# Mesh Based Numerical Hydrodynamics of Ideal Gases

**Implementation Details**

Mladen Ivkovic (mladen.ivkovic@hotmail.com)

# 1 Boundary Conditions

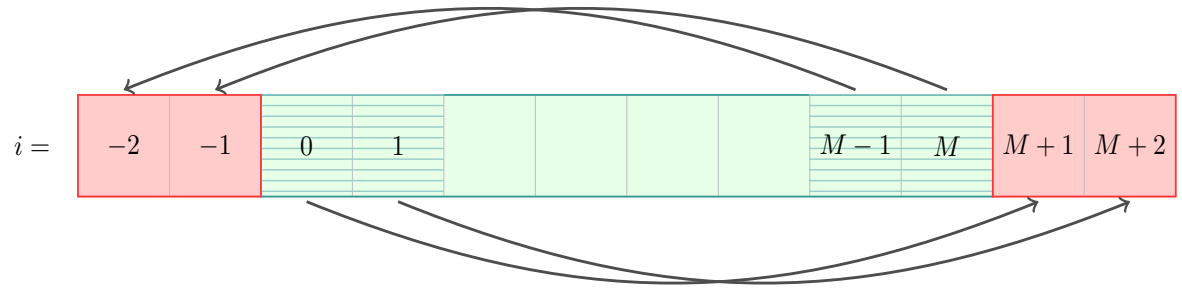


**Figure 1:** Method to obtain periodic boundary conditions. The ghost cells are red, the arrows show what will be copied where.

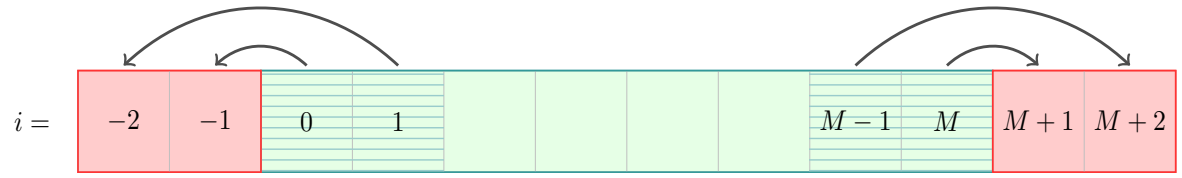


**Figure 2:** Method to obtain wall boundary conditions. The ghost cells are red, the arrows show what will be copied where.

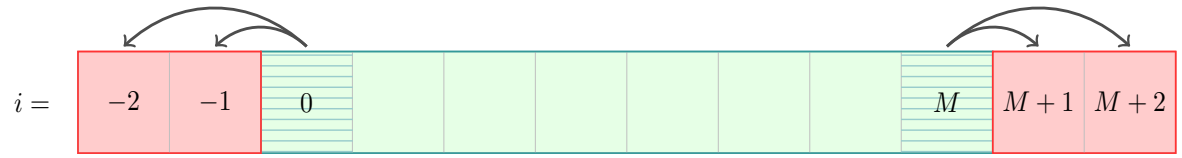


**Figure 3:** Method to obtain transmissive boundary conditions. The ghost cells are red, the arrows show what will be copied where.

There are tricks how to obtain different kinds of boundary conditions. In every case, we add additional cells ("*ghost cells*") in every dimension so we can simulate the desired behaviour. How many cells you need to add depends on the methods (and mostly stencils) you use. If you only take into account one neighbouring cell, then one ghost cell on every boundary suffices. In figures 1, 2, and 3, two ghost cells for a 1D grid are drawn.

Suppose we have 1D grid with $M$ cells and require 2 ghost cells each, which will have indices $-2$, $-1$, $M+1$, and $M+2$. Then we can get:

- **periodic boundary conditions:**

what goes over the right edge, comes back in over the left edge, and vice versa. We achieve this behaviour by enforcing (fig. 1)

$$\begin{aligned}\mathbf{U}_{-2} &= \mathbf{U}_{M-1}\\ \mathbf{U}_{-1} &= \mathbf{U}_{M}\\ \mathbf{U}_{M+1} &= \mathbf{U}_{0}\\ \mathbf{U}_{M+2} &= \mathbf{U}_{1}\end{aligned}$$

- **reflective boundary conditions:**

  pretend there is a wall at the boundary. We achieve that by "mirroring" the cells next to the boundary (fig 2):

$$\begin{aligned}\mathbf{U}_{-2} &= \mathbf{U}_{1}\\ \mathbf{U}_{-1} &= \mathbf{U}_{0}\\ \mathbf{U}_{M+1} &= \mathbf{U}_{M}\\ \mathbf{U}_{M+2} &= \mathbf{U}_{M-1}\end{aligned}$$

  However, every directional component (i.e. velocities/momentum) needs to have the negative value in the ghost cell compared to the real cell.

- **transmissive boundary conditions**:

  Just let things flow out however they want. We achieve this by copying the last boundary cell over and over again, such that it looks that the fluid appears to have that state infinitely, and there are no net fluxes to interfere with the hydrodynamics inside the actual grid (fig. 3)

$$\begin{aligned}\mathbf{U}_{-2} &= \mathbf{U}_{0}\\ \mathbf{U}_{-1} &= \mathbf{U}_{0}\\ \mathbf{U}_{M+1} &= \mathbf{U}_{M}\\ \mathbf{U}_{M+2} &= \mathbf{U}_{M}\end{aligned}$$

# 2 Riemann Solvers

## 2.1 Solution Strategy

## 2.2 Exact Solver

## 2.3 HLL Solver

## 2.4 HLLC Solver

## 2.5 Two-Rarefaction Riemann Solver

## 2.6 Two-Shock Riemann Solver

# 3 Hydrodynamics Methods

## 3.1 Upwind Godunov

# 4 Slope and Flux Limiters

# 5 Advection

What is implemented is the equation

$$\frac{\partial \mathbf{U}}{\partial t} + \mathbf{v} \cdot \frac{\partial \mathbf{U}}{\partial \mathbf{x}} = 0 \tag{1}$$

where we assume that the velocity $\mathbf{v}$ is constant. Therefore, the fluid velocity is never updated, but kept identical to the initial conditions.

You can change that behaviour by removing the `ADVECTION_KEEP_VELOCITY_CONSTANT` macro definition in `defines.h`