Neural Infalling Clouds: Increasing the Efficacy of Subgrid Models and Scientific Equation Discovery using Neural ODEs and Symbolic Regression

Brent Tan
Center for Computational Astrophysics
Flatiron Institute
New York, NY 10010
btan@flatironinstitute.org

Abstract

Galactic systems are inherently multiphase. Understanding the roles of the various phases and their interactions is the next key step towards a more complete picture of galaxy evolution. A major challenge is that the transport and dynamics of cold clouds is governed by complex small scale processes. Large scale models thus require subgrid prescriptions in the form of models validated with small scale simulations. In this work, we explore using neural ordinary differential equations (NODEs), which embed a neural network, to more accurately model these clouds. We apply Symbolic Regression (SR) to potentially discover new insights into the physics of cloud-environment interactions. We test this on both generated mock data and actual simulation data. We also extend the model to include more than one neural term. We find that NODEs in tandem with SR can be used to enhance the accuracy and efficiency of subgrid models, and/or discover the underlying equations to improve generality and scientific understanding. While our system is a relatively simple one, we highlight the potential of this scientific machine learning approach as a natural extension to the traditional modelling paradigm, both for the development of semi-analytic models and for physically intepretable equation discovery in complex non-linear systems.

1 Introduction

A key open problem in galaxy evolution is the discrepancy between observed star formation rates (SFRs) in galaxies and their available gas reservoirs. The observed SFRs are unsustainable over cosmic timescales, indicating a need for continuous gas accretion to fuel star formation [7, 13, 22]. There is observational evidence for the existence of such inflows, in the form of 'high-velocity' and 'intermediate-velocity' clouds [23] and fountain-like transport [25, 10] where clouds are lifted out of the disk by supernova-driven winds before falling back. One of the primary challenges in modeling these interactions is accurately simulating the complex interplay between the cold clouds and the hot ambient medium. Hydrodynamic instabilities can shred the clouds into smaller fragments, potentially destroying them. The cloud's ability to survive depends on various factors [12], including its size, density, and the efficiency of radiative cooling within the turbulent mixing layers formed at the interface between the cold and hot gases [1, 9, 30]. Accurately capturing these small-scale processes requires prohibitively high spatial and temporal resolution. One solution is to develop subgrid models that can capture the effects of unresolved processes. Such models can be formulated and tested against smaller scale high resolution simulations [29, 26], or in semi-analytical models [8]. Improving the fidelity of these subgrid models is thus critical towards advancing galaxy simulations.

Machine Learning and the Physical Sciences Workshop, NeurIPS 2024.

Neural ODEs (NODEs) are ODEs which include neural networks (NNs) in their parameterizations [4, 16]. They have been applied in a wide range of domains, including neuroscience [18], fluid mechanics [3], climate science [15] and epidemiology [20]. NODEs offer a way to combine traditional models with deep learning and its strength in serving as high-capacity function approximators [21, 24], and hence improve the predictive power of the model. This opens the door to using Symbolic Regression (SR) effectively to discover new insights into the underlying physics of the system [6] by limiting the scope of the physical process the NN captures (challenges of SR include how quickly the search space and difficulty in interpretability scale with complexity). In this work, we use NODEs to improve the predictive accuracy of subgrid models for infalling clouds, and SR to potentially discover new insights into the physics of cloud-environment interactions by extracting learned equations.

2 Problem Statement

The evolution of an infalling cloud growing via mixing-cooling driven accretion can be described by the following set of coupled ODEs [31]:

$$\frac{\mathrm{d}z}{\mathrm{d}t} = v, \qquad \frac{\mathrm{d}(mv)}{\mathrm{d}t} = mg - \frac{1}{2}\rho_{\rm hot} v^2 C_0 A_{\rm cross}, \qquad \frac{\mathrm{d}m}{\mathrm{d}t} = \frac{m}{t_{\rm grow}(z, v, m)} \tag{1}$$

where z, v and m represent the distance fallen, the velocity and the mass of the cloud respectively. $t_{\rm grow} \equiv m/\dot{m}$ is the growth timescale, g is the gravitational acceleration, C_0 is the drag coefficient, $\rho_{\rm hot}$ is the density of the background, and $A_{\rm cross}$ is the cross-sectional area of the cloud. The crucial term in this model is the growth timescale $t_{\rm grow}$ which is set by the physics of the turbulent radiative mixing layers that govern the rate of exchange of mass, momentum, and energy between the cloud and the background medium (see Tan et al. [31] for a detailed formulation). Clouds of different initial sizes evolve at different rates since their surface area to volume ratios initially differ, and then experience a change in geometry that is captured in the expression for $t_{\rm grow}$. However, a key uncertainty in $t_{\rm grow}$ is the time it takes for the onset of turbulence through instabilities. Since this affects the initial cloud evolution, it is important for matching the subsequent evolution of the cloud. The model accounts for this by adding a weight factor $w_{\rm kh}$, so that $t_{\rm grow} \to t_{\rm grow}/w_{\rm kh}(t, z, v, m)$. In the absence of a more detailed physical model, Tan et al. [31] used a simple ansatz for $w_{\rm kh}$ based on the the KH timescale $t_{\rm kh} \propto \sqrt{\chi} r/v$ [19], where χ is the ratio of the densities of cloud to background and r is the initial cloud radius:

$$w_{\rm kh}(t) \propto \min\left(1, \frac{t}{t_{\rm kh}}\right). \tag{2}$$

which amounts to the turbulent velocity growing linearly with time over the instability growth time until fully developed. In this work, we instead use a NN embedded in the ODEs to learn this weight factor. We then apply SR on the NN to discover the underlying learned equation.

3 Methods

We first generate mock data using the ansatz assumed above (Equation 2) to train and test our NODE model. The involves generating solutions to Equation 1 assuming that Equation 2 is the right description of $w_{\rm kh}$. We then apply SR to the learned model to see if we can recover Equation 2, since we used it to generate the 'ground truth'. This tests the effectiveness of the approach on a noiseless dataset which exactly matches the true underlying model.

We then apply the same process to data taken from high resolution 3D simulations, carried out with the publicly available MHD code Athena++ [28]. Our simulation data suite consists of 13 simulations [31], 7 of which are used for training and 6 for testing. We use the same initial conditions for all simulations, with the only difference being the initial cloud size. The cloud sizes range from 30 pc to 1 kpc. Each simulation generates 1000 data points spanning a time range of up to 225 Myr. We keep track of v, z, and m. The dataset is small since running large high resolution 3D simulations is computationally expensive. However, we show that we are still able to achieve good model performance.

To construct, solve, and train the NODE, we use the `JAX` framework [2] heavily, along with the software libraries `Diffrax` [16] for differential equations and `Equinox` [17] for NNs. This enables us to train on a single CPU core on the order of minutes. Our NODE model consists of Equations 1

with the weight factor replaced by a NN. The NN is a multilayer perceptron with 3 hidden layers of 32 nodes each. Each layer uses a GELU activation function. Weights are initialized using a Xavier/Glorot initialization scheme. We use the `ADAMW` optimizer, and train the network in 2 stages. In the first stage, we train for 1300 steps with a learning rate (LR) of 2×10^{-3} on the first 20% of the time series. In the second stage, we train for 3500 steps with a LR of 3×10^{-3} on the entire training data. This is a common strategy in training NODEs to avoid getting initially trapped in local minima. The NN takes in (z, v, m) and the initial cloud size r as input. All input data into the NN is normalized, and the mass which can grow by orders of magnitude over the simulation is log-transformed. The output is a single value corresponding to the weight factor.

We calculate our loss function as the mean squared error between the predicted and actual mass solutions. We find that only including the mass in instead of all changing variables (z, v and m) improves training stability since the mass is the quantity most directly sensitive to the growth rate. We also multiply the loss by a coefficient which decreases over time. This coefficient starts at 1 and then decreases linearly over time to a low of 0.1. This focuses the model on earlier times, which improves the training process in NODEs where late time values are sensitive to earlier ones. We numerically solve the ODEs using Tsitouras' 5/4 method [32], which is a 4th order explicit Runge-Kutta method when using adaptive step sizing, which we employ via a PID controller [27]. For SR, we use the package `PySR` [5]. The model with the highest score is selected (score is defined to be the negated derivative of the log-loss with respect to complexity). We allow the common binary operators along with the power and minimum operators, and the logarithm and exponentiation unary operators. We use a population size of 100 and ensure that we iterate for a sufficient amount of time to get a converged model. We also include physical units to so as to favor dimensionally consistent equations.

4 Results

Performance on mock data We first train on the generated mock data to evaluate the performance of the NODE on a known hidden function. In the top panel of Figure 1, we compare the trained NODE predictions against the 'ground truth'. The NODE does well for cloud sizes that fall within the range spanned by the training data. However, it does a poorer job at predicting the weight factor and hence the evolution of test clouds with sizes that are outside the range of the training data. This is unsurprising — out-of-distribution generalization is a well-known challenge in scientific machine learning. We verify that applying SR to the NODE discovers the hidden equation used to generate the data (Equation (2)), giving an almost identical equation. SR hence improves the generalization of the NODE. It also makes the results intepretable — we are able to interpret the discovered equation as with any other physical equation we might have derived using traditional methods.

Performance on simulation data We then train the NODE on actual simulation data. In the second row of Figure 1, we compare NODE predictions against the simulation data. The data from the simulation is more limited for smaller clouds due to box sizes in the simulation. Since our loss function is based on the mass, the NODE is able to capture the evolution of the mass very well, but less so for the velocity. Due to the constrained scope of the NN in our ODEs, we can infer this is likely due to the model not accounting for drag well, which only affects the velocity directly. This indicates that we need to better model the drag in order to increase the fidelity of the model. As before, we apply SR and find that the SR still recovers a very similar equation to Equation (2). This suggests that this was a good choice in the original work despite its simplicity. The third row of Figure 1 shows the SR result: velocity predictions are more accurate but mass predictions are worse.

A second neural network Since we attribute the poorer accuracy in velocity prediction to the loss function (physically motivated by the fact that turbulent mixing only directly affects mass growth) and the simplistic representation of drag in our equations (we assumed $C_0 = 1$; for growing clouds, this term is usually small [31]), we can further improve our subgrid model by using a second NN. We freeze the trained model from the previous section and replace C_0 with a second NN. Since the baseline model already works well, we train on the full time evolutions from the start. We employ early stopping to avoid overfitting. We compute the loss with respect to the v instead of the m, since C_0 only directly affects the cloud's velocity. The results can be seen in the bottom row of Figure 1, which shows that the model now does a better job at matching the velocity evolution. Training separately is more efficient since we were able to use two separate loss functions, with the first NN capturing the more important physical process in the system.

Once again, we use SR to recover a physical expression for the drag coefficient. This gives the expression $C_0 = \min(0.92/v, 8.6)$. The $\min$ prevents the function $1/v$ diverging at $t = 0$. Unlike the previous equation for the weight factor which was already known to us (discovered using manual methods in the original paper), this is a new equation that was not used in the original model, which assumed a constant value for C_0. However, it can be seen that using this expression for C_0 improves the fidelity of the model when compared to simulation results. From this, we can also make physical interpretations about how conventional drag on the cloud is evolving in our simulation. Looking into the literature, this inverse scaling of C_0 with velocity is consistent with a range of empirical data for spheres which find that the drag coefficient scales inversely with the Reynolds number, which itself scales linearly with velocity [11]. Furthermore, while the cloud begins as spherical object, it develops a long tail and becomes more streamlined as it falls and grows [31]. We hence expect C_0 to eventually approach some very small value $\ll 1$, which is consistent with the predictions of the NN in the bottom left panel of Figure 1.

5 Conclusion

Astronomical simulations traditionally rely on subgrid models to capture important unresolved processes. We have shown that NODEs can be used to improve the predictive power of such models and better capture the complex physics involved even with limited training data. Minimal resources are required compared to other machine learning methods. They are both intepretable and physically motivated, while still being flexible and predictive. We were also able to effectively employ SR for scientific equation discovery. We additionally demonstrated that we can train multiple neural terms for different physical processes, each with its own physically tailored loss functions. This further improves the accuracy of the model and increases the effectiveness of using SR. While SR is a strong tool for interpretability, we did not test how generalizable the discovered equations are for out of distribution simulation data. This would be a good avenue for future work, along with exploring how effective such methods would be for more complex systems.

In conclusion, neural ODEs in tandem with Symbolic Regression are natural extensions of traditional methods of formulating subgrid/semi-analytic models and should be seen as valuable parts of the modeling toolkit, both to develop more accurate models of complex systems and to enhance the scientific discovery process.

Acknowledgements

We thank Shirley Ho, Dustin Nguyen and Stephanie Tonnesen for helpful and insightful discussions. We thank the reviewers for their comments which have helped to improve the paper. Besides the packages mentioned in the text, we have also made use of the software packages `matplotlib` [14] and `numpy` [33] whose communities we thank for continued development and support. BT is supported by the Simons Foundation through the Flatiron Institute.

Data availability

We make our code and all data used in this work public in a GitHub repository (github.com/zunyibrt/Neural-Infalling-Cloud-Equations).

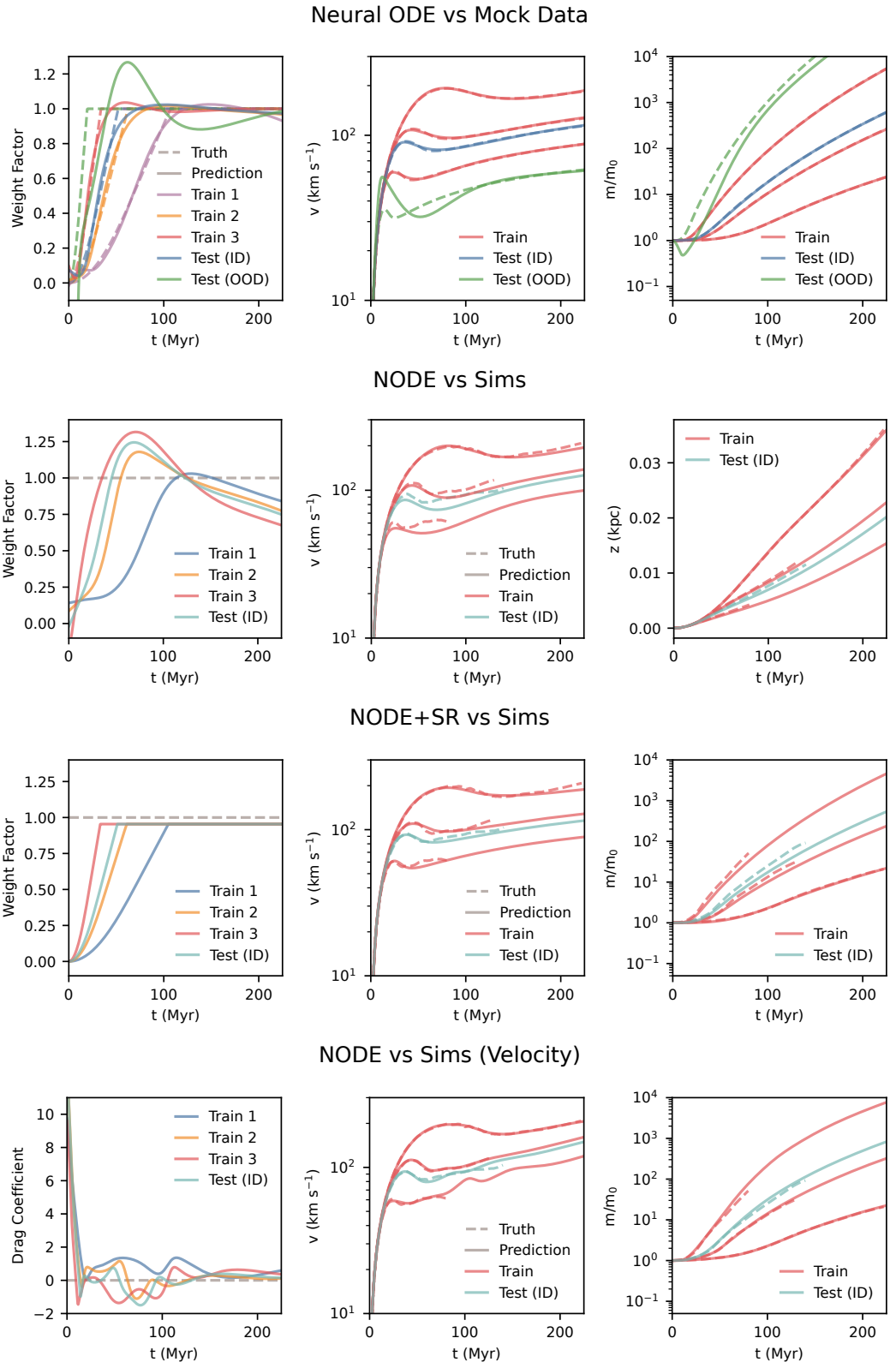

Figure 1: Columns show evolutions curves for (from left to right): weight factor/drag coefficient, velocity, and mass. Solid lines show model predictions, while generated/simulation data are represented by dashed lines. Test data is either in distribution (ID) or out of distribution (OOD). Each curve corresponds to a different cloud size. From top to bottom, the rows show results for training and testing on: mock data, simulated data, with symbolic regression on simulated data, and with a second neural network on simulated data.

References

[1] M. C. Begelman and A. C. Fabian. Turbulent mixing layers in the interstellar and intracluster medium. , 244:26P–29P, May 1990.

[2] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL `http://github.com/google/jax`.

[3] S. L. Brunton, B. R. Noack, and P. Koumoutsakos. Machine learning for fluid mechanics. *Annual review of fluid mechanics*, 52(1):477–508, 2020.

[4] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural Ordinary Differential Equations. *arXiv e-prints*, art. arXiv:1806.07366, June 2018. doi: 10.48550/arXiv.1806.07366.

[5] M. Cranmer. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl, May 2023. URL `http://arxiv.org/abs/2305.01582`. arXiv:2305.01582 [astro-ph, physics:physics].

[6] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho. Discovering symbolic models from deep learning with inductive biases. *NeurIPS 2020*, 2020.

[7] D. K. Erb. A Model for Star Formation, Gas Flows, and Chemical Evolution in Galaxies at High Redshifts. , 674(1):151–156, Feb. 2008. doi: 10.1086/524727.

[8] D. B. Fielding and G. L. Bryan. The Structure of Multiphase Galactic Winds. , 924(2):82, Jan. 2022. doi: 10.3847/1538-4357/ac2f41.

[9] D. B. Fielding, E. C. Ostriker, G. L. Bryan, and A. S. Jermyn. Multiphase Gas and the Fractal Nature of Radiative Turbulent Mixing Layers. , 894(2):L24, May 2020. doi: 10.3847/2041-8213/ab8d2c.

[10] F. Fraternali and J. J. Binney. Accretion of gas on to nearby spiral galaxies. , 386(2):935–944, May 2008. doi: 10.1111/j.1365-2966.2008.13071.x.

[11] W. R. Goossens. Review of the empirical correlations for the drag coefficient of rigid spheres. *Powder Technology*, 352:350–359, 2019.

[12] M. Gronke and S. P. Oh. The growth and entrainment of cold gas in a hot wind. , 480(1):L111–L115, Oct. 2018. doi: 10.1093/mnrasl/sly131.

[13] A. M. Hopkins, N. M. McClure-Griffiths, and B. M. Gaensler. Linked Evolution of Gas and Star Formation in Galaxies Over Cosmic History. , 682(1):L13, July 2008. doi: 10.1086/590494.

[14] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9 (3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[15] J. Hwang, J. Choi, H. Choi, K. Lee, D. Lee, and N. Park. Climate Modeling with Neural Diffusion Equations. *arXiv e-prints*, art. arXiv:2111.06011, Nov. 2021. doi: 10.48550/arXiv.2111.06011.

[16] P. Kidger. On Neural Differential Equations. *arXiv e-prints*, art. arXiv:2202.02435, Feb. 2022. doi: 10.48550/arXiv.2202.02435.

[17] P. Kidger and C. Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

[18] T. D. Kim, T. Z. Luo, J. W. Pillow, and C. D. Brody. Inferring latent dynamics underlying neural population activity via neural differential equations. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5551–5561. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/kim21h.html`.

[19] R. I. Klein, C. F. McKee, and P. Colella. On the Hydrodynamic Interaction of Shock Waves with Interstellar Clouds. I. Nonradiative Shocks in Small Clouds. , 420:213, Jan. 1994. doi: 10.1086/173554.

[20] B. Kuwahara and C. T. Bauch. Predicting covid-19 pandemic waves with biologically and behaviorally informed universal differential equations. *medRxiv*, 2023. doi: 10.1101/2023.03.11.23287141. URL `https://www.medrxiv.org/content/early/2023/03/16/2023.03.11.23287141`.

[21] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. , 521(7553):436–444, May 2015. doi: 10.1038/nature14539.

[22] M. E. Putman, J. E. G. Peek, A. Muratov, O. Y. Gnedin, W. Hsu, K. A. Douglas, C. Heiles, S. Stanimirovic, E. J. Korpela, and S. J. Gibson. The Disruption and Fueling of M33. , 703(2): 1486–1501, Oct. 2009. doi: 10.1088/0004-637X/703/2/1486.

[23] M. E. Putman, J. E. G. Peek, and M. R. Joung. Gaseous Galaxy Halos. , 50:491–529, Sep 2012. doi: 10.1146/annurev-astro-081811-125612.

[24] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, and A. Edelman. Universal Differential Equations for Scientific Machine Learning. *arXiv e-prints*, art. arXiv:2001.04385, Jan. 2020. doi: 10.48550/arXiv.2001.04385.

[25] P. R. Shapiro and G. B. Field. Consequences of a New Hot Component of the Interstellar Medium. , 205:762–765, May 1976. doi: 10.1086/154332.

[26] M. C. Smith, D. B. Fielding, G. L. Bryan, C.-G. Kim, E. C. Ostriker, R. S. Somerville, J. Stern, K.-Y. Su, R. Weinberger, C.-Y. Hu, J. C. Forbes, L. Hernquist, B. Burkhart, and Y. Li. ARKENSTONE - I. A novel method for robustly capturing high specific energy outflows in cosmological simulations. , 527(1):1216–1243, Jan. 2024. doi: 10.1093/mnras/stad3168.

[27] G. Söderlind. Digital Filters in Adaptive Time-Stepping. *ACM Transactions on Mathematical Software*, 20(1):1–26, 2003.

[28] J. M. Stone, K. Tomida, C. J. White, and K. G. Felker. The athena++ adaptive mesh refinement framework: Design and magnetohydrodynamic solvers. , 249(1):4, jul 2020. doi: 10.3847/1538-4365/ab929b.

[29] B. Tan and D. B. Fielding. Cloud atlas: navigating the multiphase landscape of tempestuous galactic winds. , 527(4):9683–9714, Feb. 2024. doi: 10.1093/mnras/stad3793.

[30] B. Tan, S. P. Oh, and M. Gronke. Radiative mixing layers: insights from turbulent combustion. , 502(3):3179–3199, Apr. 2021. doi: 10.1093/mnras/stab053.

[31] B. Tan, S. P. Oh, and M. Gronke. Cloudy with a chance of rain: accretion braking of cold clouds. , 520(2):2571–2592, Apr. 2023. doi: 10.1093/mnras/stad236.

[32] C. Tsitouras. Runge–kutta pairs of order 5 (4) satisfying only the first column simplifying assumption. *Computers & Mathematics with Applications*, 62(2):770–775, 2011.

[33] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.