Flow simulations using particles - Bridging Computer Graphics and CFD

Petros D. Koumoutsakos, Georges-Henri Cottet, Diego Rossinelli

FLOW SIMULATIONS USING PARTICLES
Bridging Computer Graphics and CFD

Petros Koumoutsakos
COMPUTATIONAL SCIENCE AND ENGINEERING LABORATORY
ETH ZÜRICH, SWITZERLAND
Georges-Henri Cottet
LABORATOIRE JEAN KUNTZMANN
UNIVERSITÉ DE GRENOBLE, FRANCE
Diego Rossinelli
COMPUTATIONAL SCIENCE AND ENGINEERING LABORATORY
ETH ZÜRICH, SWITZERLAND

CONTENTS

For the latest version of these notes, animations and software, please visit :
http://www.cse-lab.ethz.ch/teaching/classes/mulsup.html

1 INTRODUCTION

The simulation of the motion of interacting particles is a deceivingly simple, yet powerful and natural method for exploring and animating flows in physical systems as diverse as planetary dark accretion and sea waves, unsteady aerodynamics and nanofluidics.

Particle methods have been advocated for efficient simulations of multiphysics phenomena in complex deforming computational domains in several fields of science ranging from astrophysics to solid mechanics (see [63, 54] and references therein). In computer graphics, particle systems were introduced with the pioneering work of Reeves [74] and have continued over the years to be the backbone of several impressive animations [42, 88, 87, 73, 71, 81]. In the CFD community, particle methods were the first techniques to ever be used for the numerical simulation of fluids, starting with the pioneering calculations by hand of the evolution of a vortex sheet by Rosenhead [76] and continuing with the works of Chorin [16] and Leonard [52]. In Direct Numerical Simulations of compressible and incompressible flows it has been shown that caution must be exercised when using a grid free method [35] and that regularization of the particle locations is necessary in order for the method to converge [47] to the solution of the equations that have been discretized. At the same time in graphics the loss of accuracy of the method in terms of incompressibility [29, 28, 10] and conservation of geometrical constraints [25] may affect the visual realism of the flow.

Particles can be viewed as objects carrying a physical property of the flow, that is being simulated through the solution of Ordinary Differential Equations (ODE) that determine the trajectories and the evolution of the properties carried by the particles. Particle methods amount to the solution of a system of ODEs :

$$\frac{d\mathbf{x}_p}{dt} = \mathbf{u}_p(\mathbf{x}_p, t) = \sum_{q=1}^{N} K\left(\mathbf{x}_p, \mathbf{x}_q; \mathbf{w}_p, \mathbf{w}_q\right) \tag{1}$$

$$\frac{d\mathbf{w}_p}{dt} = \sum_{q=1}^{N} F\left(\mathbf{x}_p, \mathbf{x}_q; \mathbf{w}_p, \mathbf{w}_q\right) \tag{2}$$

where $\mathbf{x}_p, \mathbf{u}_p$ denote the locations and velocities of the N particles, $\mathbf{w}_p$ denote particle properties (such as density, temperature, velocity, vorticity) and K, F represent the dynamics of the simulated flow. Particle simulations are well suited to a Lagrangian formulation of the continuum equations, as in the vorticity-velocity or the velocity-pressure formulation of the Navier-Stokes equations, resulting respectively in techniques such as Vortex Methods and Smooth Particle Hydrodynamics. Particle methods such as Vortex Methods (VMs) and Smooth Particle Hydrodynamics (SPH) present an adaptive, efficient, stable and accurate computational method for simulating continuum flow phenomena and for capturing interfaces. At the same time, particle methods encounter difficulties in the accurate treatment of boundary conditions, while their adaptivity is often associated with severe particle distortion that may introduce spurious scales.

We wish to note that particle formulations of fluid mechanics phenomena can be extended also in the mesoscale and the nanoscale regimes with techniques such as Molecular Dynamics (MD) and Dissipative Particle Dynamics (DPD) inherently linked to the discrete representation of the underlying physics. In fact particle methods enable a unifying formulation that can enable systematic and robust multiscale flow simulations as

Figure 1: Particle Systems in Computer Graphics : Explosion on a planet [74](left), Artificial plant growth [42](middle), Vortex field [87](right).

well as simulations continuum and discrete systems [48] Indeed a remarkable feature of particle methods is that their computational structure involves a large number of common abstractions that help in their computational implementation, while at the same time particle methods are distinguished by the fact that they are inherently linked to the physics of the systems that they simulate.

Returning to the realm of continuum flows, particle methods, applied to the solution of convection-dominated problems in the context of Vortex Methods and Smooth Particle Hydrodynamics, enjoy an automatic adaptivity of the computational domain as dictated by the convective map. The field quantities can always be reconstructed by a linear superposition of the individual fields carried by the particles. In smooth particle methods - as opposed to point particle methods - each particle is associated with a smooth core function, or 'blob' enabling the smooth representation of the field quantities and efficient discretizations of the governing equations. The Lagrangian form of particle methods avoids the explicit discretization of the convective term in the governing transport equations and the associated stability constraints. The particle positions are modified according to the local flow map, making the method self-adaptive. This adaptation however comes at the expense of the regularity of the particle distribution as particles move in order to adapt to the gradients of the flow field. Particle regularity can be enforced by remeshing the particle locations on a regular grid as it is discussed in this session.

A key message of this class, is that once the smooth particles cease to overlap the vortex methods do not solve the equations they have discretized. In order to maintain the accuracy of the method we introduce a mesh that helps the regularization of the particle locations as discussed in the following. This may seem at first as introducing additional dissipation to the method and limiting its capability to handle complex geometries. We will demonstrate that the dissipation introduced by remeshing can be controlled and reduced below other discretization errors and in addition it enables the formulation of a consistent multiresolution particle framework. Furthermore we demonstrate that remeshing, allows the effective handling of complex geometries. We discuss penalisation and immersed boundary techniques which along with body-fitted particle systems present an arrays of effective ways for resolving boundaries in particle systems. In this class we emphasize the immersed boundary approach to model flows in complex geometries. Boundary conditions are indeed recognized as a source of technical difficulties for particle methods. This is a field where ad-hoc solutions or recipes are often used leading to algorithms that are very sensitive to a number of parameters. Immersed boundary methods can be derived from straightforward numerical approximations and

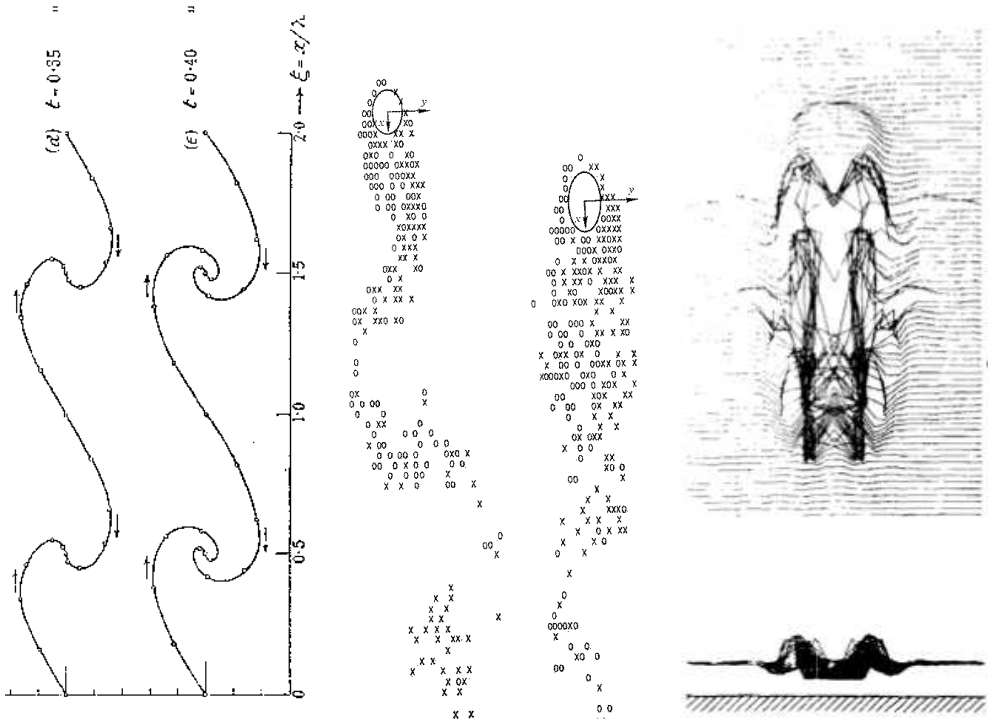

Figure 2: Particle Systems in CFD: Hand Calculations of a vortex sheet [77](left), flow past a circular cylinder [16](middle), Vortex lines in a 3D boundary layer [52](right).

lead to relatively simple algorithms. They also offer enough flexibility to adress fluid-structure interaction systems and multiphase fluids.
The course notes are structured as follows : In Section 2 we provide an overview of Particle Methods with an emphasis on their use for Flow Simulations. In Section 3 we introduce the remeshed particle methods. We discuss in section 4 how remeshing not only does not hinder the adaptivity of the method but enables multiresolution particle methods. In Section 5 we discuss the use of particle methods for multiphysics and in complex geometries. We conclude in Section 6 with the implementation of particle methods on GPUs demonstrating the possibility of translating the present framework into real time simulations.

2 PARTICLE METHODS AND FLOW SIMULATIONS

The flows we describe in this class can be effectively cast in the following form:

$$\frac{\partial u}{\partial t} + \text{div } (\mathbf{U}u) = F(u, \nabla u, \cdots) \tag{3}$$

where u is a scalar flow property (e.g density) or a vector (e.g. momentum) advected by the velocity vector field $\mathbf{U}$. Equation (3) is an advection equation in conservation form in the sense that it if $F = 0$ and no flow comes form the boundaries of the computational box, the scalar property is conserved as :

$$\frac{d}{dt} \int u \, dx = 0.$$

We note that the right hand side F can take various forms involving derivatives of u and depends on the physics of the flow systems that is being simulated. An example for F is the diffusion term ($F(u, \nabla u, \cdots) = \nabla^2 u$). The velocity vector field ($\mathbf{U}$) can itself be a function of u, which leads to *nonlinear* transport equations. A number of examples of different $F, \mathbf{U}, \mathbf{u}$ are given in the following sections.
For simplicity, we first consider the case $F \equiv 0$. The conservative form of the model can be translated in a Lagrangian framework by sampling the mass of u on individual points, or **point particles** whose locations can be defined with the help of Dirac δ-functions. Hence when u is initialized on a set of point particles it maintains this descriptions, with particle locations obtained by following the trajectories of the flow:

$$u(x,t) = \sum \alpha_p \delta(x - x_p(t)) \tag{4}$$

where

$$\frac{dx_p}{dt} = \mathbf{U}(x_p, t). \tag{5}$$

In practice this system of differential equation is solved by a time-discretization method (sometimes called in this context a *"particle pusher"*).

2.1 Smooth Particles for Simulations of Continuum Systems

The point particle approximations have useful computational features as they provide an *exact* representation of convection effects, a feature that has been extensively used in the computer animations of fluids (for example in : [29, 10]). At the same time the point particle approximations need to be enhanced in order to recover continuous fields (see [21,71] and references there in). Among the different approaches of recovering continuous fields from point samples, for reasons that will become evident below, we consider the approach of regularising their support, replacing δ by a smooth *cut-off* function which has the same mass (unity) and a small support:

$$\delta(x) \simeq \zeta_\epsilon(x) = \epsilon^{-d} \zeta(\frac{x}{\epsilon}) \tag{6}$$

where d is the dimension of the computational space and $\epsilon << 1$ is the range of the cut-off. In most cases, on uses a function with radial symmetry A typical and often used example is the Gaussian

$$\zeta(x) = \pi^{-d/2} \exp(-|x|^2)$$

Other functions that "resemble" more to the dirac mass, in the sense that they have the same values (zero) for higher moments, can be constructed in a systematic way, a topic that is beyond the scope of these class notes. We refer to [21] for further discussions on this issue.

The particle representation formula (4) then becomes a *blob* representation

$$u(x,t) \simeq u_\epsilon(x,t) \;=\; \sum \alpha_p \zeta_\epsilon(x - x_p(t)). \tag{7}$$

Most importantly, regularization can be used to compute local (e.g. algebraic functions) or non-local (in particular derivatives of any order) quantities based on u. We will show later a number of examples of how this principle is used in practice.

Using smooth particles to solve 3 in the general case ($F \neq 0$), one further needs to increment the particle strength by the amount that is dictated from the right hand side F. For that purpose, local values of F at particle locations multiplied by local volumes around particles are required. The local values of F can always be obtained from regularization formulas (7). The volumes v of the particles are updated using the transport equation

$$\frac{\partial v}{\partial t} + \text{div}\,(\mathbf{U}v) = -v\,\text{div}\,\mathbf{U} \tag{8}$$

The particle representation of the solution is therefore given by (4), (5) complemented by the differential equations

$$\frac{dv_p}{dt} = -\text{div}\,\mathbf{U}(x_p,t)\,v_p\;,\;\; \frac{d\alpha_p}{dt} = v_p F_p. \tag{9}$$

In (4), particle masses represent local integrals of the desired quantity around a particle. Typically, if particles are initialized on a regular lattice with grid size Δx, one will set $x_p^0 = (p_1\Delta x, \cdots, p_n\Delta x)$ and $\alpha_p = (\Delta x)^d\, u(x_p, t=0)$. One may also write the weight of the particles as the product of the particle strength and particle volume that are updated separately :

$$\alpha_p = v_p\, u_p. \tag{10}$$

2.2 Examples: SPH and Vortex Methods

Two of the most widely used particle methods for flow simulations are Smoothed Particle Hydrodynamics (SPH) and Vortex Methods (VM). We outline here the key elements of these methods with an emphasis on their underlying principles. Extensive reviews of these methods can be found in [63,48].

2.2.1 Compressible Flows and SPH

The method of SPH was introduced for the study of gas dynamics as they pertain to astronomical systems [56,31]. In these notes we introduce for simplicity the numerical formulation of Smoothed Particle Hydrodynamics (SPH), using the Euler equations for gas dynamics in one space dimension. The equations of gas dynamics for the density ρ and the velocity u can be cast in the following form

$$\frac{\partial \rho}{\partial t} + u\,\frac{\partial \rho}{\partial x} \;=\; -\rho\,\frac{\partial u}{\partial x} \tag{11}$$

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} \;=\; \frac{\partial \tau}{\partial x} \tag{12}$$

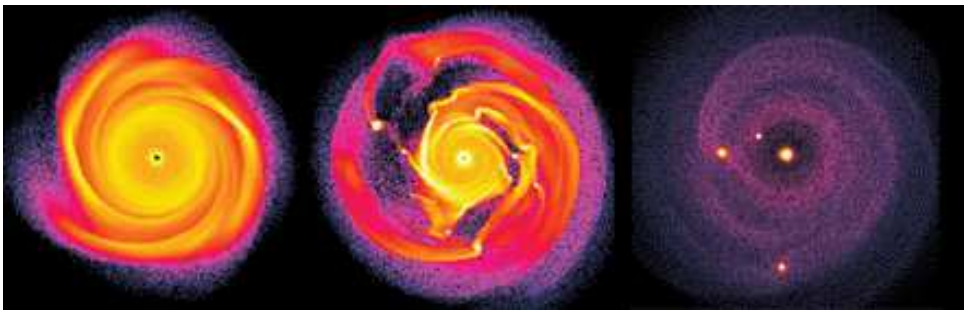

Figure 3: SPH Simulations of protoplanetary disk formation [44].

where τ denotes the fluid stress. This system needs a closure to determine the fluid stress, which in turn requires an energy equation and a constitutive law for the gas under consideration. In that case, particles weights are written using (10). To obtain local values (ρ_p, u_p) of density and velocities, the right hand side of (11),(12) are computed from regularized evaluations of the velocity. Unfortunately, there is a discrepancy between the notation used in the SPH literature and the rest of the particle literature (or the other way around !). In SPH related works the cut-off function are denoted by W, the cut-off range is h instead of ϵ, and $\zeta_\epsilon(x)$ becomes $W(x,h)$. With these notations the divergence of the velocity is given by

$$\frac{\partial u}{\partial x}(x_p) = \sum_q v_q(u_q - u_p)\frac{\partial W}{\partial x}(x_p - x_q, h)$$

The particle representation for u and ρ is therefore given by

$$\rho(x,t) = \sum_p v_p \rho_p \delta(x - x_p) \tag{13}$$

$$u(x,t) = \sum_p v_p \delta(x - x_p), \tag{14}$$

where the weights of the particle are obtained by solving the differential equations :

$$\dot{v}_p = v_p \sum_q v_q(u_q - u_p)\frac{\partial W}{\partial x}(x_p - x_q, h) \tag{15}$$

$$\dot{\rho}_p = -\rho_p \sum_q v_q(u_q - u_p)\frac{\partial W}{\partial x}(x_p - x_q, h) \tag{16}$$

$$\dot{u}_p = \sum_q v_q(\tau_q - \tau_p)\frac{\partial W}{\partial x}(x_p - x_q, h). \tag{17}$$

This system has to be closed by an additional energy equation.

Note that, in the expression giving the divergence, we have subtracted from the expected expression $\sum_q v_q u_q \frac{\partial W}{\partial x}(x_p - x_q, h)$ the term $\sum_q v_q u_p \frac{\partial W}{\partial x}(x_p - x_q, h)$. In the limit of an infinite number of particles, this term vanishes since it tends to the integral of the the function $\frac{\partial W}{\partial x}$. Its contribution is to maintain for a finite number of particles the conservativity of the method. This issue of conservation (of mass, energy ..) is indeed a central issue in SPH methods and has been the subject of many works. The approach to

derive schemes that have these properties is often very closely related to deriving particles dynamics that mimic at the discrete level the underlying physics. For these reasons, SPH simulations are very appealing and often give qualitatively satisfying results even with a rather small degrees of freedom, and this explains their popularity in the graphics community and the animation industry (see Fig.4 and [55])

However, as one may wish to increase the reliability, and not only the visual plausibility of the simulations, the use of SPH for flow simulations raise some serious concerns [11]. We will revisit this key issue in Section 3.

Figure 4: Two-way coupled SPH and particle level set fluid simulation [55].

SPH methods have been originally designed for compressible flows with an emphasis on gas dynamics of astrophysical systems (see Fig.3). In the case of incompressible flows, the need to define ad-hoc constitutive law, they have to resolve unphysical waves along with the related numerical stability constraints and the necessary artificial viscosity. In recent years a number of efforts [41,26] have been presented in order to introduce incompressibility into SPH formulations with variable degrees of success. Note that in incompressible formulations a Poisson equation is necessary in order to ensure the incompressibility of the flow and efforts to bypass this relatively expensive computation have to maintain the right balance of accuracy and computational cost. Alternatively, vortex methods based on the vorticity form of the incompressible Navier-Stokes equations can be considered as more suitable numerical tools for these flows.

2.2.2 INCOMPRESSIBLE FLOWS AND VORTEX METHODS In Vortex Methods particles discretize the velocity-vorticity formulation of the Navier-Stokes equations which takes the following form

$$\frac{\partial \omega}{\partial t} + (u \cdot \nabla)\,\omega \;=\; (\omega \cdot \nabla)\,u + \nu \Delta \omega \tag{18}$$

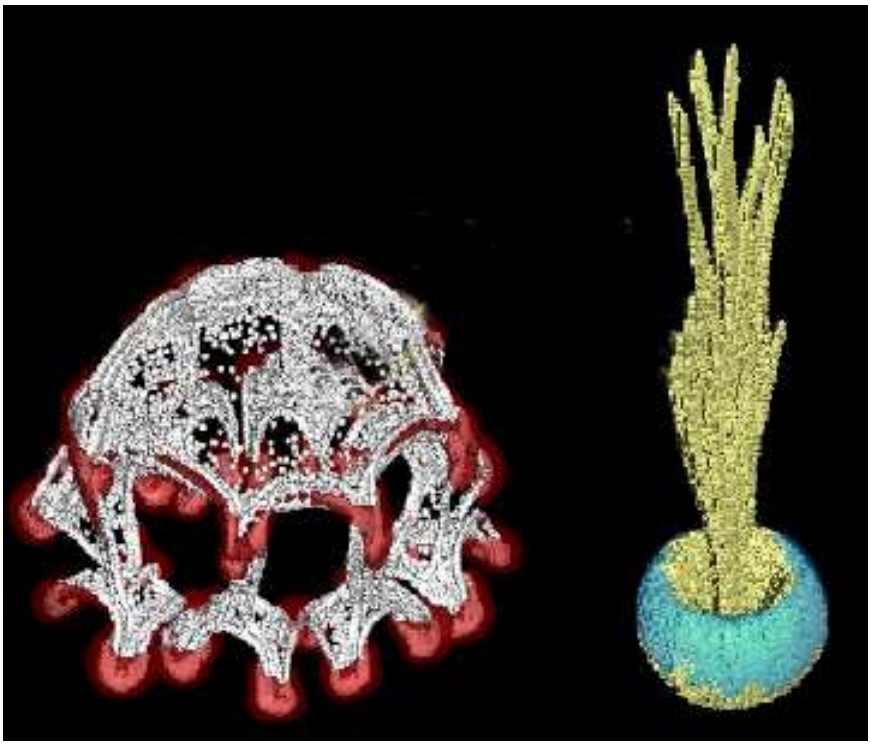

Figure 5: Vortex Method simulations of solid particle ladden flows with two way coupling [97].

The vorticity field ω is related to the velocity field u by $\omega = \nabla \times u$. This equation together with the incompressiblity constraint

$$\text{div } u = 0 \tag{19}$$

and suitable boundary conditions allows to express the velocity in terms of the vorticity (we will come back later to this important point). Equation (18) thus appears as a nonlinear advection-diffusion equation for the vorticity. The diffusion term is just one particular instance of the right hand side F in (3). One particular and appealing feature of this form of the flow equations is that the divergence free constraint is not directly involved in the transport equation. Also, the equation points straight to the very nature of vorticity dynamics in incompressible flows: transport and dissipation (in 2 and 3D) , change of orientation and strengthening (in 3D only, since in 2D the term $(\omega \cdot \nabla)\, u$ vanishes). A vortex particle method consists of sampling vorticity on points that follow the flow field:

$$\omega(x,t) = \sum_p \omega_p \delta(x - x_p) \tag{20}$$

with

$$\frac{dx_p}{dt} = u(x_p,t) \tag{21}$$

$$\frac{d\omega_p}{dt} = [\nabla u(x_p,t)]\,\omega_p + \nu\,\Delta\omega(x_p). \tag{22}$$

In the above equation $[\nabla u(x_p,t)]$ is the tensor made of all derivatives of the velocity. The way to compute this term as well as the diffusion term remains to be specified. Vortex

Methods are distinct from SPH, in that they enforce explicitly the incompressibility of the flow while in addition they resolve gradients of the flow field rather than primitive variables. Furthermore they use computational elements only where the vorticity field is non-zero which at times can be only a small fraction of the domain, thus providing increased efficiency (Fig.5). At the same time they require the solution of a Poisson equation to recover the velocity field from the vorticity, while their implementation in the presence of boundaries requires the reformulation of the velocity boundary conditions. A monograph by the authors [21] discusses several of the methodological developments of Vortex Methods.

Here we wish to mention that another particle possible approach to 3D incompressible flows is possible using filaments instead of point particles. A filament method will consist of tracking markers on lagrangian curves that sample the vorticity field and reconstructing the corresponding curves at each time step, in order to recompute velocities and so on. Filaments are curves that carry one scalar quantity called the circulation. A filament can be viewed as a vortex tube, that is a space volume with vorticity is parallel to the walls, shrunk on its centerline. The circulation is the vorticity flux across the sections of the tube. Filaments are both very physically and numerically appealing for several reasons. Their Lagrangian character and the fact that their circulation remains constant when the flow is inviscid, translates Kelvin and Helmoltz theorems which are the two major facts in incompressible flows. From a numerical point of view, they allow to give with few degrees of freedom a rather detailed description of albeit complex dynamics. They have been investigated in some of the first ever 3D simulations in CFD [52] and they have been successfully used in several graphics works(see for instance [1]). However, several points detract them from being considered as a general tool to model and simulate incompressible flows. Except in specific (nonetheless interesting) situations, like rings and jets, they are not so easy to initialize for a given flow. Moreover, following a filament eventually requires at some point ad hoc decisions to avoid tiny loops or to decide reconnections between nearby filaments (something which is called filament surgery). Finally there is no clear cut way to simulate diffusion with filaments. For all these reasons we will only consider point particles in the rest of this class.

2.3 Grid-Free and Hybrid Particle Methods

The distinction between **grid-free** particle and hybrid **particle-grid** methods emerges when dealing with flow related equations besides the advection. These additional equations may be necessary in order to determine the right hand F in (3) or when the advection velocity is not given as a function of the advected quantity (as in the Biot-Savart law for the velocity-vorticity formulation). When we discuss *Grid Free particle methods* , we imply methods that rely solely on the particles to compute these quantities. By *Hybrid: Particle-Grid methods*, we imply methods which also use and underlying fixed grid.

Hybrid methods involve combinations of mesh based schemes and particle methods in an effort to combine computational advantages of each method. The first such method involves the Particle in Cell algorithm pioneered by Harlow [34] in which a particle description replaces the non-linear advection terms and mesh based methods can be used to take advantage of the efficiency of Eulerian schemes to deal with elliptic or hyperbolic problems. In the following we give two examples of these methods one for each of the two classes of methods we have introduced.

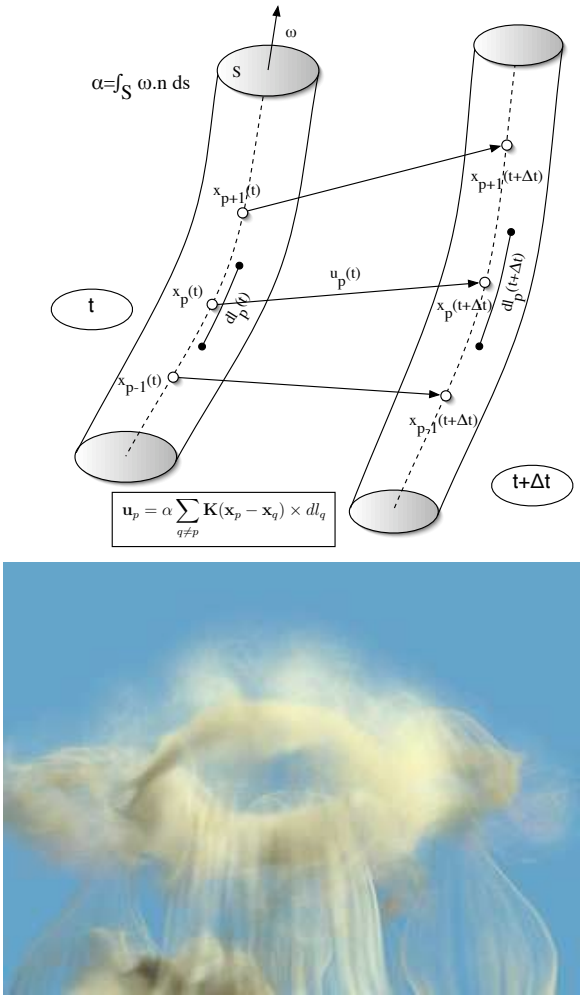

Figure 6: Top picture: vortex tube, filament and circulation. Evolution from time t to time $t + \Delta t$. the markers along the filament allow to reconstruct the curve and compute velocities for the next time step. The velocity formula is obtained from the Bio-Savart law (23).Bottom picture: example of image synthesis implementing filaments from [1].

2.3.1 SPH AND PARTICLE MESH HYDRODYNAMICS First for SPH methods, we have underlined in (15), (16) a method which is grid-free. Instead of evaluating the term $\partial u/\partial x$ with the kernel W one might have chosen to use an underlying fixed grid and to increment particle density through the following successive steps

- *assign* velocity values u_i on the grid from the known particle quantities u_p
- evaluate by finite-differences on the grid derivatives of u on the grid
- interpolate back these quantities on the particles to obtain particle quantities div_p
- finally solve $d\rho_p/dt = -\rho_p \, div_p$

The same approach can be used to determine the stresses in the right hand side of the momentum equation. In case the stresses result form an energy carried by particle, there

is an additional quantity carried by the particles, and both particles velocity and energy have to be assigned on the grid to compute the stresse which are next interpolated on the particles. The use of a mesh in the context of SPH helps accelerate the calculations and as we will see later it helps maintain the accuracy of the method. This combination of grids and particles, that we baptized *PMH : Particle-Mesh Hydrodynamics* [13]has been shown to be highly effective in a number of flow systems that have been challenging for traditional SPH. The two phases of assignment and interpolation between grid and particles are crucial to ensure that the process is both accurate and does not introduce spurious oscillations. A lot of effort has been devoted in CFD to this issue. We will come back later when we discuss remeshing which somehow is currently the most effective way to approach this problem. Grid-free SPH have a symmetric issue for the choice of the kernel W and renormalization techniques to ensure conservation properties. Both methods crucially need to care about the number of particles per grid-size (for PIC method) or inside the range of the kernel W (for grid-free methods). It is important to realize at that point that in particle methods particles have a numerical meaning not as individual points but only through their collective contribution. This is a definite difference with finite-diffrence, finite element or finite volume methods.

2.3.2 VORTEX METHODS : GRID FREE AND HYBRID For the second example, we consider vortex methods of the the inviscid ($\nu = 0$) Navier-Stokes equation (18). In that case, we only need to determine the velocity values necessary to push particles and to update particle vorticity values. The grid-free way to do it relies on the so-called Biot-Savart law.

The Biot-Savart law is an integral expression of the velocity in terms of the vorticity. Consider first the case of a non-bounded flow. A divergence-free velocity u with vorticity ω and vanishing at infinity is given by

$$u(x,t) = \int \mathbf{K}(x-y) \times \omega(y)\, dy \tag{23}$$

where the kernel $\mathbf{K}$ is given by the formula $\mathbf{K}(x) = \frac{1}{4\pi}\frac{x}{|x|^3}$. If the velocity has a given non zero value at infinity, this contribution has just to be added in the right hand side of (23).

The case of a flow with solid boundaries is more involved. In that case the boundary condition to be imposed on the velocity is in general a condition on the normal component of the velocity (a condition on the other component becomes necessary and physically relevant only for viscous flows). For the classical case of no-flow through a boundary Σ enclosing a fluid domain Ω, the theory of integral equations leads to the addition of a potential to the formula (23) :

$$u(x,t) = \int_\Omega \mathbf{K}(x-y) \times \omega(y)\, dy + \int_\Sigma \mathbf{K}(x-y) \times q(y)\, dy \tag{24}$$

where q is a potential to be determined through an integral equation such that the resulting velocity satisfies:

$$u(x,t) \cdot n(x) = 0 \text{for } x \text{on } \Sigma$$

The enforcement of the kinematic boundary conditions result in boundary integral equations that can be solved using boundary element methods [36] an approach that is widely used in engineering.

Let us now turn to the hybrid particle-grid counterpart of this method. As for the case of gas dynamics, one first needs to assign the quantity advected by particles - the

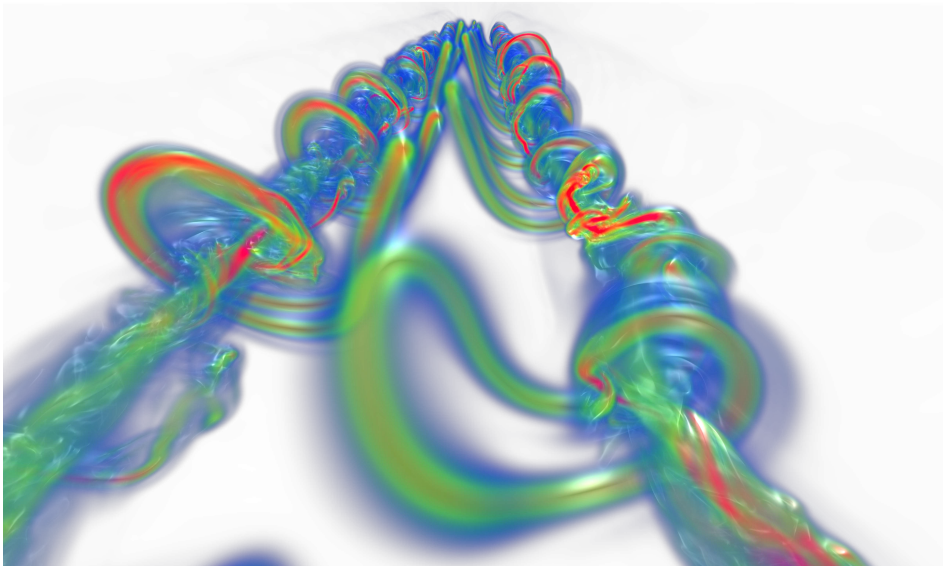

Figure 7: Billion Particle Simulations of Aircraft Wakes using Remeshed Vortex Methods [12]

vorticity values in this case - to grid nodes. Once it is done, one can reformulate the problem of finding the velocity in terms of the vorticity as a Poisson equation. Indeed, since div $u = 0$, one may write $u = \nabla \times \boldsymbol{\psi}$ where $\boldsymbol{\psi}$ is a divergence-free stream function. Then one gets $\omega = \nabla \times u = -\Delta \boldsymbol{\psi}$. We are thus left with the following Poisson equation:

$$-\Delta \boldsymbol{\psi} = \omega. \tag{25}$$

This problem can be solved by off-the-shelf grid based Poisson solvers. To handle in a simple fashion boundary conditions of no-through flow type, it is in general advisable to use an additional scalar potential ϕ and look for u under the following form:

$$u = \nabla \times \boldsymbol{\psi} + \nabla \phi. \tag{26}$$

The stream vector $\boldsymbol{\psi}$ has to to be divergence-free and satisfy (25). The scalar potential ϕ does not contribute to vorticity. To give a divergence-free contribution it must satisfy

$$-\Delta \phi = 0 \tag{27}$$

in the computational domain. Its boundary condition is then adjusted to give no-though flow at the boundary Σ: from (26) it has to satisfy

$$\frac{\partial \phi}{\partial n} = -(\nabla \times \boldsymbol{\psi}) \cdot n.$$

This is classical Neumann-type boundary condition that complements (97). The advantage of this formulation is that it facilitates the calculation of the stream function. Without this potential, the stream function would have to satisfy boundary conditions coupling its components in order to be consistent with the divergence free condition.

To illustrate the method we present below a sketch of an hybrid particle-grid method using Williamson's low-storage third order Runge-Kutta scheme number 7 to integrate the equations of the particles. The scheme limits the numerical dissipation introduced

into the flow, and it is memory efficient, requiring only one N additional storage per variable. The overall procedure is illustrated by Algorithm 1.

Set up, initial conditions, *etc.* , $t = 0$;
/* Particle quantities stored in arrays,
e.g. vorticity: $\omega \in \mathcal{R}^{\ni \times \mathcal{N}}$. For the ODE solver we need two temporary variables: $u0$, and $d\omega 0$ */
while $t \leq T$ **do**
  for $l = 1$ *to* 3 *;* /* stages of the ODE Solver */
  do
    Interpolate ω onto the grid ($\omega \rightarrow \omega_{ijk}$);
    Compute velocity u_{ijk} from ω_{ijk};
    $u0 \leftarrow$ Interpolate u_{ijk} onto the particles;
    $u0 \leftarrow u + \alpha_l\, u0$; $d\omega 0 \leftarrow d\omega + \alpha_l\, d\omega 0$; /* $\alpha = (0, -\frac{5}{9}, \frac{153}{128})$ */
    $x \leftarrow x + \delta t\, \beta_l\, u0$; $\omega \leftarrow \omega + \delta t\, \beta_l\, d\omega 0$;
  end
end

Algorithm 1: A Particle-in-Cell method using Williamson's Runge-Kutta scheme no.7.

We note that this hybrid formulation has enabled simulations using an unprecedented number of **10 billion particles** [12] of computational elements for the simulation of aircraft vortex wakes (see Fig:7)

2.3.3 GRID-FREE VS. HYBRID - THE WINNER IS.... Let us now pause to compare the respective merits of the grid-free and particle-grid approaches. Clearly the grid-free approach is appealing in that it fully maintains the lagrangian nature of the method. If short range interactions of particles are involved in the right hand side F one may devise particle interactions on physical basis. Particle methods can then be seen both as numerical methods and as discrete physical models. For incompressible flows the Biot-Savart law is required to compute non-local interactions. One is thus led to a N-body problem. If the vorticity is sampled on N particles, the simple minded calculation of the right hand side of (23) requires $O(N^2)$ operations, something which is not affordable for N beyond a few hundreds. To overcome this problem, a lot of effort has been devoted, following the pioneering work of Greengard and Rokhlin [33], to reduce this cost to something approaching $O(N)$. To summarize, the idea is to divide the particle distribution in clusters of nearby particles. The exact interaction of particles in one cluster with particles of another well separated cluster is replaced by an algebraic expansion using the moments of clusters of particles around their center. The number of terms only depends on the desired accuracy and never goes beyond a single digit number. Only interaction between particles in the same cluster are computed by direct summation. For maximal efficiency, the clustering of particles is done using a tree algorithm which creates boxes at different level of refinements containing always about the same number of particles. These fast summation formulas are now routinely used in CFD particle-based grid-free codes.

Unfortunately, practice shows that the construction of the tree, the evaluation of expansion coefficients and of the direct interaction of nearest particles, remain expansive, in particular in 3D. As a matter of fact the turnover point where the fast summation formulas become cheaper than the direct summation formulas is always beyond a few hundreds, which means that the constant in front of N or $N \log N$ in the evaluation of

the complexity of the method is quite high. The cost of fast summation formulas is definitely much higher than that of FFT-based grid Poisson solvers. This is the reason why hybrid particle-grid methods can be seen as attractive alternative to grid-free methods. In an hybrid method, one first need to overlay a fixed grid to the particle distribution. This immediately rises the question of artificially closing a computational domain which is some cases should not. In grid-based method this is classical issue but it is somehow disturbing to introduce it in particle methods which in principle could avoid it. Also, in many instances vorticity is localized in a narrow part of the space, and the grid is certainly going to waste a number of points. All these observations point to the fact that hybrid grid-particle methods go in some sense against the very nature and advantages of particle methods, and thus should only be used if they come with a significantly lower computational cost. Clearly this will heavily depend on the ratio number of particles/ number of grid points, and thus on how much the vorticity support is concentrated and how far the artificial boundaries of the grid should be pushed if we deal with an external flow (a wake or a jet for instance). It also depends on the efficiency of the grid-based Poisson solver.

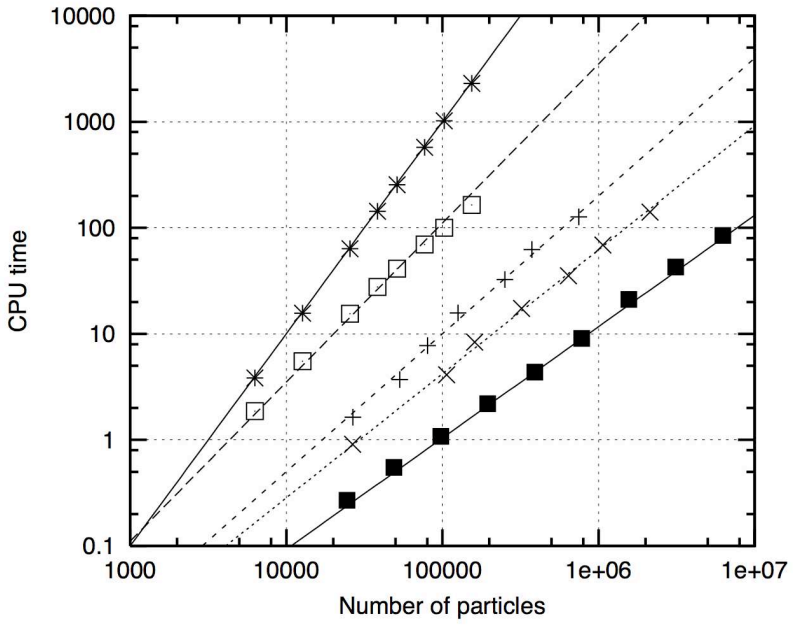

Figure 8: Implementation of a tree code/ Illustration of the clustering of particles in boxes obtained by successive subdivisions of the computational box.

In many cases of practical interest, the result of this comparison, taking into account all these factors, is in favor of hybrid particle-grid methods. Figure 8 shows the computational times on a single processor machine for various 3D methods found in the literature. The computational time shown in this experiment corresponds to one evaluation of the velocities on all particles. The grid-free method has been used in an unbounded doamin, that is without the additional complexity of identifying the boundary potential, either with direct or fast summation formulas derived in [51]. The hybrid particle-grid method has been used either in a square box, together with a FFT method, or in a cylindrical box,

with a cyclic method to solve the linear system obtained from the discretization of the Poisson equations on the grid. In the first case the particles fill the whole box (like in an homogeneous turbulence experiment), while in the other case they fill only about 25% of the box. This last configuration is typical of what we would get for the simulation of the wake behind a circular cylinder. Not surprisingly, the hybrid method gives the largest speed up (about a factor 100 compared to grid-free methods with fast summations) in the first configuration. But even in the second configuration, where one would think that many grid points are wasted, the speed up is already significative (about a factor 10).

Recently we have developed a Parallel Particle-Mesh (PPM) software library [78] that facilitates large-scale calculations of transport and related problems using particles. The library implements grid-particle methods. The library provides the mechanisms necessary to achieve good parallel efficiency and load balancing in these situations where both meshes and particles operate as computational elements. The PPM library scales to systems with up to 16,000 processors, with an efficiency of 80% and allows simulations using billions of computational elements [12]

2.3.4 FURTHER HYBRIDIZATION The above two concepts of Grid-Free and Hybrid can be further combined and extended giving rise to a variety of numerical methods. In this context *Lagrangian-Eulerian* domain decomposition methods use high order grid methods and vortex methods in different parts of the domain [19, 69] and can even be combined with different formulations of the governing equations. A finite difference scheme (along with a velocity-pressure formulation) can be implemented near solid boundaries, and vortex methods (in a velocity-vorticity formulation) can be implemented in the wake to provide the flow solver with accurate far-field conditions. In this approach Eulerian methods handle the wall boundary conditions and can be complemented with immersed boundary methods [61] to handle complex geometries. A rigorous framework for particle based immersed boundary methods has been developed based on a unified formulations of the equations for flow-structure interaction [18]. Simulations involving this formulation are a subject of ongoing investigations.

3 REMESHED PARTICLE METHODS

Particle methods are often defined as grid-free methods making them an attractive alternative to mesh based methods for flows past complex and deforming boundaries. However the adaptivity provided by the Lagrangian description can introduce errors and particle methods have to be conjoined with a grid to provide consistent, efficient and accurate simulations. The grid does not detract from the adaptive character of the method and serves as a tool to restore regularity in the particle locations via *Remeshing* while it simultaneously enables systematic *Multiresolution* particle simulations [5], allows *Fast velocity evaluations* [34] and facilitates *Hybrid Particle-Mesh* methods capable of handling different numerical methods and different equations in various parts of the domain [19].

3.1 (the need of) Remeshing for Particle Distortion

Particle methods, when applied to the Lagrangian formulation of convection-diffusion equations enjoy an automatic adaptivity of the computational elements as dictated by the flow map. This adaptation comes at the expense of the regularity of the particle distribution because particles adapt to the gradients of the flow field. The numerical analysis of vortex methods shows that the truncation error of the method is amplified exponentially in time, at a rate given by the first order derivatives of the flow that are precisely related to the amount of flow strain. In practice, particle distortion can result in the creation and evolution of spurious vortical structures due to the inaccurate resolution of areas of high shear and to inaccurate approximations of the related derivative operators.

To remedy this situation, *location processing* techniques reinitialize the distorted particle field onto a regularized set of particles and simultaneously accurately transport the particle quantities. The accuracy of remeshing has been thoroughly investigated in [47]. The Remeshing is shown to introduce numerical disspation that is far below the dissipation introduced by time and spatial discretizations. One way to regularize the particles is setting the new particle positions to be on the grid node positions and recomputing the transported quantities with a particle-mesh operation.

In order to demonstrate the need of the remeshing step, we consider the vorticity equation without the viscosity term ($\nu = 0$).
In this case the vorticity evolves according to the Euler equation $\frac{D\omega}{Dt} = 0$. As initial condition we set a radial function:

$$\omega_0(\mathbf{x}) = \omega_{\max} \cdot \max\left(0, 1 - \|\mathbf{x}\|/R\right), \tag{28}$$

where W is the maximum vorticity and R controls the support of ω_0. Since the vorticity is radially symmetric and there is no diffusion, the system is in a steady state: the exact solution in time is just the initial condition ($\omega(t) = \omega_0$). We can therefore use this problem as validation test and study how is important to remesh the particles during the simulation. Figure 9 shows the crucial difference between performing and not performing the remeshing step. In this case we used $W = 100$, $R = 0.5$, and a time step $\delta t = 5 \cdot 10^{-3}$. When no remeshing step is performed, the solver generates growing spurious structures which lead to a break in the radial symmetry of the vorticity field. This break causes an highly increasing inaccuracy of the computed solution.

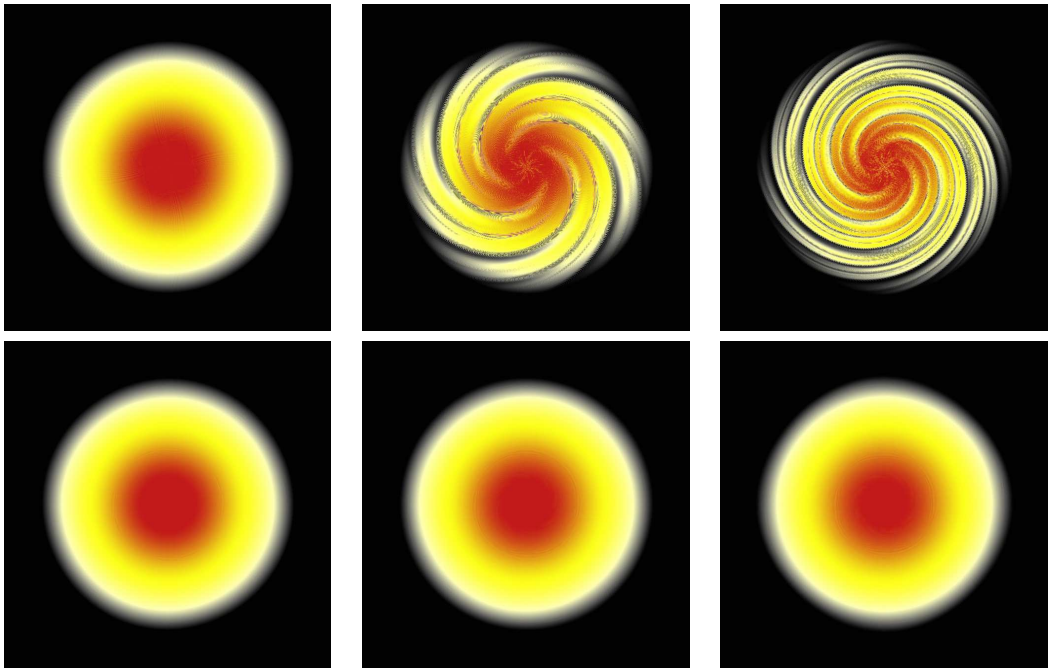

Figure 9: **Why do we need to remesh ?** Inviscid evolution of a 2D axisymmetric vorticity field (an exact solution of the Euler equations) at time $t = 0.01$, $t = 0.10$ and $t = 0.15$; using a second order time integrator for the Euler equation. A grid-free particle simulation produces spurious artifacts that becomes progressively stronger (top). Remeshing the particles enables the method to maintain the axisymmetric profile and to provide an accurate solution of the Euler equations (bottom).

The problem of extracting information on regular grid from a set of scattered points has a long history in the fields of interpolation [80]. To facilitate the analysis we restrict our attention to a one dimensional equispaced regular grid with unit mesh-size onto which we interpolate quantities (q_n) from scattered particle locations(x_n) :

$$Q(x) = \sum_n q_n \, W(x - x_n). \tag{29}$$

The properties of the interpolation formulas can be analysed through their behavior in the Fourier space [80]. The *characteristic function* $g(k)$ of the interpolating function W(x) is defined as :

$$g(k) \;=\; \int_{-\infty}^{+\infty} W(x)\, e^{-ikx}\, dx.$$

When W decays fast at infinity, g is a smooth function and the interpolation formula Eq.29 is of degree m if the following two conditions hold simultaneously: (i) $g(k)-1$ has a zero of order m at $k = 0$ and (ii) $g(k)$ has zeros of order m at all $k = 2\pi n$, $(n \neq 0)$. These requirements translated back in the physical space are nothing but the moment properties of the interpolant

$$\int W(y)\, dy = 1 \; ; \; \int y^{\alpha} W(y)\, dy = 0, \text{ if } 1 \leq |\alpha| \leq m - 1 \, .$$

This is reminiscent of the conditions for accurate function particle approximations using moment conserving kernels. In fact the interpolation accuracy [40] can be described

by splitting the interpolation error into a convolution and sampling error reminiscent of the smoothing/quadrature error for function approximations. Hence, good interpolation schemes are those that are band-limited in the physical space and are simultaneously close approximations of the ideal low pass filter in the transformed space. Monaghan [62] presents a systematic way of increasing the accuracy of interpolating functions, such as B-splines, while maintaining their smoothness properties using extrapolation. He constructs interpolation formulas such that if $m = 3$ or $m = 4$ the interpolation will be exact for quadratic functions, and the interpolation will be third or fourth order accurate. One widely used formula involves the so-called M_4' function

$$M_4'(x) = \begin{cases} 0 & \text{if } |x| > 2 \\ \frac{1}{2}(2-|x|)^2(1-|x|) & \text{if } 1 \leq |x| \leq 2 \\ 1 - \frac{5x^2}{2} + \frac{3|x|^3}{2} & \text{if } |x| \leq 1 \end{cases} \tag{30}$$

Interpolations in higher dimensions can be achieved by tensorial products of these formulas. However, these tensorial products require particle remeshing on a regular grid. For non grid-conforming boundaries, remeshing introduces particles onto areas that are outside the flow domain and violates the flow boundary conditions. Remedies such as one-sided interpolation have been proposed and a working solution can be obtained ([24] and [72]) by eliminating particles outside the domain and adjusting accordingly the modification of particle strengths by re-enforcing the boundary conditions in a fractional step algorithm. Alternatively, *Weight Processing schemes* attempt to explicitly [3] or implicitly [90] modify the particle weights in order to maintain the accuracy of the calculation but they result in rather costly computations.

Finally one lingering question is : when to remesh ? Practice indicates that remeshing at every time step does not detract from the accuracy of the method and at the same time enables the use of the grid to develop differential operators. In [37] a criterion was developed that is well suited to particle methods and their caharcteristics as a partition if unity technique. In order to determine the rate at which particle remeshing is necessary it is possible to introduce a measure of distortion. This measure relies on the fact that the weighted sum $H(t)$ over all particles must be equal to unity in a regularised particle map

$$H(t) = \sum_j v_j(t)\zeta_\epsilon(\mathbf{x}_p(t) - \mathbf{x}_j(t)) \tag{31}$$

$$H(0) = H_0 = 1. \tag{32}$$

The average change of $H(t)$ over all particles is a measurement of distortion

$$\Delta H = \frac{1}{N_p}\sum_j \frac{|H_j(t) - H_{0,j}|}{H_{0,j}}, \tag{33}$$

where N_p is the number of particles. When considering particles undergoing a solid body motion or rotation there is no particle distortion and as such $\Delta H = 0$ Remeshing can be invoked each time the function ΔH exceeds a small prespecified threshold.

3.2 Communication between particles and meshes

The combined use of particles and meshes has been dictated in the past by the need to accelerate the velocity evaluations in vortex particle methods as in the PIC methodology.

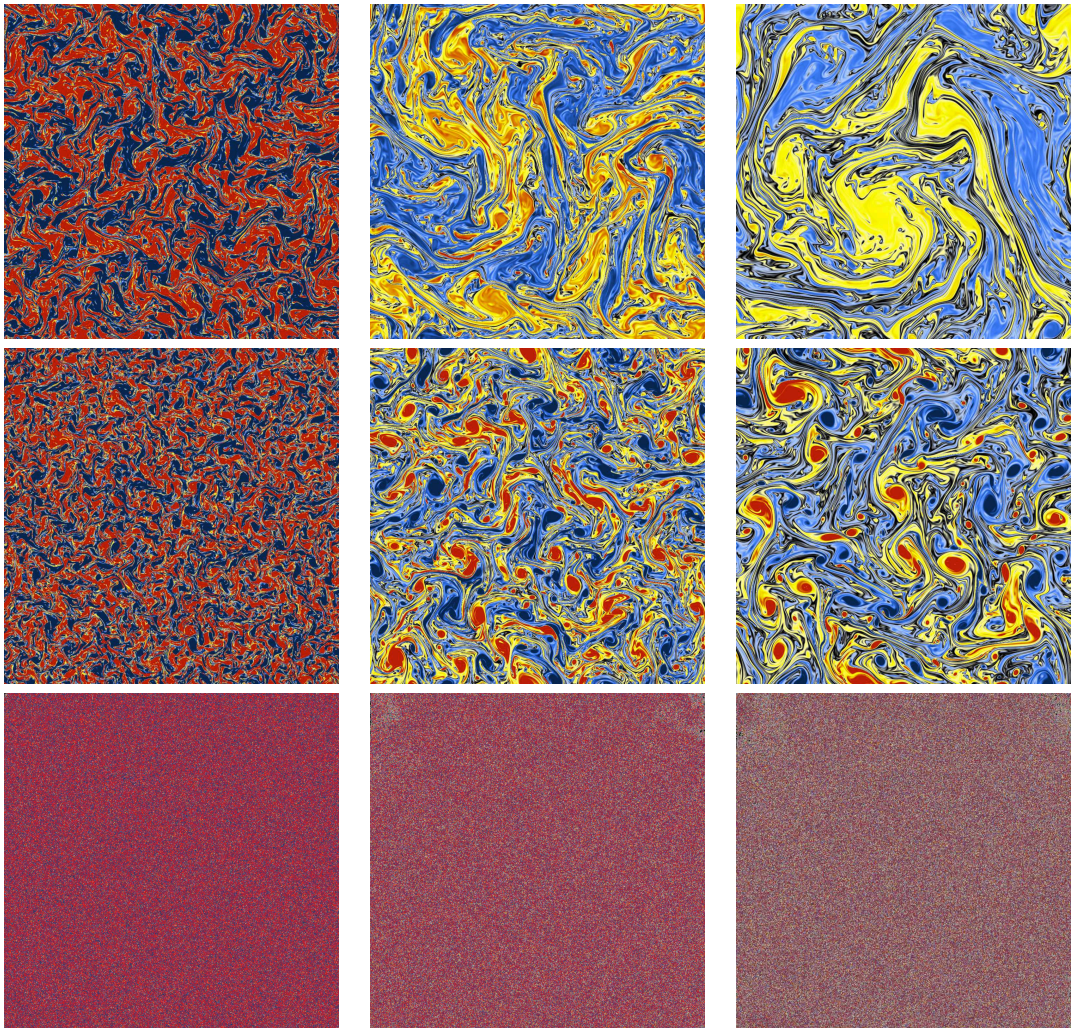

Figure 10: **Why do we need to remesh ?** Evolution of an initially random vorticity distribution, using a first order time integrator (top), a 4th order time integration (middle), a 4th order time integrator without remeshing during the simulation (bottom), at time $t = 0.01$, $t = 0.1$ and $t = 0.2$. Note that without remeshing the random field remains random, while first order time integration schemes introduce high viscosity to the flow.

As we discuss here, in addition to facilitating the computation of the velocity field the mesh is needed for the regularisation of the particle locations.

The mesh is used in the present framework to regularize the particle locations, with the grid nodes becoming particles that are convected in a Lagrangian manner in the following time step. Going between particles and mesh requires the definition of two interpolating operators. The **Particle-Mesh** interpolation ($M \leftarrow P$) is denoted as $\mathbf{I}_P^M$. On a given a set of particles $\{(\omega_p, \mathbf{x}_p)\}$, the $\mathbf{I}_P^M$ maps the particle vorticity onto grid nodes with grid spacing h as

$$(\mathbf{I}_P^M\{(\omega_p, \mathbf{x}_p)\}, \{\mathbf{x}_m^{mesh}\}) \rightarrow \omega_m^{mesh} = \sum_p \omega_p \cdot W(\frac{1}{h}(\mathbf{x}_m^{mesh} - \mathbf{x}_p)). \qquad (34)$$

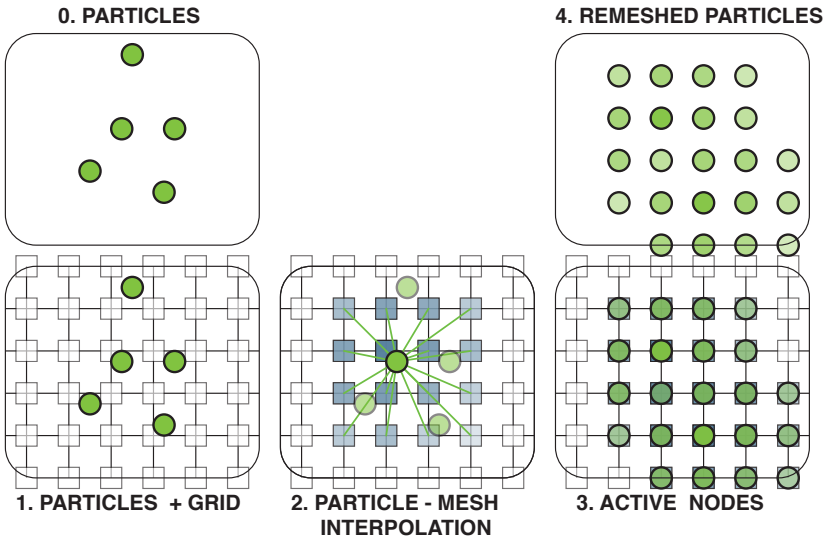

Figure 11: Remeshing of Particles on a Regular Grid. The particles are superimposed on the grid and their values are interpolated onto the grid nodes. After eliminating grid nodes with value below a threshold the grid nodes become particles ready to be convected by the flow field

The **Mesh-Particle** interpolation ($P \leftarrow M$) is denoted as $\mathbf{I}_M^P$. Given the vorticity on the mesh one can recover the vorticity of each particle by defining the mesh-particle operation $\mathbf{I}_M^P$:

$$(\mathbf{I}_P^M\{(\omega_m, \mathbf{x}_m^{mesh})\}, \{\mathbf{x}_p)\} \rightarrow \omega_p = \sum_m \omega_i^{mesh} \cdot W(\frac{1}{h}(\mathbf{x}_p - \mathbf{x}_m^{mesh})). \tag{35}$$

The interpolation kernel can be expressed as tensorial product $W(x, y) = W(x)W(y)$. For example the M_4' kernel conserves the total value, the linear and angular impulse between quantities on particles and the mesh and it is expressed as :

$$W(x)\, M_4'(x) = \begin{cases} 0 & \text{if } |x| > 2 \\ \frac{1}{2}(2 - |x|)^2(1 - |x|) & \text{if } 1 \leq |x| \leq 2 \\ 1 - \frac{5}{2}x^2 + \frac{3}{2}|x|^3 & \text{if } 1 \geq |x| \end{cases} \tag{36}$$

Note that using M_4' as interpolation kernel is equivalent to the Catmull-Rom spline interpolation used in computer graphics.

To formally express the remeshing operation, we can suppose that we have a particle set $S = \{(\omega_p, \mathbf{x}_p)\}$ and $W(.)$. Then, the result of a remeshing operation is the following particle set:

$$\text{Remeshing}(\ \{(\omega_p, \mathbf{x}_p)\}\ ) = (\mathbf{I}_P^M\{(\omega_p, \mathbf{x}_p)\}, \{\mathbf{x}_m^{mesh}\}). \tag{37}$$

3.2.1 EFFICIENT IMPLEMENTATION OF PARTICLE-MESH INTERPOLATION The efficiency gain of hybrid particle methods over non-hybrid particle methods hinges on the efficient implementation of Particle-Mesh and Mesh-Particle interpolations. As we

have seen before, the $\mathrm{P} \to \mathrm{M}$ interpolation has the form

$$q_i = \sum_p Q_p \, M(i\,h - x_p)\,. \tag{38}$$

The most straightforward way to implement (38) is a verbatim translation of the expression into code, *i.e.* to sum over all particles and evaluate the kernel function M. Clearly, this is very inefficient due to the locality of the kernel M. A more efficient approach would be to use cell lists an loop over cell lists located around the target grid point i. Still this involves calculating distances of particles to i, and we perform many extra operations.

The key to an efficient implementation of (38) lies in realizing that the operation is usually performed for a whole grid, *i.e.* a set of grid points i, and then to flip the loops as illustrated by Algorithm 2.

for $p \in \mathcal{P}$ **do**
  $\hat{x} = x_p\, h^{-1}$;
  $i1 = \mathrm{INT}(\hat{x})$; $i0 = i1 - 1$; $i2 = i1 + 1$; $i3 = i1 + 2$;
  $x1 = \hat{x} - \mathrm{REAL}(i1)$; $x0 = x1 - 1.0$; $x2 = x1 + 1.0$; $x3 = x1 + 2.0$;
  $a0 = M(x0)$; $a1 = M(x1)$; $a2 = M(x2)$; $a3 = M(x3)$;
  /* In these kernel evaluation statements we can exploit the fact that the kernel M is usually a piecewise polynomial, and that we a-priori know which interval of the kernel $x0$, $x1$, *etc.* fall into; this saves us from using any conditionals. */
  $q[i0[0], i0[1], i0[2]] = q[i0[0], i0[1], i0[2]] + Q_p\, a0[0]\, a0[1]\, a0[2]$;
  $q[i0[0], i0[1], i1[2]] = q[i0[0], i0[1], i1[2]] + Q_p\, a0[0]\, a0[1]\, a1[2]$;
  $\ldots$;
  $q[i3[0], i3[1], i3[2]] = q[i3[0], i3[1], i3[2]] + Q_p\, a3[0]\, a3[1]\, a3[2]$;
end

Algorithm 2: $\mathrm{P} \to \mathrm{M}$ interpolation; in this 3D example we assume that the computational domain starts at the origin, the indices start at zero, and that we are using a kernel with support 4. The index i is a symbolic abbreviation for i, j, k, *e.g.* $i1[1] \equiv j1$.

3.3 Evaluation of differential operators

Our hybrid particle mesh methods have a one-to-one relation between particles and the mesh. Thus we can make use of the efficiency of the evaluation of differential operators on a regular grid by interpolating particle quantities onto the grid ($\mathrm{P} \to \mathrm{M}$), evaluating the operators ($\mathrm{M} \to \mathrm{M}$) and interpolating the result back onto the particles ($\mathrm{M} \to \mathrm{P}$):

(1) $\mathrm{P} \to \mathrm{M}$ $q_i = \sum_p Q_p \, M(i\,h - x_p)$
(2) $\mathrm{M} \to \mathrm{M}$ $r_i = (\Delta^{h,\mathrm{FD}}\, q)_i$
(3) $\mathrm{M} \to \mathrm{P}$ $(\Delta q)(x_p) = \sum_j M(x_p - j\,h)\, r_j$

Together this yields

$$(\Delta q)(x_p) = \sum_i M(x_p - i\,h)(\Delta^{h,\mathrm{FD}})\Big(\sum_{p'} Q_{p'}(i\,h - x_{p'})\Big)\,, \tag{39}$$

or

$$\Delta^h q = \underline{M}^{\mathrm{T}}\,\underline{D}\,\underline{M}\,Q\,. \tag{40}$$

Next to the efficiency gain by bypassing particle-particle interactions, this approach plays an enabling role in adopting immersed interface techniques (see Chapter 3), and multiresolution (see Chapter 5).

Velocity evaluation The velocity evaluation involves solving the Poisson equation 25 and computing the curl of the resulting stream function Ψ. In the following we will be dealing with unbounded or periodic flows only. In these cases we use FFT-based Poisson solvers. As we do not require the stream function as such, but only its curl, we can perform the curl also in Fourier space. Thus the velocity evaluation takes the form: (i) Transform vorticity into Fourier space, (ii) evaluate velocity as $\hat{u}(k) = k \times \hat{\omega}\,|k|^{-2}$, (iii) transform velocity into physical space. In the case of unbounded domains we also use FFTs and apply the technique introduced by Hockney and Eastwood in [40].

Stretching and dissipation The stretching and the dissipation are computed in physical space on the grid as

$$\begin{aligned}
&\nabla^h \cdot (u_x\,\omega) + \mathrm{Re}^{-1}\,\Delta^h\,\omega_x \\
&\nabla^h \cdot (u_y\,\omega) + \mathrm{Re}^{-1}\,\Delta^h\,\omega_y \\
&\nabla^h \cdot (u_z\,\omega) + \mathrm{Re}^{-1}\,\Delta^h\,\omega_z\,,
\end{aligned}$$

where ∇^h is the fourth-order finite difference approximation of ∇, and Δ^h is the fourth-order finite difference approximation of the the Laplacian. As the velocities are spectrally accurate using fourth-order differences here is beneficial if the flow is well resolved, and leads to more accurate results.

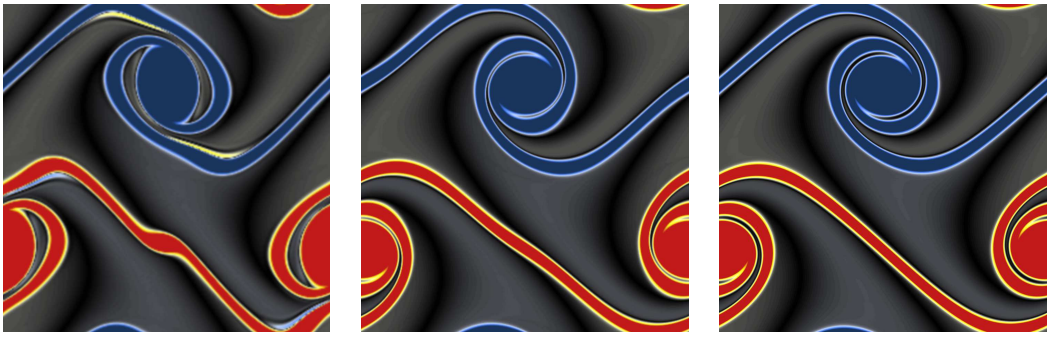

Figure 12: Simulations of the thin double shear layer. Red denotes high positive vorticity, and blue denotes high negative vorticity. From left to right: simulation using 256x256, 512x512 and 2048x2048 particles. Note the development of a spurious vortex for the two lower resolutions

3.3.1 THE ADVANTAGES OF STRUCTURE On a structured grid physical neighborhood and logical neighborhood (in memory, access) usually coincide. Compare the following evaluation of Δq in 2D at a given point (on a particle, on a grid point, respectively):

$$\Delta^{h=1} q \approx (q_{i+1,j} + q_{i-1,j} + q_{i,j+1} + q_{i,j-1} - 4\, q_{i,j});$$

Algorithm 3: Laplacian on the grid

for $p' = 1, N_{\text{neighbors}}(p)$ **do**
  $l = \text{NeighborIndex}(p')$;
  $(\Delta^{\varepsilon,h} q)_{p'} += \eta(x_p - x_l)\,(q_l - q_p)$;
end

Algorithm 4: Laplacian on particles

Evidently, the minimal size, direct access evaluation on the grid is more efficient and in general leads to less cache misses than the particle evaluation. Additionally, the grid-based evaluation is much easier to program as it does not require auxiliary data structures such as verlet lists, or cell lists.

The particle solver has been validated on a number of benchmark tests for accuracy and efficiency. The thin double shear layer is a challenging benchmark for incompressible flow solvers. Brown and Minion [60] have demonstrated that in under-resolved simulations spurious vortices infiltrate the numerical solution in discretizations by various computational methods. We have computed the double shear layer problem as presented in [60], studying the effect of solving the vorticity at low-resolutions often associated with the creation of secondary spurious vortices. The domain is again the unit square with periodic boundary conditions with the initial condition for the velocity field $\mathbf{u} = (u, v)$ in the following non-dimensional form:

$$\begin{cases} u(x,y) = \tanh(\rho \cdot \min(y - 0.25, 0.75 - y)) \\ v(x,y) = \delta \cdot sin(2\pi(x + 0.25)) \end{cases}$$

In the present simulations we consider the thin shear layer obtained by setting $\delta = 0.05$, $\rho = 80$ and a viscosity $\nu = 10^{-4}$.

All simulation were performed using the 4th order Runge-Kutta with a timestep $dt = 0.02$ for $t_{\text{end}} = 1.0$ The numerical results are depicted in 12 in the form of vorticity for three different resolutions. It is evident that a spurious vortex is present in the simulation result with the coarsest resolution. The spurious vortices are eliminated using a hybrid particle method with 512x512 grid/particles. Note however the solution shows some minor undulations instead of the expected straight line [60], near to the center of the domain.

This numerical artifact disappears when using 2048x2048 computational elements.

Next we consider the case of viscous vorticity decay from an initially uniform random distribution with an average of zero vorticity and a maximum value of 400. The considered physical domain was the unit square with periodic boundary conditions, and the viscosity was set to $\nu = 10^{-7}$. Figure 10 shows the evolution of the flow obtained with a first order time integrator and a timestep $dt = 0.001$ and with a 4th order Runge-Kutta time integrator. The utilization of a first order time integration scheme introduces a large amount of numerical viscosity producing large, weak vortex cores. On the other hand, the 4th order Runge-Kutta scheme succeeds in restraining the effects of numerical viscosity producing smaller vortices of higher intensity. Furthermore we note in particular the role of remeshing : In the absence of remeshing the random vorticity field remains random, as the low viscosity is overwhelmed by the chaotic motion of the particles and

no structure emerges as one would expect from a viscous flow field.

3.4 A REMESHED particle method

We finally present the algorithm that is being used in our function evaluations

Set up, initial conditions, *etc.* , $t = 0$;
/* Particle quantities will be stored in arrays, *e.g.* vorticity: $\omega \in \mathcal{R}^{\Im \times \mathcal{N}}$. For the ODE solver we need two temporary variables: $u0$, and $d\omega 0$ */
while $t \leq T$ **do**
  for $l = 1$ *to* 3 *;* /* stages of the ODE Solver */
  do
    Interpolate ω onto the grid ($\omega \rightarrow \omega_{ijk}$);
    Compute velocity u_{ijk} from ω_{ijk};
    $d\omega_{ijk} \leftarrow$ Compute stretching and dissipation from u_{ijk} and ω_{ijk};
    $d\omega \leftarrow$ Interpolate $d\omega_{ijk}$ onto the particles;
    $u0 \leftarrow u + \alpha_l\, u0$; $d\omega 0 \leftarrow d\omega + \alpha_l\, d\omega 0$; /* $\alpha = (0, -\frac{5}{9}, \frac{153}{128})$ */
    $x \leftarrow x + \delta t\, \beta_l\, u0$; $\omega \leftarrow \omega + \delta t\, \beta_l\, d\omega 0$; /* $\beta = (\frac{1}{3}, \frac{15}{16}, \frac{8}{15})$ */
  end
end

Algorithm 5: Overall procedure of a hybrid vortex method using Williamson's Runge-Kutta scheme no. 7.

We note here two important facts: this algorithm is fast because every differential operation is performed in an Eulerian frame, without evaluating any kernel. Also the convection is performed in a fast way: we are dealing with particles, so we do not have a classical CFL condition on $\mathbf{u}$, we can take large time steps because they are bounded according to $\delta t \sim 1/||\nabla \mathbf{u}||_2$.
At the same time the convection is solved accurately because is solved in a Lagrangian fashion, i.e. following the characteristics of the solution. This method has been implemented efficiently in parallel computer architectures leading to unprecedented simulations using billions of particles (Figure 7) the simulation of aircraft wakes [12]

4 MULTIRESOLUTION PARTICLE METHODS

Lagrangian particle methods enjoy automatic adaptivity of the computational elements to the flow map. At the same time this adaptivity is not necessarily associated with an increased resolution of the flow in areas with critical phenomena as the convection of the particles is dictated by the corresponding velocity field. Furthermore regularity in the particle resolution in general imposes a severe restriction on the overall adaptivity of particle methods. For example in bluff body flows the boundary of the body is the source of vorticity and it is important to discretize adequately the region near the surface of the body [49]. This requirement dictates the size of the particle cores. However for constant size particles, as the vorticity gradients decay in the wake, it is clear that the flow is discretized using unnecessarily large numbers of computational elements. At the same time near areas of high shear (e.g. near the solid walls) small scales are produced and there is clearly a need to resolve this scale by adapting the particle distribution in these regions. The deficiency of constant size particle methods clearly detracts from the adaptive character of the method and its capability to accurately resolve strong gradients while remaining computationally efficient.

Hence, beyond adaptivity as dictated by the flow map, it is often necessary to employ particle methods with different resolution requirements as dictated by the physics of the problem.

The use of remeshing introduces the periodic regularisation of the particle locations and enables yet another important contribution to particle methods, namely the introduction of a consistent multiresolution framework for flow simulations. Borrowing from Eulerian based methods, techniques such as AMR can be easily incorporated. Remeshing of the particles can be adapted to accommodate grids of different resolutions based on criteria partaining to the structures of the flow field. In [21] Cottet, Koumoutsakos and Ould-Salihi formulated a convergent variable core size vortex method for the Navier-Stokes equations by using mappings from a reference space with uniform blobs to the "physical" space with blobs of varying size in conjunction with an anisotropic diffusion operator. This method was extended in a domain decomposition framework to handle several mappings corresponding to different zones and grid-size requirements in the flow. All these methods require *a priori* knowledge of where the flow field should be refined, and the refinement strategy is not of adaptive nature.

Both types of methods can be viewed as extensions of the methods proposed in [21]: one uses a global adaptive mapping, while the other is based on combinations of several local mappings. They also relate to adaptive Eulerian methods. The first class pertains to r-Adaptive finite element methods [59] while the second one is reminiscent of finite-difference Adaptive Mesh Refinement (AMR) methods [7]. The concept of r-adaptivity originated in the realm of finite elements and amounts to moving the computational elements into areas of the computational domain where increased resolution capability is needed. The first finite element method that achieved improved accuracy by adaptively moving the nodes of the triangulation has been presented by Miller in [59]. The equations of motion of the nodes have been determined by minimizing a global error functional with respect to the weights of the finite element basis functions and the positions of the nodes simultaneously. One can argue, that particle methods are inherently r-adaptive due to their Lagrangian character. This is true in the sense that computational elements are moved into areas where increased capability of resolution is required. Unlike the finite element method, the characteristic length scale of these elements remains

unaltered and it is usually uniform. Thus, the analog for r-adaptivity in particle methods is to employ particles with varying core sizes which adapt with the evolution of the solution they represent.

AMR methods have been first proposed by Berger and Oliger [7] in the context of finite-difference methods. The idea here is to define blocks of uniform grid-sizes, that are defined dynamically based for instance on a posteriori error estimates. Blocks with different grid-sizes communicate by exchange of boundary conditions as in domain decomposition methods. We revisit this class of methods in the context of particle methods by extending the variable-blob techniques introduced in [21] for the case of several local mappings. Our method is heavily based on overlapping of the sub-domains and particle remeshing in the overlapping zones play the role of interface boundary conditions in finite-difference AMR methods.
Finally AMR techniques require various criteria for the introduction of mesh refinement and they are often difficult procedures to automate. This automation can be further facilitated by introducing a wavelet analysis of the flow field borrowing from ideas in signal processing [57]. The use of wavelets in flow simulations has received significant attention in recent years but it has been mostly focused on Eulerian grid based methods (see [95] and references therein). In the context of remeshed particle methods it is possible however to adopt wavelet concepts leading to a Lagrangian particle wavelet technique [5]

In summary we distinguish four types of multiresolution particle methods that are further described in the following sections :

1. Particle Methods with Variable cores [22]
2. Particle Methods with Adaptive Global Mappings [4]
3. Particle Methods with AMR capabilities [4]
4. Particle-Wavelet techniques [5]

In our two classes of techniques, one can expect that rephrasing in terms of particle methods concepts inherited from adaptive finite-element or finite-difference methods may lead to methods that will keep the essence of particle methods, namely its robustness when dealing with convection dominated problems or problems involving complex physics, while optimizing their accuracy.

4.1 Particle method with variable core sizes

We present the variable core method as it is applied to convection-diffusion problems of the form

$$\frac{\partial q}{\partial t} + \nabla \cdot (u\, q) = \nu\, \Delta q\,. \tag{41}$$

Using the particle discretizations described in Chapter 2, Equation (41) can be solved by integrating the following ODEs for the particle positions, volumes and strengths:

$$\begin{aligned} \frac{dx_j}{dt} &= \mathbf{u}(x_j,t)\,, \\ \frac{dv_j}{dt} &= \nabla\cdot\mathbf{u}(x_j,t)\,v_j\,, \\ \frac{dQ_j}{dt} &= \frac{\nu}{\varepsilon^2}\sum_k \big[Q_k\,v_j - Q_j\,v_k\big]\eta^\varepsilon(|x_j-x_k|) \end{aligned} \tag{42}$$

for $j = 1, \ldots, N$.

In order to resolve problems with variable resolution requirements particle methods with variable core sizes have been introduced in [22]. They utilize a smooth map $f : \hat{\Omega} \to \Omega$, which maps a "reference" space $\hat{\Omega} \subseteq \mathcal{R}^{\lceil}$ with uniform core sizes $\hat{\varepsilon}$ onto a "physical" space $\Omega \subseteq \mathcal{R}^{\lceil}$, so that

$$x = f(\hat{x})\,, \qquad \hat{x} = g(x)\,, \qquad \{\underline{\Phi}\}_{ij} = \frac{\partial \hat{x}_i}{\partial x_j} \qquad \text{and} \qquad |\underline{\Phi}| = \det \underline{\Phi} \tag{43}$$

and the cores in physical space become anisotropic and of the order of $\varepsilon \sim \left(\frac{1}{|\underline{\Phi}|}\right)^{\frac{1}{d}}$. Like in the uniform core size method (42), we convect the particles in physical space, but diffusion is performed in reference space, so that with N particles, located in $\{x_j(t)\}_{j=1}^N = \{f(\hat{x}_j)\}_{j=1}^N$ we find an approximate solution to (41) by integrating the following set of ODEs:

$$\begin{aligned}
\frac{dx_j}{dt} &= u(x_j, t)\,, \\
\frac{dQ_j}{dt} &= \frac{\nu}{\hat{\varepsilon}^2} \sum_k \psi_{pq}^{\hat{\varepsilon}}(\hat{x}_j - \hat{x}_k) \left(\frac{m_{pq}(\hat{x}_j) + m_{pq}(\hat{x}_k)}{2}\right) [\hat{v}_j \hat{Q}_k - \hat{v}_k \hat{Q}_j]\,, \\
\frac{d\hat{v}_j}{dt} &= \hat{\nabla} \cdot \big(\underline{\Phi} u\big)(x_j, t)\, \hat{v}_j\,.
\end{aligned} \tag{44}$$

In the above equation Q_j and $\hat{Q}_j$ denote the particle strength in physical and reference space, respectively, related by

$$\hat{Q}_j = Q_j |\underline{\Phi}|(x_j)\,.$$

4.2 Particles with Adaptive Global Mappings

The framework introduced in section 4.1 involves analytical maps for which the Jacobian is readily available. If the mapping is invertible it can be changed at any time during a simulation. In the present method, particles are convected in physical space, remeshing and diffusion are performed in reference space, so that one can envision the following adaptive algorithm (see Figure 13): In this method, analytic, invertible maps could be dynamically adapted to the flow field. For instance, this could be done by adjusting their parameters: if $f(\hat{x}) = c_1\, e^{-c_2 \hat{x}}$ was used as a map to resolve a 1D boundary layer, we could adjust c_1 and c_2 to account for the growth of the layer. However, in order to have a map that is general enough to provide heterogenous flexible adaptation, it is desirable to use a finite dimensional map. Such a map could be described by a particle representation as

$$x(\hat{x}, t) = f(\hat{x}, t) = \sum_{j=1}^{M} \chi(t)_j\, \varphi_j(\hat{x})\,. \tag{45}$$

In the present method we introduce such a map and we require that the basis functions $C^2(\hat{\Omega}) \ni \varphi_j(\hat{x}) = \varphi(\hat{x} - \xi_j)$ be positive and have local support. Positivity is desirable as it assures that monotonicity of the nodes $\{\xi_j\}_{j=1}^M$ and node values $\{\chi_j\}_{j=1}^M$ leads to monotonicity of the map. The parameters in the map that are changed in the process of adaptation are the node values $\{\chi_j\}_{j=1}^M$. Due to the lack of simple invertibility we require that $\chi_j(t) \in C^1([0, T])$ for all $j = 1, \ldots, M$, *i.e.* that the adaptation of the nodes

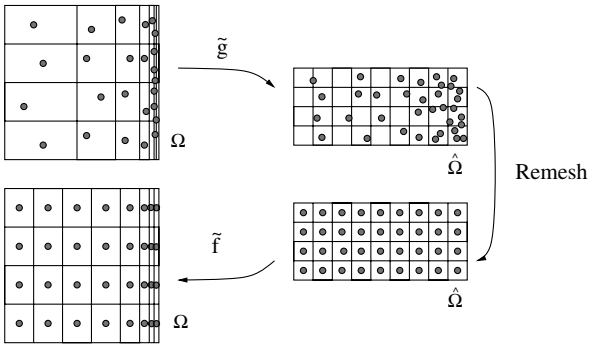

Figure 13: Illustration of an invertible adaptive mappings algorithm : Given a map f and an inverse map g and particles in Ω, 1. transfer particles to $\hat{\Omega}$ using an *adapted* inverse map g. 2. remesh particles in $\hat{\Omega}$ and perform diffusion 3. transfer particles to Ω with *adapted* map f

be continuous and differentiable. Using a map as described in (45) makes it impossible to leap back and forth from physical to reference space. However its differentiability enables us to cast the governing equations into reference space and solve the problem there, without the need of the inverse map $g : x \to \hat{x}$, *i.e.* without the need of invertibility.

4.3 Particles with Adaptive Mesh Refinement

In [21] different mappings were used in different parts of the computational domain, leading to different grid-sizes. Typically, for a flow around obstacles, one can think of local mappings adapted to a fine resolution of each boundary layer while another mapping could be used in intermediate zones with a stretched resolution away from the obstacles. These methods can be viewed as non-conforming domain decomposition methods with domain overlapping.

We propose here a method along the same lines but with piecewise constant grid-sizes adaptively adjusting to the solution. At every time-step blocks with grid-sizes, say, of the form $h2^{-l}$ are defined and discretized by particles with corresponding blob-sizes. As in the method proposed in [21], the overlapping of the blocks is essential to allow particles around the block-interfaces to exchange information and maintain a consistent approximation at the desired resolution everywhere. The exchange of information is done by interpolation at the remeshing stage that in our particle algorithms is done at every time-step. In the sequel we detail this procedure for a one-dimensional advection problem.

4.3.1 THE ONE-DIMENSIONAL ADVECTION EQUATION To describe more precisely the algorithm, we focus on equation (41) and we consider the case of two given blocks Ω_c and Ω_f:

$$\Omega_c =]-\infty, a[\quad , \quad \Omega_f =]b, +\infty[$$

with $a > b$. Ω_c and Ω_f are respectively coarse and fine resolution zones, with particle sizes H and h.

We will denote by $x^n_{c,i}$ (resp $x^n_{f,i}$) the locations of particles in the coarse (resp. fine) block at time $t_n = n\delta t$, and by $Q^n_{c,i}$ (resp $Q^n_{f,i}$) their strengths. A complete time-

step of the algorithm alternates particle motion and remeshing. Particles are pushed in both zones in the same way as in a single-resolution method. We assume that, after particles have moved they provide a consistent approximation of the solution q, at the corresponding resolution, in both domains. We denote by $\tilde{x}_{c,i}^{n+1}$ and $\tilde{Q}_{c,i}^{n+1}$ the particle locations and weights after advection in the coarse grid domain, with similar notations in the fine resolution domain. The goal of the remeshing step that follows is to make sure this this assumption will still be valid at the next time-step. For this purpose we need to extend Ω_c and Ω_f, such that there is a region of overlap. We thus define

$$\overline{\Omega}_c =]-\infty, a + l_1[\ , \ \overline{\Omega}_f =]b - l_2, +\infty[$$

We also set

$$\widetilde{\Omega}_c =]-\infty, a - l_3[\ , \ \widetilde{\Omega}_f =]b + l_4, +\infty[$$

with $l_i > 0$ such that $a - l_3 > b + l_4$, so that $\widetilde{\Omega}_c$ and $\widetilde{\Omega}_f$ overlap (see sketch on Figure 14). The remeshing step will remesh existing particles $\tilde{x}_{c,i}^{n+1}$, $\tilde{x}_{f,i}^{n+1}$ at regular locations $x_{c,i}^{n+1}$,

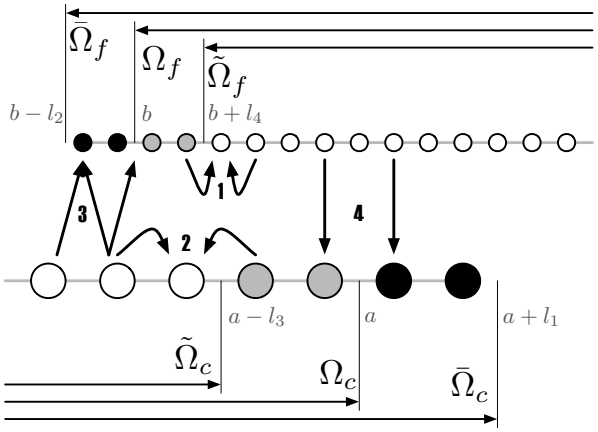

Figure 14: Sketch of coarse-fine domain decomposition. Arrows indicate how particles at the end of advection contribute to remeshed particle values in the different domains. Particles in grey (resp. in black) obtain their strength after remeshing from the domain with different resolution (resp. the same resolution). Numbers refer to the different stages in the remeshing algorithm.

$x_{f,i}^{n+1}$ respectively in $\widetilde{\Omega}_f$ and $\widetilde{\Omega}_c$, and create new particles in $\overline{\Omega}_c - \widetilde{\Omega}_c$ and $\overline{\Omega}_f - \widetilde{\Omega}_f$. More precisely, the remeshing algorithm proceeds as follows:

1. particle weights of fine-size particles in Ω_f are interpolated to give values for particles at regular locations on a fine grid in $\widetilde{\Omega}_f$.
2. particle weights of coarse-size particles in Ω_c are interpolated to give values for particles at regular locations on a coarse grid in $\widetilde{\Omega}_c$.
3. coarse-size particles in $\widetilde{\Omega}_c$ are used to compute values for particles at regular locations on a fine grid in $\overline{\Omega}_f - \widetilde{\Omega}_f$
4. fine-size particles in $\widetilde{\Omega}_f$ are used to compute values for particles at regular locations on a coarse grid in $\overline{\Omega}_c - \widetilde{\Omega}_c$

The remeshing in steps 1, 2 and 3 above are done by interpolation with the kernel used for remeshing particles. Steps 3 and 4 can be either done simultaneously with steps one and two, or follow these stages. In that case, stage 4 is just a sampling of the values obtained in stage 2.

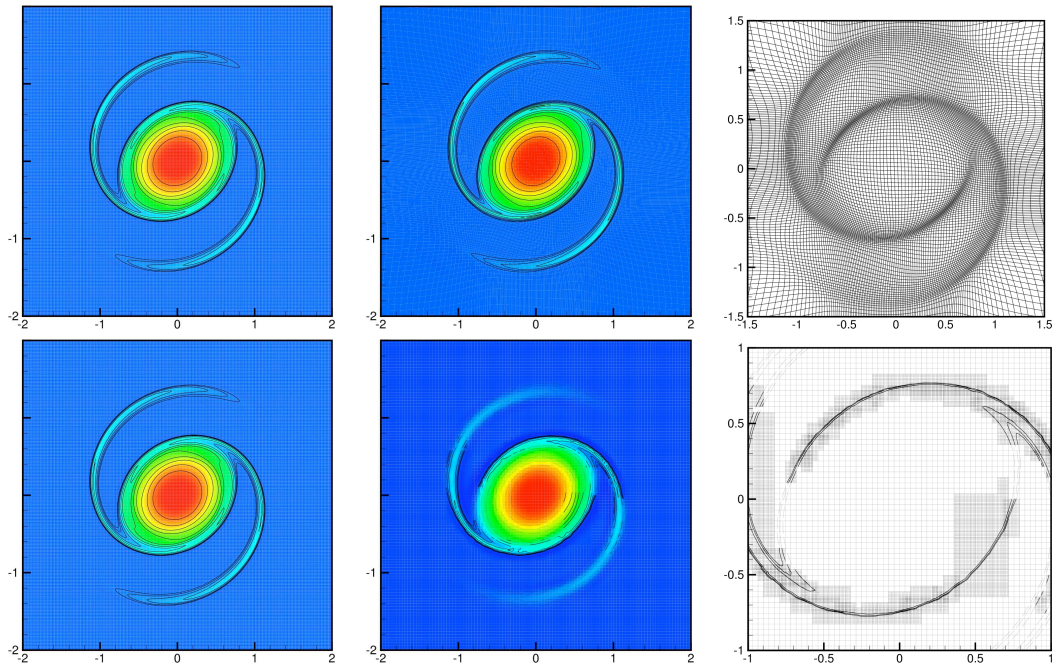

Figure 15: Top Row : Vorticity contours for the high-resolution reference calculation at t = 1.5 with N = 60'800 (*left*) and the corresponding contours for the adaptive global mappings based method with N = 15'100 (*center*). The adaptive mapping of the particles is shown on the right.
Bottom Row : Vorticity contours for the high-resolution reference calculation at t = 1.5 with N = 60,800 (*left*) and the corresponding AMR-based contours (solid lines correspond to refinement areas, dashed lines correspond to coarse areas) with N = 20,500 (*right*).The AMR remeshing of the particles is shown on the right.

To be more specific, let us clarify the conditions to ensure consistent transfers of information from one level of refinement to another. First, it is important to observe that the fine-size particles in $\widetilde{\Omega}_f$ do not suffice to give consistent remeshed values throughout $\widetilde{\Omega}_f$, unless a one-sided interpolation formula was used near the domain interfaces. We did not consider that option, as it would add some algorithmic complexity. On the other hand, using information from the coarse-grid domain, as prescribed in step 3 above, is consistent, provided the stencil needed for the interpolation remains in Ω_c. Similarly, in order for the remeshing from Ω_f to $\widetilde{\Omega}_f$ to be consistent, we need that the stencil centered at the boundary of $\widetilde{\Omega}_f$ does not extend outside Ω_f. These observations also give us overlapping rules that the domains Ω, $\widetilde{\Omega}$, and $\overline{\Omega}$ must obey. Let k be the number of points in each direction involved in the interpolation function used to remesh particles ($k = 2$) for the M_4' used here.

- consistency of stage 1 above requires $l_4 \geq (k-1)h$
- consistency of stage 2 above requires $l_3 \geq (k-1)H$

- consistency of stage 3 requires that $b - l_4 + kH \in \Omega_c$, that is $a - b - l_4 > kH$
- consistency of stage 4 requires that $a - l_3 - kh \in \Omega_f$, that is $a - b - l_3 > kh$

Denoting by $\delta = b - a$ the width of overlapping between the two domains, these conditions can be rewritten

$$hk \leq l_4 < \delta - kH \quad , \quad Hk \leq l_3 < \delta - kh \tag{46}$$

We deduce from these conditions that δ must satisfy

$$\delta > k(H + h) \tag{47}$$

Under the conditions (46) and (47), given the fact that particles $\tilde{x}_{c,i}^{n+1}$ $\tilde{x}_{f,i}^{n+1}$ with weights $\tilde{Q}_{c,i}^{n+1}$ and $\tilde{Q}_{f,i}^{n+1}$ were a consistent discretization of Ω_c and Ω_f, remeshed particles $x_{c,i}^{n+1}$ $x_{f,i}^{n+1}$ with weights $Q_{c,i}^{n+1}$ and $Q_{f,i}^{n+1}$ provide a consistent discretization of $\overline{\Omega}_c$ and $\overline{\Omega}_f$ at the corresponding resolution. It remains then to ensure that the next advection step will keep consistent particle values in Ω_c and Ω_f. If $\|u\|_\infty$ denotes the maximum advection velocity, a sufficient condition is clearly

$$l_1 \geq \|u\|_\infty \delta t \quad , \quad l_2 \geq \|u\|_\infty \delta t \tag{48}$$

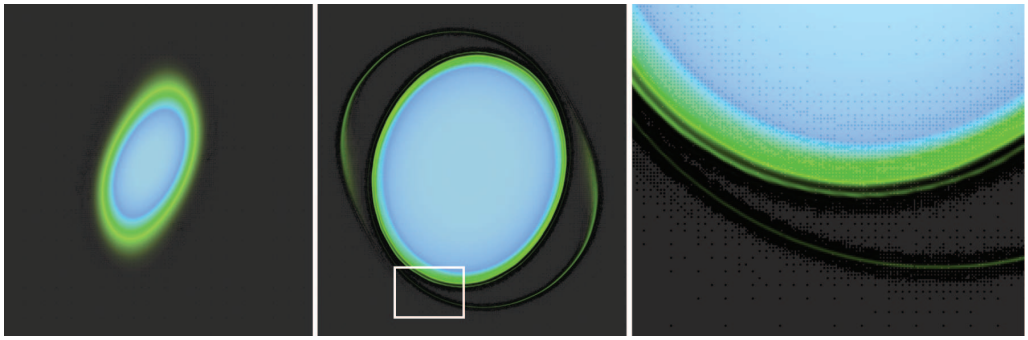

Figure 16: Inviscid Evolution of an Elliptical Vortex using Particle Wavelets. The last figure on the right shows a zoom at the vortex filaments demonstrating the different sizes of particles as identified by the wavelet analysis.

4.4 A Particle Wavelet Method

The "remeshing" procedure introduced by the Particle Mesh technique ensures particle overlap and preserves the accuracy of the method [48].

As a consequence, it distances itself from classical particle methods with an automatic adaptivity that is synonym of degraded accuracy. Adaptation can be reintroduced in Particle-Mesh techniques though, in a more controlled fashion, through adaptive mesh refinement techniques [4] or the Wavelet-based [5] approach presented here.

4.4.1 WAVELET-BASED ADAPTATION OF REMESHED PARTICLE QUANTITIES In the present framework we implement tensor-product wavelets $\psi^{l,\mu}$ and scaling functions φ^l on a sequence of $L + 1$ dyadically refined grids with mesh spacings $\{h_l\}_{l=0}^{L} =$

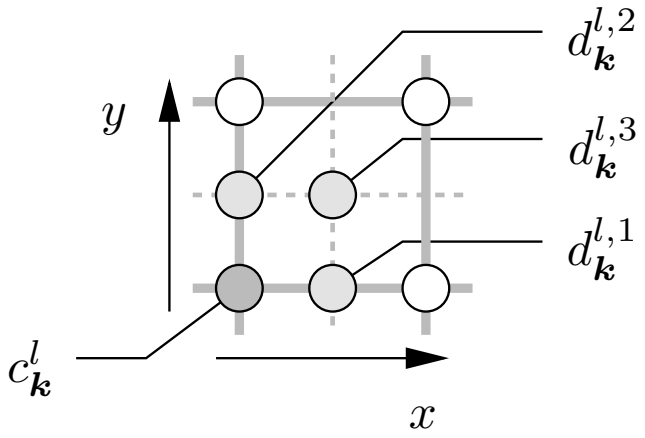

Figure 17: Each detail coefficient $d_k^{l,\mu}$, with $\mu = 1, \ldots, 2^d - 1$ corresponds to a specific grid point on the next finer level.

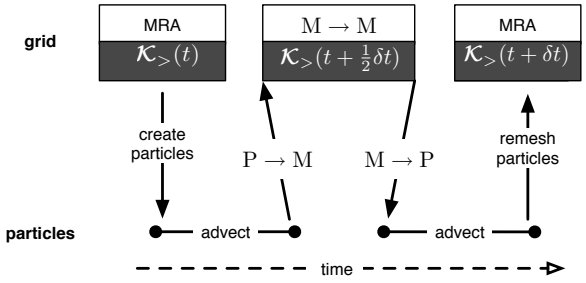

Figure 18: Particles are created on the adapted grid $\mathcal{K}_>(t)$ and advected. In the context of a two-step ODE integration scheme, the particle function representation is evaluated $(\mathrm{P} \to \mathrm{M})$ on an intermediate grid $\mathcal{K}_>(t+\frac{1}{2}\delta t)$ and the right-hand sides that are evaluated on this grid $(\mathrm{M} \to \mathrm{M})$ are interpolated back onto the particles $(\mathrm{M} \to \mathrm{P})$. At the end of the time step the particles are remeshed onto a mesh $\mathcal{K}_>(t+\delta t)$ on which the next MRA is performed.

$\{h_0\, 2^{-l}\}_{l=0}^{L}$ and grid points $k \in \{\mathcal{K}^l\}_{l=0}^{L}$. The scaling functions and wavelets are related as:

$$\varphi_j^l = \sum_k H_{j,k}^l\, \varphi_k^{l+1}\,, \qquad \psi_j^{l,\mu} = \sum_k G_{j,k}^{l,\mu}\, \varphi_k^{l+1}\,, \tag{49}$$

where $\mu = 1, \ldots, 2^d - 1$. The discrete filters $H_{j,k}^l$ and $G_{j,k}^{l,\mu}$ depend on the specific choice of wavelets employed. Using these bases the function $q(x)$ is expressed as

$$q(x) = \sum_{k \in \mathcal{K}^0} c_k^0\, \varphi_k^0(x) + \sum_{l=0}^{L-1} \sum_{k \in \mathcal{K}^l} \sum_{\mu=1}^{2^d-1} d_k^{l,\mu}\, \psi_k^{l,\mu}(x)\,. \tag{50}$$

The scaling coefficients c_k^l and detail coefficients $d_k^{l,\mu}$ can be efficiently computed using a Fast Wavelet Transform. In areas where the function $q(x)$ is smooth the detail coefficients of fine levels l will tend to be small, and a compressed representation of $q(x)$ is obtained by discarding detail coefficients for which $|d_k^{l,\mu}| < \varepsilon$. The error introduced by

this compression is bounded by

$$\|q(x) - q_{\geq}(x)\|_\infty \leq C_1\,\varepsilon \leq C_2\,\mathcal{N}^{P/d}\,, \tag{51}$$

where P is the order of the wavelets and $\mathcal{N}$ is the number of active coefficients.

It is important to note that each detail coefficient is associated with a grid point on the next finer grid, as illustrated in Figure 17. Hence, the compressed representation $q_{\geq}$ is inherently linked with a adapted grid, composed only of the grid points whose detail coefficients are significant, *i.e.* $|d_k^{l,\mu}| \geq \varepsilon$.

4.4.2 PARTICLE-MESH INTERPOLATION AND WAVELET MRA The wavelet-based MRA of the remeshed particle properties enables an enhanced multiresolution particle function representation. In order to allow for the emergence of small scales between two remeshing steps we additionally activate all children of the active grid points.

The computational steps ($\mathrm{P} \to \mathrm{M}$, $\mathrm{M} \to \mathrm{M}$, $\mathrm{M} \to \mathrm{P}$) are outlined in Figure 18 for the case of a two-step ODE integration scheme), and are based on level-wise operations. For a detailed description of multiresolution $\mathrm{P} \to \mathrm{M}$ and $\mathrm{M} \to \mathrm{P}$ interpolations we refer to [5].

5 PARTICLE METHODS FOR INTERFACES

The previous sections have been devoted to generic simulation tools in particle methods. We now come to issues that are more specific to problems and situations encountered in graphics applications. In these applications most often several physical systems, fluids, gas, rigid or elastic solids coexist and interact. We show how particle methods can be used both to simulate these different systems and to account for their interactions.

Interfaces (e.g. gas-liquid, flow-structure) deserve careful attention because they separate systems driven by different physical laws, and because it is the place where continuity conditions have to be enforced to preserve conservation of energy and momentum, and thus to give plausible physical dynamics. Generally speaking, there exist two classes of numerical methods to deal with interfaces : tracking and capturing methods.

Tracking methods solve the interface evolution equation in a Lagrangian fashion, for example by evolving marker particles. The origin of tracking methods can be traced to the 1930's and to calculations made by hand by Rosenhead [77] to describe the evolution of a vortex sheet in incompressible flows. These calculations have been followed 40 years later by the introduction of vortex methods [16] and the method of contour dynamics [98] while we consider that the Immersed boundary Method pioneered by Peskin [70] shares the spirit of these techniques. A fundamental problem of Lagrangian methods is the distortion of the locations of the computational elements resulting in an inaccurate description of the interface. Lagrangian interface capturing method require inserting or deleting points on the interface, with ad-hoc procedures. These methods are recognized to have difficulties to satisfy mass conservation. In *capturing methods*, the interface is determined by an implicit function that is advected in the computational domain. The most common interface capturing methods include Volume of Fluid [39] and Level Set methods [67, 83]. Volume of Fluid (VOF) methods are inherently linked to fluid mechanics problems and to Eulerian discretizations of the flow equations. They have enjoyed significant success in simulations of free surface and multiphase flow phenomena [79]. Level Set (*LS*) methods [85, 68, **?**, 66, 30, 86] employ an implicit function to describe the advection of the interface and are well suited to problems where interfaces undergo extreme topological changes. They have been applied with significant success to problems ranging from fluid mechanics to image processing and materials science (see the textbooks [?, 68] and references therein). The LS equation is commonly solved in an Eulerian framework by using high order finite difference methods, such as the fifth-order accurate Hamilton -Jacobi WENO schemes [9]. The accuracy of interface capturing schemes is reduced when the interface develops structures whose length scales are smaller than those afforded by the Eulerian mesh [75]. In addition time step limitations are introduced by the associated CFL condition for the discretization of the advection term. A number of remedies have been proposed to rectify this situation, such as high order ENO/WENO approximations, semi-Lagrangian techniques [91] and hybrid particle-level set techniques as introduced in [27]. In the latter work, the cells near the interface are seeded with marker particles in order to obtain sub-grid scale accuracy. This hybrid method has been shown to provide superior results for a number of benchmark problems in two and three dimensions. However, a number of open issues remain regarding the manner in which particles are introduced as well as the number of particles necessary to obtain a prescribed accuracy.

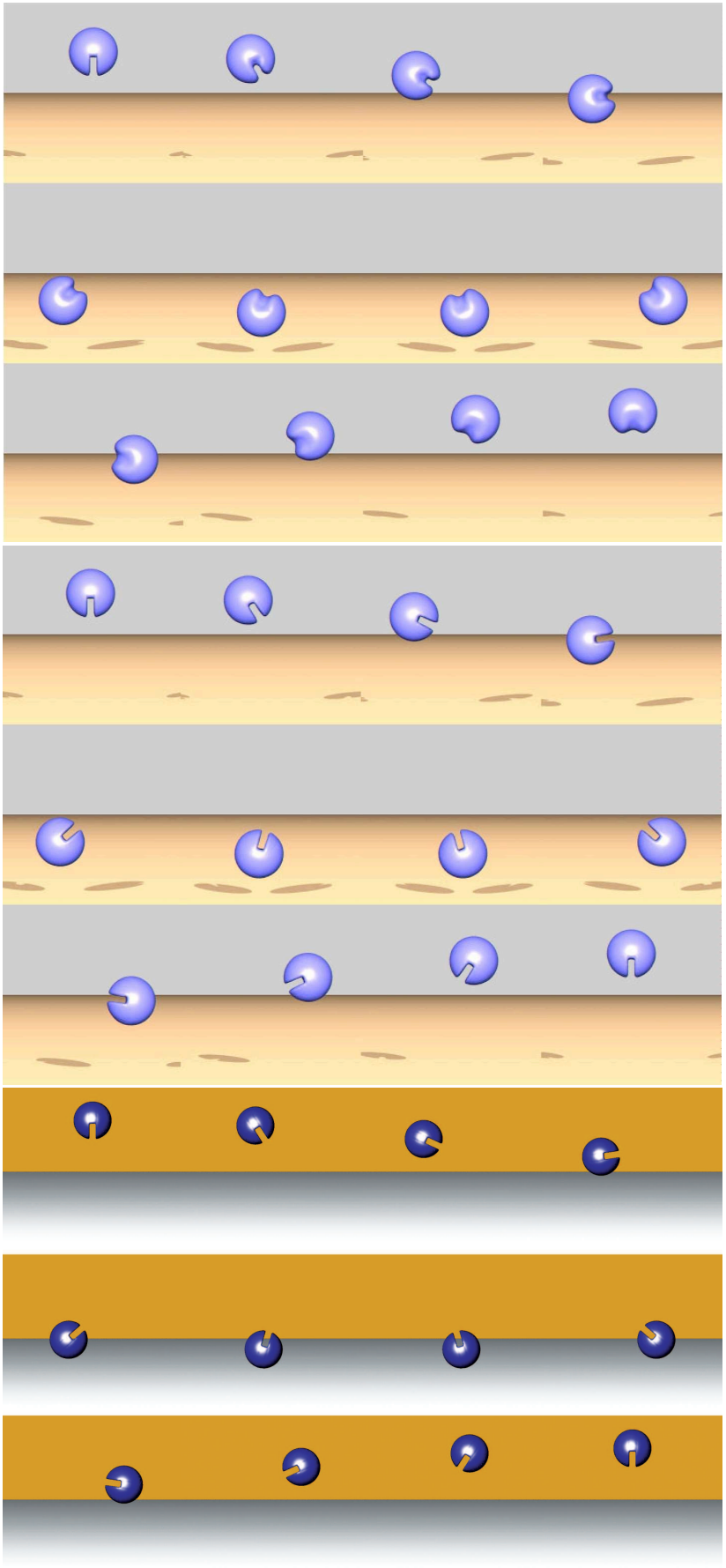

Figure 19: Zalesak's Slotted Sphere. Top: Eulerian Level Sets using 100x100x100 cells (from Enright *et al.* [27]). Middle: Hybrid Eulerian-Particle Level Sets of Enright *et al.* [27] using 100x100x100 cells and subscale particles. Bottom: Lagrangian Particle Level Sets [37]. The simulation uses 24351 Lagrangian particles in a narrow band, with an effective resolution of 64x64x64 cells.

The equations for the evolution of level sets can be cast, however in a lagrangian form enabling the use of all the tools, including remeshing, that we have developed so far in particle methods. As a matter of fact, the level set equation is an advection equation of the form (3), and as such qualifies for particle discretization.This Lagrangian Particle Level set technique was introduced in [37] and it is discussed in the following sections.

5.1 Lagrangian Particle Levels Sets for Interface Capturing

The Level Set method [67, 83] defines an interface $\Gamma(t)$ as the zero level set of a high dimensional, scalar function $\Phi(\mathbf{x}, t) : \mathbb{R}^3 \rightarrow \mathbb{R}$:

$$\Gamma(t) = \{\mathbf{x}\epsilon\Omega : \Phi(\mathbf{x}, t) = 0\}, \tag{52}$$

where Ω is the computational domain. The level set function has the following properties:

$$\begin{aligned} \Phi(\mathbf{x}, t) &> 0, \quad \mathbf{x} \in \tilde{\Omega} \\ \Phi(\mathbf{x}, t) &\leq 0, \quad \mathbf{x} \notin \tilde{\Omega}, \end{aligned} \tag{53}$$

where $\tilde{\Omega} \subset \Omega$ is an open region bounded by Γ. The motion of the interface is driven by a velocity field $\mathbf{u}(\mathbf{x}, t)$ as:

$$\frac{\partial \Phi}{\partial t} + \mathbf{u} \cdot \nabla \Phi = 0 \text{ for } t > 0, \tag{54}$$

$$\Phi(\mathbf{x}, 0) = \Phi_0(\mathbf{x}). \tag{55}$$

The specific form of the advection velocity field for the level set depends on the problem under consideration. This velocity is often considered as a function of the geometric properties of the surface, such as the normal and the curvature which are in turn computed via the definition of the level set

$$\mathbf{n} = \frac{\nabla \Phi}{|\nabla \Phi|} \quad \kappa = \nabla \cdot \frac{\nabla \Phi}{|\nabla \Phi|} \tag{56}$$

The function Φ_0 is usually chosen as the signed distance to the interface such that $|\nabla \Phi| = 1$. However, during its evolution, the level set function $\Phi(t)$ can lose the property of being the distance function [93]. Reinitialization schemes such as fast marching methods have been introduced [83, 84] in order to maintain this property. Usually the evolution of the level set function is computed using grid-based methods and the spatial derivatives in Eq. (56) are calculated by finite difference schemes [?].

In the context of the particle methods we propose in this class the level set equation can be expressed in a Lagrangian framework using the material derivative $\frac{D}{Dt} = \frac{\partial}{\partial t} + \mathbf{u} \cdot \nabla$ as

$$\frac{D\Phi}{Dt} = 0 \tag{57}$$

$$\frac{D\mathbf{x}}{Dt} = \mathbf{u}, \tag{58}$$

where $\mathbf{x}$ denotes the characteristics of the equation. The Lagrangian description of the level set equation is utilised in interface tracking methods. These methods encounter

difficulties when singularities are formed during the evolution of the interface and need to be complemented with suitable regularisation procedures [83] in order to recover a desired weak solution. In the Lagrangian Particle Level Sets (LPLS) [37] this regularisation is performed by a remeshing procedure (see Section 3).

5.1.1 RE-INITIALISATION AND REMESHING OF PARTICLE LEVEL SETS Reinitialisation of the signed distance function is a key issue in level sets and we discuss in this section how it can be combined with the remeshing step in the context of Lagrangian particle level sets.

The level set functions can be defined through the *signed distance function* (SDF) is defined by Eq. (53) along with the constraint that

$$|\nabla \Phi(\mathbf{x}, t)| = 1. \tag{59}$$

The absolute value of the SDF measures the distance to the interface and the sign of the function changes when crossing the interface. Alternatively level sets are defined by a *color function* (CF) [64] using a different characteristic constant on each subdomain separated by the interface, as in :

$$\Phi(\mathbf{x}, t) = 1, \quad \mathbf{x} \in \tilde{\Omega} \tag{60}$$

$$\Phi(\mathbf{x}, t) = 0, \quad \mathbf{x} \notin \tilde{\Omega}, \tag{61}$$

where $\tilde{\Omega} \subset \Omega$ is an open region bounded by the interface Γ. The SDF approach can be used for computing quantities such as surface tension. However, in cases where the distance information is not necessary, use of the CF can result in significant computational savings. In a particle method, the evolution of the LS function Φ amounts to evolving the particles on which it is discretised. Using smooth particles this amounts to expressing the level sets as:

$$\Phi(\mathbf{x}) = \sum_{p}^{M} v_p \Phi_p \zeta_\epsilon(\mathbf{x} - \mathbf{x}_p(t)) \tag{62}$$

The particle position $\mathbf{x}_p$, volume v_p and level set attribute Φ_p, evolve by the following system of ordinary differential equations derived from Eq. (54):

$$\begin{aligned} \frac{D\Phi_p}{Dt} &= 0 \\ \frac{Dv_p}{Dt} &= \langle \nabla \cdot \mathbf{u} \rangle_p \, v_p \\ \frac{D\mathbf{x}_p}{Dt} &= \mathbf{u}_p \end{aligned} \tag{63}$$

where $\langle \diamond \rangle_p$ denotes the derivative approximation on a particle p.

An immediate implication of the Lagrangian description is that *simulation of solid body rotation is exact*, except for the introduction of errors introduced by the particle initialisation and by the accuracy of the time integration [37]. The spatial derivatives used in Eq. (56) are computed by differentiating the regularization formula (62) or by using the mesh as it was discussed in the context of Hybrid Grid-Particle Methods.

As already discussed, remeshing is necessary when particles cease to overlap as they adapt to the flow map. Remeshing acts to suppress the evolution of scales that are smaller than the particle core and to prevent the formation of spurious scales resulting from non-overlapping particles, thus providing an entropy condition for the evolution of the level

sets. When considering a purely rigid body motion, e.g. the rotation of the Zalesak's disk, ΔH in Eq.31 is zero and no remeshing of the particles is necessary. This is an important feature of the present Lagrangian method, in particular when considering the use of level sets for simulations of solids.

During its evolution, the level set function usually ceases to be the signed distance function. Techniques such as fast marching methods [15, 85, 83] and re-distancing algorithms [92] have been introduced in order to maintain this property by reinitializing the level set function. A prerequisite for applying techniques such as the fast marching method is the regularity of the computational elements. This reinitialisation is straighforward when using an Eulerian description of the level set methods but , in general, it is not possible for an arbitrary particle distribution. Remeshing, however, offers the benefit that it distributes the particles on a cartesian mesh and it allows the implementation of the fast marching method. Remeshing also enables use of the level set redistancing algorithm introduced by Sussman [92] by solving the following equation on the regularised particle locations :

$$\Phi_t = sign(\Phi_0)\left(1 - |\nabla\Phi|\right), \Phi(\mathbf{x}, 0) = \Phi_0(\mathbf{x}) \quad \text{for} \quad t \to \infty. \tag{64}$$

The computational effort of this scheme can be significantly high when the time integration of Eq. (64) requires a small time step to ensure the convergence of the solution.

An alternative scheme, well suited for particle methods as it does not require that particles are distributed on a regular mesh, was developed based on the first order approximation of the derivative $\frac{\partial\Phi(\mathbf{x},t)}{\partial\mathbf{x}} = \frac{\Phi(\mathbf{x},t)-\Phi(\mathbf{x}_0,t)}{\mathbf{x}-\mathbf{x}_0}$ where x_0 is the position on the interface that minimizes $|\mathbf{x} - \mathbf{x}_0|$. Reformulation of this equations leads to a first order approximation of the distance to the interface that can be used in turn for reinitialization:

$$|\mathbf{x} - \mathbf{x}_0| = \left|\frac{\Phi(\mathbf{x}, t) - \Phi(\mathbf{x}_0, t)}{\frac{\partial\Phi(\mathbf{x},t)}{\partial\mathbf{x}}}\right| = \left|\frac{\Phi(\mathbf{x}, t)}{\frac{\partial\Phi(\mathbf{x},t)}{\partial\mathbf{x}}}\right|. \tag{65}$$

The approximation of the gradient of the level set that can be obtained on the particle locations using regularization formulas. An assessment of the various reinitialisation and remeshing schemes for particle level sets can be found in [37]

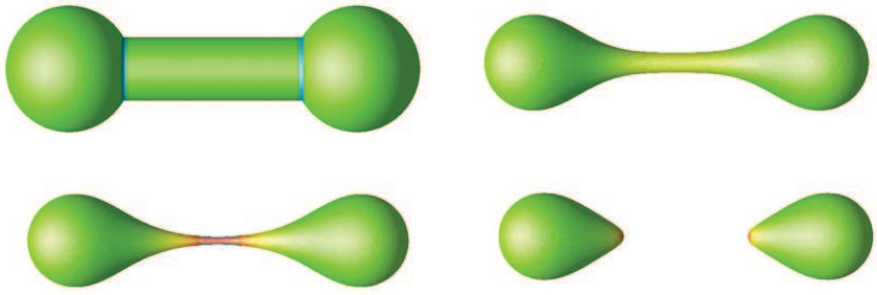

Figure 20: Evolution of a dumbbell shaped surface under mean curvature flow at $t = 0, 10^{-3}, 1.1 \cdot 10^{-3}, 1.26 \cdot 10^{-3}$ [37]

In order to reduce the computational cost of the level set method the computational elements are limited to narrow bands around the interface [83]. This concept is readily implemented in the present method due to the local support of the underlying particle

based functions. The remeshing provides a consistent process by which particles near the interface of the level set are being introduced while particles away from the interface are eliminated. The equations for the particle locations and volumes (Eq. (54)) are integrated using a Runge-Kutta method of 4th order in all cases. To reduce the computational cost involved with the reconstruction of the level set function from the individual particles (Eq. (62)) we use Linked List [40] and Verlet Lists [96]. The overall cost of the method scales linearly with the number of active particles. For 10^5 particles one time step of the method implemented in FORTRAN 90 requires 1.2 (in 2D) and 2 (in 3D) CPU seconds on an Apple Powerbook with a G4 processor of 1.25 GHz. The method is detailed in [37] where a number of comparisons with existing techniques are outlined.

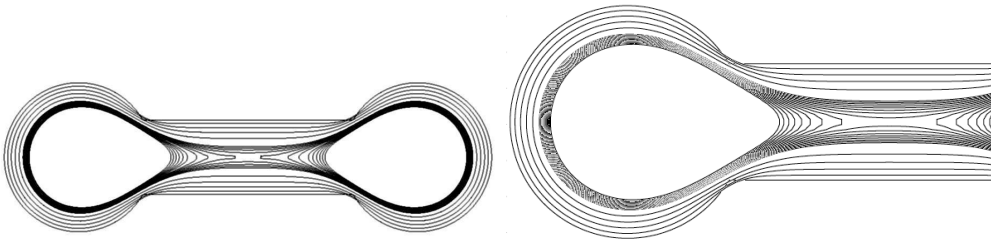

Figure 21: Crossections of the evolving dumbbell using LPLS (left) in comparison to Chopp and Sethian [14] (right).

Finally we wish to note here the proposed LPLS framework renders itself amenable to all of the advances described so far in particle methods. We wish to note in particular the applications of the particle-wavelet techniques [5] which provide an adaptive method for capturing interface that has unprecedented accuracy and efficiency (see below).

5.2 Applications of Lagrangian Particle Level Sets

We provide now a list of examples where Particle Level Sets have been employed for multiphysics simualtions. Further examples can be found in the following section discussing fluids and solid boundaries

5.2.1 BENCHMARKING THE LPLS In these notes we report results from the solid body rotation of a slotted sphere which corresponds to the well known problem of the 2D Zalesak's disk in a constant vorticity field [99]. The sphere has a radius of 0.15 and placed at (0.5, 0.75, 0.5) in a unit domain. The slot has a width of 0.05 and and length of 0.125. It rotates in the z=0.5 plane around the point (0.5, 0.5, 0.5). The velocity field describes a rigid body rotation evolving over 628 time units per revolution

$$\mathbf{v}(\mathbf{x}) = \frac{\pi}{314} \begin{bmatrix} 0.5 - y \\ x - 0.5 \\ 0 \end{bmatrix} \tag{66}$$

As reported in [27] the level set solution with 100x100x100 cells (Fig. 19) suffers from numerical diffusion which can be alleviated by the hybrid particle level set method introduced in [27] as shown in Fig. 19. In this method grid points at the interface are assisted by subgrid particles. In LPLS the discretization is equivalent to 64x64x64 and the slotted sphere maintains its sharp features (Fig. 19) as the particles follow the rigid body rotation, without any numerical diffusion effects, associated with the advection of

the level sets. Fig. 19 shows that the Lagrangian particle level set method performs very well on this problem. In this case, since the particle level set function remains a SDF there is no need for reinitialization. The present method is straightforward to implement, does not require seeding of the interface and it is exact in the case of solid body rotation.

To further illustrate the performance in three dimensions we simulate the collapse of a dumbbell that is a well known curvature flow example [14,82] as it exposes a singularity. The mean curvature flow pinches off the handle that separates into two pieces, which continue to shrink and finally vanish. Grayson [32] used this example to show that non-convex shapes in three dimensions may in fact not shrink to one sphere. The dumbbell is made up of two spheres, each of radius 0.3, and connected by a cylindrical handle of radius 0.15. The x-axis is the axis of symmetry. We choose a particle spacing of $0.09\overline{7}$ and a time step of $2 \cdot 10^{-5}$. The particles are reinitialized every 10th time step. Fig. 20 shows the surface as it appears initially, after shrinkage, when reaching the singularity and after the break up. The quality of the results is comparable with the finite difference solution of Sethian [14, 82] as seen in Fig. 21. The resolution of domain and the size of the time step are equivalent in both simulations. In the rest of this section we investigate a few applications which involve interfaces and physical processes on surfaces.

5.2.2 WAVELETS AND LPLS The wavelet particle methods [5] described in Section3 can be readily extended to LPLS as the level set function can be effectively considered a scalar subject to a multiresolution evolution.

In order to illustrate this approach we first considered the convection of a passive scalar (not only its interface !) in 2D, subject to a vortical velocity field [53]. The problem involves strong deformation of a initial circular scalar field which at the end of the simulation returns to the initial condition. The remeshing function and particle kernel were both chosen as

$$W(x) = \zeta(x) = \prod_{l=1}^{d} M_6'''\big((x)_l\big) \,,$$

where the fourth-order accurate interpolating function M_6''' is of higher order than the M_4' function at the expense of a larger support. The wavelets employed were also fourth-order accurate. The maximum CFL measured during the course of the simulation was 40.7. We also applied the presented method to the simulation of a propagating interface using a level set formulation. A "narrow band" formulation is easily accomplished with the present method by truncating the detail coefficients that are far from the interface. We consider the well-established 2D deformation test case which amounts to the propagation of a circle subject to the same velocity field as above. Figure 23 depicts the grid adaptation and comparing to Figure 22, one can clearly see the restriction of the refinement to a small neighborhood around the interface. We measure the error of the area encompassed by the interface at the final time and compare it against a non-adaptive particle level set method [37] and against the "hybrid particle level set method" of Enright *et al.* [27]. Figure 24 displays this comparison and we find that our adaptive approach performs favorably, which may be attributed in part to the adaptive character and in part to the high order of the method.

5.2.3 FREE SURFACE FLOWS We focus here on the case of variable density flows, typically liquid-gas systems. This is of course a case of interest in the animation of natural scenes. To be more specific we consider the case of variable density incompressible flows, in vorticity formulation and discretized by vortex particle methods.

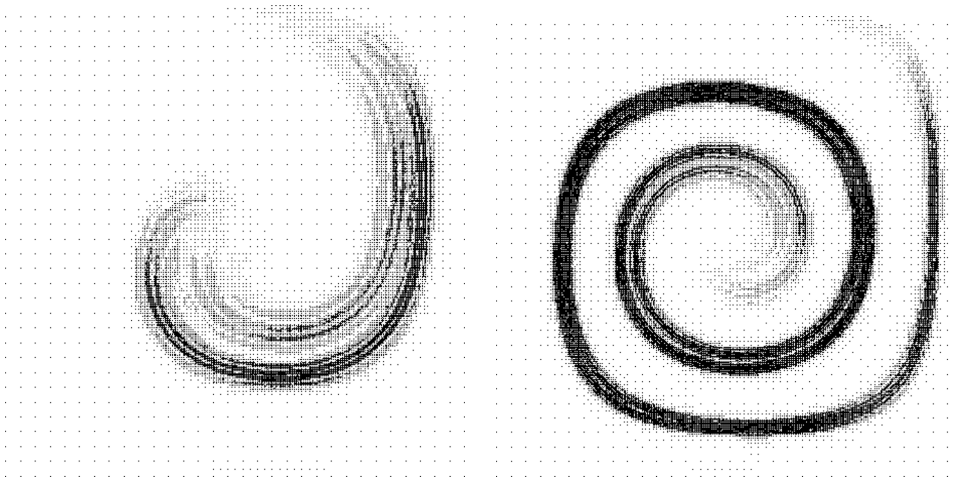

Figure 22: Active grid points/particles at two different times of the simulation of a passive scalar subject to a single vortex velocity field.

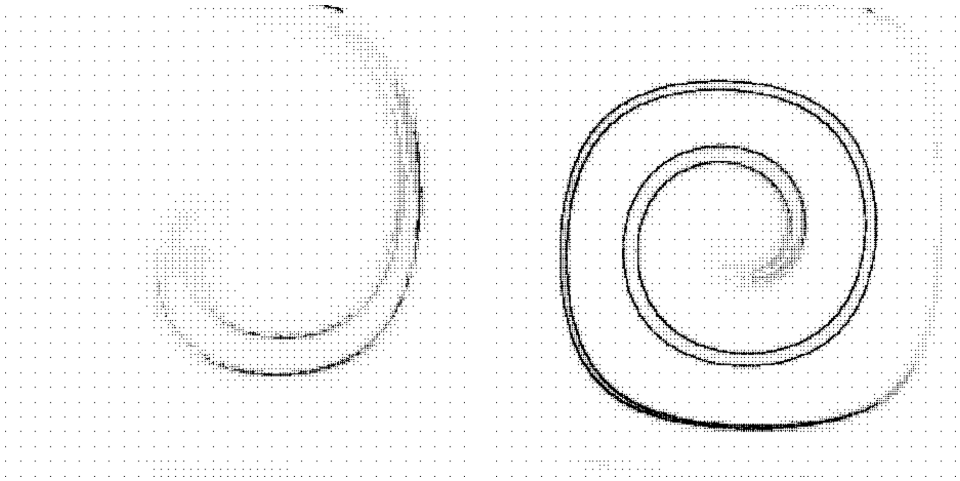

Figure 23: Active grid points/particles at two different times of the simulation of a propagating interface subject to a single vortex velocity field.

One can start from a formulation which combines transport equations for the vorticity and density gradient $\nabla\rho$, obtained by differentiating the velocity-pressure flow equations (as usual in vortex methods) and the mass conservation equation. In 2D, this gives the following system.

$$\frac{\partial\omega}{\partial t} + \text{div}(u\omega) = \mu\Delta\omega + \frac{\nabla p \times \nabla\rho}{\rho^2} + \tau\nabla\phi\times\nabla\left(\kappa(\phi)\zeta_\epsilon/\rho\right)$$

$$\frac{\partial(\nabla\rho)}{\partial t} + \text{div}(u\nabla\rho) + \left[\nabla u\right]^t\nabla\rho = 0$$

In the above system, κ is the surface tension, obtained through the level set function by (**??**) and ζ_ϵ is a usual regularization function. The right hand side of the vorticity equation shows that vorticity production results solely from density variation. This vorticity

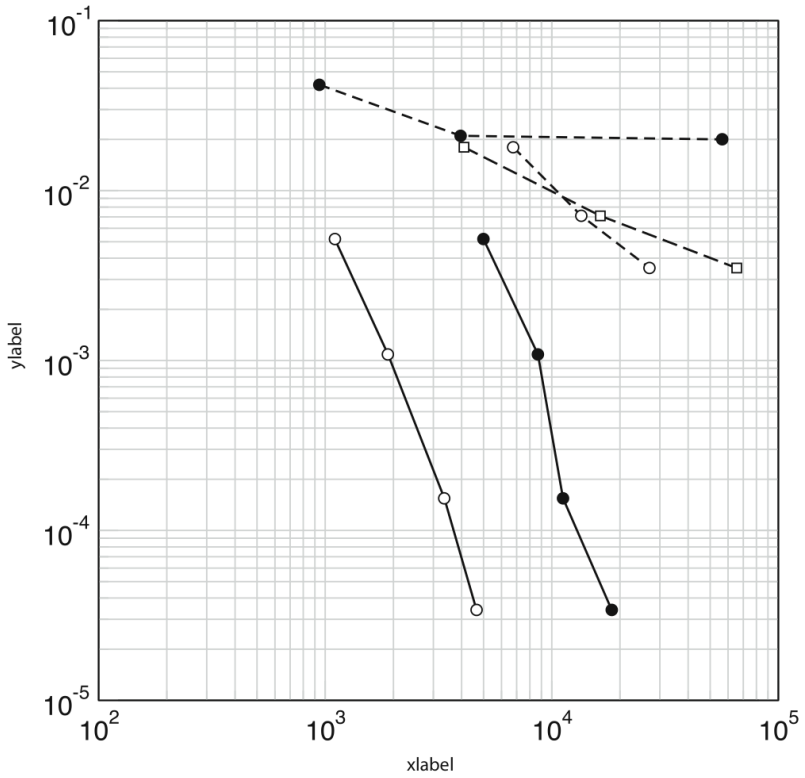

Figure 24: Plot of relative error of the area enclosed by the interface against degrees of freedom: Hieber & Koumoutsakos [37] (filled circles with dashed line , particles at time t=0), Enright *et al.* [27] (empty circles with dashed line , auxiliary particles at time t=0 and empty squares with dashed line , grid points) and present method (empty circles with solid line, active grid points at time t=0, filled circles with solid line, active grid points at the final time).

creation is the same mechanisms that produces vorticity - and thus motion - in a ring of smoke, or a cloud in air.

It is interesting to note that if the density is constant on either side of the interface S, or more generally on either side of an interface separating two different fluids, like water and air, this term will be concentrated on the interface. When a particle method is used, it will therefore create particles only a small domain. This is particularly visible on the so-called Boussinesq approximation of the problem. In this approximation, one assumes that the density variation are small. The vorticity created by this density variation is approximately solution to the equation

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \nabla)\omega = (\omega \cdot \nabla)\mathbf{u} + \nu \Delta \omega + \nabla \rho \times g + \lambda \nabla \times \chi_S(\overline{\mathbf{u}} - \mathbf{u}). \qquad (67)$$

Figure 25 represents the interface of two bubbles merging due to gravity and density

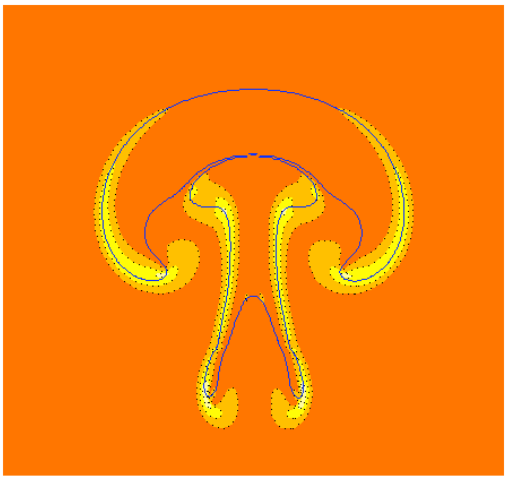

Figure 25: Merging of two bubbles : interfaces and and vorticity contours

contrast, together with a color representation of the vorticity, illustrating that the vorticity is localized around the interface. In this experiment, no reinitialization was used to preserve a signed distance function. Instead the cut-off function was normalized at each point by the norm of the gradient of the level function ϕ, as indicated in (65) .

5.2.4 VIRTUAL CUTTING USING LPLS The simplicity and efficiency of the proposed method enables simulations associated with virtual cutting of soft biological tissue [38] We consider a liver topology that was segmented from image data of the Visible Human Project. Based on the a triangular mesh of the topology, particles are placed inside the liver surface and they are assigned values following a CF approach. Fig.26 shows the surface reconstruction of the liver based on 3209 particles. In order to simulate cutting, whenever a medical device collides with one of the particles inside, the contribution of this particles it is removed from the superposition of Eq. (62). Hence reconstruction of the surface is computationally very inexpensive as the new surface is reconstructed according to:

$$\Phi(\mathbf{x})^{\text{new}} = \Phi(\mathbf{x})^{\text{old}} - \sum_{q}^{M} v_q \Phi_q \zeta_\epsilon(\mathbf{x} - \mathbf{x}_q) \tag{68}$$

where M denotes the (small) number of particles detected during the collision process.

This algorithm shows high efficiency and enables interactive simulations (Fig. 26) when the devices moved into the liver are not thinner than the particle spacing. Adaptive insertion of particles having smaller core size is necessary in order to refine this process.

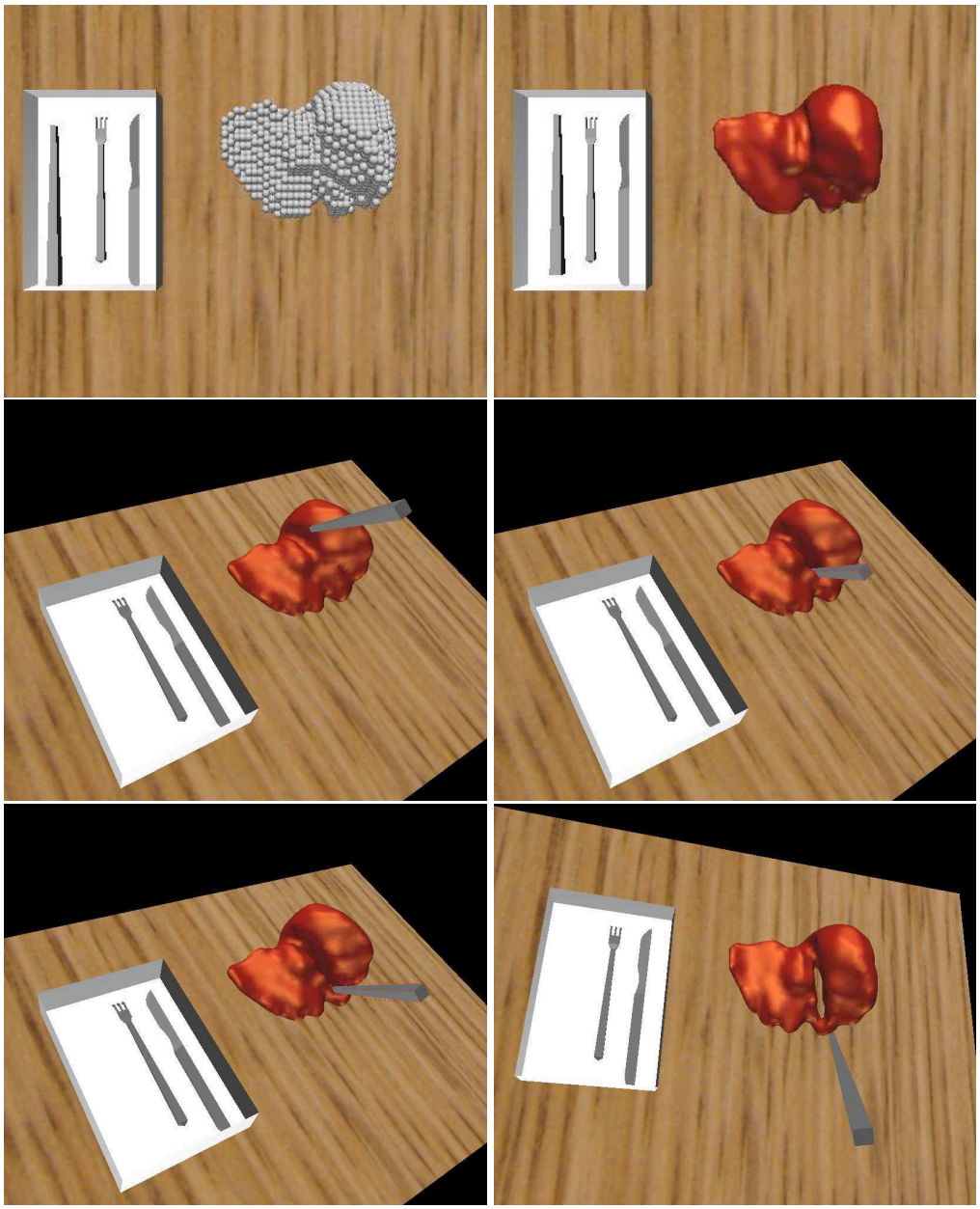

Figure 26: Particles assign with a color functions are removed from the superposition in real time when hit by an instrument [38].

5.2.5 SURFACE DIFFUSION AND GROWTH USING PARTICLES Simulations of Growth require the capability of modifying the surface and volume of the domain where the equations of the underlying physics are solved. Particle methods and in particular the Lagrangian Particle Level Sets that were introduced earlier provide a natural way of developing computational methods capable of handling diffusion, reaction and growth processes as they pertain to complex volumes and surfaces. An example of this applications is the simulation of diffusion on the surface of a biological cell organelle (the Endoplasmic Reticulum) that was reconstructed from microsections [78] (Fig.27). In the following we provide examples of growing surfaces and volumes, due to reactions diffusion processes that take place on their surfaces. Such physical systems provide models for processes such as tumor growth and tumor induced angiogenesis [58]. Further details of the formulations listed below can be found in [6].

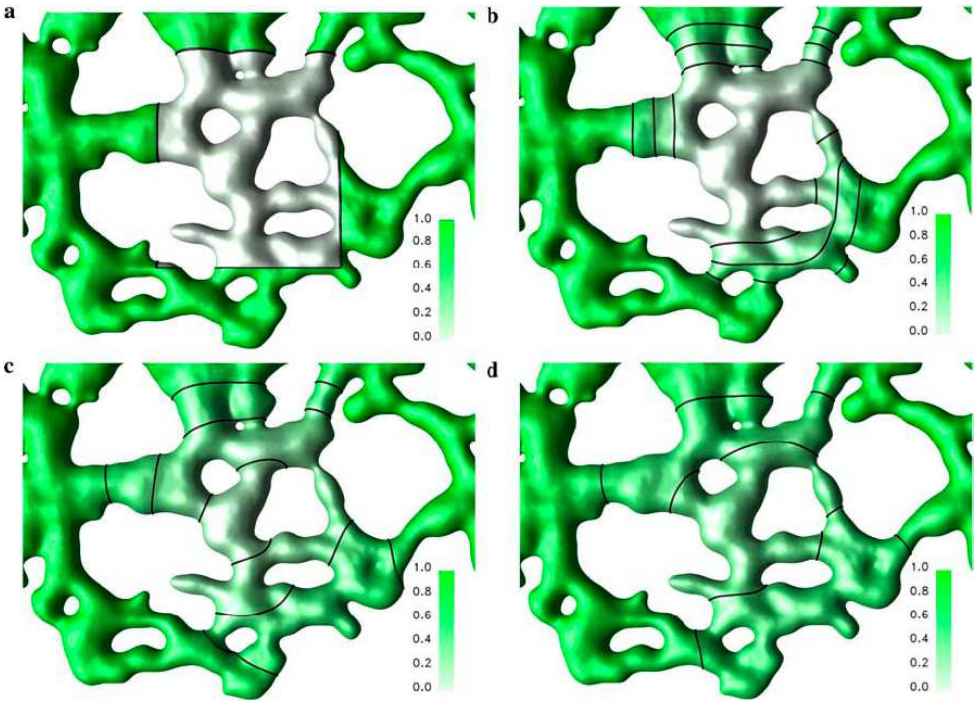

Figure 27: Simulation of the Fluorescence Recovery After Photobleaching (FRAP) diffusion process on the membrane of the cell's Endoplasmic Reticulum [78]. Solution on the membrane of an in vivo ER geometry at times t =0 (a), t =36 (b), t =216 (c), and t =441 (d). The computational diffusion constant is Dsim=1.0 and the diffusion operator is supported on 3x3x3 particles in a narrow band of half-width of 2h and extended to a larger band of k ? 3h every time step, using the second-order GMM extension method. The membrane and the concentration field are discretized using 1.7 million particles. The three lines indicate the 25%, 50%, and 75% recovery iso-lines..

We start by considering reaction diffusion systems on a closed smooth surface $\Gamma \subset \Omega \subseteq \mathbb{R}^3$. A general reaction diffusion system for N_S species on Γ can be written as,

$$\frac{\partial c_s}{\partial t} = F_s(c_1, c_2, \ldots, c_s) + \nabla_\Gamma \left(\underline{\underline{D}}_s \nabla_\Gamma c_s \right) , \tag{69}$$

where $s = 1, 2, \ldots, N_S$; F_s represents the reaction terms for species s undergoes and $\underline{\underline{D}}_s$ denotes the diffusion tensor associated with species s. For simplicity of presenta-

tion we will only consider homogeneous isotropic diffusion in the following, *i.e.* $\underline{\underline{D}}_s = D_s \mathbb{1}\,, \qquad s = 1, 2, \ldots, N_S\,,$ where D_s is a constant. Equation (70) then simplifies to

$$\frac{\partial c_s}{\partial t} = F_s(c_1, c_2, \ldots, c_s) + D_s \Delta_\Gamma c_s\,, \tag{70}$$

The operator Δ_Γ is called the Laplace-Beltrami operator on Γ.

We now consider a geometry that changes in time, *i.e.* $\Gamma(t) = \{x_\Gamma(t)\}$, with

$$\frac{dx_\Gamma}{dt} = u_n(x, c, \Gamma)\,. \tag{71}$$

Together with Equation (71) the governing equations of the full system are then given by

$$\frac{\partial c_s}{\partial t} + \nabla_\Gamma \cdot (c_s\, u) = F_s(c) + D_s\, \Delta_\Gamma c_s\,, \tag{72}$$

which can be rewritten as

$$\frac{\partial c_s}{\partial t} + \big((\mathbb{1} - n \otimes n)\, \nabla\big)(c\, u) = F_s(c) + D_s \nabla\big((\mathbb{1} - n \otimes n)\nabla c_s\big)\,, \tag{73}$$

We refer to [89] for a derivation. As we are going to solve this problem with particle methods it is more suiting to write Equation (73) as a conservation law:

$$\begin{aligned}\frac{\partial c_s}{\partial t} + \nabla \cdot (c_s\, u) &= (u \cdot n)\frac{\partial c_s}{\partial n} + c_s\, n \otimes n \nabla\, u \\ &\quad + F_s(c) + D_s \nabla\big((\mathbb{1} - n \otimes n)\nabla c_s\big)\end{aligned} \tag{74}$$

The reformulation from (73) to (74) necessitates the extension of both c_s and u from Γ to Ω. Furthermore it becomes evident that if we extend c_s and u such, that

$$\frac{\partial c_s}{\partial n} = 0\,, \text{ and } \frac{\partial (n \cdot u)}{\partial n} = 0\,, \tag{75}$$

we can simplify Equation (74) to

$$\frac{\partial c_s}{\partial t} + \nabla \cdot (c_s\, u) = F_s(c) + \nabla\big((\mathbb{1} - n \otimes n)\nabla c_s\big)\,. \tag{76}$$

In the context of particle methods that we have introduced in this class, the surface Γ is represented implicitly by the zero iso-surface of a level set function

$$\Gamma = \{\, x \mid \varphi(x) = 0 \,\}\,, \tag{77}$$

where φ is chosen as the signed-distance function of Γ so that $|\nabla\varphi| = 1$..

Surface properties can be evaluated exploiting the property (**??**): the surface normal and the mean curvature can be computed as $n = \nabla\varphi\,, \quad \kappa = \Delta\varphi$

The level set function is approximated as **??** as $\varphi^h(x,t) = \sum_p \Phi_p\, \zeta^h(x_p(t) - x_p)$ .. The discretization of Equation (76) using the particle function representation (**??**) leads to the following system of ordinary differential equations:

$$\begin{aligned}\frac{dx_p}{dt} &= u(x_p, t)\,, \\ \frac{dC_p}{dt} &= v_p\, F(c) + v_p D \nabla^h\big((\mathbb{1} - n \otimes n)\nabla^h c\big)\,, \\ \frac{dv_p}{dt} &= v_p\, \nabla \cdot u\,,\end{aligned} \tag{78}$$

which has to be numerically integrated in time.

As the level set is advected by Equation (**??**) it eventually loses its signed-distance property and it needs to be "reinitialized". The approach we employ is based on [93], where the following PDE is iterated to steady state:

$$\frac{\partial \varphi}{\partial \tau} + sign(\varphi_o)\,(1 - |\nabla \varphi|) = 0\,, \tag{79}$$

where $\varphi_o = \varphi(\tau = 0)$. Equation (79) is solved using the scheme formulated in [**?**]. As we are solving the conservation law formulation (76) to extend both the concentrations c and the velocities u off the interface Γ, while we require this extension to satisfy the requirements (75). As we are only interested in the concentrations on Γ it suffices to extend the quantities into a narrow band around the level set, which we define as

$$\Gamma_{\mathrm{e}} = \left\{\, x \;\middle|\; |\varphi(x)| \leq \gamma \right\}. \tag{80}$$

All calculations are restricted to this narrow band. We periodically extend the concentrations by using solving the following PDE:

$$\frac{\partial c_s}{\partial \tau} + sign(\varphi)\nabla\varphi \cdot \nabla c_s = 0\,, \tag{81}$$

As particles follow the flow map u their locations eventually become distorted and need to be regularized to ascertain convergence [48]. Regularization is performed by "remeshing" the particles periodically, *i.e.* resetting particle locations by interpolating the particle quantities onto a regular grid. For the level set function φ the situation is slightly different: the level set is not subject to a conservation law but to the non-conservative advection equation 54. We therefore remesh the level set function as:

$$\Phi_p^{\mathrm{new}} = \Big(\sum_{p'} \zeta^h(x_{p'} - x_p^{\mathrm{new}})\Big)^{-1} h^d \sum_{p'} \Phi_{p'}\, \zeta^h(x_{p'} - x_p^{\mathrm{new}})\,. \tag{82}$$

Alternatively we could compute the evolution of the particle volumes and remesh the values as

$$\Phi_p^{\mathrm{new}} = h^d \sum_{p'} \Phi_{p'}/v_{p'}\, \zeta^h(x'_p - x_p^{\mathrm{new}})\,. \tag{83}$$

Equation (82) however, leads to smoother solutions as it renormalizes ζ implicitly, thus removing the sampling error from the interpolation. The following algorithm describes an explicit Euler discretization of the systems in Equation 78 :

The anisotropic differential operator is discretized using second order finite differences and second order approximations of the diffusion tensor, resulting in a $3 \times 3 \times 3$ stencil. The minimal narrow band thickness is thus $\gamma = 2\,h$. For our calculations we used $\gamma = 4\,h$, so that the extension was only performed every third time step.

Reaction-Diffusion Simulations on a sphere To assess the accuracy of the present calculations we perform diffusion only simulations on the unit sphere.

We consider one species, *i.e.*

$$\frac{\partial c}{\partial t} = \Delta_\Gamma c\,, \tag{84}$$

with initial conditions

$$c(\theta, \phi, t = 0) = Y_1^0(\theta, \phi)\,, \tag{85}$$

for ($k = 0$ to $k\,\delta t = T$) **do**
  On the grid:
  Extend c from Γ to Γ_e using (81)
  Calculate reaction terms and diffusion terms in the narrow band
  Advance C_p with an Euler step of size δt
  if $mod(k, m) = 0$ **then**
    Create particles
    On the particles:
    Calculate u in Γ_e satisfying (75)
    Advance x_p with an Euler step of size $m\,\delta t$
    Interpolate φ_p and c_p onto the mesh
    Reinitialize the level set
  end if
end for

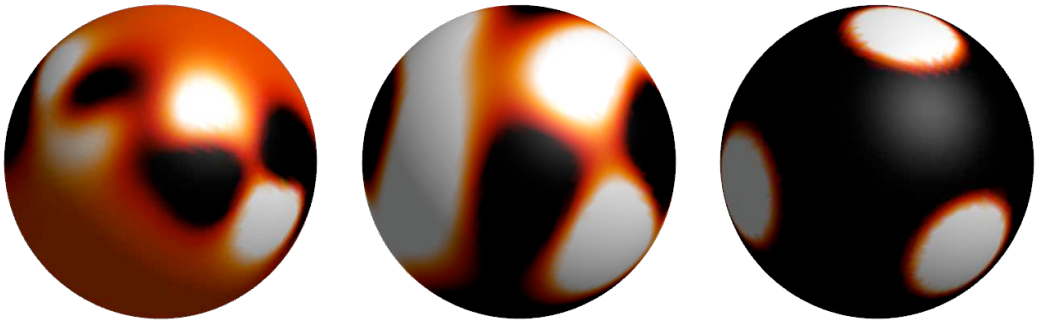

Figure 28: Simulation of the reaction diffusion system (**??**). From left to right: distribution of c_1 after iterations 1,000, 10,000 and 200,000.

Figure 29: Reaction - diffusion equations solved on the Dragon's surface

where Y_1^0 is the $(1, 0)$ spherical harmonic. The exact solution is given by

$$c(\theta, \phi, t) = e^{-2\,t}\, Y_1^0(\theta, \phi)\,. \tag{86}$$

For the time stepping we employ a TVD RK2 scheme. This case was also considered in [78], and we reproduce the second-order convergence obtained therein up to a constant (see Figure **??**). As a further example we consider two different reaction-diffusion

systems. The first system is the linearized Brusselator from [94]:

$$\begin{aligned} \frac{\partial c_1}{\partial t} &= \alpha\, c_1\,(1 - r_1\, {c_2}^2) - c_2\,(1 - r_2\, c_1) + D_1\,\Delta\, c_1\,, \\ \frac{\partial c_2}{\partial t} &= \beta\, c_2\,\left(1 + \tfrac{\alpha\, r_1}{\beta}\, c_1\, c_2\right) + c_1\,(\gamma - r_2\, c_2) + D_2\,\Delta\, c_2\,. \end{aligned} \tag{87}$$

The second system is an activator-substrate system from [**?**]:

$$\begin{aligned} \frac{\partial c_1}{\partial t} &= \rho_1 \frac{{c_1}^2\, c_2}{1 + \kappa_1\, {c_1}^2} - \mu_1\, c_1 + \sigma_1 + D_1\,\Delta_\Gamma\, c_1\,, \\ \frac{\partial c_2}{\partial t} &= -\rho_2 \frac{{c_1}^2\, c_2}{1 + \kappa_2\, {c_1}^2} + \sigma_2 + D_2\,\Delta_\Gamma\, c_2\,. \end{aligned} \tag{88}$$

We use the same parameters as Varea *et al.* [94]: $r_1 =$, $r_2 =$. The initial condition is given by $c_1 = c_2 = 0$ except on a band of width 0.2 centered on the equator, where the values of both u and v are uniformly randomly distributed in $[-0.5, 0.5]$. We obtain the same six-spot pattern as in Varea *et al.* [94] (Figure 28). The next system we consider is the activator-substrate system (88), for which we perform two different parameter sets, resulting in to spot patterns and stripe patterns, respectively on a square lattice in [?]. We demonstrate the flexiblity of our systems by solving the equations on the surface of the dragon (from the Stanford Depository) (Fig.29) and we observe similar patterning on the sphere $R = 0.3$.

Reaction Diffusion and Growth We now couple the deformation of the geometry to the reaction diffusion system by calculating the local velocity as

$$u = \alpha n\, c_1\,. \tag{89}$$

where α is a parameter that affects the patterns of growth. As $c_1 \geq 0$ this will always result in an outward motion of the geometry, and thus it will lead to an increase in surface area. This increase of surface area can be viewed as lowering the effective diffusion constants in the reaction diffusion system, as the reactions are generally independent of the surface properties.

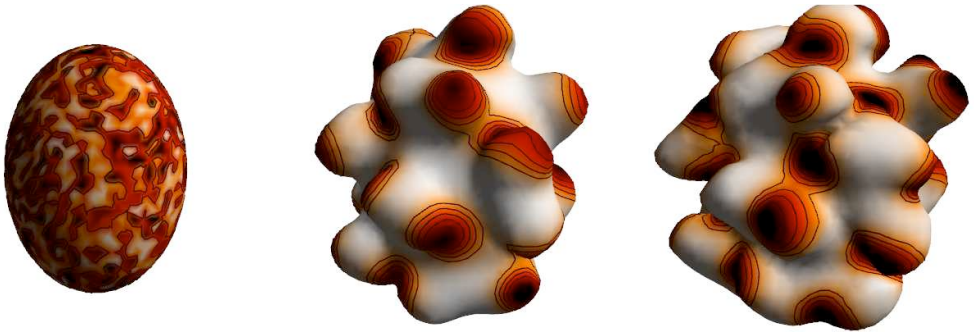

Figure 30: Growth simulations of the spot forming reaction diffusion system (88) for an ellipsoidal initial geometry. For iterations 0, 36,000, 124,000, and 170,000.

The only direct effect that growth has on the reactions is the decrease of the concentration in the sense of a decay term that depends on the growth velocity. By changing

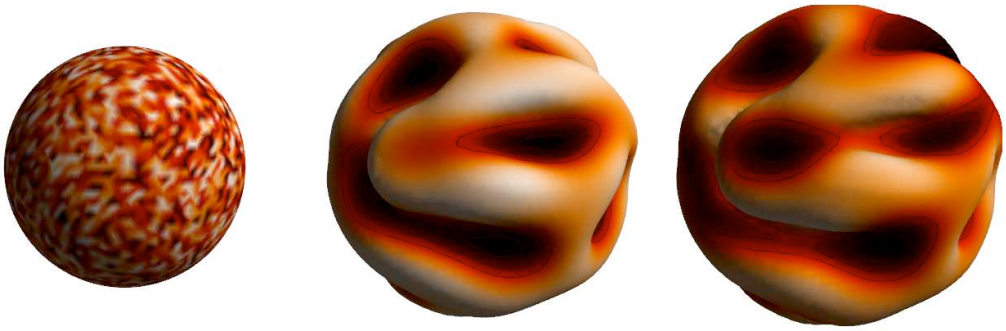

Figure 31: Growth of the stripe pattern of system (88) on a sphere. Iterations 0, 50,000 and 150,000.

the geometry, the diffusion and reaction constants as well as the velocity of the surface propagation a number of different patterns can be observed (see Fig.31 and Fig.30. For an extensive study of these systems the reader is referred to [6].

6 PARTICLE METHODS AND FLUID-STRUCTURE INTERACTIONS

In Multiphysics simulation, physical and numerical methods usually need to handle complex geometries. These geometries correspond to objects where the physics are of different nature, typically fluids and solids.The examples that come to mind are, with increasing complexity :

- a fluid around a fixed, or moving with a prescribed velocity, obstacle
- a fluid interacting with one or several rigid obstacle(s)
- a fluid interacting with an elastic obstacle.

In all cases, the problem amounts to prescribing and enforcing boundary conditions at the fluid/solid interface. As we have already seen , particle methods differ from classical grid-based methods in the discretization points have a meaning only through their collective behavior. This collective behavior is obtained at a given point by gathering the information of nearby particles. This has an immediate consequence for deriving and implementing boundary conditions : on may anticipate that assigning values at individual particles located on the interface will not be enough to enforce a desired boundary condition at these locations.

In these notes we focus on the two first problems. We first consider the case of solid and fixed boundaries. We will distinguish between grid-free and particle-grid methods, as in the questions related boundary conditions they lead to different approaches.

6.1 Fixed boundaries and grid-free particle methods

6.1.1 BACKGROUND Assume one needs to enforce a given boundary condition on a scalar field transported by a flow. This is the case in compressible viscous flows for each component of the momentum. To fix ideas, let us consider the boundary condition $u = 0$ at a solid wall. This is the classical no-slip condition, valid for a viscous flow, or for an inviscid flow where we use an artificial viscosity. If we assume that u is carried by a collection of particles, in principle u at a given point is obtained by *averaging* the strength of nearby particles. If this point is on the boundary, if the regularization blob is symmetric, which is in general the case, the cloud of particles is on both sides of the boundary. Since the flow lives only on one side, one must either consider a-symmetric clouds or include artificial particles on the other side of the boundary. The first solution, which amounts to using one-sided cut-off functions is in principle possible, but would not help to compute the field for points inside the domain at a distance less than the cut-off range. So we will not consider it.

The second solution uses ghost particles in the obstacle (see Figure 32). For the boundary condition $u = 0$, or any boundary condition which one way or another reduces to this one, there are clearly 2 choices : use ghost particles with zero strength, or with opposite strength. From numerical analysis one can expect the second choice to be more accurate. In particular, il one solves for diffusion with an initial condition which is odd across a plane boundary, one knows that this property will be preserved by diffusion and therefore the field with vanish at the boundary.

The difficulty is to create ghost particles. For a flat boundary it is straightforward, but in the general case it is not, unless some kind of mapping is explicitly known that

maps the fluid domain on a half-space. In that case, in principle one has to write the flow equation in the mapped coordinates, enforce the 'mapped' boundary condition for a flat boundary with image symmetric particles and finally go back to physical coordinates. This can be done for simple geometries (like cylinders) but become cumbersome for complex geometries, especially for moving and deforming geometries.

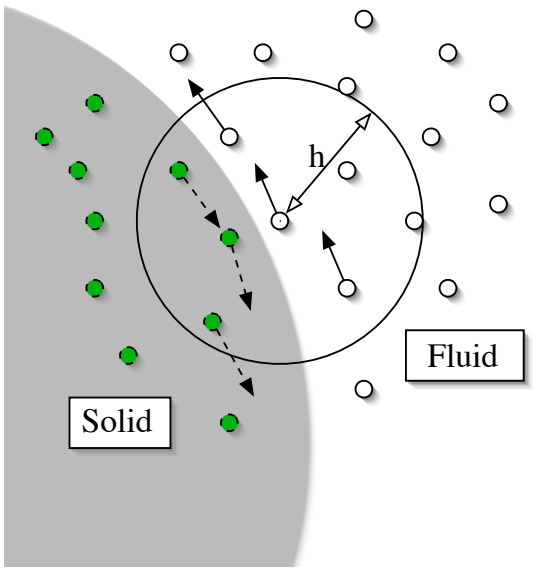

Figure 32: Boundary conditions and ghost particles (in green). The radius of the cut-off function is h

6.1.2 THE CASE OF VORTEX METHODS Let us now consider the particular case of the incompressible Navier-Stokes equation in vorticity formulation. The problem of boundary conditions faces an additional difficulty due to the fact that boundary conditions appear at two different levels : the boundary condition needed to determine the velocity from the vorticity, and the vorticity boundary conditions needed to advance the vorticity transport-diffusion equation. These two boundary condition should not be confused : they play a different role both form the mathematical and numerical point of view.

As already mentioned in session 1, the determination of the the velocity from the vorticity in general uses a boundary condition on the normal component of the velocity at the boundary. In the Biot-Savart based grid-free methods, this boundary condition is dealt with by the addition of a potential term in the integral representation of the velocity (equation (26)). Examples of integrale representations and associated integral equations for the potentials are given in the book [21]. Whatever method is chosen, the solution procedure always breaks down to to using markers on the boundary and to discretizing surface integrals by means of a panel method and finally solving a linear systems for potential values at the markers locations. Different recipes are available, depending on what kind of approximation -piecewise constant or linear - is used for the potential between two markers (in 2D) or in the panel centered around one marker (in 3D). This type of discussion is classical in the field of Boundary Element Method and we do not

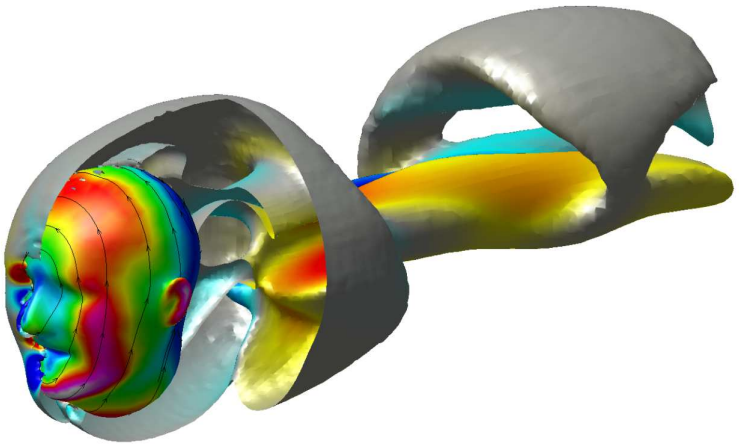

Figure 33: Vortex Methods simulations of the flow pas a head.

wish to enter in too much details.

We now come to the enforcement of the boundary conditions for the tangential velocity values. In flows around obstacles, the no-slip boundary condition is inherently related to the mechanism of vorticity creation in the flow. The common numerical approach for this problem is to mimic the physical process in a fractional step algorithm, where for each time step one successively solves for the advection and the diffusion of the vorticity:

- advection step: particles are pushed with velocities obtained by formula
- the slip at the boundary is obtained by evaluating tangential velocities on a set of markers
- a vortex sheet is created at the vorticity of the basis of this residual slip and fluxed out to vorticity particles in the fluid.

The advection step is solved by pushing particles. In the third step above, the vortex sheet is a layer of particles located on the boundary. A vortex sheet has the effect of creating a discontinuous tangential velocity field, which is exactly what is needed to cancel the residual slip. By 'fluxed out' we mean that this layer of vorticity is immediately distributed to nearby particles through diffusion. The clear-cut mathematical interpretation of this algorithm is to view the vortex sheet as a boundary condition for the normal derivative of the vorticity in the diffusion equation [50,49,20]. Note that in 3D there are two tangential components for the slip velocity and for the vorticity. The vortex sheet at the boundary thus involves two components, each of them cancelling the slip in the orthogonal direction in the tangential plane. Figure 34 summarizes this algorithm.

We note that the coupling of boundary elements and vortex methods enables the simulation of flows past complex geometries (see Fig. 33). The difficulty lies in the introduction of particles in the interior of the body when remeshing around a complex geometry using a regular grid. Empirical results however indicate that by eliminating the interior particles and reapplying the vorticity flux algorithm the method is able to recover the correct flow field.

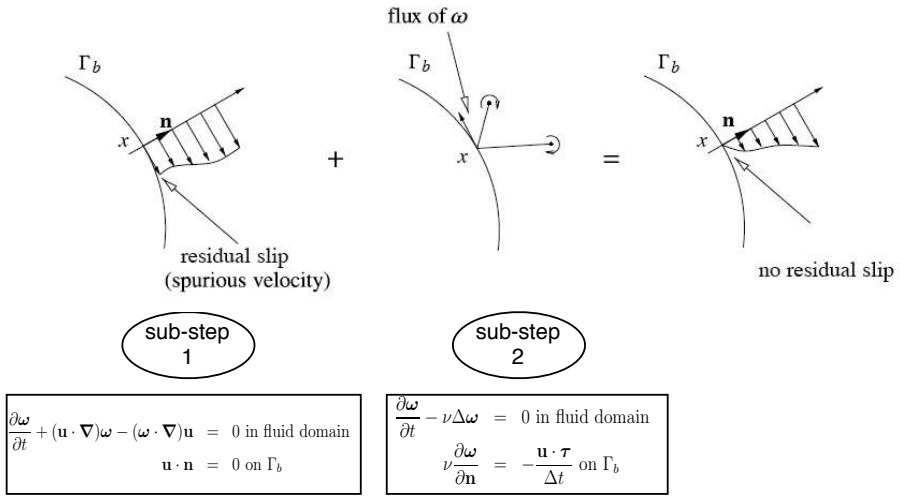

Figure 34: Fractional step algorithm to satisfy no-slip at the boundary

6.2 Fixed boundaries and hybrid particle-grid methods

Let us now consider the case of hybrid particle-grid methods and focus on vortex methods for incompressible flows. As we have seen, the grid can be used at different levels, either to compute velocity fields or to compute the velocity field and to remesh particles. To be more specific we consider the latter case, as we think it is the most suitable for both fast and accurate calculations. There are two cases to consider : the case where the grid fits with the boundary, and the case where the boundary is immersed in the grid.

6.2.1 BODY-FITTED GRIDS Like in the above section, we have to address two issues : the determination of the velocity from the vorticity and the vorticity creation to adjust the slip velocity. The approach is very similar to the case of grid-freee method, except that the condition for the normal velocity is enforced in the grid Poisson solver, using for instance the potential form of the velocity (26). To create vorticity in the flow, the residual slip is evaluated on markers introduced at the boundary by interpolation form grid velocity values. Then the algorithm proceeds along the same lines as in the grid-free case : a vortex sheet is created at the markers location then fluxed out on particles in the flow.

The efficiency of this approach is of course very much dependent on how easy a body-fitted grid can be constructed and how fast and easy to implement will be the Poisson solver associated to this grid.

A more flexible an practical method is given by immersed boundary approaches.

6.2.2 IMMERSED BOUNDARY TECHNIQUES - A FIRST APPROACH Because of their flexibility, immersed boundary techniques are receiving a lot of attention in CFD, both for grid-based or particle methods, in velocity-pressure as well as in vorticity formulations. The general idea is to consider the flow equations in the whole space, including walls and obstacles, and to replace boundary conditions on boundary points by a forcing term in the flow equations.

In some sense particle methods are by nature immersed boundary methods, at least in

the vorticity creation at the boundary. The methods that we have outlined does not assume that the boundary markers fit to the boundary. The flux of vorticity in the flow that will cancel the residual slip can actually be seen as a forcing term in the right hand side of the diffusion equation. The distinction between body-fited and immersed boundary vortex methods essentially arise for the enforcement of the normal velocity condition.

To be more specific, let us assume that we want to use a cartesian uniform grid both to remesh particles and to compute velocity fields and that the obstacle does not fit with grid-points, like in Figure 35. We follow the potential decomposition (26) but we look for potential extensions across the boundaries of the obstacles. We thus need an extension of the vorticity as well. For fixed objects, a natural choice is to extend the vorticity by 0 in the obstacles. In the discussion below we denote with a bar all the extensions of the quantities - domain, potentials, vorticity ...

The strategy to implement the boundary condition on $u \cdot n$ when the boundary Γ_b does not coincide with grid-points starts with the following observation: if $\overline{\phi}$ is a continuous harmonic extension of the exact flow potential across the boundary, in view of its gradient discontinuity we get

$$\Delta\overline{\phi} = \left[\frac{\partial\overline{\phi}}{\partial n}\right]_{\Gamma_b} \otimes \delta_{\Gamma_b}$$

where $[\cdot]$ means the jump across Γ_b and δ_{Γ_b} is the two-dimensional Dirac mass supported by Γ_b. The goal is to determine $\left[\frac{\partial\overline{\phi}}{\partial n}\right]_{\Gamma_b}$, to distribute it on grid points and then to solve the Poisson equation on the grid. One can proceed as follows : we first tag grid points which are at a distance less than the grid-size from the boundary. We denote by $\widetilde{\Gamma}$ the set made by these N grid-points. We then are looking for a function g, with support on $\widetilde{\Gamma}$ (see figure 35), such that the solution to the system

$$\Delta\overline{\phi} = g \text{ in } \overline{\Omega} \tag{90}$$

$$\frac{\partial\overline{\phi}}{\partial n} = 0 \text{ on } \overline{\Gamma} \tag{91}$$

satisfies

$$\frac{\partial\overline{\phi}}{\partial n} = -(\nabla \times \overline{\boldsymbol{\psi}}) \cdot n \boldsymbol{on} \widetilde{\Gamma}$$

This constitutes a linear system for the unknown function g over $\widetilde{\Gamma}$ of size N. A GMRES type iterative solver can be used to solve this system. Note that the vector-matrix product involved in the iterative method consists of the solution of a Poisson equation followed by the evaluations of potential derivatives on the tagged grid-points. The speed of convergence of a GMRES algorithm, and thus the efficiency of the method, is very much dependent on how well-behaved the linear system is. Unfortunately for complex geometries with angular points, the condition number of the system deteriorates. In that case, one needs to explicitly construct the inverse matrix to solve the liear system, something which becomes expensive if the body moves or changes shape.

We now come to alternative method that we believe is at the same time clear-cut from numerical analysis recipes, flexible to address complex geometries and easy to implement even in 3D. The method is based on penalization. Penalization methods are actually rather natural, even naive, ways to address boundary conditions on immersed boundaries.

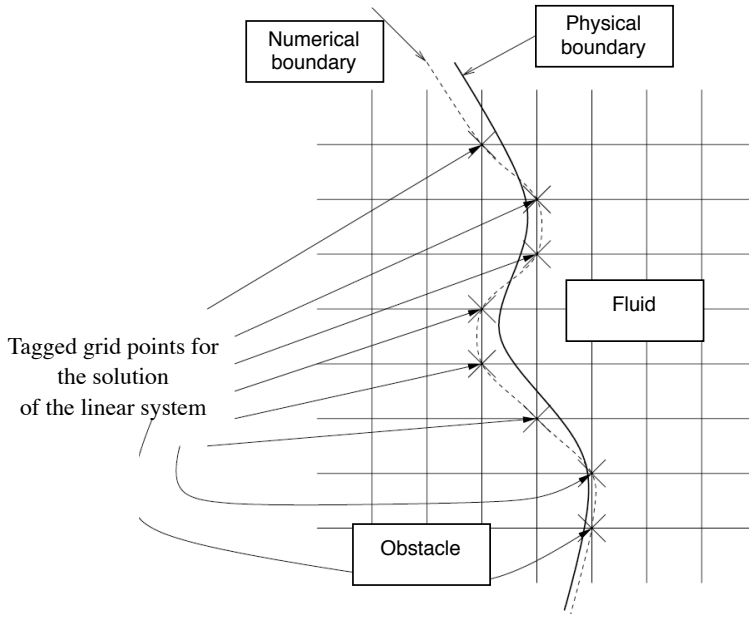

Figure 35: Immersed boundary

6.2.3 IMMERSED BOUNDARY TECHNIQUES - THE PENALIZATION METHOD The idea behind penalization method is to view obstacles, walls, etc .. as porous media which *absorb* the velocity in a small layer on the boundary of the obstacle [2]. From a mathematical point of view, it means assuming a flow everywhere, including inside the obstacles, and adding a term in the flow equation which drives the velocity back to zero - or whatever value is sought - inside the obstacles.

To be more specific, we consider, in a computational domain K, the case of an incompressible flow around an object S with prescribed velocity $\overline{u}$ inside S. We denote by λ a penalization parameter, $\lambda >> 1$, and denote by χ_S the characteristic function of S (1 inside, 0 outside). The model equation is then the following :

$$\rho\left(\frac{\partial u}{\partial t} + (u\cdot\nabla)u\right) - \nu\Delta u + \nabla p = \rho\, g + \lambda\,\rho\,\chi_S(\overline{u}-u)\,\boldsymbol{for} x \in K \qquad (92)$$

coupled with the incompressibility condition

$$\text{div } u = 0 \text{ for } x \in K \qquad (93)$$

In the above equation ρ denotes the density, with value ρ_S in S and ρ_F in the fluid outside S, $F = K - S$. If we want to use vortex particles to handle this penalization model, we first need to derive the vorticity form of (92). Taking the curl of (92) we get

$$\frac{\partial\omega}{\partial t} + (u\cdot\nabla)\omega = (\omega\cdot\nabla)u + \nu\Delta\omega - \nabla p\times\nabla(\frac{1}{\rho}) + \lambda\nabla\times\chi_S(\overline{u}-u). \qquad (94)$$

This system has to be complemented by the usual system giving the velocity in terms of the vorticity :

$$\nabla\cdot u = 0 \boldsymbol{in} K; \nabla\times u = \omega\,\boldsymbol{in} K. \qquad (95)$$

The right hand side above exhibits two terms : a so-called *barotropic* term, resulting from the density variations, already seen in variable density and free surface flows, and

a term coming from the penalization. We now continue with the derivation of the model. Developing the term $\nabla\times\chi_S(\overline{u}-u)$ one obtains

$$\frac{\partial\omega}{\partial t}+(u\cdot\nabla)\omega-(\omega\cdot\nabla)u-\nu\Delta\omega=-\nabla p\times\nabla(\frac{1}{\rho})+\lambda\chi_S(\overline{\omega}-\omega)+\lambda\delta_\Sigma\, n\times(\overline{u}-u). \quad (96)$$

In the above equation we have set $\overline{\omega}=\nabla\times\overline{u}$ and n is the normal to the interface Σ oriented towards the solid. It is interesting to note that the right hand side of this equation contains, in addition to the density-driven term, a vorticity generator coming from the no-slip condition at the fluid-solid interface.This term is also localized at the interface. It is very much reminiscent to vorticity creation algorithms that we have outlined when we have discussed gird-free methods (Figure 34). A definite difference though with previously seen immersed boundary methods, is that here, both normal and tangential components of the velocity are handled by a single term in the vorticity equation. This greatly simplifies the algorithm. A drawback is that the condition on the normal component is possibly not satisfied with the same accuracy as when it is addressed by potential sources like in 6.2.2. Therefore it may happens that a few particles cross the interface. To avoid circulation defect in the method, it is therefore important that vorticity inside the solid domain is not discarded. We will outline the algorithm box, when we adress the more general situation of a fluid fully interacting with a solid body.

6.3 Interaction of a fluid with rigid bodies

The classical approach to adress fluid solid interaction (wether the solid is rigid or elastic) is to solve separately fluid and solids, with the associated physics, and to couple them through interface conditions that translate the continuity of forces and velocities. In general the description of the physics in the fluids is Eulerian, that is equations of the fluid are solved in Eulerian coordinates on a grid, while it is Lagrangian in the solid. The grid for the fluid has to adjust to the moving interface with the solid, at least in the normal direction (whence the name ALE for Arbitrary Lagrangian Eulerian methods). These methods are rather tricky to implement in particular in 3D and/or in presence of large defomations.It is clearly possible to define ALE particle methods, here the fluid is solved by a grid-free or hybrid particle-grid algorithm combined with solid solvers (for instance based on classical Finite Element solvers). However one may anticipate that these methods will face the problems of all ALE methods, with the additional difficulty inherent to particle methods for enforcing boundary conditions.

In the following we therefore focus on an alternative approach, which is to consider fluids and solids as a single, variable density, multiphase, flow. The different phases, fluids, elastic or rigid solids, are captured by level set methods. Interface conditions are enforced by penalization methods. We first consider the case of a single rigid body in an incompressible fluid, and we show how to model it with a vortex particle method

The starting model is the penalization model (94) just seen, with two additional features:

- the solid velocity is not given, but a result of flow forces, gravity and so on ..
- the solid is moving, and its boundary is captured by a level set function.

This means that we have to complement (94) by an expression for $\overline{u}$, an advection equation for a level set following the fluid/solid interface, and an expression of the penaliza-

tion term based on this level set. The level set equation is classically :

$$\frac{\partial \phi_S}{\partial t} + (\overline{u} \cdot \nabla)\phi_S = 0. \tag{97}$$

The level set function is initialized as a distance function to the initial boundary of the solid, positive inside and negative outside. The expression of the rigid motion $\overline{u}$ is actually rather simple : it is obtained by averaging the velocity (to obtain the translation part of the rigid body - and the vorticity (to obtain the rotation) inside the body :

$$\overline{u} = \frac{1}{|S|}\int_K \chi_S u\, dx + \left(J^{-1} \int_K \chi_S u \times (x - x_G)\, dx \right) \times (x - x_G). \tag{98}$$

We have denoted by χ_S the characteristic function of the solid which can be immediately recovered from the level set function : $\chi_S(x) = (1 + \boldsymbol{sign}(\phi_S(x)))/2$. The appealing

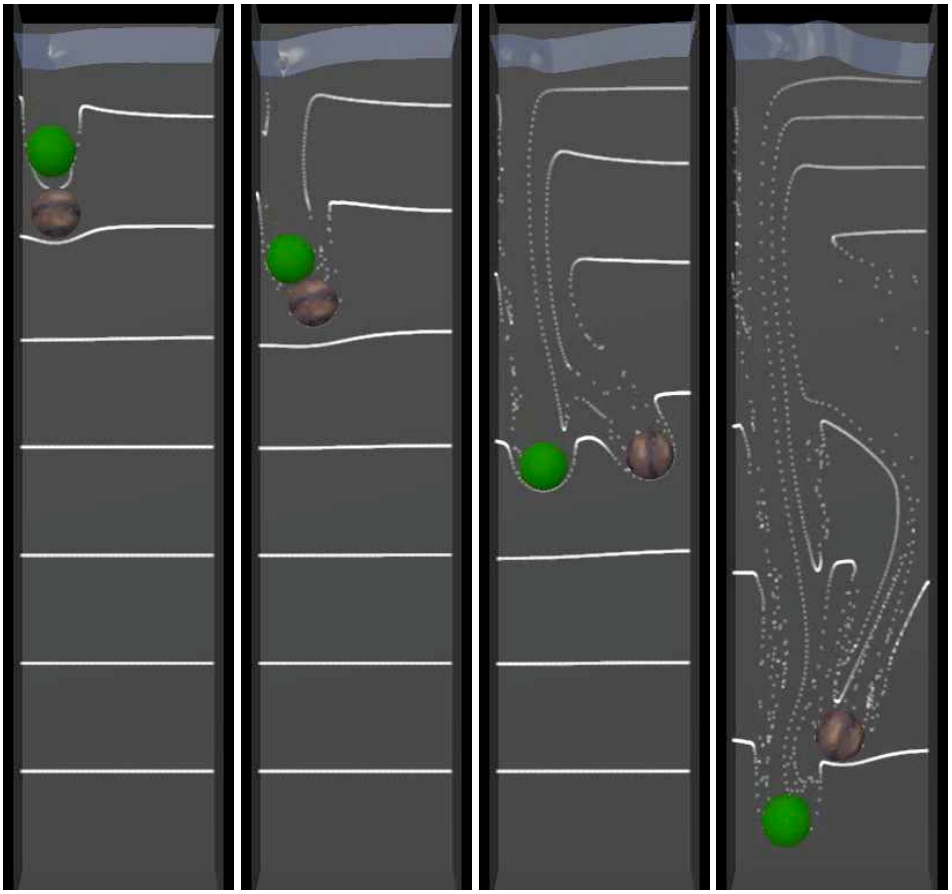

Figure 36: Kissing and tumbling of 2 spheres by a vortex level set method.

character of is model is the fact that it both relies on clear-cut numerical recipes (one can actually prove that it converges, when numerical discretization parameters tend to 0, to the exact physical solution) and is very easily implemented in a remeshed particle algorithm. Such an algorithm will combines the following steps, for each time interval :

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (u \cdot \nabla)\, \boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \nabla)u + \nu \Delta \boldsymbol{\omega} \tag{99}$$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = \lambda \nabla \times (\chi_S(\overline{u} - u)) \tag{100}$$

$$\frac{\partial \boldsymbol{\omega}}{\partial t} = -\frac{\nabla \rho}{\rho} \times \left(\frac{\partial u}{\partial t} + (u \cdot \nabla) u - g \right) \tag{101}$$

Equation (99) is solved by pushing particles (locations and weights) and remeshing them on a regular grid (note that this grid has not to comply with the walls, solid etc ..). At the end of this step, every thing (velocities and vorticity) is known on the grid so that the following steps can be solved by finite-differences on that grid. The pressure which is (unfortunately, unless a Boussinesq approximation (67) is used) necessary to solve for (101) is computed on the grid from the velocity. To improve the stability of the method when $\lambda >> 1$, it is recommended to use an implicit time-discretization of (100). Starting from an implicit formulation in velocity formulation and differentiating this formula one obtains the following scheme:

$$\widetilde{\boldsymbol{\omega}}^{n+1} = \nabla \times \left[\frac{u^n + \lambda \Delta t \chi_S \overline{u}^n}{1 + \lambda \Delta t \chi_S} \right].$$

The right hand side above is evaluated by centered finite differences on the grid on which particles have just been remeshed. In presence of several objects, it is of course very

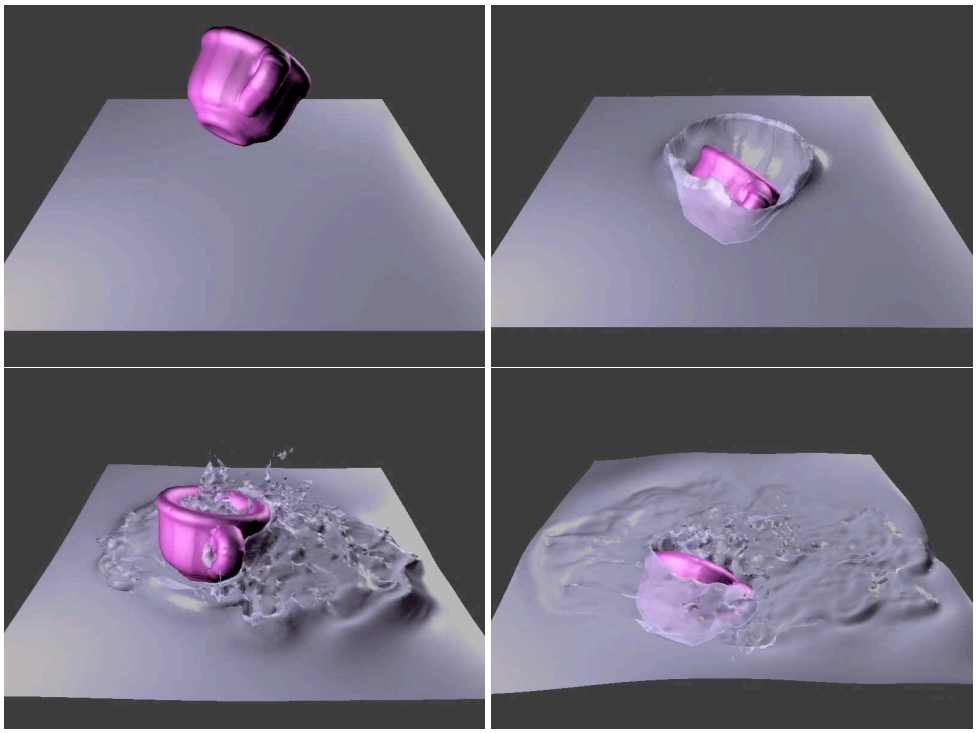

Figure 37: A cup falling into water. Level set functions are used for the air-water free surface and the cup-flow interface.

much desirable to be able to derive contact or collision methods from clear-cut physical models and numerics. This is of course independent of the chosen numerical method so we will spend too much time on this issue. Such a model derived and used in [17]

(details can be found in this reference). Contacts between several object S_i interacting in a fluid can be taken into account by adding to the vorticity equation an additional forcing term which is localized on the objects boundaries through the same level set functions which capture their interface and allow to enforce continuity of stresse. This term can be written as $\frac{\nabla \times f_{\text{col}}}{\rho}$ where the collision force is

$$f_{\text{col}}(x) = -\rho \sum_{ij} \frac{\kappa_{ij}}{\epsilon} \zeta \left(\frac{\phi_i(x)}{\epsilon} \right) \frac{\nabla \phi_j(x)}{\phi_j(x)} \exp\left(-\phi_j(x)/\epsilon\right). \tag{102}$$

Figures 36 and 37 illustrate the method. In the second case, level set are involved to capture both the air-liquid free surface and and cup-water, cup-aire interfaces. Although the physics of the different systems, the immersed boundary approach allows to view as a single flow described by its vorticity. Comparisons with grid-based methods reported in [17] on similar problems at the at same time validate the numerical approach and demonstrate the computational efficiency of the particle approach. This efficiency comes from the combined effects of the localization of the computational effort around the interface and of the use of relatively large time-steps.

7 GPU ACCELERATED PARTICLE METHODS

In this section we present an overview of the solver and we emphasize the handling of the particle-mesh interpolations by the GPU.

7.1 Representation

In GPGPU the computational elements are often mapped to textures. In this case the computational elements are both regularly spaced grid nodes and particles at arbitrary locations. Similar to [46] and [45], we employ an RGB texture to represent a set of particles where each texel contains the state of one particle. For two dimensional flows the red and green channel of a given texel represent the particle position, whereas the B channel indicates the transported vorticity. A one-to-one mapping between texels and grid nodes is used to represent the computational mesh with a texture.

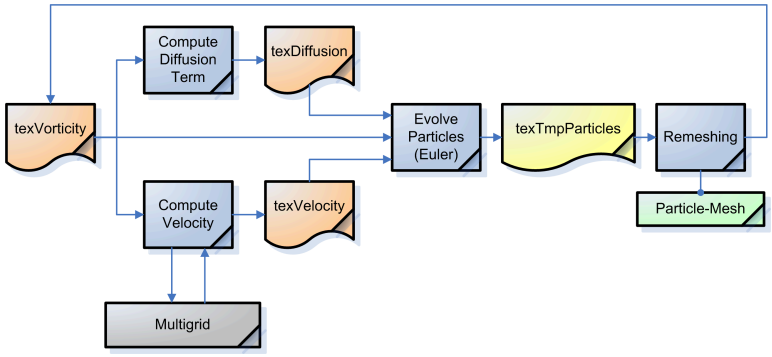

Figure 38: The components of the GPU-solver with first-order time integration.

7.2 Solver Overview

The main workflow of our GPU-Solver is illustrated in figure 38.
The core components of the solver are shown in blue, the gray component identifies a tool used as a "black box", whose subcomponents are not further explored. Each core component takes as input a set of textures and produces as output another set of textures. The managed data is represented with texture and is painted in orange or yellow. The yellow color signifies that the texture represents a particle set, therefore the RGB channels are (x_p, y_p, ω_p). The orange color indicates that the texture represents a grid, therefore it contains only ω. The green box represent the particle-mesh operation. Additionally to the texture `texTmpParticles`, the green box requires a vertex array of the same size.

7.3 Runge-Kutta time integration

Figure 39 illustrates the core components of the solver when a Runge-Kutta method is employed for the temporal advancement of the particle properties.
At each step the solver starts by reading the vorticity field that resides in `texVorticity`.
A significant part of the components employed in the Euler GPU-Solver are re-used for

the Runge Kutta GPU-Solver.
There are two more core components to create particle sets at each substep and at the final stage. Beside this, the solver has an additional array of textures for K_i.
It is easy to see that the time complexity is increased by the number of times that we must compute K_i and the main cost is attributed to the multigrid solver.

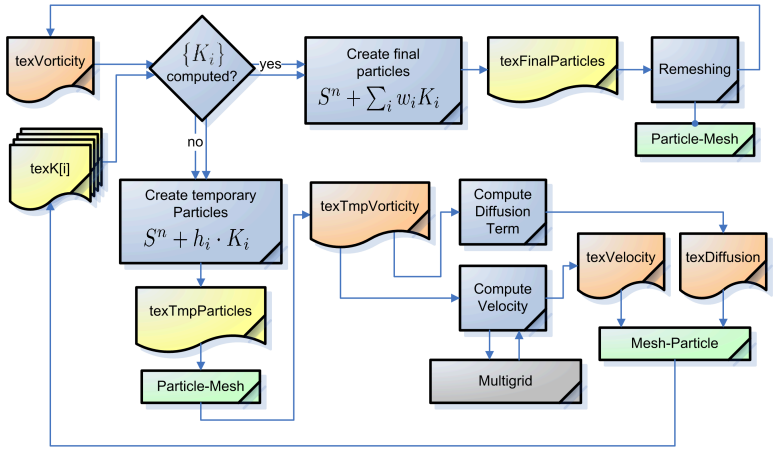

Figure 39: The components of the GPU-solver with k-th Order Runge Kutta time integration: a significant part of the component in Figure 38 is reusable here.

7.4 Particle-Mesh Operation

In the present method the particle-mesh operation is performed using a tensorial product of one-dimensional interpolating kernels. The contributions of all particles on each grid node are computed iteratively. The use of interpolation kernels with compact support implies that the contributing particles are located close to the grid nodes. If we are forced to avoid data-scatter the particle-mesh operation, then we would require an appropriate data structure like cell lists. In this way the particles can be ordered with respect to their positions. This approach has some drawbacks: the time complexity of the particle-mesh operation is sensitive to external parameters (e.g. cell size). Furthermore, the efficiency is decreased by the additional cost of building the location-processing datastructure and keeping it up-to-date.

On the other hand, this operation can be significantly simplified by utilizing data-scatter instructions. For each particle we can locate which grid nodes are affected and therefore add its respective contributions to these nodes. Based on this observation we present a concise method, to perform data-scattering particle-mesh operations on the GPU.
By employing *point-sprite* primitives, which allow the use of points rather than quads, we are able to generate texture coordinates which are interpolated across the point *Pointsprite:2003*. We start by having the status of the particle set stored in a texture `texParticles` and the mesh in the texture `texMesh`. In order to ensure that the number of vertices corresponds to the number of texels we use a vertex array with the size of `texParticles`.

The algorithm has the following steps:

1. Set the point size equal to the support of $W(.)$.
2. Enable the *point-sprite* drawing mode.
3. Attach the `texMesh` as render target and clear it with zeros.
4. Enable blending with 1 as source factor as well as destination factor.
5. Set the graphics pipeline as follows:
   Vertex shader:
   Read (x_p, y_p, ω_p) from `texParticles`. Store the transported vorticity ω_p as front color of the primitive and the location coordinates as the position of the vertex.
   Geometry shader:
   If a particle is close to the boundary, dynamically *clone* the particle to reproduce the right boundary conditions. If a particle has $\omega_p = 0$, discard the primitive.
   Fragment shader:
   Compute the vector distance $\mathbf{d} = (d_1, d_2)$ of the fragment with respect to the center of the point-sprite to produce $color = W(d_1)W(d_2)\omega_p$ as result, where ω_p is the vorticity of the current point sprite.
6. Draw the vertex array as point-sprites.

Because we are drawing point sprites, each vertex will be rasterized in a quad made of several fragments and different texture coordinate values. The distance between the center of the point sprite and the generated fragments is known at the fragment stage, and it is stored as texture coordinate. We re-scale appropriately the texture coordinate and we use it as an argument of $W(.)$. Since we know the quantity carried by the particle (as it is stored as primitive color) we multiply these together to obtain the contribution of the particle to that grid node.
Enabling the blending mode, we can sum each contribution from every particle to any node and obtain as a result an interpolated grid from values transported by the particle set.
For a given framebuffer (destination) pixel, the blending is performed as an atomic instruction. Therefore it cannot be performed in parallel with respect to the incoming source fragments. This could be a potential performance bottleneck. However, we can minimize this problem by reducing the maximal number of incoming fragments per framebuffer pixel. This is possible if the particles are not concentrated on a particular region so that their contributions will be spread uniformly in the framebuffer. This is automatically guaranteed in the remeshing stage: since by remeshing we uniformly redistribute the particles in the domain, thus avoiding that blending becomes a critical bottleneck.
The basic steps of the algorithm are illustrated in Figure 40. On the very left of this figure, we show the vertex array which is sent through the graphics pipeline. Then the vertex array is processed by the vertex shader. The latter assigns the correct value of each vertex by reading a texture where the particles status is stored.
The third image shows the vertex array after the geometry shader: the particles with zero vorticity are discarded (marked in red), and new particles are created to simulate the periodic boundary conditions (marked in green).
The last image shows the final result: after the rasterization the vertices have been transformed into point sprites, each fragment contains a contribution from a particle. The fragments are then summed together by a blending operation.

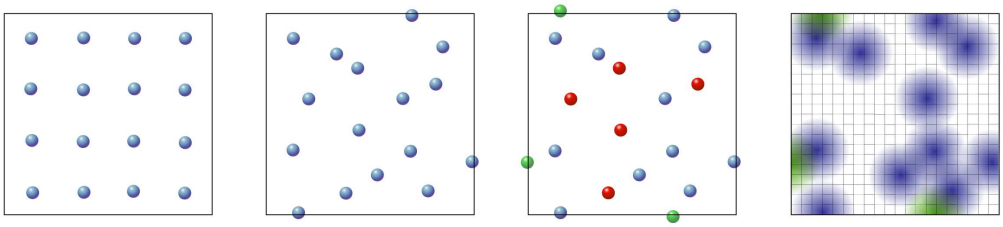

Figure 40: Particle-mesh on the GPU. From left to right: vertex array that is sent through the graphics pipeline, after the vertex stage, after the geometry stage and the result in the framebuffer.

7.5 Mesh-Particle Operation

Since the mesh-particle operation is essentially a gathering data operation, it can be done straightforwardly with a fragment shader, reading a texture representing the mesh and attaching the particle set texture as render target.
At the fragment stage, by performing texture dependent texture-fetches, we read a subset of mesh nodes and we compute $\mathbf{I}_M^P$, eventually obtaining the carried quantity for each particles.

7.6 Periodic Boundary Conditions

As depicted in Figure 40, the generated point-sprites are clipped in the rasterization. In the case of periodic boundary conditions this could be a problem, since we are discarding contributions from some particles. In order to overcome this problem, we use a geometry shader to check if the kernel assigned to each particle "touches" the boundary. If this is the case, we create a new particle with the same vorticity and with a position translated by one domain length. In this way we can generate exactly the contributions that are discarded at the rasterization stage.
The geometry shader presents an elegant choice to solve this problem, however it is not the only one. We could perform a 4-passes rendering with blending, where in each pass, a slightly shifted domain is considered and each particle has to be redrawn. This method gives exactly the same result as the geometry shader but it is much more expensive as every particle has to be rendered 4 times. Conversely, with the geometry shader, only the particles at the boundary have to be rendered 2 times (4 times for the negligibly small particle set at the corners).
Furthermore, with the geometry shader we not only have the opportunity to create particles *on-demand*, but also to *discard* particles, when they are unnecessary, i.e. when the transported vorticity is zero. This adaptivity additionally improves the performance of the proposed solver.
Even if the particle-mesh operations are cheap when compared to other components of the solver, the performance difference between these two approaches appears significant.

7.7 Solving the Poisson Equation

To solve the Poisson equation for the velocity field $\mathbf{u}$ we developed a periodic 2D multigrid solver [8] for the GPU. The GPU-Multigrid is designed for cell-centered elements, it has prolongation and restriction of order 4. We validated the GPU-Multigrid against different test problems, and we observed that on average, the relative residual was between 10^{-5} and 10^{-3} (in both L_2 and L_∞ norm). We have noticed that, for the same physical domain, higher resolution discretization causes bigger residual ($\sim 10^{-4}$). The most probable reason is the single floating point precision limitation in the arithmetic.

7.8 Performance

The presented solver involves a number of computational parameters, such as multigrid steps, order of time integration, etc. In order to quantify its performance we discuss three representative sets of parameters resulting respectively in: *Fastest*, *Trade-Off* and *Most Accurate* simulations. The *Fastest* set of parameters consists of a first order time integration, 2 V-Cycles with 2 Jacobi relaxation iterations at each level, per time step), the *Trade-Off* consists of a 2nd order Runge-Kutta with 2 V-Cycles (4 Jacobi relaxation iterations per level), per step. The last set of parameters corresponds to the one referred into Figure 10 (fourth order time integration, 4 V-Cycles with 4 Jacobi relaxation iterations at each level, per timestep). As indicated in Figure 41, one can achieve more than 25 FPS using a set of 1024x1024 particles with the *Fastest* set of parameters. The *Trade-off* configuration barely achieves 10 FPS with the same number of particles. The main decrease in performance is noticed by passing from 1024x1024 to 2048x2048 particles. For the *Most Accurate* configuration we observe the least change in performance, revealing that the texture size is not the most performance-critical parameter in this case. As we have mentioned the remeshing step can be performed either with a multi-pass approach or by utilizing a geometry shader. The diagram on the right of Figure 41 summarizes the performance of solving the random vorticity problem for both approaches as a function of the utilized number of particles. It is obvious that the geometry shader approach always is the fastest, in particular when we use 1024x1024 particles, where we obtain a speed up of 1.5, as on average the geometry shader has to render each particle just once. The multi pass approach, on the other hand, processes each particle 4 times (at least at the vertex stage).

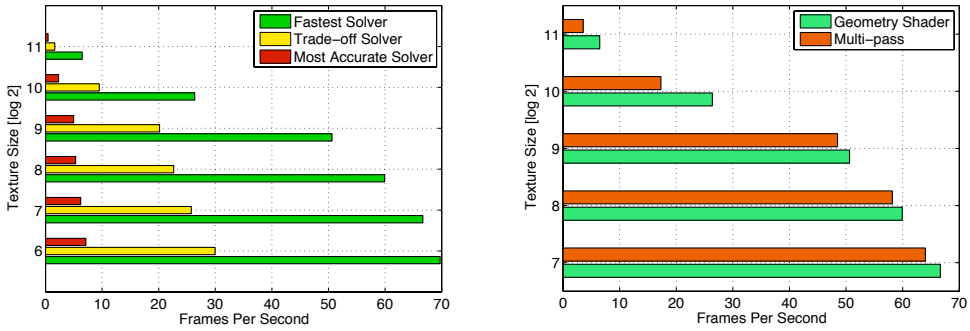

Figure 41: Overall performance measurements: on the left we compare three different configurations of our solver: *Fastest* (Euler, with a rough GPU-Multigrid), *Trade-Off* (2nd order Runge-Kutta, with an accurate GPU-Multigrid but few cycles) and *Most Accurate* (4th order Runge-Kutta, accurate GPU-Multigrid). On the right we compare the performance of the remeshing by using a multi-pass rendering method and by using a geometry shader.

8 CONCLUSIONS

We have implemented a hybrid particle-mesh algorithm for accurate and efficient simulations of incompressible vortical flows on GPUs. The solver implements a hybrid vortex methodology that utilizes both particles and mesh to best achieve accuracy and efficiency at the same time.The resulting fluid solver runs exclusively on the GPU and has second order accuracy in space and up to fourth order accuracy in time.

The algorithms have been validated in challenging benchmark problems demonstrating for example that even with the limitation of single precision arithmetic, we able to obtain a second order convergence in space in the case of the Taylor-Green and recover the correct solution for the thin double shear layer. In two different test problems we have shown the importance of adopting high-order time integration methods to achieve accuracy. Furthermore we have demonstrated that the performance of the GPU-solver depends critically on the set of computational parameters: The *fastest* set allows flow simulations with 1024x1024 particles at 25 FPS, whereas the *most accurate* only achieves 3 FPS, but with a significant greater accuracy.

The particles-mesh communication is performed in a straightforward and efficient way, by using a geometry shader, texture fetch at the vertex stage and the floating point framebuffer/blending.

Present work focuses on extending the solver to 3D domains, where the particles-mesh communication is not straightforward. In addition we aim to implement on the GPU multiresolution particle methods [5] and efficient particle methods capable of handling effectively complex, deforming geometries [23]. In addition accurate and efficient simulations on the GPU can be very helpful in applications related to optimization of flow problems using evolutionary algorithms [43] [65]. We envision that such implementation will help in the real time and accurate simulation and optimization of challenging fluid mechanics phenomena.

LITERATURE CITED

1. A. Angelidis and F. Neyret. Simulation of smoke based on vortex filament primitives. In *ACM-SIGGRAPH/EG Symposium on Computer Animation (SCA)*, 2005.
2. P. Angot, C. H. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *NUMERISCHE MATHEMATIK*, 81(4):497–520, Feb 1999.
3. J. T. Beale. A convergent 3-D vortex method with grid-free stretching. 46:401–424, 1986.
4. M. Bergdorf, G. H. Cottet, and P. Koumoutsakos. Multilevel adaptive particle methods for convection-diffusion equations. *Multiscale Modeling and Simulation*, 4(1):328–357, 2005.
5. M. Bergdorf and P. Koumoutsakos. A lagrangian particle-wavelet method. *MULTISCALE MODELING AND SIMULATION*, 5(3):980–995, 2006.
6. M. Bergdorf, I. F. Sbalzarini, and P. Koumoutsakos. Particle simulations of growth. *J. Computational Physics*, 2008 (submitted).
7. M. J. BERGER and J. OLIGER. Adaptive mesh refinement for hyperbolic partial-differential equations. *JOURNAL OF COMPUTATIONAL PHYSICS*, 53(3):484–512, 1984.
8. W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial: second edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.
9. S. Bryson and D. Levy. High-order central WENO schemes for multidimensional Hamilton-Jacobi equations. 41(4):1339–1369, 2003.
10. M. CARLSON, P. MUCHA, and G. TURK. Rigid fluid: Animating the interplay between rigid bodies and fluid, 2004.
11. A. K. Chaniotis, C. E. Frouzakis, J. C. Lee, A. G. Tomboulides, D. Poulikakos, and K. Boulouchos. Remeshed smoothed particle hydrodynamics for the simulation of laminar chemically reactive flows. 191(1):1–17, 2003.
12. P. Chatelain, A. Curioni, M. Bergdorf, D. Rossinelli, W. Andreoni, and P. Koumoutsakos. Billion vortex particle direct numerical simulations of aircraft wakes. *Computer Methods in Applied Mechanics and Engineering*, 197(13-16):1296–1304, 2008.
13. C. G. Chatelain P. and K. P. Particle mesh hydrodynamics for astrophysics simulations. *Int. J. Modern Physics C*, 18(4):610–618, 2007.
14. D. Chopp and J. Sethian. Flow under curvature: Singularity formation, minimal surfaces, and geodesics. 2(4):235–255, 1993.
15. D. L. Chopp. Computing minimal-surfaces via level set curvature flow. 106(1):77–91, 1993.
16. A. J. Chorin. Numerical study of slightly viscous flow. 57(4):785–796, 1973.
17. M. Coquerelle and G.-H. Cottet. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. *J. Comput. Phys.*, (in print), 2008.
18. G. Cottet. A particle model for fluid-structure interaction. *C.R. Acad. Sci. Paris*, Ser. I(335):833–838, 2002.
19. G. H. Cottet. A particle-grid superposition method for the Navier-Stokes equations. 89:301–318, 1990.
20. G. H. Cottet. Multi-physics and particle methods. *COMPUTATIONAL FLUID AND SOLID MECHANICS 2003, VOLS 1 AND 2, PROCEEDINGS*, pages 1296–1298, 2003.

21. G.-H. Cottet and P. Koumoutsakos. *Vortex Methods, Theory and Practice*. Cambridge University Press, 2000.
22. G.-H. Cottet, P. Koumoutsakos, and M. L. O. Salihi. Vortex methods with spatially varying cores. 162(1):164–185, 2000.
23. G. H. Cottet and E. Maitre. A level set method for fluid-structure interactions with immersed surfaces. *Mathematical Models and Methods In Applied Sciences*, 16(3):415–438, Mar 2006.
24. G.-H. Cottet and P. Poncet. Advances in direct numerical simulations of 3D wall-bounded flows by Vortex-in-Cell methods. 193(1):136–158, 2004.
25. S. Elcott, Y. Y. Tong, E. Kanso, P. Schroder, and M. Desbrun. Stable, circulation-preserving, simplicial fluids. *ACM TRANSACTIONS ON GRAPHICS*, 26(1), 2007.
26. M. Ellero, M. Serrano, and P. Espanol. Incompressible smoothed particle hydrodynamics. *JOURNAL OF COMPUTATIONAL PHYSICS*, 226(2):1731–1752, Oct 2007.
27. D. Enright, S. Marschner, and R. Fedkiw. Animation and rendering of complex water surfaces, 2002.
28. R. Fedkiw, J. Stam, and H. W. Jensen. Visual simulation of smoke. In E. Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 15–22. ACM Press / ACM SIGGRAPH, 2001.
29. N. Foster and D. Metaxas. Realistic animation of liquids. *Graphical models and image processing: GMIP*, 58(5):471–483, 1996.
30. F. Gibou, R. Fedkiw, R. Caflisch, and S. Osher. A level set approach for the numerical simulation of dendritic growth. *J. Sci. Comput.*, 19(1-3):183–199, 2003.
31. R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Month Notices Roy. Astron. Soc.*, 181:375–389, 1977.
32. M. Grayson. A short note on the evolution of surfaces via mean curvatures. 58:285–314, 1989.
33. L. Greengard and V. Rokhlin. A fast algorithm for particle simulations. 73:325–348, 1987.
34. F. H. Harlow. Particle-in-cell computing method for fluid dynamics. 3:319–343, 1964.
35. L. Hernquist. Some cautionary remarks about smoothed particle hydrodynamics. *ASTROPHYSICAL JOURNAL*, 404(2):717–722, Feb 1993.
36. J. L. Hess. Higher order numerical solution of the integral equation for the two-dimensional Neumann problem. 2:1–15, 1973.
37. S. E. Hieber and P. Koumoutsakos. A lagrangian particle level set method. *J. Computational Physics*, 210:342–367, 2005.
38. S. E. Hieber, J. H. Walther, and P. Koumoutsakos. Remeshed smoothed particle hydrodynamics simulation of the mechanical behavior of human organs. 12(4):305–314, 2004.
39. C. W. Hirt and B. D. Nichols. Volume of fluid (Vof) method for the dynamics of free boundaries. 39(1):201–225, 1981.
40. R. W. Hockney and J. W. Eastwood. *Computer Simulation Using Particles*. Institute of Physics Publishing, Bristol, PA, USA, 2 edition, 1988.
41. X. Y. Hu and N. A. Adams. An incompressible multi-phase sph method. *JOURNAL OF COMPUTATIONAL PHYSICS*, 227(1):264–278, Nov 2007.
42. Y. Kawaguchi. A morphological study of the form of nature. *SIGGRAPH Comput. Graph.*, 16(3):223–232, 1982.
43. S. Kern and P. Koumoutsakos. Simulations of optimized anguilliform swimming.

JOURNAL OF EXPERIMENTAL BIOLOGY, 209(24):4841–4857, Dec 2006.
44. R. A. Kerr. Planetary origins: A quickie birth for jupiters and saturns. *Science*, 298(5599):1698b–1699, 2002.
45. A. Kolb and N. Cuntz. Dynamic particle coupling for GPU-based fluid simulation. In *Proc. ASIM*, pages 722–727, 2005.
46. A. Kolb, L. Latta, and C. Rezk-Salama. Hardware-based simulation and collision detection for large particle systems. In *Proc. Graphics Hardware*, pages 123–131. ACM/Eurographics, 2004.
47. P. Koumoutsakos. Inviscid axisymmetrization of an elliptical vortex. *JOURNAL OF COMPUTATIONAL PHYSICS*, 138(2):821–857, Dec 1997.
48. P. Koumoutsakos. Multiscale flow simulations using particles. *ANNUAL REVIEW OF FLUID MECHANICS*, 37:457–487, 2005.
49. P. KOUMOUTSAKOS and A. LEONARD. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *JOURNAL OF FLUID MECHANICS*, 296:1–38, 1995.
50. P. Koumoutsakos, A. Leonard, and F. Pépin. Boundary conditions for viscous vortex methods. 113(1):52–61, 1994.
51. R. Krasny. A study of singularity formation in a vortex sheet by the point vortex approximation. *JFM*, 167:65–93, 1986.
52. A. Leonard. Review. vortex methods for flow simulation. 37:289–335, 1980.
53. R. J. LeVeque. High-resolution conservative algorithms for advection in incompressible flow. 33(2):627–665, 1996.
54. W. Liu, S. Jun, and S. Zhang. Reproducing kernel particle methods. 20(8–9):1081–1106, 1995.
55. F. Losasso, J. O. Talton, N. Kwatra, and R. Fedkiw. Two-way coupled sph and particle level set fluid simulation. *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS*, 14(4):797–804, Jul-Aug 2008.
56. L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *Astron. J.*, 82:1013–1024, 1977.
57. S. MALLAT and W. L. HWANG. Singularity detection and processing with wavelets. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 38(2):617–643, Mar 1992.
58. F. Milde, M. Bergdorf, and P. Koumoutsakos. A hybrid model for turmor induced angioegenesis. *Biophys J*, 2008.
59. K. Miller and R. N. Miller. Moving finite elements. I. *SIAM J. Numer. Anal.*, 18(6):1019–1032, 1981.
60. M. L. Minion and D. L. Brown. Performance of under-resolved two-dimensional incompressible flow simulations, ii. *Journal of Computational Physics*, 138:734–765, 1997.
61. R. Mittal and G. Iaccarino. Immersed boundary methods for viscous flow. 37:to appear, 2005.
62. J. J. Monaghan. Extrapolating B splines for interpolation. 60(2):253–262, 1985.
63. J. J. Monaghan. Smoothed particle hydrodynamics. *REPORTS ON PROGRESS IN PHYSICS*, 68(8):1703–1759, Aug 2005.
64. J. P. Morris. Simulating surface tension with smoothed particle hydrodynamics. 33(3):333–353, 2000.
65. S. D. Muller, I. Mezic, J. H. Walther, and P. Koumoutsakos. Transverse momentum micromixer optimization with evolution strategies. *COMPUTERS and FLUIDS*, 33(4):521–531, May 2004.
66. S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results.

169(2):463–502, 2001.
67. S. Osher and J. A. Sethian. Front propagating with curvature dependent speed: Algorithms based on Hamilton-Jacobi formulation. 79(1):12–49, 1988.
68. S. J. Osher and R. P. Fedkiw. Level set methods and dynamic implicit surfaces. *Springer Verlag*, 2002.
69. M. L. Ould-Salihi, G.-H. Cottet, and M. El Hamraoui. Blending finite-difference and vortex methods for incompressible flow computations. 22(5):1655–1674, 2000.
70. C. S. PESKIN. Numerical-analysis of blood-flow in heart. *JOURNAL OF COMPUTATIONAL PHYSICS*, 25(3):220–252, 1977.
71. H. Pfister and M. Gross. Point-based computer graphics. *IEEE COMPUTER GRAPHICS AND APPLICATIONS*, 24(4):22–23, Jul-Aug 2004.
72. P. Ploumhans, G. S. Winckelmans, J. K. Salmon, A. Leonard, and M. S. Warren. Vortex methods for direct numerical simulation of three-dimensional bluff body flows: Applications to the sphere at $Re = 300$, 500 and 1000. 178:427–463, 2002.
73. S. Premoze, T. Tasdizen, J. Bigler, A. Lefohn, and R. T. Whitaker. Particle-based simulation of fluids. *COMPUTER GRAPHICS FORUM*, 22(3):401–410, 2003.
74. W. T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. *Computer Graphics*, 17:359–376, 1983.
75. W. J. Rider and D. B. Kothe. Reconstructing volume tracking. 141:112–152, 1998.
76. L. Rosenhead. The spread of vorticity in the wake behind a cylinder. 127(A):590–612, 1930.
77. L. Rosenhead. The formation of vortices from a surface of discontinuity. 134:170–192, 1931.
78. I. F. Sbalzarini, A. Hayer, A. Helenius, and P. Koumoutsakos. Simulations of (an)isotropic diffusion on curved biological surfaces. *Biophys J*, 90(3):878–885, 2006.
79. R. Scardovelli and S. Zaleski. Direct numerical simulation of free-surface and interfacial flow. 31:567–603, 1999.
80. I. J. Schoenberg. Contribution to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math.*, 4:45–99, 112–141, 1946.
81. A. Selle, N. Rasmussen, and R. Fedkiw. A vortex particle method for smoke, water and explosions. *ACM Trans. Graph.*, 24(3):910–914, 2005.
82. J. Sethian. Numerical algorithms for propagating interfaces: Hamilton-Jacobi equations and conservations laws. 31:131–161, 1990.
83. J. A. Sethian. A fast marching level set method for monotonically advancing fronts. 93(4):1591–1595, 1996.
84. J. A. Sethian. Fast marching methods. *SIAM Rev.*, 41(2):199–235, 1999.
85. J. A. Sethian. Evolution, implementation, and application of level set and fast marching methods for advancing fronts. 169(2):503–555, 2001.
86. J. A. Sethian and P. Smereka. Level set methods for fluid interfaces. 35:341–372, 2003.
87. K. Sims. Particle animation and rendering using data parallel computation. *Computer Graphics (Siggraph '90 proceedings)*, pages 405–413, 1990.
88. A. R. Smith. Plants, fractals, and formal languages. *SIGGRAPH Comput. Graph.*, 18(3):1–10, 1984.
89. H. A. STONE. A simple derivation of the time-dependent convective-diffusion equation for surfactant transport along a deforming interface. *PHYSICS OF FLUIDS A-FLUID DYNAMICS*, 2(1):111–112, Jan 1990.
90. J. Strain. Fast adaptive 2D vortex methods. 132:108–122, 1997.
91. J. Strain. A fast semi-lagrangian contouring method for moving interfaces.

161(2):512–536, 2001.
92. M. Sussman and E. Fatemi. An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow. 20(4):1165–1191, 1999.
93. M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible 2-phase flow. 114(1):146–159, 1994.
94. C. Varea, J. L. Aragon, and R. A. Barrio. Turing patterns on a sphere. *PHYSICAL REVIEW E*, 60(4):4588–4592, Oct 1999.
95. O. V. Vasilyev. Solving multi-dimensional evolution problems with localized structures using second generation wavelets. *INTERNATIONAL JOURNAL OF COMPUTATIONAL FLUID DYNAMICS*, 17(2):151–168, Apr 2003.
96. L. Verlet. Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. 159(1):98–103, 1967.
97. J. H. Walther and P. Koumoutsakos. Three-dimensional particle methods for particle laden flows with two-way coupling. 167:39–71, 2001.
98. N. Zabusky, M. Hughes, and K. Roberts. Contour dynamics for the euler equations in two dimensions. 30:96–106, 1979.
99. S. T. Zalesak. Fully multidimensional flux-corrected transport algorithms for fluids. 31(3):335–362, 1979.