


Chemulator: Fast, accurate thermochemistry for dynamical models through emulation

J. Holdship^{1,2} , S. Viti^{1,2}, T. J. Haworth³, and J. D. Ilee⁴

¹ Leiden Observatory, Leiden University, PO Box 9513, 2300 RA Leiden, The Netherlands
e-mail: holdship@strw.leidenuniv.nl

² Department of Physics and Astronomy, University College London, Gower Street, WC1E 6BT London, UK

³ Astronomy Unit, School of Physics and Astronomy, Queen Mary University of London, London E1 4NS, UK

⁴ School of Physics & Astronomy, University of Leeds, Woodhouse Lane, Leeds LS2 9JT, UK

Received 15 January 2021 / Accepted 28 June 2021

ABSTRACT

Context. Chemical modelling serves two purposes in dynamical models: accounting for the effect of microphysics on the dynamics and providing observable signatures. Ideally, the former must be done as part of the hydrodynamic simulation but this comes with a prohibitive computational cost that leads to many simplifications being used in practice.

Aims. We aim to produce a statistical emulator that replicates a full chemical model capable of solving the temperature and abundances of a gas through time. This emulator should suffer only a minor loss of accuracy when compared to a full chemical solver and would have a fraction of the computational cost allowing it to be included in a dynamical model.

Methods. The gas-grain chemical code UCLCHEM was updated to include heating and cooling processes, and a large dataset of model outputs from possible starting conditions was produced. A neural network was then trained to map directly from inputs to outputs.

Results. Chemulator replicates the outputs of UCLCHEM with an overall mean squared error (MSE) of 1.7×10^{-4} for a single time step of 1000 yr, and it is shown to be stable over 1000 iterations with an MSE of 3×10^{-3} on the log-scaled temperature after one time step and 6×10^{-3} after 1000 time steps. Chemulator was found to be approximately 50 000 times faster than the time-dependent model it emulates but can introduce a significant error to some models.

Key words. astrochemistry – methods: numerical – methods: statistical – hydrodynamics

1. Introduction

Hydrodynamic simulations are a cornerstone of theoretical astrophysics. However, pure hydrodynamics is often insufficient, for example in scenarios where radiation, magnetic fields, or the chemical evolution plays an important role in setting the dynamics. Furthermore, the chemical composition of an astrophysical system is important for understanding observables.

Given this, there has been a substantial, sustained effort to develop numerical methods that are capable of dealing with the combination of required spatial resolution, time steps, and microphysics. However, even with a pure hydrodynamics calculation there are many scenarios where the required resolution for convergence alone makes simulations prohibitively expensive (e.g., the now-resolved problem of convergence in the behaviour of self-gravitating discs; see Meru & Bate 2011, 2012; Young & Clarke 2015; Deng et al. 2017). When additional microphysics such as radiative transfer or chemistry are included, which are typically orders of magnitude more computationally expensive than hydrodynamics per time step (and may also reduce the possible time step that can be taken), we find that most problems cannot be feasibly addressed.

Nevertheless, approximations have been made that have permitted the treatment of radiation, chemistry, and magnetic effects in hydrodynamic applications on scales of galaxies (e.g., Ilee et al. 2009; Few et al. 2012; Pillepich et al. 2018) through interstellar medium (ISM) evolution, star formation

(e.g., Glover & Clark 2012; Ali et al. 2018; Ali 2021), protoplanetary disc evolution, and planet formation (e.g., Ilee et al. 2017; Haworth & Clarke 2019; Krijt et al. 2020), right down to exoplanet atmospheres (e.g., Venot et al. 2012; Drummond et al. 2018). However, by necessity these approximations sacrifice accuracy to make the problem computationally feasible. Typical approximations regarding the chemistry are to assume equilibrium (which could be inaccurate both for the temperature and composition) and to use severely reduced networks with what are thought to be the key components for any given problem. However, it is difficult to know a priori to what extent these approximations are reasonable for any given application. Furthermore, there may even be qualitatively new processes that are overlooked when using simplified approaches to the microphysics (e.g., this was discussed and reviewed in the context of protoplanetary discs by Haworth et al. 2016).

One alternative to addressing the large computational cost is to employ an efficiency boosting tactic. For example, on-the-fly calculations can be used to determine reactions or species in a chemical network that are unimportant in any given timestep, allowing them to be left out of the calculation (Grassi et al. 2012). This can make models considerably faster when the overhead from the ‘importance calculations’ is outweighed by the acceleration of the chemical solver. Alternatively, large grids of models can be calculated in advance, and a look-up table can be interpolated over in dynamical models (e.g., Ploetckinger & Schaye 2020). Another innovation has been to

combine multiple reactions and particles into smaller sets of meta-reactions and meta-particles (e.g., Nelson & Langer 1999). However, even with these innovations there are still many problems whose solutions remain out of reach.

Despite the above, it may be possible to obtain a huge decrease in computational time whilst maintaining high accuracy through an emulator. An emulator in this context is a statistical model that returns the same outputs as a numerical model for the same inputs but is much faster to evaluate. These have been used to speed up parameter inference from radiative transfer and chemical models (de Mijolla et al. 2019) as well as to solve the thermochemistry of a small reaction network (Grassi et al. 2011).

In this work, the code Chemulator¹ is developed. The objective is to develop an emulator for solving the chemistry and gas temperature with sufficient accuracy and speed to render a host of new problems solvable. The emulator should take the state of the gas at a time t as an input and return the state of the gas (composition, temperature) at a time Δt later. This will result in a model with approximately the accuracy of a full chemical solver but with the much lower computational cost of evaluating a neural network.

To achieve this, a zero-dimensional (0D) model that includes a detailed treatment of the relevant chemical and thermal processes was produced and is described in Sect. 2. This model was then used to train the network, with the training process aided by a dimensionality reduction described in Sect. 3. The model training and basic performance metrics are presented in Sect. 4, and the work is summarized in Sect. 5.

2. The model

2.1. Model requirements

The model should take the physical parameters and chemical abundances of a gas at time t and return the temperature and abundances at a time $t + \Delta t$. It is intended to be employed as a sub-grid model in two or three dimensional dynamical simulations. This results in a crucial requirement on the list of the model specifications due to two competing constraints.

Firstly, in order for the emulator to work, the model cannot rely on information about the object beyond the initial conditions of a given time step. For example, radiative transfer solvers typically iteratively solve the optical depths along ‘rays’ from a given position. However, the emulator will not have access to internal values of the model such as those optical depths and so the model cannot rely on this method.

However, in order for the emulator to be effective at its intended task, non-local effects such as line cooling should approximate their effect in multi-dimension models. Therefore, a set of inputs must be found that allow the 0D treatment to recover the temperatures of a non-local treatment. In the following sections, the solution to this is discussed and the model is benchmarked against a 1D model.

2.2. Model description

The time-dependent gas-grain chemical code UCLCHEM (Holdship et al. 2017) was adapted to serve as the model to be emulated. It has been modified to include heating processes from UCL-PDR (Bell et al. 2005; Priestley et al. 2017), molecular line cooling (Bell et al. 2005; de Jong et al. 1980) and

Table 1. Cooling processes in the model.

Cooling process	Source
KROME	
H, He, He ⁺ collisional ionization	Cen (1992)
H ⁺ , He ⁺ , He ²⁺ recombination	Cen (1992)
He dielectric recombination	Cen (1992)
H collisional excitation	Cen (1992)
He collisional excitation	Cen (1992)
He+ collisional excitation	Cen (1992)
H2 collisional dissociation	Martin et al. (1998) Glover & Mac Low (2007)
Bremsstrahlung all ions	Cen (1992)
Compton cooling	Cen (1992)
Continuum	Hirano & Yoshida (2013)
Collisionally Induced Emission	Hirano & Yoshida (2013)
UCL_PDR	
LVG line cooling	de Jong et al. (1980)
H2 single pseudo level vibrational cooling	Röllig et al. (2006)
Dust cooling (and heating)	Hollenbach & McKee (1979)
Important endothermic reactions	Bell et al. (2005)

Notes. The processes are listed with their original source and grouped by the source for the treatment used in the model.

Table 2. Heating processes in the model with original reference.

Heating process	Source
Photoelectric heating	Weingartner & Draine (2001)
H2 formation	Hollenbach & Tielens (1999)
H2 photodissociation	Hollenbach & McKee (1979)
H2 FUV pumping	Hollenbach & McKee (1979)
C ionization	Kamp & van Zadelhoff (2001)
CR heating	Goldsmith (2001)
Important exothermic reactions	Bell et al. (2005)
Gas-grain collisions	Burke & Hollenbach (1983)
Turbulent decay	Black (1987)

Notes. The code and processes were taken from UCL_PDR (Bell et al. 2005).

other cooling processes from KROME (Grassi et al. 2014). A list of implemented heating and cooling processes are given in Tables 1 and 2.

The model consistently solves the time-dependent gas chemistry and temperature for a single position in a cloud. The required inputs are the initial chemical abundances and the physical properties of the cloud at that position as listed in Table 3. The outputs are the gas temperature and chemical abundances at a later time.

The requirement that the model solves a single position independently of other positions has been satisfied using the column density inputs. There are broadly two areas where a non-local model would use information from the wider cloud. For photoionization reactions, the total column density gives the level of visual extinction and therefore the local UV field. Further, calculating the column density of a specific species to the cloud edge corrects their rates for effects such as self-shielding. For this time-dependent model, the H₂ self-shielding and C photoionization rate can simply be calculated using the column density of C and H₂ at the start of the time step. Interestingly, despite the fact that UCLCHEM treats CO self-shielding, the CO column density was not included as an input. In initial testing, it was found that calculating the CO column density by simply using the CO

¹ <https://github.com/uclchem/Chemulator>

Table 3. Model inputs excluding the species abundances.

Parameter	Symbol	Range
Gas density	n_H	$10\text{--}10^6 \text{cm}^{-3}$
Gas temperature	T_g	$10\text{--}10^4 \text{K}$
Dust temperature	T_d	$10\text{--}10^3 \text{K}$
Radiation field	F_{UV}	$10^{-1}\text{--}10^5 \text{Habing}$
Cosmic ray ionization rate	ζ	$10^{-2}\text{--}10^5 \zeta_0$
Total column density	N	$10^{15}\text{--}10^{24} \text{cm}^{-2}$
H ₂ column density	N_{H_2}	$10^5\text{--}5 \times 10^{23} \text{cm}^{-2}$
C column density	N_C	$1.0\text{--}1.5 \times 10^{20}$
Metallicity	Z	$10^{-2}\text{--}10^{0.5}$

Notes. ζ_0 indicates a standard cosmic ray ionization rate of $1.3 \times 10^{-17} \text{s}^{-1}$. It should be noted that throughout the text the UV field is discussed in units of the Draine field to facilitate comparison with other codes. However, Chemulator uses Habing ($1 \text{ Draine} = 1.7 \text{ Habing}$).

abundance of the current position and the input column density allowed the model to pass the benchmark tests over a wide range of parameters and visual extinctions.

The other non-local effect is the calculation of the line opacities of all coolant species. For this, the on-the-spot approximation (Dyson & Williams 1997) was used in which the line opacities are calculated for a homogeneous cloud with the gas properties of the current position and a size that gives the same total column density as the input value. Whilst this does not a priori guarantee correct cooling rates, the model is benchmarked extensively in the next section.

Overall, this approach means the user needs to provide only three variables to allow the model to capture non-local effects on the temperature and chemistry. An alternative approach would be to make UCLCHEM use local variables only and shift the computation of the local UV field, the rates of photo-processes and the cooling rates to the user. It is possible this would improve the emulator by essentially pre-encoding the relationship between various inputs but would represent a much larger computational burden to the user.

The chemical network contains 33 gas phase species, interacting through 330 reactions. The gas phase network is taken from the UMIST12 database (McElroy et al. 2013), including reactions between the network species, cosmic rays and UV photons. X-rays are not considered by this model. In principle, using an emulator could allow complex chemistry to be solved without computational overhead but in an early prototype of this emulator, no working emulator could be produced from a model using a network of 215 species (Appendix A). Moreover, the gas temperature for any given set of input parameters was largely unaffected by the choice of network and so, for simplicity, this smaller network was used (see Sect. 2.4).

The temperature is solved by including it in the ordinary differential equation (ODE) system solved for the chemistry. The rate of change of temperature is calculated as

$$\frac{dT}{dt} = (\gamma - 1) \frac{\Gamma - \Lambda}{k_B n_H}, \quad (1)$$

where γ is the adiabatic constant of the gas calculated from the number densities of the most abundant gas phase species using Eq. (8) from Grassi et al. (2014). The Γ and Λ are the total heating and cooling rates in $\text{erg cm}^{-3} \text{s}^{-1}$, respectively, and n_H is the total gas number density, approximated as the total number density of hydrogen nuclei.

2.3. Benchmarking the modified UCLCHEM code

Given the simplified nature of the radiation treatment in this model, the modified UCLCHEM code was benchmarked against UCL_PDR (Bell et al. 2005; Priestley et al. 2017). UCL_PDR is a 1D photodissociation region (PDR) code that uses ray tracing to consistently solve the line opacities for all positions in the model together and to calculate column densities for the treatment of photoionization reactions. Therefore, it is an ideal test of the 0D approximation of the modified UCLCHEM. Moreover, it was itself benchmarked against many other PDR codes (Röllig et al. 2007). For this benchmarking, all cooling rates other than molecular line cooling were also turned off so that both codes shared heating and cooling mechanisms. As a result, any differences can be attributed to either the single point treatment or time-dependent chemistry of UCLCHEM.

The two codes were compared using the four benchmark models of Röllig et al. (2007) as well as models with a low radiation field, low metallicity and high cosmic ray ionization rate. Figure 1 shows the results of the (Röllig et al. 2007) benchmarks, the low radiation field and high cosmic ray ionization rate models. In every model, the temperatures between the two codes were in close agreement.

Each of the Röllig et al. (2007) benchmarks modelled a cloud of constant density. In order to investigate whether this homogeneity was important for UCLCHEM to achieve accurate results in a 0D model, further models were run for clouds with more complex density profiles. In particular, Fig. 2 shows the temperature and abundances of several species as a function of A_v for a cloud where the density at any position is a sine wave function of the distance to the cloud edge. This unphysical scenario would reveal any problems that UCLCHEM suffers when dealing with inhomogeneous clouds. However, it is clear from the figure that the agreement between the two codes is very good.

2.4. Considerations for the full model

A major conclusion of Röllig et al. (2007) was that a large degree of fine tuning is required to make PDR codes agree. Therefore, it is important to consider the effect of including additional cooling mechanisms and more complex chemistry in the model after the benchmarking tests were completed.

In particular, it should be noted that the initial elemental abundances have a large effect on the gas temperature. In the worst case, in the high density benchmark model with a radiation field of 10^5 Draine , the gas temperature using solar abundances (Asplund et al. 2009) reaches 7750 K rather than the 1380 K obtained using the Röllig et al. (2007) benchmarking abundances.

However, holding the abundances constant, the benchmarks were performed including the additional cooling mechanisms in Table 1 and a larger network of ~ 250 species. This gave a maximum temperature difference of 5% compared to the benchmarking models. This justifies the use of a smaller network for the emulator.

Finally, it is worth noting the benefits of the single point model over the 1D PDR codes that are available. UCL_PDR, like many PDR codes, assumes the chemistry and temperature reach equilibrium. In Fig. 3, the gas temperature as a fraction of the equilibrium temperature is given for different times for a sample of 20 000 random UCLCHEM models (see Sect. 3.1 for the models used). Equilibrium is often reached quickly with 80% of models reaching equilibrium by 0.1 Myr. However, 5% of models do not reach equilibrium after 0.5 Myr. Thus for a

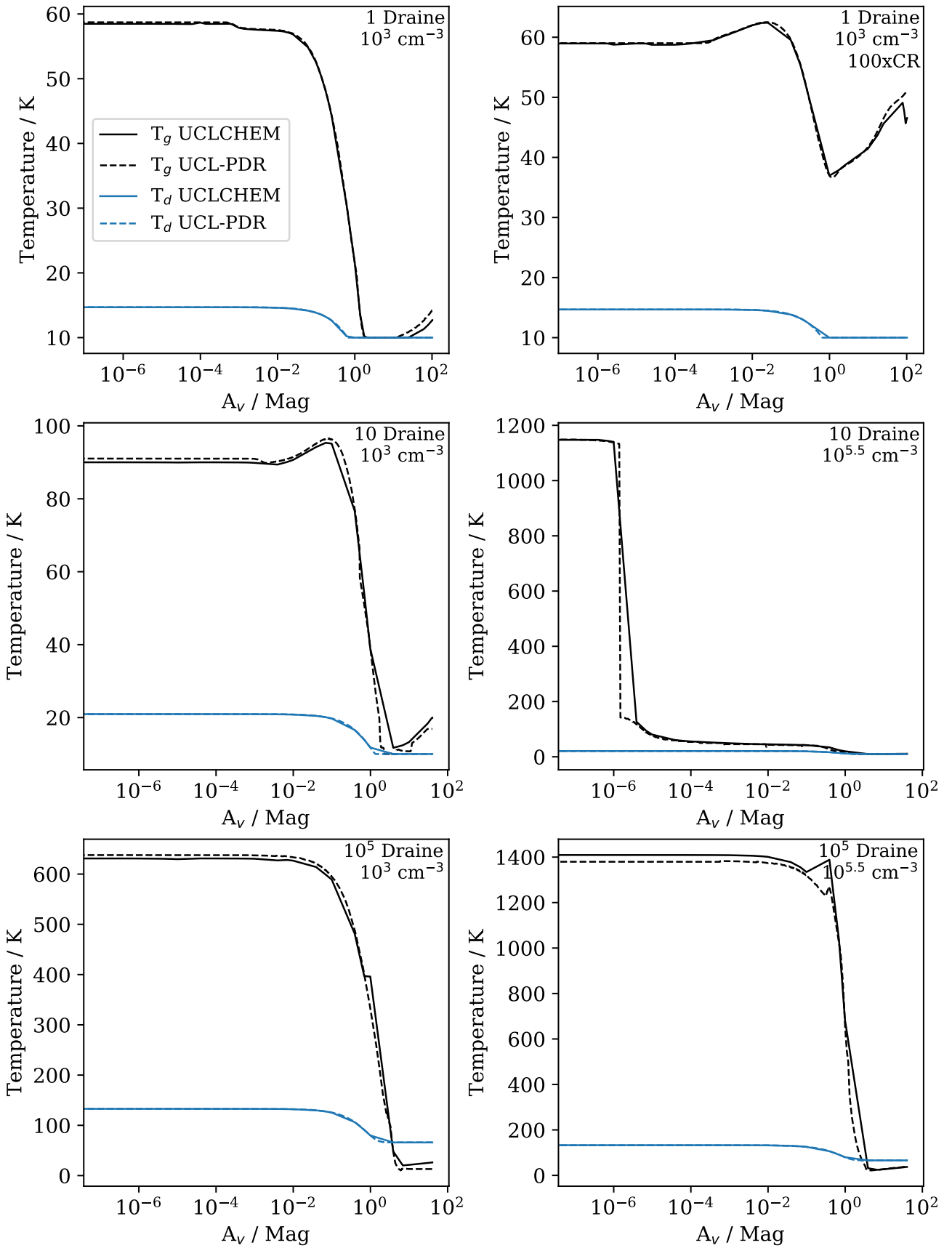


Fig. 1. Temperature as a function of visual extinction. The *lower four plots* show Röllig et al. (2007) benchmarking models. In each subplot, the UV field in Draine is noted and, in the top right, $100 \times \text{CR}$ indicates that the cosmic ray ionization rate is 100 times the standard. The models agree extremely well, allowing for the finer A_v sampling of UCL_PDR.

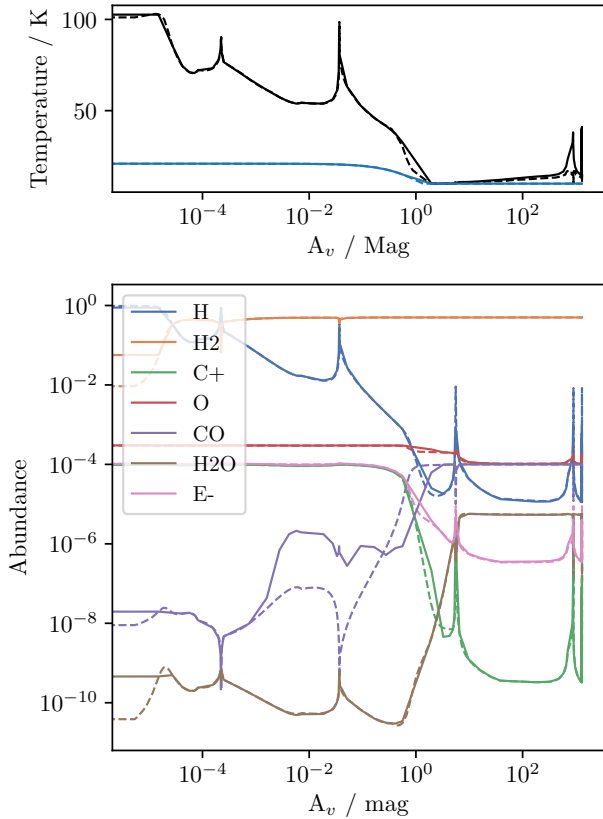


Fig. 2. Comparison between the UCLCHEM and UCL_PDR results for a cloud of sinusoidally varying density. *Upper:* gas (black) and dust (blue) temperature against A_v . *Lower:* fractional abundance against A_v . In both plots, results from UCLCHEM are plotted as solid lines and equivalent values from UCL_PDR are plotted with dashed lines in the same colour.

sub-grid model with short time steps, the model presented here offers a significant improvement over equilibrium models for many cases. Further, given that the heating and cooling processes have been shown to be implemented with adequate accuracy by the benchmarks, the differences in temperature obtained with the full network must in fact be the result of improved chemistry.

3. Dimensionality reduction

The inputs for the full model are the 33 initial chemical abundances and the physical variables listed in Table 3. Considering all inputs, there are 41 variables that would need to be sufficiently well sampled to provide adequate training data for the emulator, which poses a considerable challenge. Further, an emulator that could take all 41 variables and return them at a time Δt later would be required and a regressor with so many outputs is unlikely to be uniformly accurate.

However, the chemical abundance variables cannot be independent because the chemical network is a closed system of ODEs. Intuitively, the more atoms there are, the fewer molecules there must be and the more ions in the gas phase, the more free electrons there must be. Therefore, it should be possible to transform the abundances to a new, lower dimensional set of variables with minimal loss of accuracy.

Using such a dimensionality reduction procedure, the operation of Chemulator would be to first encode the input chemical abundances into the lower dimensional space, then emulate the advancement of the (encoded) chemistry and temperature, before

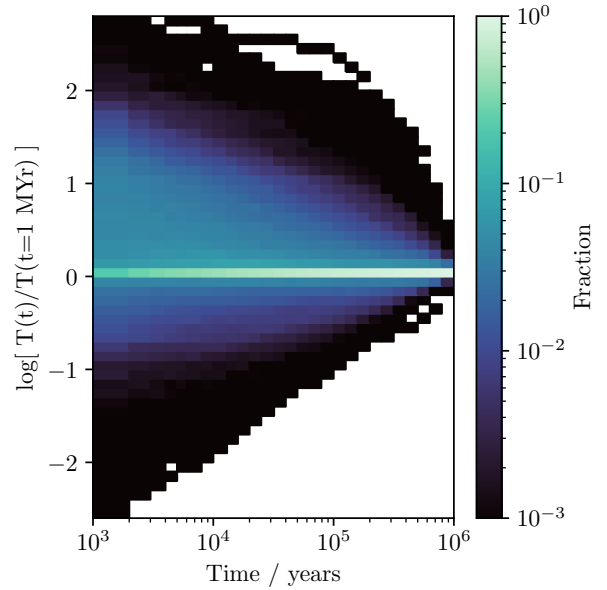


Fig. 3. Histogram where the colour shows the fraction of models at a given time that have a given log ratio of current temperature to equilibrium temperature. After as few as 1000 years, the majority of models have reached their equilibrium temperature. However, even as the time approaches 1 Myr, $\sim 1\%$ of the models have not reached equilibrium, which demonstrates that assuming equilibrium will often lead to incorrect chemical abundances.

decoding the abundances. This is illustrated in Fig. 4 and the following section details the creation of the encoder and decoder.

3.1. Dimensionality dataset

To do this, a dataset of abundances that gave a good representation of all possible abundances in the model was created. Latin hypercube sampling (LHS) (McKay et al. 1979) was used to efficiently sample the physical parameters in Table 3 in log-space within the given ranges to produce initial conditions for 10 000 models. An exception to this sampling procedure had to be made for the column densities. Whilst the total column density was sampled with the other parameters, the H₂ and C column densities are dependent on the total column density and so could not be sampled independently. To obtain sensible column densities for those molecules, the benchmarking models were examined to find possible ranges for any given total column density. These possible ranges were then randomly sampled to get a H₂ and C column density compatible with the total column density for each model.

Those models were run for 1 Myr starting with elemental gas and abundances were written out every 1000 yr. Following that, another 10 000 samples from the physical parameter space were taken and models were run for 1 Myr starting from the final abundances of a model from the first set. This introduced different chemical histories to the dataset and varied the physical parameters. The resulting dataset contained 2×10^7 sets of abundances from various stages of chemical evolution in physical conditions that span the range of those covered by the emulator.

This dataset was then used to find a transformation that could transform the abundances to a lower dimensional set of variables and then recover them with minimal losses. Ultimately, an autoencoder was chosen for this purpose. An autoencoder is a neural network that returns the input as an output. By creating

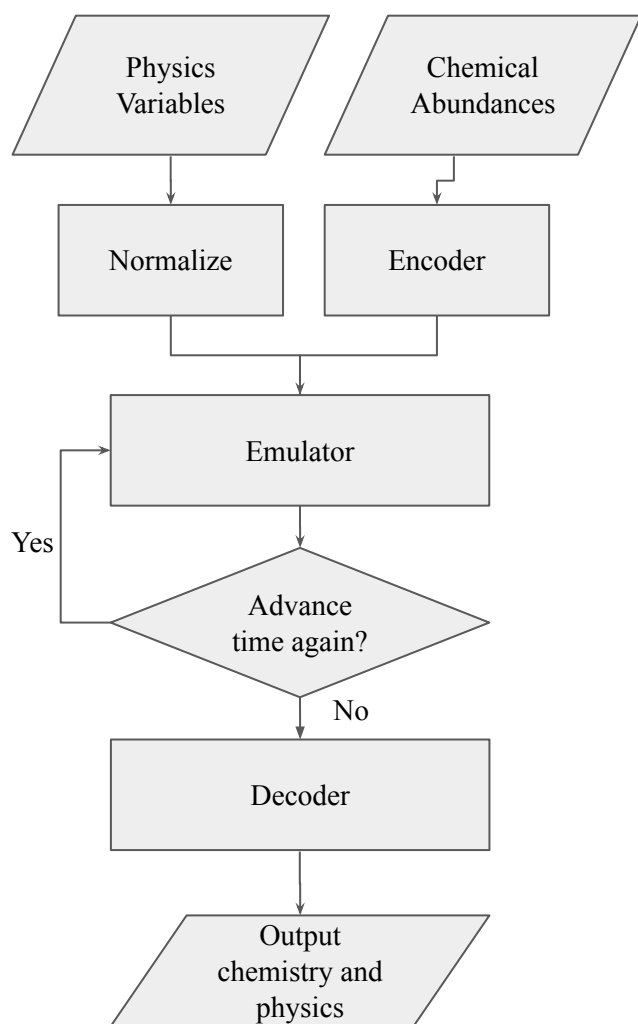


Fig. 4. Flowchart showing the operations performed by Chemulator. In particular, it shows how the autoencoder described in Sect. 3 combines with the emulated version of the chemical model to produce the final output.

an autoencoder where one of the hidden layers is very small, the abundances can be compressed.

To train the network, the dataset was transformed in the following way: (i) all abundances less than 10^{-20} were clipped to 10^{-20} ; (ii) the log of the abundance was taken; (iii) log abundances were scaled to be between 0 and 1; and (iv) if any set of scaled log abundances was identical to another to at least three decimal places, one set was removed to ensure there were no duplicate sets of abundances in the data.

The log of the abundances was used so that no preference was given to the high abundance species. Without this, a 10% error on a species like H_2 would contribute $\sim 10^4$ times as much error to the network as even an abundant species like CO.

The abundances were clipped to 10^{-20} as this greatly improved the autoencoder accuracy over UCLCHEM’s internal clipping at 10^{-30} . This negligible error of at most 10^{-20} on the species abundance was considered an acceptable given that observed abundances in the ISM are typically greater than 10^{-12} .

The removal of duplicates improved the efficiency of training the neural network as approximately 40% of the training data was removed with no loss of information. It also improves the network by preventing it from favouring accurate recovery of

Table 4. Description of autoencoder layers and configuration.

Layer	Nodes	Type
0	33	Input
1	256	Dense – Swish
2	8	Dense – Swish
3	256	Dense – Swish
4	33	Dense – Sigmoid

Notes. Layers 0–2 comprise the encoder and layers 2–4 are the decoder.

a small set of commonly repeated abundances. Finally, scaling variables is a common approach to training neural networks. It also has the advantage that the rms error becomes intuitive as it is the average fractional error on the outputs.

In addition to this pre-processing, two aspects of UCLCHEM could be used to improve the emulator. Firstly, the e^- abundance is always the sum of all ion abundances. Thus, this abundance does not need to be encoded and can be simply recovered from the ions. Secondly, the total H in the model must equal one, this is enforced in the emulator by removing H^+ when the total is larger than one and then H if the abundance is still too large. Other species have total abundances that can vary based on metallicity and so no conservation is applied to them.

3.2. Selecting an autoencoder

The abundance dataset was split 70:30 into training and validation data and a large grid of neural network configurations with different sizes and numbers of layers and different activation functions was run. Each autoencoder required approximately one minute per epoch to train and 30–50 epochs to reach a minimum validation error. The neural networks were trained by minimizing the mean squared error (MSE) on the recovered abundances. This is the mean squared difference between the input abundances and the predicted abundances. MSE values between 10^{-6} and 10^{-5} could be obtained with many neural networks with encoded sizes of as few as four variables.

However, neural networks can show strongly non-linear behaviour, particularly when multiple layers are combined. This would mean small changes to the encoded values could result in large changes to output abundances. Since the emulator will work with these encoded variables, simple behaviour was preferred because the emulator would introduce small errors to the encoded values, which should ideally correspond to small errors in the abundances. Therefore, whilst the best autoencoders had two layers in the encoding and decoding parts of the network, the final network chosen was one with only a single layer in the encoder and decoder. The model described in Table 4 has an MSE of 8×10^{-6} on the test data, which corresponds to an RMS error on the predicted log abundances of 0.3%.

This accuracy is demonstrated visually in Fig. 5. Most species show an extremely tight relationship between the auto-encoded and original abundances. The autoencoder’s performance is worst when the abundance of a species is much lower than is typical. This can be seen in the spikes in the H_2 predictions at $\sim 10^{-10}$. The figure also shows that at low abundances ($< 10^{-5}$), the H abundance is systematically under-predicted, likely as a result of the enforcement of H conservation described in Sect. 3.1. Nevertheless, the performance of this autoencoder was considered to be sufficient and therefore it was used for the emulator.

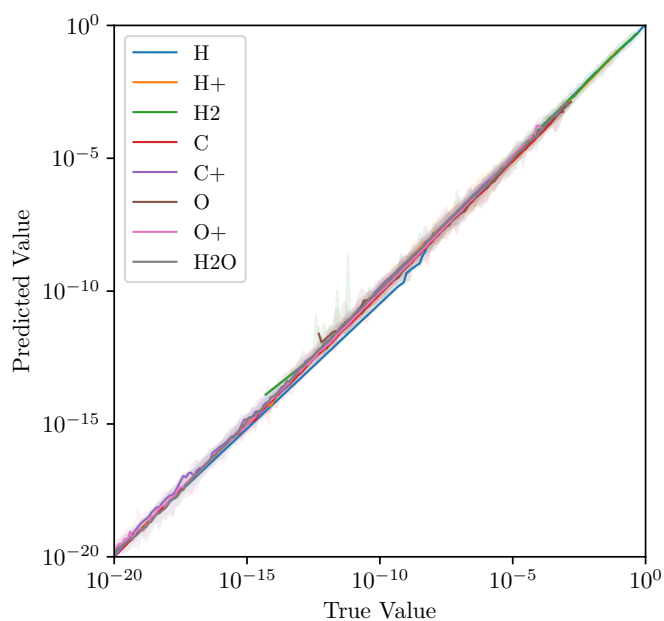


Fig. 5. Predicted abundance after a pass through the autoencoder as a function of true abundance for several important species. The solid lines show the median predicted value in the test set for a given real value, and the shaded area shows the predicted range for 95% of encoded abundance sets.

3.3. Drawbacks of the autoencoder approach

Despite using the simplest network with an acceptable accuracy, the drawback of using an autoencoder was that, upon experimentation, it was found that the chemical abundances could not be sampled in encoded space. The complexity of this dimensionality reduction method is such that any set of abundances can be accurately encoded and decoded but not all possible combinations of encoded variables will produce realistic or even reasonable abundances.

When testing this, the entirety of the dimensionality reduction data was encoded and the range of values each encoded variable took were used as limits on those variable. LHS sampling between those limits was then performed to generate a well sampled set of encoded abundances. However, upon decoding, it was found that many abundance sets were infeasible such as total carbon abundances ten times that of hydrogen meaning that this sampling procedure could not be used to generate well sampled abundances.

Given that one of the two motivators for the dimensionality reduction was improved sampling, a simpler dimensionality reduction method was tried. A principal component analysis (PCA) of the training data was conducted to investigate whether a simple linear transformation of the abundances would suffice. However, the PCA required 12 variables to describe 99% of the variance in the abundances and this gave an rms error on the log abundances of 1.4×10^{-2} . Encoding the chemistry in PCA components required too many components to be useful in order to achieve an acceptable accuracy.

In light of this, it was not possible to evenly sample physical and chemical parameter space. Therefore, the dimensionality dataset was used to train the emulator. The chemical and physical parameters for each of the 10 000 models in that dataset were written to file every 1000 years from 0 to 1 Myr inclusive. This means there were 1000 pairs of initial parameters with the corresponding values 1000 years later in each model. Thus 20

millions pairs of inputs and outputs were available to train the emulator.

Whilst the autoencoder was not used for sampling, it was still used to reduce the input parameters for the emulator. It is likely that an improved dimensionality reduction would represent a major improvement to this work and may even allow a much larger chemical network to be emulated. This will be the focus of later study.

4. Creating an emulator

The model presented so far is a complex and accurate time-dependent model for the thermodynamics and chemistry of a parcel of gas. However, unlike models such as PRIZMO (Grassi et al. 2020), which have been optimized for direct use in hydrodynamic models, it would be inefficient to run as a sub-grid model. The main aim of this work is to produce an emulator for the model that is similarly accurate but has a far lower computational cost than any equivalent model.

4.1. Training the emulator

As noted in Sect. 3.3, the dimensionality dataset was used to train the emulator. In fact, the possibility that this would be a secondary use of the dimensionality dataset is the reason a fixed timestep was used in the generation of that data. By matching each timestep to the one that follows, an emulator can be trained that always advances the temperature and chemistry by 1000 years. Larger time steps can be obtained by repeatedly running the emulator and an emulator working with small, fixed time steps is easier to develop than one that can produce variable time advancements.

All chemical abundances were encoded by the autoencoder to give eight chemical variables. The physical parameters were then log-scaled, and then all inputs were min-max scaled to take the range 0–1. To avoid bias, the data were rounded to n decimal places and duplicate rows were removed before returning to the original values. Rounding to two or three decimal places was found to greatly improve the model performance over higher values of n . Finally, this dataset was split 70:30 to train and test the emulator.

The emulator is a neural network that takes the 16 input variables and returns the gas temperature and the eight encoded abundance variables. The number of hidden layers, the size and the activation function for each layer was chosen by training a large grid of models, minimizing the MSE on the outputs.

It was found that MSE values of 10^{-5} – 10^{-4} could be obtained with networks with 2–4 hidden layers of at least 128 nodes each using rectified linear unit (ReLU) or Swish activation functions (Ramachandran et al. 2018). The performance of a typical emulator on a single time step is shown in Fig. 6. We note that despite the fact that it is an analytical function of the emulator inputs, the dust temperature has been included as a target for the emulator for ease of implementation.

A selection of species abundances is also shown in Fig. 6. Since neither the emulator nor the autoencoder are trained to conserve mass, oscillations seen in these abundances do not affect the abundances of other species. The exception is e^- , which has an abundance equal to the total ionization fraction of the gas in UCLCHEM and therefore is calculated by summing the total ion abundances in the emulator. By capturing this abundance well, the emulator is recovering that fraction.

However, a known flaw of these kinds of emulators is that over many time steps, error accumulates and the predicted

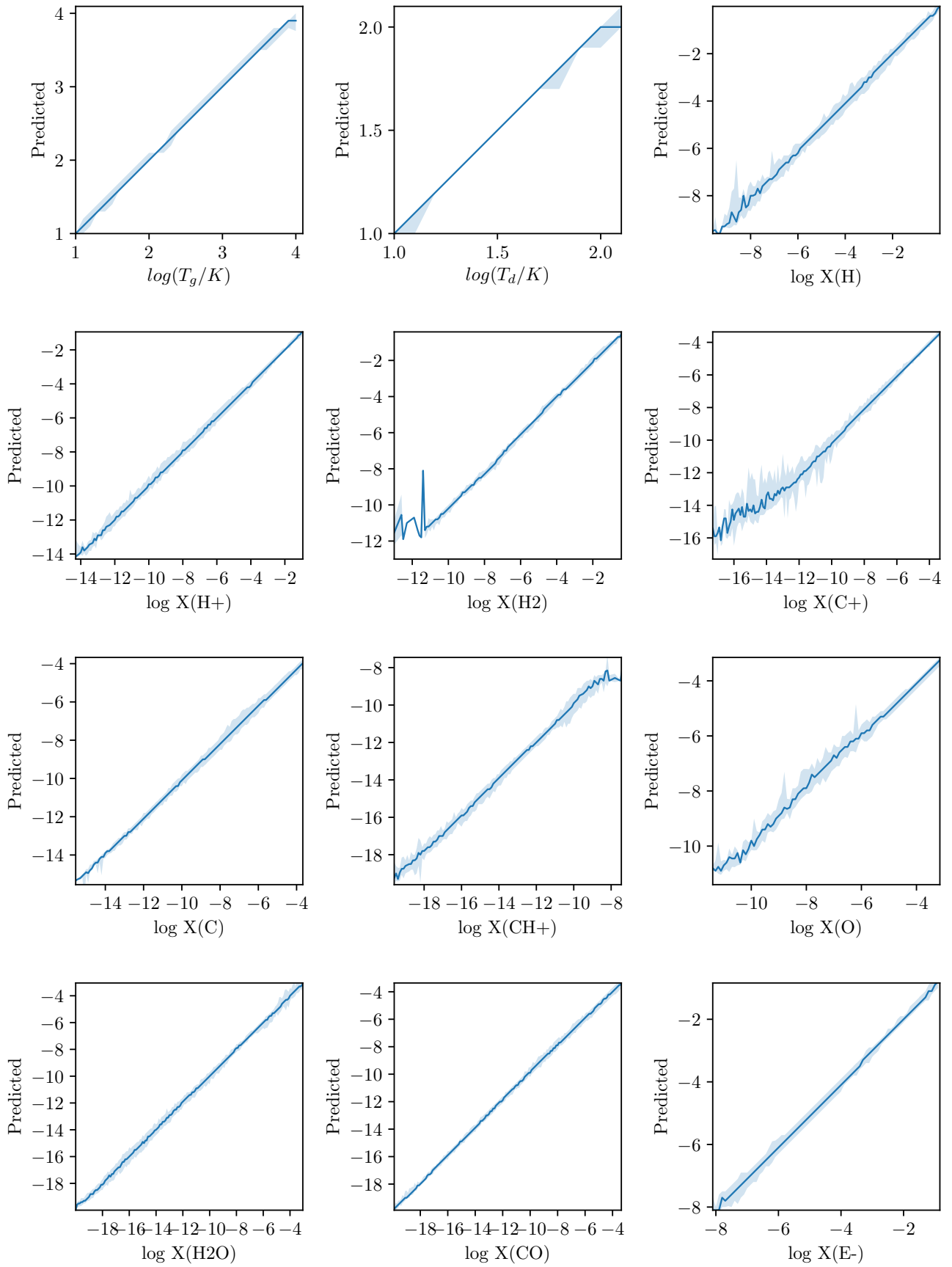


Fig. 6. Predicted against real model outputs for the emulator test set. The solid line shows the median value predicted for a given true value, and the shaded region shows the range of values predicted in 95% of cases.

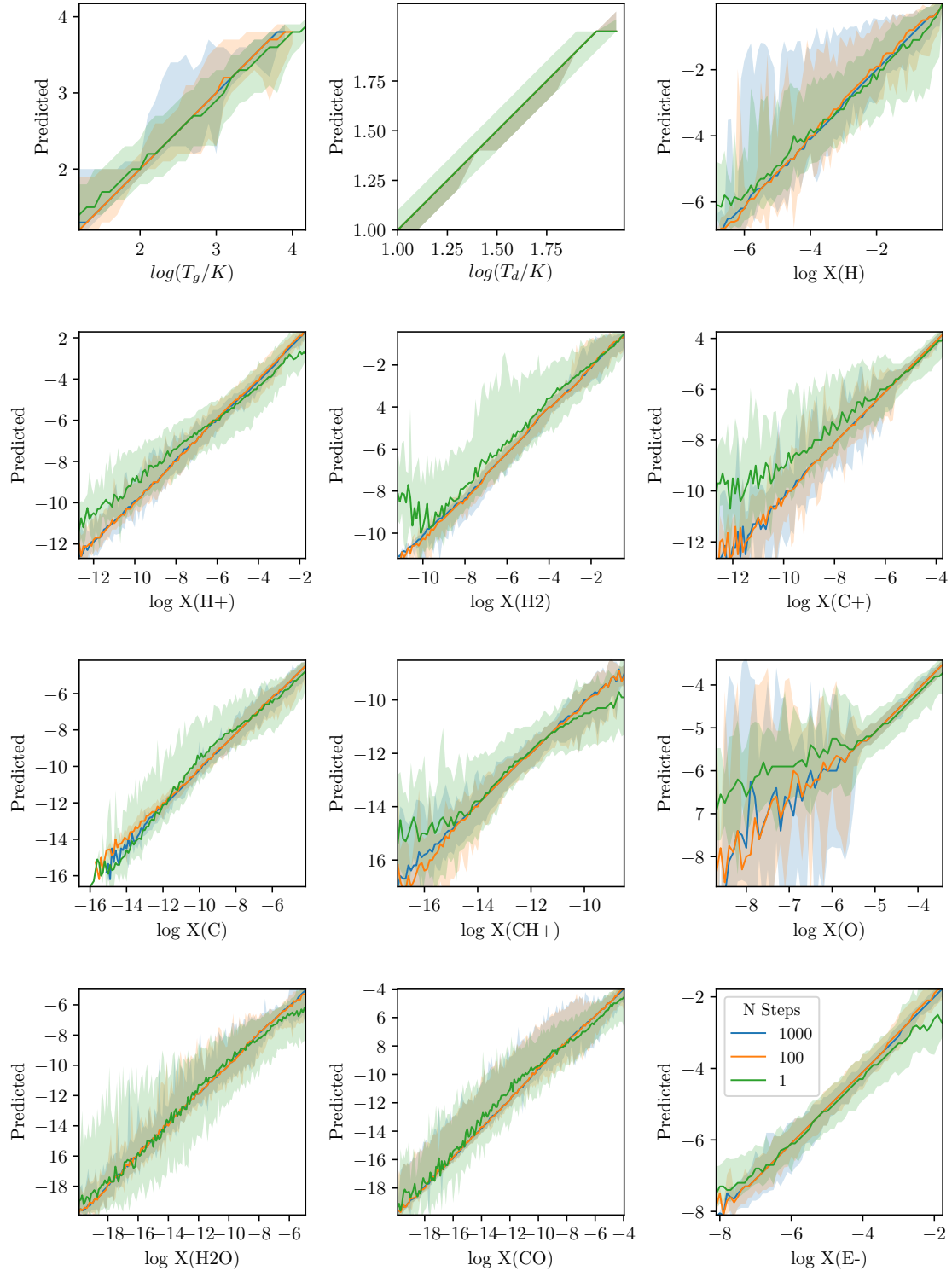


Fig. 7. Predicted against real values for time steps in the test set. The median is plotted as a line, and the shaded region shows the range predicted by 95% of the models. The colours represent the number of time steps emulated. For example, the blue lines show the predicted evolution of the chemistry for 1 Myr compared to the actual value after 1 Myr. We note that the dust temperature error is stable as it is a function of UV and column density, which do not change.

values after n iterations can be extremely far from the true value. This has been seen in similar models in the past (Grassi et al. 2011, priv. comm.). Since this emulator is intended to be used over many iterations of a hydrodynamical model, this would be unacceptable and it was found that all single emulators suffered from this issue.

4.2. Error mitigation

Once it was found that even a small single time step error produced unacceptable error growth, two methods were explored to make the emulator robust to errors that would be introduced by the neural network.

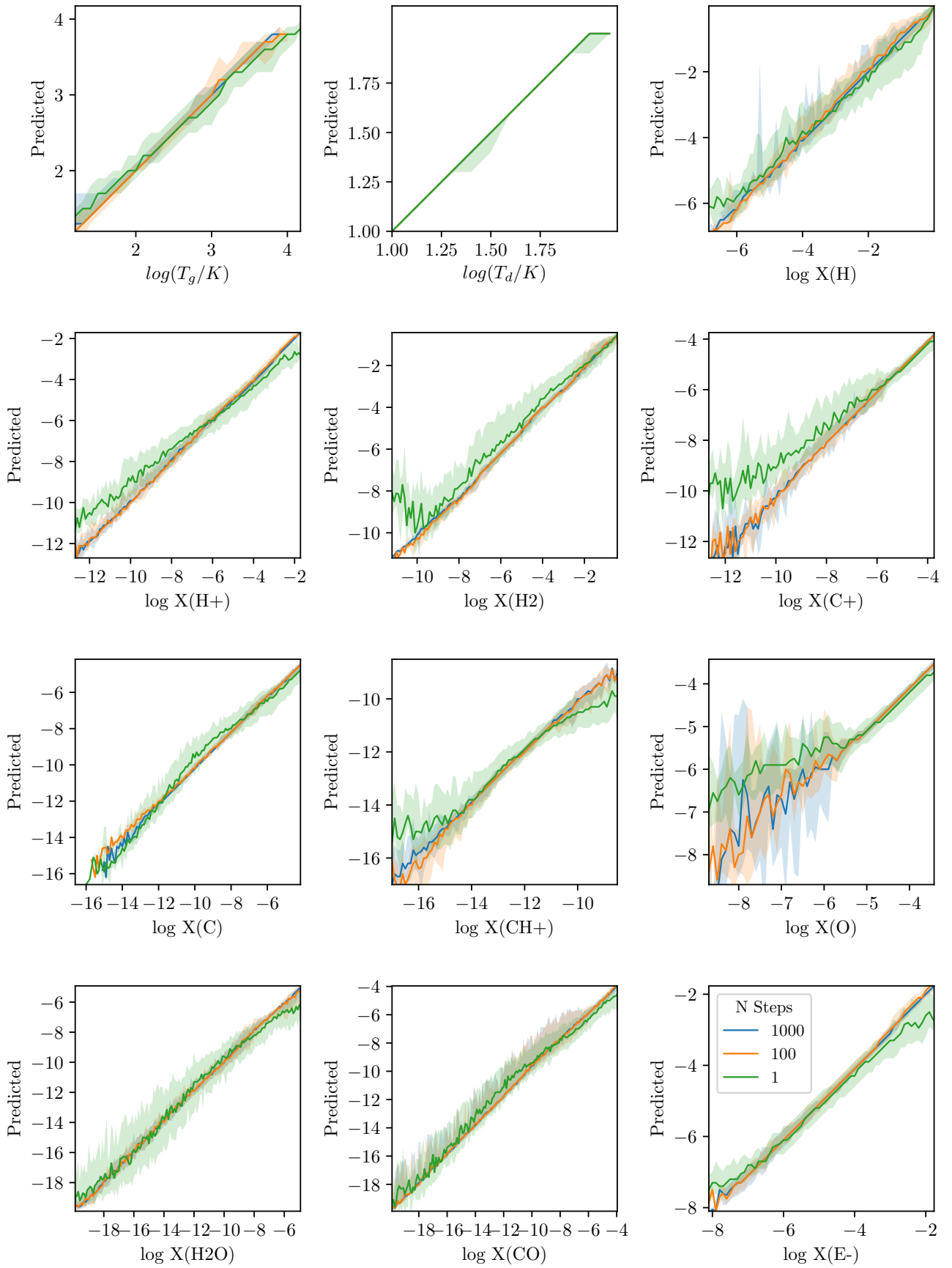


Fig. 8. Similar to Fig. 7 but the shaded region only covers the 67% of models closest to the correct value, rather than the 95%.

Firstly, a Gaussian noise layer was added to the network during both the autoencoder and emulator training. This is a neural network layer that is only active during training and introduces Gaussian noise with some standard deviation σ . This noise was added to the encoded variables in the autoencoder and to the input of the emulator. If a network can still accurately predict the outputs with this noise, it should be robust to errors of the same order as σ . Given the typical RMSE of the emulators is 0.01, values of sigma between 0.01 and 0.05 were trialed. Secondly, an ensemble model was employed. This is simply a model that contains N neural networks each trained individually on the data. They each predict the output and then the mean prediction from all N networks is the model output. In this way, if one network predicts an incorrect value, the others may counter it.

Increasing the noise was found to improve the model stability at the cost of increasing the single timestep error. Models trained using a σ of 0.05 were found to be most stable, giving a similar MSE after 1000 time steps as they give for a single time step. However, those trained with lower noise achieved better MSE error overall and often had a lower MSE error after 1000 time steps than the high noise models despite the fact that the error grows with time. It was also found that the ensemble models improved the prediction accuracy but only for ensemble sizes up to four, beyond which the MSE was unchanged.

The final ensemble model used four networks, each with two ReLU layers of 256 nodes per layer, which were trained separately before being combined using an averaging layer. It was trained with a noise σ of 0.02 to minimize error at the cost of some stability as it still performed better after 1000 time steps than a similar, more stable, network trained with a noise of 0.05. The ensemble performs just as well on a single time step as the single networks, with an overall MSE of 5.6×10^{-5} . This is equivalent to an RMS error of 0.7% on the output variables. Most importantly, the error does not continuously grow over many iterations as it did in the noise-free single models.

4.3. Model performance and benchmarking

The goal of this work is to produce a prototype of a ‘fast’ and ‘accurate’ emulator for thermochemical models. The former criterion is certainly met. The training data required 3750 CPU hours to produce. However, taking the values at $t = 0$ yr and emulating each of the 20 000 models to their final time of $t = 10^6$ yr required 5 min on a single NVIDIA GeForce GTX 1650 GPU.

To demonstrate how well Chemulator meets the second criterion, the performance of the emulator on the test data and a number of simple benchmarks are presented in this section. In Fig. 7 a plot similar to that in Fig. 6 is shown; the difference is that the predicted and real values are compared after 1, 100, and 1000 time steps to show the error growth. From the size of the shaded regions in the figure, it is clear the error does not grow continuously. One should note, however, that even after 1 time step, the error is larger than in Fig. 6. The model appears to perform worst on the first time step of each model in the database, possibly because the random conditions are not always consistent with the abundances. Quantitatively, the MSE on the log-scaled temperature is 6×10^{-3} after 1000 yr and 6×10^{-3} after 1 Myr.

However, this error is not uniform and a minority of models drastically affect the performance. Figure 8 is identical to Fig. 7 except it shows only the 67% of models closest to the true value.

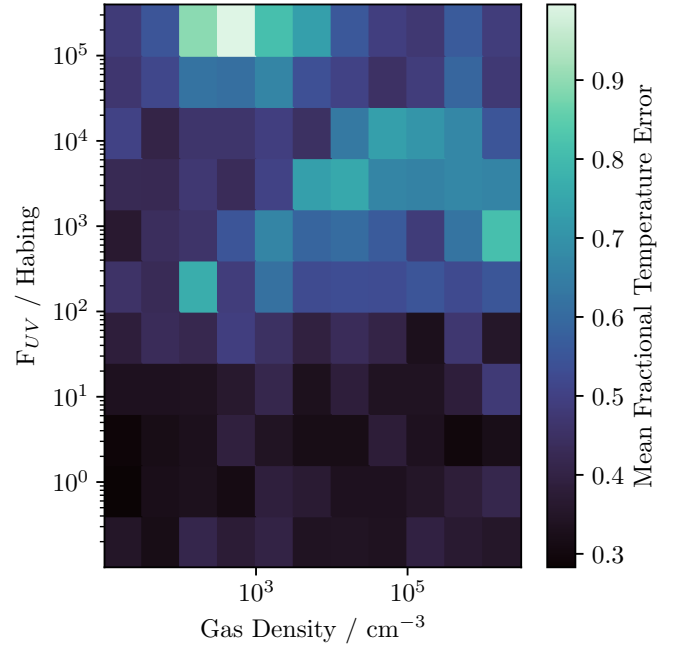


Fig. 9. Fractional error on the gas temperature after 1000 iterations (1 Myr) through the emulator as a function of gas density and external UV field.

The error distribution is much narrower, especially on the gas temperature, showing that in the majority of cases the emulator produces very accurate temperatures over long times. In order to determine the region of parameter space in which the emulator fails, the error as a function of input parameters was investigated. Figure 9 shows how the mean fractional error on the temperature varies as a function of radiation field and density after 1 Myr of evolution. The emulator gives better temperature estimates for lower UV, with a notable cutoff at a UV field of 100 Draine, above which the average error can be larger than 50%.

The second set of tests considered for the emulator are the Röllig et al. (2007) benchmark models from Sect. 2.3. Given that the original model passed this benchmarking, an emulator should too. This offers several advantages as a test. Firstly, it is a test many PDR models have passed and therefore any new PDR model should attempt to pass it. Second, the benchmarks are equilibrium models and so the stability of the emulator will be tested. Finally, every single model in the training set uses elemental abundances from Asplund et al. (2009), scaled by the metallicity. However, the benchmark models use elemental abundances that are not a single scaling of those elemental abundances and so present an interesting out of sample test.

Figure 10 shows the four benchmarking models from Röllig et al. (2007) plus two additional models as in Fig. 1. Comparing the temperature as a function of A_V between UCL_PDR and Chemulator, it appears the model struggles with low A_V values. At 10–20% error is typical in these regions and in the low density, high UV model this error is almost 50% in the low A_V region. This is likely due to the fact the emulator performs worse at higher UV, as seen in Fig. 9.

However, the model does perform well at higher A_V . The errors on the temperature are very small once the $A_V \gtrsim 1$ Mag. Moreover, these solutions are extremely stable and the same result is obtained after 2000 iterations as after the 1000 iterations (1 MYr) shown in these figures.

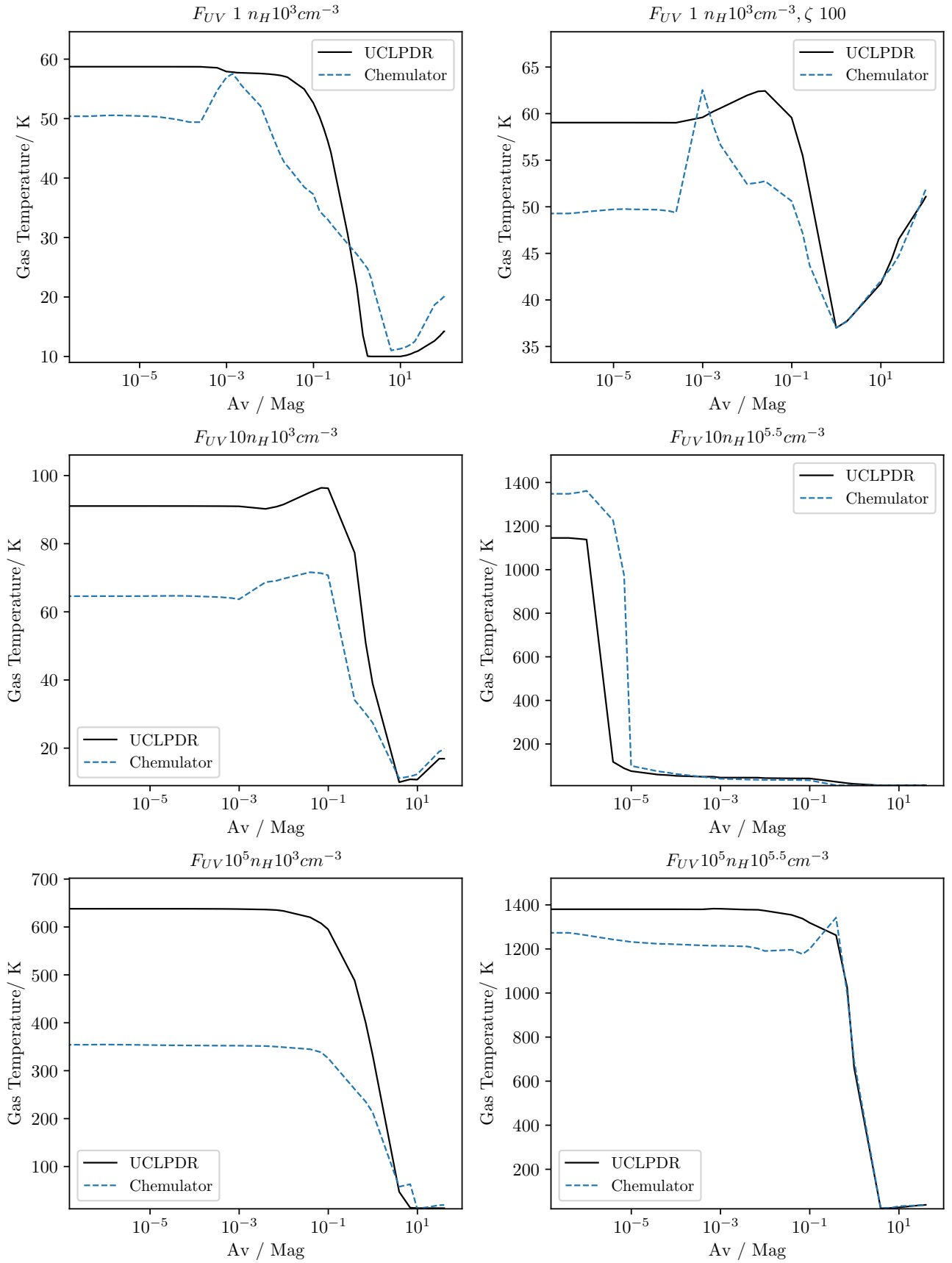


Fig. 10. Similar to Fig. 1 but comparing gas temperatures from UCL_PDR and Chemulator. The dust temperature is not included as it is an analytical function of column density and UV.

This is further demonstrated in Fig. 11, which shows the sinusoidally varying cloud used as a complex benchmark for UCLCHEM. Whilst the temperature is over-predicted at low A_v , Chemulator gives a reasonable estimate further into the cloud. Moreover, the fluctuations in the temperature that arise from the varying density are very well captured.

Figure 11 also shows the abundances from the sinusoidal model, and Fig. 12 presents the abundances from the benchmark models. Chemically, the emulator appears to give accurate estimates of the true abundances. The abundances of most species are close to the benchmark values in every model. Most interestingly, this is even true in the case of the 10^5 Draine, low density model for which the temperature is poorly captured. It appears Chemulator is able to recover the chemistry even in cases where the temperature predictions are poor.

However, Chemulator does overestimate the CO abundance at low A_v , where the abundance is low. Since the CO column density is not included in the model inputs, one might expect a poor CO self-shielding treatment is the problem. However, the UCLCHEM benchmarking (e.g., Fig. 2) shows UCLCHEM does not suffer from this issue and so it cannot be a result of the inputs. Therefore, this error is more likely due to the fact that CO is most often found at higher abundances and so the emulator struggles with the unusual case of a low CO abundance. Similar behaviour can be seen in Fig. 7 with species such as O, which often have a high abundance. The emulator performs very poorly in situations where a species has a much lower than usual abundance but performs well when the species has a high abundance.

5. Conclusions

The code Chemulator² has been presented. Chemulator is an emulator that advances the gas temperature and chemical abundances of a single position in an astrophysical gas. It is very accurate on a single timestep and is stable over many iterations with decreased accuracy.

An autoencoder was used to reduce the dimensionality of the problem. This successfully reduced the number of chemical variables in the model from 33 to 8. However, it was found that the encoded space could not be uniformly sampled as this led to spurious abundances. An improved dimensionality reduction procedure could both allow this emulator to be extended to much larger networks and lead to a better sampling of the input parameter space.

Chemulator was used to calculate the gas temperature of a standard suite of PDR models. It performed well when the visual extinction was high, demonstrating both accuracy and stability. However, it gave large errors at low visual extinctions, often underestimating the temperature and giving a temperature a factor of two too small in a model with an external UV field of 10^5 Draine and a density of 10^3 cm^{-3} .

Given the density and UV field constraints on the performance of Chemulator, it would be a useful code for applications such as large-scale ISM modelling. It should be noted that the code to develop these emulators has been released alongside the pre-trained Chemulator. Thus, more specialized applications, such as the modelling of planetary atmospheres, could also be served by retraining the emulator for a given parameter space. Overall, Chemulator is a strong first step, demonstrating the promise of this approach. However, improvements need to be made to make it more generally usable as a chem-

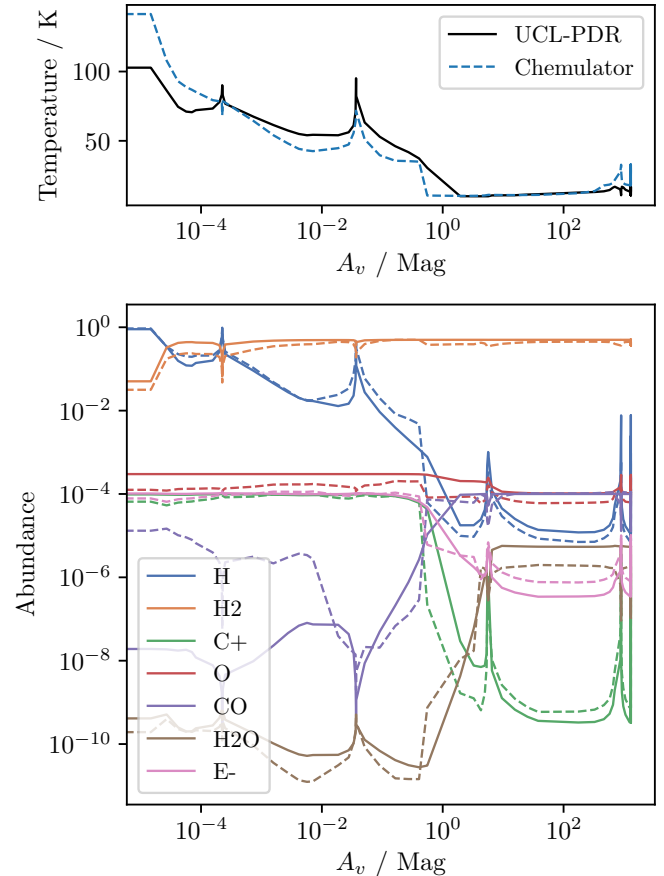


Fig. 11. Similar to Fig. 2 but comparing UCL_PDR and Chemulator. *Upper:* gas temperature against A_v for a cloud with sinusoidally varying density calculated by UCL_PDR and emulated by Chemulator. *Lower:* fractional abundance against A_v . In both plots, results from UCL_PDR are plotted as solid lines and equivalent Chemulator values are dashed.

ical tool. It is likely that the emulator could be improved by a more accurate or less complex dimensionality reduction. The introduction of a noise layer during training ensured that the emulator was resilient to small errors in the encoded variables, but it is possible that the autoencoder introduces large changes to the encoded variables for small changes in abundance, making it difficult for the emulator to learn the relationships between the encoded variables, the physics, and their subsequent values.

Further improvements could be made through the production of a training dataset that is better engineered to cover all realistic inputs. The dimensionality reduction is a part of this, but it is also important to investigate sampling techniques that will produce a dataset that uniformly covers the chemical parameter space rather than just the parameter space of the initial physical inputs. It is possible that the poor performance at low visual extinctions could be rectified by altering the training set to more strongly represent these regions.

Acknowledgements. We thank the anonymous referee for their helpful comments which improved this manuscript. This work is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme MOPPEX 833460. JDI acknowledges support from the Science and Technology Facilities

² <https://github.com/uclchem/Chemulator>

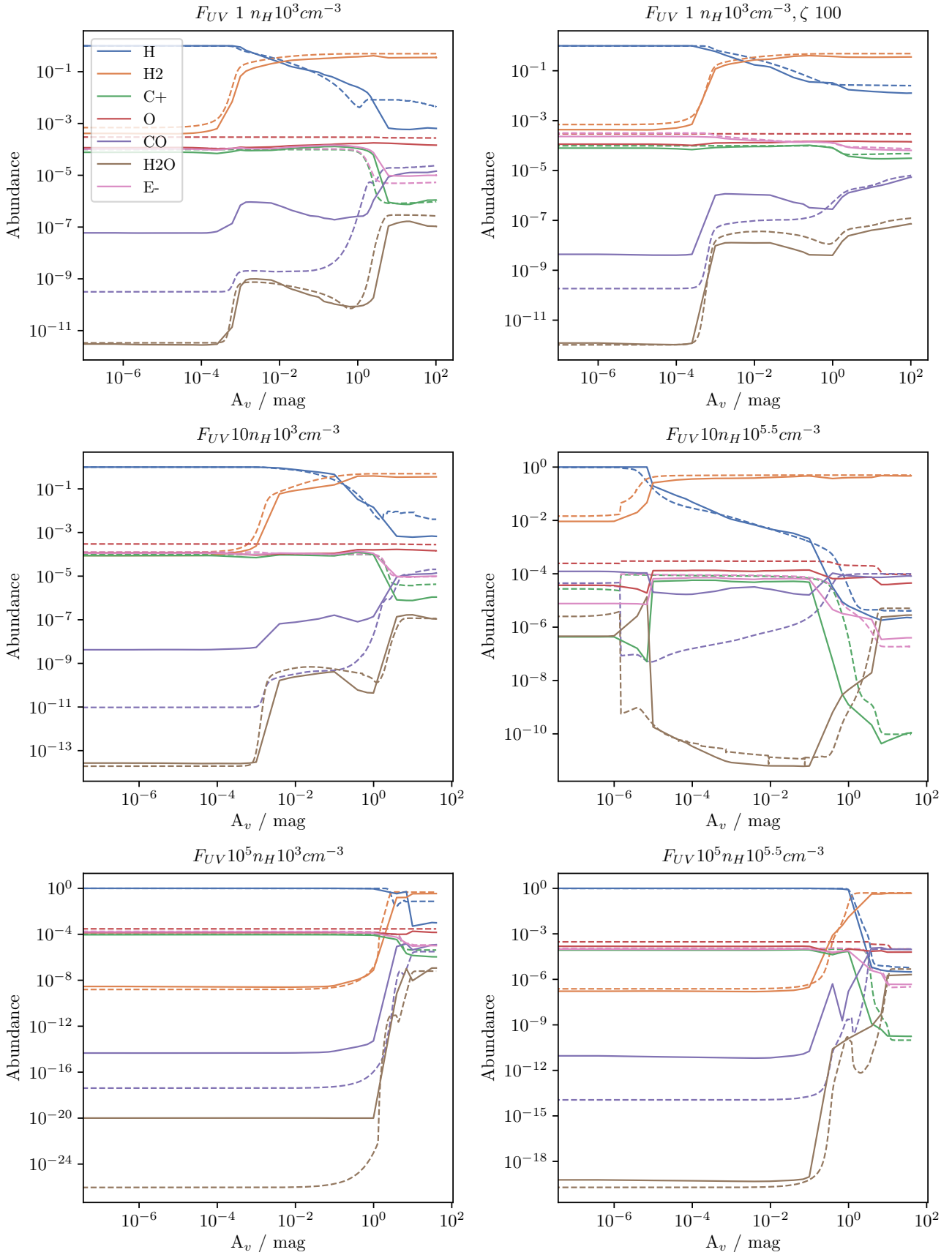


Fig. 12. Comparison of the equilibrium abundances obtained with UCL_PDR and Chemulator for each of the benchmark models. UCL_PDR abundances are shown as dashed lines, and the Chemulator abundances are shown as solid lines.

Council of the United Kingdom (STFC) under ST/T000287/1. The authors thank DiRAC for use of their HPC system which allowed this work to be performed.

References

- Ali, A. A. 2021, *MNRAS*, 501, 4136
- Ali, A., Harries, T. J., & Douglas, T. A. 2018, *MNRAS*, 477, 5422
- Asplund, M., Grevesse, N., Sauval, A. J., & Scott, P. 2009, *ARA&A*, 47, 481
- Bell, T. A., Viti, S., Williams, D. A., Crawford, I. A., & Price, R. J. 2005, *MNRAS*, 357, 961
- Black, J. H. 1987, in *Interstellar Processes*, Proceedings of a symposium, held at Grand Teton National Park, Wyo., July 1986, eds. D. J. Hollenbach, & H. A. Thronson (Dordrecht: Reidel), *Astrophys. Space Sci. Libr.*, 134, 731
- Burke, J. R., & Hollenbach, D. J. 1983, *The Gas-grain Interaction in the Interstellar Medium Thermal Accommodation And Trapping*, Tech. rep.
- Cen, R. 1992, *ApJS*, 78, 341
- de Jong, T., Dalgarno, A., & Boland, W. 1980, *A&A*, 91, 68
- de Mijolla, D., Viti, S., Holdship, J., Manolopoulou, I., & Yates, J. 2019, *A&A*, 630, A117
- Deng, H., Mayer, L., & Meru, F. 2017, *ApJ*, 847, 43
- Drummond, B., Mayne, N. J., Manners, J., et al. 2018, *ApJ*, 855, L31
- Dyson, J., & Williams, D. 1997, *The Physics of the Interstellar Medium* (CRC Press)
- Few, C. G., Courty, S., Gibson, B. K., et al. 2012, *MNRAS*, 424, L11
- Glover, S. C., & Clark, P. C. 2012, *MNRAS*, 421, 116
- Glover, S. C. O., & Mac Low, M. 2007, *ApJS*, 169, 239
- Goldsmith, P. F. 2001, *ApJ*, 557, 736
- Grassi, T., Merlin, E., Piovani, L., Buonomo, U., & Chiosi, C. 2011, ArXiv e-prints [arXiv:1103.0509]
- Grassi, T., Bovino, S., Gianturco, F. A., Baiocchi, P., & Merlin, E. 2012, *MNRAS*, 425, 1332
- Grassi, T., Bovino, S., Schleicher, D. R. G., et al. 2014, *MNRAS*, 439, 2386
- Grassi, T., Ercolano, B., Szűcs, L., Jennings, J., & Picogna, G. 2020, *MNRAS*, 494, 4471
- Haworth, T. J., & Clarke, C. J. 2019, *MNRAS*, 485, 3895
- Haworth, T. J., Ilee, J. D., Forgan, D. H., et al. 2016, *PASA*, 33, e053
- Hirano, S., & Yoshida, N. 2013, *ApJ*, 763, 52
- Holdship, J., Viti, S., Jiménez-Serra, I., Makrymallis, A., & Priestley, F. 2017, *AJ*, 154, 38
- Hollenbach, D., & McKee, C. F. 1979, *ApJS*, 41, 555
- Hollenbach, D., & Tielens, A. G. 1999, *Rev. Mod. Phys.*, 71, 173
- Ilee, J. D., Forgan, D. H., Evans, M. G., et al. 2017, *MNRAS*, 472, 189
- Iliev, I. T., Whalen, D., Mellema, G., et al. 2009, *MNRAS*, 400, 1283
- Kamp, I., & van Zadelhoff, G.-J. 2001, *A&A*, 373, 641
- Krijt, S., Bosman, A. D., Zhang, K., et al. 2020, *ApJ*, 899, 134
- Martin, P. G., Keogh, W. J., & Mandy, M. E. 1998, *ApJ*, 499, 793
- McElroy, D., Walsh, C., Markwick, A. J., et al. 2013, *A&A*, 550, A36
- McKay, M. D., Beckman, R. J., & Conover, W. J. 1979, *Technometrics*, 21, 239
- Meru, F., & Bate, M. R. 2011, *MNRAS*, 411, L1
- Meru, F., & Bate, M. R. 2012, *MNRAS*, 427, 2022
- Nelson, R. P., & Langer, W. D. 1999, *ApJ*, 524, 923
- Pillepich, A., Springel, V., Nelson, D., et al. 2018, *MNRAS*, 473, 4077
- Ploekinger, S., & Schaye, J. 2020, *MNRAS*, 497, 4857
- Priestley, F., Barlow, M. J., & Viti, S. 2017, *MNRAS*, 472, 4444
- Ramachandran, P., Zoph, B., & Le, Q. V. 2018, *6th International Conference on Learning Representations, ICLR 2018 – Workshop Track Proceedings*
- Röllig, M., Ossenkopf, V., Jeyakumar, S., Stutzki, J., & Sternberg, A. 2006, *A&A*, 451, 917
- Röllig, M., Abel, N. P., Bell, T. A., et al. 2007, *A&A*, 467, 187
- Venot, O., Hébrard, E., Agúndez, M., et al. 2012, *A&A*, 546, A43
- Weingartner, J. C., & Draine, B. T. 2001, *ApJS*, 134, 263
- Young, M. D., & Clarke, C. J. 2015, *MNRAS*, 451, 3987

Appendix A: Extending to larger networks

An early iteration of this work utilized a network of 215 species interacting through 2508 reactions including both gas and grain surface species. The possibility of producing an emulator that could solve such complex chemistry with a small computation time was an obvious goal. However, no working emulator could be produced and in this appendix, the strengths and failings of that model are discussed.

Interestingly, despite the fact that this network had almost seven times as many species as the small network, it was possible to produce an autoencoder that had a similar accuracy to the small network encoder without being much larger. A network with one hidden layer of 256 neurons in both the encoder and decoder was used, just like the final autoencoder in Sect. 3.2. With an encoded size of 12, rather than 8, chemical variables, the autoencoder for the large network could obtain MSE values $\sim 10^{-5}$. This is promising for future work as it implies even large chemical networks can be represented by very few variables.

Following this, a grid of emulators were trained and tested using the encoded chemistry and physical inputs. The best emulator had a single timestep MSE of 2.9×10^{-4} , approximately a factor of two larger than the small network emulator. However, even with the introduction of ensemble models and the Gaussian noise layer, which ensured the small network emulator had a stable error, error growth could not be prevented in this model.