Duque Campayo, Daniel and Español, Pep (2020). An assignment procedure from particles to mesh that preserves field values. "International Journal of Computational Methods", v. 17 (n. 2); p. 1850130. https://doi.org/10.1142/S021987621850130X.

AN ASSIGNMENT PROCEDURE FROM PARTICLES TO MESH THAT PRESERVES FIELD VALUES

DANIEL DUQUE * PEP ESPAÑOL †

January 14, 2025

Abstract

In Computational Fluid Dynamics there have been many attempts to combine the advantages of having a fixed mesh, on which to carry out spatial calculations, with using particles moving according to the velocity field. These ideas in fact go back to Particle-in-Cell methods, proposed about 60 years ago. Of course, some procedure is needed to transfer field information between particles and mesh. There are many possible choices for this "assignment", or "projection". Several requirements may guide this choice. Two well-known ones are conservativity and stability, which apply to volume integrals of the fields. An additional one is here considered: preservation of information. This means that assignment from the particles onto the mesh and back should yield the same field values when the particles and the mesh coincide in position. The resulting method is termed "mass" assignment, due to its strong similarities with the Finite Element Method. Several procedures are tested, including the well-known FLIP, on three scenarios: simple 1D convection, 2D convection of Zalesak's disk, and a CFD simulation of the Taylor-Green periodic vortex sheet. Mass assignment is seen to be clearly superior to other methods.

1 Introduction

Historically, there have been two points of view to describe the dynamics of fluids. In the Eulerian picture, the dynamics of a fluid is described with respect to an external frame. In the Lagrangian picture the dynamics is described with respect to the flow. These two approaches are reflected in computational fluid dynamics (CFD), in which the spatial discretization may be carried out on a

*Model Basin Research Group (CEHINAV). ETSI Navales, Universidad Politécnica de Madrid, Avd. de la Memoria 4, Ciudad Universitaria - 28040 Madrid, Spain.

†Dpto. de Física Fundamental. Universidad Nacional de Educación a Distancia. Madrid, Spain.

fixed, Eulerian, mesh, or on a Lagrangian set of moving particles. Each approach has its advantages and drawbacks (Violeau (2012).)

It is therefore natural to try to combine the best of both approaches. This idea goes back to Particle-in-Cell (PIC) methods, introduced as early as the 1950s, see Evans et al. (1957); Amsden (1966). In this framework, the calculations related to spatial derivatives are carried out on an Eulerian (fixed) mesh. The original method was finite differences, but this was later generalized to others, such as the finite element method (FEM), or spectral methods. However, advection is treated by introducing an additional set of Lagrangian (moving) particles, which are explicitly moved following the velocity field. This motion accurately simulates the convection part of the equations. The procedure obviously requires the transfer of, at least, the velocity field onto the particles, in order to move them. Other fields may also be transferred, depending on the particular problem. Once the particles are moved, some of the fields carried by them (again, the velocity field at least) must be transferred back to the fixed mesh.

This very general and appealing idea of including particles to simulate advection is, at the same time, the main drawback of PIC methods. It is obvious that some procedure is needed to transfer the field information from the mesh to the particles. A harder problem is to transfer the information from the particles, which will be disordered after moving, back to the mesh. During the development of the PIC method, a number of procedures have been proposed, such as the Cloud-in-Cell method, described below. This procedure is often called "assignment" — here, the term "projection" will be used as a synonym, as is customary in this field. However, the latter could lead to confusion with other procedures, such as the Galerkin projection (see e.g. Rapún and Vega (2010)), in which a continuous set of equations is projected on a functional space (even if, as we show below, some similarities may exist between the Galerkin procedure and our "mass" assignment).

The fluid-implicit-particle method (FLIP) (Refs. Brackbill and Ruppel (1986); Brackbill et al. (1988)) is also a descendant of the PIC. Among its features, we will discuss below its assignment procedure, and the idea of transferring only changes in fields, not their whole values. The latter idea, which is trivial to implement, yields very good results when the motion is dominated by convection. This is a key feature for its current popularity in the computer graphics community (see e.g. Bridson (2015)).

Our main task is to study the performance of different projection techniques, under the guidance of several requirements. One of them is conservativity: the total volume integral of a field must not vary upon projection. Another is stability: the integral of the square of any field should decrease upon projection. These two requirements are well known, while here an additional one is considered: preservation of information. This means that the field values at discrete points do not vary under reconstruction (extension to the continuum) followed by projection. This is enforced both for reconstruction from the mesh followed by projection back onto the mesh, and for the equivalent procedure from the particles. Another view of this property is that, if the particles' positions coin-

cide with the mesh nodes, then for assignment from the particles onto the mesh, and back, will leave the field values invariant.

The article is organized as follows. In Section 2 different assignment procedures are discussed, together with the requirements. These procedures are then tested in three scenarios in Section 3: a very simple 1D convection of a step function, Zalesak's disk 2D test, and a CFD simulation of the Taylor-Green periodic vortex sheet (which is a solution of the Navier-Stokes equations). Some finishing remarks are given in Section 4.

2 Assignment procedures

In Subsection 2.1 a general framework for assignment through functional sets is provided. The requirements of conservativity and stability will be discussed in Subsection 2.2. The simplest assignment is introduced in 2.3. The FLIP procedure is considered in Subsection 2.4. In Subsection 2.5 the additional requirement of preservation of information is introduced, thus leading to mass assignment.

2.1 Assignment functions

A set of functions $\{\psi_\mu\}$ will be used to *reconstruct* a field $A(\mathbf{r})$ from its values at particles A_μ:

$$A(\mathbf{r}) \doteq \sum_\mu A_\mu \psi_\mu(\mathbf{r}). \tag{1}$$

The functions are supposed to comply with partition of unity, since a particle distribution with constant values should yield a constant field:

$$\sum_\mu \psi_\mu(\mathbf{r}) = 1. \tag{2}$$

In this work, only the simple linear Finite Element basis functions (FEs) will be considered, even though other choices are of course possible. These are piecewise affine functions, which in 1D are equal to 1 at each particle, then linearly decrease to 0 at the two neighboring nodes. In 2D they are pyramids of height 1 at each particle, decreasing to 0 at the neighboring nodes. The Delaunay triangulation is used in order to precisely define neighbors.

For the mesh the same symbols will be kept, but the sub-indices will be Latin letters. A field is reconstructed from the values at mesh nodes $\overline{A}_i$ by mesh functions $\{\psi_i\}$:

$$\overline{A}(\mathbf{r}) \doteq \sum_i \overline{A}_i \psi_i(\mathbf{r}). \tag{3}$$

The particle-to-mesh assignment procedure consists in finding nodal values for $\overline{A}_i$ given particle values A_μ. The inverse mesh-to-particle yields particle $\overline{\overline{A}}_\mu$ given mesh $\overline{A}_i$.

2.2 Conservativity and stability

A property that may lead us on our research of possible assignment methods is conservativity, in the sense of Cottet and Koumoutsakos (2000): the integral of a field not changing on projection. Integrating field A of Eq. (1),

$$\int A(\mathbf{r})d\mathbf{r} = \sum_{\mu} A_\mu v_\mu, \tag{4}$$

where particle volumes are given by

$$v_\mu := \int \psi_\mu(\mathbf{r})d\mathbf{r}. \tag{5}$$

Equivalently, mesh volumes may be defined from Eq. (3),

$$v_i := \int \psi_i(\mathbf{r})d\mathbf{r}. \tag{6}$$

(As shown below, the FLIP method does *not* use this definition of the mesh volume.)

Whichever the particular definition of the volumes, conservativity is expressed as:

$$\sum_i \overline{A}_i v_i = \sum_\mu A_\mu v_\mu. \tag{7}$$

When applied to the components of the linear momentum, this would guarantee the global conservation of linear momentum. For a density field, this would guarantee conservation of mass. It therefore would seem as a vital requirement. However, it will be seen that methods which do not comply with this condition may still deviate little from it in practice.

Of course, the same requirement could be asked when projecting from the mesh onto the particles:

$$\sum_\mu \overline{\overline{A}}_\mu v_\mu = \sum_i \overline{A}_i v_i. \tag{8}$$

Another requirement is stability: for any energy-like expression defined on the particles and on the mesh:

$$E_\mathrm{p} := \sum_\mu v_\mu A_\mu^2 \qquad E_\mathrm{m} := \sum_i v_i \overline{A}_i^2, \tag{9}$$

we require

$$E_\mathrm{m} \leq E_\mathrm{p}. \tag{10}$$

This guarantees that there is no overshoot in e.g. the kinetic energy upon assignment, i.e. that the assignment process is disipative. For a general field this enforces a diminishing second momentum, which prevents overshooting in a global sense. Of course, the same could be required when going from the mesh to the particles.

method	part $\to$ mesh	mesh $\to$ part
δ	$\overline{A}_i = \sum_\mu A_\mu \psi_\mu(\mathbf{r}_i)$ (11)	$\overline{\overline{A}}_\mu = \overline{\overline{A}}(\mathbf{r}_i) = \sum_i \overline{A}_i \psi_i(\mathbf{r}_\mu)$ (12)
FLIP	$\overline{A}_i := \sum_\mu A_\mu v_\mu \psi_i(\mathbf{r}_\mu) / \sum_\mu v_\mu \psi_i(\mathbf{r}_\mu)$ (13,14)	
mass-δ	$\overline{A}_i := \sum_j m_{ij}^{-1} \int d\mathbf{r} A(\mathbf{r}) \phi_j(\mathbf{r})$ (15, 22)	
full mass	$\overline{A}_i := \sum_j m_{ij}^{-1} \int d\mathbf{r} A(\mathbf{r}) \psi_j(\mathbf{r})$ (15, 22)	$\overline{\overline{A}}_\mu := \sum_\nu m_{\mu\nu}^{-1} \int d\mathbf{r} \overline{A}(\mathbf{r}) \psi_\nu(\mathbf{r})$ (24)
mass - lumped	$\overline{A}_i := \sum_j m_{ij}^{-1} \int d\mathbf{r} A(\mathbf{r}) \psi_j(\mathbf{r})$ (15, 22)	$\overline{\overline{A}}_\mu := \frac{1}{v_\mu} \int d\mathbf{r} \overline{A}(\mathbf{r}) \psi_\mu(\mathbf{r})$ (19)
lumped	$\overline{A}_i := \frac{1}{v_i} \int d\mathbf{r} A(\mathbf{r}) \psi_i(\mathbf{r})$ (15, 19)	

Table 1: Features of the methods considered (left column), the procedure by which fields are projected from particles to mesh (middle column), and the reverse procedure (right column). References are given to relevant equations in the text. The last two methods have been tested, but results are not given here, for the sake of brevity.

2.3 δ assignment

The simplest assignment would be to define mesh values as the local values reconstructed from the particles:

$$\overline{A}_i = \overline{A}(\mathbf{r}_i) = \sum_\mu A_\mu \psi_\mu(\mathbf{r}_i). \tag{11}$$

Also,

$$\overline{\overline{A}}_\mu = \overline{\overline{A}}(\mathbf{r}_\mu) = \sum_i \overline{A}_i \psi_i(\mathbf{r}_\mu). \tag{12}$$

Particle volume may simply be defined as in (5) and (6). It is easy to check that this procedure does not satisfy either conservativity or stability. It is, nevertheless, very simple to implement, and was the method used in our previous study, Duque and Español (2018). In Table 1 the relevant expressions for this method is shown, along others for methods that will be considered next.

2.4 FLIP assignment

This procedure starts from the expression

$$\overline{A}_i := \frac{1}{v_i} \sum_\mu A_\mu v_\mu \psi_i(\mathbf{r}_\mu). \tag{13}$$

The particle volumes may be defined as in (5). If (6) is used for the mesh volumes, the PIC expression is recovered (see Refs. Evans et al. (1957); Amsden (1966)).

The FLIP procedure proposes the alternative mesh volume:

$$v_i := \sum_\mu v_\mu \psi_i(\mathbf{r}_\mu). \tag{14}$$

This procedure can be shown to satisfy both conservativity and stability.

A later projection onto the particles is exactly as in the δ-assignment, Eq. (12), and this can be seen to again satisfy conservativity and stability.

This procedure may seem to be a great improvement since a simple change in the mesh volume restores conservativity and stability. It is also convenient to code, since the same mesh functional set, $\{\psi_i\}$, is used for both particle to mesh assignment and its reverse. However, it may seem somewhat strange that the volume of a nodal mesh (14) does not depend on the mesh itself, but on the particles around it. A node may even have a vanishing volume if no particles are close to it and the $\{\psi_\mu\}$ have compact support. Also, the two assignments, (12) and (13) are clearly unsymmetrical. A summary of the method is given in Table 1.

2.5 Mass assignment

Let us consider the assignment procedure

$$\overline{A}_i := \int d\mathbf{r} A(\mathbf{r}) \phi_i(\mathbf{r}). \tag{15}$$

Here, the assignment functional set $\{\phi_i(\mathbf{r})\}$ consists of normalized functions:

$$\int d\mathbf{r} \phi_i(\mathbf{r}) = 1, \tag{16}$$

so that a constant field yields constant mesh values. Notice the $\{\psi_\mu\}$ functions carry no physical units, but the $\{\phi_\mu\}$ have units of length^{-d}, where d is the spatial dimension. The δ method is recovered as a special case if Dirac δ functions are used, $\phi_i(\mathbf{r}) = \delta(\mathbf{r} - \mathbf{r}_i)$, which in retrospect explains its name.

For conservativity, let us evaluate

$$\sum_i \overline{A}_i v_i = \sum_i \left(\int d\mathbf{r} A(\mathbf{r}) \right) \phi_i(\mathbf{r}) v_i = \int d\mathbf{r} A(\mathbf{r}) \left(\sum_i \phi_i(\mathbf{r}) v_i \right). \tag{17}$$

If the last parenthesis was equal to 1, conservativity would apply (with particle volumes as in (5). Therefore, these functions must satisfy

$$\sum_i v_i \phi_i(\mathbf{r}) = 1. \tag{18}$$

If this condition holds, it is straightforward to proof that stability is also satisfied.

The simplest way to define these functions would be as normalized versions of the $\{\psi_i\}$:

$$\phi_i(\mathbf{r}) = \frac{1}{v_i}\psi_i(\mathbf{r}) \tag{19}$$

This would correspond to a "lumped mass" method, a term that will become clear very soon.

Let us however consider $\{\phi_i\}$ that "preserves nodal information". This means that a reconstruction procedure, followed by projection, should leave the nodal values invariant:

$$\overline{A}_i := \int d\mathbf{r}\phi_i(\mathbf{r}) \left(\sum_j \overline{A}_j \psi_j(\mathbf{r}) \right). \tag{20}$$

(See also Cottet and Koumoutsakos (2000), p. 243, for an application of this idea to an iterative method.) Notice these two operations are carried out on the mesh only (or, on particles only). It is, in general, impossible to satisfy this requirement for a projection from the mesh onto the particles, back onto the mesh (or its equivalent, starting from the particles.) Alternatively, the two operations may be thought of as taking place for the spacial situation in which the particles' position coincide with the mesh nodes. This is indeed a common choice for the initial configuration of a simulation, and also for methods in which particles are re-created at each time step.

Eq. (20) means

$$\int d\mathbf{r}\phi_i(\mathbf{r})\psi_j(\mathbf{r}) = \delta_{ij}, \tag{21}$$

where the latter δ is Kronecker's. This will be called the "preservation property". If the $\{\phi_i\}$ are linear combinations of the $\{\psi_i\}$ set, it is simple to show that

$$\phi_i(\mathbf{r}) = \sum_j m_{ij}^{-1}\psi_j(\mathbf{r}), \tag{22}$$

where the inverse of the mass matrix appears, the latter defined as having elements

$$m_{ij} =: \int d\mathbf{r}\psi_i(\mathbf{r})\psi_j(\mathbf{r}). \tag{23}$$

It can be shown that the functions defined by (22) do comply with requirements (16) and (18). As a consequence, the resulting procedure will be conservative and stable.

For the sake of symmetry, a similar procedure is employed for projection onto the particles:

$$\overline{\overline{A}}_\mu := \int d\mathbf{r}\overline{\overline{A}}(\mathbf{r})\phi_\mu(\mathbf{r}). \tag{24}$$

The resulting procedure can also be shown to satisfy both conservativity and stability.

Some remarks are in order.

- The integration needed in (15) may be cumbersome to carry out. A simplified quadrature rule that involves a quadratic reconstruction for function $\overline{A}(\mathbf{r})$ is therefore employed, as explained below in B .
- This procedure requires matrix inversion. This is not such a problem for the mesh, since in a typical CFD computation these matrices must be assembled and inverted on the mesh anyway. On the particles however, such a calculation is often not needed, whereas it certainly is within the present procedure.
- The simple (19) would correspond to a lumped mass approximation in the language of the FEM. Indeed, $\sum_j m_{ij} = v_i$.
- There is an appealing correspondence with the usual FEM, as explained next.

To demonstrate the correspondence with the FEM, let us begin with a simple diffusion equation:

$$g - \nabla^2 A = 0, \tag{25}$$

which may be projected onto a nodal functional space (it is immaterial for this discussion whether the following occurs on the particles, or on the mesh):

$$\sum_i g_i \psi_i(\mathbf{r}) - \sum_i A_i \nabla^2 \psi_i(\mathbf{r}) \approx 0. \tag{26}$$

The equality is no longer satisfied in general, but the residual may be required to be orthogonal to all the shape functions (as in the method of weighted residuals, as explained e.g. in Reddy (2005)):

$$\int d\mathbf{r} \left(\sum_i g_i \psi_i(\mathbf{r}) - \sum_i A_i \nabla^2 \psi_i(\mathbf{r}) \right) \psi_j(\mathbf{r}) = 0 \qquad \forall j. \tag{27}$$

This results in the expression

$$\sum_i m_{ij} g_i = \int d\mathbf{r} \psi_j(\mathbf{r}) \nabla^2 \left(\sum_i A_i \psi_i(\mathbf{r}) \right). \tag{28}$$

Now, the inverse of the mass matrix may be applied, and recalling the definition (22) we finally obtain

$$g_j = \int d\mathbf{r} \phi_j(\mathbf{r}) \nabla^2 \left(\sum_i A_i \psi_i(\mathbf{r}) \right) = \int d\mathbf{r} \phi_j(\mathbf{r}) \nabla^2 \overline{A}(\mathbf{r}) \tag{29}$$

This is an expression for the second derivative that entails: the reconstruction from the nodal values A_i to a function $\overline{A}(\mathbf{r})$, deriving this function twice, then projecting back onto the nodes. A classical FEM approach is therefore seen to yield an expression that is consistent with our mass assignment. Indeed, if the

method to solve the equations of motion is of the FEM type, the calculations are likely already implemented in the code (at least, for the mesh).

Two instances of mass projection will be considered. In the first one, mass projection will be used from the particles to the mesh, but simple δ projection will be used when projecting back. This is because in a class of simulations called "Particle FEM" (or "pFEM", see e.g. Refs Oñate et al. (2004); Idelsohn et al. (2004, 2013)) the matrices necessary for the former projection will be already computed, at the start of the simulation (they will not change since the mesh is fixed). If performance was already good, the numerical implementation of this projection will therefore not imply a large increase in computational resources. We will call this method "mass-δ", for obvious reasons. The δ projection is not conservative nor stable, but we will see below that departs little from these requirements.

Our second choice will be "full mass" projection. This is the most symmetric mass assignment method, which will comply with conservativity and stability. It clearly requires the particle mass matrix must be calculated, and inverted, at each time step. We will see that this additional computational burden may be compensated by its superior performance.

Another possible candidate would be a "mass-lumped" method, where only the particle volumes are needed. Of course, a purely "lumped" method, bothways, can be considered. These two lumped procedures are listed at the end of Table 1, but the results are not shown here, since in all cases they are very similar to the mass-δ results. They do satisfy conservativity and stability.

3 Numerical experiments

The procedures discussed above are tested in three scenarios in this Section. The first one, in Subsection 3.1, is a very simple 1D convection of a step function. This may be considered a 1D version of Zalesak's disk 2D test, which is considered in Subsection 3.2. Finally, a CFD simulation of the Taylor-Green periodic vortex sheet, a solution of the Navier-Stokes equations, is given in Subsection 3.3.

3.1 Advected step function in one dimension

On the $[0, 1)$ segment, let us consider a region with a color field A that has a value of 1 for $x \in (0.25, 0.75)$ and 0 otherwise. The field is simply advected:

$$\frac{dA}{dt} = 0 \qquad \frac{dx}{dt} = u. \tag{30}$$

In our case, $u = 1$, simply a constant positive velocity (toward the right).

Since the field is just advected, it makes little sense to project from and onto the mesh at every time-step: with no projection, the shape translates to the right. Nevertheless, the A field is projected from the mesh to the particles and vice versa at every time step, in order to benchmark our method. We employ

200 particles and 200 mesh nodes, with a time step given by a Courant number Co:= $u(\Delta t)/(\Delta x) = 0.1$. Periodic boundary conditions are imposed at the ends.

In the following, we will restrict our attention to the four procedures highlighted in Table 1. The fist one is the δ projection, which is the simplest one. We will also consider the FLIP method. To be precise, we are only evaluating the FLIP volume assignment (14), the FLIP idea of projecting only changes in fields at each time step is not applied here on purpose, since we want to assess assignment effects (otherwise, the field A would not change, there being no source term to it.) Two different mass methods are evaluated: mass-δ and full mass.

For the mesh, and also for the particles in the mass methods, we use simple linear FE functions. In the context of vortex methods, these procedure is termed "area-weighting scheme", and also "Cloud-in-Cell", Christiansen (1973).

In Figure 1 we show the final profiles at time $T = 1$, at which the particles have traversed the system and come back to their initial positions. At the left results are for a regular particle set up, while at the right particles are disturbed $\pm 40\%(\Delta x)$ about their initial positions. Notice that in the former case the FLIP method is equivalent to the δ one.

The full mass method is seen to be the only one to preserve the plateaus at 0 and 1, while the other methods spread out the function that was initially sharp. This comes at the cost of undershoots below 0, and overshoots above 1, close to the interface, resembling Gibbs phenomena. This may be understood by the fact that the assignment functions are not positive, as is obvious from the requirement in Eq. (21). Indeed, for $j = i + 1$ the equation reads, for FEs in 1D:

$$\int_{x_i}^{x_{i+1}} dx \phi_i(x) \psi_{i+1}(x) = 0, \tag{31}$$

but since $\psi_{i+1}(x)$ is positive, it follows that $\phi_i(x)$ is not.

In table 2 we provide measures of the accuracy of the final profiles. First of all, we evaluate the relative change in the integral of A:

$$E_1 := \sqrt{\frac{\sum_\mu \left((v_\mu A_\mu)(T=1) - (v_\mu A_\mu)(T=0)\right)}{\sum_\mu (v_\mu A_\mu)(T=0)}} \tag{32}$$

This quantity should be null for methods that comply with (7). Indeed we see this is satisfied to machine precision by all methods for a regular particle arrangement. However, small differences occur, as expected, when particles are distorted, except for the FLIP method. Recall that the full mass method, while in principle compliant with (7), is implemented with an approximate quadrature which will result in departures from this property, see Appendix B.

We also check the energy-like second moment of the profile, which according to (10) should always be a number that decreases with each iteration, although of course, a small decrease is desirable. We have checked it does, and measure

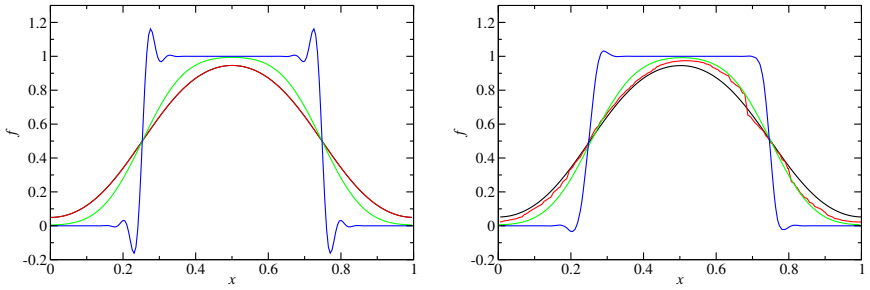

Figure 1: Results for the advected step function. Field after one traversal of the simulation cell. Left: regular particle distribution, right: distorted. Black: δ projection, red: FLIP, green: mass-δ, blue: full mass.

method	E_1	E_2	L_2
δ	0	-29%	35%
FLIP	0	-25%	35%
mass-δ	0	-21%	29%
full mass	0	0.85%	10%

method	E_1	E_2	L_2
δ	-0.24%	-29%	35%
FLIP	0	-25%	35%
mass-δ	0.7%	-20%	29%
full mass	-0.7%	-4%	12%

Table 2: Results for 1D spacing. Left table: regular, right: distorted

its decrease by its relative value at the last time step.

$$E_2 := \sqrt{\frac{\sum_\mu \left((v_\mu A_\mu^2)(T=1) - (v_\mu A_\mu^2)(T=0)\right)}{\sum_\mu (v_\mu A_\mu^2)(T=0)}} \tag{33}$$

The full mass procedure is seen to produce a less distorted final profile, as evident also in Figure 1. (The small, positive value of the full mass procedure for regular spacing can again be traced back to the approximate quadrature.)

Finally, we measure the L_2 relative distance between the final profile and the initial one:

$$L_2 := \sqrt{\frac{\sum_\mu \left((v_\mu A_\mu)(T=1) - (v_\mu A_\mu)(T=0)\right)^2}{(v_\mu A_\mu)(T=0)}}, \tag{34}$$

again confirming that the full mass projection is more accurate.

3.2 Zalesak's disk

Let us consider a region with a color field α that has a value of 1 for points inside a domain and 0 for points outside and which is simply advected:

$$\frac{d\alpha}{dt} = 0 \qquad \frac{d\mathbf{r}}{dt} = \mathbf{u}. \tag{35}$$

The domain is a circle with a slot. The circle's radius is given a value of $R = 0.5$, while the slot was a width of $1/6$, and a height of $5/6$. The simulation

box is a $(-1.5, 1.5) \times (-1.5, 1.5)$ square, and the number of nodes is set to 90×90, so that the mesh spacing is $H = 3/90 = 1/30$, these are the same value as in Idelsohn et al. (2015). The time step is $\Delta t = 0.01$, which corresponds to $\mathrm{Co}_H := u(\Delta t)/H \approx 0.94$ for nodes on the rim of the disk.

The velocity field is a pure rotation:

$$u_x = -\omega y \tag{36}$$

$$u_y = \omega x, \tag{37}$$

where $\omega = 2\pi/\tau$, and the period of rotation is set to $\tau = 1$. Periodic boundary conditions are used in this simulation, but this fact is not really important since the only region that is actually moved is inside a circle of radius 1.4, within a larger simulation box, of size 3×3.

For the mesh, and also for the particles in the mass methods, we use linear FE functions, as in the previous 1D example. For all mass methods, a moving Delaunay triangulation must be maintained for the projection from the mesh to the particles. In addition to that, the calculation of the corresponding mass matrix, and its inversion, is needed for the full mass method.

Again, it would make little sense to project from and onto the mesh at every time-step, except for benchmarking purposes. As in the 1D case, the FLIP idea of projecting only increments is not applied. Results are given in Figure 2, showing contour plots for values between 0.49 and 0.51 of the α field on the mesh nodes. We include the initial contour, the contour after one revolution, $T = \tau$, and after two revolutions, $T = 2\tau$.

On the top left contours are shown for the δ procedure, and at the top right, for the FLIP procedure. Both are seen to greatly smear out the initial slab. At the bottom left, the full-δ procedure is shown. This time, some remains of the slab are seen after one revolution. Finally, the full mass projection (at the bottom right) produces quite good profiles even after two revolutions.

As in 1D, the good results of the full mass method maintaining the plateaus are accompanied by undershoots below 0 and overshoots above 1. In Figure 3 we show more detailed contours for the full mass procedure. The isocontour for $\alpha = 0.5$ is shown again, but the $\alpha = 0$ contour reveals a corona of slightly negative values, as low as -0.05 approximately. Values of α about 1.05 are also seen in the inner regions.

We again employ the same error measures for our profiles. For the relative L_2 distance we introduce a refinement, by comparing the final profile and the one for a simulation in which the particle field is simply advected. This way we subtract out the errors due to time integration, by which particles' trajectories are not exactly circles. These errors are quite small anyway.

3.3 Taylor-Green vortices

The Taylor-Green vortex sheet is an analytic solution to the Navier-Stokes equations for an incompressible Newtonian fluid:

$$\frac{d\mathbf{u}}{dt} = -\nabla p + \nu \nabla^2 \mathbf{u}. \tag{38}$$

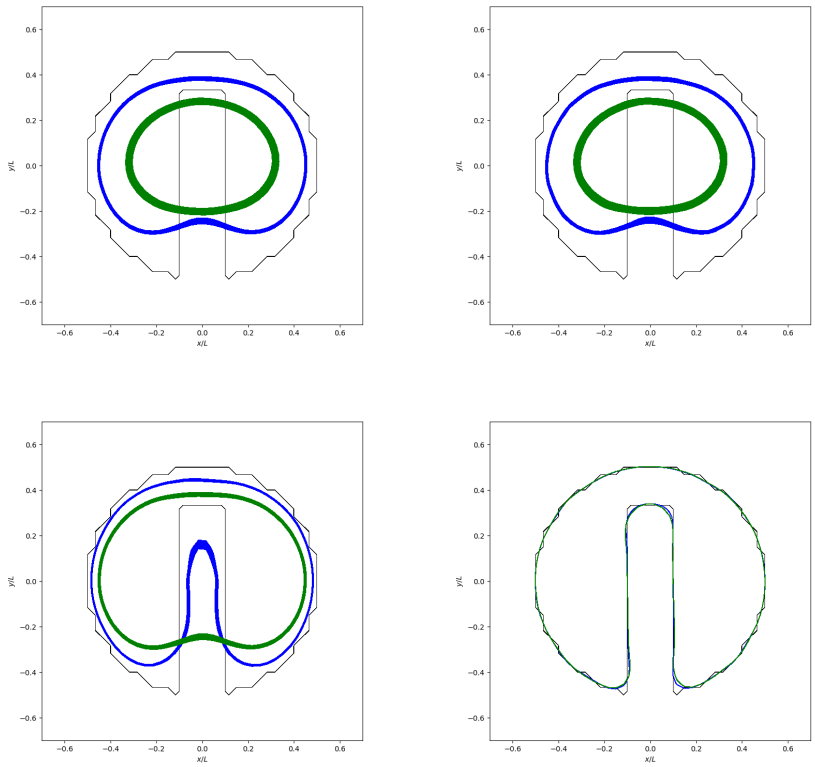

Figure 2: Results for the rotation of Zalesak's disk. Isocontours for $\alpha \in (0.49, 0.51)$. Initial field in black, after one rotation in blue, after two in green. Top left: δ method. Top right: FLIP. Bottom left: mass-δ. Bottom right: full mass.

method	E_1	E_2	L_2
δ	-0.5%	-29%	67%
FLIP	0.14%	-61%	67%
mass-δ	-0.5	-50%	61%
full mass	0.5%	-9%	30%

Table 3: Results for the rotation of Zalesak's disk.

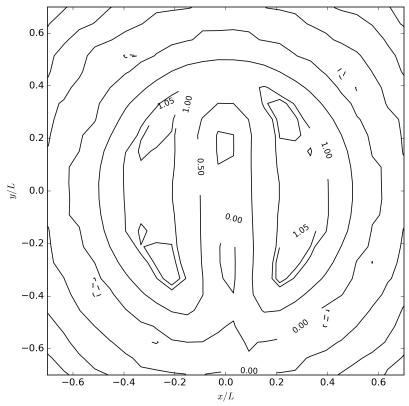

Figure 3: Results for the rotation of Zalesak's disk using the full mass method. Isocontours for $\alpha \in (-0.05, 0, 0.5, 1, 1.05)$ after two rotations.

The solution, with a periodic length of L, is the velocity field

$$\mathbf{u}_x = f(t)\sin(kx)\cos(ky) \tag{39}$$

$$\mathbf{u}_y = -f(t)\cos(kx)\sin(ky), \tag{40}$$

where $k = 2\pi/L$, and the time-dependent prefactor function is given by

$$f(t) = u_0 \exp\left(-8\pi^2 t^*/\mathrm{Re}\right). \tag{41}$$

The Reynolds number is defined as Re:= $u_0 L/\nu$, and the dimensionless time is $t^* := tu_0/L$. We set $u_0 = 1$, $L = 1$, and $\nu = 0.005$, thus setting a Reynolds number of Re=200.

For the numerical solution of the Navier-Stokes equation, a standard splitting approach is used, see Codina (2001). The procedure is a simple mid-point time integration, as in Duque and Español (2018). All space derivatives are calculated on the mesh. As explained there, and in Idelsohn et al. (2015), even this integrator, with Δt^2 accuracy, will result in a Δt overall accuracy. This is because the projection procedure causes a Δt bottleneck in the calculation. A possible remedy, proposed by Duque et al. (2017), is to include higher order basis functions. This idea is completely compatible with the different assignments discussed, but for the sake of simplicity we will not apply it here.

The main difference with the method in Duque and Español (2018) is that the FLIP incremental scheme (projecting only increments in fields at each time step) is now applied to all simulation, both in FLIP assignment (of course), and the others.

In order to quantify the accuracy of the different methods, the relative L_2 distance between the velocity field obtained by simulation and its exact value is

computed:

$$L_2 := \sqrt{\frac{\sum_{i=1}^{N} V_i \left|\mathbf{u}(\mathbf{r}_i) - \mathbf{u}_i\right|^2}{\sum_{i=1}^{N} V_i |\mathbf{u}(\mathbf{r}_i)|^2}}, \tag{42}$$

where $\mathbf{u}(\mathbf{r}_i)$ is the exact velocity field as in (39), evaluated on particle i position. The same measure may be evaluated for mesh nodes, with very similar results.The other field which could be monitored is the pressure field. This field is not given an initial value, since it is determined, at each half step, from the incompressibility constraint on the guessed velocity field. To be precise, it is the solution of the pressure Poison equation, which is solved at the mesh nodes. Results are not given since they convey the same information as those for the velocity field.

This error is expected to start at a very low value and increase approximately linearly as the simulation proceeds. In order to compare between methods, in Fig. 4 the value of this error at $T^* = 1$ is plotted. At this time, $f(T) = \exp(-8\pi^2/200) = 0.67$, so that the velocity field should have decreased to about 67% of its initial value.

The error for the velocity field is seen to decrease with Δt. The interparticle and mesh spacing decreases as Δt does, in order to fix a Courant number of $\mathrm{Co}_H = 0.5$ (the number of nodes and particles therefore increases quadratically with $1/\Delta t$). Like in Zalesak's disk test, there are as many particles as nodes. The particles are created at the beginning of the simulation and moved according to the velocity field. As expected, the order of convergence of the error roughly agrees with a Δt^1 power in all cases. Results from the full mass procedure are much more accurate. However, at finer resolutions the latter results cross over to a power law with an exponent lower than 1, for reasons that are not clear at present.

Despite its superior performance, the full mass is clearly more costly computationally than the alternatives. It therefore seems interesting to plot the error as a function of CPU time. A bad scaling as the number of nodes and particles is increased would make this procedure less appealing. However, in Fig. 5 the full mass procedure is seen to scale similarly to other procedures, with L_2 roughly proportional to $\Delta t^{1/4}$, while being about ten times more accurate for the same computational cost. Similarly to the previous Figure, the full mass results cross over to a power law with an exponent lower than 1/4.

The CPU run times clearly depend on the machine used, but a faster one would likely accelerate all simulations by a similar factor. This would result in a horizontal translation of all the curves in a logarithmic scale. Results do depend on the particular linear algebra algorithm used, details can be found in Appendix A.

One may therefore conclude from these observations that the higher computational cost of a full mass method will be compensated by its higher accuracy.

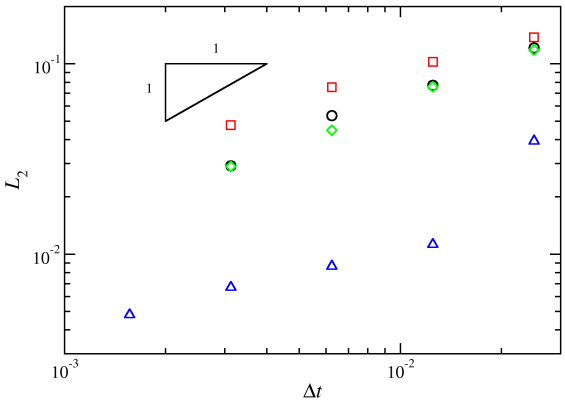

Figure 4: L_2 error of the velocity field at time $T^* = 1$, versus time step Δt. Black circles: δ method, red squares: FLIP, green diamonds: mass-δ, blue triangles: full mass. The triangle shows a Δt^1 power-law.

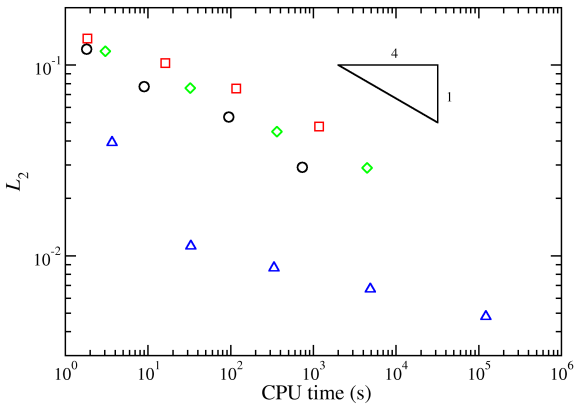

Figure 5: L_2 error of the velocity field at time $T^* = 1$, versus CPU time in seconds Black circles: δ method, red squares: FLIP, green diamonds: mass-δ, blue triangles: full mass. The triangle shows a Δt^1 power-law.

4 Conclusions

We have described several assignment, or projection, procedures by which field information may be transferred between particles and mesh. We have focused on four procedures: the simplest δ method, the FLIP method, the mass-δ method and the full mass method.

Each method is tested against several requirements: conservativity, which makes sure that the total integral of a field is not changed; stability, that ensures that the integral of the square of a field decreases upon projection; and preservation of information, which means that the assignment procedure carried twice leaves the field values invariant if the particles and the mesh have the same positions.

We have tested the method in 1D and 2D advection problems. Conservativity is satisfied exactly by construction in the FLIP method, and rather well satisfied in the other cases. Stability is satisfied in all four methods, but the full mass method is seen to be superior, as it produces a smaller decrease of the values. On the other hand, it also leads to higher over- and undershoots at sharp interfaces.

Finally, the full mass method is clearly superior in our CFD simulations of the Taylor-Green vortex sheet, where an approximate ten-fold increase in accuracy is achieved for the same simulation clock time. The puzzling moderation of the convergence rates at higher resolutions is under study. Preliminary results show that re-creating the particles at every time step (as opposed to keeping them from the beginning of the simulation) eliminates this effect, very likely due to the additional control of particle disorder that is achieved this way.

We would like to stress the fact that the methods presented here are linear in resolution. This means that the L_2 error should decrease as Δx, as indeed does. This slow convergence means that the time integration needs only be linear in time, since a constant Courant number forces Δx and Δt to be proportional. Results with the quadratic basis functions (not shown here) described in Duque et al. (2017) should converge as $L2 \sim \Delta x^2$, but in this case the time integrator must be careful implemented so not to spoil the convergence rate (in particular, a predictor-corrector loop for the mid-step velocity has been confirmed as a valid candidate.) When coupled with particle re-creation, the rates seem to be $L_2 \sim \Delta t^2$, as expected. The procedure is quite costly computationally, but worth it in terms of accuracy versus CPU time: its performance yields a much faster rate of $L_2 \sim \mathrm{CPU}^{-0.6}$.

The final conclusion is that the full mass method should be seriously considered as an assignment procedure, despite its inherent higher computational complexity.

As a future work, the treatment of boundary conditions should be addressed. Here, these are neglected by assuming periodicity, but any simple tests, such as the lid-driven cavity flow, involve boundaries. Boundary conditions in purely Lagrangian simulations are known to be problematic. Within a PIC approach, however, these problems are alleviated, since the conditions are usually imposed of the mesh. For instance, a Dirichlet boundary condition on the velocity field

would be naturally translated on the particles at each time-step. There could be problems with particles leaving the simulation cell, either at outlets, which is to be expected, or across walls, which would of course be an artifact. These particles should then be replaced or brought back to the cell. The simplest way to deal with this potential problems would be, again particle re-creation at each time step, instead of keeping the particles from one time step to the next.

Acknowledgments

We wish to thank Prof. David Le Touzé for his suggestions regarding the FLIP method. We also thank Prof. Michael Schick for hosting a research stay, during which this work was finished. The research leading to these results has received funding from the Ministerio de Economía y Competitividad of Spain (MINECO) under grants TRA2013-41096-P "Optimización del transporte de gas licuado en buques LNG mediante estudios sobre interacción fluido-estructura" and FIS2013-47350-C5-3-R "Modelización de la Materia Blanda en Múltiples escalas".

References

Damien Violeau. *Fluid Mechanics and the SPH method: theory and applications.* Oxford University Press, 2012.

Martha W Evans, Francis H Harlow, and Eleazer Bromberg. The particle-in-cell method for hydrodynamic calculations. Technical Report LA-2139, Los Alamos Scientific Laboratory, 1957.

A.A. Amsden. The particle-in-cell method for the calculation of the dynamics of compressible fluids. Technical Report LA-3466, Los Alamos Scientific Laboratory, 1966.

María-Luisa Rapún and José M Vega. Reduced order models based on local POD plus Galerkin projection. *Journal of Computational Physics*, 229(8): 3046–3063, 2010.

JU Brackbill and HM Ruppel. FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions. *Journal of Computational Physics*, 65(2):314–343, 1986.

Jeremiah U Brackbill, Douglas B Kothe, and Hans M Ruppel. FLIP: A low-dissipation, particle-in-cell method for fluid flow. *Computer Physics Communications*, 48(1):25–38, 1988.

Robert Bridson. *Fluid simulation for computer graphics.* CRC Press, 2015.

Georges-Henri Cottet and Petros D Koumoutsakos. *Vortex methods: theory and practice.* Cambridge university press, 2000.

Daniel Duque and P Español. Rapid convergence for simulations that project from particles onto a fixed mesh. *International Journal for Numerical Methods in Engineering*, 2018. Under review.

J Reddy. *An Introduction to the Finite Element Method*. McGraw-Hill Series in Mechanical Engineering. McGraw-Hill Education, 2005.

E Oñate, S R Idelsohn, F del Pin, and R Aubry. The particle finite element method - an overview. *International Journal of Computational Methods*, 1 (2):267–307, 2004.

S.R. Idelsohn, E. Oñate, and F. Del Pin. The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *International Journal for Numerical Methods in Engineering*, 61(7): 964–989, 2004.

Sergio Rodolfo Idelsohn, Norberto Marcelo Nigro, Juan Marcelo Gimenez, Riccardo Rossi, and Julio Marcelo Marti. A fast and accurate method to solve the incompressible navier-stokes equations. *Engineering Computations*, 30(2): 197–222, 2013. doi: 10.1108/02644401311304854.

IP Christiansen. Numerical simulation of hydrodynamics by the method of point vortices. *Journal of Computational Physics*, 13(3):363–379, 1973.

Sergio Idelsohn, Eugenio Oñate, Norberto Nigro, Pablo Becker, and Juan Gimenez. Lagrangian versus Eulerian integration errors. *Computer Methods in Applied Mechanics and Engineering*, 293:191–206, 2015.

Ramon Codina. Pressure stability in fractional step finite element methods for incompressible flows. *Journal of Computational Physics*, 170(1):112–140, 2001.

Daniel Duque, Pep Español, and Jaime Arturo de la Torre. Extending linear finite elements to quadratic precision on arbitrary meshes. *Applied Mathematics and Computation*, 301:201–213, 2017.

CGAL Editorial Board. Cgal, Computational Geometry Algorithms Library. http://www.cgal.org.

Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

Y Chen, T A Davis, W W Hager, and S Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM Trans Math Software*, 35(3), 2009.

Daniel Duque. polyFEM 1.0. https://github.com/ddcampayo/polyFEM, 2017. URL `https://github.com/ddcampayo/polyFEM`, `10.5281/zenodo.155375`.

A Numerical methods

For all computational geometry procedures the CGAL 4.7 libraries (CGAL Editorial Board) are used. In particular, the 2D Periodic Delaunay Triangulation package, overloading the vertex base to contain the relevant fields, and the face base to contain information relevant to the edges.

The Eigen 3.0 linear algebra libraries (Guennebaud et al. (2010)) are also employed. For the small linear algebra problem involved in the calculation of the A coefficients, SVD is used, with automatic rank detection. For the large problems involved in the Galerkin procedure, the sparse matrix package is used. The linear systems are solved by a direct method, with best results obtained using the CHOLMOD (Chen et al. (2009)) routines of the suitesparse project (through Eigen wrappers for convenience, class CholmodSupernodalLLT). Slightly worse results are obtained with eigen's build-in SimplicialLDLT class.

Our computations took place on a 4-core Pentium 4 machine with 16 Gb RAM. The code employed, named polyFEM, may be found in (Duque (2017)) under an open source license.

B Quadrature

As explained in the main text, the "mass" assignments involve integrals as in Equation (15).

In this work, the $A(\mathbf{r})$ is a piece-wise linear function, that connects the values of A_μ at the particles. The fact that the locations of mesh nodes and particles do not match makes the integral somewhat cumbersome to evaluate. We have therefore implemented a simple quadrature rule. It is best visualized in 1D, see Figure 6. For the interval between node i and $i+1$ function $A(x)$ is evaluated at x_i, x_{i+1} and the position in between. The resulting three values are then fit to a parabola, whose overlap integral (functional scalar product) with ϕ_i is trivial to evaluate. As shown in the Figure, if no particles lie on the interval, this approximation is exact: the parabola will degenerate to a line in this case. If some particle lies in the interval the result will be, in general, an approximation. For each node i there will be an additional integration from $i-1$ to i.

The same procedure is carried out in 2D, as also depicted in Figure 6. The view is from above, and for simplicity only 2D points are drawn, not functions. Function $A(\mathbf{r})$ is evaluated at the nodes of the triangle that connects node i and two of its neighbors in the triangulation. In addition, it is also evaluated at the mid-points of each segment. With these six values, a recontructing quadratic function is found, whose overlap integral with ϕ_i is trivial. Similar to the 1D case, if the mesh triangle is fully contained within a particle triangle, the integral is exact (since in this case A is linear). If, however, the mesh triangle is crossed by at least one edge of the particle triangulation, the result will be approximate. For each node i the integration must be performed over all its other incident triangles. An equivalent procedure is applied in the inverse procedure of (24).

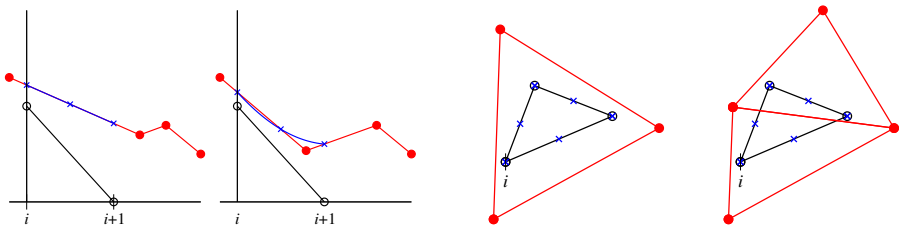

Figure 6: Illustration of the quadrature used in 1D (left two figures), and 2D (right two figures). In 1D, $\phi_i(x)$ is shown as a black line, and $A(x)$ as a red line. The quadratic approximation to the latter is shown as a blue line. In 2D, the black empty circles show the position of nodes, red full circles, the position of particles. Blue crosses are quadrature points in both cases.