The Astrophysical Journal Supplement Series, 256:42 (8pp), 2021 October https://doi.org/10.3847/1538-4365/ac2511

A Coscaling Grid for Athena++

Roark Habegger[1] and Fabian Heitsch[2]
[1] University of Wisconsin, Dept. of Astronomy, Madison, WI 53705, USA
[2] University of North Carolina, Dept. of Physics and Astronomy, Chapel Hill, NC 27599, USA
Received 2021 July 2; revised 2021 August 24; accepted 2021 September 7; published 2021 October 5

Abstract

We present a coscaling-grid formalism and its implementation in the magnetohydrodynamics code Athena++. The formalism relies on flow symmetries in astrophysical problems involving expansion, contraction, and center-of-mass motion. The grid is evolved at the same time order as the fluid variables. The user specifies grid evolution laws, which can be independent of the fluid motion. Applying our implementation to standard hydrodynamic test cases leads to improved results and higher efficiency, compared to the fixed-grid solutions.

Unified Astronomy Thesaurus concepts: Computational methods (1965); Hydrodynamical simulations (767); Astronomical simulations (1857)

1. Introduction

Many astrophysical phenomena involve evolution in length scale over orders of magnitude. Supernova and kilonova ejecta expand from stellar radii to parsecs before fully mixing with the interstellar gas (Ostriker & McKee 1988; Montes et al. 2016; Metzger 2019). Ionization and stellar-wind-driven supershells expand from the scales of a small stellar association to tens of parsecs (McCray & Kafatos 1987), impacting the ambient gas and possibly triggering star formation (Elmegreen & Lada 1977). Protostellar collapse occurs from parsec scales to a few astronomical units.

Numerical modeling of phenomena evolving over a large range of scales requires methods capable of adapting the scale of the spatial discretization. Lagrangian methods such as smoothed-particle hydrodynamics (Monaghan 1992) achieve this by following fluid elements defined by a fixed mass rather than a fixed volume. Eulerian methods—the choice for many applications because conservation laws are more easily realized—require adaptive mesh algorithms to cover similar scale ranges as smoothed-particle hydrodynamics (Krumholz et al. 2007; Klein 2017). Lagrangian remapping combines elements of Lagrangian methods with those of Eulerian ones, evolving fluid elements in a Lagrangian frame, but continually remapping the motion onto an Eulerian grid (Lufkin & Hawley 1993). A more recent development is moving-mesh codes, solving flux-conservative problems on meshes that move with the fluid in a Lagrangian fashion (Springel 2010; Hopkins 2015), preserving the strength of finite volume methods over smoothed-particle hydrodynamics for some applications (Heitsch et al. 2011).

A conceptually simpler alternative to moving-mesh codes exploits possible symmetries in an astrophysical problem. The relativistic hydrodynamic codes JET and DISCO have shown the effectiveness of limiting mesh movement to a particular direction in cylindrical coordinates (Duffell & MacFadyen 2013; Duffell 2016). While symmetries appear during spherical and cylindrical expansion and contraction, they also appear as a "comoving" motion in a single Cartesian direction. For problems involving a drastic change in spatial scale, a coscaling grid can be more efficient than adaptive mesh-refinement techniques (Röpke 2005). If uniformity of dissipative properties is relevant, such as for problems involving turbulent transport, a coexpanding grid may be preferable over adaptive mesh refinement. To make use of these advantages for coscaling grids, we implemented the method in the Eulerian grid code Athena++ (Stone et al. 2020).[3]

The grid can be comoving or rescaled, in both cases retaining the initial cell aspect ratio. The grid evolution is integrated at the same time order as the fluid variables. The motion of cell walls necessitates additional wall-flux terms when updating the fluid variables. The time dependence of the grid scaling is defined by a user-specified function. The coscaling grid can be combined with the adaptive mesh capabilities of Athena++.

The method improves results of standard test cases. Here, we include the Sod shock tube test and a spherical blast wave test. The sphericity of multidimensional blast waves is preserved on Cartesian grids by a factor of 3 better than for fixed-grid simulations. While the implementation is a factor ∼1.1 slower across resolutions and processor number than the stock version of Athena++ for standard hydrodynamics, the advantage of the coscaling grid lies in its ability to cover spatial (and thus dynamical) ranges over orders of magnitude, resulting in a net efficiency gain.

2. Formalism

Eulerian ideal magnetohydrodynamics solve the conservation laws

$$\frac{\partial \boldsymbol{U}^T(\vec{x}, t)}{\partial t} = -\vec{\nabla}^T \cdot \overline{\boldsymbol{\Gamma}}(\vec{x}, t). \tag{1}$$

The row vector $\boldsymbol{U}^T$ contains the conservative variables. The matrix $\overline{\boldsymbol{\Gamma}}$ has columns with the flux of each conservative quantity. These fluxes have rows corresponding to the various coordinate directions ($\hat{x}_1$, $\hat{x}_2$, and $\hat{x}_3$) (Stone et al. 2008). The length of $\boldsymbol{U}^T$ depends on the physics of the problem. For ideal MHD, $\boldsymbol{U}^T$ has eight components and the matrix $\overline{\boldsymbol{\Gamma}}$ has three rows and eight columns. Altogether, the right-hand side is the flux divergence. For a Cartesian grid, the matrix $\overline{\boldsymbol{\Gamma}}$ has the form

$$\overline{\boldsymbol{\Gamma}} = \boldsymbol{F}^T\hat{x} + \boldsymbol{G}^T\hat{y} + \boldsymbol{H}^T\hat{z} \tag{2}$$

[3] https://github.com/roarkhabegger/athena-TimeDependentGrid

where each boldfaced vector of conservative variables is the flux of those quantities in the given direction.

By integrating Equation (1) over a discrete volume ΔV, the differential equation becomes an integrodifferential equation. For static grids, this equation can be rewritten as an ordinary differential equation for the conservative variables $\boldsymbol{U}$ of each cell, indexed by (i, j, k):

$$\begin{aligned}\frac{d}{dt}\boldsymbol{U}_{i,j,k} = &-\frac{1}{\Delta x_i}(\boldsymbol{F}_{i+\frac{1}{2},j,k} - \boldsymbol{F}_{i-\frac{1}{2},j,k}) \\ &- \frac{1}{\Delta y_j}(\boldsymbol{G}_{i,j+\frac{1}{2},k} - \boldsymbol{G}_{i,j-\frac{1}{2},k}) \\ &- \frac{1}{\Delta z_k}(\boldsymbol{H}_{i,j,k+\frac{1}{2}} - \boldsymbol{H}_{i,j,k-\frac{1}{2}}),\end{aligned} \tag{3}$$

where the conservative variables $\boldsymbol{U}$ are averaged over the cell volume and the flux vectors $\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{H}$ are averaged over a cell wall (see Stone et al. 2020; Felker & Stone 2018). In Equation (3), we have removed the row and column vector notation since there are no vector operations left. Therefore, either case (row or column vectors) would satisfy the equation as it is shown. We show a more detailed derivation of Equation (3) in the Appendix.

A critical part of the discrete averaging leading to Equation (3) is moving the time derivative out of the volume integral on the left-hand side of Equation (1) (see the Appendix). The justification for that step is the Reynolds transport theorem for a quantity f over a volume V and boundary B,

$$\frac{d}{dt}\int_V dV\, f = \int_V dV\, \frac{\partial f}{\partial t} + \int_B dA\, (\vec{w}\cdot\hat{n}) f. \tag{4}$$

For a static grid, the velocity $\vec{w}$ of the boundary is 0, so Equation (4) reduces to

$$\frac{d}{dt}\int_V dV\, f = \int_V dV\, \frac{\partial f}{\partial t}, \tag{5}$$

allowing the time derivative to be moved in and out of any volume integral.

A coscaling grid will lead to additional fluxes due to moving cell walls, rendering the surface integral in Equation (4) nonzero (Springel 2011). The differential equation now reads (see Appendix A.2 for details)

$$\begin{aligned}&\frac{d}{dt}\boldsymbol{U}_{i,j,k} \\ &= -\frac{1}{\Delta x_i}(\boldsymbol{F}_{i+\frac{1}{2},j,k} - \boldsymbol{F}_{i-\frac{1}{2},j,k} - \boldsymbol{W}_{i+\frac{1}{2},j,k} + \boldsymbol{W}_{i-\frac{1}{2},j,k}) \\ &-\frac{1}{\Delta y_j}(\boldsymbol{G}_{i,j+\frac{1}{2},k} - \boldsymbol{G}_{i,j-\frac{1}{2},k} - \boldsymbol{V}_{i,j+\frac{1}{2},k} + \boldsymbol{V}_{i,j-\frac{1}{2},k}) \\ &-\frac{1}{\Delta z_k}(\boldsymbol{H}_{i,j,k+\frac{1}{2}} - \boldsymbol{H}_{i,j,k-\frac{1}{2}} - \boldsymbol{S}_{i,j,k+\frac{1}{2}} + \boldsymbol{S}_{i,j,k-\frac{1}{2}}),\end{aligned} \tag{6}$$

where $\boldsymbol{W}$, $\boldsymbol{V}$, $\boldsymbol{S}$ are the volume-averaged wall fluxes in the various Cartesian coordinate directions. For example, the average wall flux in the $\hat{x}$ direction is given by the integral

$$\begin{aligned}\boldsymbol{W}_{i+\frac{1}{2},j,k} = &\frac{1}{\Delta y_j \Delta z_k} \\ &\times \iint dydz[w_x(x_{i+\frac{1}{2}}, t)\boldsymbol{U}(x_{i+\frac{1}{2}}, y, z, t)].\end{aligned} \tag{7}$$

While implementing the above correction is an important step, there is another correction hidden in Equation (6). The code will use a time integrator to solve the differential equation. Regardless of the particular integrator, the time integration reads

$$\begin{aligned}\boldsymbol{U}_{i,j,k}(t^{n+1}) = &\boldsymbol{U}_{i,j,k}(t^n) \\ &+ \frac{1}{t^{n+1}-t^n}\int_{t^n}^{t^{n+1}} dt\left[\frac{d}{dt}\boldsymbol{U}_{i,j,k}\right].\end{aligned} \tag{8}$$

This assumes a static grid. The assumption is hidden in the notation: for a static grid, the volume averages of $\boldsymbol{U}$ are taken over the same volume. To correct for this in the time-dependent grid case, we need to change Equation (8) to

$$\begin{aligned}\boldsymbol{U}_{i,j,k}(t^{n+1}) = &\frac{V_{i,j,k}(t^n)}{V_{i,j,k}(t^{n+1})}[\boldsymbol{U}_{i,j,k}(t^n) \\ &+ \frac{1}{t^{n+1}-t^n}\int_{t^n}^{t^{n+1}} dt\left[\frac{d}{dt}\boldsymbol{U}_{i,j,k}\right]\Bigg].\end{aligned} \tag{9}$$

Here, $V_{i,j,k}(t)$ is the volume of the (i, j, k) cell at time t.

Thus, a comoving, coscaling, or generically time-dependent grid requires two corrections. The first is to include cell-wall movement by using the true flux (Equation (6)). The second is to include the change in cell volume, scaling the conserved quantities (Equation (9)).

3. Implementation

Athena++ solves Equation (1) over a static grid (Stone et al. 2008, 2020). A coscaling grid requires the integration of the grid's motion over time, in addition to the integration of the physical variables. After this grid integration, we add corrections to the physical variables in the form of wall fluxes and volume scaling (Section 3.1; derived in Section 2). Finally, all coordinate variables need to be updated throughout the full mesh hierarchy, including derived quantities such as cell volumes and areas, and reconstruction coefficients. This requires changes to the task list implemented in Athena++ (Section 3.2).

3.1. Wall-flux and Volume-change Corrections

To include the corrections to the update equation (Equation (6)), we need to approximate the wall-flux integral (Equation (7)). Assuming the cell wall's velocity and the conservative quantity are constant on the cell wall, Equation (7) reads

$$\boldsymbol{W}_{i+\frac{1}{2},j,k} = w_x(x_{i+\frac{1}{2}}, t)\boldsymbol{U}(x_{i+\frac{1}{2}}, t), \tag{10}$$

providing a simple definition for the wall flux.

To add these fluxes, we introduce an "expansion" source function. Here, we add the wall fluxes to the conservative variables in the same manner used in the base Athena++ code to add the hydrodynamic fluxes (except we need to consider the difference in sign, see Equation (6)). Then, we multiply all conservative variables by the volume expansion factor

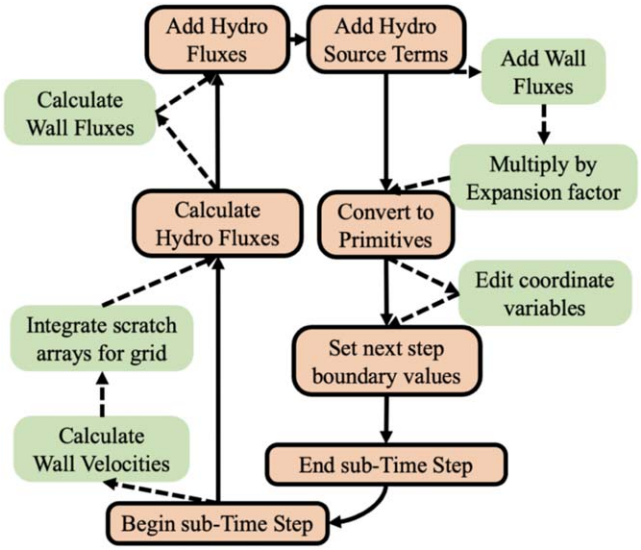

Figure 1. Changes to the task list in Athena++ for the coscaling-grid module. The orange shaded boxes (with black borders) show the normal progression through the task list during a given substep of a time integrator. The green boxes (with no borders) and dashed arrows are the detours necessary for a coscaling grid. For simplicity, we only show base Athena++ tasks affected by the coscaling grid. For more detailed flow charts of Athena++ task lists, see Stone et al. (2008, 2020).

$V_{i,j,k}(t^n)(V_{i,j,k}(t^{n+1}))^{-1}$ (see Equation (9)). With this last step, the conservative variables are fully updated. These steps are shown as a part of Figure 1.

3.2. Task-list Changes

Athena++ uses a task list to control and optimize the sequence of operations necessary to solve Equation (1) (Stone et al. 2008, 2020). For any time integrator, the code completes the task list for every time *substep*. For example, when Athena ++ runs with a fourth-order time integrator, the task list completes four times during a time step, once for each substep. Each loop through the list is slightly different, because Athena ++ uses minimum-register time integrator methods (Ketcheson 2010). By incorporating the coscaling-grid integration into this task list, the implementation works for any time integrator available in Athena++.

The coscaling grid requires additional tasks during a substep. The first is an evaluation of the user-prescribed velocity function for every cell wall of the grid. The second task takes those stored velocities and integrates the grid, over time, to determine where each cell wall will be at the end of the substep. The third edits the stored coordinates to reflect the change in location of each cell wall. The other detour boxes in Figure 1 regarding wall-flux calculation and correcting the conservative variables are implemented within other tasks in the task list.

The first two new tasks are executed before any hydrodynamic or magnetic field calculations, since the wall velocity function only uses information from the previous time substep (see Figure 1).

We update the coordinate grid after the conservative and primitive variables have been fully updated, and before boundary values are calculated for the next time substep. As a result, all variables (grid and physical) are fully updated and available for output or the next integration step.

4. Tests

We assess the accuracy and stability of the coscaling-grid implementation with the 1D Sod shock tube (Sod 1978; Stone et al. 2008), and with the 2D cylindrical blast wave. For the latter, we compare the accuracy and computational cost of our implementation to the equivalent static-grid simulation. To highlight the applicability of the coscaling grid to astrophysical problems, we finish with the evolution of a blast wave from free expansion to the Sedov–Taylor phase.

Each test uses a gas with a specific heat capacity ratio $\gamma = \frac{5}{3}$.

4.1. Sod Shock Test

The Sod shock tube is a popular test of a numerical code's accuracy and stability (Sod 1978; Stone et al. 2008) by comparing the numerical solution to the corresponding Riemann problem (e.g., Toro 2019). The initial conditions of the test are a density and pressure discontinuity at the origin with 0 velocity throughout the simulation. The test uses Cartesian coordinates in one dimension.

The left side of the initial discontinuity has density $\rho_l = 1$ and pressure $P_l = 1.0$, whereas the right side has density $\rho_r = 0.125$ and pressure $P_r = 0.1$ (Sod 1978).

Figure 2 compares a coscaling-grid simulation (64e5) with a static grid at the same number of grid points (64s). We also show a higher-resolution static-grid simulation (2048s) as an approximation of the analytic solution.

For the coscaling grid, the domain initially extends over $-0.1 \leqslant x \leqslant 0.1$ and expands by a factor of 5, reaching the static-grid domain size of $-0.5 \leqslant x \leqslant 0.5$ at $t = 0.25$. Thus, the final output of the simulations can be directly compared (see Figure 3). The expanding grid keeps the cell size uniform.

In terms of code validation, the solution for the coscaling grid is consistent with the analytic solution (Figure 2). Specifically for $x \leqslant 0$, the coscaling grid approximates the analytic solution more closely than the static grid. Large differences are expected at the discontinuities (see also Figure 3), since the discontinuities cannot be resolved—slopes just get steeper with increasing resolution. Generally, slopes are slightly steeper for the coscaling grid, suggesting that tracking the three waves with the coscaling grid improves the accuracy of the solution. Figure 3 indicates the coscaling grid is more accurate over the evolution of the test, with steeper discontinuities and a more accurate shock location.

The expanding and static simulations used a piecewise linear reconstruction method. The piecewise parabolic reconstruction method (PPM) is known to cause oscillations in the velocity (Lee 2011). PPM combined with the coscaling grid results in higher oscillations than when using the static grid. Since the coscaling grid requires more time steps and thus more reconstructions, oscillations can reach higher amplitudes.

4.2. 1D Blast Wave Test

Our second test is the one-dimensional blast wave in spherical coordinates. This allows us to check the volume expansion correction in other coordinate systems. The initial condition consists of an inner region with $r < 1.0$, which is overpressured by a factor of 10^4 and overdense by a factor of 10^3 with respect to the ambient medium ($\rho_{\rm amb} = 0.1$, $P_{\rm amb} = 1.0$).

As for the Sod shock tube, we compare three models, one at high resolution (1000 grid points), and the fixed- and

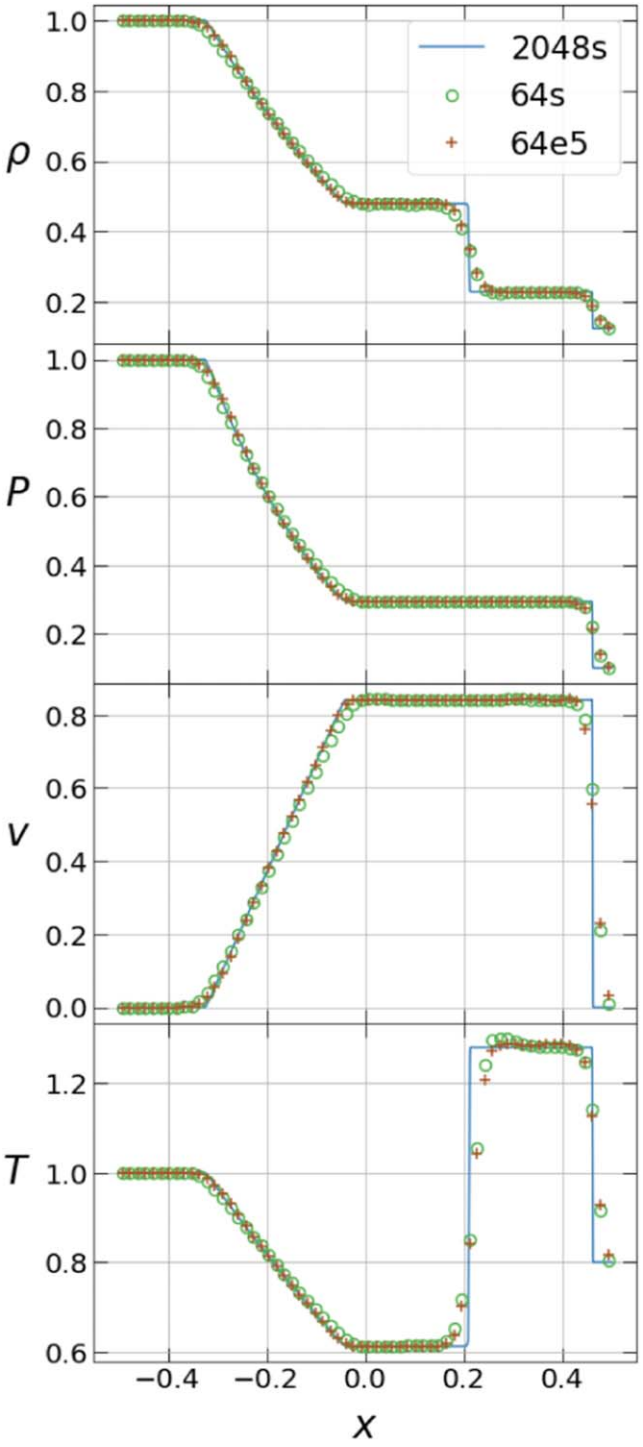

Figure 2. The primitive variables for the Sod shock tube at $t = 0.25$. The line corresponds to a resolution of $N = 2048$ on a static grid, and the green circles represent a static-grid model at $N = 64$ cells. The corresponding coscaling-grid model at $N = 64$ is indicated by the brown plus markers. The latter starts out with a domain $[-0.1, 0.1]$ and expands to the domain shown by a factor of 5.

expanding-grid models at 100 grid points each. Figure 4 summarizes the results, showing the density profile and normalized residuals for two time instances. While discontinuities introduce large residuals, the shock position is more accurately traced by the coscaling-grid model—for the fixed-grid model, the shock position leads compared to the high-resolution model.

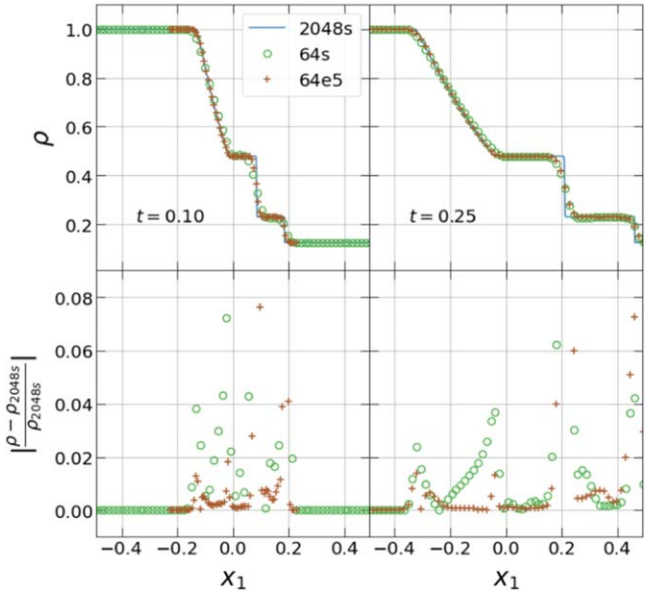

Figure 3. Time evolution of the Sod shock tube. The top row shows the density profile for the three models 64e5, 64s, and 2048s, and the bottom row the normalized residuals with respect to the 2048s model. Large errors at the discontinuities arise because the discontinuities cannot be resolved physically. For smooth flow regions, the coscaling-grid solution approximates the 2048s simulation more closely.

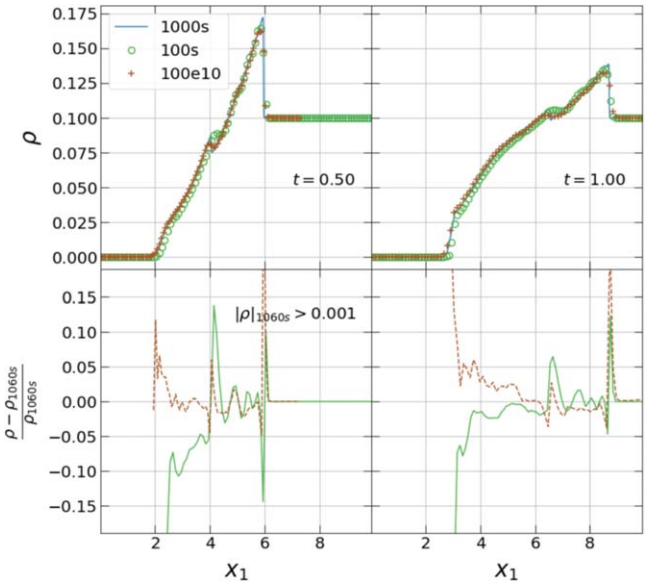

Figure 4. Density and normalized residual profiles at two times for the spherical 1D blast wave at a resolution of 100 cells. Residuals are calculated with respect to the static high-resolution simulation at 1000 grid points (blue solid line). Discontinuities introduce large residuals. The expanding grid initially matches the resolution of the high-resolution profile, leading to more accurate shock locations.

Early in the expanding simulation, the peaks are significantly more resolved than in the static-grid simulation. The sharpness of discontinuities plays an important role in radiative losses. Therefore, simulations with radiative losses will be more accurate if they use a coscaling grid.

4.3. 2D Blast Wave Test

We test the multidimensional performance of the coscaling grid via the 2D blast wave, both for cylindrical and Cartesian

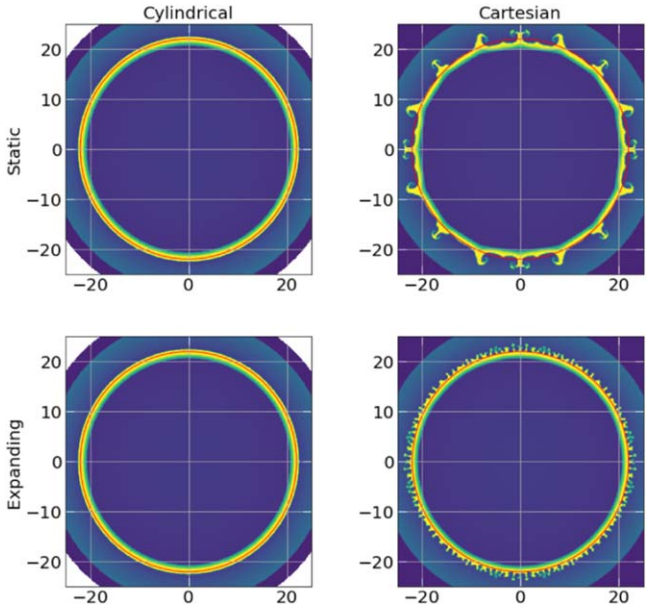

Figure 5. Density maps of four two-dimensional cylindrical blast wave models. For reference, a red circle is plotted at $r = 22.5$ in all the maps. This is the radial coordinate of the peak in density for the cylindrical simulations. Models on cylindrical grids (left, $n_r = 512$) are nearly indistinguishable. The strength of the coscaling grid (bottom row) becomes clear when comparing Cartesian grid models (right column, $n_x = n_y = 512$). Rayleigh–Taylor fingers triggered by the discretization are nearly uniformly distributed for the coscaling grid.

coordinates. Cartesian coordinates introduce directionally dependent numerical diffusion, since the resolution is effectively lower along the diagonals by a factor of $\sqrt{2}$.

Figure 5 compares four 2D blast wave models. Cylindrical models on a grid with resolution $(n_r, n_\theta) = (512, 64)$ are shown on the left, Cartesian ones at linear resolution of 512 on the right, while the top row shows static-grid models, and the bottom row coscaling-grid ones. We track the shell to determine the grid expansion rate required to keep the blast wave within the simulation domain.

The cylindrical models are essentially indistinguishable, as expected. Using cylindrical (instead of spherical) coordinates allows us to more easily compare to the 2D Cartesian case.

Both Cartesian cases suffer from directionally dependent numerical diffusion, yet the coscaling grid achieves a more spherical solution. We expect Rayleigh–Taylor fingers to be triggered at the grid scale. Since the expanding grid starts with a smaller scale, the instabilities are seeded at a smaller scale, leading to a more uniform distribution of Rayleigh–Taylor fingers and a more circular appearance of the blast wave.

Figure 6 provides a more detailed view of the deviation from sphericity. The vertical axis measures the difference between the largest outer radius and smallest inner radius of the shell Δr for the Cartesian grid, normalized by the same quantity for the cylindrical grid. An effective shell thickness ratio of

$$\mathcal{R} \equiv \frac{\Delta r_{\rm cart}}{\Delta r_{\rm cyl}} = 1 \tag{11}$$

indicates a perfectly circular ring. The less circular the shell, the larger $\Delta r_{\rm cart}$ will become, and thus $\mathcal{R} > 1$. Results for the coscaling grid (solid lines) improve with higher resolution. With time, deviations from circularity grow because the discretization leads to Rayleigh–Taylor instabilities (see Figure 5). The effective shell thickness ratios $\mathcal{R}$ for the static

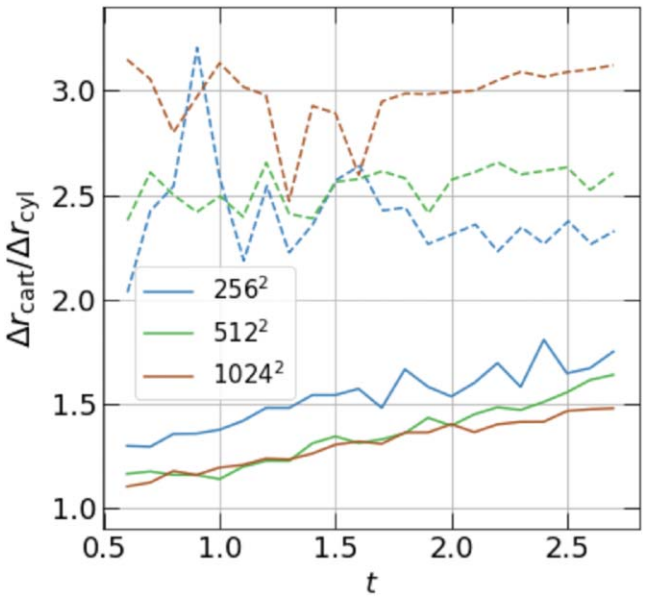

Figure 6. Deviation from circular shape (Equation (11)), against time, for the blast wave on a Cartesian grid. The dashed lines are static-grid simulations and the solid lines are expanding-grid simulations. At the same resolution the coscaling-grid simulations produce shells which are more uniformly circular.

grid (dashed lines) are at least a factor of 2 larger than for the coscaling grid and vary substantially with time right from the start of the shell expansion.

4.4. Performance

To compare the performance of the stock version of Athena ++ with the coscaling-grid implementation (Figure 7), we ran the blast wave test in three dimensions in Cartesian coordinates at a resolution of 256^3 and 512^3. We use basic hydrodynamics, i.e., no magnetic fields or other additional physics, except for the coscaling grid. The coscaling-grid implementation tracks the stock version closely, running at 90% of the base speed for small processor numbers, and without perceptible loss for large processor numbers. This behavior extends to the 512^3 resolution, and thus seems resolution independent, demonstrating that our modification to Athena++ is *minimally invasive*. While the additional steps clearly slow the code down, the speed decrease is offset with increased accuracy of the coscaling implementation (Section 4.3).

4.5. Long-term Blast Wave Evolution

As a final demonstration of the code's capabilities, we follow the evolution of a point explosion from free expansion to the Sedov–Taylor phase as in a supernova or kilonova remnant (Ostriker & McKee 1988; Montes et al. 2016). During free expansion, the velocity is constant and the radius scales as $r_s \propto t$. When the ejecta mass reaches the mass of the swept-up ambient gas, the blast wave enters the energy-conserving Sedov–Taylor phase with $r_s \propto t^{2/5}$. Once radiative losses become dominant, the snowplow (momentum-conserving) phase is reached. Here, we only consider the first two phases, leaving the implementation of radiative losses for a later contribution. The explosion is initialized with a total energy of $E = 10^8$, with kinetic energy $E_{\rm kin} = 0.99E$. The ejecta density is set to $\rho_e = 10^4$ within a radius of $r_s(0) = 1.2$. The ambient density and pressure are $\rho_a = 10^{-2}$ and $P_a = 10^{-7}$. These

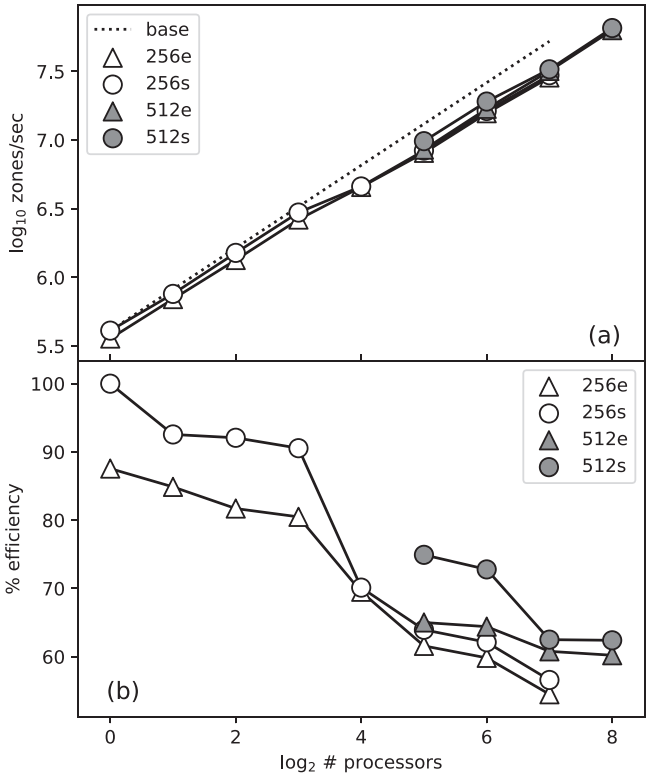

Figure 7. (a) Speed of the fixed grid (circles) and coscaling grid (triangles) against processor number, as a strong scaling measure. (b) Efficiency of the fixed and coscaling grids measured with respect to base efficiency of the fixed grid using one processor. The coscaling grid tracks the stock version of Athena ++, running at ∼90% of the base speed for small processor numbers, and without perceptible loss for large numbers.

values result in a transition radius between free expansion and Sedov–Taylor phase of

$$r_{ST} = r_s(0)\left(\frac{\rho_e}{\rho_a}\right)^{1/3} = 120. \tag{12}$$

We implemented the test for the expanding grid at 128 grid points and for the fixed grid at an approximate equivalent of 8192 points (Figure 8). Results agree with the analytical estimate for both implementations. The advantage of the expanding grid is obvious—it can follow the evolution to arbitrary time values. A more sophisticated implementation would include radiative losses to allow the blast wave to enter the snowplow phase.

5. Discussion

The coscaling-grid implementation provides a generalization of the coexpanding-grid formalism of Röpke (2005), applicable to expanding, contracting, or comoving frames. Its closest relative is Lagrangian remapping (Lufkin & Hawley 1993), where the fluid equations are solved in a Lagrangian frame to reduce advection errors, but the solution is interpolated (remapped) back onto an Eulerian grid. The underlying grid in our method is not strictly Eulerian any more, requiring additional fluxes due to cell-wall motion. Therefore, the effect of the grid motion can be integrated at the same time order as the fluid equations. The grid shape cannot change, which renders the method less flexible than moving-mesh codes (Springel 2010; Hopkins 2015). While dissipative properties

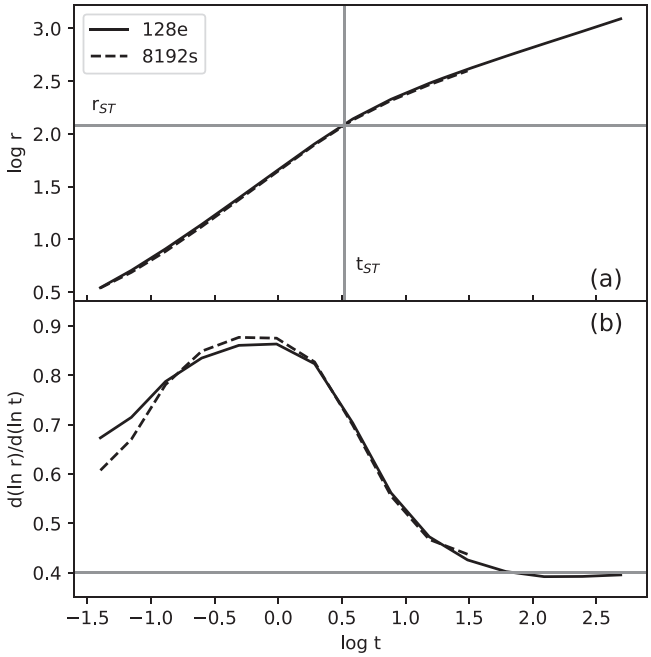

Figure 8. (a) Blast wave radius against time for the fixed-grid (8192f) and the expanding-grid (128e) model. Gray solid lines indicate the transition radius (Equation (12)) and time. (b) Logarithmic slope of $r(t)$, indicating the transition from the free expansion to the Sedov–Taylor phase ($d(\ln r)/d(\ln t) = 0.4$).

can vary with time due to expansion or contraction of the underlying grid, they stay constant across the grid, in difference to adaptive mesh-refinement techniques (Fryxell et al. 2000; Teyssier 2002; O'Shea et al. 2004; Cunningham et al. 2009; Stone et al. 2020). The closest implementations to ours are JET and DISCO, two moving-mesh codes which are restricted to singular dimensions (Duffell & MacFadyen 2013; Duffell 2016).

6. Summary

We present an implementation of a coscaling (expanding, contracting, or comoving) grid for the magnetohydrodynamics code Athena++ (Stone et al. 2020). The method can be used to follow the evolution of a system over orders of magnitude in scale, as long as the underlying assumption of an existing flow symmetry persists. The scaling prescription ensures the preservation of cell aspect ratios. The method's main strength lies in covering orders of magnitude in spatial scales for isotropically expanding or contracting systems, or for comoving systems, while keeping dissipative properties constant across the grid. It provides less flexibility than moving-mesh codes, but it can be combined with Athena++'s native adaptive mesh refinement.

We thank the University of North Carolina at Chapel Hill's Information Technology Services for providing the computational resources.

We also thank Dr. Ellen Zweibel for her help in editing this paper and for encouraging the completion of this project. This work was aided by Dr. Zweibel's NSF grant AST-2007323. We thank the anonymous referee for a very constructive report, and especially for pointing out an inconsistency in an earlier version of our grid implementation.

Appendix
Update Equation Derivation

We show the derivation for both time-independent and time-dependent grids (including our coscaling example) to highlight their differences, using a Cartesian grid as an example.

A.1. Fixed Eulerian Grid Method

For a static, time-independent grid, the integration of the left-hand side (LHS) of Equation (1) over space gives

$$\int_V dV\left[\frac{\partial \boldsymbol{U}(\vec{x}, t)}{\partial t}\right] = \frac{d}{dt}\int_V dV\,[\boldsymbol{U}(\hat{x}, t)]. \tag{A1}$$

Because the volume is independent of time, we can move the time derivative outside of the volume integral (see Equation (5)).

We then apply the same integration to the right-hand side (RHS) of Equation (1). Considering a Cartesian coordinate system with unit vectors $\hat{x}$, $\hat{y}$, $\hat{z}$, the matrix of fluxes can be split into the directional quantities ($\boldsymbol{F}$, $\boldsymbol{G}$, $\boldsymbol{H}$). This means the RHS is

$$-\int_V dV\,[\vec{\nabla}^T \cdot (\boldsymbol{F}^T\hat{x} + \boldsymbol{G}^T\hat{y} + \boldsymbol{H}^T\hat{z})]. \tag{A2}$$

By splitting the above volume integral into its constituent directions, the gradients in each direction can be removed. The integral of the flux divergence in the $\hat{x}$ direction is

$$\int_V dV\left[\frac{\partial \boldsymbol{F}}{\partial x}\right] = \iint dydz[\boldsymbol{F}(x_b, t) - \boldsymbol{F}(x_a, t)], \tag{A3}$$

where x_b and x_a are the upper and lower bounds, respectively, of the volume V in the $\hat{x}$ direction.

Using notation from Stone et al. (2008) and Stone et al. (2020), the LHS and RHS can be written in a computationally usable form. The first step in using this notation is to consider solving the equation on a computational grid or mesh, where each direction (x, y, z) is indexed by an integer (i, j, k) and each cell in the grid has a unique tuple of these integers. Each cell has a volume which we define as $V_{i,j,k}$. With these assumptions, the volume average of the conservative values $\boldsymbol{U}$ at time t is

$$\boldsymbol{U}_{i,j,k} = \frac{1}{V_{i,j,k}}\int_{V_{i,j,k}} [\boldsymbol{U}(x_i^1, x_j^2, x_k^3, t)]dV. \tag{A4}$$

Written with this notation, the LHS (Equation (A1)) becomes

$$V_{i,j,k}\frac{d\boldsymbol{U}_{i,j,k}}{dt}. \tag{A5}$$

We also need to define the integral over the divergence of the fluxes. To discretize Equation (A3), we can use the term

$$\boldsymbol{F}_{i+\frac{1}{2},j,k} = \frac{1}{\Delta y_j \Delta z_k}\iint dydz[\boldsymbol{F}(x_{i+\frac{1}{2}}, y, z, t)], \tag{A6}$$

where $x_{i+\frac{1}{2}}$ is the upper bound of the cell centered (with respect to the $\hat{x}$ coordinate) on x_i. The Δy_j and Δz_k are the width of the cell in the $\hat{y}$ and $\hat{z}$ directions respectively. Equation (A6) is the value of the flux of each conservative variable at the given wall of the cell. This notation can be used not only for $\boldsymbol{F}$, but also for $\boldsymbol{G}$ and $\boldsymbol{H}$. Altogether, the RHS (Equation (A2)) can be written as

$$-[\Delta y_j \Delta z_k(\boldsymbol{F}_{i+\frac{1}{2},j,k} - \boldsymbol{F}_{i-\frac{1}{2},j,k}) + \Delta x_i \Delta z_k(\boldsymbol{G}_{i,j+\frac{1}{2},k} - \boldsymbol{G}_{i,j-\frac{1}{2},k}) + \Delta x_i \Delta y_j(\boldsymbol{H}_{i,j,k+\frac{1}{2}} - \boldsymbol{H}_{i,j,k-\frac{1}{2}})]. \tag{A7}$$

Combining the LHS and RHS, we get the update equation (Equation (3)) for a discrete Cartesian grid. This is the final update equation and it is used in Athena++ to evolve the system (Stone et al. 2008, 2020). The entire derivation assumes that the grid does not depend on time. The next section outlines how the update equation changes when cell positions and sizes depend on time.

A.2. Time-dependent Eulerian Grid Method

The most important change to the static-grid update equation (Equation (3)) derivation in the coscaling-grid case comes from the Reynolds transport theorem, Equation (4). Integrating Equation (1) over space to make the LHS into Equation (A1) is still valid in the moving-grid method. However, we cannot simply move the time derivative outside of the volume integral. Instead, the LHS will be

$$\int_V dV\left[\frac{\partial \boldsymbol{U}(\vec{x}, t)}{\partial t}\right] = \frac{d}{dt}\int_{V(t)} dV\,[\boldsymbol{U}(\vec{x}, t)] - \int_B dA(\vec{w}\cdot\hat{n})\boldsymbol{U}. \tag{A8}$$

The above expression clearly indicates that there is an additional flux due to grid motion $\vec{w}$. This term can be directly incorporated to the RHS expression. Moving the extra term in Equation (A8) to the RHS, Equation (A2), the RHS for a Cartesian grid becomes

$$-\left[\int_V dV\,[\vec{\nabla}^T \cdot (\boldsymbol{F}^T\hat{x} + \boldsymbol{G}^T\hat{y} + \boldsymbol{H}^T\hat{z})] - \int_B dA\,(\vec{w}\cdot\hat{n})\boldsymbol{U}\right]. \tag{A9}$$

This is the most general formulation of the so-called "true flux" (Springel 2011) for a time-dependent grid. As seen in the fixed-grid derivation, the volume integral of the fluxes becomes an *area* average over each directional flux, $\boldsymbol{F}$, $\boldsymbol{G}$, and $\boldsymbol{H}$. The RHS will be

$$-\left[\int_{V(t)} dV\left[\frac{\partial \boldsymbol{F}}{\partial x} + \frac{\partial \boldsymbol{G}}{\partial y} + \frac{\partial \boldsymbol{H}}{\partial z}\right] - \sum_{m=1}^{6}\int_{A_m} dA\; w_m\boldsymbol{U}\right], \tag{A10}$$

where we have split the integration over the boundary B into a sum of integrals over the six walls of the Cartesian volume element. The velocity $w_m = \vec{w}_m \cdot \hat{n}_m$ is evaluated on the wall in the direction of the normal vector to A_m.

Ignoring the y and z directions, the RHS is

$$-\left[\int_{y(t)}\int_{z(t)} dydz[\boldsymbol{F}(x_{i+\frac{1}{2}}) - \boldsymbol{F}(x_{i-\frac{1}{2}}) - [w_x(x_{i+\frac{1}{2}})\boldsymbol{U}(x_{i+\frac{1}{2}}) - w_x(x_{i-\frac{1}{2}})\boldsymbol{U}(x_{i-\frac{1}{2}})]]\right]. \tag{A11}$$

The negative sign for the flux at the $x_{i-\frac{1}{2}}$ wall results from the normal vector $\hat{n} = -\hat{x}$ at that wall. Taking note of this

relationship between $\hat{n}$ and $\hat{x}$ allows us to write the flux using the x component of $\vec{w}$ at that wall, which is defined as $w_x(x_{i-\frac{1}{2}})$.

To formulate a new update equation, we need to define a wall flux for the various directions. We use $\boldsymbol{W}$, $\boldsymbol{V}$, and $\boldsymbol{S}$ to denote the wall flux in the x, y, and z directions respectively. As a result, the numerical term for the moving wall flux is

$$\boldsymbol{W}_{i+\frac{1}{2},j,k} = \frac{1}{\Delta y_j \Delta z_k} \times \iint dydz[w_x(x_{i+\frac{1}{2}}(t))\boldsymbol{U}(x_{i+\frac{1}{2}}(t),\, t)]. \tag{A12}$$

Using the notation above, the RHS is

$$\begin{aligned} -[&\Delta y_j \Delta z_k(\boldsymbol{F}_{i+\frac{1}{2},j,k} - \boldsymbol{F}_{i-\frac{1}{2},j,k} - \boldsymbol{W}_{i+\frac{1}{2},j,k} + \boldsymbol{W}_{i-\frac{1}{2},j,k}) \\ &+ \Delta x_i \Delta z_k(\boldsymbol{G}_{i,j+\frac{1}{2},k} - \boldsymbol{G}_{i,j-\frac{1}{2},k} - \boldsymbol{V}_{i,j+\frac{1}{2},k} + \boldsymbol{V}_{i,j-\frac{1}{2},k}) \\ &+ \Delta x_i \Delta y_j(\boldsymbol{H}_{i,j,k+\frac{1}{2}} - \boldsymbol{H}_{i,j,k-\frac{1}{2}} - \boldsymbol{S}_{i,j,k+\frac{1}{2}} + \boldsymbol{S}_{i,j,k-\frac{1}{2}})]. \end{aligned} \tag{A13}$$

Combining the LHS and RHS that we have derived above, we get an update equation (Equation (6)) for a simulation with a time-dependent grid. Considering the time integration involved in solving the ordinary differential equation (Equation (6)), we also find a necessary volume-change correction (see Equation (9)).

ORCID iDs

Roark Habegger https://orcid.org/0000-0003-4776-940X
Fabian Heitsch https://orcid.org/0000-0002-4775-039X

References

Cunningham, A. J., Frank, A., Varnière, P., Mitran, S., & Jones, T. W. 2009, ApJS, 182, 519
Duffell, P. C. 2016, ApJS, 226, 2
Duffell, P. C., & MacFadyen, A. I. 2013, ApJ, 775, 87
Elmegreen, B. G., & Lada, C. J. 1977, ApJ, 214, 725
Felker, K. G., & Stone, J. M. 2018, JCoPh, 375, 1365
Fryxell, B., Olson, K., Ricker, P., et al. 2000, ApJS, 131, 273
Heitsch, F., Naab, T., & Walch, S. 2011, MNRAS, 415, 271
Hopkins, P. F. 2015, MNRAS, 450, 53
Ketcheson, D. I. 2010, JCoPh, 229, 1763
Klein, R. I. 2017, MmSAI, 88, 642
Krumholz, M. R., Klein, R. I., & McKee, C. F. 2007, ApJ, 656, 959
Lee, D. 2011, in ASP Conf. Ser., 444, 5th Int. Conf. of Numerical Modeling of Space Plasma Flows (ASTRONUM 2010), ed. N. V. Pogorelov, E. Audit, & G. P. Zank (San Francisco, CA: ASP), 236
Lufkin, E. A., & Hawley, J. F. 1993, ApJS, 88, 569
McCray, R., & Kafatos, M. 1987, ApJ, 317, 190
Metzger, B. D. 2019, LRR, 23, 1
Monaghan, J. J. 1992, ARA&A, 30, 543
Montes, G., Ramirez-Ruiz, E., Naiman, J., Shen, S., & Lee, W. H. 2016, ApJ, 830, 12
O'Shea, B. W., Bryan, G., Bordner, J., et al. 2004, arXiv:astro-ph/0403044
Ostriker, J. P., & McKee, C. F. 1988, RvMP, 60, 1
Röpke, F. K. 2005, A&A, 432, 969
Sod, G. A. 1978, JCoPh, 27, 1
Springel, V. 2010, MNRAS, 401, 791
Springel, V. 2011, arXiv:1109.2218
Stone, J. M., Gardiner, T. A., Teuben, P., Hawley, J. F., & Simon, J. B. 2008, ApJS, 178, 137
Stone, J. M., Tomida, K., White, C. J., & Felker, K. G. 2020, ApJS, 249, 4
Teyssier, R. 2002, A&A, 385, 337
Toro, E. F. 2019, ShWav, 29, 1065