Solver-in-the-Loop Applications in Astrophysical (Magneto)hydrodynamics

Leonard Storcks leonard.storcks@iwr.uni-heidelberg.de
Tobias Buck tobias.buck@iwr.uni-heidelberg.de
AstroAI Lab, Interdisciplinary Center for Scientific Computing
Heidelberg University

Abstract

We present two promising applications of training machine learning models inside a differentiable astrophysical (magneto)hydrodynamics simulator. First, we address the problem of slow convergence in hydrodynamical simulations of wind-blown bubbles with radiative cooling. We demonstrate that a learned cooling function can recover high-resolution dynamics in low-resolution simulations. Secondly, we train a convolutional neural network to correct 2D magnetohydrodynamics simulations of a specific blast wave problem. These case studies pave the way for the principled application of more general machine learning models inside astrophysical simulators. The code is available open source under `https://github.com/leo1200/eurips25corr`.

1 Introduction

From a single star to galactic or even cosmological scales, astrophysical fluid simulators are required to encompass multiple physical effects on a wide range of spatial and temporal scales [1]. Limitations in compute and memory constrain the resolutions possible at small scales. Under-resolving small-scale dynamics can impact the overall accuracy of a simulation. For instance, under-resolving the feedback regions around stars can lead to inaccurate momentum input [2, 3]. This necessitates the development of subgrid models [4].

Machine learning techniques and tools are opening new avenues for developing subgrid models. For instance Hirashima et al. [5] learn a surrogate model for supernovae and replace regions of interest in the simulation with the surrogate results. However, extracting and replacing whole regions in the simulation is not suitable for every application. For example, under-resolved cooling results in inaccurate dynamics throughout the simulation [3].

Differentiable simulators open another option: machine learning models can be trained inside the simulator to effectively model unresolved dynamics. Such solver-in-the-loop techniques[1] [6] allow for the trained model to be aware of its interaction with the simulator.

Traditional astrophysical solvers like [7, 8, 9, 10] are not differentiable and differentiable fluid solvers like [11, 12] lack the physics required for astrophysical applications. In astrophysical gases, magnetic fields, self-gravity, energy losses through radiation (radiative cooling) etc. can play important roles. An exception is `jf1uids`[2] [13] the first differentiable magnetohydrodynamical (MHD) simulator of its kind. Using `jf1uids` we present two novel solver-in-the-loop applications for astrophysics, an effective radiative cooling model restoring high-resolution dynamics on a wind-blown bubble and a corrector for an MHD blast wave problem which retains a divergence-free magnetic field.

[1] Solver-in-the-loop refers to the integration of the numerical solver, i.e. the differentiable simulator, into the training loop of a machine learning model.

[2] https://github.com/leo1200/jf1uids/

1st Workshop on Differentiable Systems and Scientific Machine Learning @ EurIPS 2025

The governing set of equations relevant for our test cases is (compare Equation Set 2 in [14] for the MHD part and Eq. 2 in [15] for the cooling)

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\nabla \cdot (\rho \vec{v}) \\
\frac{\partial \vec{v}}{\partial t} &= -(\vec{v} \cdot \nabla)\vec{v} - \frac{1}{\rho}\nabla p + \frac{1}{\rho}(\nabla \times \vec{B}) \times \vec{B} \\
\frac{\partial p}{\partial t} &= -\vec{v} \cdot \nabla p - \gamma p \nabla \cdot \vec{v} - (\gamma - 1)\frac{\rho^2}{\mu_e \mu_H}\Lambda(T) \\
\frac{\partial \vec{B}}{\partial t} &= \vec{\nabla} \times (\vec{v} \times \vec{B})
\end{aligned} \tag{1}$$

with density ρ, velocity $\vec{v}$, pressure p, magnetic field $\vec{B}$, adiabatic index γ, effective molecular weights μ_e, μ_H and a cooling function $\Lambda(T)$. The temperature is given by $T = \frac{p\mu}{\rho k}$ with mean molecular weight μ and Boltzmann constant k. The cooling function $\Lambda(T)$ (at solar metallicity) is taken from Schure et al. [16]. `jf1uids` uses a constrained transport scheme based on Pang and Wu [17] for the MHD, differentiation through a simulation with adaptive time stepping is enabled by optimal online checkpointing [18].[3] For radially symmetric simulations like the first test case the geometric formulation of the Euler equations from Crittenden and Balachandar [19] is applied.

2 Cooling functions trained to recover high-resolution dynamics

We consider the radially symmetric stellar wind setup of van Marle and Keppens [3] without magnetic fields ($\vec{B} = \vec{0}$). van Marle and Keppens [3] demonstrate that the evolution of wind-blown bubbles with radiative cooling is highly resolution dependent, converging only at resolutions above approximately 10^5 cells corresponding to 9 levels of adaptive mesh refinement in their setup.[4] This problem is illustrated in Fig. 4 in the appendix. van Marle and Keppens [3] conclude »Since the number of gridpoints would quickly become prohibitive if this simulation was carried out on a fixed grid, adaptive mesh refinement can be considered a necessity for simulations of this kind«. But even for a single star in a full three-dimensional simulation with adaptive mesh refinement, effective resolutions comparable to the radial one-dimensional cases will be prohibitive. We propose an alternative cost-efficient approach which works on low-resolution static grids: An effective cooling function $\Lambda_\Phi(T)$ parameterized by a neural network (in our case a multilayer perceptron with three hidden layers and 256 neurons each) is trained to restore high-resolution dynamics. $\Lambda_\Phi(T)$ is pre-trained on the Schure et al. [16] cooling.

The optimization goal is given by

$$\begin{aligned}
\phi^\star = \arg\min_\phi \mathcal{L}(\phi) = \arg\min_\phi \left\langle \left\| \frac{\mathbf{U}_i(\phi, t) - \mathbf{U}_i^{\text{ref}}(t)}{\mathbf{U}_{\max}^{\text{ref}}} \right\|_2^2 \right\rangle_{N_{\text{low-res}} > i > i_{\min}, t \in \mathcal{T}} \\
\mathcal{T} = \left\{ t_j = \frac{j}{N_{\text{snap}} + 1} t_{\max}^{\text{train}} \,\middle|\, j = 2, 3, \ldots, N_{\text{snap}} + 1 \right\}, t_{\max}^{\text{train}} = 1.25 \cdot 10^{12}\text{s}, N_{\text{snap}} = 5
\end{aligned} \tag{2}$$

where $\mathbf{U}_i = (\rho_i, v_i, p_i)$ is the fluid state of cell i, the reference $\mathbf{U}_i^{\text{ref}}$ is obtained by down-averaging a high-resolution simulation with $N_{\text{high-res}} = 10^4$ and $\mathbf{U}_{\max}^{\text{ref}}$ collects the maximum density, velocity, and pressure across time $t \in \mathcal{T}$ and $i > i_{\min}$ with $i_{\min} = \lfloor N_{\text{low-res}}/4 \rfloor$. In Sec. A.3 we discuss these choices in more detail and additionally consider the case of $N_{\text{high-res}} = 10^5$. For $N_{\text{low-res}} = 500$ taking 150 optimization steps with ADAM [20] takes ~ 1 hour on an NVIDIA GeForce RTX 2080 Ti. The resulting radial profiles and the loss over the simulation time are shown in Fig. 1. For

[3] The optimal online checkpointing was implemented in `https://github.com/patrick-kidger/equinox/blob/main/equinox/internal/_loop/checkpointed.py` by Patrick Kidger.

[4] This resolution sensitivity relates to the cooling instability - high density regions cool more rapidly ($\frac{dp}{dt}_{cool} \propto \rho^2 \Lambda(T)$), get compressed by the surrounding gas and therefore cool even more rapidly.

$N_{\text{low-res}} = 1000, 2000$ we provide the results in Fig. 5 and 6 in the appendix. The corresponding effective cooling curves are shown in Fig. 2. Apart from the higher numerical diffusivity expected at lower resolutions, the effective cooling curves enable the simulator to capture the high-resolution dynamics remarkably well. The existence of such an effective cooling function is already a very interesting result which can benefit future theoretical research on low-resolution cooling behavior and paves the way for more general effective models.

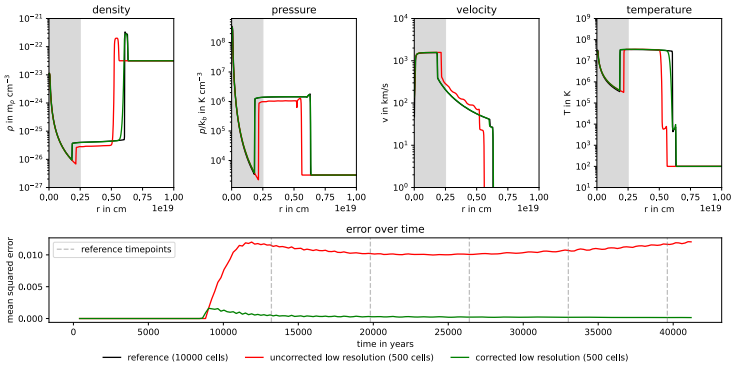

Figure 1: Radial profiles of density, pressure, velocity, and temperature (top row) comparing the high-resolution reference simulation ($N = 10000$ cells), the uncorrected low-resolution simulation ($N = 500$ cells), and the corrected low-resolution simulation using the learned effective cooling function. The shaded region is ignored for reasons discussed in Sec. A.3. The bottom panel shows the mean squared error between the low-resolution and high-resolution solutions over time.

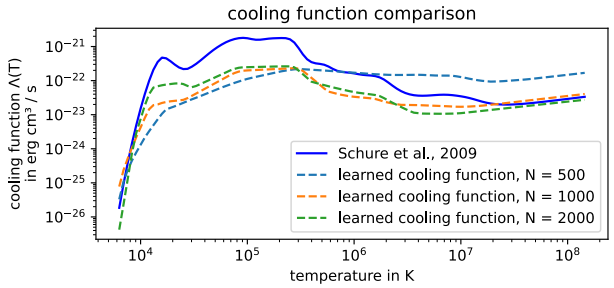

Figure 2: Comparison between the radiative cooling function from Schure et al. [16] and the learned effective cooling functions for simulations at different resolutions. The learned functions compensate for under-resolved high-resolution details.

3 Convolutional neural networks as correctors in magnetohydrodynamics simulations

Next we consider the two-dimensional MHD blast wave problem of [21, Sec. 3.3] without cooling ($\Lambda(T) = 0$). We train a corrector improving low-resolution dynamics to mimic high-resolution dynamics for the MHD blast. More specifically, for a low-resolution grid with 64^2 cells we train a convolutional neural network (CNN) $\mathbf{C} : \mathbb{R}^{7x64x64} \to \mathbb{R}^{7x64x64}$ (7 scalar fluid variables, $\rho, v_x, v_y, B_x, B_y, B_z, p$). To ensure that the CNN does not introduce any magnetic field divergence, we apply a discrete curl transformation to its magnetic field output, and call this divergence-free CNN $\tilde{\mathbf{C}}$. The correction step applied to the state array $\mathbf{U} \in \mathbb{R}^{7\times64\times64}$ after each MHD time-step then reads

$$\tilde{\mathbf{U}} = \mathbf{U} + \Delta t\, \tilde{\mathbf{C}}(\mathbf{U}) \tag{3}$$

In our experiment, the CNN consists of a single hidden layer with 16 channels. We train the CNN inside the simulator against a down-averaged 512^2 high-resolution reference simulation with 4

evaluation points in time between $t = 0.05$ and $t = 0.2$. The results are shown in Fig. 3. In the upper panel, a visual comparison between the uncorrected and corrected low-resolution and reference density fields is shown. The improvement due to the corrector is visually apparent. This visual impression is consistent with the mean squared error (MSE) of the corrected and uncorrected low-resolution results to the reference simulation over time in the lower panel. Until roughly $t = 0.01$ the MSEs are similar, at later times the corrected simulation features errors only a fraction of those from the uncorrected simulation. One epoch of training took ~ 8.8 seconds on an NVIDIA GeForce RTX 2080 Ti and we trained for 2000 epochs. The training loss over epochs is given in Fig. 8 in the appendix.

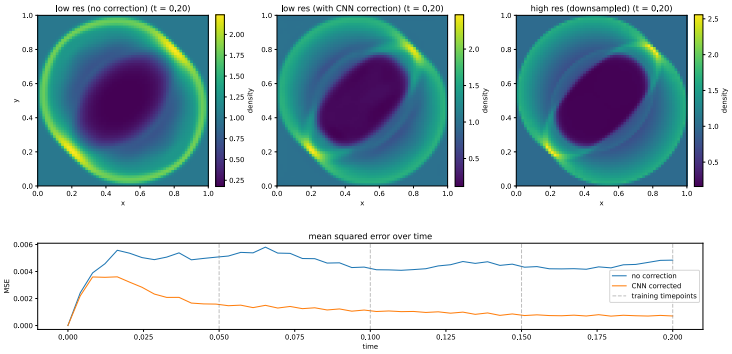

Figure 3: In the top row from left to right the final density field for an MHD blast setup is shown for a low-resolution simulation with 64^2 cells, a low-resolution with 64^2 cells but a corrective convolutional neural network applied to the state after each time step, and a high-resolution simulation down-averaged from 512^2 to 64^2 cells. In the bottom panel, the mean squared error (MSE) over time of the uncorrected and corrected low-resolution simulations with respect to the down-averaged high-resolution simulation is shown.

4 Conclusion and outlook

We presented two promising solver-in-the-loop applications in astrophysical fluid dynamics, specifically machine learning models trained inside the astrophysical fluid simulator `jf1uids` to improve low-resolution dynamics. Firstly, we experimentally demonstrated that there exists a learnable effective cooling function which can recover high-resolution dynamics in low-resolution radially symmetric stellar wind simulations with radiative cooling, challenging the need for adaptive mesh refinement as stated by van Marle and Keppens [3] regarding this specific problem. Secondly, we trained an explicit corrector term for two-dimensional MHD blast wave dynamics retaining a divergence-free magnetic field. This corrector only operates on the current fluid state. To our knowledge, our MHD corrector is the first published application of automatic differentiation through a magnetohydrodynamical simulation.

The next step is to generalize such models across resolutions and test problems. We've shown that a specific cooling function can lead to improved dynamics. In the future, a neural operator approach might be able to provide problem-specific cooling functions. Additionally, it would be interesting to transfer the effective cooling function approach to wind-blown bubbles in turbulent media with and without magnetic fields, potentially alleviating the convergence challenges reported by Lancaster et al. [2]. We would also like to highlight that machine-learning-based adaptations of physical terms already present in the solver, like our effective cooling function, are easy to integrate into existing non-differentiable simulators after training. Concluding, we hope that these experiments can motivate further research regarding the benefits of differentiable simulators for astrophysical (magneto)hydrodynamics.

References

[1] Romain Teyssier. »Grid-based hydrodynamics in astrophysical fluid flows«. In: *Annual Review of Astronomy and Astrophysics* 53.1 (2015), pages 325–364.

[2] Lachlan Lancaster et al. *Geometry, Dissipation, Cooling, and the Dynamical Evolution of Wind-Blown Bubbles*. 2024. URL: https://arxiv.org/abs/2405.02396.

[3] Allard Jan van Marle and Rony Keppens. »Radiative cooling in numerical astrophysics: The need for adaptive mesh refinement«. In: *Computers and Fluids* 42.1 (2011), pages 44–53. DOI: https://doi.org/10.1016/j.compfluid.2010.10.022. URL: https://www.sciencedirect.com/science/article/pii/S0045793010002914.

[4] Greg Stinson et al. »Star formation and feedback in smoothed particle hydrodynamic simulations – I. Isolated galaxies«. In: *Monthly Notices of the Royal Astronomical Society* 373.3 (Nov. 2006), pages 1074–1090. DOI: 10.1111/j.1365-2966.2006.11097.x. URL: http://dx.doi.org/10.1111/j.1365-2966.2006.11097.x.

[5] Keiya Hirashima et al. *ASURA-FDPS-ML: Star-by-star Galaxy Simulations Accelerated by Surrogate Modeling for Supernova Feedback*. 2025. URL: https://arxiv.org/abs/2410.23346.

[6] Kiwon Um et al. *Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers*. 2021. URL: https://arxiv.org/abs/2007.00016.

[7] Volker Springel. »E pur si muove:Galilean-invariant cosmological hydrodynamical simulations on a moving mesh«. In: *Monthly Notices of the Royal Astronomical Society* 401.2 (Jan. 2010), pages 791–851. DOI: 10.1111/j.1365-2966.2009.15715.x. URL: http://dx.doi.org/10.1111/j.1365-2966.2009.15715.x.

[8] J. M. Stone et al. »Athena: A New Code for Astrophysical MHD«. In: *The Astrophysical Journal Supplement Series* 178.1 (Sept. 2008), pages 137–177. DOI: 10.1086/588755. URL: http://dx.doi.org/10.1086/588755.

[9] R. Teyssier. »Cosmological hydrodynamics with adaptive mesh refinement: A new high resolution code called RAMSES«. In: *Astronomy and Astrophysics* 385.1 (Apr. 2002), pages 337–364. DOI: 10.1051/0004-6361:20011817. URL: http://dx.doi.org/10.1051/0004-6361:20011817.

[10] Timothée David–Cléris, Guillaume Laibe, and Yona Lapeyre. *The Shamrock code: I- Smoothed Particle Hydrodynamics on GPUs*. 2025. URL: https://arxiv.org/abs/2503.09713.

[11] Deniz A. Bezgin, Aaron B. Buhendwa, and Nikolaus A. Adams. *JAX-Fluids 2.0: Towards HPC for Differentiable CFD of Compressible Two-phase Flows*. 2024. URL: https://arxiv.org/abs/2402.05193.

[12] Mohammadmehdi Ataei and Hesam Salehipour. »XLB: A differentiable massively parallel lattice Boltzmann library in Python«. In: *Computer Physics Communications* 300 (2024), page 109187.

[13] Leonard Storcks and Tobias Buck. *Differentiable Conservative Radially Symmetric Fluid Simulations and Stellar Winds – jf1uids*. 2024. URL: https://arxiv.org/abs/2410.23093.

[14] David A. Clarke. *A Primer on Magnetohydrodynamics*. Lecture Notes. Copyright © David A. Clarke, 2015. Halifax, Nova Scotia, Canada: Saint Mary's University, Department of Astronomy and Physics, Feb. 2015. URL: https://www.ap.smu.ca/~dclarke/home/documents/byDAC/mhd_primer.pdf.

[15] R. H. D. Townsend. »AN EXACT INTEGRATION SCHEME FOR RADIATIVE COOLING IN HYDRODYNAMICAL SIMULATIONS«. In: *The Astrophysical Journal Supplement Series* 181.2 (Mar. 2009), page 391. DOI: 10.1088/0067-0049/181/2/391. URL: https://dx.doi.org/10.1088/0067-0049/181/2/391.

[16] KM Schure et al. »A new radiative cooling curve based on an up-to-date plasma emission code«. In: *Astronomy and Astrophysics* 508.2 (2009), pages 751–757.

[17] Dongwen Pang and Kailiang Wu. *Provably Positivity-Preserving Constrained Transport (PPCT) Second-Order Scheme for Ideal Magnetohydrodynamics*. 2024. URL: https://arxiv.org/abs/2410.05173.

[18] Philipp Stumm and Andrea Walther. »New algorithms for optimal online checkpointing«. In: *SIAM Journal on Scientific Computing* 32.2 (2010), pages 836–854.

[19] P. E. Crittenden and S. Balachandar. »The impact of the form of the Euler equations for radial flow in cylindrical and spherical coordinates on numerical conservation and accuracy«. In: *Shock Waves* 28.4 (Mar. 2018), pages 653–682. DOI: 10.1007/s00193-017-0784-y. URL: http://dx.doi.org/10.1007/s00193-017-0784-y.

[20] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. URL: https://arxiv.org/abs/1412.6980.

[21] Ruediger Pakmor, Andreas Bauer, and Volker Springel. »Magnetohydrodynamics on an unstructured moving grid: MHD on an unstructured moving grid«. In: *Monthly Notices of the Royal Astronomical Society* 418.2 (Sept. 2011), pages 1392–1401. DOI: 10.1111/j.1365-2966.2011.19591.x. URL: http://dx.doi.org/10.1111/j.1365-2966.2011.19591.x.

A Technical Appendices and Supplementary Material

A.1 Problem setting of slow convergence of wind-blown bubble simulations with radiative cooling

Density profiles for wind-blown bubbles with and without radiative cooling at different resolutions are shown in Fig. 4.

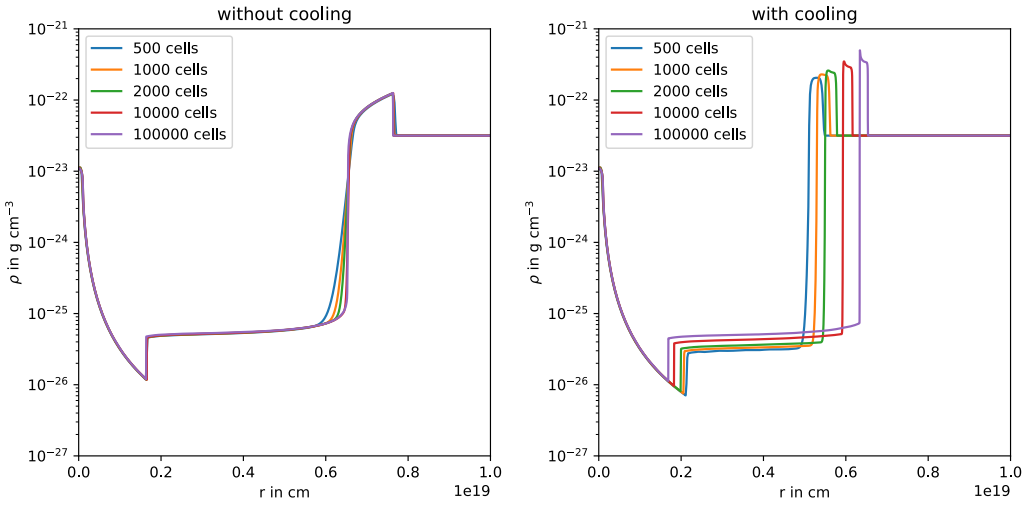

Figure 4: Density profiles for stellar wind simulations following the setup of van Marle and Keppens [3] at $t = 1.25 \cdot 10^{12}$ s without (left panel) and with cooling (right panel) for different resolutions of $N = \{500, 1000, 2000, 10000\}$ cells. The position of the shock and the profile of the swept-up gas are largely independent of the resolution in the case without cooling but only converge slowly if radiative cooling is introduced [3].

A.2 Corrected cooling dynamics at different resolutions

In addition to the corrected results of Fig. 1 for $N_{\text{low-res}} = 500$ we show the results for $N_{\text{low-res}} = 1000$ in Fig. 5 and $N_{\text{low-res}} = 2000$ in 6.

A.3 Experimental setup choices in the cooling correction

We would like to explain two choices here, the resolution of our reference simulations $N_{\text{high-res}} = 10^4$ and the spatial starting index for the loss $i_{\min} = \lfloor N_{\text{low-res}}/4 \rfloor$. A reference resolution of $N_{\text{high-res}} = 10^4$ seemed distinct enough from $N_{\text{low-res}} = 500$ to be interesting while still being cheap (on the order of a minute on an NVIDIA GeForce RTX 2080 Ti). The resolution of convergence $N_{\text{high-res}} = 10^6$ takes on the order of half an hour on the same machine. We are currently looking into $N_{\text{high-res}} = 10^6$ as a reference resolution. Optimization seems to be harder in this case but including the density as a further input to the cooling network might help. This is ongoing research but we provide first results in Fig. 7. Regarding $i_{\min} = \lfloor N_{\text{low-res}}/4 \rfloor$ we made this choice as

- up to the reverse shock the pressure profile depends on the injection mechanism of the stellar wind, in our case this is always a thermal energy injection into cells within a radius of $0.05L$ with L being the box length, but it might generally vary
- in terms of total volume and mass this innermost region is relatively unimportant (spherical simulation) but weighting with the volume in the loss seemed to put too much weight on the outer regions, so masking this innermost region seemed like a fair compromise

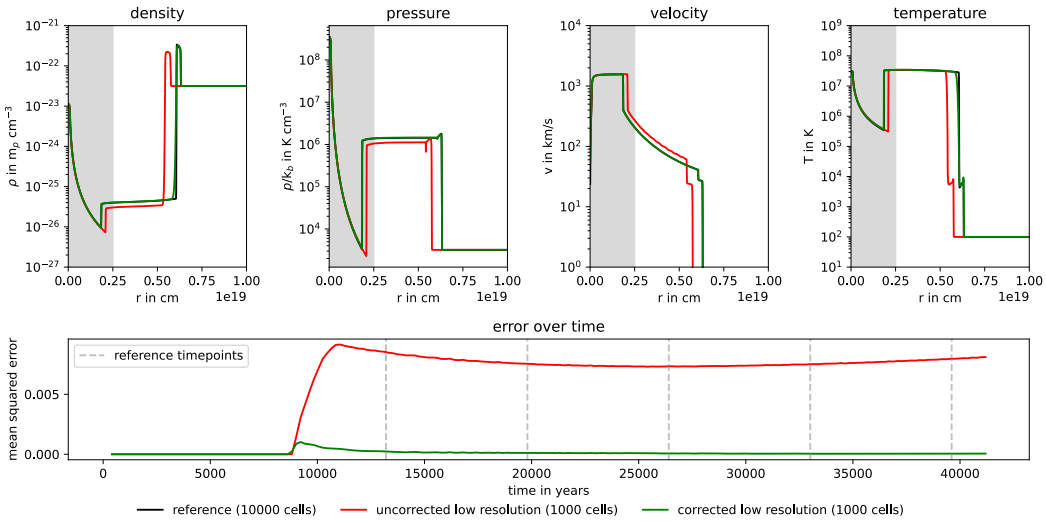

Figure 5: Radial profiles of density, pressure, velocity, and temperature (top row) comparing the high-resolution reference simulation ($N = 10000$ cells), the uncorrected low-resolution simulation ($N = 1000$ cells), and the corrected low-resolution simulation using the learned effective cooling function. The bottom panel shows the mean squared error between the low-resolution and high-resolution solutions over time.

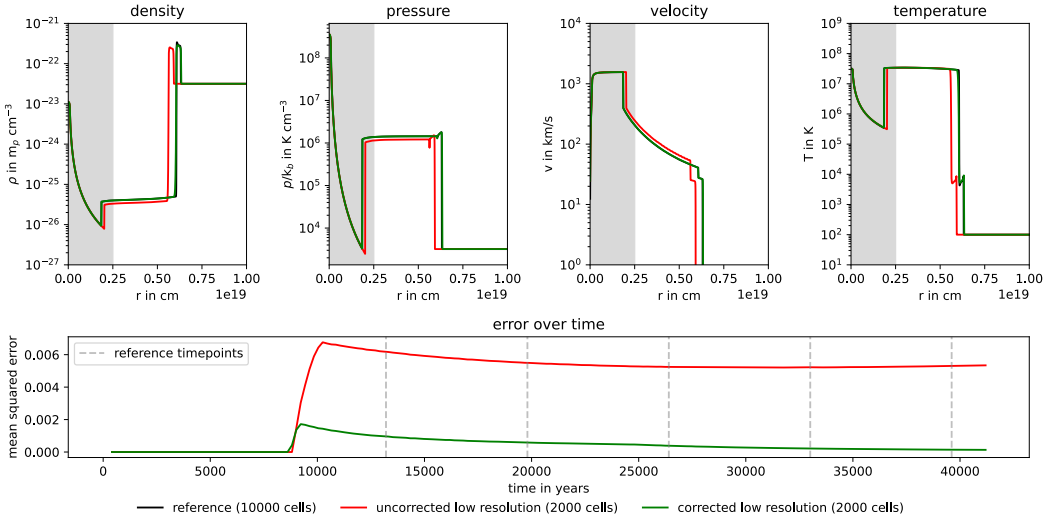

Figure 6: Radial profiles of density, pressure, velocity, and temperature (top row) comparing the high-resolution reference simulation ($N = 10000$ cells), the uncorrected low-resolution simulation ($N = 2000$ cells), and the corrected low-resolution simulation using the learned effective cooling function. The bottom panel shows the mean squared error between the low-resolution and high-resolution solutions over time.

A.4 Training loss of the convolutional neural network corrector for the magnetohydrodynamics simulations

The training loss over epochs for the MHD corrector is given in Fig. 8.

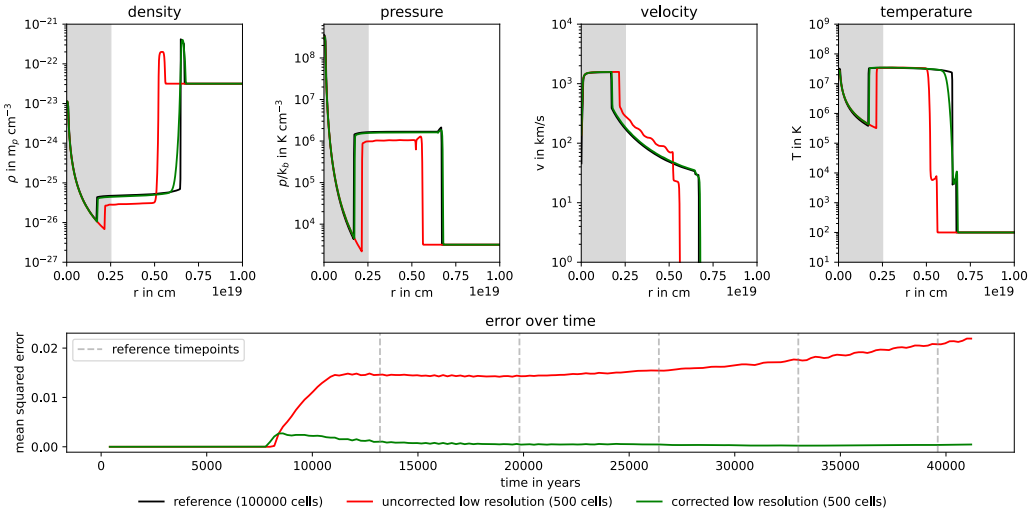

Figure 7: Radial profiles of density, pressure, velocity, and temperature (top row) comparing the high-resolution reference simulation ($N = 10000$ cells), the uncorrected low-resolution simulation ($N = 2000$ cells), and the corrected low-resolution simulation here using a temperature and density dependent learned cooling function. Adding the density simplified the training, but we think that by adapting the training procedure or network architecture a purely temperature dependent solution might also be found here.

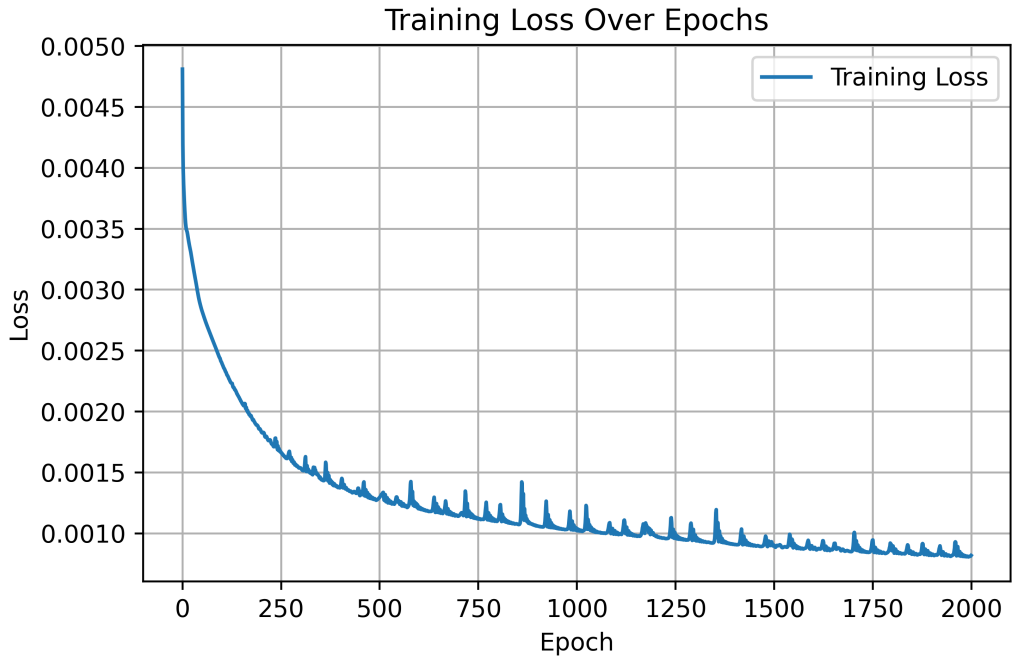

Figure 8: Training loss over epochs of the convolutional neural network corrector for the magnetohydrodynamic (MHD) blast wave simulation.