A Stable Semi-Implicit Advection Remapping Algorithm for Computational Fluid Dynamics

Brian D. Giffin

August 2, 2018

Abstract

A number of simple linear advection schemes are considered for incompressible fluid flow simulations. The proposed method is implemented within a simple C++ code, and the simulated results are displayed in real time. The primary goal of the method is to achieve reasonable accuracy and stability, while maintaining a high degree of computational efficiency. The method in question utilizes an advection scheme based upon a semi-implicit ALE formulation. Solution variables located both at the nodal positions of the mesh, and at the cell centers are remapped through two distinct tranfer operators which are shown to be composed of the same two underlying steps, but applied in different orders. This allows for the computations pertaining to the advection of both nodal- and cell-based quantities to be consolidated.

1 Introduction

In [1], a simple semi-Lagrangian scheme is employed which utilizes a remapping scheme very similar to the one presented herein. Where these two methods differ lies in the nature of the remapping computations, and in the underlying assumptions regarding the placement of mesh unknowns (either at the cell centers, or at the nodes). In [1], all solution variables are presumed to be collocated with the nodes of the mesh. In what follows, only the velocity field is presumed to be an inherrently nodal field, whereas other field variables may be placed at the cell-centers. This is particularly useful if a mixed pressure-velocity discretization is chosen, wherein the pressure is interpolated in a piece-wise constant fashion. This ensures that the LBB condition is satisfied, thereby yielding a stable numerical method.

In section 2, the proposed numerical method is presented, and various assumptions and simplifications are justified. In section 3, the implementation is described in detail, with some consideration given to the efficiency of the computations. Section 4 presents some preliminary results of the method.

2 Formulation

Consider the incompressible Navier-Stokes equations for an isothermal fluid:

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{1}$$

$$\frac{\partial \rho}{\partial t} = -(\mathbf{u} \cdot \nabla)\rho + \kappa \nabla^2 \rho + S, \tag{2}$$

where $\nu = \mu/\rho_0$ is the kinematic fluid viscosity, κ is the isotropic diffusivity of species ρ, $\mathbf{u}$ is the velocity field for the fluid, and $\mathbf{f}$ and S represent source terms for the momentum and mass conservation equations, respectively.

A common practice in the fluid mechanics literature consists of applying an operator splitting methodology to apply multiple distinct differential operators in sequence, rather than solve the equations as a whole. In the context of fluid mechanics, this consists of splitting the advection terms from the diffusive terms and considering each process in turn. Considering an explicit-in-time (forward-Euler) approximation of the form:

$$\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{t_n} \approx \frac{\mathbf{u}(t_{n+1}) - \mathbf{u}(t_n)}{\Delta t}, \tag{3}$$

$$\left.\frac{\partial \rho}{\partial t}\right|_{t_n} \approx \frac{\rho(t_{n+1}) - \rho(t_n)}{\Delta t}, \tag{4}$$

we may consider an operator splitting-based update algorithm for the Euler equations, evolving the solution state at time t_n to time t_{n+1}:

$$\mathbf{u}_{n+\frac{1}{2}} = \mathbf{u}_n - (\mathbf{u}_n \cdot \nabla)\mathbf{u}_n \Delta t, \tag{5}$$

$$\rho_{n+\frac{1}{2}} = \rho_n - (\mathbf{u}_n \cdot \nabla)\rho_n \Delta t, \tag{6}$$

$$\mathbf{u}_{n+1} = \mathbf{u}_{n+\frac{1}{2}} + \left[\nu \nabla^2 \mathbf{u}_{n+\frac{1}{2}} + \mathbf{f}\right] \Delta t, \tag{7}$$

$$\rho_{n+1} = \rho_{n+\frac{1}{2}} + \left[\kappa \nabla^2 \rho_{n+\frac{1}{2}} + S\right] \Delta t. \tag{8}$$

Equations (5) and (6) constitute the "advection" step, whereas equations (7) and (8) are termed the "diffusion" step.

Solving the advection step for the advected pseudo-mid-step field variables presents the greatest challenge in the realm of fluid dynamics. The remainder of our discussion will focus on a method for solving equations (5) and (6) in an efficient manner.

In what follows, we shall consider an Arbitrary Lagrangian-Eulerian formulation for computing the advected velocity and mass quantities. This consists of a two-step process involving:

1.) Lagrange step: Individual fluid particles bearing the degrees of freedom for the problem are advected in a Lagrangian sense by integrating the equations of motion in time.

2.) Remapping step: The solution fields a remapped back onto the original discretization of the domain.

The Lagrange step is virtually identical to solving the equations of motion for solids (i.e. a "moving-mesh" method.) The remapping step amounts to little more than a methodolgy for transfering the updated solution on the deformed discretization onto the original discretization for the fluid domain.

The manner in which we have chosen to perform our operator splitting yields a very simple form for the Lagrange step:

$$\tilde{\mathbf{x}} = \mathbf{x}_n + \mathbf{u}_n \Delta t, \tag{9}$$

where $\tilde{\mathbf{x}}$ denotes the advected coordinates of the fluid. This is permissible because no forces act upon the fluid during the Lagrange step (by construction). An issue that will need to be addressed later on regards the divergence-free property of the velocity field. Suffice it to say, the propsoed Lagrange step will not respect the incompressibility constraint of the fluid. This property may be enforced by a simple projection approach as suggested in [1].

The remapping step consists of solving for a generic remapped solution field (denoted u) given the solution field $\tilde{u}$ represented on the advected mesh following the Lagrange step. In a mathematical sense, we seek:

$$\min_u \Pi(u), \tag{10}$$

where

$$\Pi(u) \equiv \frac{1}{2}||u - \tilde{u}||^2_\Omega = \frac{1}{2}\int_\Omega (u - \tilde{u})^2 \, d\Omega. \tag{11}$$

In the above, Ω represents the domain occupied by the fluid following the Lagrange step.

For simplicity, consider a two-dimensional Cartesian grid as the discretization of interest (the target mesh in the remapping process). Further, we suppose that the source mesh is a Cartesian mesh which has been only slightly perturbed. The (nodal) fields represented on the source and target meshes may be written in terms of standard bi-linear FE basis functions, respectively:

$$\tilde{u}(\mathbf{x}) = \sum_a \tilde{\varphi}_a(\mathbf{x})\tilde{u}_a, \qquad u(\mathbf{x}) = \sum_a \varphi_a(\mathbf{x}) u_a. \tag{12}$$

The Galerkin approximation to (10) thus yields the following system of equations:

$$u_a = M^{-1}_{ab}\tilde{F}_b \quad \forall a, \tag{13}$$

where

$$M_{ab} = \int_\Omega \varphi_a \varphi_b \, d\Omega, \tag{14}$$

$$\tilde{F}_a = \sum_b \left[\int_\Omega \varphi_a \tilde{\varphi}_b \, d\Omega\right] \tilde{u}_a. \tag{15}$$

Clearly, M_{ab} is the same $\forall t$, and may therefore be solved for once, at the beginning of the analysis. In contrast, $\tilde{F}_a$ will be different for each time step, as it depends implicitly upon the velocity field at time t_n.

If we make the assumption (impose the condition) that $||\mathbf{u}||_{\max} \leq \frac{\Delta x}{2\Delta t}\ \forall t$, then we may carry out an approximate integration over an element $\tilde{\Omega}_q$ in the deformed mesh via 1-point quadrature, leading to the following expression for $\tilde{F}_a$:

$$\int_{\tilde{\Omega}_q} \varphi_a \tilde{u}\, d\tilde{\Omega}_q \approx \varphi_a(\tilde{\mathbf{x}}_q)\, \tilde{u}(\tilde{\mathbf{x}}_q)\, |\tilde{\Omega}_q|, \tag{16}$$

where

$$\tilde{\mathbf{x}}_q = \sum_{a=1}^{4} N_a(\mathbf{0})\, \tilde{\mathbf{x}}_a = \frac{1}{4}\sum_{a=1}^{4} \tilde{\mathbf{x}}_a, \qquad \tilde{u}(\tilde{\mathbf{x}}_q) = \frac{1}{4}\sum_{a=1}^{4} \tilde{u}_a, \tag{17}$$

$$|\tilde{\Omega}_q| \approx \tilde{J}(\tilde{\mathbf{x}}_q)\Delta x^2, \qquad \tilde{J}(\tilde{\mathbf{x}}_q) = \det(\tilde{\mathbf{F}}(\tilde{\mathbf{x}}_q)), \qquad \tilde{\mathbf{F}}(\tilde{\mathbf{x}}_q) = \mathbf{1} + \nabla_x \mathbf{u}_n|_{\tilde{\mathbf{x}}_q} \Delta t. \tag{18}$$

In the above, $N_a(\boldsymbol{\xi})$ denotes the bi-linear FE shape functions for the element Ω_q, evaluated in parent space:

$$N_a(\boldsymbol{\xi}) = \frac{1}{4}(1 + \xi_{1a}\xi_1)(1 + \xi_{2a}\xi_2). \tag{19}$$

Following some straight-forward manipulations, it can be shown that

$$\varphi_a(\tilde{\mathbf{x}}_q) = N_a\left(2\frac{\Delta t}{\Delta x}\tilde{\mathbf{u}}(\tilde{\mathbf{x}}_q)\right), \tag{20}$$

and

$$\tilde{\mathbf{F}}(\tilde{\mathbf{x}}_q) = \mathbf{1} + \frac{\Delta t}{2\Delta x}\sum_{a=1}^{4} \tilde{\mathbf{u}}_a \otimes \boldsymbol{\xi}_a. \tag{21}$$

Additionally, instead of evaluating $\tilde{J}(\tilde{\mathbf{x}}_q)$ exactly, we may wish to compute a first-order approximation to the determinant for the sake of efficiency, as follows:

$$\tilde{J}(\tilde{\mathbf{x}}_q) \approx 1 + \frac{\Delta t}{2\Delta x}\sum_{a=1}^{4} \tilde{\mathbf{u}}_a \cdot \boldsymbol{\xi}_a. \tag{22}$$

For the sake of convenience, let us define the following cell-centered interpolation operator:

$$\mathcal{I}_{qa} \equiv \varphi_a(\mathbf{x}_q) = \tilde{\varphi}_a(\tilde{\mathbf{x}}_q), \tag{23}$$

such that $\mathcal{I}_{qa} : u_a \mapsto u_q$ maps values of a given solution field stored at the nodes of the mesh to values stored at the center of each element. This operator remains the same before and after the mesh coordinates have been advected. Additionally, let us define the following advection-remapping operator:

$$\mathcal{P}_{aq} = M^{-1}_{ab} R_{bq}, \qquad R_{bq} = \varphi_b(\tilde{\mathbf{x}}_q)\, |\tilde{\Omega}_q|, \tag{24}$$

where $\mathcal{P}_{aq} : \tilde{u}_q \mapsto u_a$ maps the cell-centered values of a given solution field at time t_n to their remapped values at the nodes of the target mesh at $t_{n+\frac{1}{2}}$. Consequently, we may compose the preceding operations in different orders to obtain remapping operators for nodal and cell-centered fields, respectively:

$$u_a(t_{n+\frac{1}{2}}) = \mathcal{P}_{aq}\mathcal{I}_{qb}\, u_b(t_n), \tag{25}$$

$$u_q(t_{n+\frac{1}{2}}) = \mathcal{I}_{qa}\mathcal{P}_{ap}\, u_p(t_n). \tag{26}$$

Given the above operators, we may advect any and all solution fields in a field-agnostic manner. Moreover, pre-computing and storing the inverse of the mass matrix and the interpolation matrix will aid in speeding up the computations significantly. The following section presents a simple implementation of the aforementioned advection algorithm.

3 Implementation

An outline of the procedure for the advection step is given below:

0.) Pre-compute and store the following quantities at the beginning of the analysis:

   a.) $\mathcal{I}_{qa}$ via equation (23).

   b.) $\mathcal{M}_{ab}$ via equation (14), and its inverse $\mathcal{M}_{ab}^{-1}$ (or appropriate decomposition).

2.) At each time step, compute and store $\mathcal{R}_{aq}$, $\mathcal{P}_{aq}$ via equation (24).

3.) Advect all nodal fields via equation (25), and all cell-centered fields via equation (26).

4 Numerical Results

A number of simple experiments are presented herein.

References

[1] Jos Stam. Real-time fluid dynamics for games. March 2003.