NeuralPDR: neural differential equations as surrogate models for photodissociation regions

Vermariën, G.; Bisbas, T.G.; Viti, S.; Zhao, Y.; Tang, X.; Ravichandran, R.

NeuralPDR: Neural Differential Equations as surrogate models for Photodissociation Regions

Gijs Vermariën[1,2], Thomas G. Bisbas[3], Serena Viti[1,4,5], Yue Zhao[2], Xuefei Tang[3], Rahul Ravichandran[1]

[1]Leiden Observatory, Leiden University, P.O. Box 9513, 2300 RA Leiden, The Netherlands

[2]SURF, Amsterdam, The Netherlands

[3]Research Center for Astronomical Computing, Zhejiang Lab, Hangzhou 311100, China

[4]Transdisciplinary Research Area (TRA) 'Matter'/Argelander-Institut für Astronomie, University of Bonn, Bonn, Germany

[5]Department of Physics and Astronomy, University College London, Gower Street, London, UK

E-mail: `vermarien@strw.leidenuniv.nl`

Abstract. Computational astrochemical models are essential for helping us interpret and understand the observations of different astrophysical environments. In the age of high-resolution telescopes such as JWST and ALMA, the substructure of many objects can be resolved, raising the need for astrochemical modeling at these smaller scales, meaning that the simulations of these objects need to include both the physics and chemistry to accurately model the observations. The computational cost of the simulations coupling both the three-dimensional hydrodynamics and chemistry is enormous, creating an opportunity for surrogate models that can effectively substitute the chemical solver. In this work we present surrogate models that can replace the original chemical code, namely Latent Augmented Neural Ordinary Differential Equations. We train these surrogate architectures on three datasets of increasing physical complexity, with the last dataset derived directly from a three-dimensional simulation of a molecular cloud using a Photodissociation Region (PDR) code, `3D-PDR`. We show that these surrogate models can provide speedup and reproduce the original observable column density maps of the dataset. This enables the rapid inference of the chemistry (on the GPU), allowing for the faster statistical inference of observations or increasing the resolution in hydrodynamical simulations of astrophysical environments.

Keywords: Astrochemistry, Interstellar Medium, Dynamical Systems, Surrogate models, Machine Learning

1. Introduction

Computational models of the interstellar medium help us to understand the physical structure and chemical content that we observe in astronomical regions such as the Orion bar (Peeters et al.; 2024). In order to understand the transition from the low density medium into the high density medium, three-dimensional simulations are performed. In order to match these simulations to the observables, the chemistry of these regions must be simulated as well. It is this coupling with the chemistry that causes a critical slowdown of the simulation. One solution is to develop surrogate models that can rapidly evaluate the chemistry, rebalancing the computational budget.

We model these regions in interstellar space, known as Photodissociation Regions (PDR) (Wolfire et al.; 2022), by simulating their physical structure using hydrodynamical codes. Through a snapshot of such a simulation, we take many lines of sight from all directions, representing the rays along which we could observe this object. We then solve for the chemistry along these rays, with the independent variable being the visual extinction A_V. Visual extinction $A_V$is a measure of the decrease in radiation as we move into an astronomical object, and is related to the amount of hydrogen nuclei along a line of sight (Güver and Özel; 2009). Solving the chemistry as a function of the visual extinction is computationally expensive, since it needs to iteratively solve for both the coupled temperature and chemistry, accounting for the processes of cooling, heating, creation, and destruction of the species. A comprehensive review and benchmarking of different codes is provided in (Rollig et al.; 2007). In this work, we use the `3D-PDR` code (Bisbas et al.; 2012) to post-process three physical structures: a homogeneous cloud in one dimension, an inhomogeneous cloud in one dimension, and finally an actual three-dimensional simulation of the interstellar medium. We then train surrogate models that are drop-in replacements for the original expensive chemical code.

Surrogate modeling has become a widespread tool for solving and helping interpret astrochemical problems. The goal of a surrogate model is to replace the original code, increasing the inference speed, at the cost of some accuracy or specialization to a predetermined parameter space. These surrogate models can be partitioned into two categories, one in which only one steady-state solution or solution at a time of the chemistry is achieved, and the other in which a full depth, time, or space-dependent solution is required. Good examples of the first are neural networks for the direct emulation of emission spectra (de Mijolla et al.; 2019; Grassi et al.; 2025) and regression forests for chemical abundances in order to help with explainability (Heyl et al.; 2023). The second category has been studied more widely in the past years, with first attempts applying autoencoders directly to abundances (Holdship et al.; 2021), Physics Informed Neural Networks (Branca and Pallottini; 2022), Latent (Neural) Differential Equations (Grassi et al.; 2021; Tang and Turk; 2022; Sulzer and Buck; 2023; Maes et al.; 2024), operator learning (Branca and Pallottini; 2024) and neural fields (Ramos et al.; 2024). Efforts to gather different datasets and compare architectures are also being made (Janssen et al.; 2024). The main goal of these surrogate models is to replace the plethora

Table 1. Properties of the datasets used for training the surrogate models with the dynamic ranges of the auxiliary parameters listed in brackets.

	Samples	Length	Species	$n_{\mathrm{H,nuclei}}(\mathrm{cm}^{-3})$	$T(K)$	F_{UV}(Habing)	ζ (s^{-1})
v1	8192	302	19	$[10, 9.4 \times 10^3]$	$[10^2, 10^7]$	$[10, 10^5]$	$[10^{-17}, 10^{-15}]$
v2	1024	490	28	$[10, 1.7 \times 10^4]$	$[0.1, 10^6]$	$[0,10^3]$	$[10^{-17}, 10^{-14}]$
v3	301945	up to 592	31	$[10,2.6 \times 10^5]$	[10,260]	[0, 6.6]	10^{-17}

of computationally expensive astrochemical codes. The speedup of these surrogates enables the faster inference of observational results and the simulations of astronomical objects. With enough speedup, it could enable the direct inference of observations using coupled three-dimensional hydrodynamical and astrochemical codes, something which is currently prohibitively expensive. These coupled simulations are so expensive that they can currently only be run on university clusters and supercomputers (Seifried et al.; 2017; Grudić et al.; 2021; Gong et al.; 2023; Yue et al.; 2024).

In this article, we discuss a total of three datasets of increasing physical complexity, all computed using the `3D-PDR` code. The first two datasets consist of two simple spherical models, whereas the third dataset is derived from a three-dimensional simulation of a molecular cloud. We then introduce latent Neural Ordinary Differential Equations (NODEs) as a surrogate model that can be trained to emulate these datasets. This is followed by a description of the architecture, parameters, and strategies we use to effectively train these surrogate models. We then briefly discuss the results of the surrogate models trained on the first two datasets. Next, we present more extensively the results of the training on the last dataset, showing that the surrogate model can accurately reproduce the original observable column densities. Finally, we conclude the paper with a discussion and an outlook of what is needed to advance the application of these surrogate models.

2. Methods

2.1. Models of Photodissociation Regions

Models of photodissociation regions are essential to model the transition of chemistry as we go from the low-density interstellar medium into higher density filaments and eventually into dense star-forming regions. The density, which is defined as the hydrogen nuclei number density per cubic centimeter: $n_{\mathrm{H},nuclei} = n_{\mathrm{H}} + 2n_{\mathrm{H}_2}$ with n_{H} and n_{H_2} the hydrogen and molecular hydrogen number densities in cm^{-3} respectively, is the dominant physical parameter that dictates how the temperature, radiation, and subsequently the chemistry behave. The visual extinction and density are related via the integral $A_V \propto \int n_{\mathrm{H},nuclei} \mathrm{d}s$ along the line of sight s. At low visual extinction $A_{\mathrm{V}} < 1$, the medium is radiation-dominated and the densities are low, allowing ionized and atomic species to dominate. As the visual extinction increases to $A_{\mathrm{V}} > 1$, however, radiation is attenuated and cooling becomes more effective, allowing the gas to cool down and

species to tend towards their molecular forms. At the highest densities, molecules such as carbon monoxide (CO) start to form effectively. The underlying physical processes are described by a system of differential equations with one ODE per species, and an ODE for the temperature:

$$\frac{\mathrm{d}n_i}{\mathrm{d}t} = \sum_{j,l} k_{jl} n_j n_l + \sum_j k_j n_j - n_i (\sum_{i,l} k_{il} n_l + \sum_j k_j), \tag{1}$$

$$\frac{\mathrm{d}T}{\mathrm{d}t} = \frac{1}{k_b n_{\mathrm{H},nuclei}} \left(\sum_m \Gamma_m - \sum_m \Lambda_m \right), \tag{2}$$

with i, j and l the species indices and m the cooling and heating process indices (Bovino and Grassi; 2023) and k_b the Boltzmann constant in erg $\cdot$ K^{-1}. The first system of differential equations describes the unimolecular and bimolecular reactions with the positive signs accounting for creation of the species and negative sign accounting for the destruction. The second equation describes the evolution of the energy in erg$\cdot$cm$^{-3}\cdot s^{-1}$‡. The first term includes the heating processes and the second the cooling processes. The coupling of this nonlinear system of equations is strong, since the reaction rate equations depend on the temperature, $k_{ij}(T)$ and the change in temperature depends on chemistry, density, and temperature$\{\Gamma_m, \Lambda_m\}(n_i, n_{\mathrm{H},nuclei}, T)$. In order to solve this system of differential equations along a line of sight in `3D-PDR`, a guess is made of an initial temperature, after which it tries to chemically and energetically converge to a steady-state solution. When the temperature or chemistry changes, this process must be repeated, resulting in costly evaluations. A more detailed description of the process can be found in Appendix A of (Bisbas et al.; 2012).

2.1.1. Uniform density one-dimensional models (v1) As a first benchmark of the surrogate model, we choose a spherically symmetric cloud of uniform density. This 1-dimensional model allows us to approximate the depth-dependent chemistry of a line of sight into the cloud. The initial conditions are chosen to reflect the Orion Cloud. We first vary the initial density $n_{\mathrm{H,nuclei}}$, which plays an important role in determining the rates at which reactions take place, how much heating and cooling can take place, and how much radiation can enter the cloud. Secondly, the initial radiation field F_{UV} is varied, determining the amount of energy available in the outer parts of the cloud and how deep in the cloud the transition from atomic to molecular species takes place. Lastly, the cosmic-ray ionization rate ζ is varied: this rate is not attenuated along the line of sight and provides a mechanism to destroy molecules even deep within the cloud. By varying these three inputs as input parameters into `3D-PDR`, we can compute the abundances and temperature along a line of sight directly into the cloud. A summary of the chosen parameters and the range of others can be found in Table 1. This dataset was generated in 864 CPU core hours using a Intel® Core™ i9-13900 Processor.

‡ The erg is a convenient energy unit in astronomy and is equal to 10^{-7} joule.

2.1.2. Non uniform density one-dimensional models (v2) The first models assume a spherical geometry with uniform density, which is a good first-order approximation for the chemistry. However, it does not account for the fact that, in the interstellar medium, objects are extended and have a density profile that rapidly increases towards the center. We subsequently use the PDFChem dataset (Bisbas et al.; 2023), which was created with the goal to use probability density functions to rapidly infer the average densities of molecules. This provides convenient training data to test models of varying density. The dataset varies its initial radiation field $F_{\rm UV}$ as well as the cosmic ray ionisation rate ζ, but it does not vary the initial density value $n_{\rm H,nuclei}$, which now changes as a function of depth instead.

2.1.3. Three-dimensional simulations of the Interstellar medium (v3) For the final dataset, we then proceed to a physical structure that much more closely resembles that of actual astrophysical objects. For the `3D-PDR` setup, we use a three-dimensional model representing a typical Milky Way giant molecular cloud presented in Seifried et al. (2020) using a uniform grid consisting of 128^3 cells. From each cell, a hierarchy of 12 HEALPix rays (Górski et al.; 2005) is emanated, along which we compute the column densities of species and the line cooling by adopting a large velocity gradient escape probability formalism. For the PDR model, we assume a constant cosmic-ray ionization rate of $\zeta_{\rm CR} = 10^{-17}\,{\rm s}^{-1}$ and an isotropic radiation field with intensity of $\chi/\chi_0 = 1$ (normalized to the spectral shape of Draine; 1978). Once 3D-PDR is converged, we output the gas temperatures and the abundances of species along the HEALPix hierarchy of 12-rays for all cells, under the assumption that each HEALPix ray is considered to be an independent one-dimensional PDR model. We thus generate a significantly large database of one-dimensional models (with a total number of $128^3 \times 12$ rays). Although they share the same PDR environmental parameters of $\zeta_{\rm CR}$ and χ/χ_0, they differ in terms of density distribution along each HEALPix line-of-sight. This dataset takes a total of 1792 CPU core hours (Intel® Xeon® Gold 6348 Processor) to process the chemistry along all rays. We subsequently use a subset of 1/80 total rays, resulting in a dataset with 314573 A_V-series. During training time, we limit ourselves to all series with more than $n > 48$ samples, effectively using only 158948 models.

2.2. Structure of the data and preprocessing

Typically in astrochemistry, the abundances of each molecule are computed in terms of fractional abundances, $x_i = \frac{n_i}{n_{\rm H,nuclei}}$ with n_i (cm^{-3}) the number density. This allows one to investigate the relative abundances of each molecule, regardless of changes in the density of the medium. Inherently, abundances have a large dynamic range. Observable molecules have fractional abundances ranging between $10^{-12} > x_i > 1$, the chemical model thus inherently has a dynamical range of 12 orders of magnitude. In order to also account for molecules below the observational limit, we subsequently choose a lower boundary of $x_i \geq 10^{-20}$ for the training data by introducing a minor offset to each

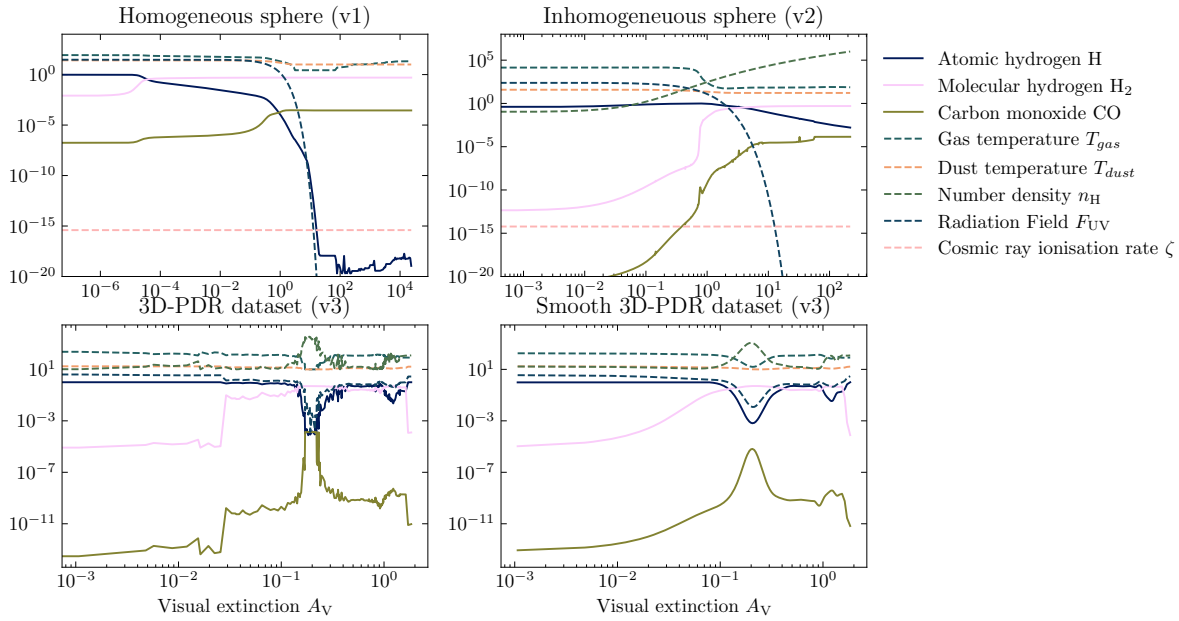

Figure 1. An example of an A_V dependent model for dataset *v1, v2, v3* and *v3* with smoothing.

fractional abundance: $\epsilon_{x_i} = 10^{-20}$. With this large dynamic range, it is more useful to compute our losses in this logarithmic space, so all species are modeled correctly, even when less abundant. To this end, we transform all abundances into log-space.

In log space we then wish to ensure that the distribution of the input features has a distribution close to a standard distribution. To this end, we standardize the data by either the statistics per species (v1 and v2) or the statistics of all species at once (v3). This gives us the following data preprocessing step:

$$D'_i = \frac{\log_{10}(D_i + \epsilon_i) - \tilde{\mu}}{\tilde{\sigma}}, \tag{3}$$

with $\tilde{\mu}$, $\tilde{\sigma}$ being the mean and standard deviation in log-space respectively.

For the auxiliary parameters, we choose the physical parameters that vary for each of the datasets $\vec{p_i} = [A_v, T_{gas}, T_{dust}, n_{H,nuclei}, F_{UV}, (\xi)]$. We choose to include the temperatures as physical parameters, instead of co-evolving them with the abundances in the latent space, as was done in (Vermariën et al.; 2024).

For the *v3* dataset, there are some numerical artifacts where the healpix ray tracing scheme rapidly alternates between two cells with a vastly different chemical composition, resulting in jumps in the chemistry on a non-physical timescale. Due to the recurrent nature of training NODEs in latent space, this nonphysical high-frequency noise introduces large gradients that destabilize training. To combat this, we fit a smooth spline (Zemlyanoy; 2022) in the log abundance space and resample each of the abundances. The smoothing spline for the abundances uses a regularization parameter $\lambda = 10^{-4}$, and a lower and higher boundary of $x_i \in [-30, 0]$ in log space, so that values can never exceed 1 in linear space or become too small. For the physical parameters, we

use the same regularization parameter, but no boundaries. After applying the smoothing spline in log-space, the data is transformed back into linear space. The original and smoothed v3 data can be seen in Figure 1.

2.3. Latent Augmented Neural Ordinary Differential Equations

In order to emulate the chemical series, which are governed by the differential equations defined earlier, we choose Neural Ordinary Differential Equations (NODE)(Chen et al.; 2019; Kidger; 2022) as a data-driven approach, replacing the original $x_{i+1} = \texttt{ODEsolve}(\vec{x_i}, \vec{p_i})$ with a new neural network approximator in the latent space $z_{i+1} = \texttt{NODESolve}(\vec{z}, \vec{p_i})$ with $\vec{z}$ being the latent chemical state vector. We can describe this latent integral over visual extinction as follows:

$$\vec{z}_{i+1} = \Psi(\vec{z}_i, \vec{p}_i, A_V, A_{V+1}) = \vec{z}_i + \int_{A_{\mathrm{v},i}}^{A_{\mathrm{v},i+1}} \psi(\vec{z}_i, \vec{p}_i) dA'_v \tag{4}$$

where $\vec{z}_i \in \mathbb{R}^Z$ is the latent state vector, A_v is the visual extinction, which serves as the independent variable to integrate along the line of sight and $\vec{p}_i \in \mathbb{R}^P$ auxiliary parameters that are concatenated to the input of the nonlinear transformation $\psi : \mathbb{R}^{Z+P} \to \mathbb{R}^Z$. Additionally, we define the shorthand notation without explicit mention of the limits: $\Psi(\vec{z}_i, \vec{p}_i)$. The addition of auxiliary parameters $\vec{p}$, allows us to train a latent model that generalizes over many different physical models with different physical parameters. The practice of enhancing the state vector with extra dimensions and features to find more expressive NeuralODEs has been coined as augmented ODEs (Dupont et al.; 2019) and parameterized ODEs (Lee and Parish; 2021). In this article, we employ the term "auxiliary parameters", since they provide auxiliary information about the physical state of the system to the latent space. This is essential to enable the application of this architecture to the post-processing of simulations, as they provide these physical parameters. But also for directly coupled hydrodynamical simulations in the future, the architecture relies on physical parameters computed by other codes. A diagram showing how the architecture is connected can be found in Figure 2.

These latent neural differential equations require encoder and decoder transformations (Kramer; 1991), allowing one to construct a state for the latent ODE, which can typically be solved at, a lower cost (Grathwohl et al.; 2018; Rubanova et al.; n.d.). This latent ODE can be defined by a small dummy chemical network (Grassi et al.; 2021), constant terms (Sulzer and Buck; 2023) or a tensor expression akin to a larger chemical network (Maes et al.; 2024). Our choice is purely a data-driven NODE with a latent bottleneck size l, enabling us to capture both the chemical and physical state in the latent space. This latent space can then be evolved by solving the learned latent differential equation as a function of visual depth. Specifically, we use a fifth-order Runga-Kutta differential equation solver (Tsitouras; 2011).§

§ The code can be found at `https://github.com/uclchem/neuralpdr`

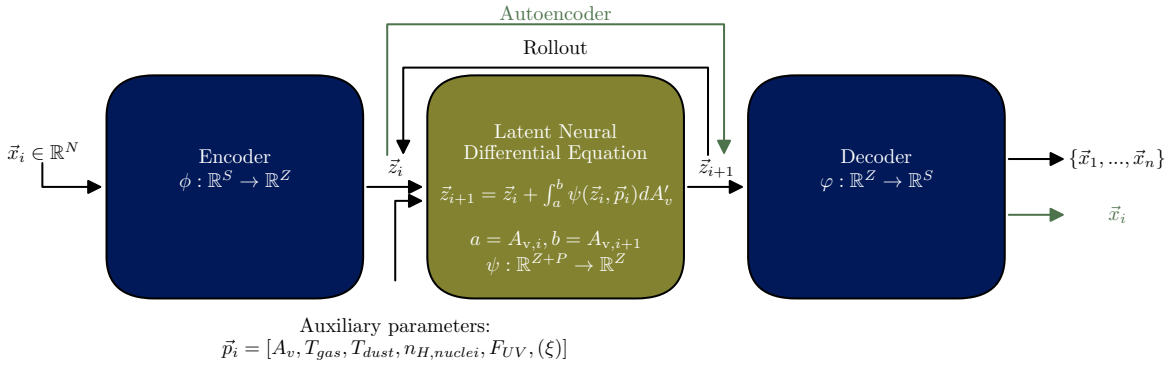

Figure 2. A diagram of the Latent Augmented NeuralODE architecture, the *rollout* pathway produces a series of abundances: $\vec{x}_0, \{\vec{p}_i\} \to \{\vec{x}_1, ..., \vec{x}_n\}$ whilst the *autoencoder* pathway just autoregresses: $\vec{x}_i \to \vec{x}_i$. The blocks contain the neural networks ϕ, ψ, φ with the center block representing the latent differential equation Ψ.

2.4. Batching variable length series

In dataset *v3*, the number of visual extinctions that are sampled along a ray can vary, resulting in a distribution of different series lengths. The distribution of the lengths can be found in Figure 3. We first impose a lower bound of $n \geq 48$ because the shorter series have a high similarity and are less dynamic, resulting in a bias in the training data towards steady state solutions.

We then proceed to use a batching strategy to account for the fact that each series has a different length, with samples of similar lengths having a possibility of being relatively similar. The same problem exists in text-to-speech synthesis, where sorting variable length sentences by length might result in less randomness than desired in each batch (Ge et al.; 2021). On the other hand, if the distribution of lengths is similar to the one we have, batches can be filled with zero-padding to account for the difference in lengths. We adapt the strategy of semirandom batching, adapted for the large power-law distribution of our lengths. We propose to sort the dataset using a small random offset:

$$n' = \log_{10}(n) + \epsilon, \text{ where} \tag{5}$$

$$\epsilon \sim \mathcal{U}(-\alpha, \alpha), \tag{6}$$

with n_i the length of each series, and ϵ a randomly sampled offset factor. We then sort the series by n', create batches by grouping along the sorted axis, and then shuffling the batches. The effect of the offset factor α to the fraction of zero padded elements(ZPF) for batch size 64 and dataset *v3* is shown in Table 2. Based on these values, we select the offset $\alpha = 0.01$, since it only induces a zero padding fraction of 2%.

α	0.0	0.001	0.01	0.025	0.05	0.075	0.1	0.5	-
ZPF	0.000	0.001	0.019	0.052	0.102	0.149	0.192	0.512	0.619

Table 2. The Zero Padded Fraction (ZPF), the number of zero elements needed to pad all batch elements up to the longest length, as a function of the offset factor α for the semi-random sorting. − indicates infinite offset, resulting in random sorting.

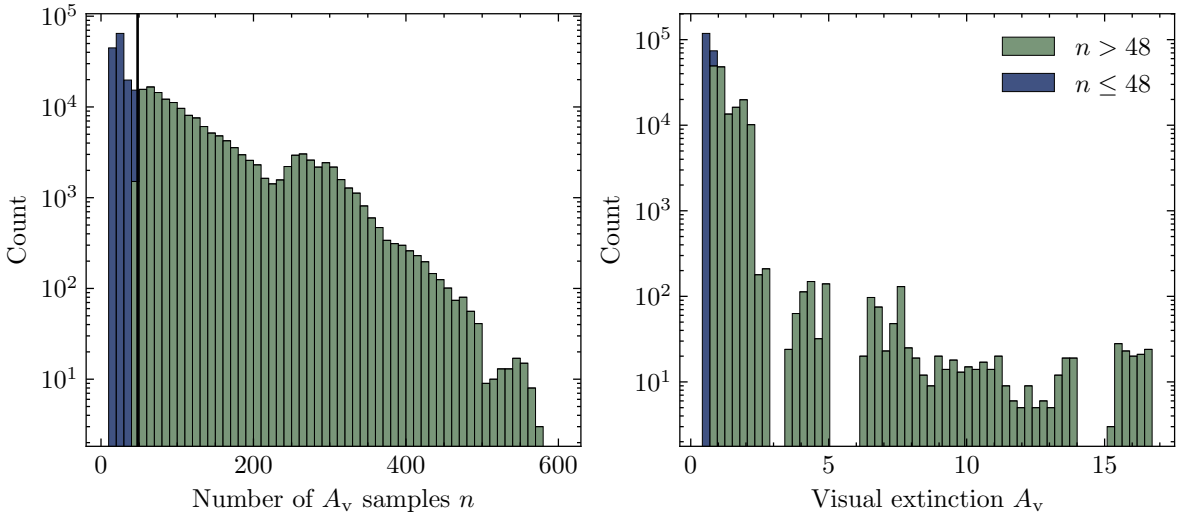

Figure 3. The distribution of the series length n and maximum visual extinction in dataset *v3*. The lower bound $n = 48$ is used during training.

2.5. Training neural differential equations

2.5.1. Loss functions The architecture consists of three main building blocks, the encoder ϕ, the latent NODE block with a vanilla Multi Layer Perceptron (MLP) as the nonlinear function transformation ψ and lastly the decoder φ. This architecture can be trained in two modes typically: directly as an autoencoder $\vec{x}_i \to \vec{x}_i$, or in a recurrent fashion, $\vec{x}_0 \to \{\vec{x}_1, ..., \vec{x}_n\}$ for n rollout steps. For training the architecture we utilize both, starting with a large contribution of the autoencoder loss:

$$\mathcal{L}_{auto} = \sum_{a \in \vec{A}_V} \text{MSLE}(\vec{x}_a, \varphi(\phi(\vec{x}_a))), \tag{7}$$

where MSLE is defined as the Mean Square Logarithmic Error and is defined as $\text{MSLE}(A, B) = \text{MSE}(\log_{10}(A), \log_{10}(B))$ and $\text{MSE}(A, B) = \frac{1}{N}\sum_n (A - B)^2$. The rollout loss is then computed by evolving the state in the latent space, decoding its values back into the physical space and computing the loss

$$\mathcal{L}_{rollout} = \sum_{a \in \vec{A}_V} \text{MSLE}(\vec{x}_a, \varphi(\psi(\phi(\vec{x}_0), \{\vec{p}_0, ..., \vec{p}_a\}, a))). \tag{8}$$

Lastly, we introduce a loss to directly penalize the latent state of the autoencoder and rollout training to stay close enough to each other, directly penalizing their square

distance in the latent space:

$$\mathcal{L}_{latent} = \sum_{a \in \vec{A}_{\mathrm{V}}} \mathrm{MSE}(\phi(\vec{x}_a), \psi(\phi(\vec{x}_0), \{\vec{p}_0, ..., \vec{p}_a\}, a)) \tag{9}$$

All these losses are then combined into $\mathcal{L} = \sum_i \lambda_i \mathcal{L}_i$ for the training process. The computation of these losses is highlighted by the paths shown in Figure 2. These rollout and autoregressive losses on the training and validation set are computed using the standardized log-abundances and the corresponding predictions.

In order to train the latent differential equation solver Ψ and its MLP ψ, one needs to backpropagate through the solver. Several numerical methods exist for this process, namely "discretise-then-optimise", "optimise-then-discretise" and "reversible-ODE-solvers". We use the default 'Diffrax' method of "dicretise-then-optimise", directly propagating through all the operations within the solver, with the added benefit of accuracy and speed at the cost of memory footprint. A more detailed discussion of different methods to obtain gradients from differential equations can be found in chapter five of (Kidger; 2022).

2.5.2. Training strategy The loss weights start out with a large auto weight $\lambda_{auto} = 1$ and a small rollout weight $\lambda_{rollout} = 4 \times 10^{-2}$, but after 15 epochs, this relationship inverses in the span of 15 epochs, as can be seen in Figure 4. The latent loss weight is chosen to have a small values of $\lambda_{latent} = 10^{-3}$. For the validation loss, we only utilize the rollout term, since this is the only relevant metric at inference time.

We combine this multi-objective loss function with a training scheme where we only train with a subset of points in each individual sample by taking a random contiguous subset of the series in the A_{v} axis. We increase the size of the subset after a number of epochs, until we sample the full extent of each series. For the v3 dataset, the subsampling size is shown in the top of Figure 4. For *v3* we use an increasing subset size of 64, 128, 256, 512 and finally all steps, after 0, 5, 10, 20 and 30 epochs respectively. For each of these intervals, the learning rate follows a cosine learning rate schedule with a linear warmup profile(Loshchilov and Hutter; 2016), performing a warm restart for each increase in subset size. For *v1* and *v2*, we follow the same schedule, but with only half the subset size.

Altogether, we train the architecture for a total of 100 epochs. The learning rate optimizer is AdamW with a weight decay factor of 10^{-5} (Loshchilov and Hutter; 2017) and a peak learning rate λ_{learn}. This is combined with the global gradient clipping to improve training stability (Pascanu et al.; 2013). For the training we use a batch size B, a latent bottleneck size l. The encoder ϕ, latent ψ and decoder φ MLPs all consist of H hidden layers of width W, with the ψ having a final activation function tanh, allowing it to map to the range $[-1, 1]$. The used hyperparameters for training on dataset *v1*, *v2* and *v3* can be found in table 3.

Table 3. The hyperparameters for training on the three datasets.

Dataset	B	l	H	$W_{\{\phi,\varphi\}}$	W_ψ	λ_{learn}
v1	32	16	3	128	32	2×10^{-4}
v2	32	16	3	128	32	2×10^{-4}
v3	32	128	3	512	128	2×10^{-4}
v3	64	128	3	512	128	3×10^{-4}
v3	128	{8,16,32,64,128}	3	512	128	5×10^{-4}

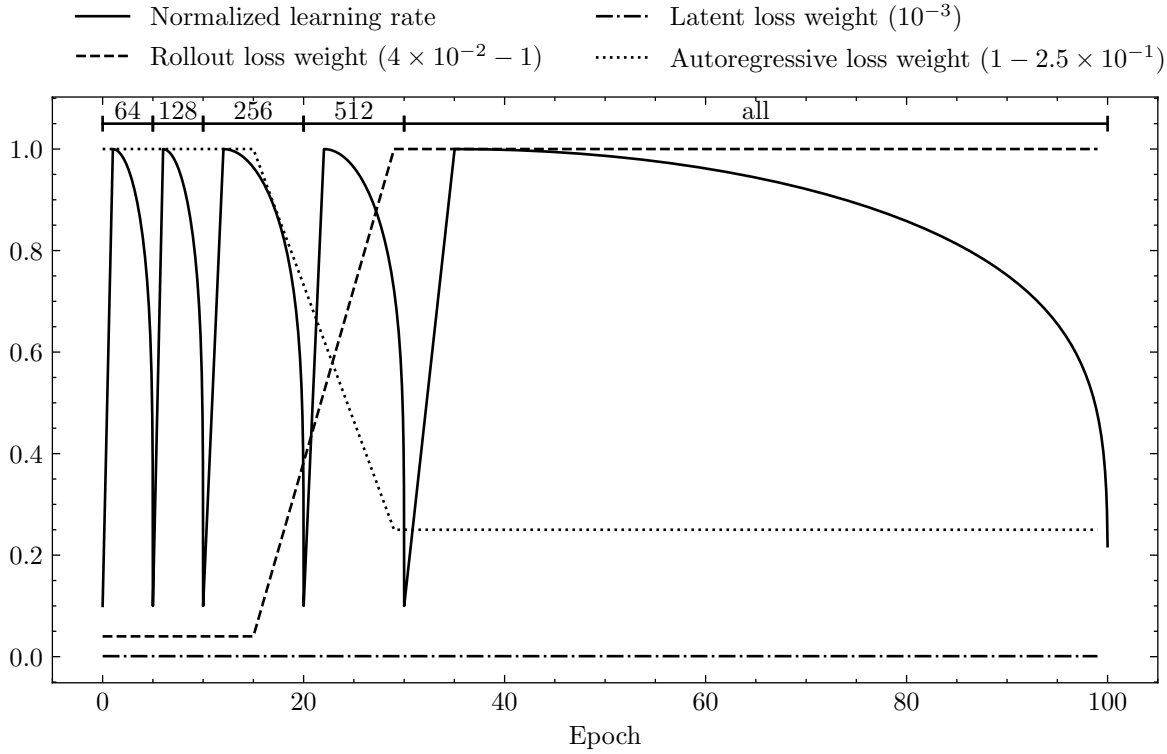

Figure 4. The scheduling of the learning rate and the weights of the loss function for the training on dataset v3.

3. Results

For each of the three datasets, we train the models using 70% of the available data, using 15% as validation set and keeping 15% available as a test set, which is the set we use for the figures in the results section. The Mean Absolute Erorr (MAE) we compute now in the log-space, without standardization; this results in a scaling of the mean of the test set compared to training and validation sets by a factor of 3.

3.1. Homogenous models in one dimension

The one-dimensional model takes 81 minutes to train (using an NVIDIA V100), reaching a final validation loss of $\mathcal{L}_{val} = 0.02$. The loss curves can be found in Figure 5. These show that the training loss decreases quickly during the first 15 epochs, with the validation loss, which is evaluated using only the rollout loss term, lacking behind. We can see a small increase in the loss after expanding the length of the series. After the 15th epoch, as the autoencoder loss weight start decreasing and the rollout loss weight starts increasing, the training loss start increasing with the validation loss coming down, indicating that the latent NODE is being trained effectively. Once the loss weights are constant again at epoch 30, the training loss starts decreasing again. The validation loss is lower than the training loss, indicating that there is a trade-off between the autoregressive and latent loss.

We show both the data and rollout prediction for one sample from the test dataset in Figure 6. The plot is constrained to a subset of species to allow for easier comparison. It shows a chemistry that is evolving as soon as the visual extinction reaches $A_V = 0.1$, with the auxiliary gas temperature and radiation field rapidly decreasing. The rollout predictions follow the data, but then as the chemistry starts changing more rapidly around $A_V = 7$, it fails to capture the rapid dynamics, instead smoothing out the chemical evolution. In the end, however, it does recover and converges to the steady-state solution of the chemistry. The over smoothened prediction for the chemistry at intermediate $A_V$can be seen as a peak in the error in Figure 8, indicating that the surrogate model could still be improved there. The error does quickly reduce after the peak, indicating the approximation can correctly predict the steady state solution without a catastrophic buildup of the error at intermediate A_V. The error does not show a similar peak as a function of the index, since the visual extinction at which the chemistry rapidly changes depends on the initial radiation field, density and cosmic ray ionization rate, the largest changes occur at different indices within the series, resulting in no distinct peak in error, only a slightly larger error at the end of each series.

3.2. Variable density models in one dimension

The variable density model has a similar loss curve, as can be seen in Figure 5, with the training time taking 32 minutes (using an NVIDIA V100). However due to the smaller size of the dataset and greater physical complexity, the performance is not as great as the *v1* model at a similar number of epochs. We see a similar pattern appear with the train and validation losses, where the validation loss seems to converge well after a peak in loss after increasing the series length at epoch 30. The final validation loss it achieves is $\mathcal{L}_{val} = 0.076$.

The greater chemical complexity due to the increase in density is reflected in the fact that there are now several small jumps in the data, as can be seen in Figure 7. The neural network provides smooth interpolations, but the complexity of the surrogate model is not great enough to capture the quick changes in chemistry, indicating it must

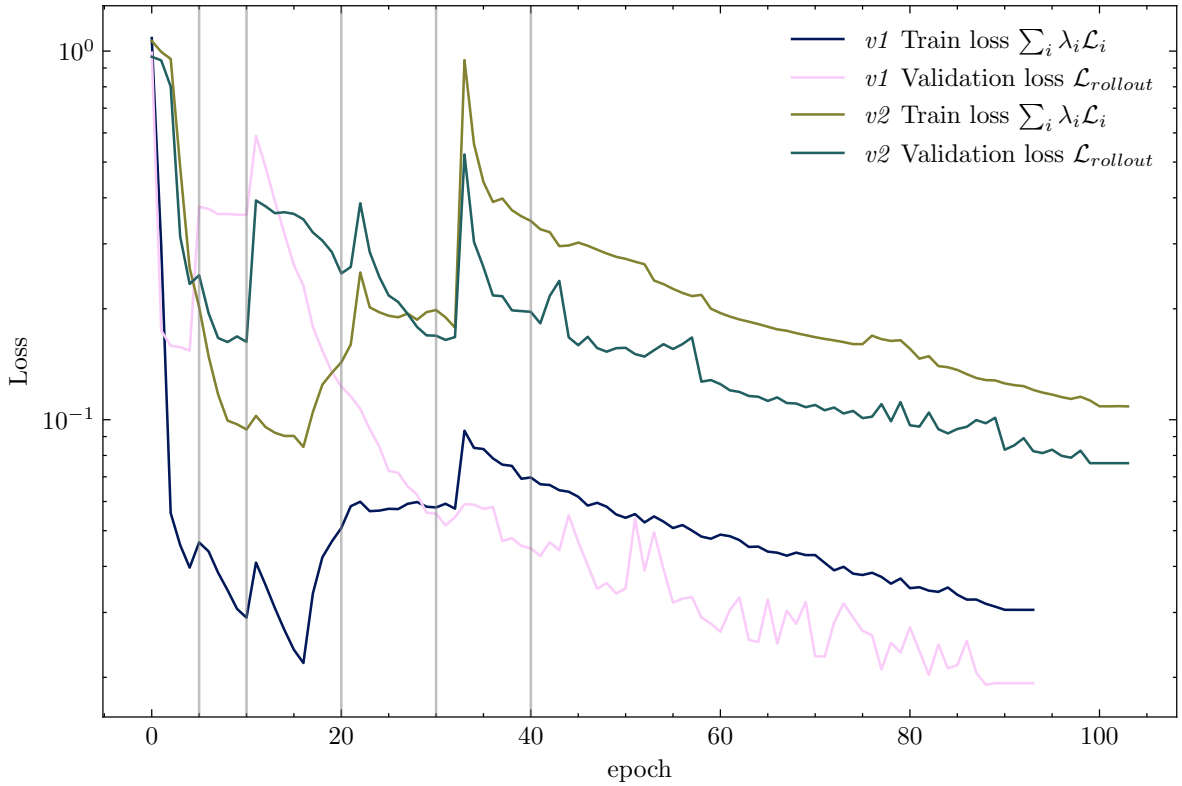

Figure 5. The training and validation loss curves for dataset *v1* and *v2*

either be trained longer still or have a larger model complexity. This is reflected by the loss as a function of index and visual extinction as shown in Figure 9. It again has a peak, after which the error decreases as the surrogate converges to the steady-state solution of the chemistry. The lower performance of the dataset *v2* than *v1* thus motivates the choice to use larger MLPs and latent size and more series to train on the dynamics of the *v3* dataset.

3.3. Interstellar medium models in three dimensions

3.3.1. Varying the batch and latent bottleneck size We proceed to train the surrogate model on the three-dimensional dataset. We tried several combinations of latent bottleneck size l and batch size B, as listed in Table 3. The resulting validation loss curves can be found in Figure 10. This shows that trying to utilize the smaller bottleneck sizes does not result in the surrogate models training successfully. The end of all these runs is marked by the latent differential equation producing a Not a Number in a batch, which can happen when an integrator tries to integrate a badly constrained function. Since these runs with bottleneck sizes of $l = \{8, 16, 32\}$ did not show any improvement in the loss, the runs were not resumed. The model with $l = 64$ does improve in loss at the start of training, but in epoch 42 the training results in Not a Number gradients, effectively halting the training process. This Not a Number gradient is caused by the ODE solver not converging, resulting in the maximum number of integration steps being

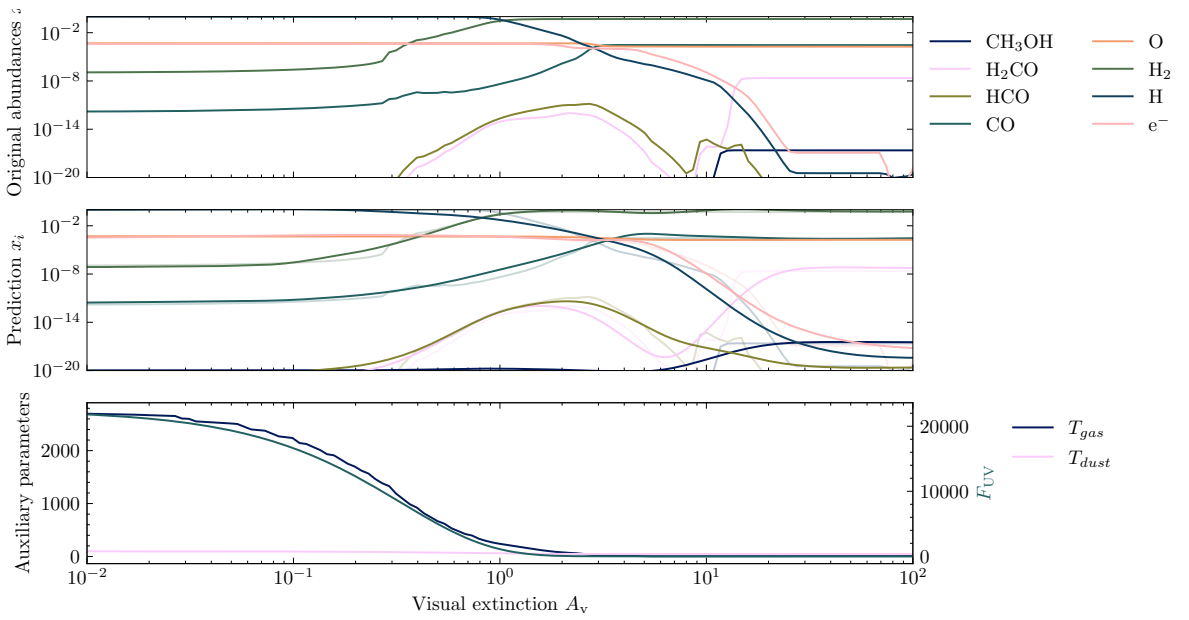

Figure 6. A comparison between a test sample from *v1* and its prediction.

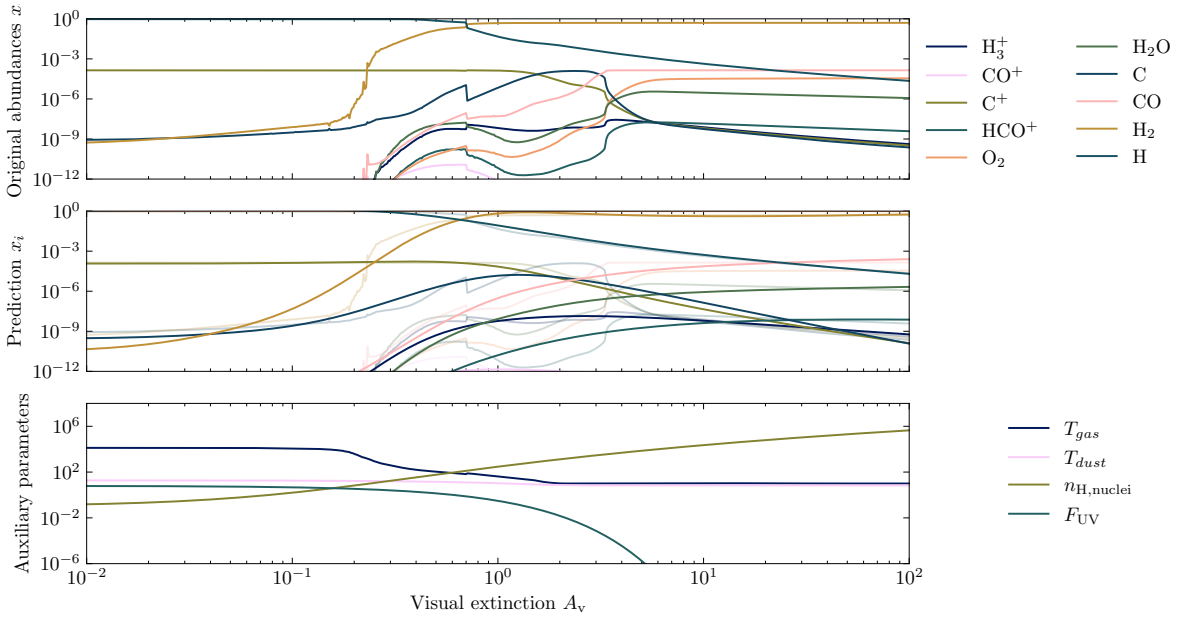

Figure 7. A comparison between a test sample from *v2* and its prediction.

reached. Upon restarting at epoch 40 with the same weights, it quickly results in another Not a Number gradient, indicating that the weights are not converging towards a stable solution. Thus, the hyperparameter configuration is discarded. This only leaves the runs with the largest latent bottleneck size $l = 128$. For the lowest batch size $l = 32$, the loss seemed to improve the fastest, but in epoch 28, a Not a Number gradient occurs, and trying to resume the training process quickly results in other Not a Number losses, effectively discarding the hyperparameter configuration. This only leaves the

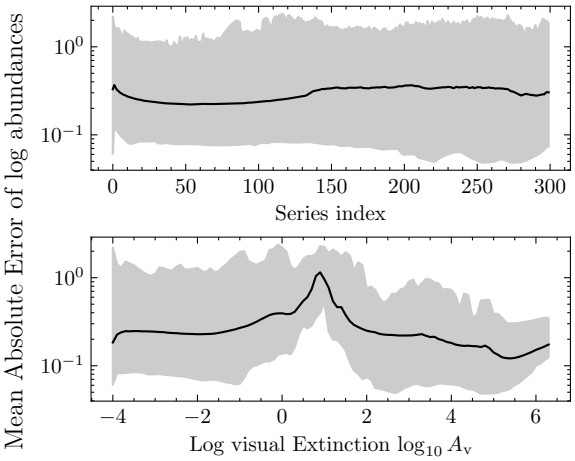

Figure 8. The MAE in log space for the *v1* test dataset.

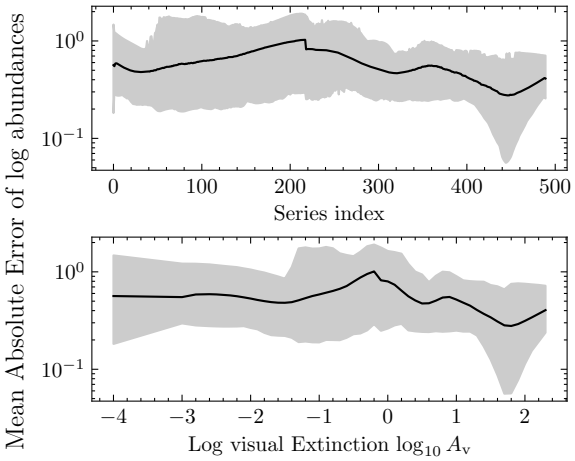

Figure 9. The MAE in log space for the *v2* test dataset.

batch sizes $B = \{64, 128\}$, with the latter needing a restart after dealing with Not a Number gradients in epoch 26, but then it does train successfully until epoch 84. We subsequently choose the only run that ran continuously to achieve the lowest validation loss of $\mathcal{L}_{val} = 2.6 \times 10^{-3}$ in 94 epochs.

3.3.2. Depth dependent approximation and column density maps We now take the best-performing model, and see how well we perform on the test dataset. To inspect the performance of the surrogate, we select a sample with a high carbon monoxide to carbon ratio. This ratio indicates that the ray has traced a high density region, resulting in the attenuation of the radiation and decrease in temperature, subsequently allowing for the formation of molecules (especially CO, HCO^+ and H_2O) in the cold and dense gas. The original unsmoothed data, smoothed training data and prediction are shown in Figure 11. It shows clearly that between $A_V = [0.2, 0.4]$ a high density region is traced, resulting in more complex molecules to peak with CO being as abundant as 10^{-4}. We see that compared to the original data, the smoothing of the data has resulted in a less wide peak, meaning that the integral of the peak is lower. The neural network correctly predicts the peak of the more complex molecules, and the subsequent loss of them as the density drops, again increasing the temperature and radiation field.

The evolution of the error on the test set as a function of index and visual extinction is shown in fig. 12. This shows that the MAE moves around 0.1 in log abundance space. As the rollout increases beyond index 300, we start to see an increase in the error, indicating the errors are accumulating in the latent space. Since there are only few models that proceed until these higher visual extinctions, see fig. 3, the surrogate model has not fit these longer rays as well as the shorter ones. We can see this rapid increase in error in the bottom visual extinction plot as well.

We then take all the rays from the test set, and derive the column density maps. These column density N_i (cm^{-2}) maps integrate the number densities n_i (cm^{-3}) of

each molecule along the lines of sight, resulting in an image that can be compared to observations. In order to go from the rays back to these images, we must first compute the number densities for the entire three-dimensional object. We choose a three-dimensional grid of $256 \times 256 \times 256$ cells, and then compute the mean fractional abundance of each molecule $x_{i,x,y,z}$ for each cell. We can then recover the column density by multiplying each fractional abundance by the density of the cells $n_{\mathrm{H,nuclei}}$, and then summing this quantity over each cell that is non-zero, multiplying by the depth of each cell $\Delta z = 0.44$ parsec. This results in maps of each species. We show the column densities of atomic hydrogen H, molecular hydrogen H_2 and carbon monoxide (CO) in Figure 13. Here we can see that even with the smoothing of the data, the maps of both atomic and molecular hydrogen are recovered well. Atomic hydrogen traces regions of intermediate density, where it is more abundant, but is not yet captured in molecular hydrogen at lower temperatures. In the lower parts of the images, we see the higher density and low-temperature regions, where the hydrogen is captured in its molecular form. We can also see how the rays with high visual extinction pass through several structures with higher densities. Lastly, we can see the effect of the smoothing on the CO column densities. Its density is reduced by smoothing the data, resulting in both a lower peak value and a less extended region. Individual errors for each molecule can be found in Appendix B.

Lastly, we investigate the relationship between the individual error of each prediction compared to the standard deviation of each abundance. This tells us whether the surrogate models have learned equally for each of the molecules. The result can be seen in Figure 14; here we can see that all species lie on a straight line, indicating that the error in the prediction scales with the dynamic range of each species. Species that barely vary, namely ionized carbon C^+ and e^-, only change in abundance when they recombine in the highest-density areas, as seen in Figure 11, and thus their predictions have the lowest error. The species with the higher dynamic ranges have a larger error, which makes sense, as the latent differential equation can only try to approximate them, accumulating some error as it integrates and smoothing high-frequency changes.

3.3.3. Computational cost of training and inference and critical speedup The latent differential equations for the best hyperparameter configuration took approximately 84 GPU hours with an NVIDIA H100. This highlights that NODEs are expensive to train for a relatively small data volume of 159K samples. The many failed runs underline the instability and challenges of training neural differential equations. Nevertheless, the resulting surrogate model performs well enough to reconstruct both the depth-dependent chemistry and the resulting mock observation at a much lower computational cost at inference. The inference of all 159K samples takes 200 seconds without any optimization for throughput. This means the whole dataset could be inferred in little over 8 GPU hours compared to the 1792 CPU hours needed for generating the original dataset. This results in a considerable speedup and the effective utilization of the GPU, allowing the CPU to be utilized for gravity, hydrodynamics, and radiative transport.

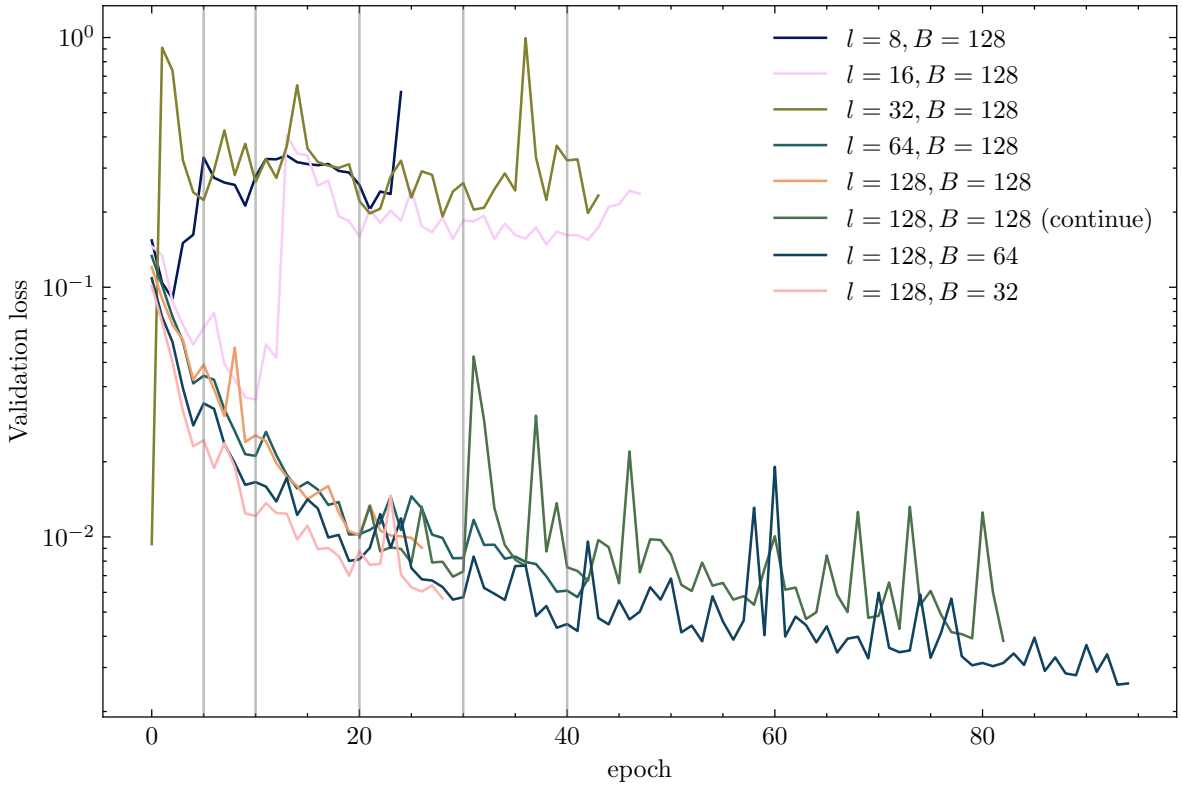

Figure 10. The loss curves for different hyperparameters latent bottleneck size l and batch size B, as the latent bottleneck size is decreased, training becomes decreasingly stable. Smaller batch size seem to improve performance, but for $B = 32$ training became instable after 28 epochs.

4. Conclusion and discussion

We have shown that the latent neural differential equation architecture can be scaled up to be trained on data from three-dimensional simulations of the interstellar medium. This enables fast inference of the astrochemistry without the need for the classical codes. This speedup is essential for high-resolution simulations of astrophysical objects and the statistical inference of observations.

The dynamics of the first two datasets *v1* and *v2* is approximated reasonably well with worse performance in the intermediate regime of visual extinction, with rapidly changing abundances. As soon as the chemistry equilibrates, however, the surrogate model achieves lower error when approximating the steady-state solution at the end of the series. This indicates the latent dynamics could still be improved to capture the faster regime of the chemical evolution more accurately. Potential improvements could include more physics-informed losses, such as the derivative of the abundances. Furthermore, the performance on these datasets can be further improved by a more extensive hyperparameter performance optimization (HPO) than was in the scope of this work, also training these networks until there is no further improvement in the loss function.

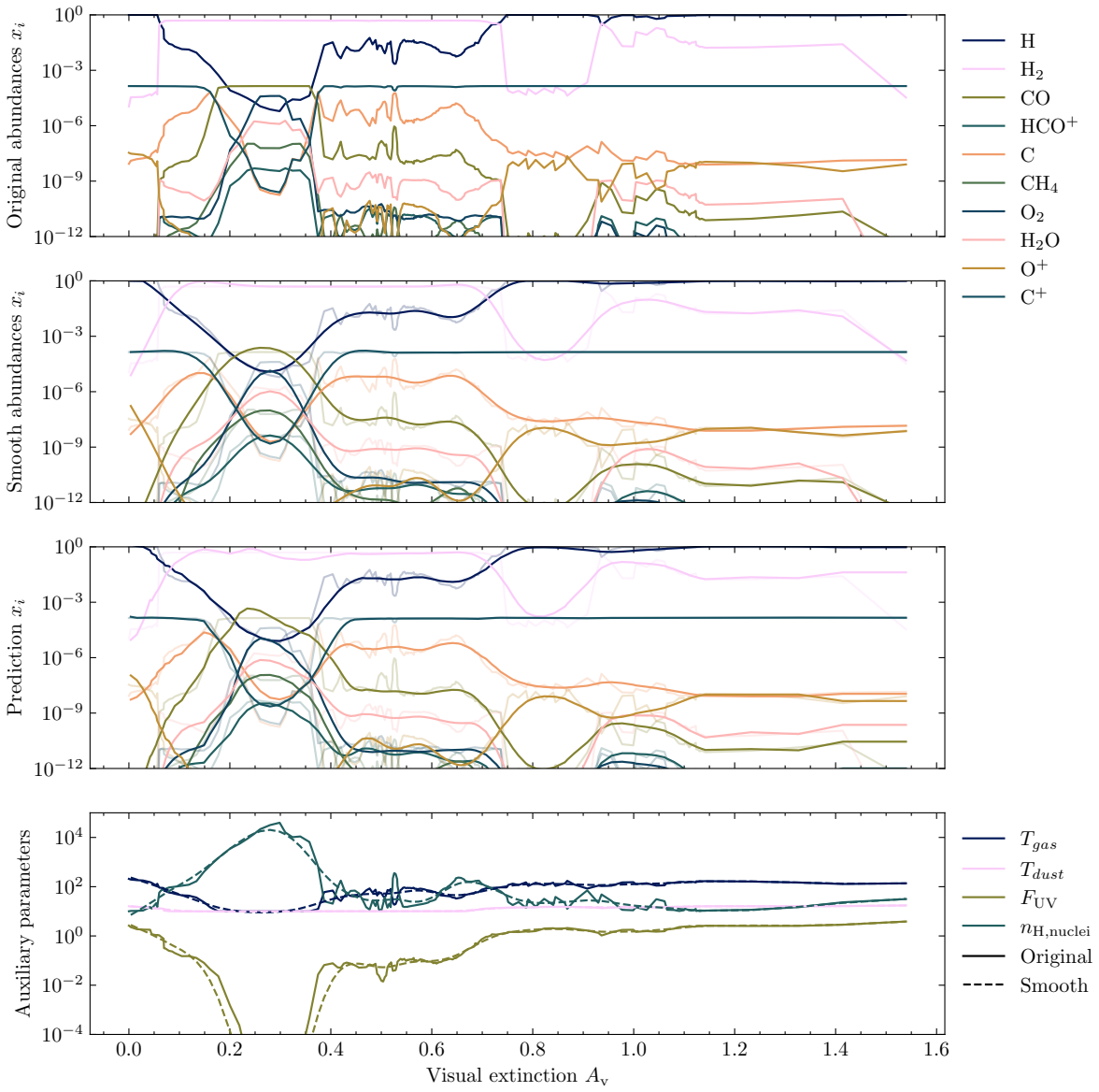

Figure 11. The original, smooth training and prediction abundances for a series with a peak in abundance at low visual extinction.

For the three-dimensional dataset, we have shown that we can train a surrogate model with good enough accuracy to reproduce the dynamics of the smooth dataset, and to a lesser degree the dynamics of the original dataset. However, this is the first time that a surrogate model was trained directly on the post-processed chemistry of a three-dimensional astronomical simulation.

The process of training the latent NODEs is still a non-trivial task, as highlighted by the many hyperparameter configurations that encounter a Not a Number gradient at some point. The fact that these NODEs are trained with single precision for the sake of performance means that the neural network can move towards a set of weights that causes the internal differential equation solver to no longer be stable and either cause the

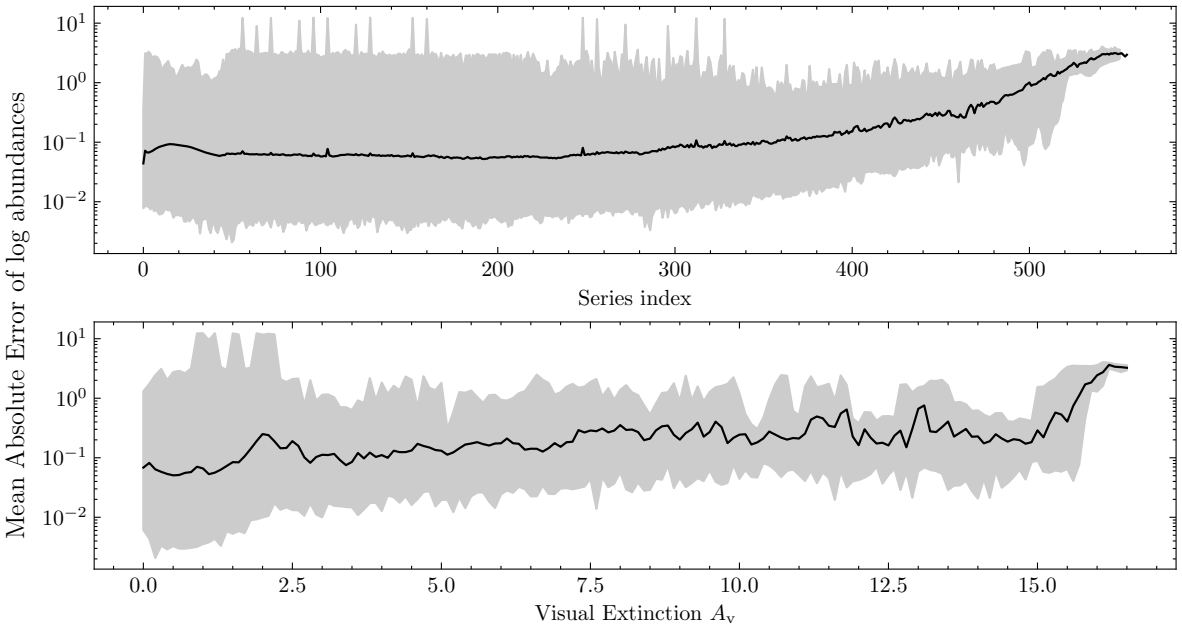

Figure 12. Figure showing the MAE for *v3* as a function of both the index and the visual extinction.

solver to fail, introducing Not a Numbers directly, or introduce Not a Number gradients indirectly, both effectively halting the training process. The fact that reverting to a few epochs earlier with the same weights quickly results in another Not a Number batch indicates that this is a fundamental problem and not a one-off problem with a singular bad batch. Potential solutions for these instable learning dynamics could be introducing double precision training or using a stiff solver, such as (Kværnø; 2004).

The resulting surrogate model then was used to generate column density maps of the simulation. By either removing the numerical jumps from `3D-PDR` directly or introducing a more advanced smoothing algorithm, the dynamics of the system should be recovered more accurately. A more detailed HPO must also be performed to see how the error in both the high visual extinction and high density regime can be reduced. Additionally, the fact that the chemistry cannot be compressed to a lower dimensional latent space requires further investigation. A hypothesis is that the latent space is forced to embed both the chemical state and physical history of the system directly, and thus a larger latent space is required; this could also explain why there is a trade-off between the autoregressive and rollout losses. Further experiments with the structure of the latent space, and compression methods, such as dynamic sparsity (Correia et al.; 2020) might help resolve this problem. Furthermore, this work only proposed the usage of one architecture, and the only benchmark provided is the accuracy of the autoencoder versus the rollout. It should be benchmarked (Janssen et al.; 2024) against architectures that also include rollout, e.g. (Branca and Pallottini; 2024; Maes et al.; 2024; Ramos et al.; 2024) Additionally, new architectures for time series such as xLSTM(Beck et al.; 2024) and Mamba(Gu and Dao; 2024) have not yet been applied to this domain either.

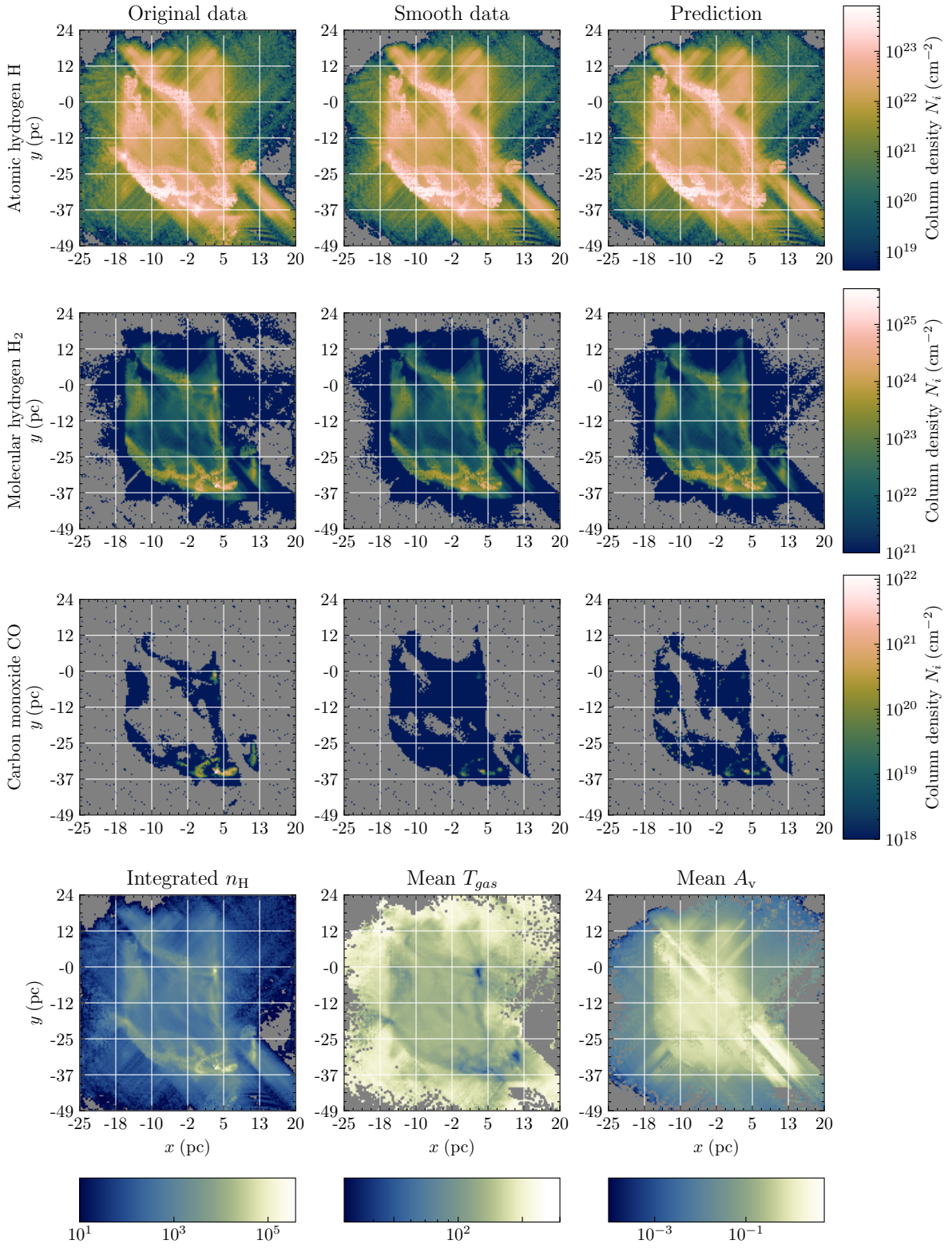

Figure 13. Integrated column densities, representing the integral of the predicted number density of a species along each line of sight in the z-direction of each species. In the bottom the integrated number density, mean gas temperature and mean visual extinction are shown

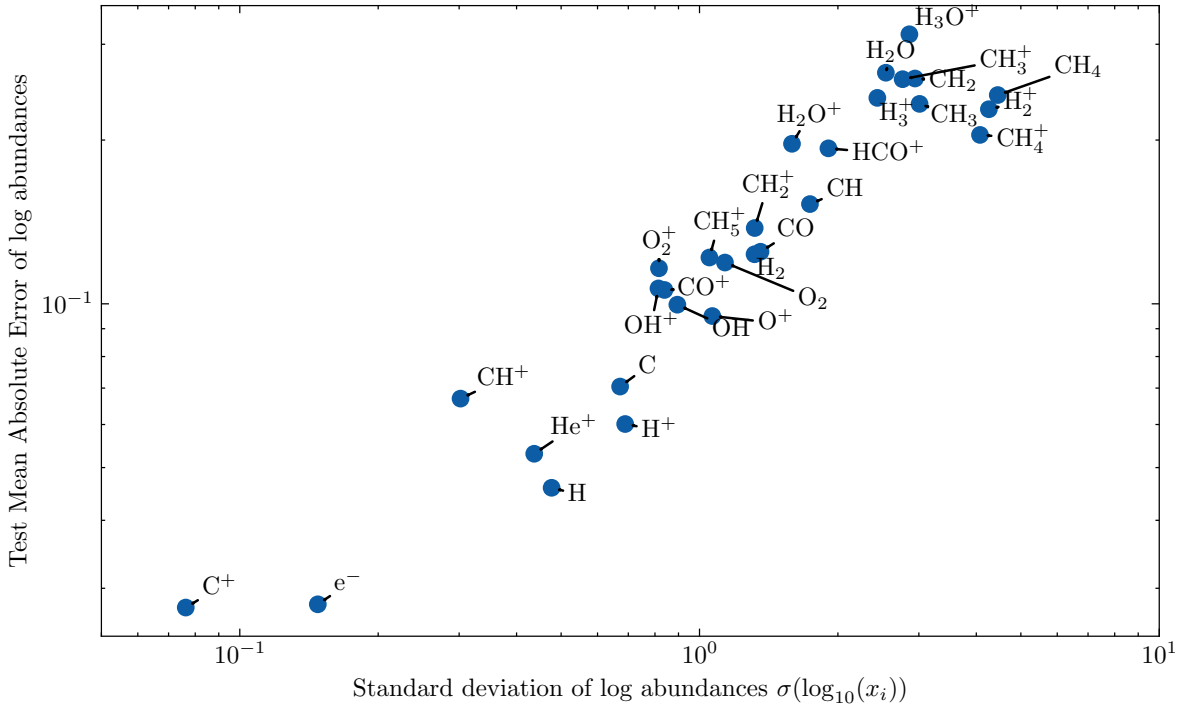

Figure 14. A comparison of the standard deviation of each species versus the Mean Absolute Error in log space

Acknowledgements

S.V. acknowledges support from the European Research Council (ERC) Advanced grant MOPPEX 833460. T.G.B. acknowledges support from the Leading Innovation and Entrepreneurship Team of Zhejiang Province of China (Grant No. 2023R01008). The authors declare that they have no competing interests.

The ANODEs were implemented using `diffrax`(Kidger; 2022) and `jax`(Bradbury et al.; 2018). Plots were made using `matplotlib` (Hunter; 2007) with the colormaps from (Crameri; 2023). The dataset are serialized using `h5py` (Collette; 2013).

References

Beck, M., Pöppel, K., Spanring, M., Auer, A., Prudnikova, O., Kopp, M., Klambauer, G., Brandstetter, J. and Hochreiter, S. (2024). xLSTM: Extended Long Short-Term Memory.

Bisbas, T. G., Bell, T. A., Viti, S., Yates, J. and Barlow, M. J. (2012). 3D-PDR: A new three-dimensional astrochemistry code for treating Photodissociation Regions, *Monthly Notices of the Royal Astronomical Society* **427**(3): 2100–2118.

Bisbas, T. G., van Dishoeck, E. F., Hu, C.-Y. and Schruba, A. (2023). PDFCHEM: A new fast method to determine ISM properties and infer environmental parameters

using probability distributions, *Monthly Notices of the Royal Astronomical Society* **519**: 729–753.

Bovino, S. and Grassi, T. (2023). *ASTROCHEMICAL MODELLING Practical Aspects of Microphysics in Numerical*, ELSEVIER - HEALTH SCIENCE, S.l.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S. and Zhang, Q. (2018). JAX: Composable transformations of Python+NumPy programs.

Branca, L. and Pallottini, A. (2022). Neural networks: Solving the chemistry of the interstellar medium, *Monthly Notices of the Royal Astronomical Society* **518**(4): 5718–5733.

Branca, L. and Pallottini, A. (2024). Emulating the interstellar medium chemistry with neural operators.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J. and Duvenaud, D. (2019). Neural Ordinary Differential Equations.

Collette, A. (2013). *Python and HDF5*, O'Reilly.

Correia, G. M., Niculae, V., Aziz, W. and Martins, A. F. T. (2020). Efficient Marginalization of Discrete and Structured Latent Variables via Sparsity.

Crameri, F. (2023). Scientific colour maps, Zenodo.

de Mijolla, D., Viti, S., Holdship, J., Manolopoulou, I. and Yates, J. (2019). Incorporating astrochemistry into molecular line modelling via emulation, *Astronomy and Astrophysics* **630**: A117.

Draine, B. T. (1978). Photoelectric heating of interstellar gas, *The Astrophysical Journal Supplement Series* **36**: 595.

Dupont, E., Doucet, A. and Teh, Y. W. (2019). Augmented Neural ODEs.

Ge, Z., Kaushik, L., Omote, M. and Kumar, S. (2021). Speed up Training with Variable Length Inputs by Efficient Batching Strategies, *Interspeech 2021*, ISCA, pp. 156–160.

Gong, M., Ho, K. W., Stone, J. M., Ostriker, E. C., Caselli, P., Grassi, T., Kim, C.-G., Kim, J.-G. and Halevi, G. (2023). Implementation of Chemistry in the Athena++ Code, *The Astrophysical Journal Supplement Series* **268**: 42.

Górski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M. and Bartelmann, M. (2005). HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere, *The Astrophysical Journal* **622**(2): 759.

Grassi, T., Nauman, F., Ramsey, J. P., Bovino, S., Picogna, G. and Ercolano, B. (2021). Reducing the complexity of chemical networks via interpretable autoencoders.

Grassi, T., Padovani, M., Galli, D., Vaytet, N., Jensen, S. S., Redaelli, E., Spezzano, S., Bovino, S. and Caselli, P. (2025). Mapping Synthetic Observations to Prestellar Core Models: An Interpretable Machine Learning Approach.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I. and Duvenaud, D. (2018). FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models.

Grudić, M. Y., Guszejnov, D., Hopkins, P. F., Offner, S. S. R. and Faucher-Giguère, C.-A. (2021). STARFORGE: Toward a comprehensive numerical model of star cluster formation and feedback, *Monthly Notices of the Royal Astronomical Society* **506**(2): 2199–2231.

Gu, A. and Dao, T. (2024). Mamba: Linear-Time Sequence Modeling with Selective State Spaces.

Güver, T. and Özel, F. (2009). The relation between optical extinction and hydrogen column density in the Galaxy, *Monthly Notices of the Royal Astronomical Society* **400**(4): 2050–2053.

Heyl, J., Butterworth, J. and Viti, S. (2023). Understanding molecular abundances in star-forming regions using interpretable machine learning, *Monthly Notices of the Royal Astronomical Society* **526**(1): 404–422.

Holdship, J., Viti, S., Haworth, T. J. and Ilee, J. D. (2021). Chemulator: Fast, accurate thermochemistry for dynamical models through emulation, *Astronomy & Astrophysics* **653**: A76.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment, *Computing in Science & Engineering* **9**(3): 90–95.

Janssen, R., Sulzer, I. and Buck, T. (2024). CODES: Benchmarking Coupled ODE Surrogates.

Kidger, P. (2022). On Neural Differential Equations.

Kramer, M. A. (1991). Nonlinear Principal Component Analysis Using Autoassociative Neural Networks, *AIChE Journal* **37**(2).

Kværnø, A. (2004). Singly Diagonally Implicit Runge–Kutta Methods with an Explicit First Stage, *BIT Numerical Mathematics* **44**(3): 489–502.

Lee, K. and Parish, E. J. (2021). Parameterized neural ordinary differential equations: Applications to computational physics problems, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences* **477**(2253): 20210162.

Loshchilov, I. and Hutter, F. (2016). SGDR: Stochastic Gradient Descent with Warm Restarts, https://arxiv.org/abs/1608.03983v5.

Loshchilov, I. and Hutter, F. (2017). Decoupled Weight Decay Regularization, https://arxiv.org/abs/1711.05101v3.

Maes, S., De Ceuster, F., Van de Sande, M. and Decin, L. (2024). MACE: A Machine learning Approach to Chemistry Emulation.

Pascanu, R., Mikolov, T. and Bengio, Y. (2013). On the difficulty of training Recurrent Neural Networks.

Peeters, E., Habart, E., Berné, O., Sidhu, A., Chown, R., Putte, D. V. D., Trahin, B., Schroetter, I., Canin, A., Alarcón, F., Schefter, B., Khan, B., Pasquini, S., Tielens,

A. G. G. M., Wolfire, M. G., Dartois, E., Goicoechea, J. R., Maragkoudakis, A., Onaka, T., Pound, M. W., Vicente, S., Abergel, A., Bergin, E. A., Bernard-Salas, J., Boersma, C., Bron, E., Cami, J., Cuadrado, S., Dicken, D., Elyajouri, M., Fuente, A., Gordon, K. D., Issa, L., Joblin, C., Kannavou, O., Lacinbala, O., Languignon, D., Gal, R. L., Meshaka, R., Okada, Y., Robberto, M., Röllig, M., Schirmer, T., Tabone, B., Zannese, M., Aleman, I., Allamandola, L., Auchettl, R., Baratta, G. A., Bejaoui, S., Bera, P. P., Black, J. H., Boulanger, F., Bouwman, J., Brandl, B., Brechignac, P., Brünken, S., Buragohain, M., Burkhardt, A., Candian, A., Cazaux, S., Cernicharo, J., Chabot, M., Chakraborty, S., Champion, J., Colgan, S. W. J., Cooke, I. R., Coutens, A., Cox, N. L. J., Demyk, K., Meyer, J. D., Foschino, S., García-Lario, P., Gerin, M., Gottlieb, C. A., Guillard, P., Gusdorf, A., Hartigan, P., He, J., Herbst, E., Hornekaer, L., Jäger, C., Janot-Pacheco, E., Kaufman, M., Kendrew, S., Kirsanova, M. S., Klaassen, P., Kwok, S., Labiano, Á., Lai, T. S.-Y., Lee, T. J., Lefloch, B., Petit, F. L., Li, A., Linz, H., Mackie, C. J., Madden, S. C., Mascetti, J., McGuire, B. A., Merino, P., Micelotta, E. R., Misselt, K., Morse, J. A., Mulas, G., Neelamkodan, N., Ohsawa, R., Paladini, R., Palumbo, M. E., Pathak, A., Pendleton, Y. J., Petrignani, A., Pino, T., Puga, E., Rangwala, N., Rapacioli, M., Ricca, A., Roman-Duval, J., Roser, J., Roueff, E., Rouillé, G., Salama, F., Sales, D. A., Sandstrom, K., Sarre, P., Sciamma-O'Brien, E., Sellgren, K., Shenoy, S. S., Teyssier, D., Thomas, R. D., Togi, A., Verstraete, L., Witt, A. N., Wootten, A., Ysard, N., Zettergren, H., Zhang, Y., Zhang, Z. E. and Zhen, J. (2024). PDRs4All - III. JWST's NIR spectroscopic view of the Orion Bar, *Astronomy & Astrophysics* **685**: A74.

Ramos, A. A., Plaza, C. W., Navarro-Almaida, D., Rivière-Marichalar, P., Wakelam, V. and Fuente, A. (2024). A fast neural emulator for interstellar chemistry, *Monthly Notices of the Royal Astronomical Society* **531**(4): 4930–4943.

Rollig, M., Abel, N. P., Bell, T., Bensch, F., Black, J., Ferland, G. J., Jonkheid, B., Kamp, I., Kaufman, M. J., Bourlot, L., Petit, F. L., Meijerink, R., Chirivella, O. M., Ossenkopf, V., Roueff, E., Shaw, G., Sternberg, A. and Stutzki, J. (2007). A PDR-Code comparison study.

Rubanova, Y., Chen, R. T. Q. and Duvenaud, D. (n.d.). Latent ODEs for Irregularly-Sampled Time Series.

Seifried, D., Haid, S., Walch, S., Borchert, E. M. A. and Bisbas, T. G. (2020). SILCC-Zoom: H2 and CO-dark gas in molecular clouds - the impact of feedback and magnetic fields, *Monthly Notices of the Royal Astronomical Society* **492**: 1465–1483.

Seifried, D., Walch, S., Girichidis, P., Naab, T., Wünsch, R., Klessen, R. S., Glover, S. C. O., Peters, T. and Clark, P. (2017). SILCC-Zoom: The dynamic and chemical evolution of molecular clouds, *Monthly Notices of the Royal Astronomical Society* **472**(4): 4797–4818.

Sulzer, I. and Buck, T. (2023). Speeding up astrochemical reaction networks with autoencoders and neural ODEs.

Tang, K. S. and Turk, M. (2022). Reduced Order Model for Chemical Kinetics: A case study with Primordial Chemical Network.

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

Vermariën, G., Viti, S., Ravichandran, R. and Bisbas, T. G. (2024). 3D-PDR Orion dataset and NeuralPDR: Neural Differential Equations for Photodissociation Regions.

Wolfire, M. G., Vallini, L. and Chevance, M. (2022). Photodissociation and X-Ray-Dominated Regions, *Annual Review of Astronomy and Astrophysics* **60**(Volume 60, 2022): 247–318.

Yue, N., Wang, L., Bisbas, T., Quan, D. and Li, D. (2024). Turbulent Diffuse Molecular Media with Nonideal Magnetohydrodynamics and Consistent Thermochemistry: Numerical Simulations and Dynamic Characteristics, *The Astrophysical Journal* **973**(1): 37.

Zemlyanoy, E. (2022). *Construction of Smoothing Splines by the Generalized Cross-Validation Method*, Bachelor Thesis, HSE University, Moscow.

Appendix A. Additional loss curve

In Figure A1, the loss curves for the *v3* model is shown including the train error per batch and per epoch.

Appendix B. Additional errors maps

In Figure B1 and Figure B2, the error or each of the molecules is shown on the 2-dimensional map.

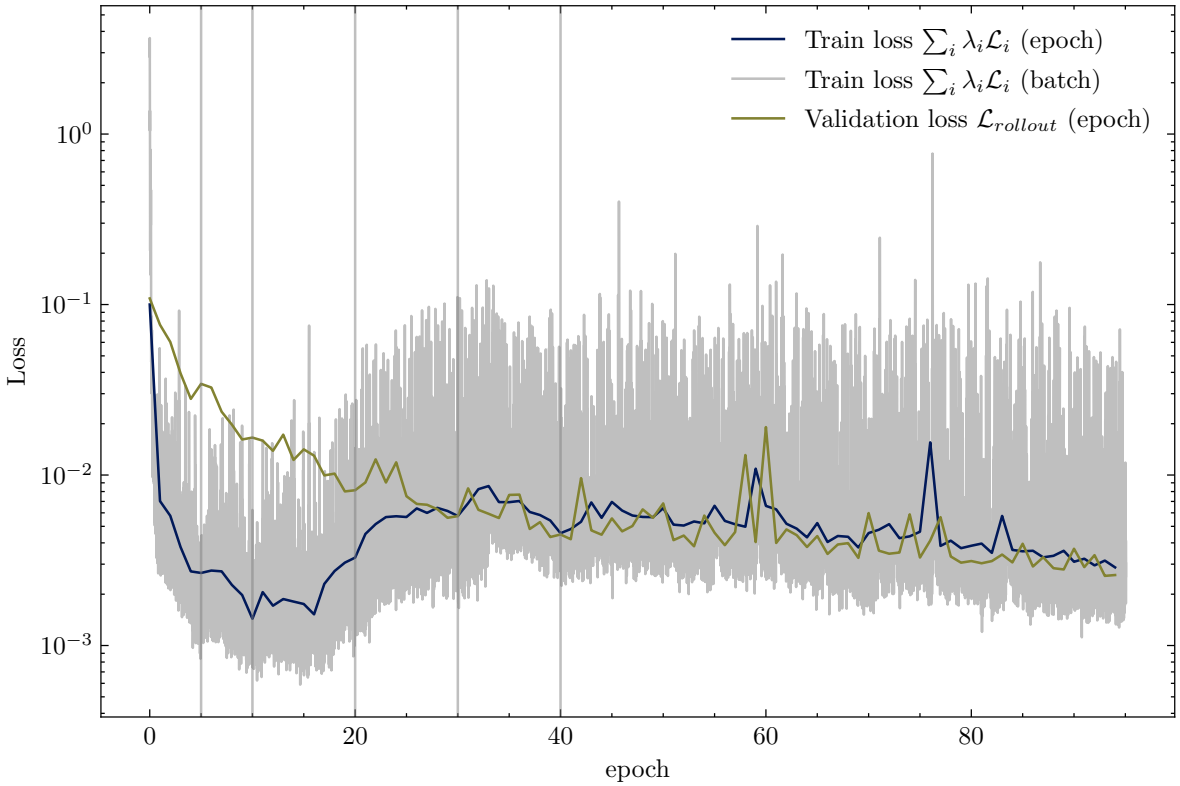

Figure A1. The loss curves for batch size 64 and latent bottleneck size 128 training on dataset *v3*.

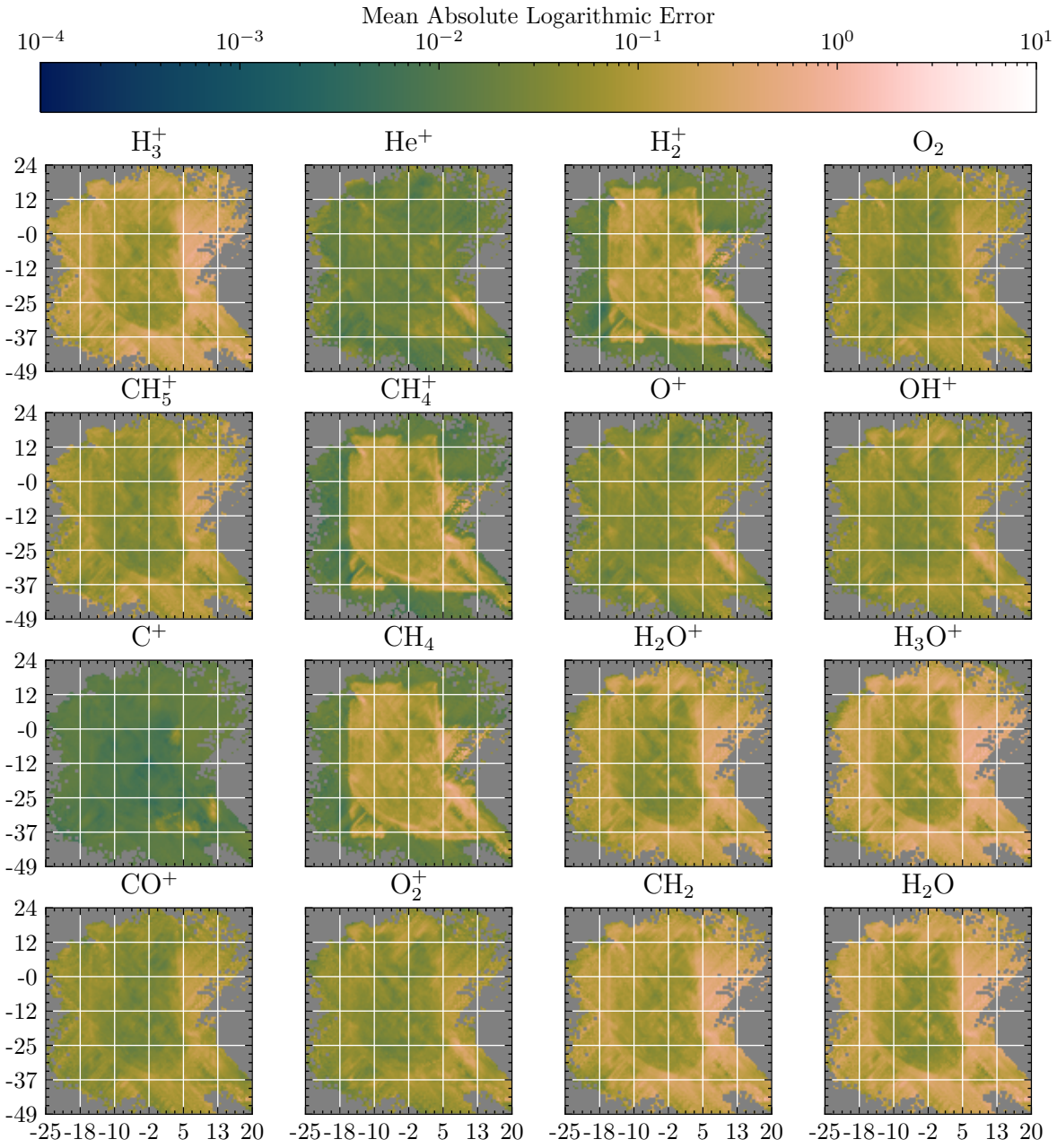

Figure B1. The mean logarithmic error per cell, mapped onto the same sight lines as in Figure 13.

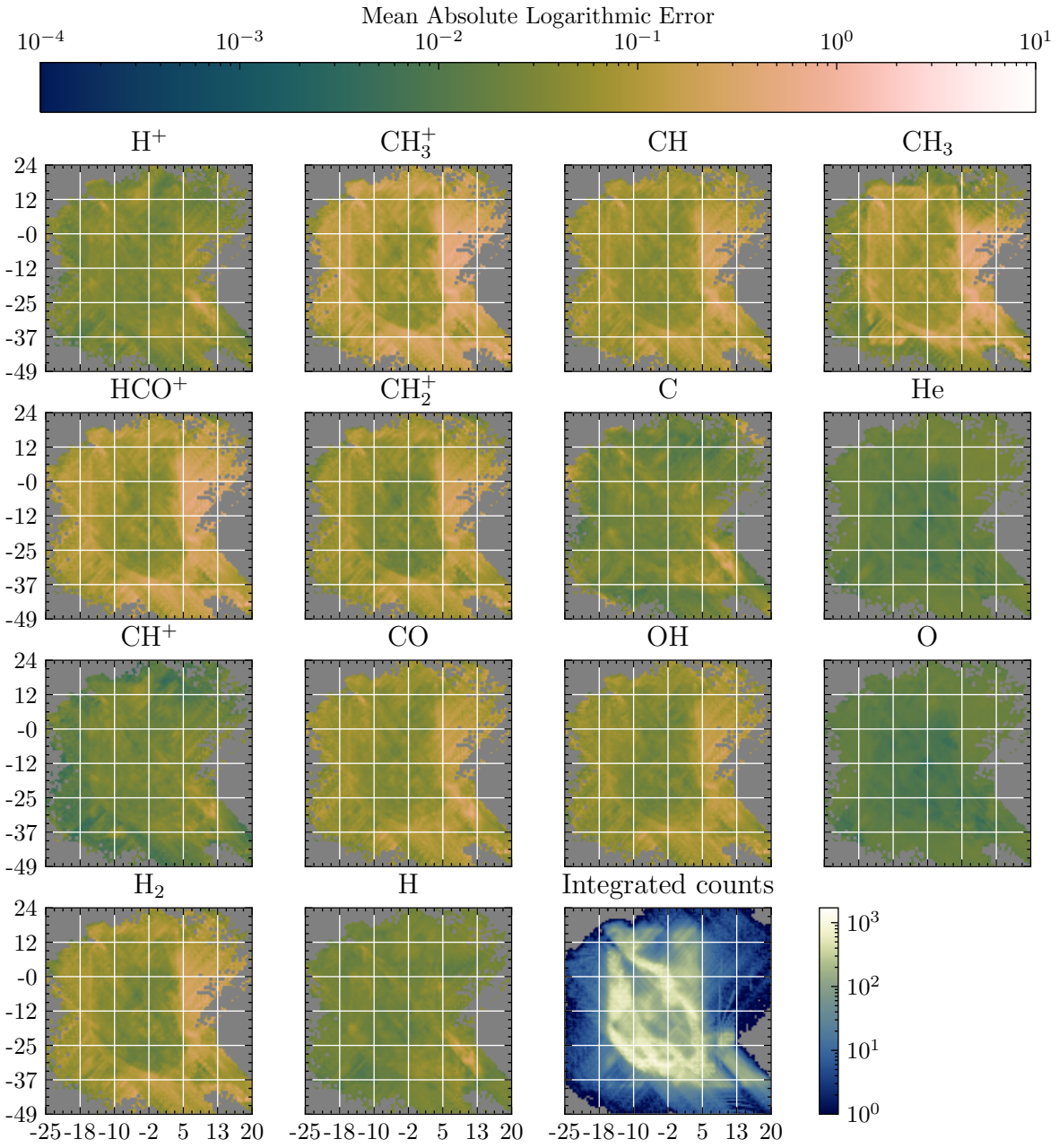

Figure B2. The mean logarithmic error per cell, mapped onto the same sight lines as in Figure 13.