frac{\boldsymbol{\omega}^{n+1}}{\rho^{n+1}} - \frac{\boldsymbol{\omega}^{n}}{\rho^{n}}\right) = \left(\frac{\boldsymbol{\omega}^{n}}{\rho^{n}}\right)\cdot\nabla\boldsymbol{u}^{n} \tag{18}$$

which implies that

$$\boldsymbol{\omega}^{n+1} = \left(\frac{\rho^{n+1}}{\rho^{n}}\right)\left(\boldsymbol{\omega}^{n} + \Delta t\boldsymbol{\omega}^{n}\cdot\nabla\boldsymbol{u}^{n}\right). \tag{19}$$

Therefore the new strength of a vortex element, after being stretched and advected, shall be scaled by $\frac{\rho^{n+1}}{\rho^{n}}$.

Figure 12: *Rising smoke hitting a spherical obstacle: the velocity boundary condition is handled entirely by the pressure solve, and doesn't enter into the IVOCK scheme at all. This example includes a small additional amount of vorticity confinement to illustrate how the methods can be trivially combined.*

In compressible flow, mass conservation can be written as

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = \frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho \nabla \cdot \boldsymbol{u} = 0. \tag{20}$$

Rewriting equation 20 gives

$$\frac{1}{\rho}\frac{\mathrm{D}\rho}{\mathrm{D}t} = \frac{\mathrm{D}\ln\rho}{\mathrm{D}t} = -\nabla \cdot \boldsymbol{u}, \tag{21}$$

and by using Forward Euler again we get

$$\frac{\rho^{n+1}}{\rho^n} = \exp\left(-\Delta t \nabla \cdot \boldsymbol{u}\right). \tag{22}$$

Plugging equation 22 into equation 17, we observe that in compressible flow, the vorticity field after being stretched and advected, should be scaled by a factor of $\exp\left(-\Delta t \nabla \cdot \boldsymbol{u}\right)$. This scaling is the only change we make to IVOCK for compressible flow. We combined the modified IVOCK scheme with traditional volumetric combustion models described by Feldman et al. [2003]; figure 14 shows example frames from such an animation.

5 Conclusion

We argue IVOCK is an interesting stand-alone method to cheaply enrich the highly flexible grid-based velocity-pressure framework with much better resolved vorticity at large time steps, and which can be applied to a variety of advection schemes and fluid phenomena. We believe it brings many of the advantages of vortex solvers to velocity-pressure schemes, but with only $10 \sim 25\%$ extra computation and without the complexity of handling boundary conditions etc. in a vorticity formulation. However, there are some limitations and areas for future work we would highlight.

Currently IVOCK is limited to fluid simulations on uniform grids. A version for adaptive grid or tetrahedral mesh solvers doesn't appear intrinsically difficult, as it could reuse the solver's Poisson infrastructure. Fully Lagrangian velocity-pressure particle methods, especially those that already require some form of vorticity confinement to combat intrinsic numerical smoothing, such as position-based fluids [Macklin and Müller 2013], may also benefit from the IVOCK idea.

The presented IVOCK scheme doesn't exactly recover total circulation (or energy) as Mullen et al.'s symplectic integrator does [2009], nor does it reproduce turbulent flow quite as well as Lagrangian vortex methods. We also have at present little more than numerical evidence to argue for its stability, and we haven't thoroughly investigated the trade-offs of using just an approximate multigrid solution for the vorticity-velocity step. It could be worthwhile to enhance the stability by exploring Elcott et al.'s circulation preserving method [2007].

The IVOCK scheme alone doesn't provide any insight into boundary layer models or unresolved subgrid details, hence one possible future project could be to combine it with subgrid turbulence models, for post-processing [Kim et al. 2008] or on-the-fly synthesis [Pfaff et al. 2009]. However, we did observe stability problems with IVOCK compensation active in strongly shearing boundary layers; until a better understanding of boundary layers is reached, we recommend only including the vorticity difference driving IVOCK at some distance away from boundary layers.

6 Acknowledgement

We thank the anonymous reviewers for their useful suggestions. We also thank Dr. Mark J. Stock for inspiring discussions regarding the Barnes-Hut summation method for the open boundary condition, and the Natural Sciences and Engineering Research Council of Canada for supporting this research.

References

BARNES, J., AND HUT, P. 1986. A hierarchical O(N log N) force-calculation algorithm. *Nature 324*, 446–449.

BATTY, C., BERTAILS, F., AND BRIDSON, R. 2007. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph. (Proc. SIGGRAPH) 26*, 3, 100.

BRIDSON, R. 2008. *Fluid Simulation for Computer Graphics*. A K Peters / CRC Press.

BROCHU, T., KEELER, T., AND BRIDSON, R. 2012. Linear-time smoke animation with vortex sheet meshes. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comp. Animation*, 87–95.

EDWARDS, E., AND BRIDSON, R. 2014. Detailed water with coarse grids: combining surface meshes and adaptive discontinuous Galerkin. *ACM Trans. Graph. (Proc. SIGGRAPH) 33*, 4, 136:1–9.

ELCOTT, S., TONG, Y., KANSO, E., SCHRÖDER, P., AND DESBRUN, M. 2007. Stable, circulation-preserving, simplicial fluids. *ACM Trans. Graph. 26*, 1 (Jan.).

FEDKIW, R., STAM, J., AND JENSEN, H. W. 2001. Visual simulation of smoke. In *Proc. ACM SIGGRAPH*, 15–22.

FELDMAN, B. E., O'BRIEN, J. F., AND ARIKAN, O. 2003. Animating suspended particle explosions. *ACM Trans. Graph. (Proc. SIGGRAPH) 22*, 3, 708–715.

FOSTER, N., AND FEDKIW, R. 2001. Practical animation of liquids. In *Proc. ACM SIGGRAPH*, 23–30.

HARLOW, F. H., AND WELCH, J. E. 1965. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surfaces. *Physics of fluids 8*, 2182–2189.

Figure 14: *Fire simulations making use of our combustion model. Top row: BFECC. Bottom row: BFECC augmented with IVOCK using SL3 for vorticity and scalar fields. The temperature is visualized simply by colouring the smoke according to temperature.*

KIM, B., LIU, Y., LLAMAS, I., AND ROSSIGNAC, J. 2005. FlowFixer: Using BFECC for fluid simulation. In *Proc. First Eurographics Conf. on Natural Phenomena*, NPH'05, 51–56.

KIM, T., THUREY, N., JAMES, D., AND GROSS, M. H. 2008. Wavelet turbulence for fluid simulation. *ACM Trans. Graph. (Proc. SIGGRAPH) 27*, 3, 50.

LENTINE, M., AANJANEYA, M., AND FEDKIW, R. 2011. Mass and momentum conservation for fluid simulation. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comp. Anim.*, 91–100.

LIU, C. H., AND DOORLY, D. J. 2000. Vortex particle-in-cell method for three-dimensional viscous unbounded flow computations. *International Journal for Numerical Methods in Fluids 32*, 1, 23–42.

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

MACKLIN, M., AND MÜLLER, M. 2013. Position based fluids. *ACM Trans. Graph. (Proc. SIGGRAPH) 32*, 4, 104.

MCADAMS, A., SIFAKIS, E., AND TERAN, J. 2010. A parallel multigrid poisson solver for fluids simulation on large grids. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comp. Anim.*, 65–74.

MULLEN, P., CRANE, K., PAVLOV, D., TONG, Y., AND DESBRUN, M. 2009. Energy-preserving integrators for fluid animation. *ACM Trans. Graph. (Proc. SIGGRAPH) 28*, 3, 38:1–38:8.

NGUYEN, D. Q., FEDKIW, R., AND JENSEN, H. W. 2002. Physically based modeling and animation of fire. *ACM Trans. Graph. (Proc. SIGGRAPH) 21*, 3, 721–728.

PARK, S. I., AND KIM, M.-J. 2005. Vortex fluid for gaseous phenomena. In *Proc. ACM SIGGRAPH / Eurographics Symp. Comp. Animation*, 261–270.

PFAFF, T., THUEREY, N., SELLE, A., AND GROSS, M. 2009. Synthetic turbulence using artificial boundary layers. *ACM Trans. Graph. 28*, 5, 121.

PFAFF, T., THUEREY, N., AND GROSS, M. 2012. Lagrangian vortex sheets for animating fluids. *ACM Trans. Graph. (Proc. SIGGRAPH) 31*, 4, 112:1–8.

SELLE, A., RASMUSSEN, N., AND FEDKIW, R. 2005. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3, 910–914.

SELLE, A., FEDKIW, R., KIM, B., LIU, Y., AND ROSSIGNAC, J. 2008. An Unconditionally Stable MacCormack Method. *J. Scientific Computing 35*, 2–3, 350–371.

SETALURI, R., AANJANEYA, M., BAUER, S., AND SIFAKIS, E. 2014. SPGrid: A sparse paged grid structure applied to adaptive smoke simulation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 33*, 6, 205:1–205:12.

STAM, J. 1999. Stable fluids. In *Proc. ACM SIGGRAPH*, 121–128.

STEINHOFF, J., AND UNDERHILL, D. 1994. Modification of the Euler equations for "vorticity confinement": Application to the computation of interacting vortex rings. *Physics of Fluids 6*, 2738–2744.

TABAK, E. G., 2002. Vortex stretching in incompressible and compressible fluids. Courant Institute, Lecture Notes (Fluid Dynamics II), http://www.math.nyu.edu/faculty/tabak/vorticity.pdf.

WEISSMANN, S., AND PINKALL, U. 2010. Filament-based smoke with vortex shedding and variational reconnection. *ACM Trans. Graph. (Proc. SIGGRAPH) 29*, 4, 115.

YAEGER, L., UPSON, C., AND MYERS, R. 1986. Combining physical and visual simulation–creation of the planet Jupiter for the film "2010". *Proc. SIGGRAPH 20*, 4 (Aug.), 85–93.

ZHANG, X., AND BRIDSON, R. 2014. A PPPM fast summation method for fluids and beyond. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 33*, 6, 206:1–11.

ZHU, Y., AND BRIDSON, R. 2005. Animating sand as a fluid. *ACM Trans. Graph. (Proc. SIGGRAPH) 24*, 3, 965–972.