centered.h documentation

Haochen(Langford) Huang

March 26, 2025

version:draft

1 Introduction

This solver addresses the incompressible Navier-Stokes equations[4, 3]:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u}) = \frac{1}{\rho} \left(-\nabla p + \nabla \cdot (2\mu \mathbf{D})\right) + \mathbf{a} \tag{1}$$

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

where the deformation tensor is defined as $\mathbf{D} = \frac{1}{2}(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)$.

For multiphase flows, the acceleration term $\mathbf{a}$ includes the effects of interfacial forces, such as surface tension. With the incompressibility constraint in equation 2, the velocity $\mathbf{u}$ and pressure gradient ∇p at the next time step can be determined using approximate projection methods. The discrete form is:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^n}{\Delta t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u})^{n+1/2} = \frac{1}{\rho^{n+1/2}} \left(-\nabla p^{n+1} + \nabla \cdot \left(2\mu^{n+1/2} \mathbf{D}^{**}\right)\right) + \mathbf{a}^{n+1/2} \tag{3}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0 \tag{4}$$

Here, the viscosity term is computed implicitly using the intermediate velocity $\mathbf{u}^{**}$.

The following subsections will provide a theoretical overview of each step, with practical program analysis presented in subsequent sections. Notably, in the following subsections, face-centered variables will be denoted by the subscript f, such as μ_f, to distinguish them from cell-centered variables. Readers are also referred to the excellent introduction written by Edoardo Cipriano [2], which has been a great inspiration for this documentation.

1.1 Algorithm Overview

To summarize, solving the Navier-Stokes equations involves four main steps: **Property Update, Advection, Viscosity, and Projection**, corresponding to distinct stages of the solution process. The first three steps involve solving the following equations:

$$\frac{c^{n+1/2} - c^{n-1/2}}{\Delta t} + \mathbf{F}(c, \mathbf{u}) = 0 \tag{5}$$

$$\rho^{n+1/2} \left[\frac{\mathbf{u}^{**} - \mathbf{u}^n}{\Delta t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u})^{n+1/2} - \mathbf{a}^{n-1/2} + \frac{\nabla p^n}{\rho^{n-1/2}}\right] = \nabla \cdot \left(2\mu_f^{n+1/2} \mathbf{D}^{**}\right) \tag{6}$$

$$\mathbf{u}^{***} = \mathbf{u}^{**} - \Delta t \left(\mathbf{a}^{n-1/2} - \frac{\nabla p^n}{\rho^{n-1/2}}\right) \tag{7}$$

Once $\mathbf{u}^{***}$ is computed, we obtain $\mathbf{u}^{n+1}$ by:

$$\mathbf{u}^{n+1} = \mathbf{u}^{***} + \Delta t \left(\mathbf{a}^{n+1/2} - \frac{\nabla p^{n+1}}{\rho^{n+1/2}}\right) \tag{8}$$

Using the incompressibility constraint:

$$\nabla \cdot \mathbf{u}^{n+1} = 0 \tag{9}$$

the pressure gradient ∇p^{n+1} can be computed by solving a Poisson equation in the **Projection** step.

1.1.1 Property Update

Equation 5 is the discrete form of advection equation for markers

$$\frac{\partial c}{\partial t} + \nabla \cdot (c\mathbf{u}) = 0 \tag{10}$$

the scheme of advection term $\mathbf{F}(c, \mathbf{u})$ varies depending on the practical condition. If c denotes the volume of fluids and the header file `two-phase.h` is included, the geometric advection scheme in `vof.h` is activated. Otherwise, if c represents tracers, the BCG scheme is applied to solve the equation. For further details, refer to the `tracers.h` and `vof.h` documentation.

The density ρ and viscosity μ_f at the $n + 1/2$ time step are determined based on the distribution of c, this step is implemented in `two-phase-generic.h`. For those condition with large density ratio, there is option to smear the $\alpha_f = 1/\rho$ in the same header file, as it is considered in poisson equation solver. All these steps are achieved by event succession, note although the body force such as surface tension is associated with distribution of c, it is actually implemented in the **Projection** step also through succession of event 'acceleration'.

1.2 Advection

In this step, we merge the first two terms on the L.H.S. of Equation 6, solving:

$$\rho^{n+1/2}\left[\frac{\mathbf{u}^* - \mathbf{u}^n}{\Delta t} + \nabla \cdot (\mathbf{u} \otimes \mathbf{u})^{n+1/2}\right] = 0 \tag{11}$$

where $\mathbf{u}^*$ is the intermediate velocity. Given the face velocity $\mathbf{u}_f^{n+1/2}$, this equation is solved using functions provided by the header file `bcg.h`, following the BCG scheme. For further details on the BCG scheme, please refer to the `bcg.h` documentation.
The face velocity $\mathbf{u}_f^{n+1/2}$ is first computed using the BCG scheme and then projected to satisfy the non-divergence constraint, as required by the functions provided in `bcg.h`. Since these functions do not directly output values at $n + 1/2$, this computation is performed separately in `centered.h`.

1.3 Diffusion

This step aims to compute $\mathbf{u}^{***}$. Starting from Equation 11, Equation 6 becomes:

$$\rho^{n+1/2}\left[\frac{\mathbf{u}^{**} - \mathbf{u}^*}{\Delta t} - \mathbf{a}^{n-1/2} + \frac{\nabla p^n}{\rho^{n-1/2}}\right] = \nabla \cdot \left(2\mu_f^{n+1/2}\mathbf{D}^{**}\right) \tag{12}$$

Here, $\mathbf{u}^{**}$ is computed implicitly, and $\mathbf{u}^{***}$ is subsequently determined by equation 7.

1.4 Projection

The final step enforces the non-divergence constraint. A key challenge is managing the conversion between cell-centered and face-centered values, as velocities are stored at cell centers, accelerations are stored at cell faces, and the projection function operates on face values.

1.4.1 From Cell-Centered to Face-Centered: Computing p^{n+1}

We compute the pressure p^{n+1} from:

$$\frac{\mathbf{u}_f^{n+1} - \mathbf{u}_f^{***}}{\Delta t} = -\frac{\nabla p^{n+1}}{\rho^{n+1/2}} + \mathbf{a}_f^{n+1/2} \tag{13}$$

$$\nabla \cdot \mathbf{u}_f^{n+1} = 0 \tag{14}$$

which simplifies to:

$$\nabla \left(\frac{\nabla p^{n+1}}{\rho^{n+1/2}} \right) = \nabla \left(\frac{\mathbf{u}_f^{***}}{\Delta t} + \mathbf{a}_f^{n+1/2} \right) \tag{15}$$

Using this relationship, the cell-centered pressure is computed with a multigrid solver applied to the face velocity:

$$\mathbf{u}_f^{****} = \frac{\mathbf{u}^{***}[] + \mathbf{u}^{***}[-1]}{2} + \Delta t \mathbf{a}_f^{n+1/2} \tag{16}$$

Though not required for subsequent steps, the divergence-free face velocity $\mathbf{u}_f^{n+1}$ is also updated.

1.4.2 From Face-Centered to Cell-Centered: Updating $\mathbf{u}^{n+1}$

With p^{n+1} computed, the cell-centered velocity $\mathbf{u}^{n+1}$ is updated:

$$\frac{\mathbf{u}^{n+1} - \mathbf{u}^{***}}{\Delta t} = -\frac{\nabla p^{n+1}}{\rho^{n+1/2}} + \mathbf{a}^{n+1/2} \tag{17}$$

To transform face-centered values to cell-centered values, the face-centered pressure gradient is computed and combined with the face-centered acceleration $\mathbf{a}_f^{n+1/2}$ to yield:

$$\mathbf{g}_f^{n+1} = \mathbf{a}_f^{n+1/2} + \frac{\nabla p^{n+1}}{\rho_f^{n+1/2}} \tag{18}$$

The cell-centered $\mathbf{g}^{n+1}$ is obtained by averaging:

$$\mathbf{g}^{n+1} = \frac{\mathbf{g}_f^{n+1}[1] + \mathbf{g}_f^{n+1}[]}{2} \tag{19}$$

Finally, the velocity is updated:

$$\mathbf{u}^{n+1} = \mathbf{u}^{***} + \Delta t \mathbf{g}^{n+1} \tag{20}$$

1.4.3 Approximate vs. Exact Projection

The approximate projection method, as initially described, pertains to an algorithm where the discrete Laplace operator does not precisely equal the discrete divergence of the discrete gradient, according to Almgren (2000)[1]. This situation arises exclusively on grids where only the cell-center is active, due to the misalignment caused by the face-centered vector. However, `Basilisk` circumvents this issue. Nonetheless, as previously discussed, while the face-centered velocity u_f maintains strict non-divergence, the cell-centered velocity u_c does not, as it is derived by subtracting the average of the updated pressure-acceleration at the face center. Consequently, this is why the projection method utilized by `Basilisk` is termed an approximate projection.

References

[1] A. S. Almgren, J. B. Bell, and W. Y. Crutchfield. "Approximate projection methods: Part I. Inviscid analysis". In: *SIAM J. Sci. Comput.* 22.4 (2000), pp. 1139–1159.

[2] E. Cipriano. *Introduction for centered.h.* `http://basilisk.fr/sandbox/ecipriano/doc/centered`. 2024.

[3] S. Popinet. "A quadtree-adaptive multigrid solver for the Serre–Green–Naghdi equations". In: *J. Comput. Phys.* 302 (2015), pp. 336–358.

[4] S. Popinet. "Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries". In: *J. Comput. Phys.* 190.2 (2003), pp. 572–600.