A Vortex Method for Bi-phasic Fluids Interacting with Rigid Bodies

Mathieu Coquerelle[*] GRAVIR/IMAG and LMC/IMAG

Jérémie Allard[†] GRAVIR/IMAG

Georges-Henri Cottet[‡] LMC/IMAG

Marie-Paule Cani[§] GRAVIR/IMAG

Abstract

We present an accurate Lagrangian method based on vortex particles, level-sets, and immersed boundary methods, for animating the interplay between two fluids and rigid solids. We show that a vortex method is a good choice for simulating bi-phase flow, such as liquid and gas, with a good level of realism. Vortex particles are localized at the interfaces between the two fluids and within the regions of high turbulence. We gain local precision and efficiency from the stable advection permitted by the vorticity formulation. Moreover, our numerical method straightforwardly solves the two-way coupling problem between the fluids and animated rigid solids. This new approach is validated through numerical comparisons with reference experiments from the computational fluid community. We also show that the visually appealing results obtained in the CG community can be reproduced with increased efficiency and an easier implementation.

1 Introduction

Fluids interacting with solid objects are a common, yet fascinating every-day life experience. Our tendency to stare at turbulent liquids and at smoke dynamic behavior, or to observe different objects splashing into water makes us very critical when we see such phenomena reproduced with a computer. Yet, the demand for plausible fluids in computer-generated movies and games is high. This has led many computer graphics researchers to tackle the challenge, although still considered as one of the most difficult problem in the computational fluids community.

The interactions we generally observe in real life involve several components: typically, water with a free surface in contact with the air, which interacts with both still and moving rigid bodies. These interactions result in really complex phenomena (turbulences, splashes, bubbles, interesting subsequent motion of the solids), due to the interplay between the two fluids and the solids. This paper presents a Lagrangian approach, based on vortex particles, for accurately, yet efficiently simulating this interplay. We show that vortex particles provide a good solution to the modelling of bi-phase flow with a liquid component. Since they concentrate the computational power in discontinuous and turbulent regions, the complex phenomena occurring at the interface between the two fluids are represented with a good level of precision. Meanwhile, the numerical method we present straightforwardly solves the two-way coupling problem between the fluids and animated rigid solids, without the need for complex boundary conditions.

[*]e-mail: Mathieu.Coquerelle@imag.fr
[†]e-mail: Jeremie.Allard@imag.fr
[‡]e-mail: Georges-Henri.Cottet@imag.fr
[§]e-mail: Marie-Paule.Cani@imag.fr

Previous Work

The simulation of fluids and of their interplay with solid objects has attracted a lot of attention in the CG community within the past few years. Great advances were recently made towards this goal.

[17] presented the first 3D simulation of liquids. They relied on an Eulerian formulation to solve the Navier-Stokes equations for incompressible fluids. A semi-Lagragian method for advecting the fluid which guaranties the stability of the simulation was then proposed by [30]. Because of high numerical dissipation, these kind of simulations loose part of their vorticity over time, a problem which was tackled in [16].
Lagrangian methods based on particles provide a good alternative to Eulerian simulations, since they enable to dynamically follow the fluid. The *Smoothed Particles Hydrodynamics* (SPH) formulation was adapted by [12] in order to transport an implicit boundary with surface tension. More recently, [26] used this formulation to simulate fluid-fluid interactions such as boiling water and managed to obtain interesting phenomena such as air bubbles in water. Closer to the Navier-Stokes equations for incompressible fluids, [28] proposed a particle based solution called *Moving Particle Semi-Implicit* (MPS). While straightforwardly able to track multiple fluids, particles-only methods mostly suffer from the difficulty to conserve the dynamic properties of the fluids, especially when dealing with boundaries.
Among these approaches, vortex methods [18, 27, 2] were used in for animating gaseous phenomena. They are particularly interesting since they focus the resolution in the regions of high turbulence. They rely on the vorticity formulation of the Navier-Stokes equations, well known in applied mathematics for being an accurate alternative to Eulerian methods (see [9]). Vortex particles were also introduced to add subgrid turbulences on top of an Eulerian simulation of liquids or gazes [29], impressively counteracting the numerical dissipation due to the underlying Eulerian solver. Up to now, no technique was developed in graphics to simulate liquids or multi-phases fluids from the vorticity formulation of Navier-Stokes. This may be due to the inherent cost of particle methods (from $O(N^2)$ to $O(NlogN)$ using [24]), more expensive than Eulerian methods since a large number of particles ($N \gg 1000$) are necessary to reach a good level of precision. One of the contributions of this paper is to show that when adequately implemented, the vortex formulation achieves accurate yet efficient simulations.

Animation and visualization of water in contact with the air require the precise transport and rendering of the interface between them. Because of the discontinuity of the fluids' physical properties at the interface, the resolution of the Navier-Stokes equations was most of the time restricted to the liquid component and to the interface region [14, 13]. However, this prevents simulating some interesting phenomena such as air bubbles inside the liquid. A multi-phase method was recently presented to take care of both fluids, achieving impressive bubbles movements [21]. As in this last work, we simulate bi-phase fluids. We present a different solution, based on the vorticity formulation, which solves the problem in an intuitive and efficient way.

Fluid-structure interaction is considered as a really tough problem by both the CG and applied mathematics communities. The most difficult issue is to compute the fluid's velocity at the objects boundaries while ensuring that no fluid penetrates any obstacle. [19] presented one of the first methods in CG for generating interaction forces between fluids and rigid solids. Recently, [5] provided a solution that treats the rigid body as a fluid and extracts its associated movement from the fluid's velocity. [20] presented the first technique able to handle the interactions with thin deformable and rigid objects represented by meshes. This method was further improved by [25] to allow the melting and burning of these solids. Closer to our vortex particles approach, [27] used the panel method to impose no-slip and no-through constraints on the fluid by emitting new particles at the solid's boundaries. This one-way interaction method, while well adapted for gases, is based on an explicit representation of the object which, if the density of particles was too low, does not prevent the flow from penetrating. Although achieving impressive results, these methods are sometimes hard to implement or require complex treatments where the object touches the fluid. Furthermore, they do not provide any validation of the correctness of the fluid's behaviors at the boundaries. Although in the spirit of the rigid fluid method [5], the solution we present avoids the explicit tracking of the fluid/body interface. Instead, we use a level set formulation on top of a vorticity creation algorithm to apply forces and account for the continuity of velocity at the fluid/body interface.

Overview

Our motivation for using vortex particles lies in the following features: they allow to concentrate numerical efforts on regions of interest and remain stable for large time-step values. Their main limitation, the associated computational cost, can be alleviated by using an auxiliary 3D grid. This is done while keeping the vorticity formulation of the equations and remaining physically accurate.

Our first contribution is to use vortex particles to simulate bi-phase fluids such as liquid and gas. As we show, the vortex formulation is very well adapted to tackle this problem, since the vortex particles are created near the interface between the two fluids. The second contribution is a novel algorithm to simulate the two-way interactions of the bi-phase fluid with animated rigid bodies. Our approach provides a physically sound and clear-cut fluid-solid model thanks to an algorithm that is both robust and easy to implement.

Section 2 reviews vortex methods. Section 3 explains how we use them to simulate bi-phase flow, such as water in contact with air. Section 4 focuses on the animation of solids interacting with the fluids. Section 5 validates the method through numerical comparisons with reference experiments from the computational fluid community. Lastly, Section 6 shows that the visually appealing results obtained in the CG community can be reproduced with an increased efficiency.

2 Vortex Particle Methods

In most fluid simulations, only a part of the flow has an interesting behavior. The vorticity formulation derived from the Navier-Stokes equations allows to focus the computational cost on the region of interest using vortex particles.
This section gives a quick overview of the vortex particles method, we refer the reader to [9] and the references therein for detailed numerical analysis and discussions of this method.

Definition

We start from the incompressible viscous 3D Navier-Stokes equations:

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = 0 \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0 \tag{2}$$

where $\mathbf{u}_t$ is the velocity's time derivative of the fluid's velocity field $\mathbf{u}$, $\nabla\cdot\mathbf{u}$ is the divergent of $\mathbf{u}$, p is the pressure and ν the kinematic viscosity of the fluid. The vorticity is defined as the curl of the velocity:

$$\omega = \nabla\times\mathbf{u}$$

(note that in the particular case of a 2D fluid, the vorticity is a scalar). Taking the curl of equations (1) and (2) (after some linear algebra) leads to the vorticity formulation of the Navier-Stokes equations:

$$\omega_t + (\mathbf{u}\cdot\nabla)\omega = (\omega\cdot\nabla)\mathbf{u} + \nu\Delta\omega. \tag{3}$$

Solving this single equation is equivalent to solving equations (1) and (2). The term $(\mathbf{u}\cdot\nabla)\omega$ represents the transport of the vorticity by the fluid's velocity, the left-most term of the right-hand side of equation (3)

represents the stretching (change of orientation) of the vorticity vector while the latter term represents the vorticity diffusion due to viscosity.
While it would be possible to solve this equation on a traditional 3D grid, the use of particles to transport vorticity permits to gain in precision and efficiency. Contrary to velocity which is mostly non zero in the whole domain, vorticity is localized in region of turbulences even if there can be high velocities everywhere. In consequence, by the use of particles, computational precision is focused where vorticity exists. Another advantage is that the advection of vorticity is not subjected to the time-step constraint inherent to Eulerian methods, which stands that $\delta t < |\mathbf{u}|_{max}$.
Vortex particle methods consist in representing the vector field ω by a set of particles:

$$\omega(\mathbf{x}) = \sum_p v_p \omega_p \zeta(\mathbf{x} - \mathbf{x}_p)$$

where $\mathbf{x}_p$, ω_p and v_p are respectively the location, strength and volume of particle p and ζ is a smooth distribution function, typically a Gaussian. Due to the incompressibility constraint, the volumes v_p remain constant. Rewriting equation (3) in a Lagrangian formulation, the particles' location and strength are integrated using:

$$\frac{\mathbf{D}\mathbf{x}_p}{\mathbf{D}t} = \mathbf{u}(\mathbf{x}_p, t), \tag{4}$$

$$\frac{\mathbf{D}\omega_p}{\mathbf{D}t} = (\omega \cdot \nabla)\mathbf{u} + \nu\Delta\omega, \tag{5}$$

where $\mathbf{D}q/\mathbf{D}t = \partial q/\partial t + (\mathbf{u} \cdot \nabla)q$ denotes the rate of change of a quantity q in the Lagrangian frame of a fluid element advected by the fluid. Particles' velocity and vorticity derivatives have to be determined in a self-consistent way from the vorticity field.

Coupling Vortex Particles with a Grid

As we plan to simulate fluids which may become turbulent and thus require many particles, we use both vortex particles and a 3D grid: firstly, spatial differentiations are cheaper on a grid; secondly it guaranties an approximately constant cost when solving the equations; lastly, it solves with no extra cost the problem of redistributing the particles over time. This last point is important because particles advected by the fluid will naturally tend to cluster in some area or to move away from each other thus leaving unresolved spaces between them.
A fast and accurate solution is to superpose a uniform grid on the particle distribution: at each time step particles are remeshed on that grid (thus, the particles' volume $v_p = h^3$ where h is the grid's cells spacing). Remeshing at every time-step can be interpreted as a class of high order finite-difference schemes, as long as it is done with high order redistribution schemes (see [1] for example).
Given the vorticity field, velocity is computed on the grid by means of a fast grid solver (in our experiments we used the FishPack library [31] in order to solve the following Poisson equation:

$$\Delta\psi = -\omega, \tag{6}$$

to obtain the so-called stream function ψ which is differentiated to obtain the velocities :

$$\mathbf{u} = \nabla \times \psi. \tag{7}$$

This solution to compute the velocity field from the vorticity is faster than the particle based solution used in [27] when the number of particles increases.
In practice particles are advanced with a second or fourth order Runge-Kutta method. During each substep velocity and vorticity's time derivative (the right hand side of the vorticity equation (3)) are interpolated from

the grid onto the particle locations. The accuracy of the overall algorithm heavily relies on the quality of the interpolation formulas used to remesh particles on the grid and to interpolate back fields at particle locations. It is common practice in CFD to use smooth interpolation formulas which preserve moments of order up to 2. The stencil on which particles are remeshed extends to the four nearest grid points in each direction. The cost of the algorithm thus scales linearly with the number of particles. A number of numerical validations on challenging test cases in CFD has allowed to attest the accuracy of remeshed particle methods. These studies are backed by a recent theoretical analysis which demonstrate that remeshed particle methods are equivalent to a class of high order finite-difference schemes not subject to CFL conditions ([11]).

3 Bi-phase fluids

We present here an intuitive method for simulating bi-phase fluids which benefits from the vortex particles formulation. Before explaining the particularity of bi-phase fluids, we first consider the vorticity problem in the presence of a fluid of variable density.

For variable density and viscosity flows, the vorticity equation (3) must be replaced by:

$$\begin{aligned}\omega_t + (\mathbf{u}\cdot\nabla)\omega = (\omega\cdot\nabla)\mathbf{u} + \nabla\cdot(\nu\nabla\omega) + (\nabla p\times\nabla\rho)\rho^{-2} \\ + \nabla\times(\rho\mathbf{g}) + \nabla\cdot[(\nabla\times\nu)\nabla\mathbf{u}]\end{aligned} \tag{8}$$

where $\mathbf{g}$ is the gravity field, ρ is the density and $\nu = \nu(\rho)$ is the variable kinematic viscosity. This equation has to be supplemented by a transport equation for ρ or for $\nabla\rho$.
This system is often simplified by assuming small density variations: in the so-called Boussinesq approximation, pressure and velocity terms in (8) disappear and we are left with

$$\omega_t + (\mathbf{u}\cdot\nabla)\omega = (\omega\cdot\nabla)\mathbf{u} + \nabla\cdot(\nu\nabla\omega) + \nabla\times(\rho\mathbf{g}). \tag{9}$$

It is important to note that, in the case of two fluids with constant viscosity and variable density, this equation shows that vorticity is produced where density gradients are located, that is at the interface between the fluids. Thus computation is localized in this narrow band which clearly reduces the theoretical cost compared to traditional methods. Accordingly to (9), the particle's vorticity change in equation (5) becomes:

$$\frac{\mathbf{D}\omega_p}{\mathbf{D}t} = (\omega\cdot\nabla)\mathbf{u} + \nabla\cdot(\nu\nabla\omega) + \nabla\times(\rho\mathbf{g}). \tag{10}$$

We now consider the simpler problem of a bi-phase fluid in a domain Ω where the density (resp. viscosity) takes only two different values: ρ_1 (resp. ν_1) in a domain Ω_1 and ρ_2 (resp. ν_2) in a domain Ω_2 ($\Omega = \Omega_1\cup\Omega_2$). We use a level set (noted ϕ) to capture the interface between those two domains:

$$\phi(\mathbf{x})\begin{cases} < 0 \text{ for } \mathbf{x}\in\Omega_1 \ , \\ = 0 \text{ for } \mathbf{x}\in\Omega_1\cap\Omega_2 \ , \\ > 0 \text{ for } \mathbf{x}\in\Omega_2 \ . \end{cases}$$

Typically, we initialize ϕ as a signed distance. This level set function satisfies a transport equation which can be solved either in its primitive form:

$$\phi_t + (\mathbf{u}\cdot\nabla)\phi = 0 \tag{11}$$

or in its gradient (vector) form:

$$(\nabla\phi)_t + (\mathbf{u}\cdot\nabla)\nabla\phi = -(\nabla\phi\cdot\nabla)\mathbf{u} \tag{12}$$

The latter equation is very similar to the vorticity equation and thus gradient of ϕ and ω are both located near the interface.

In this case particles carry strengths of vorticity and of $\nabla\phi$, from which we can deduce $\nabla \times \rho$ needed in equation (10) (see below).

But contrary to [8], we advect the level set ϕ on the grid with a semi-Lagrangian method. The surface tension is defined in terms of the curvature of ϕ (see [6]):

$$\tau\kappa\zeta(\phi)\nabla\phi.$$

where τ is the surface tension coefficient. This term is added to the vorticity equation (10). The level set is interpolated onto the particles during the remeshing step.
In order to be able to solve equation (10), we need to know the density and viscosity fields. They are defined in the whole domain thanks to a smoothed version H of the Heaviside function (with values 0 and 1 respectively for positive and negative arugments, and where ε is a small parameter of the order of the grid-size) based on the level set:

$$\rho = \rho_1 H(\phi/\varepsilon) + \rho_2(1 - H(\phi/\varepsilon)).$$

The same convention is used for viscosity. From this formulation, it is trivial to deduce the density gradient $\nabla\rho$: $\nabla\rho = (\rho_1 - \rho_2)\nabla\phi\zeta(\phi/\varepsilon)/\varepsilon$. Obtaining $\nabla \times \rho$ is straightforward.
As the computation of 8 is much more involved and expensive then equation 9, we have implemented the latter. Despite the fact that the Boussinesq approximation applies on small density variations, we have found that simulating a bi-phase fluid with a high difference of density (e.g. $\rho_{air} \approx 1$ and $\rho_{water} \approx 1000$) using this technique provides visually realistic results (for both fluid's dynamic and interface localization).

Algorithm

In summary, the following process is applied to advance from time $t_n = n\Delta t$ to t_{n+1}. Only steps 5 and 8 differ from the standard implementation we presented in Section 2:

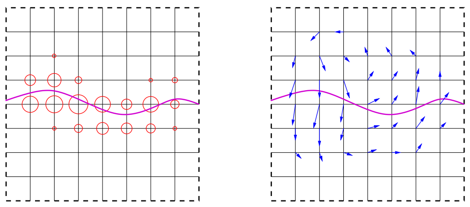
1. Find the stream function by solving equation (6) on the grid,
2. Compute the velocity $\mathbf{u}^{n+1}$ from equation (7),
3. Compute the stretching term $(\omega \cdot \nabla)\mathbf{u}$ on the grid,
4. Interpolate velocity and the stretching term on the particles,
5. Advance the particles: advect them with velocity (equation (4)) and vorticity change (equation (10)); Advect the fluid's level set with the velocity on the grid,
6. Distribute the particles onto the grid to get the advected vorticity,
7. Compute and integrate the viscosity term $\nabla \cdot (\nu\Delta\omega)$ on the grid,
8. Create fresh vortex particles, carrying ω and $\nabla\phi$, from the grid if vorticity is greater than a threshold.

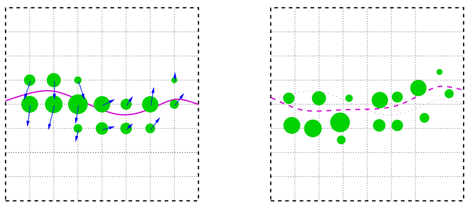
Note that we used a *viscous splitting algorithm* [7] to handle separatly the advection and the diffusion of the vorticity. Please refer to figure 1 for a visual explanation of the algorithm.

4 Coupling Fluids with Animated Solids

We propose a new approach which, with the adjoint power of the vortex particles and level set methods, can be used to compute fluid-solid interactions in an intuitive manner and for a low additional cost. Our approach is to consider the fluid-body system, during the interaction processing step, as two fluids of different densities and different constitutive laws.

Unlike the method in [5], our method computing the forces applied by the solid on the fluid (and vice-versa) does not require the explicit tracking of the interface. We instead use a level set method combined with a


(a) Steps 1 and 2: from vorticity to velocity (on the grid, vorticity is denoted by red circles). The magenta curve represents the interface between the two fluids.


(b) Steps 4 and 5: interpolation and advection of particles (particles' vorticity is denoted by green discs). The thick dashed (resp. thin dotted) magenta represents the advected level set (resp. level set at previous time).

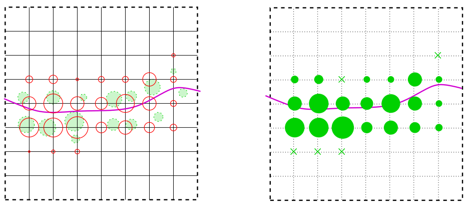

(c) Step 6 (left figure): particles (in transparent green on background) are distributed onto the grid. Step 8: remeshing step: create fresh particles where vorticity is strong enough (green crosses denote the fact that particles were not created because vorticity was too small).

Figure 1: Algorithm summary

vorticity creation penalization term that enforces velocity continuity at the fluid-solid interface.

In order to simplify the equations we assume here a single solid, the extension to multiple solids being straightforward.

At time zero the body is represented by the zero surface of a level set function noted here φ in order to not confuse the reader with the fluid's level set ϕ. Typically, we initialize the level set function as a signed distance to the body boundary, negative inside the solid. This is the only costly part of the method but it does not influence its efficiency since it is done as a precomputation.
We now denote by $\widetilde{\mathbf{u}}^{n+1}$ the velocities found in step 2 of the previous algorithm. After this step, we project the velocities in the grid onto rigid body velocity $\mathbf{U}^{n+1}$ inside the solid using the following formula:

$$\mathbf{U}^{n+1} = (\int \widetilde{\mathbf{u}}^{n+1}\, H(\varphi/\varepsilon)\, d\mathbf{x})/(\int H(\varphi/\varepsilon)\, d\mathbf{x}). \tag{13}$$

$\mathbf{U}^{n+1}$ stands for the mean velocity inside the solid. A similar equation defines the mean vorticity. Integration is performed on the whole domain covered by the solid). At the same time, we enforce continuity of velocities at the fluid-solid interface with a penalization technique that we will describe below.

At this stage, velocity and vorticity in the whole domain are obtained by the formula :

$$\mathbf{u}^{n+1} = \mathbf{U}^{n+1} H(\varphi/\varepsilon) + \widetilde{\mathbf{u}}^{n+1}(1 - H(\varphi/\varepsilon)). \tag{14}$$

A similar formula is used to obtain the vorticity field ω^{n+1}. Step 3 and 4 of the algorithm now use these fields for computing the stretching term and for the interpolations on the particles.
Step 5 is modified in two ways. First, we take care of the solid's density exactly in the same way as we have done for bi-phase fluids. Thus, particles now carry the gradient of a fluid with three different densities.

Secondly, we need to advect the solid's level set. This is done just after step 5 of the algorithm of Section 3. As the solid's velocity gives a translation and a rotation terms, the latter problem is easily solved by simply applying the rigid transformation between the initial level set position at time t_0 and its current position. In consequence, this scheme does not suffer from temporal diffusion and only implies a low diffusion depending on the order of the spatial interpolation. In our experiments, we use a simple first order interpolant.

It remains to explain the penalization method to enforce velocity continuity. We use the penalization method of [4]: assume we want to solve Navier-Stokes equations in a fluid domain outside a solid domain S, with velocity $\bar{\mathbf{u}}$ at the interface. The penalization model reads

$$\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} - \nu\Delta\mathbf{u} + \nabla p = \lambda\chi_S(\bar{\mathbf{u}} - \mathbf{u}), \tag{15}$$

where χ_S denotes the characteristic function of the solid domain S (1 inside and 0 outside) and $\lambda \gg 1$ is a penalization parameter. Adapting this method to the vorticity formulation for our problem leads to the substitution of the initial vorticity equation (3) by:

$$\begin{aligned}\omega_t + (\mathbf{u}\cdot\nabla)\omega = (\omega\cdot\nabla)\mathbf{u} + \nu\Delta\omega + \lambda\chi_S(\bar{\omega} - \omega)\\ + \lambda\delta_{\partial S}\mathbf{n}\times(\bar{\mathbf{u}} - \mathbf{u}),\end{aligned} \tag{16}$$

where $\delta_{\partial S}$ is the 1D Dirac mass supported by the solid boundary and $\mathbf{n}$ is the unit normal, pointed inward. Thus, to be consistent with the level set representation for solids, the final equation is:

$$\begin{aligned}\omega_t + (\mathbf{u}\cdot\nabla)\omega = (\omega\cdot\nabla)\mathbf{u} + \nu\Delta\omega + \lambda H(\varphi/\varepsilon)(\bar{\omega} - \omega)\\ + \lambda\frac{\zeta(\varphi/\varepsilon)}{\varepsilon}\nabla\varphi\times(\bar{\mathbf{u}} - \mathbf{u}).\end{aligned} \tag{17}$$

The penalization term tends to cancel the vorticity difference inside the body and adds vorticity at the fluid-solid interface. This term is computed on the grid after the computation of the fluid's velocity at step 2 of the algorithm. It is then interpolated on the particles in step 4. We use an explicit scheme to discretize it and choose $\lambda = 1/\delta t$ to enforce the stability of this scheme.

5 Validation

In this section, we validate the robustness and the physical accuracy of our method for computing the interactions between fluids and solids. We compare our results to the reference work from [22] obtain with the STAR-CD[1] software, a well-known software in the Computational Fluid Dynamic community, implementing fluid-body interactions from [15].
For this purpose, we first study the 2D case of a falling cylinder (which may be seen as a falling disk) in a fluid of constant density, submited to gravitational forces. We use the following parameters: boundary conditions are periodic in a square domain of size 1.0×1.0, cylinder's radius is 0.1, fluid's and cylinder's density are respectively $\rho_{fluid} = 1$ and $\rho_{cylinder} = 2$, kinematic viscosity ν is 0.001, gravity is $-1\mathbf{e}_y$ and $\varepsilon = 2\delta x = 2\delta y = h$ (ε is used for computing the smoothed Heaviside function and Dirac distribution values). Time steps (resp. cells size) are $\delta t_{128} = 0.01$ (resp. $h_{128} = 7.8125\cdot10^{-3}$), $\delta t_{256} = 0.0038$ (resp. $h_{256} = 3.90625\cdot10^{-3}$) and $\delta t_{300} = 0.0027$ (resp. $h_{300} = 3.33\cdot10^{-3}$) for grids of dimension 128×128, 256×256 and 300×300 respectively. The two last time steps are constrained by the diffusion stability condition for an exptheslicit scheme. Their simulation uses a time step of $\delta t_{Kern} = 0.0005$ and a 2D radial grid following the cylinder, 15800 cells, 80×200 nodes (radial$\times$tangential) and first cell at $\delta r = 0.65\cdot10^{-3}$. This type of grid is time consuming, but well adapted for a sharp resolution of this particular problem as it permits to focus the computations

[1]Get more informations on STAR-CD at http://www.cd-adapco.com/sitemap.html

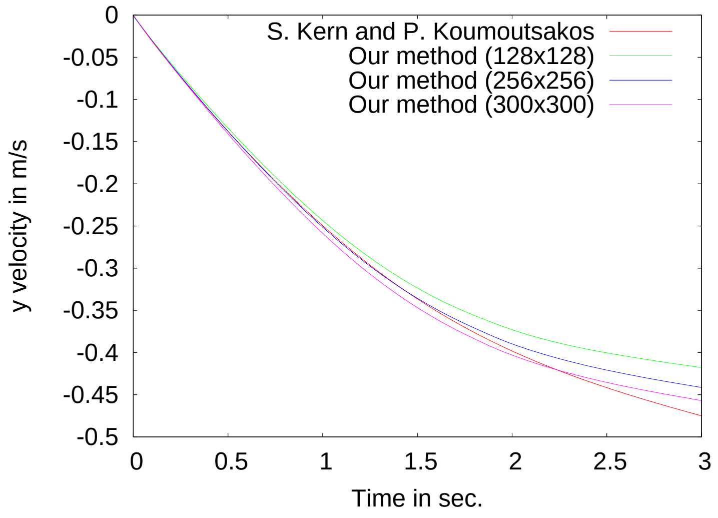


Figure 2: y velocities of the cylinder under gravity influence with a RK2 advection for our method, compared to the reference curve.

near the cylinder where precision is the most important. Both ours and Kern's scheme are 2^{nd} order in space and Kern's method is 1^{st} order in time while we are using a Runge-Kutta 2 method for advecting the particles.

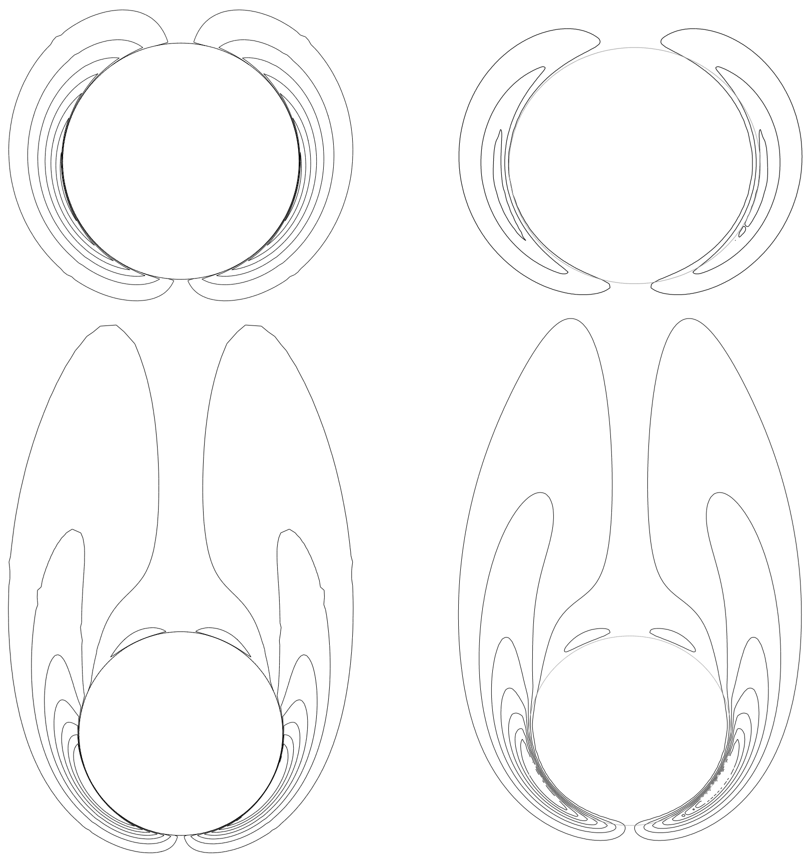
The y component of the velocity obtained with our method (see figure 2) is quite similar to the one obtained with [22] thus proving that our method well computes the solid's velocity coupled with the application of gravity to the solid. As the spatial precision increase, the curves converge to $v_y = -0.47$. These results are quite interesting for computer graphics since, even though we are using time steps 20 times bigger (for 128×128) than the other method and cells 10 times bigger, we still perform a robust treatment of boundaries.

The vorticity created around the cylinder is also a revealing quantity, both physically (for fluid's dynamics) and visually (for turbulences visualisation) important. We compare our vorticity field to the one obtained with Kern's method, isovalues of ω are shown in figure 3: vorticity creation and localization are similar. Notice that vorticity is not null inside our cylinder, this is because of the small width ε used to represent the solid. The vorticity trail created behind the cylinder is the signature of a good boundary treatment, we refer the reader to [23] for high-resolution simulations and vorticity analysis of the flow around a cylinder with constant velocity.

6 Implementation and results

We have implemented our algorithm with a Runge-Kutta 2 advection of vortex particles and second order in space differentiations. For the following examples, we have used a low order interpolation for advecting the level set.

In all the results we present now, we use a common gravity $\mathbf{g} = -10\mathbf{e}_z$, viscosity for water is $\nu_{water} = 1.0 \cdot 10^{-6}$, for air $\nu_{air} = 0.82 \cdot 10^{-6}$, water's density is set to 1 and air's density to 0.001. The surface tension coefficient is 10^{-6}. All our results were computed on an Opteron 2GHz with 2Go of memory.
We present different types of simulations, in figures 4 and 7 cups are falling in water, rendering is done with MAYA$^{\text{TM}}$. In figure 6 an initialy vertical volume of water is falling on several solids and in figure 5, we have



(a) Reference vorticity isovalues from [22].

(b) Vorticity isovalues using our method with a 300×300 grid.

Figure 3: Vorticity isovalues at time 0.5 (top) and 2.5 (bottom).

made two spheres fall in a tank of water, similarly to [5]. For those last two examples, the rendering was done with OpenGL. A summary of the performance of the method is provided in table 1.

In the first example, shown in figure 4, the cup has density $\rho_{cup} = 1.5$. We see that, when entering the liquid, the cup encapsulates air which makes it roll over after a while, as the captured volume tends to ride up due to gravitational forces. One can see in the last two images of the clip, the big bubble escaping and merging back with the air. Figure 7 shows similar simulations with the same cup starting with different orientations. One can observe the effect of surface tension and air capturing on the dynamic of the cups and fluids. The grid's dimension is 100×100 and the time step used for the two simulations was 0.01. The total time spent for 1300 iterations per simulation was approximately 3 hours.
As a second example, we took a "wall" of water which, under gravity, falls down and breaks the construction (see 6). One can see in the third image the creation of a wave on the top of the water surface. This wave then breaks and merges with the water under it.
Figure 5 shows two spheres falling in water, this case was inspired from [5] and we are using the same grid resolution. While obtaining similar dynamics for solids and fluids (the turbulent flow is observable by looking at the white "dust" particles), we have computed this sequence with a smaller time step of 0.01, 400 iterations were performed in 40 minutes which represent approximately a gain of 58.

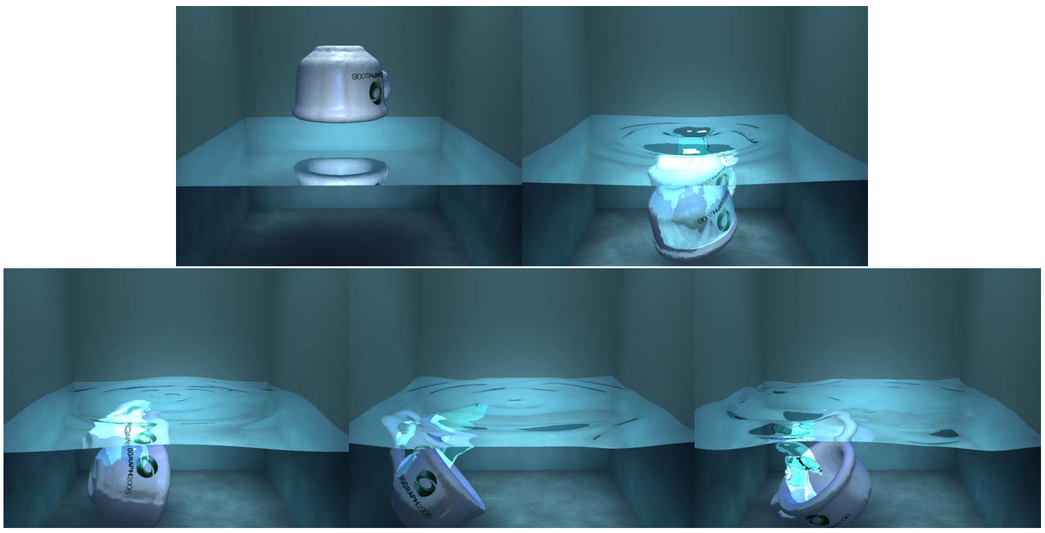


Figure 4: Interactions of a rigid object with both air and water allows to simulate complex movements.

Sequence	Grid Resolution	Duration	Time Steps	CPU Time / Step
Cup 1	$100 \times 100 \times 100$	10 s	1300	8.60 s
Cup 2	$64 \times 64 \times 64$	10 s	1000	2.44 s
Spheres	$68 \times 24 \times 292$	4 s	400	5.96 s
Pyramid	$80 \times 80 \times 80$	3 s	600	12.4 s

Sequence	Stream Solve	Particules	Grid	Level-Set	Rigids Coupling	Rigid Solver	Surface Tension	Other
Cup 1	32.4 %	4.93 %	18.6 %	23.3 %	8.28 %	3.08 %	5.78 %	3.66 %
Cup 2	32.1 %	4.60 %	19.5 %	21.0 %	8.17 %	5.38 %	5.58 %	3.69 %
Spheres	42.2 %	10.32 %	21.18 %	21.2 %	8.35 %	1.23 %	-	3.04 %
Pyramid	33,1 %	2.43 %	6.81 %	25.5 %	24.5 %	1.06 %	2.05 %	4.50 %

Table 1: Performance measurements. *Particles* stands for particle RK2 advection and distribution, *Grid* for computation on grid (steps 3 and 7), *Level-Set* for level set advection, diffusion and reinitialiation.

One can see in the performance synthesis (table 1) that the a third of the computational cost is due to the solving of the Poisson equation, while only a small amount of time is dedicated to the particles. The second costly part of the algorithm stands for the level set operations. The rigid/fluid coupling takes a little more time to compute than grid-based finite differences computations such as the stretching or the viscous term computation.

7 Conclusion and outlook

We have presented the first vortex particle method that handles a bi-phase fluid interacting with animated rigid bodies. Fully based on the vorticity formulation of the Navier-Stokes equations, our method takes benefits of the advection of the vortex particles to achieve precision and robustness for large time steps. Meanwhile, it relies on a 3D grid for efficiently computing some of the terms and redistributing the particles at each time

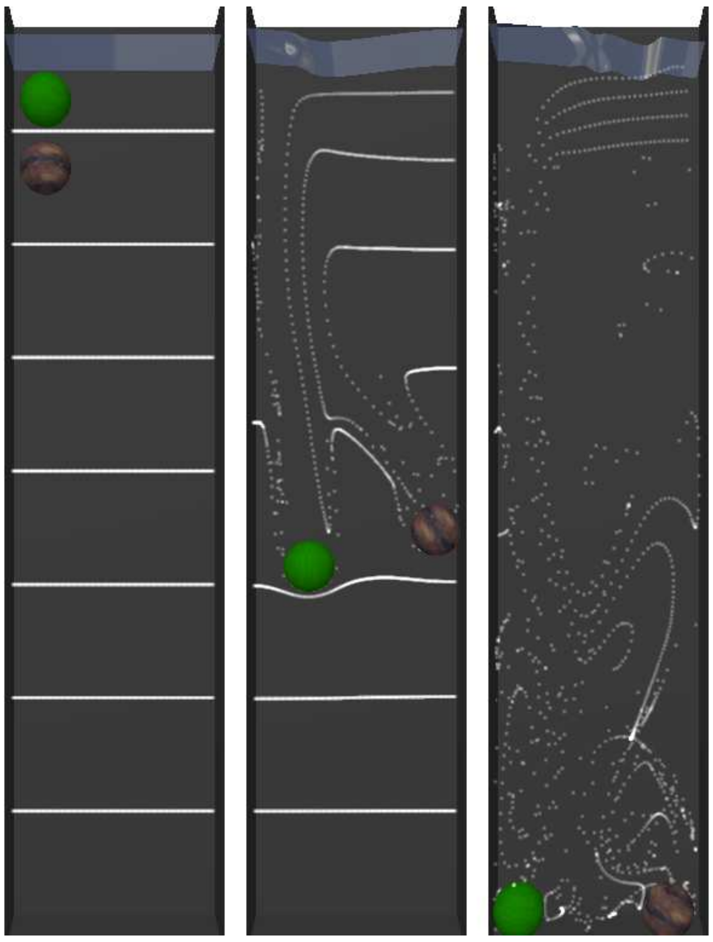


Figure 5: Two sinking spheres showing drafting, kissing and tumbling effect [5], reproduced by our method.

step. Both the bi-phase fluid and the moving solids are treated using the vorticity equations for a fluid of variable density, the associated level sets being advected by the fluid. Moreover, the rigid motion of the solid is accurately computed by ensuring the continuity of velocities with the surrounding fluids.

As our results show, this method brings the two benefits of being a numerically accurate and an efficient way to simulate fluids interacting with rigid bodies.

Our level set approach for modeling the interface between the fluids and immersed bodies could be extended to other cases. One can for instance enrich the type of physics underlying the fluid-body interaction, while keeping the simplicity provided by the immersed interface approach and its particle discretization. Using the framework developed in [10] it is actually possible to handle flexible bodies interacting with fluids, a possibility that we plan to explore. Another idea for future work is to implement multi-level particle methods in our fluid-body models. One may envision two types of multi-level discretization. One would be to capture the interfaces, that is the level-set functions, and the flow, that is the vorticity with different grid resolution. Using finer resolution for the level-set function should allow to capture finer details on the interfaces with little computational overhead. Indeed it should be emphasized that, since the time-step of particle methods is only constrained by the amount of strain in the flow, refining the resolution for the level-set functions does not result in a smaller time-step. Alternatively, one may consider using a full (for the flow and the interfaces) multi-level vortex method, in the sprit of AMR methods, along the lines of [3]. Fluid-body interactions is a

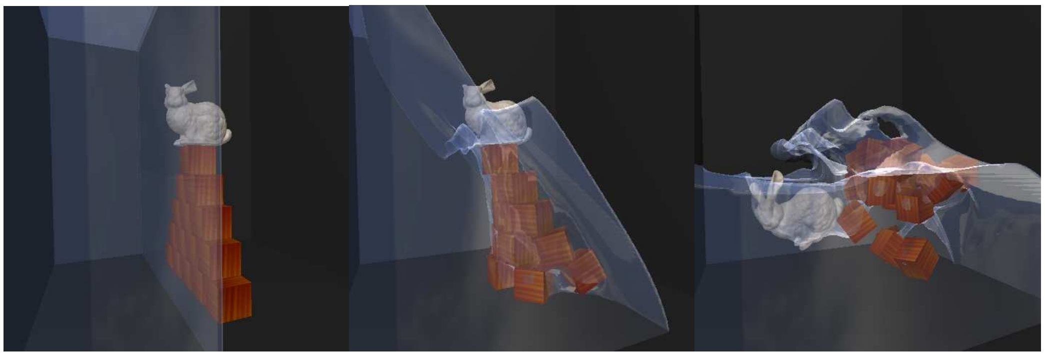


Figure 6: A pyramid encountering a wall of water.

challenging problem to clarify the respective benefits of these two approaches.

References

[1] A. K. Chaniotis, D. P., and Koumoutsakos, P. Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. *Journal of Computational Physics 1982*, 1 (2002), 67–90.

[2] Angelidis, A., and Neyret, F. Simulation of smoke based on vortex filament primitives. In *Proc. of the ACM SIGGRAPH'05/EG Symp. on Comp. Anim.* (2005).

[3] Bergdorf, M., Cottet, G.-H., and Koumoutsakos, P. Multilevel adaptive particle methods for convection-diffusion equations. *SIAM Multiscale Modeling and Simulation 4*, 1 (2005), 328–357.

[4] Bruneau, C. H., and Fabrie, P. New efficient boundary conditions for incompressible navier-stokes equations : a well-posedness result. *Math. Mod. and Num. Analysis 30*, 7 (1996), 815–840.

[5] Carlson, M., Mucha, P., and Turk, G. Rigid fluid: Animating the interplay between rigid bodies and fluid. *Proc. of ACM SIGGRAPH 04 23* (2004), 377–384.

[6] Chang, Y. C., Hou, T. Y., Merriman, B., and Osher, S. A level set formulation of eulerian interface capturing methods for incompressible fluid flows. *Journal of Computational Physics 124* (1996), 449–464.

[7] Chorin, A. J., Hughes, T. J. R., McCracken, M. F., and Marsden, J. E. Product formulas and numerical algorithms. *Comm. Pure Appl. Math 31* (1978), 205–256.

[8] Cottet, G.-H. Multi-physics and particle methods. In *Computational Fluid and Solid Mechanics 2003* (Boston, USA, June 2003), K. Bathe, Ed., Elsevier, pp. 1296–1298.

[9] Cottet, G.-H., and Koumoutsakos, P. *Vortex methods: Theory and practice*. Cambridge University Press, 2000.

[10] Cottet, G.-H., and Maitre, E. A level-set method for fluid-structure interactions with immersed surfaces. *Mathematical Models and Methods in the Applied Sciences 16* (2006), 415–438.

[11] Cottet, G.-H., and Weynans, L. Particle methods revisited: a class of high-order finite-difference scheme. *C.R. Acad. Sci. Paris 343* (2006), 51–56.

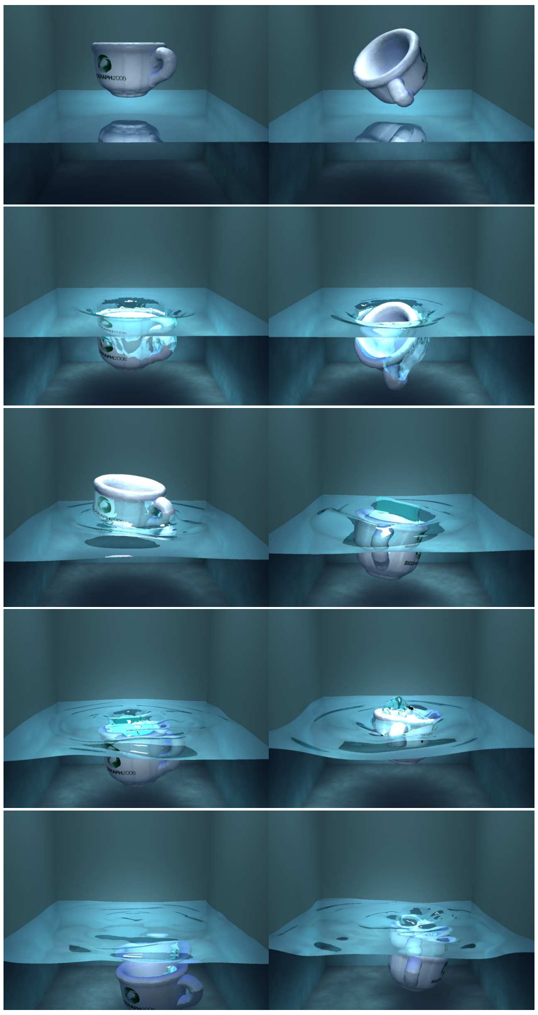


Figure 7: Two cups enter the fluid with different orientation.

[12] Desbrun, M., and Cani, M.-P. Active implicit surface for animation. In *Graphics Interface* (1998), pp. 143–150.

[13] Enright, D., Losasso, F., and Fedkiw, R. A fast and accurate semi-lagrangian particle level set method. *Computers and Structures 83* (2005), 470–490.

[14] Enright, D., Marschner, S., and Fedkiw, R. Animation and rendering of complex water surfaces. In *Proc. of ACM Tran. Graph. SIGGRAPH 02* (2002), pp. 736–744.

[15] Farhat, C., and Lesoinne, M. Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. *Comp. Meth. Appl. Mech. 182* (2000), 499–515.

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

[17] Foster, N., and Metaxas, D. Realistic animation of liquids. *Graph. models and image processing 58*, 5 (1996), 471–483.

[18] GAMITO, M. N., LOPES, P. F., AND GOMES, M. R. Two dimensional simulation of gaseous phenomena using vortex particles. In *Proc. of the EG'95 Workshop on Comp. Anim. and Sim.* (1995), pp. 3–15.

[19] GÉNEVAUX, O., HABIBI, A., AND DISCHLER, J.-M. Simulating fluid-solid interaction. In *Graphics Interface* (June 2003), pp. 31–38.

[20] GUENDELMAN, E., SELLE, A., LOSASSO, F., AND FEDKIW, R. Coupling water and smoke to thin deformable and rigid shells. *Proc. of ACM Trans. Graph. SIGGRAPH 2005* (2005), 973–981.

[21] HONG, J.-M., AND KIM, C.-H. Discontinuous fluids. In *Proceedings of ACM SIGGRAPH 2005* (2005).

[22] KERN, S., AND KOUMOUTSAKOS, P. Simulations of optimized anguilliform swimming. *(submitted work)* (2005).

[23] KOUMOUTSAKOS, P., AND LEONARD, A. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech. 296* (1995), 1–38.

[24] LINDSAY, K., AND KRASNY, R. A particle method and adaptive tree-code for vortex sheet motion in 3d flow. In *Journal of Comp. Physics* (2001), vol. 172, pp. 879–907.

[25] LOSASSO, F., IRVING, G., GUENDELMAN, E., AND FEDKIW, R. Melting and burning solids into liquids and gases. *IEEE Transactions on Visualization and Computer Graphics* (2005).

[26] MÜLLER, M., SOLENTHALER, B., KEISER, R., AND GROSS, M. Particle-based fluid-fluid interaction. In *Proc. of the ACM SIGGRAPH'05/EG Symp. on Comp. Anim.* (2005), pp. 237–244.

[27] PARK, S. I., AND KIM, M. J. Vortex fluid for gaseous phenomena. In *Proc. of the ACM SIGGRAPH'05/EG Symp. on Comp. Anim.* (August 2005).

[28] PREMOZE, S., TASDIZEN, T., BIGLER, J., LEFOHN, A., AND WHITAKER, R. Particle based simulation of fluids. In *Comp. Graph. Forum (EuroGraphics proc.)* (2003), vol. 22, pp. 401–410.

[29] SELLE, A., RASMUSSEN, N., AND FEDKIW, R. A vortex particle method for smoke, water and explosions. In *Proceedings of ACM Trans. Graph. (SIGGRAPH 2005)* (2005), vol. 24, pp. 910–914.

[30] STAM, J. Stable fluids. In *Proc. of SIGGRAPH 99* (1999), pp. 121–128.

[31] SWARZTRAUBER, P., AND SWEET, R. Fishpack: Efficient fortran subprograms for the solution of elliptic partial differential equations. Tech. Rep. TN/IA-109, Nat. Cent. for Atmosph. Research, July 1975.