Short description of the mixed Front-Tracking/Volume-of-Fluid/Level-set algorithm of TrioCFD

Guillaume Bois

Abstract

Short description of the Front-Tracking algorithm implemented in the open-source code TrioCFD.

1 Governing equations

A finite-difference method with a mixed Front-Tracking/Volume-of-Fluid (FT/VoF) algorithm is implemented in TrioCFD. We consider a two-phase flow, with discontinuous interfaces. The formulation rely on the one-fluid Navier-Stokes equations [**Kataoka1986**, **Bunner2003**]

$$\frac{\partial \chi_v}{\partial t} + \mathbf{u} \cdot \nabla \chi_v = 0, \tag{1}$$

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

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u}\,\mathbf{u}) = -\nabla P + \rho \mathbf{g} + \nabla \cdot \left[\mu \left(\nabla \mathbf{u} + \nabla^T \mathbf{u}\right)\right] + \sigma \kappa \mathbf{n}_v \delta^i, \tag{3}$$

$$\frac{\partial \rho\, c_p T}{\partial t} + \nabla \cdot (\rho \mathbf{u} c_p T) = \nabla \cdot (\lambda \nabla T) \tag{4}$$

where each of the one-fluid variables is defined as a mixture of phase variables: $\psi = \sum_k \chi_k \phi_k$ (ψ can be $\mathbf{u}$, P, T, ρ, μ, λ or c_p, respectively the velocity, pressure, temperature, density, dynamic viscosity, conductivity or thermal capacity at constant pressure in phase k). Physical properties are assumed constant within each phase. Both phases are considered incompressible. $\mathbf{g}$ is the gravity vector, σ is the constant surface tension, δ^i is a three-dimensional Dirac impulse at the interface i. $\kappa = -\nabla_s \cdot \mathbf{n}_v$ is twice the mean curvature (usually negative for bubbles) defined from the surface divergence ($\nabla_s \cdot$) of the unit normal to the interface $\mathbf{n}_v$, orientated towards the liquid. The normal vector is related to the phase indicator function χ_v by $\nabla \chi_v = -\mathbf{n}_v \delta^i$ where χ_v is equal to one in the vapour and zero in the liquid. This phase indicator function is transported in equation (1) by the interfacial velocity $\mathbf{u}^i = \mathbf{u}_k$ which is simply the value of the (continuous) fluid velocity at the interface location.

2 Numerical method

The numerical method mostly rely on the front-tracking method [**Bunner2003**], where the interfaces are tracked with a moving surface mesh. While volume of fluid (VoF) and level set (LS) methods only use one fixed volume grid (referred to as eulerian mesh), the front-tracking method involves both a fixed grid and a dynamic surface grid (termed lagrangian mesh or front). The main drawback of this method is the complexity of the dynamic re-meshing algorithms that ensure a good lagrangian mesh quality and handles topology changes. But this costly choice can be used to an advantage to accurately represent the location of the interfaces and associated jumps, the contact angle and the contact line velocity. Although based on a moving surface mesh, our implementation of the front-tracking method also takes advantage of many interesting aspects of the VoF and LS methods.

In the following, we present the main features of the general algorithm which illustrates the time advancement procedure.

2.1 Coupled resolution and time advancement scheme

Many variables are connected and time advancement of the system has to be performed. Here, we present it for the simplest explicit time scheme (euler explicit); TrioCFD code also proposes a coupled 3rd order Runge–Kutta scheme [**Toutant2006th**].

Each equation is updated in turn, but the update of the indicator function is delayed from the actual mesh transport. This enables the computation of velocity, pressure and temperature in an explicit fashion, without having advanced values for physical properties or curvature.

The interface is first transported by the interfacial velocity field ($\mathbf{u}_i^n$ based on an interpolation of the Eulerian $\mathbf{u}^n$. Then, remeshing algorithms are used. Then, temperature or scalar fields are updated before Navier-Sokes equations are solved by a predictor/corrector SIMPLE algorithm to compute and update P and $\mathbf{u}$ at timestep $n+1$. Finally, auxiliary variables (ϕ and ρ, etc.) like interfacial potential, physical properties are updated.
`(see from Probleme_base::iterateTimeStep`
`Schema_Temps_IJK_base::iterateTimeStep )`
The general procedure is :

- Interface motion
- Compute geometrical properties
- Energy update
- Navier-Stokes update
- update all fields

Some fields have two time storages, while others do not. Updating a field F in the code is noted :

$$F \hookleftarrow F^{n+1} \texttt{ --> turn\_the\_wheel() or old\_to\_new() or affectation;} \tag{5}$$

At initialisation, before the time-loop, all variables are initialized (with the same value in both old and new fields when available).

2.1.1 Interface motion

Compute the new interface position:

$$\mathbf{x}^n \to \mathbf{x}^* \to \mathbf{x}^{n+1} \tag{6}$$

$$\mathbf{x}^* = \mathscr{T}(\mathbf{u}^n) = \mathbf{x}^n + \mathscr{I}(u^n)\Delta t \quad \texttt{--> interfaces\_->transporter\_maillage();} \tag{7}$$

$$\mathbf{x}^{n+1} = \mathscr{R}(\mathbf{x}^*) \quad \texttt{--> interfaces\_->remailler\_interface();} \tag{8}$$

`--> interfaces_.deplacer_interface();`
Corresponding code's unkowns are: $\mathbf{x}_i$, the markers positions (and connectivity). $\mathscr{T}$ formally denotes the transport, $\mathscr{I}$ the interpolation from Eulerian faces to Front nodes and $\mathscr{R}$ the complex remeshing operations.

2.1.2 Compute geometrical properties

`--> Class IJK_Geometrical_properties;` Proposal
Compute:

$$\text{Intersect}\left(\mathbf{x}^{n+1}\right) \quad \texttt{--> gp\_->parcourir();}$$

$$\kappa \hookleftarrow \kappa^{n+1} = \kappa(\mathbf{x}^{n+1}) \quad \texttt{--> maillage.calculer\_courbure();}$$

Geometrical properties are different in baseline and Cut-cell.

Baseline Compute :

$$I^{n+1}, \rho^{n+1}, \mu^{n+1}, \lambda^{n+1}, \rho\, c_p^{n+1} \quad \texttt{--> gp\_->compute();}$$

$$\texttt{ou gp\_->calculer\_rho\_mu\_indic...();}$$

$$\phi^{n+1} = \phi(\mathbf{x}^{n+1}) \quad \texttt{--> gp\_.calculer\_potentiel();}$$

Curvature is updated at compute step (no dual storage). However, the potential at the element (i,j,k) is computed from the FT vertex and stored in the future value $n+1$:

$$\phi^{n+1}(i,j,k) = \sum_f a_i^{n+1}(f)\phi^{n+1}(f)/\mathscr{A}^{n+1}(i,j,k)$$

$$\text{with} \quad \phi^{n+1}(f) = \sum_{k=1}^{3} \hat{\phi}^{n+1}(k) \quad \text{and} \quad \hat{\phi}^{n+1}(k) = a_g \mathbf{x}_k^{n+1} \cdot \mathbf{g}\,(\rho_v - \rho_l) + \kappa^{n+1}(k)\sigma$$

where (i,j,k) refers to an Eulerian cell and f to a Front element in the cell, a_i its portion of surface within cell (i,j,k), $\mathscr{A}^{n+1} = \sum_f a_i^{n+1}(f)$ the total interfacial area in that cell, while $\mathbf{x}_k$ is at a Front vertex. $a_g = 1$ when the option `terme_gravite gradI` is activated.

Corresponding code's unkowns are: I, ρ, μ, λ, $\rho\, c_p$ and ϕ. Their present values are not updated yet, unlike the curvature and markers' positions that are updated to κ^{n+1} and x^{n+1}.

Cut-cell Compute :

$$V^{n+1}, \overline{S}_f^{n+1}, \mathbf{b}_{ij}^{n+1} \quad \texttt{--> gp\_->xxxx();} \tag{9}$$

Corresponding code's unkowns are: $\overline{S}_f$, V, $\mathbf{b}_{ij}$, wetting-fraction of cell-faces, partial volume, and face-barycentre and volume-barycentre of a given phase.

Update:

Dorian A
DEPLAC
au bon en
droit

$$\mathbf{b}_{ij} \hookleftarrow \mathbf{b}_{ij}^{n+1} \tag{10}$$

$$\overline{S}_f \hookleftarrow \mathbf{S}^{n+1} \quad \text{or} \quad \left(\mathbf{S}^n + \mathbf{S}^{n+1}\right)/2 \quad \text{or} \quad \overline{S}_f^{n+1} \text{ from Eq. (12)} \quad \texttt{--> gp\_->compute();} \tag{11}$$

with dynamic volume-conservation given by:

$$\overline{S}_f^{n+1} \quad \text{such that} \quad \overline{S}_f^{n+1} = \overline{S}_f^n + \Delta t \delta S \tag{12}$$

$$\mathbf{A}\delta S = B \tag{13}$$

2.1.3 Energy update

For each temperature field in the list, compute:

$$T^\star = T^n \quad \text{or} \quad T^{GFM} = \mathscr{E}(T^t) \quad \texttt{--> thermal\_->gfm(T\_);} \tag{14}$$

$$\frac{dT}{dt} = \left(-\mathscr{C}(T^n) + \frac{1}{\rho\, c_p^n}\mathscr{D}(T^n)\right) \quad \texttt{--> thermal\_->derivee\_en\_temps\_inco(dT\_);} \tag{15}$$

$$T \hookleftarrow T^{n+1} = T^n + \Delta t \frac{dT}{dt} \quad \texttt{--> thermal\_->euler\_explicit\_update();} \tag{16}$$

$$\tag{17}$$

Differences between baseline and cut-cell lie in the operators, in the time-advancing procedure (Reynolds Theorem) and on the general algorithm.

Baseline

$$V\mathscr{C}(T^n) = -\sum_f \left(T^n \mathbf{u}^n \cdot \mathbf{S}_f\right) \tag{18}$$

$$V\mathscr{D}(T^n) = \sum_f \left(\lambda^n \nabla T^n \cdot \mathbf{S}_f\right) \tag{19}$$

Cut-cell

$$\mathscr{C}(T^n) = \tag{20}$$

$$\mathscr{D}(T^n) = \tag{21}$$

2.1.4 Navier-Stokes update

The pressure-velocity coupling is solved with a SIMPLE algorithm with prediction/correction strategy. We compute:

- Prediction:

$$\dot{\mathbf{v}} = \frac{\partial \mathbf{u}^\star}{\partial t} = \underline{\mathscr{C}}(\mathbf{u}^n) + \frac{1}{\rho^n}\underline{\mathscr{D}}(\mathbf{u}^n) + \frac{1}{\rho^n}\underline{\mathscr{L}}_\sigma(I^n, \phi^n) + (1 - a_g)\mathbf{g}$$

$$\texttt{--> ns\_->derivee\_en\_temps\_inco(d\_velocity\_);} \tag{22}$$

$$\mathbf{u}^\star = \mathbf{u}^n + \Delta t \dot{\mathbf{v}} \tag{23}$$

- Projection:

$$\nabla \cdot \frac{1}{\rho^n}\nabla P^{n+1/2} = \nabla \cdot \dot{\mathbf{v}} = \frac{\nabla \cdot \mathbf{u}^\star}{\Delta t} \tag{24}$$

- Correction:

$$\mathbf{u}^{n+1} = \mathbf{u}^\star - \Delta t \frac{1}{\rho^n}\nabla P^{n+1/2} \tag{25}$$

Pressure and velocity fields are updated at the end of this step:

$$\mathbf{u} \hookleftarrow \mathbf{u}^{n+1} \tag{26}$$

$$P \hookleftarrow P^{n+1/2} \tag{27}$$

Baseline

$$V\underline{\mathscr{C}}(\mathbf{u}^n) = V\nabla\cdot(\mathbf{u}^n\mathbf{u}^n) = \sum_f (\mathbf{u}^n\mathbf{u}^n\cdot\mathbf{S}_f) \tag{28}$$

$$V\underline{\mathscr{D}}(\mathbf{u}^n) = \nabla\cdot\left[\mu^n\left(\nabla\mathbf{u}^n + \nabla^T\mathbf{u}^n\right)\right] = \sum_f \left(\mu^n\nabla^\dagger\mathbf{u}^n\cdot\mathbf{S}_f\right) \tag{29}$$

$$\underline{\mathscr{S}}_\sigma^n = \underline{\mathscr{S}}_\sigma(I^n,\phi^n) = -\phi^n\nabla I^n \tag{30}$$

with $\nabla^\dagger\mathbf{u}^n = \nabla\mathbf{u}^n + \nabla^T\mathbf{u}^n$

Cut-cell

$$\underline{\mathscr{C}}(\mathbf{u}^n) = \tag{31}$$

$$\underline{\mathscr{D}}(\mathbf{u}^n) = \tag{32}$$

2.1.5 Update all fields with dual-storage

We update all properties: `--> gp_.turn_the_wheel() or gp_.old_to_new();`

$$I \hookleftarrow I^{n+1} \tag{33}$$

$$\rho \hookleftarrow \rho^{n+1} \quad \text{and} \quad \mu, \lambda, \rho\, c_p \quad \texttt{--> gp\_.calculer\_rho\_mu\_indic();} \tag{34}$$

$$\phi \hookleftarrow \phi^{n+1} \quad \texttt{--> gp\_.calculer\_eulerian\_potentiel();} \tag{35}$$

$$\underline{\mathscr{S}}_\sigma \hookleftarrow \underline{\mathscr{S}}_\sigma^{n+1} \quad \texttt{--> gp\_.calculer\_eulerian\_potentiel();} \tag{36}$$