Numerical schemes included in Simflowny

Notes written by Carlos Palenzuela
IAC3
(Dated: November 15, 2019)

I. INTRODUCTION

Here we will describe in detail the numerical schemes available in Simflowny and their implementation into the SAMRAI infrastructure. We will discuss explicitly the sub-cycling in time for the Adaptive Mesh Refinement algorithms, including several useful tests. First we introduce the Method of Lines and the time discretization. Then we describe the spatial discretization for smooth and for non-smooth solutions.

Finally we introduce the particle based method SPH, its particularities and coupling to mesh based simulations.

II. THE METHOD OF LINES

The prototype of (first-order in time) evolution equations systems can be written as

$$\partial_t u = f(u) \tag{1}$$

where u is the set of evolution fields and $f(u)$ is an operator containing first and second order derivatives of the fields. We will consider that the system is hyperbolic with a spectral radius c_h(ie, the absolute value of the maximum eigenvalue). This continuum problem can be transformed to a semidiscrete one by allowing only for discrete space positions $\{x_i = i\,\Delta x, y_j = j\,\Delta y, z_k = k\,\Delta z\}$. At each point the evolution is given by the ODE

$$\partial_t U = f(U) + Q_d^m(U) \tag{2}$$

where Q_d is an artificial dissipation operator included to remove the high frequency modes of the solution along a particular direction m, which can not be accurately resolved and achieve stability. The system can be fully discretized by choosing discrete timesteps $t^n = n\Delta t$. Explicit schemes are those for which the future solution which can be written in terms of the current one, namely

$$U^{n+1} = L(U^n) \tag{3}$$

where $L(U^n)$ can be a complicated operator depending on the time integrator, the space discretization and the dissipation.

The discrete system is stable, consistent and convergent to the continuum solution if locally stable time integrator is employed for the time evolution, like a Runge-Kutta of at least third order. Notice that the numerical scheme will be stable as long as the CFL condition $\Delta t \leq \Delta x/c_h$ is fulfilled.

A. The Runge-Kutta time integrator

A explicit Runge-Kutta scheme, applied to system (1), takes the form

$$U^{(i)} = U^n \; + \; \sum_{j=1}^{i-1} b_{ij} k_j \quad , \quad k_j = \Delta t f(U^{(j)})$$

$$U^{n+1} = U^n \; + \; \sum_{i=1}^{s} c_i k_i$$

where $U^{(i)}$ are the auxiliary intermediate values of the Runge-Kutta with s stages. The matrices $B = (b_{ij})$, with $b_{ij} = 0$ for $j \geq i$ are $s \times s$ matrices such that the resulting scheme is explicit and of order p. A Runge-Kutta is characterized by this matrix and the coefficient vector c_i, which can be represented by a tableau in the usual Butcher notation ([1])

$$\begin{array}{c|c} a & B \\ \hline & c^T \end{array}$$

where the coefficients $\tilde{c}$ used for the treatment of non-autonomous systems are given by the consistency relation $a_i = \sum_{j=1}^{i-1} b_{ij}$. These schemes can be denoted as RK(s,p), where the doblet (s,p) characterizes the number of s stages of the explicit scheme and the order p of the scheme.

One can find Runge-Kuttas of order $p = s$ up to $p \leq 4$, making this choice optimal. A very well known fourth order Runge-Kutta with an effective CFL of 2 is given in Table I.

TABLE I: Tableau for a very standard explicit RK(4,4)

0	0	0	0	0
1/2	1/2	0	0	0
1/2	0	1/2	0	0
1	0	0	1	0
	1/6	2/6	2/6	1/6

Solutions of conservation equations have some norm that decreases in time. It would be desirable, in order to avoid spurious numerical oscillations arising near discontinuities of the solution, to maintain such property at a discrete level by the numerical method. If U^n represents a vector of solution values at the time $t = n\ \Delta t$, then a numerical scheme is said to be strong stability preserving (SSP) if maintains $\|U^{n+1}\| \leq \|U^n\|$ for all $n \geq 0$ in a given norm $\|\cdot\|$.

There are different SSP-RK schemes available in the literature. A very popular third order one [2], with an optimal CFL of 1, is given in Table II.

TABLE II: Tableau for the explicit RK-SSP(3,3) scheme

0	0	0	0
1	1	0	0
1/2	1/4	1/4	0
	1/6	1/6	4/6

Notice that a dense ouput interpolator can be constructed by using the sub-steps of the RK. Its generic forms is

$$U^{n+\theta} = U^n + \sum_{j=1}^{s} b_j(\theta) k_j \quad , \quad \theta = \frac{t - t^n}{t^{n+1} - t^n} \tag{4}$$

where $b_i(\theta)$ are the coefficients to build the interpolator for a given RK scheme. Notice that the $m-$derivative can also be computed from this dense output interpolator

$$\frac{d^m}{dt^m} U(t^n + \theta \Delta t) = \frac{1}{h^m} \sum_{j=1}^{s} k_j \frac{d^m}{d\theta^m} b_j(\theta) + O(h^{4-m}) \quad , \tag{5}$$

For the standard RK$(4,4)$, it can be shown that there is a unique third order interpolator that can be written as

$$b_1(\theta) = \theta - \frac{3}{2}\theta^2 + \frac{2}{3}\theta^3 \quad , \quad b_2(\theta) = b_3(\theta) = \theta^2 - \frac{2}{3}\theta^3 \quad , \quad b_4(\theta) = \frac{-1}{2}\theta^2 - \frac{2}{3}\theta^3 \tag{6}$$

For the SSP-RK$(3,3)$ there is a second order interpolator which also satisfies the SSP condition

$$b_1(\theta) = \theta - \frac{5}{6}\theta^2 \quad , \quad b_2(\theta) = \frac{1}{6}\theta^2 \quad , \quad b_3(\theta) = \frac{4}{6}\theta^2 \tag{7}$$

III. SPATIAL DISCRETIZATION FOR SMOOTH SOLUTIONS

As a default we will use standard fourth-order centered finite difference. The first order derivative operators have the form

$$\partial_x U_{i,j,k} \ = \ \frac{1}{12\Delta x} \left(U_{i-2,j,k} - 8\, U_{i-1,j,k} + 8\, U_{i+1,j,k} - U_{i+2,j,k} \right) + \mathcal{O}(\Delta x^4) \tag{8}$$

The second order can be constructed as the first order derivate applied twice on $U_{i,j,k}$. This is an acceptable choice for the cross-derivatives. For the instance, for the xy-derivative,

$$\partial_{xy} U_{i,j,k} \;=\; \partial_x \left(\partial_y U_{i,j,k}\right) = \partial_y \left(\partial_x U_{i,j,k}\right) \tag{9}$$

However, the stencil of the second order xx-derivative is twice larger, so it is preferable to change a different operator that is still fourth order and keeps the original stencil

$$\partial_{xx} U_{i,j,k} \;=\; \frac{1}{12\Delta x^2}\left(-U_{i-2,j,k} + 16\,U_{i-1,j,k} - 30\,U_{i,j,k} + 16\,U_{i+1,j,k} - U_{i+2,j,k}\right) + \mathcal{O}(\Delta x^4) \tag{10}$$

We use centered derivative operators for all the derivative terms except for the advection terms, which are generically proportional to a vector β^i. In those case, we will use one-side derivative schemes,

$$\partial_x U_{i,j,k} \;=\; \frac{1}{12\Delta x}\left(-U_{i-3,j,k} + 6\,U_{i-2,j,k} - 18\,U_{i-1,j,k} + 10\,U_{i,j,k} + 3\,U_{i+1,j,k}\right) + \mathcal{O}(\Delta x^4) \quad , \quad \beta^x < 0 \tag{11}$$

$$\partial_x U_{i,j,k} \;=\; \frac{1}{12\Delta x}\left(U_{i+3,j,k} - 6\,U_{i+2,j,k} + 18\,U_{i+1,j,k} - 10\,U_{i,j,k} - 3\,U_{i-1,j,k}\right) + \mathcal{O}(\Delta x^4) \quad , \quad \beta^x > 0 \tag{12}$$

Since numerical schemes for fluids are usually only third order in space, we can also consider only third order space discretizations to decrease the stencil of the scheme. This can be done for the advection terms by using the following operators

$$\partial_x U_{i,j,k} \;=\; \frac{1}{6\Delta x}\left(U_{i-2,j,k} - 6\,U_{i-1,j,k} + 3\,U_{i,j,k} + 2\,U_{i+1,j,k}\right) + \mathcal{O}(\Delta x^3) \quad , \quad \beta^x < 0 \tag{13}$$

$$\partial_x U_{i,j,k} \;=\; \frac{1}{6\Delta x}\left(-U_{i+2,j,k} + 6\,U_{i+1,j,k} - 3\,U_{i,j,k} - 2\,U_{i-1,j,k}\right) + \mathcal{O}(\Delta x^3) \quad , \quad \beta^x > 0 \tag{14}$$

A. The Dissipation

As it was mentioned before, we use artificial dissipation to remove the high frequency modes of the solution which are not truly represented in our numerical grid (i.e., their wavelength is smaller than the grid size δx). We will use the Kreiss-Oliger dissipation operator [3]

$$Q_d^x = \sigma(-1)^{r-1}\Delta x^{2r-1}\left(D_+^x\right)^r\left(D_-^x\right)^r \tag{15}$$

where

$$D_+^x U_{i,j,k} = \frac{U_{i+1,j,k} - U_{i,j,k}}{\Delta x} \qquad , \qquad D_-^x U_{i,j,k} = \frac{U_{i,j,k} - U_{i-1,j,k}}{\Delta x} \tag{16}$$

where $\sigma \geq 0$ is dissipative parameter. If the accuracy of the scheme without artificial dissipation is q, choosing $2r - 1 \geq q$ does not affect the accuracy of the scheme.

For a fourth order scheme we must use $r = 3$, leading to an operator for the x-direction

$$\begin{aligned} Q_d^x U_{i,j,k} \;&=\; \sigma(\Delta x)^5\left(D_+^x\right)^3\left(D_-^x\right)^3 U_{i,j,k} \\ &=\; \frac{\sigma}{64\Delta x}\left(U_{i-3,j,k} - 6\,U_{i-2,j,k} + 15\,U_{i-1,j,k} - 20\,U_{i,j,k} + 15\,U_{i+1,j,k} - 6\,U_{i+2,j,k} + U_{i+3,j,k}\right) + \mathcal{O}(\Delta x^5) \end{aligned} \tag{17}$$

For a third order scheme we can use $r = 2$, leading to an operator for the x-direction

$$\begin{aligned} Q_d^x U_{i,j,k} \;&=\; -\sigma(\Delta x)^3\left(D_+^x\right)^2\left(D_-^x\right)^2 U_{i,j,k} \\ &=\; -\frac{\sigma}{16\Delta x}\left(U_{i-2,j,k} - 4\,U_{i-1,j,k} + 6\,U_{i,j,k} - 4\,U_{i+1,j,k} + U_{i+2,j,k}\right) + \mathcal{O}(\Delta x^3) \end{aligned} \tag{18}$$

IV. METHODS FOR FLUIDS

However, these simple finite difference operators are not the optimal choice for the spatial discretization of fluxes in intrinsically non-linear systems like the eMHD. In that case, it is advisable to use High-Resolution-Shock-Capturing (HRSC) methods [**?**], to deal with the possible appearance of shocks and to take advantage of the existence of weak

solutions in the equations. We will therefore use a conservative scheme to discretize the fluxes, which in one dimension reads:

$$\partial_t \mathbf{U}_i = -\frac{1}{\Delta x}\left(\mathbf{F}^x_{i+1/2} - \mathbf{F}^x_{i-1/2}\right)$$

where $\mathbf{F}^x_{i\pm1/2}$ are the set of fluxes along the x-direction evaluated at the interfaces between two neighbouring cells, located at $x_{i\pm1/2}$. The crucial issue in HRSC methods is how to approximately solve the Riemann problem, by reconstructing the fluxes at the interfaces such that no spurious oscillations appear in the solutions. This calculation consists in two steps:

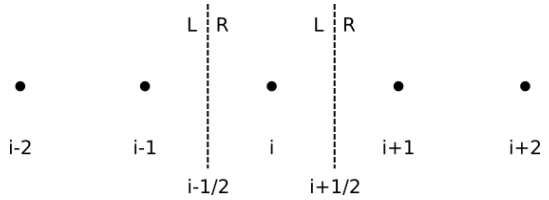


FIG. 1: The computational uniform grid x_i. The left (L) and right (R) states reconstructed at the interfaces $x_{i\pm1/2}$ are required to evolve the solution U_i.

- We consider the following combination of the fluxes and the fields, at each node i:

$$F^{\pm}_i = \frac{1}{2}\left(F_i \pm \lambda U_i\right) \tag{19}$$

where λ is the maximum propagation speed of the system in the neighboring points. Then, we reconstruct the flux at the left (right) of each interface, e.g. $F^L_{i+1/2}$ ($F^R_{i+1/2}$), using the values $\{F^+\}$ ($\{F^-\}$) from the neighboring nodes $\{x_{i-n},..,x_{i+1+n}\}$. The number $2(n+1)$ of such neighbors used in the reconstruction procedure depends on the order of the method. *Simflowny* already incorporates some commonly used reconstructions, like PPM [**?**] and MP5 [**?**], as well as other implementations like the FDOC families [**?**] which are almost as fast as centered finite difference schemes at the cost of some bounded spikes near the shock regions. Here we mostly focused on the Weighted-Essentially-Non-Oscillatory (WENO) reconstructions [**? ?**], which is our preferred choice for their flexibility (i.e., they can achieve any order of accuracy) and robustness. The detailed implementation of the WENO flavors used here can be found in [**?**].

- We use a flux formula to compute the final flux at each interface, e.g.:

$$F_{i+1/2} = F^L_{i+1/2} + F^R_{i+1/2} \tag{20}$$

Note that our reconstruction method does not require the characteristic decomposition of the system of equations (i.e., the full spectrum of characteristic velocities). At the lowest order reconstruction $F^L_{i+1/2} = F^+_i$ and $F^R_{i+1/2} = F^-_{i+1}$, so that the flux formula (28) reduces to the popular and robust Local-Lax-Friedrichs flux [**?**]. The LLF flux formulat can be summarized as follows: once we have reconstructed the fields at a given interface from the left (L) and from the right (R) we can use all that information to construct the HLL flux

$$F^{HLL} = \frac{1}{S^R - S^L}\left[S^R F^L - S^L F^R + S^R S^L (U^R - U^L)\right] \tag{21}$$

where $F^L = F(U^L)$,$F^R = F(U^R)$ and S^L, S^R are the fastest speed traveling to the left and to the right, respectively, namely

$$S^L = {}^{(-)}\lambda^L \quad , \quad S^R = {}^{(+)}\lambda^R \quad , \tag{22}$$

A more restrictive choice would be

$$S^L = min({}^{(-)}\lambda^L, {}^{(-)}\lambda^R) \quad , \quad S^R = max({}^{(+)}\lambda^L, {}^{(+)}\lambda^R) \quad . \tag{23}$$

The simplest choice assumes that $S^L = -S^R = S$. Substituting this expression into the HLL flux one can obtain the Local-Lax-Friedrichs flux (or Rusanov)

$$F^{LLF} = \frac{1}{2}\left[F^L + F^R - S(U^R - U^L)\right] \tag{24}$$

The reconstruction procedure can be performed at different orders. We have implemented several of the most commonly used reconstructions, like PPM [4] and MP5 [5], and other implementations like the FDOC families [6] which are almost as fast as centered Finite Difference at the cost of some bounded oscillations near the shock region. Here we present a short summary of the Weighted-Essentially-Non-Oscillatory (WENO) reconstructions [7, 8], which is our preferred choice for their flexibility (i.e., they can achieve any order of accuracy) and robustness. The detailed implementation of the WENO flavors used here can be found in Appendices **??** and **??**, while that details of the other methods can be found in a recent review [9].

There is another decomposition that allow us to recover the HLL flux formula[10]. We consider the following combination of the fluxes and the fields, at each node i:

$$F_i^+ = \frac{+\lambda^{(+)}}{\lambda^{(+)} - \lambda^{(-)}} \left(F_i - \lambda^{(-)} U_i \right) \tag{25}$$

$$F_i^- = \frac{-\lambda^{(-)}}{\lambda^{(+)} - \lambda^{(-)}} \left(F_i - \lambda^{(+)} U_i \right) \tag{26}$$

where $\lambda^{(\pm)}$ are the maximum propagation speed of the system moving to the right/left in the neighboring points, namely

$$\lambda^{(-)} = min(\lambda_i^{(-)}, 0) \quad , \quad \lambda^{(+)} = max(\lambda_i^{(+)}, 0) \tag{27}$$

Then, we reconstruct the flux at the left (right) of each interface, e.g. $F^L_{i+1/2}$ ($F^R_{i+1/2}$), using the values $\{F^+\}$ ($\{F^-\}$) from the neighboring nodes $\{x_{i-n}, .., x_{i+1+n}\}$. The number $2(n+1)$ of such neighbors used in the reconstruction procedure depends on the order of the method. Finally, we use a flux formula to compute the final flux at each interface, e.g.:

$$F_{i+1/2} = F^L_{i+1/2} + F^R_{i+1/2} \tag{28}$$

A. Linear reconstruction

The simplest reconstruction is the piecewise linear, where the field in the cell i can be approximated as

$$u_{i,j}(\tilde{x}, \tilde{y}) = \bar{u}_{i,j} + \Delta_x \bar{u}_{i,j} \tilde{x} + \Delta_y \bar{u}_{i,j} \tilde{y} \tag{29}$$

where $\{\tilde{x} = (x - x_i)/\Delta x, \tilde{y} = (y - y_j)/\Delta y\}$ and with $\{\Delta_x \bar{u}_{i,j}, \Delta_x \bar{u}_{i,j}\}$ representing the linear variation of $\bar{u}_{i,j}$ within the cell, namely

$$\Delta_x \bar{u}_{i,j} = Limiter\left(\bar{u}_{i+1,j} - \bar{u}_{i,j}, \bar{u}_{i,j} - \bar{u}_{i-1,j} \right) \tag{30}$$

$$\Delta_y \bar{u}_{i,j} = Limiter\left(\bar{u}_{i,j+1} - \bar{u}_{i,j}, \bar{u}_{i,j} - \bar{u}_{i,j-1} \right) \tag{31}$$

In order to avoid oscillations, these slopes must be limited. The safest choice is probably the minmod limiter

$$minmod(a, b) = \frac{1}{2}(sgn(a) + sgn(b))\, min(|a|, |b|) \tag{32}$$

Another popular choice, much less dissipative, is the Monotonized Central (MC) limiter

$$MC(a, b) = \frac{1}{2}(sgn(a) + sgn(b))\, min(\frac{1}{2}|a + b|, 2|a|, 2|b|) \tag{33}$$

B. Parabolic reconstruction

The next order of reconstruction is the piecewise parabolic method (PPM), where the field in the cell i can be approximated as

$$u_i(\tilde{x}) = \bar{u}_i + \hat{u}_x \tilde{x} + \hat{u}_{xx}(\tilde{x} - 1/12) \tag{34}$$

The procedure is at follows:

- construct the limited slopes in each zone

$$\Delta u_{i+1} = MC\Big(\bar{u}_{i+2} - \bar{u}_{i+1}, \bar{u}_{i+1} - \bar{u}_i\Big) \tag{35}$$

$$\Delta u_i = MC\Big(\bar{u}_{i+1} - \bar{u}_i, \bar{u}_i - \bar{u}_{i-1}\Big) \tag{36}$$

- calculate $u^L_{i+1/2}$ and $u^R_{i+1/2}$ in that cell by using

$$u^L_{i+1/2} = \frac{1}{2}(\bar{u}_{i+1} + \bar{u}_i) - \frac{1}{6}(\Delta u_{i+1} - \Delta u_i) \tag{37}$$

and $u^R_{i+1/2} = u^L_{i+1/2}$.

- enforce the monotonicity conditions resetting $u^R_{i-1/2}$ and $u^L_{i+1/2}$ in each zone as follows

$$if \quad (u^L_{i+1/2} - \bar{u}_i)(\bar{u}_i - u^R_{i-1/2}) \le 0 \quad \rightarrow \quad u^L_{i+1/2} = \bar{u}_i \quad and \quad u^R_{i-1/2} = \bar{u}_i \tag{38}$$

$$if \quad (u^L_{i+1/2} - u^R_{i-1/2})\big(\bar{u}_i - \frac{1}{2}(u^L_{i+1/2} + u^R_{i-1/2})\big) > \frac{1}{6}(u^L_{i+1/2} - u^R_{i-1/2})^2 \quad \rightarrow \quad u^R_{i-1/2} = 3\bar{u}_i - 2u^L_{i+1/2} \tag{39}$$

$$if \quad \frac{-1}{6}(u^L_{i+1/2} - u^R_{i-1/2})^2 > (u^L_{i+1/2} - u^R_{i-1/2})\big(\bar{u}_i - \frac{1}{2}(u^L_{i+1/2} + u^R_{i-1/2})\big) \quad \rightarrow \quad u^L_{i+1/2} = 3\bar{u}_i - 2u^R_{i-1/2} \tag{40}$$

Notice that a less restrictive condition can substitute the first one, namely

$$if \quad (u^L_{i+1/2} - \bar{u}_i)(\bar{u}_i - u^R_{i-1/2}) \le 0 \quad \rightarrow \quad u^L_{i+1/2} = \bar{u}_i + \Delta u_i/2 \quad and \quad u^R_{i-1/2} = \bar{u}_i - \Delta u_i/2 \tag{41}$$

C. WENO schemes

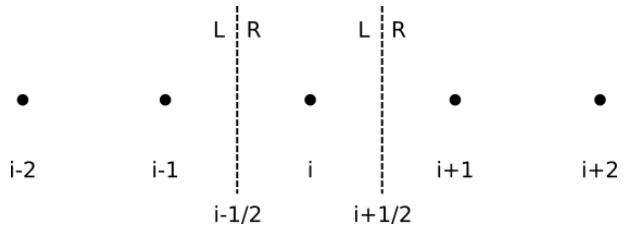


FIG. 2: *WENO notation.*

Given the cell averages u_i of a function $u(x)$, for each cell I_i, we obtain upwind biased $(2k1)$-th order approximations to the function $u(x)$ at the cell boundaries, denoted by ${}^R u_{i-1/2}$ and ${}^L u_{i+1/2}$, in the following way:

- Obtain the k reconstructed values ${}^L u^{(r)}_{i+1/2}$ and ${}^R u^{(r)}_{i-1/2}$ of k-th order accuracy,

$$\begin{aligned} {}^L u^{(r)}_{i+1/2} &= \sum_{j=0}^{k-1} c_{r,j} u_{i-r+j} \\ {}^R u^{(r)}_{i-1/2} &= \sum_{j=0}^{k-1} c_{r-1,j} u_{i-r+j} \end{aligned} \tag{42}$$

with $r = 0..k-1$.

- Find the smooth indicators ${}^L\beta^{(r)}_{i+1/2}$ and ${}^R\beta^{(r)}_{i+1/2}$,that will depend on the order k

- Find the $(2k-1)^{\text{th}}$ order reconstruction

$$\begin{aligned} u^L_{i+1/2} &= \sum_{r=0}^{k-1} \omega^{(r)}_{i+1/2} {}^L u^{(r)}_{i+1/2} \\ u^R_{i-1/2} &= \sum_{r=0}^{k-1} \tilde{\omega}^{(r)}_{i-1/2} {}^R u^{(r)}_{i-1/2} \end{aligned} \tag{43}$$

with $r = 0..k-1$. The weights, $\omega^{(r)}_{i+1/2}$ for the left and $\tilde{\omega}^{(r)}_{i+1/2}$ for the right, can be constructed in the following way

$$\begin{aligned} \omega^{(r)}_{i+1/2} &= \frac{\alpha^{(r)}_{i+1/2}}{\sum_{s=0}^{k-1} \alpha^{(s)}_{i+1/2}} \quad , \quad \alpha^{(r)}_{i+1/2} = \frac{d_r}{(\epsilon + {}^L\beta^{(r)}_{i+1/2})^2} \\ \tilde{\omega}^{(r)}_{i-1/2} &= \frac{\tilde{\alpha}^{(r)}_{i-1/2}}{\sum_{s=0}^{k-1} \tilde{\alpha}^{(s)}_{i-1/2}} \quad , \quad \tilde{\alpha}^{(r)}_{i-1/2} = \frac{\tilde{d}_r}{(\epsilon + {}^R\beta^{(r)}_{i-1/2})^2} \end{aligned} \tag{44}$$

where $\tilde{d}_r = d_{k-1-r}$ and ϵ is a very small number of the order 10^{-10}.

The coefficients $c_{r,j}$ for the $k = 2$ and $k = 3$ can be found in the tables.

TABLE III: Table for $k = 2$

r	j=0	j=1
-1	3/2	-1/2
0	1/2	1/2
1	-1/2	3/2

TABLE IV: Table for $k = 3$

r	j=0	j=1	j=2
-1	11/6	-7/6	1/3
0	1/3	5/6	-1/6
1	-1/6	5/6	1/3
2	1/3	-7/6	11/6

while that the coefficients d_r are

$$k = 2 \qquad d_0 = 2/3 \quad , \quad d_1 = 1/3 \tag{45}$$

$$k = 3 \qquad d_0 = 3/10 \quad , \quad d_1 = 6/10 \quad , \quad d_2 = 1/10 \tag{46}$$

1. Third order WENO (k=2)

Let us write explicitly the procedure for the third order WENO, obtained with $k = 2$.

- The k reconstructed values ${}^L u^{(r)}_{i+1/2}$ and ${}^R u^{(r)}_{i-1/2}$ of k-th order accuracy,

$$\begin{aligned} {}^L u^{(0)}_{i+1/2} &= \frac{1}{2}u_i + \frac{1}{2}u_{i+1} \equiv A^{(1)}(u_i, u_{i+1}) \\ {}^L u^{(1)}_{i+1/2} &= -\frac{1}{2}u_{i-1} + \frac{3}{2}u_i \equiv A^{(0)}(u_{i-1}, u_i) \\ {}^R u^{(0)}_{i-1/2} &= -\frac{1}{2}u_{i+1} + \frac{3}{2}u_i \equiv A^{(0)}(u_{i+1}, u_i) \\ {}^R u^{(1)}_{i-1/2} &= \frac{1}{2}u_i + \frac{1}{2}u_{i-1} \equiv A^{(1)}(u_i, u_{i-1}) \end{aligned} \tag{47}$$

The ${}^L u^{(r)}_{i-1/2}$ and ${}^R u^{(r)}_{i+1/2}$ can be obtained by substituting i by $i \pm 1$ in the previous expressions.

- Find the smooth indicators ${}^{L}\beta^{(r)}_{i+1/2}$ and ${}^{R}\beta^{(r)}_{i+1/2}$

  First option is when the smooth indicator is the same along all the cell ${}^{L}\beta^{(r)}_{i+1/2} = {}^{R}\beta^{(r)}_{i-1/2}$

$$
\begin{aligned}
{}^{L}\beta^{(0)}_{i+1/2} &= (u_{i+1}-u_i)^2 = B^{(1)}(u_{i+1},u_i) && (48)\\
{}^{L}\beta^{(1)}_{i+1/2} &= (u_i-u_{i-1})^2 = B^{(0)}(u_i,u_{i-1}) \\
{}^{R}\beta^{(0)}_{i-1/2} &= (u_i-u_{i+1})^2 = B^{(0)}(u_i,u_{i+1}) == B^{(0)}(u_{i+1},u_i) && (49)\\
{}^{R}\beta^{(1)}_{i-1/2} &= (u_{i-1}-u_i)^2 = B^{(1)}(u_{i-1},u_i) = B^{(0)}(u_i,u_{i-1}) && (50)
\end{aligned}
$$

  Second option is when the smooth indicator is mirrored ${}^{L}\beta^{(0)}_{i+1/2} = {}^{R}\beta^{(1)}_{i-1/2}$ and ${}^{L}\beta^{(1)}_{i+1/2} = {}^{R}\beta^{(2)}_{i-1/2}$, switching $\pm$ by $\mp$,

$$
\begin{aligned}
{}^{L}\beta^{(0)}_{i+1/2} &= (u_{i+1}-u_i)^2 = B^{(1)}(u_{i+1},u_i) && (51)\\
{}^{L}\beta^{(1)}_{i+1/2} &= (u_i-u_{i-1})^2 = B^{(0)}(u_i,u_{i-1}) \\
{}^{R}\beta^{(0)}_{i-1/2} &= (u_i-u_{i+1})^2 = B^{(0)}(u_{i-1},u_i) && (52)\\
{}^{R}\beta^{(1)}_{i-1/2} &= (u_{i-1}-u_i)^2 = B^{(1)}(u_i,u_{i+1}) && (53)
\end{aligned}
$$

  Again, ${}^{L}\beta^{(r)}_{i-1/2}$ and ${}^{R}\beta^{(r)}_{i+1/2}$ can be obtained by substituting i by $i\pm 1$ in the previous expressions.

- Find the 3$^{\text{rd}}$ order reconstruction

$$
\begin{aligned}
u^L_{i+1/2} &= \omega^{(0)}_{i+1/2}\,{}^{L}u^{(0)}_{i+1/2} + \omega^{(1)}_{i+1/2}\,{}^{L}u^{(1)}_{i+1/2} \\
u^R_{i-1/2} &= \tilde{\omega}^{(0)}_{i-1/2}\,{}^{R}u^{(0)}_{i-1/2} + \tilde{\omega}^{(1)}_{i-1/2}\,{}^{R}u^{(1)}_{i-1/2} && (54)
\end{aligned}
$$

  with weights $\omega^{(r)}_{i+1/2}$ and $\tilde{\omega}^{(r)}_{i+1/2}$ constructed by using the generic formulas

$$
\omega^{(r)}_{i+1/2} = \frac{\alpha^{(r)}_{i+1/2}}{\sum_{s=0}^{k-1}\alpha^{(s)}_{i+1/2}} \quad , \quad \tilde{\omega}^{(r)}_{i-1/2} = \frac{\tilde{\alpha}^{(r)}_{i-1/2}}{\sum_{s=0}^{k-1}\tilde{\alpha}^{(s)}_{i-1/2}} \qquad (55)
$$

  by using

$$
\begin{aligned}
\alpha^{(0)}_{i+1/2} &= \frac{2/3}{(\epsilon + {}^{L}\beta^{(0)}_{i+1/2})^2} \quad , \quad \alpha^{(1)}_{i+1/2} = \frac{1/3}{(\epsilon + {}^{L}\beta^{(1)}_{i+1/2})^2} \\
\tilde{\alpha}^{(0)}_{i-1/2} &= \frac{1/3}{(\epsilon + {}^{R}\beta^{(0)}_{i-1/2})^2} \quad , \quad \tilde{\alpha}^{(1)}_{i-1/2} = \frac{2/3}{(\epsilon + {}^{R}\beta^{(1)}_{i-1/2})^2} && (56)
\end{aligned}
$$

  Notice that we need also the reconstructed values from the other cells

$$
\begin{aligned}
u^L_{i-1/2} &= \omega^{(0)}_{i-1/2}\,{}^{L}u^{(0)}_{i-1/2} + \omega^{(1)}_{i-1/2}\,{}^{L}u^{(1)}_{i-1/2} \\
u^R_{i+1/2} &= \tilde{\omega}^{(0)}_{i+1/2}\,{}^{R}u^{(0)}_{i+1/2} + \tilde{\omega}^{(1)}_{i+1/2}\,{}^{R}u^{(1)}_{i+1/2} && (57)
\end{aligned}
$$

  by substituting i by $i\pm 1$ in the previous expressions.

Lately there have been some proposals by Carpenter(2009) to change the weights by using

$$
\alpha^{(r)}_{i+1/2} = d_r\left(1+\frac{\tau}{\epsilon + {}^{L}\beta^{(r)}_{i+1/2}}\right) \quad , \quad \tilde{\alpha}^{(r)}_{i-1/2} = \tilde{d}_r\left(1+\frac{\tau}{\epsilon + {}^{R}\beta^{(r)}_{i-1/2}}\right) \qquad (58)
$$

with $\tau = (u_{i+1}-2u_i+u_{i-1})^2$. Another choice leads to the WENO3-P, which seems quite well behaved $\tau = |\frac{\beta^{(0)}+\beta^{(1)}}{2} - \frac{1}{4}(u_{i-1}-u_{i+1})^2| = \frac{3}{12}(u_{i+1}-2u_i+u_{i-1})^2$.

2. Fifth order WENO (k=3)

Let us write explicitly the procedure for the fifth order WENO, obtained with $k = 3$.

- The three reconstructed values ${}^{L}u^{(r)}_{i+1/2}$ and ${}^{R}u^{(r)}_{i-1/2}$ of 3$^{\rm rd}$ order accuracy,

$$\begin{aligned}
{}^{L}u^{(0)}_{i+1/2} &= \frac{2}{6}u_i + \frac{5}{6}u_{i+1} - \frac{1}{6}u_{i+2} \equiv A^{(0)}(u_i, u_{i+1}, u_{i+2}) \\
{}^{L}u^{(1)}_{i+1/2} &= -\frac{1}{6}u_{i-1} + \frac{5}{6}u_i + \frac{2}{6}u_{i+1} \equiv A^{(1)}(u_{i-1}, u_i, u_{i+1}) \\
{}^{L}u^{(2)}_{i+1/2} &= \frac{2}{6}u_{i-2} - \frac{7}{6}u_{i-1} + \frac{11}{6}u_i \equiv A^{(2)}(u_{i-2}, u_{i-1}, u_i) \\
{}^{R}u^{(0)}_{i-1/2} &= \frac{2}{6}u_{i+2} - \frac{7}{6}u_{i+1} + \frac{11}{6}u_i \equiv A^{(2)}(u_{i+2}, u_{i+1}, u_i) \\
{}^{R}u^{(1)}_{i-1/2} &= -\frac{1}{6}u_{i+1} + \frac{5}{6}u_i + \frac{2}{6}u_{i-1} \equiv A^{(1)}(u_{i+1}, u_i, u_{i-1}) \\
{}^{R}u^{(2)}_{i-1/2} &= \frac{2}{6}u_i + \frac{5}{6}u_{i-1} - \frac{1}{6}u_{i-2} \equiv A^{(0)}(u_i, u_{i-1}, u_{i-2})
\end{aligned} \tag{59}$$

  The ${}^{L}u^{(r)}_{i-1/2}$ and ${}^{R}u^{(r)}_{i+1/2}$ can be obtained by substituting i by $i \pm 1$ in the previous expressions.

- Find the smooth indicators ${}^{L}\beta^{(r)}_{i+1/2}$ and ${}^{R}\beta^{(r)}_{i+1/2}$

$$\begin{aligned}
{}^{L}\beta^{(0)}_{i+1/2} &= \frac{13}{12}(u_i - 2u_{i+1} + u_{i+2})^2 + \frac{1}{4}(3u_i - 4u_{i+1} + u_{i+2})^2 \equiv B^{(0)}(u_i, u_{i+1}, u_{i+2}) \\
{}^{L}\beta^{(1)}_{i+1/2} &= \frac{13}{12}(u_{i-1} - 2u_i + u_{i+1})^2 + \frac{1}{4}(u_{i-1} - u_{i+1})^2 \equiv B^{(1)}(u_{i-1}, u_i, u_{i+1}) \\
{}^{L}\beta^{(2)}_{i+1/2} &= \frac{13}{12}(u_{i-2} - 2u_{i-1} + u_i)^2 + \frac{1}{4}(u_{i-2} - 4u_{i-1} + 3u_i)^2 \equiv B^{(2)}(u_{i-2}, u_{i-1}, u_i) \\
{}^{R}\beta^{(0)}_{i-1/2} &= \frac{13}{12}(u_{i+2} - 2u_{i+1} + u_i)^2 + \frac{1}{4}(u_{i+2} - 4u_{i+1} + 3u_i)^2 \equiv B^{(2)}(u_{i+2}, u_{i+1}, u_i) \\
{}^{R}\beta^{(1)}_{i-1/2} &= \frac{13}{12}(u_{i+1} - 2u_i + u_{i-1})^2 + \frac{1}{4}(u_{i+1} - u_{i-1})^2 \equiv B^{(1)}(u_{i+1}, u_i, u_{i-1}) \\
{}^{R}\beta^{(2)}_{i-1/2} &= \frac{13}{12}(u_i - 2u_{i-1} + u_{i-2})^2 + \frac{1}{4}(3u_i - 4u_{i-1} + u_{i-2})^2 \equiv B^{(0)}(u_i, u_{i-1}, u_{i-2})
\end{aligned} \tag{60}$$

  Again, ${}^{L}\beta^{(r)}_{i-1/2}$ and ${}^{R}\beta^{(r)}_{i+1/2}$ can be obtained by substituting i by $i \pm 1$ in the previous expressions.

- Find the 5$^{\rm th}$ order reconstruction

$$\begin{aligned}
u^{L}_{i+1/2} &= \omega^{(0)}_{i+1/2}\,{}^{L}u^{(0)}_{i+1/2} + \omega^{(1)}_{i+1/2}\,{}^{L}u^{(1)}_{i+1/2} + \omega^{(2)}_{i+1/2}\,{}^{L}u^{(2)}_{i+1/2} \\
u^{R}_{i-1/2} &= \tilde{\omega}^{(0)}_{i-1/2}\,{}^{R}u^{(0)}_{i-1/2} + \tilde{\omega}^{(1)}_{i-1/2}\,{}^{R}u^{(1)}_{i-1/2} + \tilde{\omega}^{(2)}_{i-1/2}\,{}^{R}u^{(2)}_{i-1/2}
\end{aligned} \tag{61}$$

  with weights $\omega^{(r)}_{i+1/2}$ and $\tilde{\omega}^{(r)}_{i+1/2}$ constructed by using the generic formulas

$$\omega^{(r)}_{i+1/2} = \frac{\alpha^{(r)}_{i+1/2}}{\sum_{s=0}^{k-1}\alpha^{(s)}_{i+1/2}} \quad , \quad \tilde{\omega}^{(r)}_{i-1/2} = \frac{\tilde{\alpha}^{(r)}_{i-1/2}}{\sum_{s=0}^{k-1}\tilde{\alpha}^{(s)}_{i-1/2}} \tag{62}$$

  by using

$$\begin{aligned}
\alpha^{(0)}_{i+1/2} &= \frac{3/10}{(\epsilon + {}^{L}\beta^{(0)}_{i+1/2})^2} \quad , \quad \alpha^{(1)}_{i+1/2} = \frac{6/10}{(\epsilon + {}^{L}\beta^{(1)}_{i+1/2})^2} \quad , \quad \alpha^{(2)}_{i+1/2} = \frac{1/10}{(\epsilon + {}^{L}\beta^{(2)}_{i+1/2})^2} \\
\tilde{\alpha}^{(0)}_{i-1/2} &= \frac{1/10}{(\epsilon + {}^{R}\beta^{(0)}_{i-1/2})^2} \quad , \quad \tilde{\alpha}^{(1)}_{i-1/2} = \frac{6/10}{(\epsilon + {}^{R}\beta^{(1)}_{i-1/2})^2} \quad , \quad \tilde{\alpha}^{(2)}_{i-1/2} = \frac{3/10}{(\epsilon + {}^{R}\beta^{(2)}_{i-1/2})^2}
\end{aligned} \tag{63}$$

where ϵ is usually set to a very small number such that the expected convergence rate is achieved when $\epsilon = \Delta x^2$. Notice that we need also the reconstructed values from the other cells

$$\begin{aligned} u^L_{i-1/2} &= \omega^{(0)}_{i-1/2}\, {}^L u^{(0)}_{i-1/2} + \omega^{(1)}_{i-1/2}\, {}^L u^{(1)}_{i-1/2} + \omega^{(2)}_{i-1/2}\, {}^L u^{(2)}_{i-1/2} \\ u^R_{i+1/2} &= \tilde{\omega}^{(0)}_{i+1/2}\, {}^R u^{(0)}_{i+1/2} + \tilde{\omega}^{(1)}_{i+1/2}\, {}^R u^{(1)}_{i+1/2} + \tilde{\omega}^{(2)}_{i+1/2}\, {}^R u^{(2)}_{i+1/2} \end{aligned} \tag{64}$$

by substituting i by $i \pm 1$ in the previous expressions.

Lately there have been some proposals of a WENO-Z where the weights are changed by using

$$\alpha^{(r)}_{i+1/2} = d_r \left(1 + \left[\frac{{}^L\tau_{i+1/2}}{\epsilon + {}^L\beta^{(r)}_{i+1/2}} \right]^q \right) \quad , \quad \tilde{\alpha}^{(r)}_{i-1/2} = \tilde{d}_r \left(1 + \left[\frac{{}^R\tau_{i-1/2}}{\epsilon + {}^R\beta^{(r)}_{i-1/2}} \right]^q \right) \tag{65}$$

where $q = 1, 2$ and with ${}^L\tau_{i+1/2} = |{}^L\beta^{(0)}_{i+1/2} - {}^L\beta^{(2)}_{i+1/2}|$ and ${}^R\tau_{i-1/2} = |{}^R\beta^{(0)}_{i-1/2} - {}^R\beta^{(2)}_{i-1/2}|$. The scheme becomes more dissipative when the parameter q is increased. WENO-Z is 4^{th}-order near simple smooth critical points (i.e., where $u'_j = 0$) for $q = 1$ and attains the designed 5^{th}-order for $q = 2$, at the price of being more dissipative. The parameter ϵ is usually set to a very small number and the expected convergence rate is achieved if $\epsilon = \Delta x^4$.

An additional improvement was achieved along this lines with teh WENO-Z+ (Acker 2006 an improved WENOZ scheme)

$$\alpha^{(r)}_{i+1/2} = d_r \left(1 + \left[\frac{\epsilon + {}^L\tau_{i+1/2}}{\epsilon + {}^L\beta^{(r)}_{i+1/2}+} \right]^q + \lambda \left[\frac{\epsilon + {}^L\beta^{(r)}_{i+1/2}}{\epsilon + {}^L\tau_{i+1/2}} \right] \right) \ , \tag{66}$$

$$\tilde{\alpha}^{(r)}_{i-1/2} = \tilde{d}_r \left(1 + \left[\frac{\epsilon + {}^R\tau_{i-1/2}}{\epsilon + {}^R\beta^{(r)}_{i-1/2}} \right]^q + \lambda \left[\frac{\epsilon + {}^R\beta^{(r)}_{i-1/2}}{\epsilon + {}^R\tau_{i-1/2}} \right] \right) \tag{67}$$

where $q = 1, 2$ and $\lambda = \Delta x^{2/3}$. It is recommended to set $\epsilon = 10^{-40} < \Delta x^4$ to avoid spurious oscillations. For the same reason, it is preferred the more dissipative choice $q = 2$.

V. PARTICLES METHODS AND SPH

A. Basics of particle methods

Particle methods is an interpolation method that allow us to write down any field as a function of its values computed in any finite set of arbitrary points (i.e., the particles). Generically, any field $A(\mathbf{r})$ can be interpolated by using the integral formula

$$< A(\mathbf{r}) >= \int A(\mathbf{r}')\, W(\mathbf{r} - \mathbf{r}', h) \mathrm{d}\mathbf{r}' \ , \tag{68}$$

where the integration is performed in all the space coordinates and W is the interpolation *kernel*, which must satisfy two basic conditions,

$$\int W(\mathbf{r} - \mathbf{r}', h) \mathrm{d}\mathbf{r}' = 1 \qquad , \qquad \lim_{h \to 0} W(\mathbf{r} - \mathbf{r}', h) = \delta(\mathbf{r} - \mathbf{r}') \ . \tag{69}$$

Therefore, the interpolation formula Eq. (68) is approximated by the following summation over the neighboring particles:

$$< A(\mathbf{r}) >= \sum_b \Delta V_b\, A(\mathbf{r}_b)\, W(\mathbf{r} - \mathbf{r}_b, h), \tag{70}$$

where summation goes over all the particles b, which have a position $\mathbf{r}_b$, velocity $\mathbf{v}_b$ and an associated volume ΔV_b.

One of the main advantages of particle methods is that we can built a differentiable interpolation of any field by using its values on the particles and a differentiable analytical *kernel*. Consequently, derivatives of the field can

be obtained without using finite differences on a grid. For instance, in order to compute ∇A, we can just derivate Eq. (68), namely

$$< \nabla A(\mathbf{r}) >= \sum_b \Delta V_b \, A(\mathbf{r}_b) \, \nabla W(\mathbf{r} - \mathbf{r}_b, h). \tag{71}$$

The previous expressions can be used to compute the field and its gradient at the a, namely

$$< A_a > \; = \; \sum_b \Delta V_b \, A_b \, W_{ab}, \tag{72}$$

$$< \nabla A_a > \; = \; \sum_b \Delta V_b \, A_b \nabla_a W_{ab} \tag{73}$$

where we have simplified the notation by defining $A_b \equiv A(\mathbf{r}_b)$ and $W_{ab} \equiv W(\mathbf{r}_a - \mathbf{r}_b, h)$. If the *kernel* is radially symmetric, then $W(\mathbf{r}_a - \mathbf{r}_b, h) = W(\mathbf{r}_b - \mathbf{r}_a, h) = W(|\mathbf{r}_b - \mathbf{r}_a|, h)$, that is, $W_{ab} = W_{ba}$.

B. Different first order derivative operators

One can use different relations to write down

$$\nabla A \; = \; \frac{1}{\Phi} \left(\nabla(\Phi A) - A \nabla \Phi \right) \, , \tag{74}$$

$$\nabla A \; = \; \Phi \left[\frac{A}{\Phi^2} \nabla(\Phi) + \nabla \left(\frac{A}{\Phi} \right) \right] \, , \tag{75}$$

being Φ any differentiable function. The particle approximation of these derivatives would be

$$< \nabla A_a > \; = \; \sum_b \Delta V_b \frac{\Phi_b}{\Phi_a} (A_b - A_a) \nabla_a W_{ab} \tag{76}$$

$$< \nabla A_a > \; = \; \sum_b \Delta V_b \left[\frac{\Phi_a}{\Phi_b} A_b + \frac{\Phi_b}{\Phi_a} A_a \right] \nabla_a W_{ab} \tag{77}$$

Typical choices are $\Phi = 1$ and $\Phi = \rho$.

C. Normalization issues

Let us restrict this discussion to the one-dimensional case. As it was discussed before, any field $A(\mathbf{r})$ can be evaluated at the position $\mathbf{r} = \mathbf{r}_a$ by using the neighboring particles at located at $\mathbf{r}_b$, conveniently weighted with a Kernel $W_{ab} \equiv W(\mathbf{r}_a - \mathbf{r}_b, h)$ that depends on the smoothing length h and their separation. Notice however that the Kernel must be normalized, and that even then this produces inconsistencies near the boundaries. One way to prevent this problem is by a suitable normalization of the particle formulas. For instance, the averaged value of the function can be normalized as:

$$< A_a >= \frac{\sum_{b=1}^{N} \Delta V_b \, A_b W_{ab}}{\sum_{b=1}^{N} \Delta V_b \, W_{ab}} \tag{78}$$

where the volume ΔV_b associated to each particle. If the density is an evolved field, the associated volume can be estimated as $\Delta V_b = m_b / \rho_b$, where m_b is the particle mass and ρ_b its density.

The first derivative can be estimated in a similar way,

$$< \nabla A(\mathbf{r}) >= \frac{\sum_{b=1}^{N} \Delta V_b \, \left[A_b - A(\mathbf{r}) \right] \nabla W(|\mathbf{r} - \mathbf{r}_b|)}{\sum_{b=1}^{N} \Delta V_b \, \left[\mathbf{r}_b - \mathbf{r} \right] \nabla W(|\mathbf{r} - \mathbf{r}_b|)} \tag{79}$$

which satisfies that it is zero for a constant field and constant for a linear one. For instance, the x-component would be

$$< \partial_x A >_a = \frac{\sum_{b=1}^{N} \Delta V_b \, \left[A_b - A_a \right] (x_b - x_a) / r_{ab} \, W'(q)}{\sum_{b=1}^{N} \Delta V_b \, W'(q) \, r_{ab} / D} \tag{80}$$

We can use a symmetric operator for the first derivative, such that it can also be estimated as,

$$< \nabla A(\mathbf{r}) >= \frac{\sum_{b=1}^{N} \Delta V_b \; [A_b + A(\mathbf{r})] \, \nabla W(|\mathbf{r} - \mathbf{r}_b|)}{\sum_{b=1}^{N} \Delta V_b \; [\mathbf{r}_b - \mathbf{r}] \, \nabla W(|\mathbf{r} - \mathbf{r}_b|)} \tag{81}$$

which satisfies that it is zero for a constant field and constant for a linear one. For instance, the x-component would be

$$< \partial_x A >_a = \frac{\sum_{b=1}^{N} \Delta V_b \; [A_b + A_a] \, (x_b - x_a)/r_{ab} \; W'(q)}{\sum_{b=1}^{N} \Delta V_b \, W'(q) \; r_{ab}/D} \tag{82}$$

It is more convenient to write the argument of the Kernel in term of the dimensionless variable $q \equiv |\mathbf{r} - \mathbf{r}_b|/h = r_{ab}/h$. This introduces no changes on the calculation of $< f(\mathbf{r}) >$, just changing $W(|\mathbf{r} - \mathbf{r}_b|) \to W(q)/h^D$ because of the later normalization. For the derivative it just add a sign,

$$\partial_x W(q) = \frac{\partial W(q)}{\partial q} \, \frac{\partial q}{\partial r_{ab}} \, \frac{\partial r_{ab}}{\partial x} = W'(q) \, \frac{1}{h} \, \frac{(x - x_b)}{r_{ab}} \tag{83}$$

This means that the normalization factor can be written as

$$[\mathbf{r}_b - \mathbf{r}] \, \nabla W(|\mathbf{r} - \mathbf{r}_b|) = [\mathbf{r}_b - \mathbf{r}] \, \frac{\nabla W(q)}{h} = (\mathbf{r}_b - \mathbf{r}) \frac{(\mathbf{r} - \mathbf{r_b})}{r_{ab}} \frac{W'(q)}{h} = -\frac{W'(q)}{h} r_{ab}$$

Second derivative are more tricky and difficult to compute in the general case. However, one can found a simplified formula for the Laplacian operator

$$< \nabla^2 A >_a = \frac{\sum_{b=1}^{N} \Delta V_b \; [A_b - A_a] \, W'(q)/r_{ab}}{\sum_{b=1}^{N} \Delta V_b \, W'(q) \; r_{ab}/(2 \; D)} \tag{84}$$

The implementation of specific second derivatives is not as straightforward, but one can use an approximated formula which reduces to the Laplacian in the right limit

$$< \nabla A(\mathbf{r}) >= \frac{\sum_{b=1}^{N} \Delta V_b \; [A_b - A(\mathbf{r})] \left[4(\mathbf{r}_b - \mathbf{r})/|\mathbf{r} - \mathbf{r}_b|^2 - \delta \right] \nabla W(|\mathbf{r} - \mathbf{r}_b|)}{\sum_{b=1}^{N} \Delta V_b \; [\mathbf{r}_b - \mathbf{r}] \, \nabla W(|\mathbf{r} - \mathbf{r}_b|)} \tag{85}$$

so that the explicit expressions would be

$$< \partial_{xx} A >_a \; = \; \frac{\sum_{b=1}^{N} \Delta V_b \; [A_b - A_a] \, W'(q)/r_{ab} \; \left[4(x_b - x_a)^2/r_{ab}^2 - 1 \right]}{\sum_{b=1}^{N} \Delta V_b \, W'(q) \; r_{ab}/D} \tag{86}$$

$$< \partial_{yy} A >_a \; = \; \frac{\sum_{b=1}^{N} \Delta V_b \; [A_b - A_a] \, W'(q)/r_{ab} \; \left[4(y_b - y_a)^2/r_{ab}^2 - 1 \right]}{\sum_{b=1}^{N} \Delta V_b \, W'(q) \; r_{ab}/D} \tag{87}$$

$$< \partial_{xy} A >_a \; = \; \frac{\sum_{b=1}^{N} \Delta V_b \; [A_b - A_a] \, W'(q)/r_{ab} \; \left[4(x_b - x_a)(y_b - y_a)/r_{ab}^2 \right]}{\sum_{b=1}^{N} \Delta V_b \, W'(q) \; r_{ab}/D}$$

For the 3D case one should only change the factor 4 to a factor 5 in the expressions above, namely

$$\left[\frac{4(x_b - x_a)(y_b - y_a)}{r_{ab}^2} \right] \to \left[\frac{5(x_b - x_a)(y_b - y_a)}{r_{ab}^2} \right] \tag{88}$$

D. Time derivatives

Particle methods is included in the semi-Lagrangian type, in the sense that time derivatives D_t are performed by following the particles instead of the Eulerian fixed-position time derivative ∂_t, but not for the spatial derivatives. Therefore, we need to convert the standard Eulerian time derivative into a Lagrangian one, namely $\partial_t \to D_t - v^k \partial_k$. Let us consider a generic equation for the field A, namely

$$\partial_t A = L(A, \partial_A) \tag{89}$$

where L is the rhs operator which can involve fields and their derivatives. Therefore, this equation translates into

$$D_t A - v^k \partial_k A = L(A, \partial_A) \tag{90}$$

For generic systems one should evaluate these terms separately at the position of the particle i, since each one has a different normalization, namely

$$D_t A_i - < v^k >_i < \partial_k A >_i = < L(A, \partial_A) >_i \tag{91}$$

where $< v^k >_i$ can be computed in different ways. For instance, you could use the interpolation value with the Kernel. Generically, it will be just the particle velocity $v_i^k = dx_i^k/dt$.

However, this equations can be simplified if the density and velocity are evolved fields. The mass-conservation can be written as

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0 \rightarrow D_t \rho + \rho \partial_i (v^i) = 0 \tag{92}$$

which can be easily discretized as

$$\frac{D\rho_a}{Dt} + \rho_a \sum_b \frac{m_b}{\rho_b} (\mathbf{v}_b - \mathbf{v}_a) \cdot \nabla_a W_{ab} = 0 \, . \tag{93}$$

that has the additional advantage that is divergence free when the speed is constant.

By using the continuity equation one can rearrange the advection terms by considering the specific field $\hat{A} \equiv A/\rho$. Expanding the derivative and using the continuity equation, one can easily show that

$$\partial_t \hat{A} + v^j \partial_j \hat{A} = D_t \hat{A} = \frac{1}{\rho} \left[L + \partial_k \left(\rho \hat{A} v^k \right) \right] \quad . \tag{94}$$

which can be discretized as

$$D_t \hat{A}_a = \frac{1}{\rho_a} < L_a > + \frac{1}{\rho_a} \sum_b \frac{m_b}{\rho_b} \left(\rho_b \hat{A}_b \mathbf{v}_b - \rho_a \hat{A}_a \mathbf{v}_a \right) \nabla_a W_{ab} \, . \tag{95}$$

Using the same notation as before, the field $\hat{A}$ and the derivative of the flux $\rho \hat{A} v^k$ would be discretized with the following normalized expressions

$$< \hat{A}_i > = \frac{\sum_{j=1}^N \hat{A}_j W(x_i - x_j) \Delta V_j}{\sum_{j=1}^N W(x_i - x_j) \Delta V_j} \tag{96}$$

where $\Delta V_j = m_j/\rho_j$, and

$$< \partial_x \left(\rho \hat{A} v^k \right) >_i = \frac{\sum_{j=1}^N [\rho_j \hat{A}_j v_j^k - \rho_i \hat{A}_i v_i^k] \partial_{x_i} W(x_i - x_j) \Delta V_j}{\sum_{j=1}^N [x_j - x_i] \partial_{x_i} W(x_i - x_j) \Delta V_j} \quad . \tag{97}$$

If the original equations can be written as a system of balance law then these equations can be further simplified by merging all the fluxes. Since it is really very problem dependent, we will not discuss in more detail those options.

E. Particle trajectory correction

XSPH is a variant, proposed by Monaghan (1994), corrects particle velocity assuring more ordered flow and prevents penetration between continua when high speed or impact occur:

$$\frac{d\mathbf{x}_a}{dt} = \mathbf{v}_a + \epsilon \sum_b \frac{m_b}{\rho_{ab}} (\mathbf{v}_b - \mathbf{v}_a) W_{ab} \tag{98}$$

where $\rho_{ab} \equiv (\rho_a + \rho_b)/2$.

F. Particle-mesh interaction

In particle and mesh simulations, it is posible to transfer information between fields held in particles and fields in cells. Currently, linear lagrangian interpolation method is implemented to interpolate fields from mesh into a particle, and kernel based interpolation is implemented to interpolate from particles in a cell.

VI. THE AMR ALGORITHM

The AMR algorithm is constructed by using basic blocks (i.e.,routines and functions) provided by SAMRAI.

A. The algorithm

There are two refinement tagging strategies provided by SAMRAI integrated in Simflowny:

- **Fixed Mesh Refinement (FMR).** The user specifies statically a set of boxes where the refinement is located. Every level allows different boxes as long as they are nested in coarsen level boxes.
- **Adaptive Mesh Refinement (AMR).** The user sets a criteria used to calculate dynamically the cells to be refined. There are two criteria currently implemented by Simflowny:
  - **Gradient.** Checked on a field, when its gradient surpass certain value, the cell must be tagged for refinement.
  - **Function.** For a certain field (which can be a function) when its value surpass certain threshold the cell is refined.

Notice that both kind of tagging strategies can be combined in the same simulation.

As the simulation evolves, the AMR tagging criteria may have changed, needing new refinement areas or disposing older ones. This re-meshing procedure is periodically run and is parameterizable for every simulation.

If a new refinement box is added when the simulation has already started, the number of cells inside the box are increased respecting the coarser level and need to have a value, which is interpolated from the coarser level. That process is called **prolongation** and is further explained in the next subsection.

The algorithm skeleton is as follows:

```
initialization
refinement_tagging
while not simulation end do
    for all Level l do
        for all Runge Kutta step do
            calculate_rhs (derivatives + dissipation)
            integrate_time
            if last Runge Kutta step then
                restrict from l to l-1
                level synchronization(l - 1)
            end if
            level synchronization(l)
            prolong from l-1 to l (prepare ghost zones)
            calculate_physical_boundaries
        end for
        if has to regrid(l) then
            refinement tagging(l)
        end if
    end for
end while
```

The restriction procedure is straightforward, it consists on copying the data from fine levels to coarser ones for the overlapping cells.

The previous algorithm is similar for every level. However, prolongation, restriction and number of executions depend whether subcycling is active. Currently, there are three available time integrations in Simflowny, without subcycling, tappering or Berger-Oliger.

Figures 3 and 4 show the behaviour of synchronization and steps for a simulation without subcycling. Every level advance the same value of Δt. The main advantage is that prolongation is straightforward, that is because time at every substep is the same for all levels. The main disadvantage is the value of Δt, which must fit to Courant constraint in the finest level. Therefore, coarser levels run slower than they could.

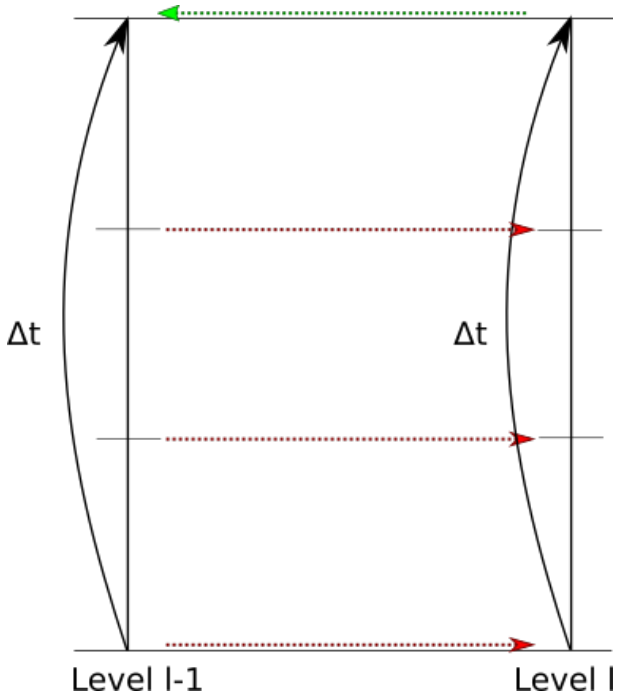


FIG. 3: Synchronization between two levels without subcycling using a Runge-Kutta with three substeps. (Prolongation in red, restriction in green)

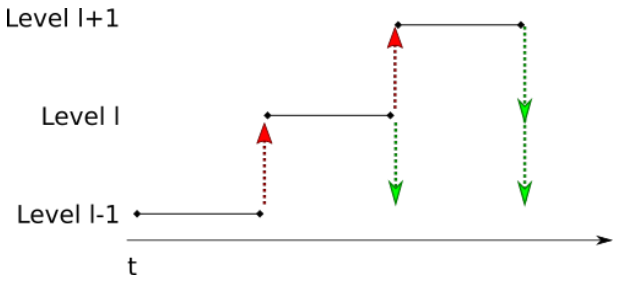


FIG. 4: Time diagram for synchronization between three levels without subcycling. Each segment represents a complete step. (Prolongation in red, restriction in green)

When using subcycling time integration with tappering, each level in the hierarchy run the longer Δt it can. In the other hand, the sublevels must repeat the substep to reach parent Δt. Additionally, the tappering algorithm artificially increases the number of ghost nodes to reduce the number of synchronizations in substeps. In Figure 6, there is only one prolongation at the beginning of the first step in the finer level. Nevertheless, this prolongation is expensive due to the extra ghost zone.

Figure 7 shows the intercalation of level steps in a three level simulation over time. Notice the variable length of each step in different levels.

A scheme of the different steps (i.e., prolongation, restriction and time-integration) is shown in Figure 5 for the different sub-cycling options. In the case without sub-cycling, each level advance the same value of Δt. Prolongation is then straightforward, mainly because time at every sub-step is the same for all levels. As we mentioned before, the main disadvantage of this approach is that the value of Δt must fit the CFL constraint in the finest level. Consequently, coarse levels run slower than they could. This drawback is not present when there is sub-cycling in the time integration. With the tappering strategy, each level in the hierarchy run the longest Δt allowed by the CFL

condition. As it is typical for this approach, the children levels need to perform at least two time-steps to reach their parent Δt. The main drawback of the tappering algorithm is that, in order to avoid prolongations from the coarse grid during the RK sub-steps, it artificially increases the number of grids on the ghost zones. In Figure 5 there is only one prolongation at the beginning of the first step in the finer level and one restriction at the very end. Nevertheless, this prolongation is expensive due to the extra ghost zone.

Finally, the last option is Berger-Oliger time integration, either the standard one or the new one ÅWhen we first introduced it we did not mention it was new, and here is mentioned as if known. Shouldn't we say explicitly we have developed a new BO variation? without order reduction. This case is more complex, since it requires a larger number of prolongations steps, as it is shown in Figure 5. Due to the time misalignment on the RK sub-steps, a time interpolation must be performed in the coarser level, which might lead to a reduction of the accuracy if it is not performed carefully. Despite those disadvantages, this method is faster and more efficient than tappering due to the smaller ghost zones. More details about our preferred choice, the Berger-Oliger without order reduction, can be found in the Appendix.

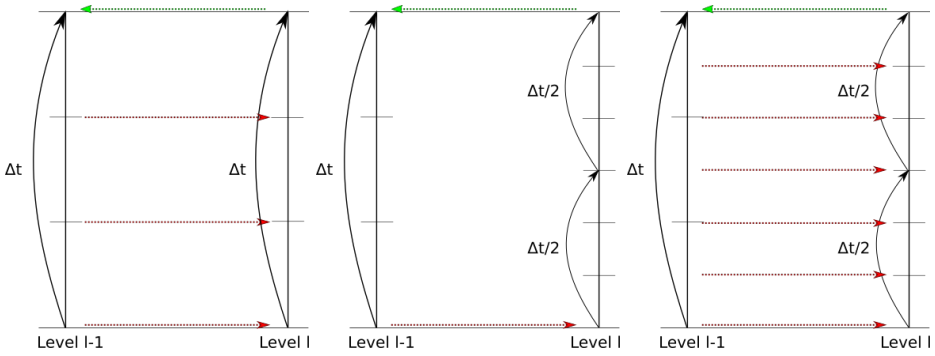


FIG. 5: Synchronization between two levels with no sub-cycling (left), tappering (middle) and Berger-Oliger (right), using a Runge-Kutta with three substeps. Prolongation between levels is denoted with red arrows, while that restriction is denoted with green ones.

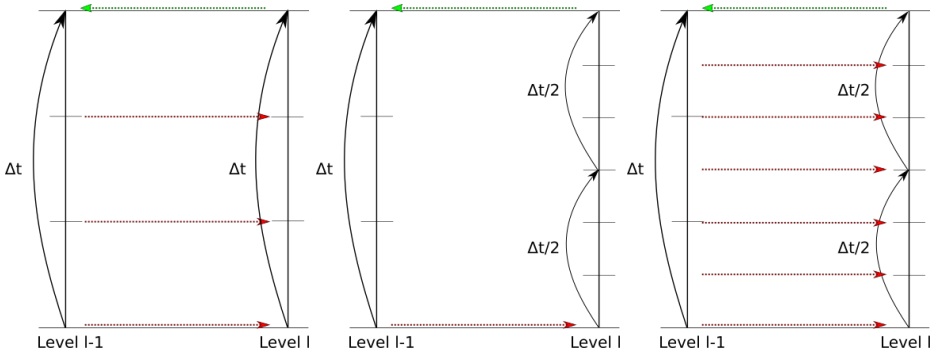


FIG. 6: Synchronization between two levels with tappering subcycling using a Runge-Kutta with three substeps. (Prolongation in red, restriction in green)

Finally, the last option is Berger-Oliger time integration. This case requires more complex and more number of prolongations as shows in Figure 8. Due to the misalignment of substep variables between levels, a time interpolation must be performed in the coarser level. Despite those disadvantages, this method is faster than tappering due to the smaller ghost zone.

Berger-Oliger time diagram (Figure 9) is similar to Tappering, but needing mode prolongation synchronizations.

B. The prolongation

The prolongation consists on filling points outside the refinement region with the same fine resolution, by interpolating from the coarse grid. To avoid spoiling the accuracy of the spatial discretization we will usually use higher order interpolation. The simplest option is to use a Lagrange interpolator function. Given a point defined by its

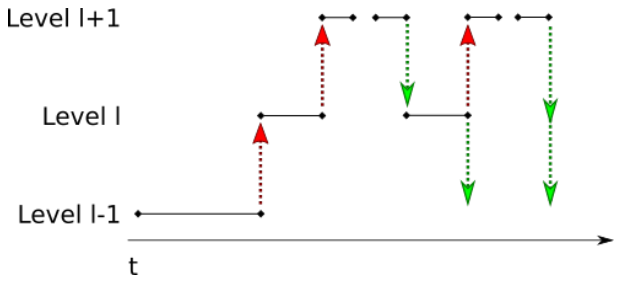


FIG. 7: Time diagram for synchronization between three levels with tappering subcycling.(Prolongation in red, restriction in green)

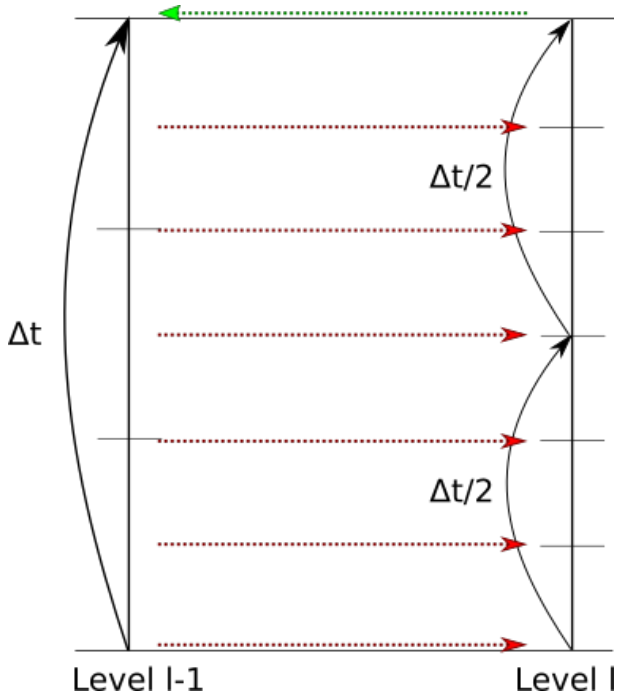


FIG. 8: Synchronization between two levels with Berger-Oliger subcycling using a Runge-Kutta with three substeps. (Prolongation in red, restriction in green)

position x_i and its value y_i, we can construct a Lagrangian polynomial function of order k passing through a $k+1$ set of points $\{(x_1, y_1), (x_2, y_2), ...(x_k, y_k), (x_{k+1}, y_{k+1})\}$, namely

$$p(x) = \sum_{j=1}^{k+1} y_j l_j(x) \quad , \quad l_j(x) = \prod_{\substack{m=1 \\ m \neq j}}^{k+1} \frac{x - x_m}{x_j - x_m} \tag{99}$$

where x is the point position in which the value is interpolated.

To construct a Lagrangian polynomial of fifth order six points are required, three at each side of the point to interpolate. The classical Lagrangian polynomial interpolation for six points is as follows:

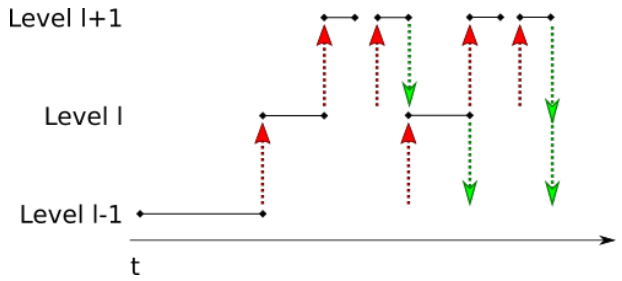


FIG. 9: Time diagram for synchronization between three levels with Berger-Oliger subcycling.(Prolongation in red, restriction in green)

$$((x-x_2)*(x-x_3)*(x-x_4)*(x-x_5)*(x-x_6))/((x_1-x_2)*(x_1-x_3)*(x_1-x_4)*(x_1-x_5)*(x_1-x_6))*(y_1 \quad (100)$$
$$+((x-x_1)*(x-x_3)*(x-x_4)*(x-x_5)*(x-x_6))/((x_2-x_1)*(x_2-x_3)*(x_2-x_4)*(x_2-x_5)*(x_2-x_6))*(y_2 \quad (101)$$
$$+((x-x_1)*(x-x_2)*(x-x_4)*(x-x_5)*(x-x_6))/((x_3-x_1)*(x_3-x_2)*(x_3-x_4)*(x_3-x_5)*(x_3-x_6))*(y_3 \quad (102)$$
$$+((x-x_1)*(x-x_2)*(x-x_3)*(x-x_5)*(x-x_6))/((x_4-x_1)*(x_4-x_2)*(x_4-x_3)*(x_4-x_5)*(x_4-x_6))*(y_4 \quad (103)$$
$$+((x-x_1)*(x-x_2)*(x-x_3)*(x-x_4)*(x-x_6))/((x_5-x_1)*(x_5-x_2)*(x_5-x_3)*(x_5-x_4)*(x_5-x_6))*(y_5 \quad (104)$$
$$+((x-x_1)*(x-x_2)*(x-x_3)*(x-x_4)*(x-x_5))/((x_6-x_1)*(x_6-x_2)*(x_6-x_3)*(x_6-x_4)*(x_6-x_5))*(y_6 \quad (105)$$
$$(106)$$

For the case of interpolation on AMR, there are a set of characteristics that are constant and may simplify the calculation.

- The point to interpolate is always between x_3 and x_4.
- The distance between x_i points is constant (δx).
- The distance from point x to x_3 is directly related to the refinement ratio and its position in the new divided cell. $x_3 = x - \delta x/R * P$

The interpolation operator is simplified by the previous premises.

$$\begin{aligned}(1.0/120)*(\\ &(6*R^4*P-5*R^3*P^2-5*R^2*P^3+5*R*P^4-P^5)*y_{x-3}\\ &+(-60*R^4*P+80*R^3*P^2-5*R^2*P^3-20*R*P^4+5*P^5)*y_{x-2}\\ &+(120*R^5-40*R^4*P-150*R^3*P^2+50*R^2*P^3+30*R*P^4-10*P^5)*y_{x-1}\\ &+(120*R^4*P+80*R^3*P^2-70*R^2*P^3-20*R*P^4+10*P^5)*y_{x+1}\\ &+(-30*R^4*P-5*R^3*P^2+35*R^2*P^3+5*R*P^4-5*P^5)*y_{x+2}\\ &+(4*R^4*P-5*R^2*P^3+P^5)*y_{x+3}\\ )/(R^5))\end{aligned} \quad (107)$$

where R is the refinement ratio, P is the position of the node to refine respecting immediate-left coarser node. In Figures 10 and 11 there are two examples of interpolation for ratios 2 and 3. The value of P varies depending on the position of y to be interpolated.

The standard choice is $R = 2$ and $P = 1$, so that the interpolator can be further reduce to

$$p(x) = \frac{1}{256} * [150(y_{x-1} + y_{x+1}) - 25(y_{x-2} + y_{x+2}) + 3(y_{x-3} + y_{x+3}))] \quad (108)$$

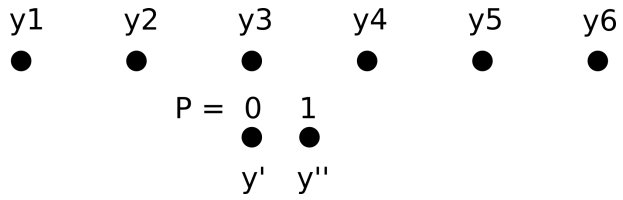


FIG. 10: Interpolation with ratio 2.

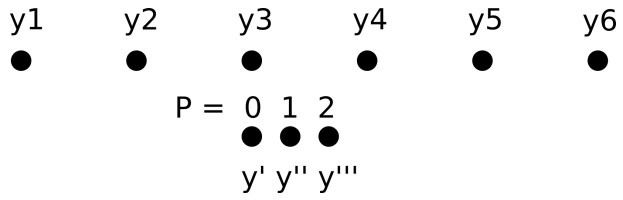


FIG. 11: Interpolation with ratio 3.

C. The subciclying in time

Regions with different resolutions can be evolved stably by using the smallest Δt (corresponding to the finest grid resolution Δx_l), such that all the grids satisfy the CFL condition. However, this is an expensive choice, since then the coarser grids are evolved with Δt much smaller than the ones allowed by their local CFL condition.

Let us consider a uniform unidimensional mesh of n points which takes $N \times Q$ operations to step Δt by using a Runge-Kutta of order q. Assume also that there is 2:1 FMR hierarchy with L levels of refinement. Therefore, to advance all the grid points to $t + \Delta t$ by using the same Δt for all the levels (i.e., which will be given by the CFL condition in the finest grid), will require a number of operations

$$Q \times \left(N + 2 \times N + 4 \times N ... + 2^{L-1} \times N\right) \approx Q \times 2^L \times N \rightarrow Q \times 2^L \times N^d \tag{109}$$

where we have generalized the result to d-dimensions.

The number of operations can be reduced significantly by using the same CFL in all the grids to advance the solution, such that $\Delta t_{l-1} = 2\Delta t_l$. Therefore, the minimum number of operations to advance all the grid points will be

$$Q \times (N + N + ... + N) = Q \times L \times N \rightarrow Q \times L \times N^d \tag{110}$$

which is much smaller than the above estimation.

If we do not evolve all the grids with the same Δt it is not clear how to evolve the interior points (i.e., from the fine grid) at each refinement boundary, since the solution is not evaluated at the same time on the coarser grid. There have been several well motivated attempts to fill in this missing information

- Berger-Oliger interpolator (BO). The solution of the coarser grid is evolved first up to $n + 1$. Then, with the information from $\{U^{n+1}, U^n, ..\}$, we can interpolate in time to calculate the solution at the required times of the Runge-Kutta of the finer grid. Spatial interpolation is also required to fill the points in the positions needed by the spatial discretization scheme. This algorithm is cheap, fast and efficient, since it requires to interpolate only in a number of points equal to the stencil of the spatial discretization scheme. The total number of operations is then

  $$Q \times L \times (N + 2S) \rightarrow Q \times L \times (N + 2S)^d \tag{111}$$

  where S is the stencil of the spatial discretization and the factor 2 comes from the two refinement boundaries present in 1D. Usually $S = 3$ for fourth order centered derivatives with sixth-order dissipation. However, the original version is second order and it actually reduces to first order for very high resolution, as we will discuss later, due to the order reduction problem.

- Tappering (T). The fine grid is extended by a number of points $2Q \times S$ on each direction perpendicular to the refinement boundary. This way, the boundary point at the second final substep of the fine grid (i.e., when it reaches U^{n+1}) is inside the numerical domain of dependence of the extended initial grid. This algorithm is

algorithm	$d=1$	$d=2$	$d=3$
No subcycling	1.02×10^5	1.02×10^7	1.02×10^9
minimum	3.2×10^3	3.2×10^5	3.2×10^7
BO/BOR	3.4×10^3	3.7×10^5	4.0×10^7
Tappering	8.3×10^3	2.1×10^6	5.6×10^8
BO + T	5.0×10^3	1.0×10^6	1.9×10^8

TABLE V: Number of operations for a fourth order RK $Q=4$ with a fourth order accurate spatial derivative $S=4$ with a grid base of $N=100$ in each direction and $L=8$ levels of refinement.

expensive because it involves extending each refinement grid by a large number of points in each direction (i.e., for instance, with a 4th order RK and 4th order space discretization it would be around 16 points on each side of the fine grid). However, it is very accurate, since it leads to an evolution without many effects from the change of grid resolutions. The total number of operations would be in this case

$$\begin{aligned} L &\times [(N+2Q\times 2S)+(N+2(Q-1)\times 2S)+...+(N+4S)] \\ &\approx Q\times N\times L+4L\times S\times Q\times(Q+1)/2 \\ &\approx Q\times L\times[N+2S\times Q\times(Q+1)]\rightarrow Q\times L\times[N+2S\times Q\times(Q+1)]^d \end{aligned} \tag{112}$$

- Tappering + Berger-Oliger (BOT) . The fine grid is extended by a number of points *stencil* $s-RK stages$, such that the boundary point at the first final substep of the fine grid (i.e., when it reaches $U^{n+1/2}$) is inside the numerical domain of dependence of the extended initial grid. Then it uses the values of the coarse grid at $U^n, U^{n+1}, ...$ to interpolate in time and repopulate again *stencil* $s-RK stages$ points to be evolved again as the first RK step. This algorithm is less expensive than the full tapering, but its accuracy is restricted by the time interpolation at $n+1/2$.

$$Q\times L\times[N+S\times Q\times(Q+1)]\rightarrow Q\times L\times[N+S\times Q\times(Q+1)]^d \tag{113}$$

- Berger-Oliger without order reduction (BOR). The solution of the coarser grid is evolved first up to $n+1$. Then, with the information from the substeps of the RK we can build an internal dense output interpolator $\{U^n, U^{(i)}, U^{n+1}\}$ of order $q=p-1$ and compute all the k_i of the fine grid. Using the standard RK formula with these k_i will cancel the error terms to lead to a final scheme of at least order q in time, and for some RK it might even reach p. This algorithm is cheap, fast, efficient and accurate, and it will be our preferred choice.

$$Q\times L\times(N+2S)^d \tag{114}$$

Notice that one can use the dense output for the interpolation required in all the variants of Berger-Oliger algorithm (i.e, BO, BOT and BOR).

Let us explain in detail the different steps of the BOR algorithm, which is the most efficient and the less known in the field. A direct Taylor expansion of the solution at $t=t^n$ leads to

$$U_{n+1}=U_n+\Delta t\,U'_n+\frac{1}{2}\Delta t^2\,U''_n+\frac{1}{6}\Delta t^3\,U'''_n+O(\Delta t^4) \tag{115}$$

By performing a similar expansion on the k_i of the RK we obtain

$$k_1 = \Delta t\,U'_n \tag{116}$$

$$k_2 = \Delta t\,U'_n+c_2\Delta t^2\,U''_n+\frac{1}{2}c_2^2\Delta t^3\,[U'''_n-f_U U''_n]+O(\Delta t^4) \tag{117}$$

$$k_3 = \Delta t\,U'_n+c_3\Delta t^2\,U''_n+\frac{1}{2}\Delta t^3\left[c_3^2U'''_n-\left(c_3^2-2\sum_{j=1}^{3}a_{3j}c_j\right)f_U U''_n\right]+O(\Delta t^4) \tag{118}$$

$$k_4 = \Delta t\,U'_n+c_4\Delta t^2\,U''_n+\Delta t^3\left[\frac{1}{2}c_4^2U'''_n-\left(\frac{1}{2}c_4^2-\sum_{j=1}^{4}a_{4j}c_j\right)f_U U''_n\right]+O(\Delta t^4) \tag{119}$$

where f_U is the Jacobian of f. Notice that one could solve now the derivatives of U in terms of k_i. However, the equations are not linearly independent and it is impossible to solve them. Instead, we will compute the derivatives here from the dense output interpolator 5. Once we have these derivatives, we can calculate the k_i corresponding to the RK steps of the fine grid, that is, by doing $\Delta t \to \Delta t/2$ in eqs(116). From there we can calculate the solution at the different RK sub-steps required for the evolution of the boundary points of the fine grid.

1. RK4 with ratio 2

Let us be more explicit and write down the steps for the standard fourth order RK:

1. we start the fist RK step of the fine grid by computing $\{U'_n, U''_n, U'''_n, f_U U''_n\}$ from the dense output interpolator as a function of $\{k_1, k_2, k_3, k_4\}$, that is, at $t = t^n$ or $\theta = 0$. The Jacobian can be obtained directly from the k_i of the coarser grid by computing $f_U U''_n = 4(k_3 - k_2)/\Delta t^3$.

2. compute $\{k_1, k_2, k_3, k_4\}$ of the fine grid by using $\Delta t_F = \Delta t/2$. For this RK$(4,4)$

$$
\begin{aligned}
k_1 &= \Delta t_F\, U'_n && (120)\\
k_2 &= \Delta t_F\, U'_n + \frac{1}{2}\Delta t_F^2\, U''_n + \frac{1}{8}\Delta t_F^3\, [U'''_n - f_U U''_n] && (121)\\
k_3 &= \Delta t_F\, U'_n + \frac{1}{2}\Delta t_F^2\, U''_n + \frac{1}{8}\Delta t_F^3\, [U'''_n + f_U U''_n] && (122)\\
k_4 &= \Delta t_F\, U'_n + \Delta t_F^2\, U''_n + \frac{1}{2}\Delta t_F^3 U'''_n && (123)
\end{aligned}
$$

3. use in each sub-step of the first RK step its intermediate value, that for our RK4 is

$$
\begin{aligned}
U^{(1)} &= U^n && (124)\\
U^{(2)} &= U^n + \frac{1}{2}k_1 && (125)\\
U^{(3)} &= U^n + \frac{1}{2}k_2 && (126)\\
U^{(4)} &= U^n + k_3 && (127)\\
U^{n+1} &= U^n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) && (128)
\end{aligned}
$$

4. we start the second RK step of the fine grid by computing $\{U'_{n+1/2}, U''_{n+1/2}, U'''_{n+1/2}, f_U U''_{n+1/2}\}$ from the dense output interpolator as a function of $\{k_1, k_2, k_3, k_4\}$, that is, at $t = t^{n+1/2}$ or $\theta = 1/2$.

5. compute $\{k_1, k_2, k_3, k_4\}$ of the fine grid by using $\Delta t_F = \Delta t/2$ and the $\{U'_{n+1/2}, U''_{n+1/2}, U'''_{n+1/2}, f_U U''_{n+1/2}\}$.

6. use in each sub-step of the second RK step its intermediate value

$$
\begin{aligned}
U^{(1)} &= U^{n+1/2} && (129)\\
U^{(2)} &= U^{n+1/2} + \frac{1}{2}k_1 && (130)\\
U^{(3)} &= U^{n+1/2} + \frac{1}{2}k_2 && (131)\\
U^{(4)} &= U^{n+1/2} + k_3 && (132)\\
U^{n+1/2} &= U^{n+1/2} + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) && (133)
\end{aligned}
$$

   where $U^{n+1/2}$ is computed again from the dense output interpolator. The second RK step finalizes at t^{n+1}.

2. RK4 with ratio R

Let us be more explicit and write down the steps for the standard fourth order RK for an arbitrary ratio R. First we define the time-step on the fine grid $\Delta t_F = \Delta t/R$. Then we start a loop over the steps on the fine grid, going from $r = 0, R-1$

1. define $t^{n+r/R} \equiv t + r\Delta t_F$ and evaluate the solution $U_{n+r/R}(t = t^{n+r/R})$ by using the dense output interpolator.
2. Computing $\{U'_{n+r/R}, U''_{n+r/R}, U'''_{n+r/R}, f_U U''_{n+r/R}\}$ from the dense output interpolator as a function of $\{k_1, k_2, k_3, k_4\}$, that is, at $t = t^{n+r/R}$ or $\theta = r/R$. The Jacobian can be obtained directly from the k_i of the coarser grid by computing $f_U U''_{n+r/R} = 4(k_3 - k_2)/\Delta t^3$.
3. compute $\{k_1, k_2, k_3, k_4\}$ of the fine grid by using its time-step Δt_F, namely

$$
\begin{aligned}
k_1 &= \Delta t_F\, U'_{n+r/R} && (134)\\
k_2 &= \Delta t_F\, U'_{n+r/R} + \frac{1}{2}\Delta t_F^2\, U''_{n+r/R} + \frac{1}{8}\Delta t_F^3 \left[U'''_{n+r/R} - f_U U''_{n+r/R}\right] && (135)\\
k_3 &= \Delta t_F\, U'_{n+r/R} + \frac{1}{2}\Delta t_F^2\, U''_{n+r/R} + \frac{1}{8}\Delta t_F^3 \left[U'''_{n+r/R} + f_U U''_{n+r/R}\right] && (136)\\
k_4 &= \Delta t_F\, U'_{n+r/R} + \Delta t_F^2\, U''_{n+r/R} + \frac{1}{2}\Delta t_F^3 U'''_{n+r/R} && (137)
\end{aligned}
$$

4. use in each sub-step of the first RK step its intermediate value, that for our RK4 is

$$
\begin{aligned}
U^{(1)} &= U_{n+r/R} && (138)\\
U^{(2)} &= U_{n+r/R} + \frac{1}{2}k_1 && (139)\\
U^{(3)} &= U_{n+r/R} + \frac{1}{2}k_2 && (140)\\
U^{(4)} &= U_{n+r/R} + k_3 && (141)\\
U^{n+\frac{r+1}{R}} &= U_{n+r/R} + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) && (142)
\end{aligned}
$$

The final RK step finalizes at t^{n+1}.

3. RK3 with ratio 2

The procedure for the RK-SS$(3,3)$ is very similar, but simpler since there is one stage less:

1. we start the fist RK step of the fine grid by computing $\{U'_n, U''_n\}$ from the dense output interpolator as a function of $\{k_1, k_2, k_3\}$, that is, at $t = t^n$ or $\theta = 0$.
2. compute $\{k_1, k_2, k_3\}$ of the fine grid by using $\Delta t_F = \Delta t/2$. For this RK-SSP$(3,3)$

$$
\begin{aligned}
k_1 &= \Delta t_F\, U'_n && (143)\\
k_2 &= \Delta t_F\, U'_n + \Delta t_F^2\, U''_n + O(\Delta t_F^3) && (144)\\
k_3 &= \Delta t_F\, U'_n + \frac{1}{2}\Delta t_F^2\, U''_n + O(\Delta t_F^3) && (145)
\end{aligned}
$$

3. use in each sub-step of the first RK step its intermediate value, that for our RK3-SSP is

$$
\begin{aligned}
U^{(1)} &= U^n && (146)\\
U^{(2)} &= U^n + k_1 && (147)\\
U^{(3)} &= U^n + \frac{1}{4}k_2 + \frac{1}{4}k_3 && (148)\\
U^{n+1/2} &= U^n + \frac{1}{6}\left(k_1 + 2k_2 + 4k_3\right) && (149)
\end{aligned}
$$

4. we start the second RK step of the fine grid by computing $\{U'_{n+1/2}, U''_{n+1/2}\}$ from the dense output interpolator as a function of $\{k_1, k_2, k_3\}$, that is, at $t = t^{n+1/2}$ or $\theta = 1/2$.

5. compute $\{k_1, k_2, k_3\}$ of the fine grid by using $\Delta t_F = \Delta t/2$ and the $\{U'_{n+1/2}, U''_{n+1/2}, U'''_{n+1/2}\}$.

6. use in each sub-step of the second RK step its intermediate value

$$U^{(1)} = U^{n+1/2} \tag{150}$$
$$U^{(2)} = U^{n+1/2} + k_1 \tag{151}$$
$$U^{(3)} = U^{n+1/2} + \frac{1}{4}k_2 + \frac{1}{4}k_3 \tag{152}$$
$$U^{n+1} = U^{n+1/2} + \frac{1}{6}(k_1 + 2k_2 + 4k_3) \tag{153}$$

where $U^{n+1/2}$ is computed again from the dense output interpolator. The second RK step finalizes at t^{n+1}.

D. AMR in particles

The previous subsections refer to AMR for mesh based simulations. The challenges of running multi-resolution particle simulations are faced using the Multi Level Multi Domain approach (MLMD from now on) [11].

The MLMD architecture is composed of a collection of different domains each simulated with the full physical description and interlocked to the others through the exchange of particle information. Particles are created at initialization in each level of the grid they belong to and are bounded to their level of origin, not being allowed to transition from coarser to refined grids and vice versa. Refined particles are lost when they exit their domain of origin and new refined level particles are created from the coarse grid particle distribution at the boundary cells of the refined grids.

The algorithm of particle splitting creates a collection of particles (in a factor of a ratio parameter) displaced in space, keeping the same physical fields as parent particle *pp* but volume. Volume of children particles *cp* is evenly distributed all over the new set of created particles.

$$V_{pp} = \prod_{cp} V_{cp} \tag{154}$$

More detail and properties of this method are explained in detail in [12] (Algorithm S1).

An interesting characteristic of MLMD systems is that interaction between coarse and fine levels only occurs at fine boundary zone. Coarser particles creates new finer particles at that areas enforcing continuity, however, those finer particles (so as coarser particles) are free to evolve. Consequently, a different evolution across the levels is possible in the overlap areas simulated by different levels.

The possibility of obtaining different evolution across the levels, within the limits of boundary continuity, is considered a point of strength in MLMD, since it shows the capability of refined levels to evolve according to dynamics not accessible, because of the reduced resolution, to the coarser levels. This characteristic is one of the major differences from AMR systems, while AMR aims at consistency in the overlap area across the levels, MLMD only enforces particle continuity at grid interfaces and aims an efficient interlocking between the levels in the overlap areas.

VII. BOUNDARY CONDITIONS

Simflowny handles the boundary condition by extrapolating to get ghost points and being able to take derivatives.

A. Boundary conditions

The main part of the boundary conditions assumes that there is an outgoing radial wave with some speed v_0,

$$X = X_0 + \frac{u(r - v_0 t)}{r} \tag{155}$$

where X is any of the tensor components of the evolved variables, X_0 its value at infinity and u a spherically symmetric perturbation. Notice that $\{X_0, v_0\}$ depend on the particular variable. The time derivative can be written as

$$\partial_t X = -v^i \partial_i X - v_0 \frac{X - X_0}{r} \tag{156}$$

where $v^i = v_0 x^i/r$ and ∂_i are evaluated using centered finite differencing where possible and one-sided finite differencing elsewhere. We can also account for the non-wave art of the solutions by assuming that these parts decay with a certain power p of the radius. Given a source term $(\partial_t X)$, the corrected term $(\partial_t X)^*$ can be computed as

$$(\partial_t X)^* = \partial_t X + \left(\frac{r}{r - n^i \partial_i r}\right)^p n^i \partial_i (\partial_t X) \tag{157}$$

where n^i is the normal vector of the corresponding boundary face and we assume a second order decay $p = 2$. This new correction can be written explicitly, namely

$$\partial_t X = -v_0 \frac{x^i}{r} \partial_i X - v_0 \frac{X - X_0}{r} - \lambda_{BC} v_0^2 \left(\frac{r}{r-1}\right)^p \left(\frac{x^i x^j}{r^2} \partial_i \partial_j X + \frac{x^i}{r^2} \partial_i X - \frac{X}{r^2}\right) \tag{158}$$

VIII. TESTS AND SIMPLE APPLICATIONS

We will focus on two specific models. First, we will perform some tests with the scalar wave equation to check the accuracy and convergence of our numerical schemes, in combination with the Adaptive Mesh Refinement algorithms. Then we will use to Einstein Equations to study some simple problems like gravitational wave radiation of a single black hole and binary solitonic boson stars.

A. Wave equation

We will focus here in the simple wave equation written as a system first order in time and second order in space, namely

$$\partial_t \phi = -\Pi \tag{159}$$
$$\partial_t \Pi = -\eta^{ij} \partial_i \partial_j \phi \tag{160}$$

where $\eta^{ij} = 1$ for $i = j$ and zero otherwise. Our set up consists on a 1D channel with a domain $[-2, 8]$ and periodic boundary conditions. Our initial configuration is given by a time-symmetric pulse centered at $x = 0$, namely

$$phi_0(x) = \phi(x, t = 0) = e^{-x^2/\varrho^2} \quad , \quad \Pi = 0 \quad . \tag{161}$$

with $\varrho = 0.173$. Within this initial condition, the initial gaussian profile splits in two identical pulses propagating in opposite directions. These two pulses overlaps at the initial location after a full crossing time $t = 10$.

We evolve this problem with fourth order space differencing and sixth order Kreiss-Oliger dissipation, such that the semi-discrete problem is consistent to the continuum one to fourth accuracy in Δx. We will use two different Runge-Kutta (i.e., a 3rd order RK3 and 4th order RK4) to integrate the semi-discrete ODE, such that the fully discrete problem is accurate at third and fourth order in time respectively. In this unigrid setup it is straightforward to show that our numerical solution converges to the analytical solution $\phi(x, t) = \phi_0(x - t)/2 + \phi_0(x + t)/2$ to the expected order (i.e., third order with RK3 and fourth order with RK4) when using the resolutions $\Delta x = \{1/40, 1/80, 1/160\}$.

The problem becomes more interesting by including an additional fixed grid between $[1, 2]$ with twice the resolution of the coarse original grid. The pulse traveling to the right will cross the fine region and then interact with the one traveling to the left before returning to its initial position. The pulse traveling to the left will also cross the the refined region after the interaction. The solution at different times is displayed in Figure 12).

The convergence of the numerical scheme will depend now not only on the order of the space discretization and time integrator, but also on the choice of time refinement algorithm. We have considered here several choices here; no sub-cycling, Tappering, Berger-Oliger (BO) with an internal dense output third order interpolator Berger-Oliger without order reduction (BOR). Additionally, we have considered the case with a single transition point between the coarse and the fine grid, which is evolved by interpolating both evolution methods at the boundary point: evolution of the fine grid and evolution of ghost zone points. In Figure 13 we show the convergence rate obtained for the RK4 and RK3.

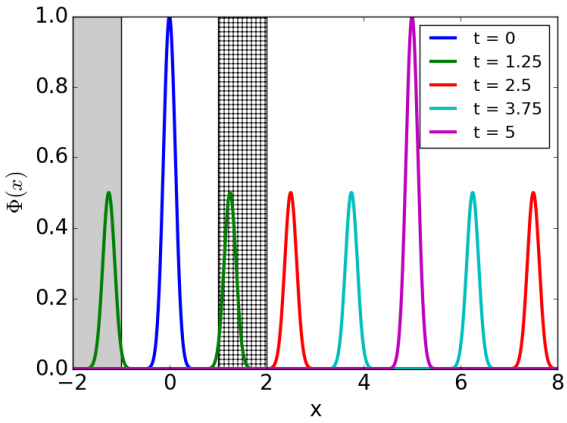


FIG. 12: Scalar wave evolution for half a crossing time. Notice that the pulse traveling to the right crosses the refined region (in grey) in the first half crossing time, while that the one traveling to the left will do it during the second half.

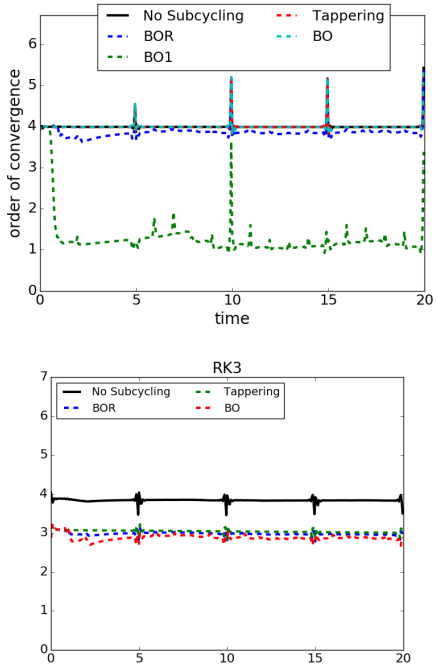


FIG. 13: Convergence order of RK4 (top panels) and RK3 (bottom panels) with transition zone (left) and without (right) by using $\Delta x = \{1/40, 1/80, 1/160\}$. Only the Tappering and the Berger-Oliger with no order reduction (BOR) achieves the expected fourth order convergence achieved without sub-cycling in time. The convergence is degraded when the original Berger-Oliger (BO) is used, even if it is third order. The BO convergence improves when using a transition zone of a single point between the coarse and the fine grid.

As it was expected, only the Tappering and the Berger-Oliger without order reduction achieves the expected convergence rate, while that the original Berger-Oliger reach a much lower convergence rate. Notice however that using the transition zone improves somehow the convergence order of the Berger-Oliger schemes.

In addition to the convergence rate we can also calculate the reflections produced by the change of grid size when the pulse traveling to the right crosses the refined grid. We will restrict to the RK4 here, and will measure these reflections as the integral in the interval $[-1, 0]$ of the norm of the scalar field. In figure 14 the reflections are displayed as a function of time, as it crosses the refined region and produces two reflected waves (i.e., one at the enter and one at the exit).

We can now study the convergence of our algorithm with fully AMR, by setting a refinement criterium such that

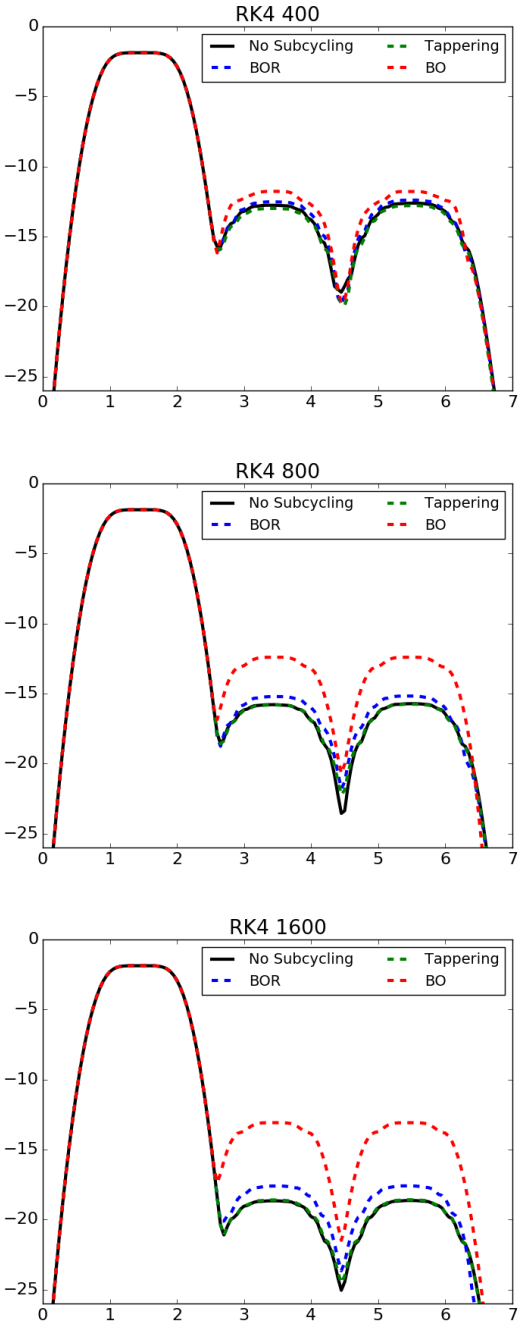


FIG. 14: Reflections produced by the pulse traveling to the right as it enters and exits the refined region for $\Delta x = \{1/40, 1/80, 1/160\}$ in the coarse grid.The RK4 without and with transition zone have been considered here. Notice that the presence of the transition point reduces the reflections for all the AMR algorithms.

the refined grid follows the pulses (i.e., the grid is refined whenever $\phi \geq 10^{-6}$). The convergence with RK4, for the different AMR algorithms, is shown in Figure 15.

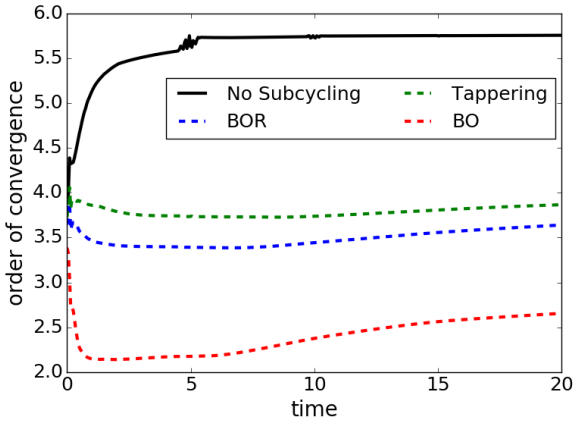


FIG. 15: AMR convergence. **TODO: repeat the convergence calculation with $N = 3200$ points as a higher resolution**

B. Newtonian ideal MHD equations

The ideal MHD equations, describing a magnetized perfect fluids, can be written in terms of the total energy E and the momentum density S_i

$$E = \frac{1}{2}\rho v^2 + \rho\epsilon + \frac{B^2}{2} \quad , \quad S_i = \rho v_i \tag{162}$$

where ρ is the fluid density, ϵ its internal energy, v_i its velocity and B^i the magnetic field. Within these definitions, the complete set of evolution equations can be written as

$$\partial_t \rho + \partial_k[\rho v^k] = 0 \tag{163}$$

$$\partial_t E + \partial_k[(E + p + \frac{B^2}{2})v^k - (v_j B^j)B^k] = 0 \tag{164}$$

$$\partial_t S_i + \partial_k[\rho v^k v_i + \delta^k_i(p + \frac{B^2}{2}) - B^k B_i] = 0 \tag{165}$$

$$\partial_t B^i + \partial_k[v^k B^i - v^i B^k + \delta^{ki}\psi] = 0 \tag{166}$$

$$\partial_t \psi + c_h^2 \partial_i B^i = -\kappa_\psi \psi \tag{167}$$

where ψ is a scalar introduced to enforce dynamically the solenoid constraint $\nabla_i B^i$. This divergence cleaning approach allows to propagate the constraint violations with a speed c_h and damp them exponentially in a timescale $1/\kappa$.

In order to close this system of equations one needs to provide an additional equation relating the pressure to the other fluid variables $p = p(\rho, \epsilon)$ (i.e., an Equation of State (EoS)). A common choice, that we will follow here, is to consider the ideal gas EoS $p = (\Gamma - 1)\rho\epsilon$. Within this choice, all the fluid variables can be recovered through algebraical relations from the evolved fields.

1. *Circularly polarized Alfven wave*

Our first benchmark test of the numerical scheme for fluids is the 2D circularly polarized Alfven wave problem [13], which is the advection of a smooth solution of the ideal compressible MHD equation in a periodic 2D box in the x, y-plane. The initial conditions are set as follows:

$$\rho = 1 \tag{168}$$

$$p = 0.1 \tag{169}$$

$$B_\parallel = 1 \tag{170}$$

$$B_\perp = v_\perp = 0.1 \sin(2\pi x_\parallel) \tag{171}$$

$$B_z = v_z = 0.1 \cos(2\pi x_\parallel) \tag{172}$$

$$x_\parallel = (x \cos\alpha_k + y \sin\alpha_k) \tag{173}$$

where α_k is the angle between the wavevector $\mathbf{k}$ and the x direction, which is related to the ratio of the domain lengths, $\tan\alpha_k = L_y/L_x$. The subscripts $\parallel$ and $\perp$ refer to the components parallel and perpendicular to $\mathbf{k}$, respectively; the perpendicular component of the magnetic field is related to the B_x, B_y components by $B_\perp = B_y\cos\alpha_k - B_x\sin\alpha_k$. Such setup admits an analytical, stationary solution, consisting in the advection of the magnetic field along the domain diagonal, with a period $t_{cycle} = L_x/\cos\alpha_k$.

We use the ideal EoS with $\Gamma = 5/3$, domain size $L_x = L_y = 2$, corresponding to $\alpha = \pi/4$, $t_{\rm cycle} = 2\sqrt{2}$. We first perform tests with a single mesh of N^2 points, with five resolutions $N = 16, 32, 64, 128, 256$. First we explore different reconstruction methods without any mesh refinement, with a LLF flux formula and the $4^{\rm th}$ order Runge-Kutta schema for the time integration with a time-step $\Delta t = k_{\rm cfl}\Delta x/\sqrt{2}$ where $k_{\rm cfl} < 1$ is the Courant factor. We set a conservative value $k_{\rm cfl} = 0.25$, which is low enough to ensure that the discretization errors are dominated by the spatial terms. We evolve the solutions up to $t = 10$ (about 3.5 cycles) and then verify the absolute error compared to the analytical solution, as well as the convergence rate of the different methods. For each simulation, we calculate the error by integrating over the entire dominion the L1-norm of the relative difference of the values of B_x between the numerical solution after exactly 3 cycles and the initial analytical data. We checked that the errors calculated over other magnetic field components behave in the same way. We check that, for each method and resolution, the accumulated calculated errors grow linearly with the number of cycles.

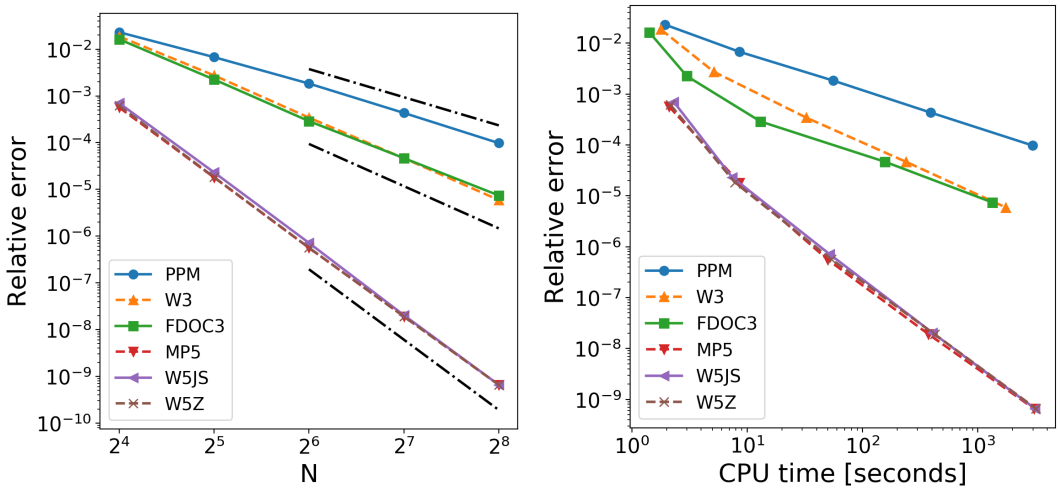


FIG. 16: Relative errors for different methods and resolutions (coloured lines), measured as the L1-norm of the difference between the numerical and the analytical solution of B_x after 3 cycles ($t = 6\sqrt{2}$), as a function of number of points (top) and CPU time (bottom). Clearly, the fifth-order schemes achieve a smaller error for a given number of points with less CPU time. The slopes of the three black dotted lines represent, from above to below, the nominal $2^{\rm nd}, 3^{\rm rd}$ and $5^{\rm th}$ convergence orders.

The relative errors of the solution after exactly 3 cycles, are shown in Fig. 16. One can clearly see that all methods, with the exception of FDOC5 (see middle panel of Fig. 18), behave as their nominal convergence predict (Fig. 17): WENO5-JS, WENO5-Z and MP5 show the same convergence fifth order, WENO3, FDOC3 and FDOC5 converge at third order and PPM converge only at second order. The relative errors for different resolutions allow us to note that, among the 5^th order methods, WENO5-Z shows a slight improvement in accuracy, compared to WENO5JS and MP5. Note that for $N \geq 256$, the calculation of error is inaccurate because of the limited number of digits. Actually, any small difference (like the last digit of an input value) brings to a visible apparent deviation from the accuracy order, but this effect vanishes as soon as the solution accumulates higher errors.

We also check different values of ϵ for the MP5 and WENO5 methods, with some cases shown in Fig. 18 (left panel). For this specific smooth problem, we found no differences for a range of values of ϵ in WENOs methods (see a few undistinguishable cases in the figure), while, for MP5 high resolution runs, the values of $\epsilon \lesssim 10^{-6}$ brings to a higher error than for $\epsilon \geq 10^{-6}$. Note also that MP5 is known to be less robust. Hereafter, we set the nominally optimal values $\epsilon = \delta x^2$ for WENO3, $\alpha = 4$ and $\epsilon = 10^{-6}$ for MP5, and $\epsilon = \delta x^4$ for WENO5-JS and WENO5-Z.

Besides the accuracy measured by the relative errors as a function of N, we are also interested on the computational cost. The CPU time [16] required to achieve a given accuracy for the different methods is shown at the bottom panel of Figure 16. Note that with the lowest resolution ($N = 16$), the CPU time is dominated by the initial data setup, thus it is less dependent on method. We have verified that, for our code and setup, the CPU time here shown stochastically vary by about $5\% - 20\%$ depending on the case. Our results indicate that, among the lower order methods, for a given accuracy FDOC5 is the fastest. The high-order methods are comparable in speed.

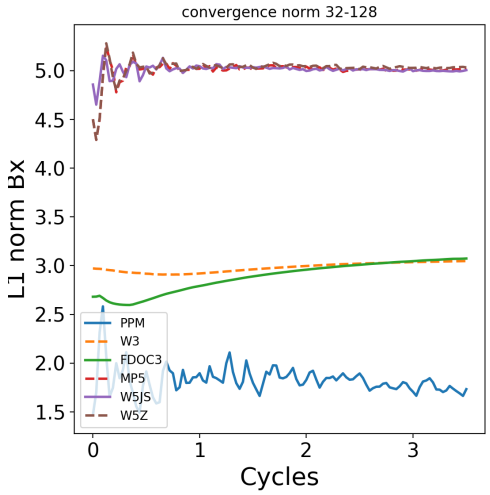


FIG. 17: Convergence order calculated with the relative L1-errors of B_x after 3 cycles, for $N = 32, 64, 128$. Here we show the unigrid case, but the FMR behaves in the same way.

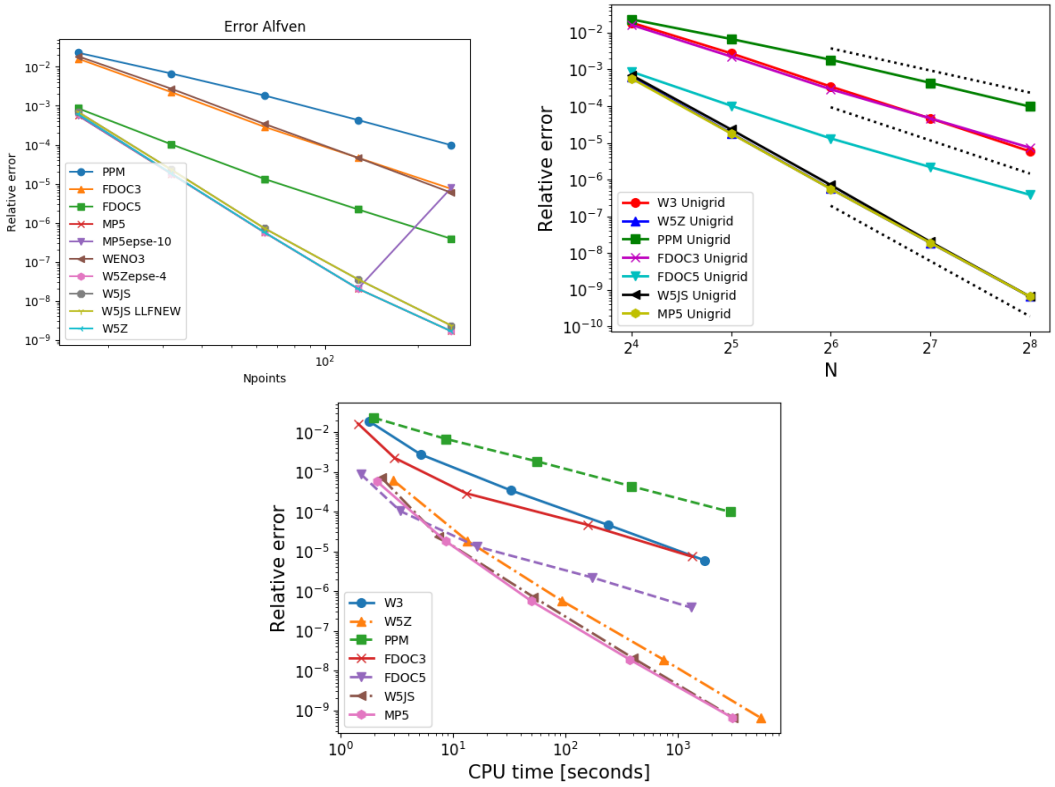


FIG. 18: Relative L1-errors of B_x after 3 cycles for other choices of ϵ (left, function of N), and including FDOC5 (middle, function of N, and right, function of CPU time).

We run the same battery of tests for refined meshes, considering two cases:

• FMR1x4: a refined mesh with refinement factor 4 and boundaries $[0.5, 1.5] \times [0.5, 1.5]$

• FMR2x2: two refined meshes, with relative refinement factor 2: the first with boundaries $[0.45, 1.55] \times [0.45, 1.55]$, the second $[0.5, 1.5] \times [0.5, 1.5]$. This case is shown in Fig. 19.

We checked that, for the tested schemas, the convergence order is maintained, and the errors are slightly reduced, as shown in Fig. 20 for WENO3 and WENO5Z. Note that, in a realistic case, this choice of FMR is not computationally

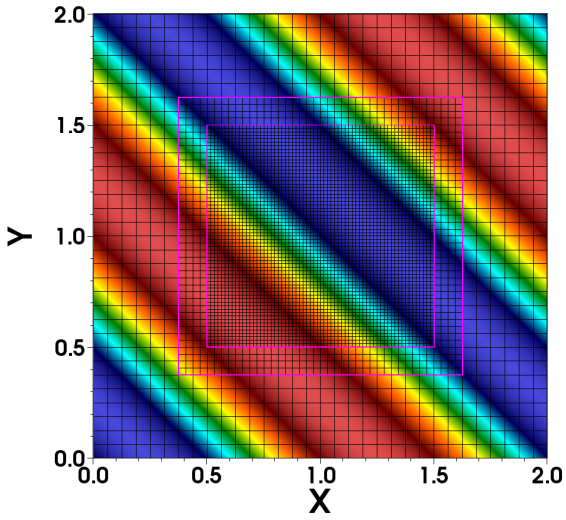


FIG. 19: *2D CP Alfven waves in the FMR2x2 case*: B_z component and refined meshes.

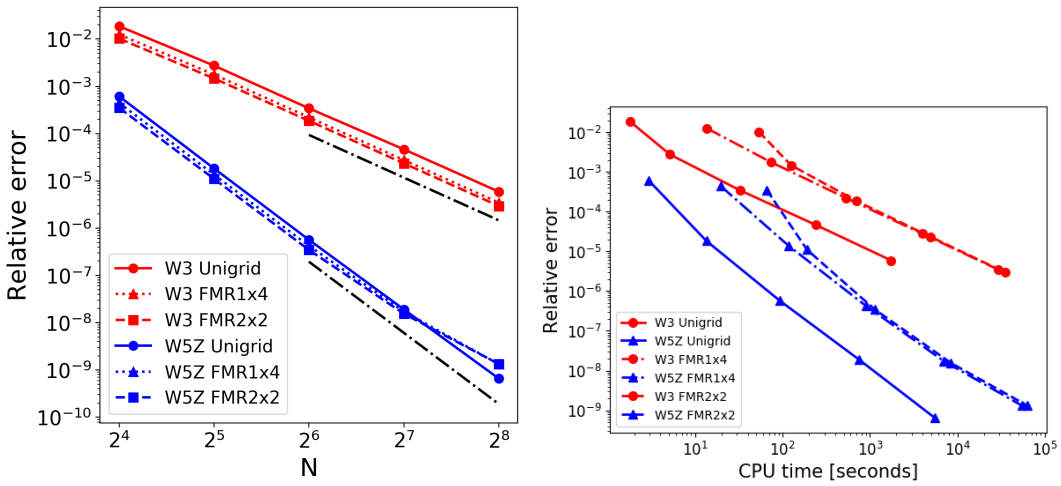


FIG. 20: *2D CP Alfven waves errors*: comparison of relative L1-errors of B_x without/with FMR for WENO3 and WENO5Z, as a function of N (left) and CPU time (right). The slopes of the black dotted lines represent, from above to below, the nominal 2^{nd} and 3^{rd} convergence orders.

convenient (see right panel of Fig. 20), because the solution is propagating in and out from the refined region. This implies that the error, calculated on the main mesh, is dominated by the non-refined region. However, this test is useful to prove that the convergence order is maintained for all the tested methods and no numerical artifacts appear, thus confirming the results found for the scalar wave, but for a more challenging model, even though consisting in a smooth solution.

2. *Magnetic shock tube*

We further test our code capabilities to capture shocks through a non-smooth MHD problem: the Brio and Wu 1D magnetic shock tube [14], which is the MHD extension of the classical SOD shock tube. The left state ($0 \leq x < 0.5$) is given by $\rho = 1$, $p = 1$, $B_y = 1$, the right state ($0.5 \leq x \leq 1$) is given by $\rho = 1$, $p = 1$, $B_y = 1$. Everywhere, $\mathbf{v} = 0$, $B_x = 0.75$ and $\gamma = 2$. Note that this magnetic version of the shock tube is more challenging, especially with such relatively high values of B_x, B_y.

We test the evolved profiles for PPM, WENO3, WENO5Z, MP5 with $N = 50, 100, 200, 400$.[17] We use a timestep

$dt = kcfldx$, with $k_{cfl} = 0.2$. We compare the results with the exact solution (solid line, "HR"), evaluated by running the same problem with $N = 4000$, with PPM. We find results quantitatively consistent with tested codes [15].

Hereafter we analyze the results for $t = 0.2$. In Figs. 21, 22 we show the profiles of ρ (top left sub panels), B_y (top right sub-panels), v_x (bottom left sub-panels), p (bottom right sub-panels), with different methods for $N = 50, 100, 200, 400$ from top to bottom, respectively. As expected, the convergence order is around one, due to the presence of shocks. Note that PPM, as it is well known, is able to attain a satisfactory accuracy for non-smooth solutions, similar to the highest order method, WENO5Z, and much better than WENO3. The velocity component v_x is the hardest to be reproduced, with oscillations shown even for $N = 400$ in all methods.

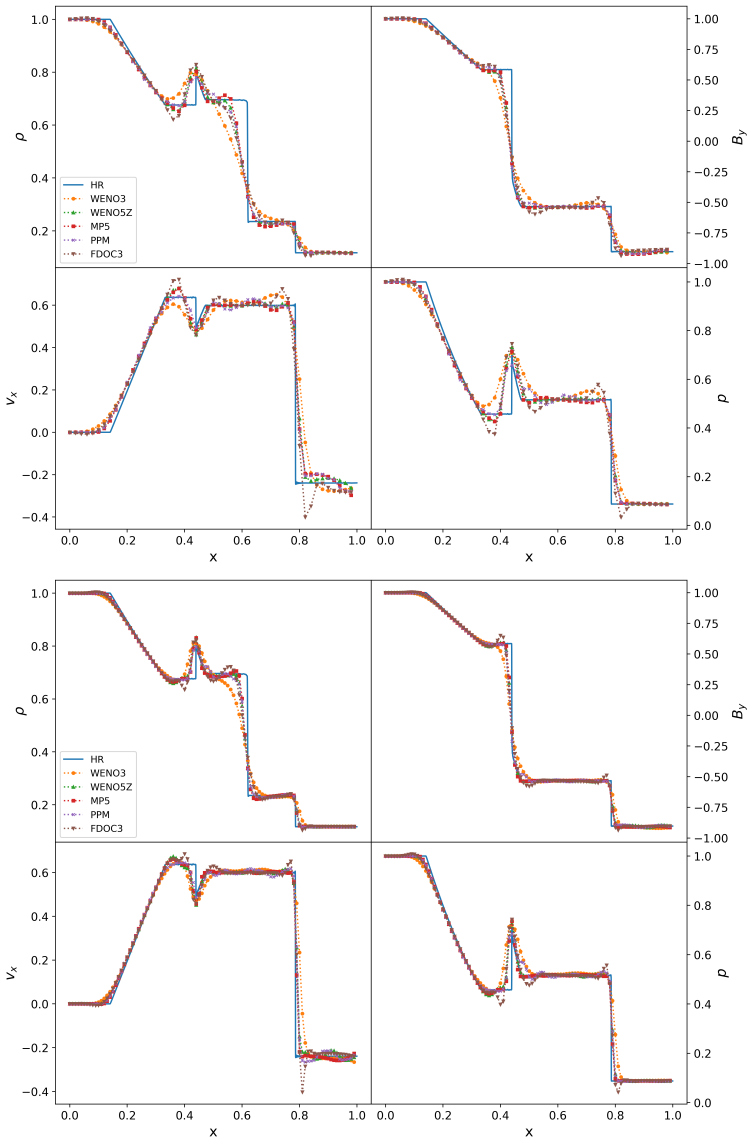


FIG. 21: Brio & Wu shock tube test in unigrid at $t = 0.2$ for $N = 50, 100$ (from top to bottom, respectively): profiles of ρ (top left sub panels), B_y (top right sub-panels), v_x (bottom left sub-panels), p (bottom right sub-panels), with different methods. The exact solution (solid line, "HR"), has been evaluated by running the same problem with $N = 4000$, with PPM.

We repeat the same test for WENO3 and WENO5Z by with a FMR, considering two cases:
• FMR1x4: a refined mesh with refinement factor 4 and boundaries $[0.4, 0.6]$ (dark grey in figures)
• FMR2x2: two refined meshes, with relative refinement factor 2: the first with boundaries $[0.34, 0.66]$ (light grey band in figures), the second at $[0.4, 0.6]$. These regions are soon crossed by both the rarefaction wave propagating to the left, and the shock front propagating to the right.

In Fig. 23 we show the same profiles as before, for $N = 400$, with and without FMR, for WENO3 (top), WENO5Z (middle), MP5 (bottom). We show the details of the profiles for WENO3 (Fig. 25), WENO5Z (Fig. 26), at different

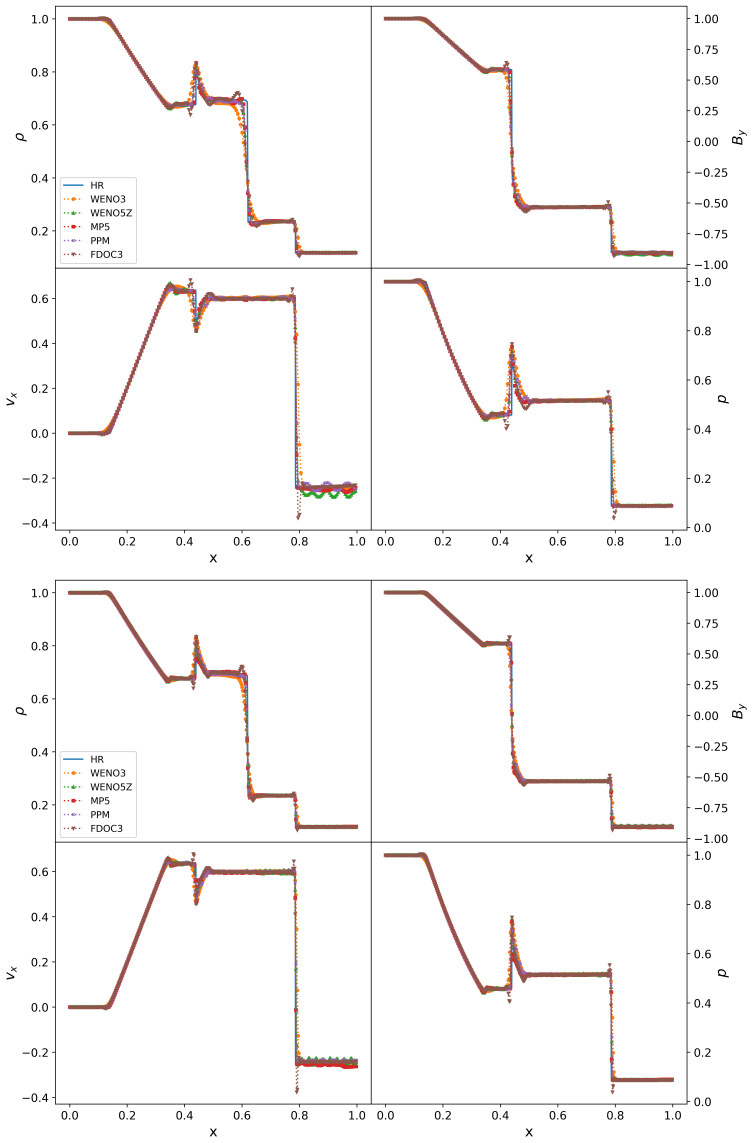


FIG. 22: Same as Fig. 21, for $N = 200, 400$.

resolutions.

Such choice allows us to evaluate the gain in accuracy. In Fig. **??** and Fig. **??** we show the comparison of L1-errors of ρ, B_y, v_x as a function of time (with $N = 400$) and number of points (at $t = 0.2$), respectively. We note that the convergence norm considering the errors for $N = 100, 200, 400$ (see Fig. **??**) oscillates a lot, both in time and with different resolutions. Keeping this in mind, the most accurate methods are PPM, WENO5Z and MP5. WENO5Z shows more oscillations in v_x and B_y but on average it adheres to the exact solution as well as MP5 and PPM. WENO3 is the worst one in terms of accuracy.

C. Gresho-Chan Vortex

The Gresho-Chan Vortex problem has been implemented using a particle-based simulation.

This test sets a 2D stationary vortex that should be in stable equilibrium. Since centrifugal forces and pressure gradients balance exactly, any deviation from the initial configuration that develops over time is spurious and of purely numerical origin. The azimuthal component of the velocity in this test rises linearly up to a maximum value of v_0 which is reached at $r = R1$ and subsequently decreases linearly back to zero at $2R1$

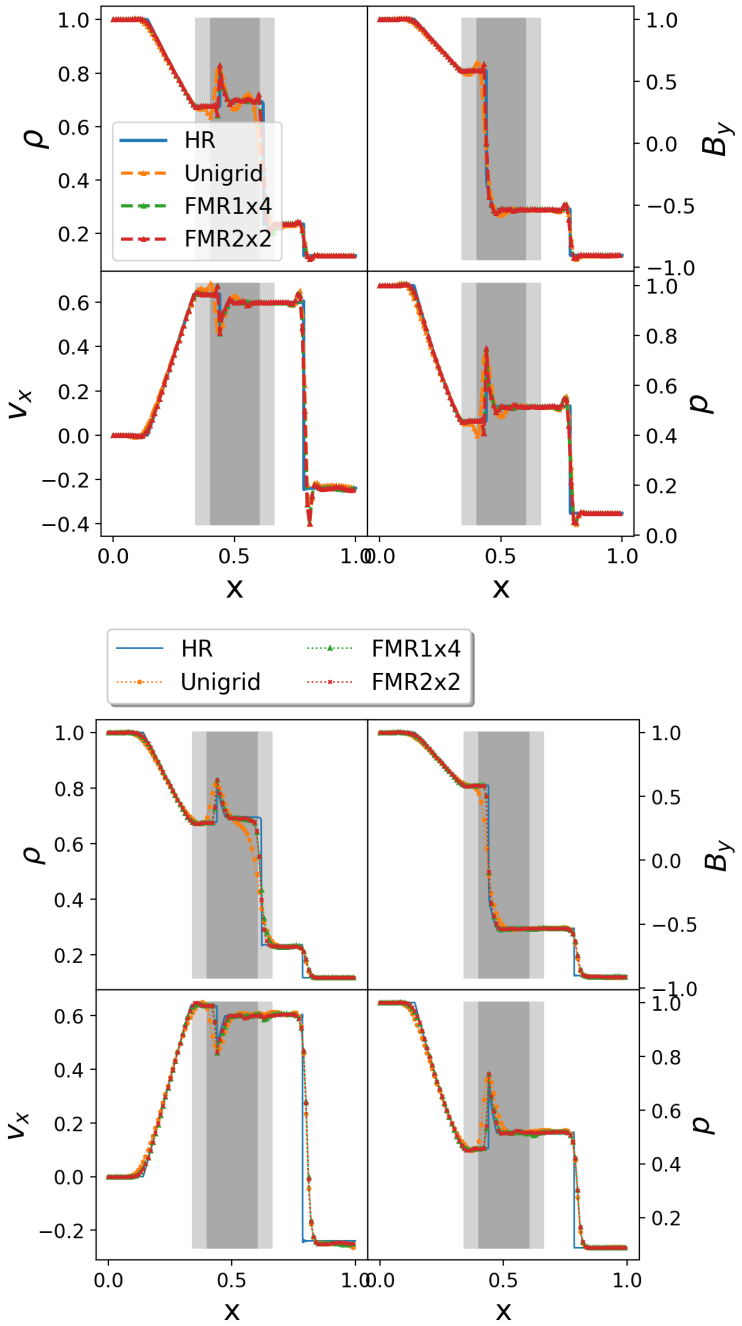


FIG. 23: Brio & Wu shock tube test for $N = 400$ at $t = 0.2$, for FDOC3, WENO3 (top to bottom): profiles of ρ (top left sub panels), B_y (top right sub-panels), v_x (bottom left sub-panels), p (bottom right sub-panels) with and without FMR (see legends).

$$v_\phi(r) = v_0 \begin{cases} u & u \leq 1 \\ 2-u & 1 < u \leq 2 \\ 0 & u > 2 \end{cases} \tag{174}$$

$$P(r) = P_0 \begin{cases} \frac{1}{2}v_0^2 u^2 & u \leq 1 \\ 4v_0^2(\frac{u^2}{8} - u + \ln u + 1) & 1 < u \leq 2 \\ 4v_0^2(\ln 2 - \frac{1}{2}) & u > 2 \end{cases} \tag{175}$$

We are using $v_0 = 1$, together with $R1 = 0.2$, and a polytropic exponent of $\frac{5}{3}$. Four different resolutions has been run: 40, 80, 160, and 320. Those numbers refers to the number of particles by axis. Figure 27 shows the profiles over R for the different resolutions. The solid lines correspond to the particle averages of $|v|$, while the dots correspond to the real particles velocities. Increasing the solution, the ideal triangular profile is reach.

Figure 28 shows the average errors over time. It can be seen that the error clearly diminishes when resolution is high.

D. Particle-Mesh Interaction Test

In order to test integration and interaction between mesh and particle systems an adaptation of the Advection Equation has been implemented as follows:

$$\partial_t \Psi = -(1-\mu)\partial_k \Psi - \mu \partial_k \Phi \tag{176}$$

$$D_t \Phi - v^k \partial_k \Phi = -(1-\eta)\partial_k \Phi - \eta \partial_k \Psi \tag{177}$$

where Φ is a particle field and Ψ is a mesh field.

This way, using appropriate values for μ and η, interaction between mesh and particle RHS can be tested. A value of 0 means that current field is calculated without the influence of the other field, while the value of 1 indicates complete influence from the other field and no influence from current field. Middle values balances influence of both fields.

The initial data consists on a gaussian profile:

$$\Phi, \Psi = 0.5e^{(-\frac{(x-1)^2+(y-1)^2}{0.1})} \tag{178}$$

in a periodic domain of $[0, 2]^2$.

There must be said that the numerical method used to solve mesh field equation is order 4, while particle method is order 2. Figure 30 shows a first convergence test performed to check the correct implementation of those numerical methods.

Then, three resolutions have been tested in order to calculate convergence, 25, 50, and 100 for both cells and particles. Figure 29 shows four differente selection for μ and η parameters. It is remarkable that the mesh reduces its order of convergence to 2 when field particle influences on mesh equation. Furthermore, particle gets the convergence order of mesh when its equation gets all the influence of mesh field and none of its own field.

Additional tests have been performed to test particle velocity and its effects on convergence. The tests use an average from particle fields into the underlying mesh (50, 100 and 200 particles in 25, 50 and 100 meshes respectively). The average uses the same kernel calculation that particle method uses.

Three particle velocities are tested: $v = 0.0$, $v = 0.5$, $v = 1.0$ Figures 30, 31, and 32 respectively. Convergence of particles and mesh keep their respective values.

The following graphic 33 represents the error of middle resolution:

E. The Integration

An integrator for Simflowny variables is available. There are currently two options, Integral and L2-Norm. Their formulas are as follows in 2D (easily to get 3D ones):

$$dx * dy * \frac{u(i,j) + u(i+1,j) + u(i,j+1) + u(i+1,j+1)}{4} \tag{179}$$

$$dx * dy * (\frac{u(i,j) + u(i+1,j) + u(i,j+1) + u(i+1,j+1)}{4})^2 \tag{180}$$

---

[1] J. C. Butcher, *Numerical Methods for Ordinary Differential Equations* (John Wiley and Sons, Ltd, 2008), ISBN 9780470753767, URL http://dx.doi.org/10.1002/9780470753767.fmatter.

[2] L. Pareschi and G. Russo, J. Sci. Comput. **25**, 112 (2005).

[3] G. Calabrese, L. Lehner, O. Reula, O. Sarbach, and M. Tiglio, Classical and Quantum Gravity **21**, 5735 (2004), gr-qc/0308007.

[4] P. Colella and P. R. Woodward, Journal of Computational Physics **54**, 174 (1984).

[5] A. Suresh and H. Huynh, Journal of Computational Physics **136**, 83 (1997), ISSN 0021-9991, URL http://www.sciencedirect.com/science/article/pii/S0021999197957454.

[6] C. Bona, C. Bona-Casas, and J. Terradas, Journal of Computational Physics **228**, 2266 (2009), 0810.2185.

[7] G.-S. Jiang and C.-W. Shu, Journal of Computational Physics **126**, 202 (1996), ISSN 0021-9991, URL http://www.sciencedirect.com/science/article/pii/S0021999196901308.

[8] C.-W. Shu, *Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws* (Springer Berlin Heidelberg, Berlin, Heidelberg, 1998), pp. 325–432, ISBN 978-3-540-49804-9, URL https://doi.org/10.1007/BFb0096355.

[9] D. S. Balsara, Living Reviews in Computational Astrophysics **3**, 2 (2017), 1703.01241.

[10] *flux decomposition formulas.* (2017), URL https://wjrider.wordpress.com/2017/10/06/thinking-about-flux-splitting-for-general-riemann-solvers/.

[11] M. Innocenti, G. Lapenta, S. Markidis, A. Beck, and A. Vapirev, Journal of Computational Physics **238**, 115 (2013), ISSN 0021-9991, URL http://www.sciencedirect.com/science/article/pii/S0021999112007590.

[12] G. Lapenta, Journal of Computational Physics **181**, 317 (2002), ISSN 0021-9991, URL http://www.sciencedirect.com/science/article/pii/S0021999102971263.

[13] G. Tóth, Journal of Computational Physics **161**, 605 (2000).

[14] M. Brio and C. C. Wu, Journal of Computational Physics **75**, 400 (1988).

[15] L. Del Zanna, O. Zanotti, N. Bucciantini, and P. Londrillo, Astronomy and Astrophysics **473**, 11 (2007), 0704.3206.

[16] these tests are performed in a single processor, on a desktop DELL XPS computer, Processor Intel Core i7-7700 CPU, 3.60 GHz

[17] Due to the need of running a 2D simulation in Simflowny, we set in the vertical direction a number $N/10$ of points.

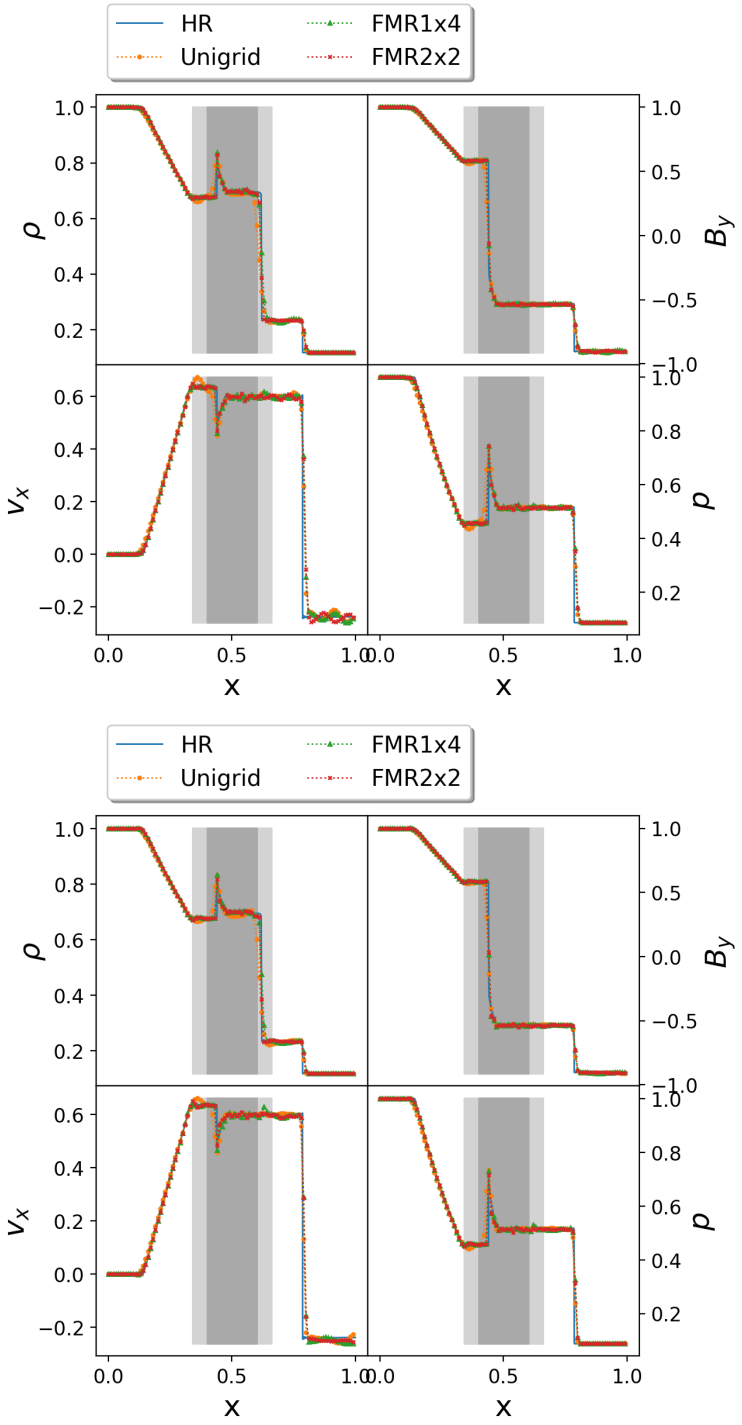
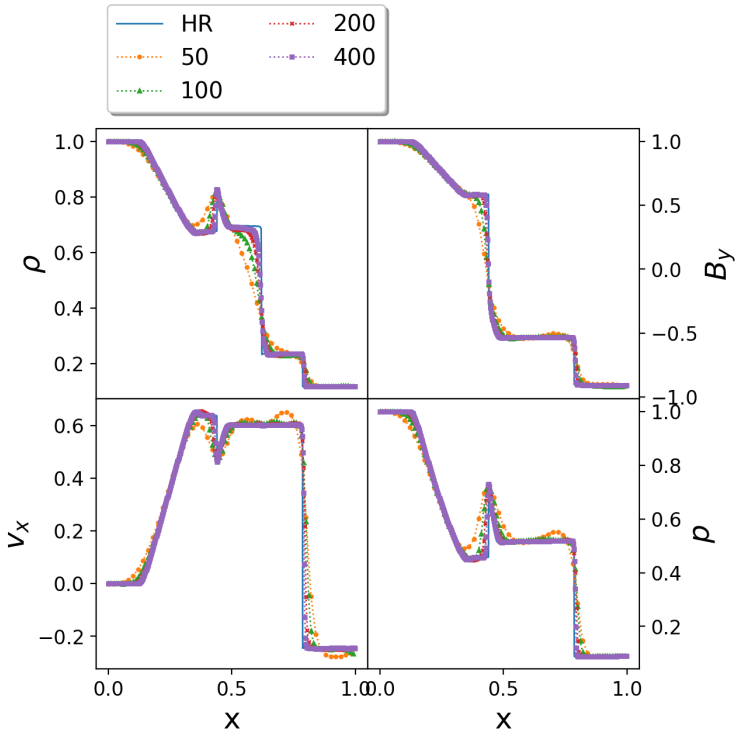
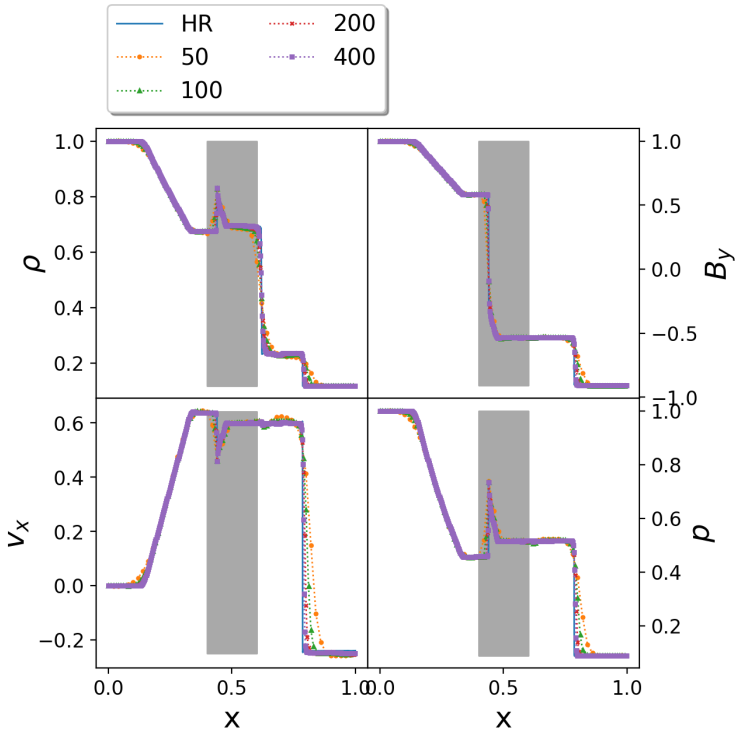
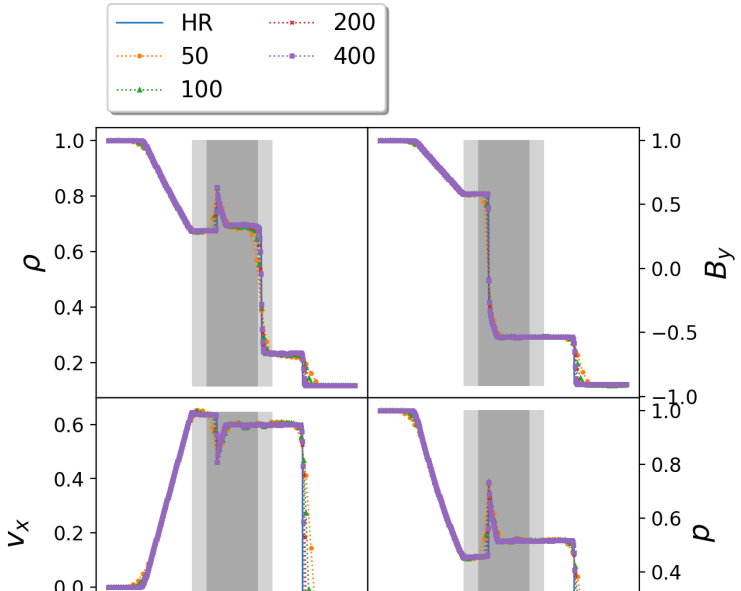
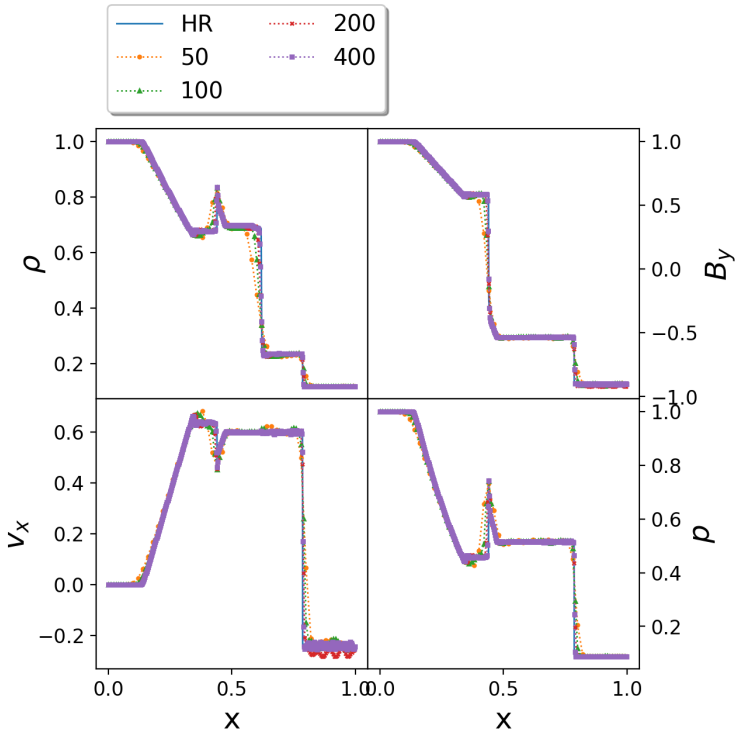


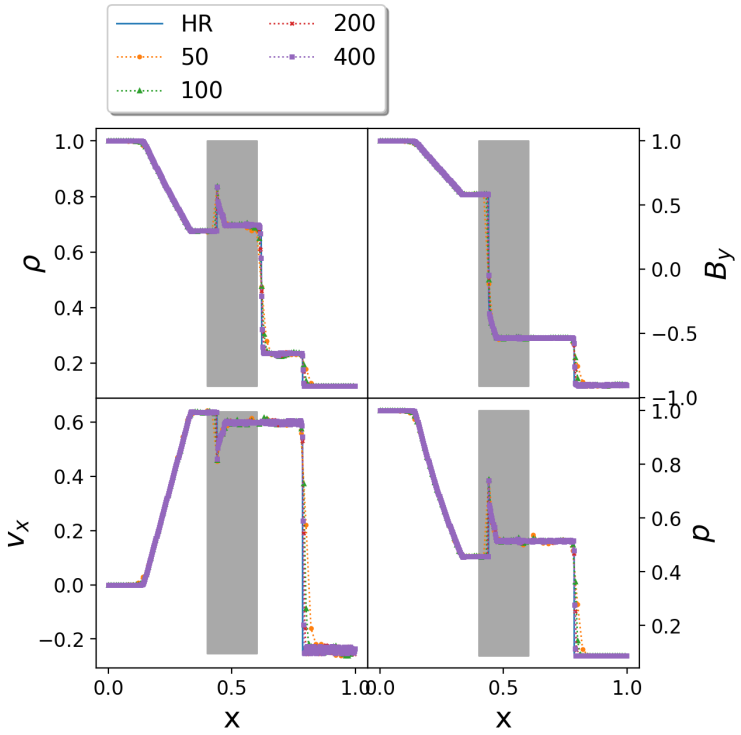
FIG. 24: Brio & Wu shock tube test for $N = 400$ at $t = 0.2$, for WENO5Z, MP5 (top to bottom): profiles of ρ (top left sub panels), B_y (top right sub-panels), v_x (bottom left sub-panels), p (bottom right sub-panels) with and without FMR (see legends).

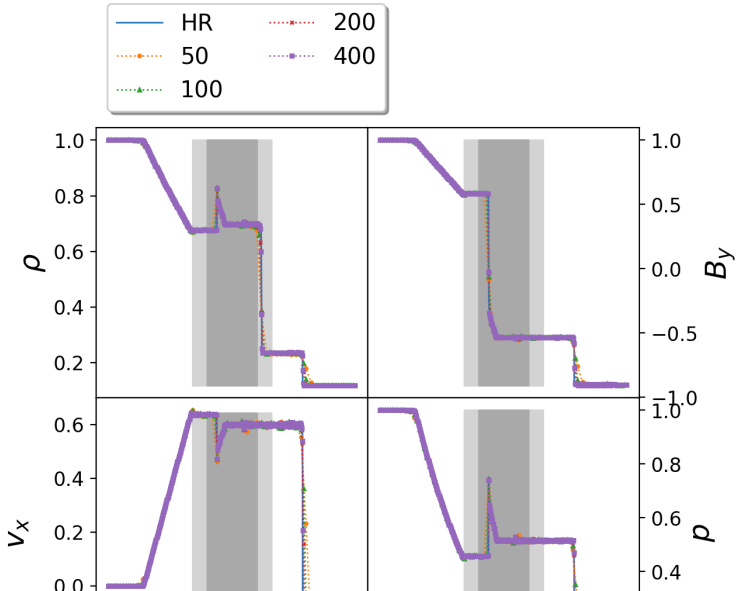

HR
50
100
200
400
ρ
B_y
v_x
p
x
x


HR
50
100
200
400
ρ
B_y
v_x
p
x
x


HR
50
100
200
400
ρ
B_y
v_x
p


HR
50
100
200
400
ρ
B_y
v_x
p
x
x


HR
50
100
200
400
ρ
B_y
v_x
p
x
x


HR
50
100
200
400
ρ
B_y
v_x
p

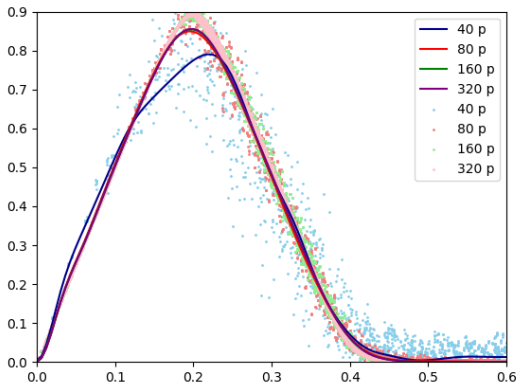


FIG. 27: *Different resolution* $|v|$ *at time 1.* 40, 80, 160, 320

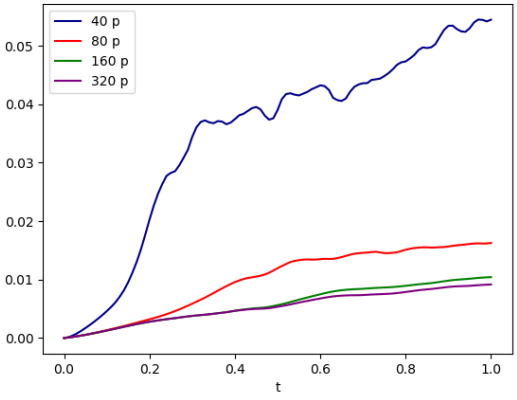


FIG. 28: *Different resolution error.* 40, 80, 160, 320

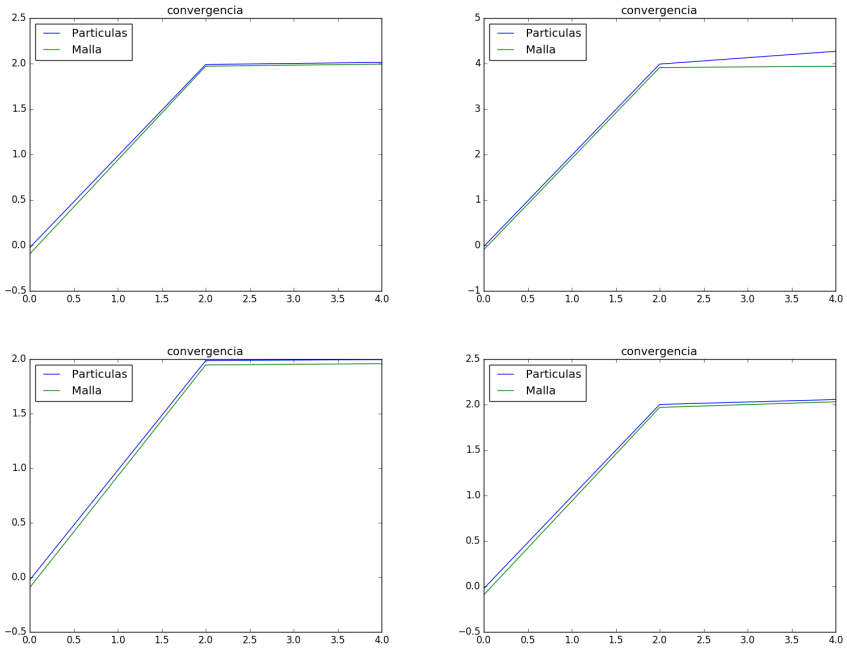


FIG. 29: *Convergence.* From left to right and top to bottom; $\mu = 0.5$ $\eta = 0.5$, $\mu = 0$ $\eta = 1$, $\mu = 1$ $\eta = 0$, $\mu = 1$ $\eta = 1$

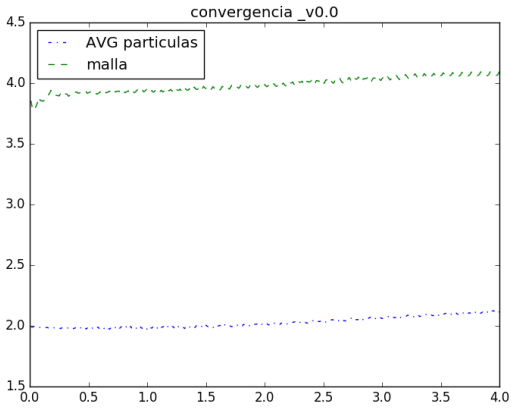


FIG. 30: *Convergence.* $v = 0$

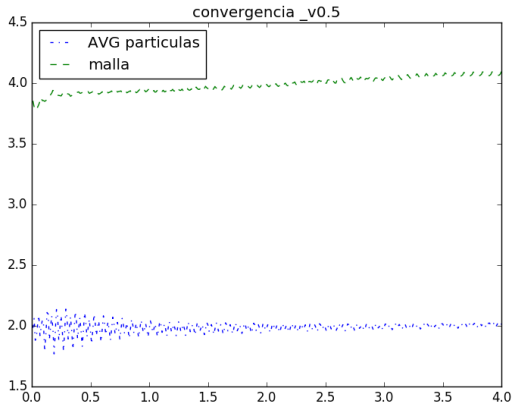


FIG. 31: *Convergence.* $v = 0.5$

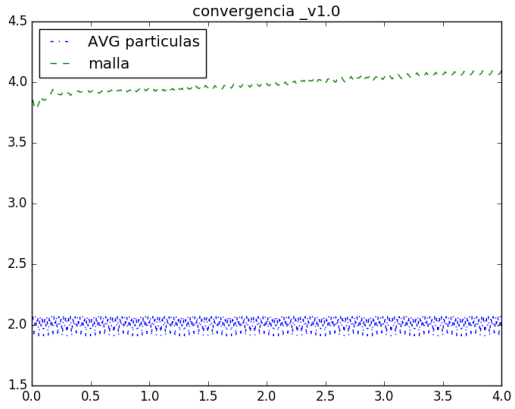


FIG. 32: *Convergence.* $v = 1$

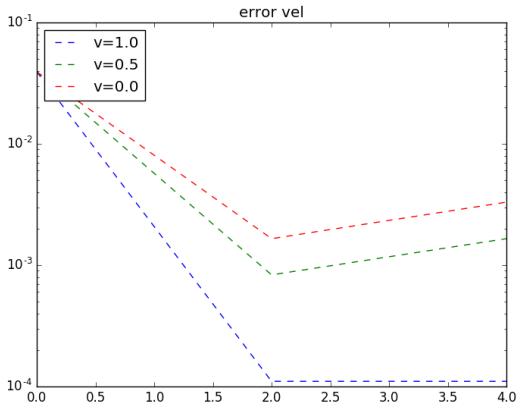


FIG. 33: *Simulation error.* Error in central position of gaussian profile (every t = 2)