# 1. Lare1D code writing

## 1.1. Language choice

Originally started using Python with a view to using Cython, but realised that it may be a little progressive. Same with using Julia. Using C++ instead and may port over to Cython at some point as an exercise/tutorial writing opportunity. 1D though so no need to involve threading or MPI/openMP.

### 1.1.1. Advantages to C++

- Fast
- Modern constructs compared to fortran
- More readable than Fortran
- I know it…

### 1.1.2. Disadvantages to C++

- Still more clunky/boilerplate code than Python
- Iteration cycle reasonably slow
- Unit testing a little annoying to get started but easy once established

### 1.1.3. Python for analysis

- Quick to write/change/iterate scripts for analysis once variable importing/exporting in place
- Fast enough for decent analysis & can be improved through Cython or other c impls

### 1.1.4. Code problems and possible preventions (AKA HPC code smells)

- Off by one errors; match equations to code so that reading code === reading equations
- Unit testing small simple parts (like derivatives) to consistenly ensure fewer mistakes in tested parts
- Difficulty - Equations hard to unit test, humans can't do so well. Best strategy seems to be match eqns & double check everything

## 1.2. Lagrangian-Remap (LARE)

### 1.2.1. Advantages

- Deals with shocks well (when flux limiters + artificial shock viscosity added)
- Gives comparable results to Roe solvers
- Due to Lagrangian step, equation solving is much simpler than Eulerian equiv

### 1.2.2. Disadvantages

- Complicated two step process
- BUT once remap step solved (& just tedious geometrical problem), solved forever (physics invariant)

### 1.2.3. Steps

- Predictor
- Corrector
- Remap

## 1.3. Setup

### 1.3.1. Equations

In Lagrangian form, the 1D Euler equations are written as

$$\frac{D\rho}{Dt} + \rho\frac{\partial u}{\partial x} = 0, \tag{1}$$

$$\frac{Du}{Dt} + \frac{1}{\rho}\frac{\partial p}{\partial x} = 0, \tag{2}$$

$$\frac{D\varepsilon}{Dt} + \frac{p}{\rho}\frac{\partial u}{\partial x} = 0, \tag{3}$$

where $\rho$ is the density, $u$ is the fluid velocity, $p$ is the pressire and $\varepsilon$ is the specific internal energy density. Use has been made of the Lagrangian or material derivative, $D/Dt = \partial/\partial t + u\partial/\partial x$.

### 1.3.2. Grid

The grid on which the variables are defined is staggered, that is the velocity is defined at the boundary of the cells, with all other variables being defined at the cell centres. This is a strategy used to avoid the checkerboard instability, which arises when the pressure and velocity data points are defined on the same grid TODO cite?. Due to the staggered grid, some care has to be taken to ensure calculations involving mixtures of velocity and another variables are interpolated properly, to avoid numerical leaking of conserved quantities.

A natural method of taking the first derivatives is to use a simple first order finite-difference scheme. Due to the staggered grid, we we can define the derivative of the velocity at the centre of a cell to be the finite difference of the velocity at the boundaries,

$$\frac{\partial u}{\partial x}_i = \frac{u_i - u_{i-1}}{dxb_i}, \tag{4}$$

and similarly we can define the derivative of density (or any of the other cell-centered variables) at the boundaries,

$$\frac{\partial \rho}{\partial x}_i = \frac{\rho_i - \rho_{i-1}}{dxc_i}, \tag{5}$$

noting that the change in $x$ is now given as $dxc_i$.

## 1.4. Predictor step