Reduced Order Model for Chemical Kinetics: A case study with Primordial Chemical Network

Kwok Sun Tang [1] **Matthew Turk** [1 2]

Abstract

Chemical kinetics plays an important role in governing the thermal evolution in reactive flows problems. The possible interactions between chemical species increase drastically with the number of species considered in the system. Various ways have been proposed before to simplify chemical networks with an aim to reduce the computational complexity of the chemical network. These techniques oftentimes require domain-knowledge experts to handcraftedly identify important reaction pathways and possible simplifications. Here, we propose a combination of autoencoder and neural ordinary differential equation to model the temporal evolution of chemical kinetics in a reduced subspace. We demonstrated that our model has achieved a close-to 10-fold speed-up compared to commonly used astro-chemistry solver for a 9-species primordial network, while maintaining 1 percent accuracy across a wide-range of density and temperature.

1. Introduction

Chemistry plays a key role in regulating the cooling and the thermodynamical properties of the gas in astrophysical envirnoment. Chemical species are fundamental to our understanding of the formation of stars with different metalicities (Omukai, 2000; Omukai et al., 2005), interstellar medium (Gong et al., 2017), protoplanetary disk evolution (Kamp et al., 2017), early universe (Richings et al., 2014a;b; Smith et al., 2017) and etc.

The chemical network kinetics can in general be described as a initial value problem

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}),$$
$$\mathbf{y}(t) = \mathbf{y}(0) + \int_0^t f(t', \mathbf{y}(t'))dt',$$

where $\mathbf{y}(t)$, the state vector, corresponds to the chemical abundance, and the thermal energy at a given instant in time t. f specifies the interactions and dynamics among different species and guides the evolution of the state vector with time.

Scientific communities have been taking advantage of the recent advancement in deep learning (Champion et al., 2019; Vinuesa & Brunton, 2021; Brunton et al., 2020; Vlachas et al., 2022) to model dynamical systems. Many, if not all, of these approaches seek to identify the low-dimensional coordinates for an inherently high-dimensional systems that can reconstruct the dynamics in the physical space with less computational cost. It has a long history in the community of fluid dynamics. These are often termed as reduced-order models and are closely related to dimension reduction techniques like principal orthogonal decomposition, principal-component analysis, dynamic-model decomposition. The introduction of autoencoders (Baldi & Hornik, 1989; Goodfellow et al., 2016) generalized these powerful techniques from learning linear subspace to learning coordinates on a curved manifold and have shown to improve greatly the performance of classical ROM models (Vinuesa & Brunton (2021) for a detailed recent review).

Recently, data-driven methods have been introduced to study chemistry in astrophysical environments. Grassi et al. (2021) applied the autoencoder with a combination of a latent fiducial chemical network to simplify a 29-species chemical network with 224 reactions into a reduced network with 5 species and 12 reactions. Their work has demonstrated a significant (65-times) speed-up. However, this study is limited to a constant density, temperature and cosmic-ionization rate environment.

In this work, we expand on the model proposed by Grassi et al. (2021) with a modified architecture and apply it to a 9-species primordial chemical network problem. Instead of focusing on a fixed density, and temperature grid, our

[1]Department of Astronomy, University of Illinois Urbana-Champaign, US [2]School of Information Sciences, University of Illinois at Urbana-Champaign, US. Correspondence to: Kwok Sun Tang <kwoksun2@illinois.edu>.

ICML 2022 Workshop on Machine Learning for Astrophysics, Baltimore, Maryland, USA, 2022.

proposed model is trained and evaluated on a snapshot taken from a full cosmological volume, with density ranging from $10^{-28} - 10^{-12}$ g cm^{-3} and temperature ranging from $50 - 2000$K. We address the two questions: 1.How accurate could neural network based reduced system be? 2. How much speed up could we gain with reduced models?

2. Data

To train our proposed model, physically realistic initial conditions are taken from a snapshot of a full cosmological simulations of first stars with the open-source simulation codebase `Enzo` (Bryan et al., 2014). The density and temperature considered spans 14 and 2 orders of magnitudes. The detail of the simulations and the phase-space distribution of the initial conditions are outlined and shown in Appendix A. The chemical abundances and thermal energy from the simulation grid cell are taken as initial conditions and are evolved for one freefall time with logarithmically-spaced timesteps to capture the wide range of dynamical behavior across different timescale with the meta-solver `Dengo` (Tang & Turk, In Prep) and the ODE integrator `Sundials CVODE` (Hindmarsh et al., 2005). Note that the spatial information of each cell is discarded. In this study, we have limited ourselves to studying the 9-species network which includes $\mathrm{H}_2, \mathrm{H}_2^+, \mathrm{H}, \mathrm{H}^+, \mathrm{H}^-, \mathrm{He}, \mathrm{He}^+, \mathrm{He}^{++}, \mathrm{e}^-$ and thermal energy. The total state space vector $\mathbf{x} \in \mathbb{R}^N$ lives in 10 dimension space. We refer interested readers to the `Grackle` method paper (Smith et al., 2017) for a more in-depth discussion of the 9-species network.

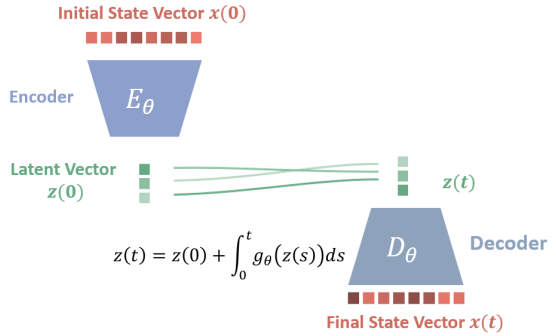

Figure 1. A Schematic of our proposed model to evolve the system for one timestep. The encoder E_θ takes the initial condition $x_0 \in \mathbb{R}^n$ and maps it into the latent space $z_0 \in \mathbb{R}^m$. The neural ODE adopted is an autonomous system that does not explicitly dependent on independent variables i.e t. z_t can be obtained by integrating the neural ODE from $t = 0$ to t. The Decoder D_θ decode the latent vector z_t alongside the initial latent vector and the back to the abundance space $x_{\mathrm{t,pred}}$.

3. Model

Our proposed model consist of three separate neural networks, the Encoder E_θ, Decoder D_θ, and the neural ODE function f_θ and a schematic of the architecture is shown in Figure 1. The overall model can be specified with the equations below.

$$
\begin{aligned}
\mathbf{z}_0 &= E_\theta(\tilde{\mathbf{x}}_\mathbf{0}), \quad \mathbf{x} \in \mathbb{R}^n \\
\frac{d\mathbf{z}}{dt} &= g_\theta(t, \mathbf{z}), \quad \mathbf{z} \in \mathbb{R}^m \\
\mathbf{z}_t &= \mathbf{z}_0 + \int_0^t g_\theta(\mathbf{z}_{t'})dt' \\
\tilde{\mathbf{x}}_{\mathrm{t,pred}} &= \tilde{D}_\theta(\mathbf{z}_t) \\
\mathbf{x}_{\mathrm{t,pred}} &= D_\theta(\mathbf{z}_t, z_0, x_0)
\end{aligned}
\tag{1}
$$

The encoder E_θ is a learnable function that takes the log-normalized state vector as input $\tilde{\mathbf{x}} \in \mathbb{R}^n$, and maps it to the latent space $z \in \mathbb{R}^m$. In our experiments, the dimension of the state vector n and the latent vectors m are 10 and 3 respectively. Hereafter $\tilde{\mathbf{x}}$ corresponds to log-normalized state vector and $\mathbf{x}$ refers to the unnormalized state vector. The RHS function g_θ defines the dynamics of the ODE in the latent space. The encoder $E_\theta : \mathbb{R}^{\mathrm{n}} \to \mathbb{R}^m$ and the ODE function $g_\theta : \mathbb{R}^{\mathrm{m}} \to \mathbb{R}^m$ are parametrized with a multi-layer perceptron (MLP) with 4 layers and 32 hidden units with the `ELU` activation function. In order to limit the potentially arbitrarily large $\frac{dz}{dt}$ learnt by the neural ODE network, we have further applied an additional `tanh` activation with a learnable scale factor $\mathbf{s}_z \in \mathbb{R}^m$ to stabilize the flow field learned by the neural network, i.e. $\frac{d\mathbf{z}}{dt} = \mathbf{s}_z \circ \tanh(\mathrm{MLP}_\theta(\mathbf{z})/\ \mathbf{s}_z)$. By restricting the limit of the $\frac{dz}{dt}$ to between $[-\mathbf{s}_z, \mathbf{s}_z]$, it is empirically effective at stabilizing training in our experiments.

As for the decoder D_θ, we have experimented with two different architectures. One of them is a MLP $\tilde{D}_\theta : \mathbb{R}^{\mathrm{m}} \to \mathbb{R}^n$ that takes the latent vector z as input and output the abundance in log-normalized space $\tilde{\mathbf{x}} \in \mathbb{R}^n$. We termed it the "vanilla decoder". It is also parameterized with a 4-layer MLP with 32 hidden units with `ELU` activation function. This setup is similar to the one proposed in Grassi et al. (2021), except that in our work the latent network dynamics is replaced with a more versatile MLP g_θ. We have also device a new type of decoder that incorporates the initial condition of the ODE $D_\theta : \mathbb{R}^{2\mathrm{m}+\mathrm{n}} \to \mathbb{R}^n$, $\mathbf{x}_{t,\mathrm{pred}} = \mathbf{x}_0 \circ \exp(\mathbf{s} \circ \mathrm{MLP}_\theta(\mathrm{z_t}, \mathrm{z_0}, \tilde{\mathrm{x}}_0))$. Here $\circ$ corresponds to elementwise multiplication.

Instead of directly learning the mapping from the latent space $\mathbf{z}$ to the abundance space $\mathbf{x}$, we have the decoder predict the log-variation of the abundance with respect to its initial abundance. This kind of parametrization also guarantees the output from the decoder is always positive definite. $\mathbf{s} \in {\mathbb{R}_+}^n$ here corresponds to a learnable scale

factor that accounts for the scale over which the abundance varied across the timescale of interest. For comparison, we have also used a vanilla encoder-decoder combination with a single linear layer that mimics a latent space generated from linear projection.

One of the measure of the dynamical timescale is the free-fall timescale $t_{\rm ff}$ and it can be estimated directly by $t_{\rm ff} \approx \frac{1}{\sqrt{G\rho}}$, where G is the gravitational constant, and ρ is the density of the cell. In order to accommodate a wide range of dynamical timescale of interest, time axis for each of the trajectory t are normalized by the respective freefall timescale given by the density implied from the initial condition $\mathbf{x}_0$.

The loss function $\mathcal{L}$ for each given trajectory is specified in terms of the true solution $\mathbf{x}_t$ and the predicted solution $\mathbf{x}_{\rm t,pred}$ as below:

$$\mathcal{L}_{\rm path}(\mathbf{x}, \mathbf{x}_{\rm pred}) = \sum_t |\log(\mathbf{x}_{\rm t}) - \log(\mathbf{x}_{\rm t,pred})|,$$

$$\mathcal{L}_{\rm recon}(\mathbf{x}) = \sum_t |\log(\mathbf{x}_t) - \log D_\theta(E_\theta(\mathbf{x}_t), \mathbf{z}_0, \mathbf{x}_0)|,$$

$$\mathcal{L}_{\rm conserv}(\mathbf{x}) = \mathcal{L}_{\rm conserv,H}(\mathbf{x}) + \mathcal{L}_{\rm conserv,He}(\mathbf{x}) + \mathcal{L}_{\rm conserv,e}(\mathbf{x})$$

$$\mathcal{L} = \mathcal{L}_{\rm path} + \lambda_1 \mathcal{L}_{\rm recon}(x) + \lambda_2 \mathcal{L}_{\rm conserv}(x_{\rm pred})$$

$\mathcal{L}_{\rm path}$ corresponds to the absolute log-difference of the predicted values and the actual trajectory, and $\mathcal{L}_{\rm recon}$ enforces that the initial condition is consistently encoded in the latent space. This is often termed the reconstruction loss in the context of an autoencoder, where the objective of an autoencoder is to reconstruct the state vector x. The total mass density in hydrogen H and helium He should be kept the same as the time progress. The net charge of each trajectory should also stay zero in the reconstructed solution. $\mathcal{L}_{\rm conserv}$ encourages the system to penalize solutions that do not obey the law of conservation. λ_1, λ_2 are both set to unity in the rest of our experiments. The loss function defined above is specified for our proposed initial-condition guided autoencoder. This is defined similarly for the log-normalized outputs from the plain decoder.

These three neural networks are optimized jointly to minimize the loss function $\mathcal{L}$. We have made used of the `torchdiffeq` package to perform ODE integration, and backpropagation through the ODE solution using the memory-efficient adjoint method (Chen et al., 2018). We refer our readers to Appendix B for the details on the implementation and training. The code are made available on Github[1].

[1]https://github.com/hisunnytang/neuralODE

10^{-3}	1-layer	Vanilla	Our Model
$\mathcal{L}_{\rm path}$	296	74.3	**6.10**
$\mathcal{L}_{\rm recon}$	288	84.2	**5.35**
$\mathcal{L}_{\rm conserv}$	2.04	11.8	**0.10**
Mean Pct. Err. (%)	36.0	9.00	**0.97**

Table 1. Error Metrics for three of the models experimented. Each of the loss term is presented in 10^{-3}. The last row corresponds to the mean percentage error evaluated in the physical space.

4. Results

4.1. Trajectory Prediction and Accuracy

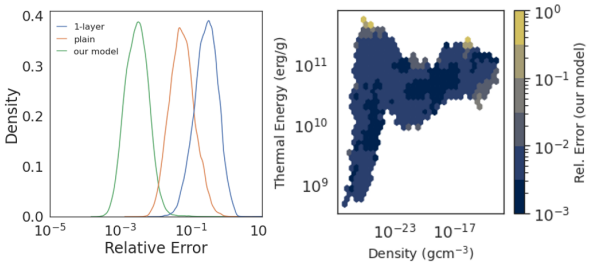

Figure 2. Left-Panel: Relative Error Distribution calculated in the physical space (not log-normalized space) for the three different models experimented. Right-Panel: Relative error distribution in the density-thermal energy phase space our initial value guided autoencoder. It on average shows an relative error of less than 1% across the phase space, except for regions with high temperatures.

The summary statistics and the relative error distribution of the three models on the test-set data is tabulated and reported in Table 1 and in Figure 2. The absolute log error decreases with increasing model complexity. The one-layer model which mimics a simple linear projection performs the worst out of these models. This indicates a simple linear projection mapping is incapable of finding a good latent space for proper reconstruction. This also motivates and justifies the use of a more complicated autoencoder that is able to capture the non-linear interactions between the chemical species. Our plain-autoencoder architecture, which shares the most resemblance with the one proposed by Grassi et al. (2021), shows on average a 10% relative error. Our proposed initial-value guided decoder architecture achieves the best performance out of all three models considered with a relative error of 1%. On the right panel of Figure 2, we showed the distribution of relative error across the density-thermal-energy phase space. It shows on average an relative error of less than 1% across the phase space, except for regions with high temperatures and high density.

As a demonstration, various initial conditions are drawn from the test set, and are evolved with our trained models. The true trajectories are shown alongside with the predicted trajectories in Figure 3. Despite the wide range of order of magnitude involved across not only density and temperature,

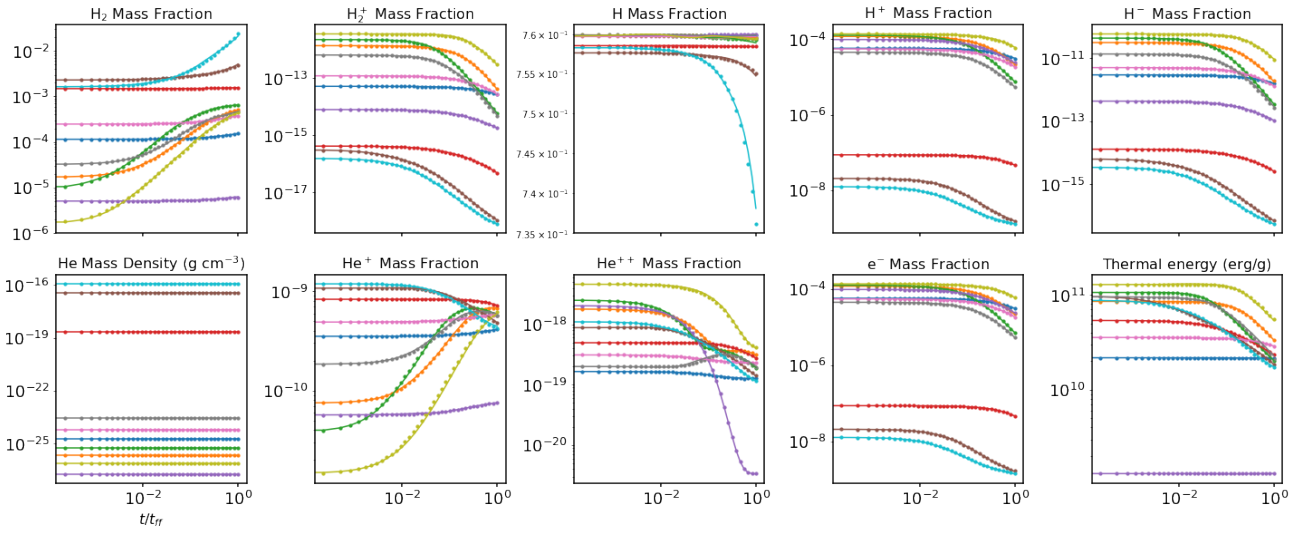

Figure 3. Sample Trajectories from various initial conditions with our proposed initial value guided autoencoder. The time axis are all normalized to the freefall timescale. The dots represent the ground-truth solution generated from `Dengo` solver with a relative tolerance of 10^{-6}. The lines show the respective trajectories from our latent ODE integration. Different initial conditions are highlighted in different colors. Except for `He` and thermal energy, the rest of the species of interest are normalized by the total density and expressed in mass fractions. Our model is capable of handling initial conditions that spans 10 orders of magnitude in density (can be inferred from the He Mass Density Panel), more than 2 orders of magnitude in thermal energy across various freefall timescales defined by the density. Sample trajectories from our 1-layer autoencoder, and vanilla autoencoder is also shown in Appendix C.

but also the species abundances, our model is still capable of recovering the morphology of the correct trajectories up to one freefall timescale. We refer our readers to Appendix C for the trajectories from the rest of the two models.

4.2. Performance Comparison

In this section, we look into the performance of our proposed model and `Grackle`. The learnt neural ODE, Encoder, Decoder are exported to `TorchScript` from the `PyTorch` (Paszke et al., 2019) model checkpoint. The resulting `TorchScript` model can be run independently from `Python` as a standalone `C++` program in a production environment. Similar to the above described procedures, the abundances are transformed into the latent space and from latent space to abundances spaces with the Encoder and Decoder model respectively. The numerical integration is performed with the `CVode` (Hindmarsh et al., 2005) with the neural ODE function learnt in the latent space z. Since the neural ODE model is fully differentiable, the respective Jacobian $\frac{\partial g_\theta}{\partial z}$ can be constructed with almost no cost with the the `autograd` function available to the `PyTorch` model. The table below shows a comparison between our various proposed models.

The experiments are performed with the test set data with on `Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz` with 40-threads `OpenMP` acceleration. The `TorchScript` model are deployed in parallel with batch size of 2048. The results are shown and tabulated in Table 2. Our proposed model with modest parallelization has achieved an almost tenfold speed up compared to the widely adopted primordial chemistry solver `Grackle` on this particular 9-species model.

Per-cell runtime (10^{-6}s)	0.1 $t_{\rm ff}$	$t_{\rm ff}$
`Grackle`	39.2 ± 2.8	41.6 ± 3.1
`TorchScipt+CVODE`	$\mathbf{3.83 \pm 0.05}$	$\mathbf{6.32 \pm 0.10}$

Table 2. Runtime Comparison in production environment among the `Grackle` and Our Model.

5. Discussion and Future Work

We introduce our new architecture for reducing chemical network in a data-driven way and show that our proposed architecture is capable of recovering the temporal trajectories faithfully across a wide-range of initial conditions in a deployment setting with one-tenth of the runtime compared to a commonly used astro-chemistry library. We note that when comparing the performance between our solvers and `Grackle`, our models are deployed on CPU only. By extending the current `Torchscript` model and its interface with `CVODE` to GPU (Balos et al., 2021), we should expect to see a further speedup in terms of the runtime. The model can be further improved by adding an additional terms in the loss function that penalize the stiffness of the latent

space dynamics equation. Such regularization can implicitly enforce the learnt vector field g_θ in the latent space to be smoother and render it easier to integrate.

References

Baldi, P. and Hornik, K. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53–58, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90014-2. URL https://www.sciencedirect.com/science/article/pii/0893608089900142.

Balos, C. J., Gardner, D. J., Woodward, C. S., and Reynolds, D. R. Enabling GPU accelerated computing in the SUNDIALS time integration library. *Parallel Computing*, 108: 102836, 2021.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P. Machine learning for fluid mechanics. *Annual Review of Fluid Mechanics*, 52(1):477–508, 2020. doi: 10.1146/annurev-fluid-010719-060214. URL https://doi.org/10.1146/annurev-fluid-010719-060214.

Bryan, G. L., Norman, M. L., O'Shea, B. W., Abel, T., Wise, J. H., Turk, M. J., Reynolds, D. R., Collins, D. C., Wang, P., Skillman, S. W., Smith, B., Harkness, R. P., Bordner, J., hoon Kim, J., Kuhlen, M., Xu, H., Goldbaum, N., Hummels, C., Kritsuk, A. G., Tasker, E., Skory, S., Simpson, C. M., Hahn, O., Oishi, J. S., So, G. C., Zhao, F., Cen, R., and and, Y. L. ENZO: AN ADAPTIVE MESH REFINEMENT CODE FOR ASTROPHYSICS. *The Astrophysical Journal Supplement Series*, 211(2):19, mar 2014. doi: 10.1088/0067-0049/211/2/19. URL https://doi.org/10.1088%2F0067-0049%2F211%2F2%2F19.

Champion, K., Lusch, B., Kutz, J. N., and Brunton, S. L. Data-driven discovery of coordinates and governing equations, 2019. URL https://arxiv.org/abs/1904.02107.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural ordinary differential equations. *Advances in Neural Information Processing Systems*, 2018.

Gong, M., Ostriker, E. C., and Wolfire, M. G. A Simple and Accurate Network for Hydrogen and Carbon Chemistry in the Interstellar Medium. *ApJ*, 843(1):38, July 2017. doi: 10.3847/1538-4357/aa7561.

Goodfellow, I. J., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. http://www.deeplearningbook.org.

Grassi, T., Nauman, F., Ramsey, J. P., Bovino, S., Picogna, G., and Ercolano, B. Reducing the complexity of chemical networks via interpretable autoencoders, 2021. URL https://arxiv.org/abs/2104.09516.

Hahn, O. and Abel, T. Multi-scale initial conditions for cosmological simulations. *MNRAS*, 415(3):2101–2121, August 2011. doi: 10.1111/j.1365-2966.2011.18820.x.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., and Woodward, C. S. SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

Kamp, I., Thi, W. F., Woitke, P., Rab, C., Bouma, S., and Ménard, F. Consistent dust and gas models for protoplanetary disks. II. Chemical networks and rates. *AAP*, 607:A41, November 2017. doi: 10.1051/0004-6361/201730388.

Omukai, K. Protostellar Collapse with Various Metallicities. *ApJ*, 534(2):809–824, May 2000. doi: 10.1086/308776.

Omukai, K., Tsuribe, T., Schneider, R., and Ferrara, A. Thermal and Fragmentation Properties of Star-forming Clouds in Low-Metallicity Environments. *ApJ*, 626(2): 627–643, June 2005. doi: 10.1086/429955.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance.pdf.

Richings, A. J., Schaye, J., and Oppenheimer, B. D. Non-equilibrium chemistry and cooling in the diffuse interstellar medium - I. Optically thin regime. *MNRAS*, 440(4): 3349–3369, June 2014a. doi: 10.1093/mnras/stu525.

Richings, A. J., Schaye, J., and Oppenheimer, B. D. Non-equilibrium chemistry and cooling in the diffuse interstellar medium - II. Shielded gas. *MNRAS*, 442(3):2780–2796, August 2014b. doi: 10.1093/mnras/stu1046.

Smith, B. D., Bryan, G. L., Glover, S. C. O., Goldbaum, N. J., Turk, M. J., Regan, J., Wise, J. H., Schive, H.-Y., Abel, T., Emerick, A., O'Shea, B. W., Anninos, P., Hummels, C. B., and Khochfar, S. GRACKLE: a chemistry and cooling library for astrophysics. *MNRAS*, 466: 2217–2234, April 2017. doi: 10.1093/mnras/stw3291.

Tang, K. S. and Turk, M. J. Dengo: An engineer for chemistry solvers, In Prep.

Turk, M. J., Smith, B. D., Oishi, J. S., Skory, S., Skillman, S. W., Abel, T., and Norman, M. L. yt: A Multi-code Analysis Toolkit for Astrophysical Simulation Data. *ApjS*, 192(1):9, January 2011. doi: 10.1088/0067-0049/192/1/9.

Vinuesa, R. and Brunton, S. L. The potential of machine learning to enhance computational fluid dynamics, 2021. URL https://arxiv.org/abs/2110.02085.

Vlachas, P. R., Arampatzis, G., Uhler, C., and Koumoutsakos, P. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, Apr 2022. ISSN 2522-5839. doi: 10.1038/s42256-022-00464-w. URL https://doi.org/10.1038/s42256-022-00464-w.

A. Cosmological Simulation and Data Sampling

The simulation is initialized at $z = 50$ in a $0.3\ \mathrm{Mpc}/h$ co-moving box with `MUSIC` (Hahn & Abel, 2011) initial conditions generator. We have first run a dark matter only simulation to identify the isolated halos of interest with mass ranges between $10^5 - 10^6 M_\odot$. These initial conditions are re-centered and generated again based on the location of the most massive halo identified above. Cells are flagged for refinement if the local Jeans length is not resolved by 64 cells. Evolution timescale is also limited to a tenth of the thermal timescale. A snapshot at $z = 18$ is taken as our initial state vector for the generation of the trajectory dataset.

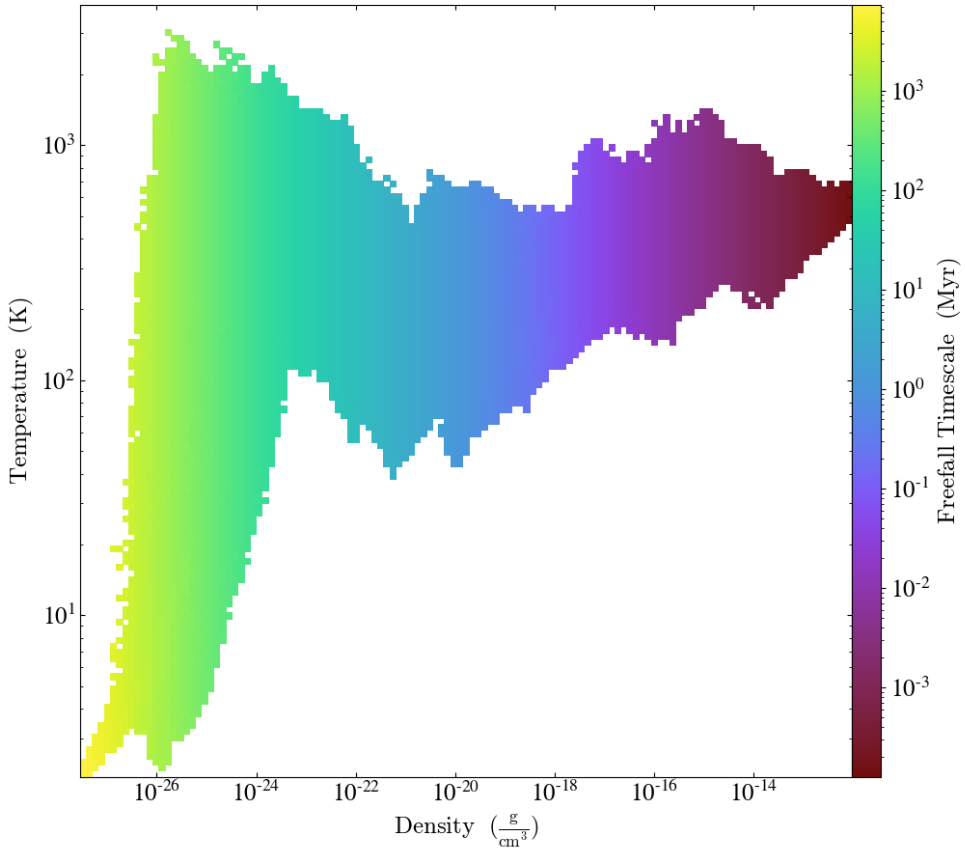

Figure 4. The phase distribution of the density and temperature of the initial conditions taken from the simulation snapshot. The colorbar demonstrates the characteristic freefall timescale which spans from $10^{-3} - 10^3$Myr.

For any user-specified chemistry network, `Dengo` (Tang & Turk, In Prep) can generate the corresponding RHS function (f), the Jacobian function ($\frac{\partial f}{\partial y}$) and the interface to solve the system of ODE with integrator `Sundials CVODE` (Hindmarsh et al., 2005). The availability of the Jacobian allows for a noticeable speed up in the integration process particular for stiff ODE system. The relative tolerance of `Dengo` is set to 10^{-6} when the trajectory is generated. The output timesteps are logarithimcally spaced. There are a total of 3068798 number of grid cells. As a pilot study, we limit our dataset to a subset that is representative of the physical conditions in the simulations. The dataset is sub-sampled on a $64 \times 64 \times 64$ grid of density, temperature and H_2 mass fraction, where in each unique bin, 5 samples are select. This leaves us with 114401 cells. It is split into train-validation-test set with a ratio of $0.8 : 0.1 : 0.1$ The data analysis and data extraction are performed with `yt` (Turk et al., 2011).

B. Model and Training Details

The models outlined in Section 3 are all implemented in `PyTorch` (Paszke et al., 2019). The models are trained with `Adam` optimizer with an initial learning rate of 10^{-3}`ReduceLROnPlateau` learning rate scheduler with default parameters and

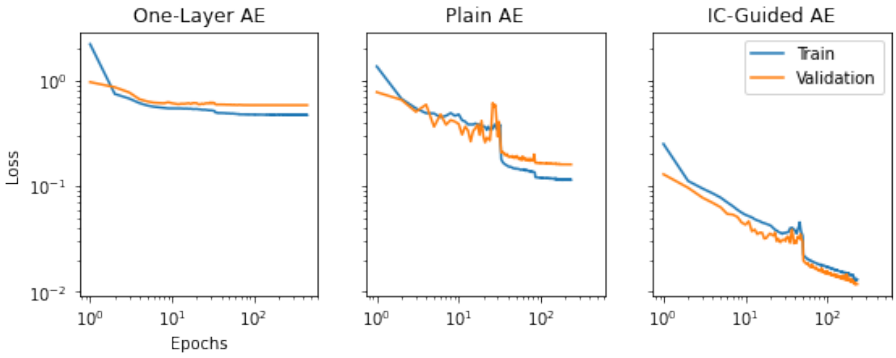

Figure 5. Training and validation loss for the three models experimented in this work.

a minimum learning rate of 10^{-6}. The training is stopped early with a patience of 5 epochs. The training and validation curve for the various models are shown in Figure 5.

C. Trajectories of 1-layer, and Plain Autoencoder architecture

Similar to Fig. 3, the trajectories for various initial conditions are shown for the 1-layer model, and the plain autoencoder model. Our proposed model shows a

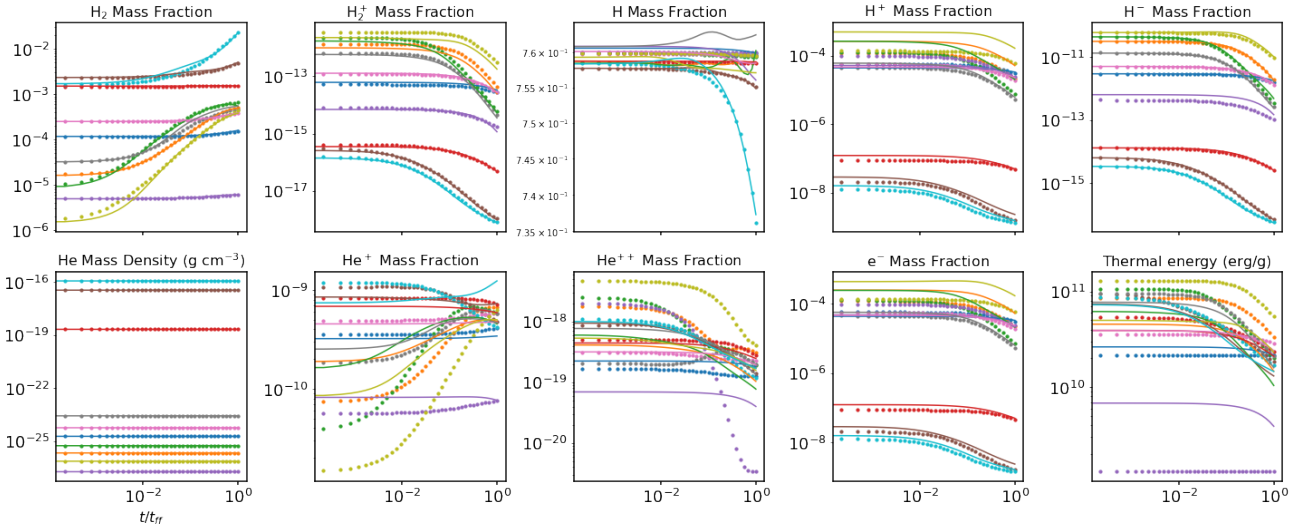

Figure 6. Trajectory Predictions from our 1-layer autoencoder model

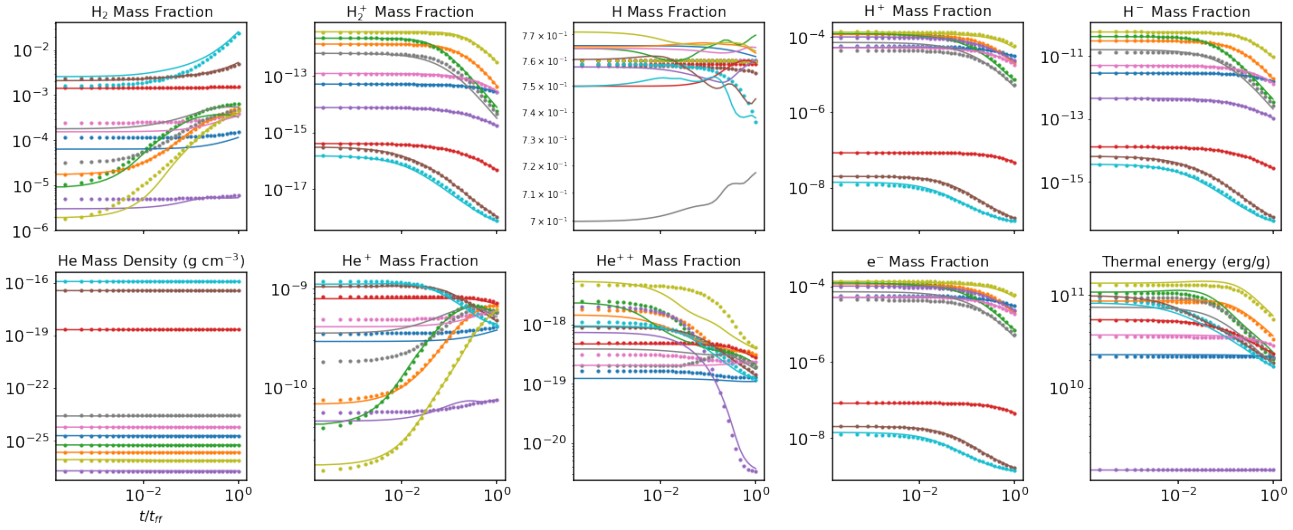

Figure 7. Trajectory Predictions from our Plain autoencoder model

D. Error Distribution of the 1-layer, Plain Autoencoder and our model

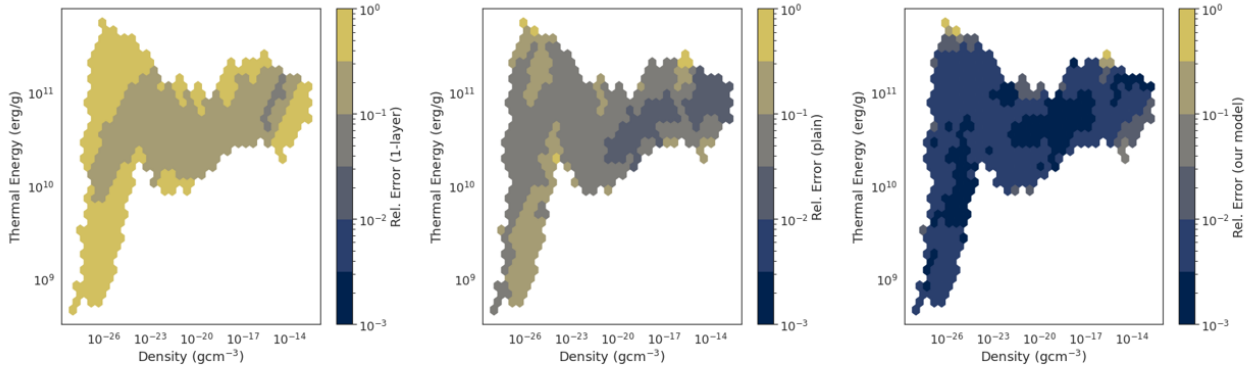

Figure 8. The error distribution of the three models presented in density-temperature phase space