Design Document

APC 524

Dingyun Liu, Bingjia Yang, Kehan Cai, and Tal Rubin

(Dated: 2021)

I. INTRODUCTION

Fluids are ubiquitous in physical systems of all scales: from astronomical-scale Magnetohydrodynamics, planetary weather models, rivers and pipe-flows, to the smallest blood-vessels. Fluids can describe liquids, gases and plasmas, and appear in the limit of short mean-free-path of an N-body problem.

In physics, fluids are usually described by a set of partial differential equations, which are specified according the model we choose. We can apply different models to study different regimes. First of all, the Navier-Stokes equations predict the motion of Newtonian Fluids. They are composed of the conservations of particle, momentum, and energy, and sometimes accompanied by an equation of state. If we consider conductive fluid subeject to the electromagnetic field such as plasma, then the set of N-S equations must be associated with Maxwell's equations of electromagnetism to account for the reciprocation of the fluid and the field. Furthermore, when we consider the anisotropic transport caused by the gyromotion of magnetized plasma, then we will need to apply Braginskii's equations. The given initial conditions are often dicontinuous to the 1st order in the study of shocks and rarefaction waves with these models, which makes a Riemann problem in the classical system.

In this work, we want to build an fluid for a set of hyperbolic advection-diffusion PDEs. To begin with, we have a preprocessor to generate the mesh and prescribe the initial conditions. Then we can set the boundary condition and select a model that applies to the problem. A robust solver is developped to accostom to the different models. And the results will be plotted by the postprocessor. This solver may be a symplectic phase-space volume preserving, or a "regular" multi-step time stepper such as Runge-Kutta.

II. MATHEMATICAL MODEL

The theory we will attempt to implement appears in[1].

The mathematical description of fluids boils down to conservation laws.

Conservation of mass, or particle number density is called the continuity equation, and

is written in Eulerian form as,

$$\frac{\partial n}{\partial t} + \nabla \cdot (n\mathbf{v}) = 0, \tag{1}$$

with n being the number density, t, the time, and $\mathbf{v}$ the velocity vector.

The momentum evolution equation is,

$$\frac{\partial \mathbf{P}}{\partial t} + \nabla \cdot (\mathbf{vP} + p\mathbf{I} + \pi) = \mathbf{F}, \tag{2}$$

with $\mathbf{P} = mn\mathbf{v}$ being the linear momentum vector, with m denoting the mass of a fluid particle, $\mathbf{vP}$ is a dyadic tensor, obtained by a tensor product of the velocity and momentum, p is the scalar pressure, $\mathbf{I}$ is the unit tensor, π is the viscous stress tensor, and $\mathbf{F}$ is the body force acting on a fluid element.

Possible body forces are Lorentz force for a fluid plasma, friction force for a multi-fluid system, gravity in the presence of a gravitational field. Other body forces can be fictitious such as centrifugal and Coriolis forces. The viscous stress introduces momentum diffusion.

In the case of $\mathbf{F} = 0$, the momentum equation becomes a conservation law:

$$\frac{\partial \mathbf{P}}{\partial t} + \nabla \cdot (\mathbf{vP} + p\mathbf{I} + \pi) = 0. \tag{3}$$

In some fluid models (e.g. Navier–Stokes), these equations are sufficient to close the system of equations. In other models, an energy equation is added:

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + p)\mathbf{v} + \mathbf{q}) = \mathbf{v} \cdot \mathbf{F} + Q, \tag{4}$$

with E being the internal energy of a fluid element, $\mathbf{q}$ begin the heat diffusion, $\mathbf{v} \cdot \mathbf{F}$ is the work done on the fluid element by external forces, and Q being energy source terms such as external heating.

Again, in the case of $\mathbf{v} \cdot \mathbf{F} = 0$ and $Q = 0$, this is a conservation equation.

$$\frac{\partial E}{\partial t} + \nabla \cdot ((E + p)\mathbf{v} + \mathbf{q}) = 0. \tag{5}$$

A. Finite volume method

A popular numerical discretization of this equation set is the finite volume method. Assume the solution domain is divided into several non-overlapping "control volumes", denoted

by C. Integrating the equations over a cell yields,

$$\int_C \frac{\partial n}{\partial t} dV + \int_C \nabla \cdot (n\mathbf{v}) dV = 0, \tag{6}$$

with dV being a volume element of the right dimensionality. Using $\overline{X} = \int_C X dV$, and $\hat{n}$ as the unit vector normal to the surface ∂C of the volume C, the equations for the average density, momentum and energy in a volume become:

$$V\frac{\partial \overline{n}}{\partial t} + \int_{\partial C} \hat{n} \cdot (n\mathbf{v}) dS = 0, \tag{7}$$

$$V\frac{\partial \overline{\mathbf{P}}}{\partial t} + \int_{\partial C} \hat{n} \cdot (\mathbf{v}\mathbf{P} + p\mathbf{I} + \pi)\, dS = \overline{\mathbf{F}}, \tag{8}$$

$$V\frac{\partial \overline{E}}{\partial t} + \int_{\partial C} \hat{n} \cdot ((E+p)\mathbf{v} + \mathbf{q})\, dS = \overline{\mathbf{v} \cdot \mathbf{F}} + \overline{Q}, \tag{9}$$

with dS being a surface element of the right dimensionality, and V being the volume of a cell.

In other words, an average quantity in a cell depends on the fluxes in and out of a cell, and on the average source of that quantity in the cell. Because the computational domain is divided divided into control volumes that share boundaries, the flux out of a cell A though a boundary with cell B is the same flux going into cell B through the same boundary.

Determination of the flux out of a cell boundary is in the heart of the Finite Volume method. The boundary term (appearing in the surface integrals of the previous equations) is called the flux function. The discrete numerical method would generally compute different values of the flux function on each side of the boundary, using the different quantities quantities at each cell.

One can define a "numerical flux" function, which depends on the fluxes on the two sides of the boundary, $F_{num}(F_L, F_R)$. A constraint on this function is $F_{num}(F, F) = F$.

III. PROPOSED WORK

We propose to build a code capable of handling different fluid models, and different solvers.

The code would be split into three computational steps:

1. Pre-processor:

   Meshing, Initial conditions.

2. Solver:

   Time-stepping, Boundary conditions.

3. Post-processor:

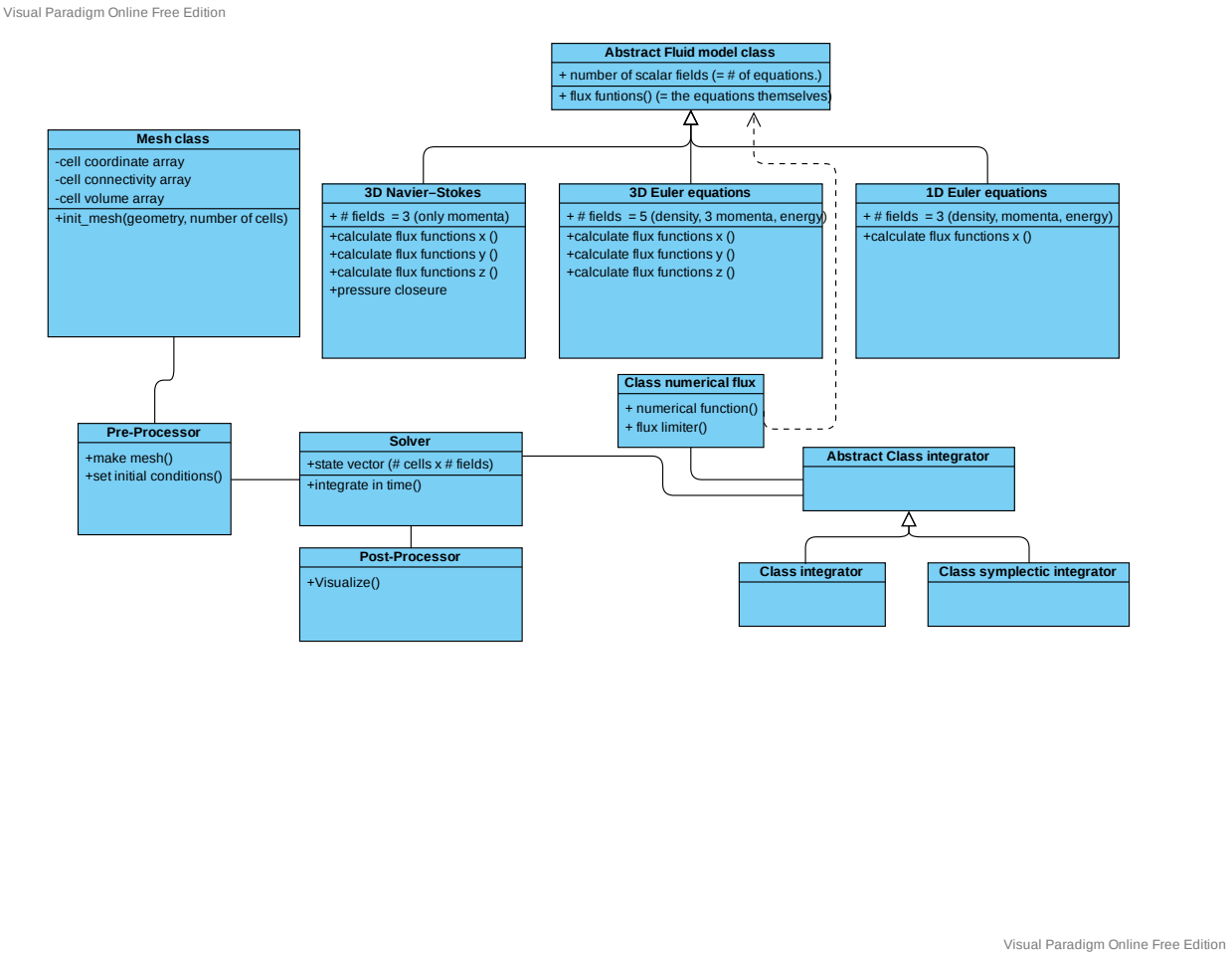
   Visualization.

The code could be written to be extensible from 1 to higher dimensionality.

It is likely we would attempt to use an external library for the pre-processor (such as *MFEM*, or others). We might want to have a choice between "regular" time-stepping algorithms such as Runge-Kutta or Adams-Moulton, and symplectic algorithms such as leap-frog.

The fluid model must be written such that it fits a symplectic time-stepper.

IV. UML DIAGRAM



V. MILESTONES

We have approximately one month to finish the alpha version, and about 2 weeks later the project is due.

The table below lists the major dates and milestones for the project.

Week	Date	Task
1	31 Oct	Revise design document
2	07 Nov	Git + CI setup
3	14 Nov	Implement Riemann solver code
4	22-23 Nov	Check-in with AIs
5	30 Nov	Alpha version due
6	7 Dec	Testing and code modifications
7	14 Dec	Final version due

REFERENCES

[1]E. F. Toro, *Riemann solvers and numerical methods for fluid dynamics: a practical introduction* (Springer Science & Business Media, 2013).