GRACKLE: a chemistry and cooling library for astrophysics

Grackle: a Chemistry and Cooling Library for Astrophysics

Britton D. Smith,[1,2]⋆ Greg L. Bryan,[3,4] Simon C. O. Glover,[5] Nathan J. Goldbaum,[6]
Matthew J. Turk,[7,8] John Regan,[9,10] John H. Wise,[11] Hsi-Yu Schive,[6] Tom Abel,[12,13]
Andrew Emerick,[3,14] Brian W. O'Shea,[15,16,17] Peter Anninos,[18]
Cameron B. Hummels,[19] and Sadegh Khochfar[1]

[1]*Institute for Astronomy, University of Edinburgh, Royal Observatory, Edinburgh EH9 3HJ, UK*
[2]*San Diego Supercomputer Center, University of California, San Diego, 10100 Hopkins Drive, La Jolla, CA 92093*
[3]*Columbia University, Department of Astronomy, New York, NY, 10025, USA*
[4]*Simons Center for Computational Astrophysics, New York, NY, USA*
[5]*Universität Heidelberg, Zentrum für Astronomie, Institut für Theoretische Astrophysik, Albert-Ueberle-Straße 2, 69120 Heidelberg, Germany*
[6]*National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign, IL, 61820, USA*
[7]*School of Information Sciences, University of Illinois, Urbana-Champaign, IL, 61820, USA*
[8]*Department of Astronomy, University of Illinois, Urbana-Champaign, IL, 61820, USA*
[9]*Institute for Computational Cosmology, Durham University, South Road, Durham, DH1 3LE, UK*
[10]*School of Mathematical Sciences, Dublin City University, Dublin, Ireland*
[11]*Center for Relativistic Astrophysics, Georgia Institute of Technology, 837 State Street, Atlanta, GA 30332, USA*
[12]*Kavli Institute for Particle Astrophysics and Cosmology, Stanford University, Menlo Park, CA 94025, USA*
[13]*Department of Physics, Stanford University, Stanford, CA 94305, USA*
[14]*American Museum of Natural History, Department of Astrophysics, New York, NY, USA*
[15]*Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA*
[16]*Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, MI 48824, USA*
[17]*JINA: Joint Institute for Nuclear Astrophysics, Michigan State University, East Lansing, MI 48824, USA*
[18]*Lawrence Livermore National Laboratory, Livermore, CA 94550*
[19]*California Institute of Technology, Pasadena, CA 91125, USA*

ABSTRACT

We present the `Grackle` chemistry and cooling library for astrophysical simulations and models. `Grackle` provides a treatment of non-equilibrium primordial chemistry and cooling for H, D, and He species, including H_2 formation on dust grains; tabulated primordial and metal cooling; multiple UV background models; and support for radiation transfer and arbitrary heat sources. The library has an easily implementable interface for simulation codes written in C, C++, and Fortran as well as a Python interface with added convenience functions for semi-analytical models. As an open-source project, `Grackle` provides a community resource for accessing and disseminating astrochemical data and numerical methods. We present the full details of the core functionality, the simulation and Python interfaces, testing infrastructure, performance, and range of applicability. `Grackle` is a fully open-source project and new contributions are welcome.

Key words: astrochemistry – methods: numerical – galaxies: formation

1 INTRODUCTION

1.1 Why are chemistry and radiative cooling necessary in simulations?

Modeling of plasma chemistry and radiative cooling is absolutely essential in a wide range of astrophysical phenomena. At a fundamental level, virtually all astrophysical objects begin as a cloud of diffuse plasma in a gravitational poten-

⋆ E-mail: brittonsmith@gmail.com (BDS)

arXiv:1610.09591v2 [astro-ph.CO] 14 Dec 2016

tial that is created primarily either by the plasma itself (i.e., stars) or by a dark matter halo (i.e., cosmological structure). In these situations, in the absense of any additional physical processes, the plasma will arrange itself so that pressure is in rough equilibrium with gravity, and no further evolution will occur without some external influence. Some physical process that allows this plasma to lose energy is necessary in order to break this stalemate, thus allowing the formation of stars and galaxies. The process that typically enables this energy loss is radiative cooling, often facilitated by a series of chemical reactions that further enhance the plasma's ability to lose energy.

Radiative cooling plays a critical role in several important astrophysical processes. There is a complex interplay of gas- and dust-phase chemistry in star formation, which completely dominates the dynamics of the evolving pre-stellar cloud, and may strongly influence the resulting stellar initial mass function (Abel et al. 2002; Turk et al. 2011b; Glover 2008). At a smaller physical scale, radiative cooling can profoundly affect the structure and behavior of accretion disks around stars and compact objects (Shakura & Sunyaev 1973). The shape of the 'cooling curve' of diffuse astrophysical plasmas – i.e., the cooling rate as a function of density and temperature – is responsible for the thermal instability that generates a multiphase interstellar medium (McKee & Ostriker 1977; Spitzer 1978). In cosmological structure formation, gas collapsing into dark matter halos generally experiences a strong shock at roughly the virial radius, which heats the gas to roughly the virial temperature. Optically-thin radiative cooling of this plasma allows gas to concentrate at the center of dark matter halos, and ultimately to form molecular clouds and stars (Rees & Ostriker 1977; White & Frenk 1991).

Chemistry is often an important component of the evolution of astrophysical plasmas. The creation of simple molecules via gas- and dust-phase chemical reactions can greatly enhance the efficacy of cooling (Hollenbach & McKee 1979; Omukai et al. 2005). The formation and destruction of simple molecules can be energetically important in some circumstances, such as Population III star formation (Omukai & Nishi 1998; Abel et al. 2002; Glover & Abel 2008; Turk et al. 2009). And, from a dynamical perspective, the non-equilibrium evolution of particular ions can have a strong effect on the ability of gas to cool (Abel et al. 1997; Anninos et al. 1997) and on common observational quantities such as the column density of O VI absorption line systems in the intergalactic medium that are used to estimate the metal content of the intergalactic medium and to trace the "missing baryons" (Cen & Fang 2006; Smith et al. 2011; Hummels et al. 2013; Oppenheimer & Schaye 2013; Shull et al. 2014).

1.2 Why is a multi-code library a good thing?

Utilizing identical implementations of microphysical solvers, such as those for chemistry, provides several distinct advantages to the developers and users of simulation codes as well as the broader community of researchers who synthesize results of those simulation codes.

Firstly, and most importantly, the development of a microphysical solver that can be applied to multiple independent simulation codes reduces the technical, social and cognitive overhead for collaboration amongst simulators. This

collaboration is a positive benefit in and of itself (particularly as it reduces duplication of effort) but also provides opportunities to share understanding, propagate bug fixes, and also to collaboratively implement algorithms that provide higher-fidelity results. Increasing the ability of researchers to directly collaborate around technology will increase overall productivity.

With libraries such as the one presented here, where there is a set of reference physical values (such as chemical kinetic rate coefficients, drawn from experimentation or detailed theoretical calculation), the existence of such a library provides a fixed reference point for calculations that utilize those reference values. For instance, by citing a particular version of the libary in a simulation paper, researchers can indicate which set of values were used in the calculation. In cases where results are found to sensitively depend on these values, or when these reference values are updated, this can guide deeper physical understanding.

An often underrecognized benefit of portable libraries is the impact they have on the education and training of early-stage researchers. For researchers at early stages in their career (such as graduate students and postdocs) where changing institutions may mean utilizing a different simulation platform, a shared chemistry package enables them to immediately transfer developed code to new projects, rather than having to reimplement in a new system. Additionally, exposure to such a code that is a mix of legacy and modern development styles, from several different maintainers, provides insight into practices in computational science.

Perhaps the most obvious benefit to utilizing a multi-platform library such as that which we shall now present, particularly in domains such as galaxy formation (where controlled comparisons are necessary to decouple effects of star formation prescriptions), is the simplification of directly comparing and reducing the number of potential sources of difference between multiple calculations.

1.3 Introducing Grackle

In this paper, we introduce the `Grackle` chemistry and cooling library for astrophysical simulations. The aim of the `Grackle` project is to provide all of the benefits outlined above: to provide a community resource for accessing and disseminating data and methods, create a citable version history for evolving functionality, simplify comparison of results, and reduce the overhead for collaboration. In Section 2, we give an account of the history of the `Grackle` source code and its development leading up to this publication. Following this, we describe the library in full. In Section 3, we detail the non-equilibrium primordial chemistry solver. In Section 4, we discuss the radiative cooling and heating processes that are included. In Section 5, we describe the treatment of UV radiation backgrounds. In Section 6, we discuss implementation details, including the simulation code API, the Python interface, the organization of the source code, and the procedure for adding to the code. In Section 7, we describe the testing infrastructure, the optimization strategy, and present performance metrics. Finally, we conclude in Section 8 with a discussion of the physical conditions in which the code is valid, other limitations that should be considered, a list of simulation codes known to have implemented `Grackle`, and some future directions for the project.

Table 1. Important `Grackle` resources.

Source code	https://bitbucket.org/grackle/grackle
Documentation	https://grackle.readthedocs.io/
Mailing list	"grackle-cooling-users" on Google Groups

Table 2. Dates of major `Grackle` releases.

Version	Release Date
1.0	January 10, 2014
1.1	October 1, 2014
2.0	October 1, 2014
2.1	June 3, 2015
2.2	May 18, 2016
3.0	November 1, 2016

Table 1 lists the locations of important `Grackle`-related resources.

2 ORIGINS AND EARLY HISTORY

The original `Grackle` source code dates back to work done in 1995 by Peter Anninos and collaborators (Anninos et al. 1997) who developed a static Eulerian code focused on primordial gas (e.g. Lyman alpha forest, first stars). This code was then incorporated into the `Enzo` codebase in 2000 and modified to include more rates and physical processes. Metal cooling using tables from `Cloudy` was added in 2007 (Smith et al. 2008). In 2012, the AGORA simulation comparison project (Kim et al. 2014) was first organized with the goal of using a "common physics package" to eliminate differences in results due to the use of cooling solvers that were specific to each simulation code. The modularity of `Enzo`'s cooling solver made it relatively straightforward to extract it from the `Enzo` source. The initial commit to the `Grackle` repository, marking the extraction of the core chemistry and cooling machinery, was made on October 2, 2012. The first official release (`Grackle` 1.0) occurred on January 10, 2014, with five additional major releases following over the next two and half years. Integer increments of the version number marked changes to the application program interface (API), while decimal releases included only new features and bugfixes. A list of all major releases is given in Table 2. A summary of `Grackle`'s development history can be found at `https://www.openhub.net/p/thegrackle`.

3 PRIMORDIAL CHEMISTRY

The treatment of primordial chemistry (i.e. the chemistry of metal-free gas) used in `Grackle` is closely based on the treatment in the Enzo AMR code (Bryan et al. 2014), although `Grackle` accounts for a few processes that are not included in the latest version of Enzo available at the time of writing (version 2.5). The Enzo primordial chemistry itself is based originally on the work of Abel et al. (1997) and Anninos et al. (1997), although the current version has been modified substantially compared to the original Abel et al. network. In this section, we describe in detail the chemistry

Table 3. Chemical reactions in the six species network

Reaction			Reference	
			Data	Fit
$H + e^-$	$\rightarrow$	$H^+ + e^- + e^-$	1	2
$H^+ + e^-$	$\rightarrow$	$H + \gamma$	3	2, 4
$He + e^-$	$\rightarrow$	$He^+ + e^- + e^-$	1	2
$He^+ + e^-$	$\rightarrow$	$He + \gamma$	5, 6	4, 6, 7
$He^+ + e^-$	$\rightarrow$	$He^{++} + e^- + e^-$	1	2
$He^{++} + e^-$	$\rightarrow$	$He^+ + \gamma$	3, 8	4, 9
$H + H$	$\rightarrow$	$H^+ + e^- + H$	10	11
$H + He$	$\rightarrow$	$H^+ + e^- + He$	12	11
$H + \gamma$	$\rightarrow$	$H^+ + e^-$	13	—
$He + \gamma$	$\rightarrow$	$He^+ + e^-$	13	—
$He^+ + \gamma$	$\rightarrow$	$He^{++} + e^-$	13	—

Key: 1 – Janev et al. (1987); 2 – Abel et al. (1997); 3 – Ferland et al. (1992); 4 – Hui & Gnedin (1997); 5 – Burgess & Seaton (1960); 6 – Aldrovandi & Pequignot (1973); 7 – Black (1981); 8 – Spitzer (1978); 9 – Cen (1992); 10 – Gealy & van Zyl (1987); 11 – Lenzuni et al. (1991); 12 – van Zyl et al. (1981); 13 – see Section 5

included in `Grackle` and discuss how the resulting set of chemical rate equations is solved.

3.1 Chemistry Network

`Grackle` provides three different primordial chemistry networks, differing in the number of chemical species that they include. The choice of chemical network is controlled by the `primordial_chemistry` parameter. Setting `primordial_chemistry = 1` selects the six species network, which tracks the abundances of the species H, H^+, He, He^+, He^{++} and e^- and is designed for modelling atomic and/or ionized gas. Setting `primordial_chemistry = 2` selects the nine species network. This includes all of the species and reactions included in the six species network, but adds molecular hydrogen (H_2), plus the two ions primarily responsible for its formation in primordial gas (H^- and H_2^+). Finally, setting `primordial_chemistry = 3` selects the twelve species network, which is an extension of the nine species model that includes D, D^+ and HD.

The reactions included in the six species network are listed in Table 3. The rate coefficients for these reactions are implemented in `Grackle` using simple temperature-dependent analytical fits. In the Table, we list the references from which we take these fits, and also the references that are the original sources of the theoretical or experimental data on which these fits are based.

A few of the reactions listed in Table 3 deserve further comment:

(i) By default, the recombination of H^+, He^+ and He^{++} is modelled using the case A recombination rate coefficients (the optically-thin approximation in which recombination photons above 1 Ryd escape). However, the case B rate coefficients (in which recombination photons above 1 Ryd are locally re-absorbed, Osterbrock 1989) can instead be selected by setting `CaseBRecombination = 1`. The additional complication that photons from the recombination of He^+ can bring about the photoionization of hydrogen (discussed in some detail in Osterbrock 1989) is not accounted for, but

Table 4. Chemical reactions in the nine species network

Reaction			Reference	
			Data	Fit
$H + e^-$	$\rightarrow$	$H^- + \gamma$	1	2
$H^- + H$	$\rightarrow$	$H_2 + e^-$	3	3
$H + H^+$	$\rightarrow$	$H_2^+ + \gamma$	4	5
$H_2^+ + H$	$\rightarrow$	$H_2 + H^+$	6	6
$H_2 + H^+$	$\rightarrow$	$H_2^+ + H$	7	8
$H_2 + e^-$	$\rightarrow$	$H + H + e^-$	9	9
$H_2 + H$	$\rightarrow$	$H + H + H$	10	10
$H^- + e^-$	$\rightarrow$	$H + e^- + e^-$	11	12
$H^- + H$	$\rightarrow$	$H + e^- + H$	11	12
$H^- + H^+$	$\rightarrow$	$H + H$	13	14
$H^- + H^+$	$\rightarrow$	$H_2^+ + e^-$	15	12, 16
$H_2^+ + e^-$	$\rightarrow$	$H + H$	17	12
$H_2^+ + H^-$	$\rightarrow$	$H_2 + H$	18	18
$H + H + H$	$\rightarrow$	$H_2 + H$	19	19
$H + H + H_2$	$\rightarrow$	$H_2 + H_2$	20, 21	22
$H^- + \gamma$	$\rightarrow$	$H + e^-$	23	—
$H_2^+ + \gamma$	$\rightarrow$	$H + H^+$	23	—
$H_2 + \gamma$	$\rightarrow$	$H_2^+ + e^-$	23	—
$H_2^+ + \gamma$	$\rightarrow$	$H^+ + H^+ + e^-$	23	—
$H_2 + \gamma$	$\rightarrow$	$H + H$	23	—
$H + H + grain$	$\rightarrow$	$H_2 + grain$	—	24*

Note: the nine species network also includes all of the reactions listed in Table 3.

Key: 1 – Wishart (1979); 2 – Stancil et al. (1998); 3 – Kreckel et al. (2010); 4 – Ramaker & Peek (1976); 5 – Latif et al. (2015); 6 – Karpas et al. (1979); 7 – Krstić (2002); 8 – Savin et al. (2004a,b); 9 – Trevisan & Tennyson (2002); 10 – Martin et al. (1996); 11 – Janev et al. (1987); 12 – Abel et al. (1997); 13 – Fussen & Kubach (1986); 14 – Croft et al. (1999); 15 – Poulaert et al. (1978); 16 – Shapiro & Kang (1987); 17 – Schneider et al. (1994); 18 – Dalgarno & Lepp (1987); 19 – See text; 20 – Sutton (1962); 21 – Ham et al. (1970); 22 – Cohen & Westberg (1983); 23 – see Section 5; 24 – Tielens & Hollenbach (1985); Omukai (2000); * - This reaction included as an additional option when metals are present.

in most circumstances this will only lead to a small error in the H^+ and He^+ abundances.

(ii) The rates of the photoionization reactions are not calculated internally by `Grackle`, but instead are specified either via an input data file or passed directly to `Grackle` via the `Grackle` API as described in Section 5.

(iii) The six species network implemented in `Grackle` includes two additional reactions that were not part of the original Abel et al. (1997) six species network: the collisional ionization of H by collisions with H and He atoms. Often, these reactions are unimportant. However, they can become competitive with the collisional ionization of H by electrons if the fractional ionization of the gas is very low (see, e.g. Glover 2015, for an example of when this can be important).

The nine species network includes all of the reactions in Table 3 plus the additional reactions listed in Table 4. Again, a couple of reactions deserve further discussion:

(i) The treatment of H_2 collisional dissociation by H atom collisions now used in `Grackle` is taken from Martin et al. (1996) and accounts for both the temperature and the density dependence of this process. It therefore remains valid in the high density limit, where the H_2 level populations approach their local thermodynamical equilibrium (LTE) val-
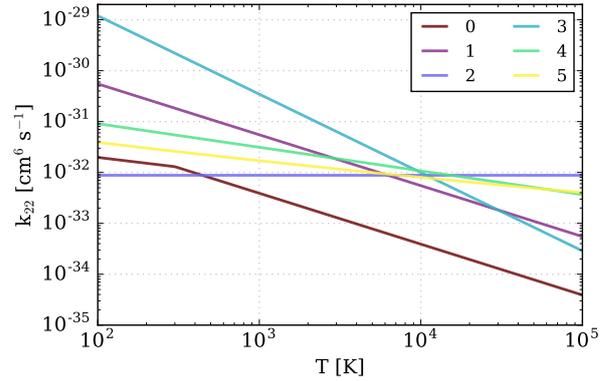

Figure 1. The available three-body H_2 formation rates as a function of temperature: 0 - Abel et al. (2002), 1 - Palla et al. (1983), 2 - Cohen & Westberg (1983), 3 - Flower & Harris (2007), 4 - Glover (2008), 5 - Forrey (2013).

ues. This is important, because H_2 is far more susceptible to collisional dissociation in this limit than when it is solely in the vibrational ground state. It should also be noted that the treatment of this process in `Grackle` accounts for effects of dissociative tunneling as well as direct collisional dissociation; previously, Enzo only accounted for the latter process and hence underestimated the dissociation rate at low temperatures (Latif et al. 2014; Glover 2015)

(ii) In view of the considerable uncertainty in the rate of the three-body reaction

$$H + H + H \rightarrow H_2 + H, \tag{1}$$

discussed in detail in Glover (2008) and Turk et al. (2011b), `Grackle` provides several different rate coefficients for this process. The user can select which of these rate coefficients to adopt by means of the `three_body_rate` parameter. The options are

0: Rate coefficient from Abel et al. (2002), based on an extrapolation from low temperature calculations by Orel (1987). This is the default option.

1: Rate coefficient from Palla et al. (1983), derived using detailed balance applied to the H_2 collisional dissociation rate measured by Jacobs et al. (1967).

2: Rate coefficient recommended by Cohen & Westberg (1983), based on a survey of the experimental data available at that time.

3: Rate coefficient from Flower & Harris (2007), also derived from Jacobs et al. (1967) using detailed balance, but with a different treatment of the H_2 partition function.

4: Rate coefficient from Glover (2008), derived from the Martin et al. (1996) high-density H_2 collisional dissociation rate using detailed balance

5: Rate coefficient computed directly by Forrey (2013).

We plot each of these rates in Figure 1.

Finally, the twelve species network includes the reactions in Tables 3 and 4, plus a small number of additional reactions involving D^+, D and HD, listed in Table 5. The intent of the twelve species network is to allow the HD abundance of the gas to be tracked accurately, since in cold gas HD can become a more effective coolant than H_2 despite its much lower fractional abundance (see e.g. Johnson &

Table 5. Additional reactions included in the twelve species network

	Reaction		Reference	
			Data	Fit
$H^+ + D$	$\rightarrow$	$H + D^+$	1, 2	3
$D^+ + H$	$\rightarrow$	$D + H^+$	1, 2	3
$H_2 + D^+$	$\rightarrow$	$HD + H^+$	4	5
$HD + H^+$	$\rightarrow$	$H_2 + D^+$	4	5
$H_2 + D$	$\rightarrow$	$HD + H$	6	7
$HD + H$	$\rightarrow$	$H_2 + D$	8	5, 9
$D + H^-$	$\rightarrow$	$HD + e^-$	10, 11	10

Note: the twelve species network also includes all of the reactions listed in Tables 3 & 4.
Key: 1 – Igarashi & Lin (1999); 2 – Zhao et al. (2000); 3 – Savin (2002); 4 – Gerlich (1982); 5 – Galli & Palla (2002); 6 – Mielke et al. (2003); 7 – Clark et al. (2011); 8 – Shavitt (1959); 9 – Ripamonti (2007); 10 – Kreckel et al. (2010); 11 – Miller et al. (2012)

Bromm 2006; McGreer & Bryan 2008). It is therefore necessary to include only a small number of reactions, as the direct conversion of H_2 to HD by collisions with D^+ and D, together with the corresponding inverse reactions generally dominate the production and destruction of HD.

Two reactions in the twelve species network require further discussion:

(i) The rate coefficient that we adopt for the reaction

$$HD + H \rightarrow H_2 + D \tag{2}$$

is an analytical fit presented in Galli & Palla (2002), based on data from Shavitt (1959). However, this fit blows up at temperatures $T < 100$ K, yielding an unphysically large value for the rate coefficient. We therefore follow Ripamonti (2007) and McGreer & Bryan (2008) and assume that the rate at $T < 100$ K is the same as the rate at $T = 100$ K. Note that as this reaction proceeds extremely slowly at temperatures below a few hundred K, this is unlikely to be a significant source of error.

(ii) We assume that the rate coefficient for the associative detachment of H^- by D

$$D + H^- \rightarrow HD + e^- \tag{3}$$
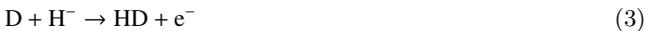
is the same as for the corresponding reaction between H and H^-, since measurements by Miller et al. (2012) suggest that there is not a significant isotope effect for this reaction. However, in the solver, we multiply the rate coefficient by a factor of two when computing the HD formation rate to account approximately for the contribution from the reaction

$$H + D^- \rightarrow HD + e^-. \tag{4}$$

Note that we do not explicitly include this reaction in our network because it would require us to track the abundance of the D^- ion, thereby adding significant additional complexity to the model for only a marginal increase in accuracy.

3.2 Solving and Updating the Network

Chemical networks such as the ones described above are often challenging to evolve due to the very different time scales

that various rates may have – creation and destruction time scales can differ by orders of magnitude among the different species. Such "stiff" sets of equations are often solved with implicit methods, which permit longer timesteps. Several packages exist to do this, and can even switch between methods for solving stiff and non-stiff equations, such as LSODAR (Hindmarsh 1983); however, for multi-dimensional simulations it is useful to have an implementation which is optimized for the case at hand.

Our kinetic network for solving the rate of change of species density n_i has the general form

$$\frac{\partial n_i}{\partial t} = \sum_j \sum_l k_{jl} n_j n_l + \sum_j I_j n_j \tag{5}$$

where k_{jl} is the rate for reactions involving species j and l while I_j is the appropriate radiative rate. Note, that the terms on the right-hand side of Equation 5 can denote either creation or destruction reactions. For creation reactions, the corresponding term is positive; for destruction reactions, it is negative. In rare cases there may be an additional term for three-body reactions. To solve this kinetic network, `Grackle` follows closely the procedure described in Anninos et al. (1997) and Bryan et al. (2014) by grouping creation and destruction rates to rewrite Eq. 5 as,

$$\frac{\partial n_i}{\partial t} = C_i(T, n_j) - D_i(T, n_j) n_i \tag{6}$$

where C_i represents the total creation rate of species i (given the temperature T and other species densities) while $D_i n_i$ is the destruction rate of the same species (which must be proportional to n_i), including both radiative and collisional processes. Ideally, we would solve these ordinary differential equations using a higher order method; however, here we adopt a very simple low order backwards difference formula (BDF) due to its stability. Anninos et al. (1997) explored a variety of higher-order solution techniques but found that this simple BDF scheme was generally more stable and competitive for the level of accuracy required. In particular, the BDF version of Equation 6 is:

$$n^{t+\Delta t} = \frac{C^{t+\Delta t} \Delta t + n^t}{1 + D^{t+\Delta t} \Delta t} \tag{7}$$

Unfortunately, we are not able to fully implement a BDF scheme due to the difficulty of evaluating the C_i and D_i at the advanced time. Instead, we attempt to mimic a BDF method through a set of partial updates combined with subcycling. The partial forwarding updating is done by solving the various species in a specified order and using the updated species densities in the following partial step. The ordering developed by Anninos et al. (1997) was based on empirical tests under a wide range of conditions and is as follows for the six species model: H, H^+, e^-, He, He^+, He^{++}. For the nine species model, we then add H_2, H^-, and H_2^+. The H_2^+ timescale is sufficiently short that it can be decoupled and we use instead the equilibrium value:

$$n_{H_2^+} = \frac{k_9 n_H n_{H^+} + k_{11} n_{H_2} n_{H^+} + k_{17} n_{H^-} n_{H^+} + k_{29} n_{H_2}}{k_{10} n_H + k_{18} n_e + k_{19} n_{H^-} + k_{28} + k_{30}} \tag{8}$$

Finally, for twelve species model, we add D, D^+ and HD. In each case, the updated species of the previous step are used in the next.

The time step passed to `Grackle` (often the hydrodynamic timestep of a parent simulation) can be quite large,

which may potentially cause large errors in our low-order chemical integrator. To get around this issue, we sub-cycle the BDF step described above, constraining the chemical timestep such that the H and e$^-$ abundances change by no more than 10% in any sub-cycle step of length Δt:

$$\Delta t = 0.1 \min \left(\frac{n_{\rm H}}{\dot{n}_{\rm H}}, \frac{n_{\rm e^-}}{\dot{n}_{\rm e^-}} \right) \qquad (9)$$

In some cases, particularly close to equilibrium, we slightly modify this. After more than 50 subcycle steps (if we have not yet integrated a full hydro timestep), we replace the analytically calculated time derivatives in the above expression with numerical time derivatives (i.e. change from the previous sub-cycle step). This is helpful when we are close to equilibrium and the integrator is taking very small steps and regularly overshooting the equilibrium value.

4   HEATING AND COOLING

`Grackle` can evolve the Lagrangian energy equation, taking into account a wide range of radiative cooling and heating processes:

$$\frac{de}{dt} = -\dot{e}_{\rm cool} + \dot{e}_{\rm heat} \qquad (10)$$

In this paper, we split our discussion of radiative cooling/heating and chemistry, but in the code, they are solved together, using a simple first-order integrator. To enhance accuracy, the integrator is sub-cycled with a timestep constraint

$$\Delta t \leq 0.1 \frac{e}{\dot{e}}. \qquad (11)$$

If both chemistry and cooling/heating are turned on, these are integrated at the same time, using a timestep which is a minimum of the chemistry and cooling constraints. In the following sections, we describe the various supported cooling and heating options. We begin with radiative cooling and heating due solely to hydrogen and helium, and then turn to heavier atomic elements ("metals"), and finally dust.

4.1   Primordial Heating and Cooling

To solve for the effects of primordial (H and He only) heating and cooling, `Grackle` includes two options: (1) a non-equilibrium solver, the chemistry part of which is described in detail in Section 3, and (2) a tabulated version, which assumes ionization equilibrium to compute the cooling and heating rates due to primordial chemistry. In both cases, photo-heating from an external radiation source can be important – this is described in Section 5.

4.1.1   Non-equilibrium

We include a variety of cooling rates due to transitions of the non-equilibrium species. We begin with a list appropriate for the six species network (`primordial_chemistry = 1`):

(i) Collisional excitation cooling rates involving the following species: $n_{\rm e}n_{\rm H}$, $n_{\rm e}^2 n_{\rm He^+}$, and $n_{\rm e}n_{\rm He^+}$ (Black 1981; Cen 1992);

(ii) Collisional ionization cooling for $n_{\rm e}n_{\rm H}$, $n_{\rm e}n_{\rm He}$, $n_{\rm e}n_{\rm He^+}$ and $n_{\rm e}^2 n_{\rm He^+}$ (Shapiro & Kang 1987; Cen 1992; Abel et al. 1997);

(iii) Recombination cooling: $n_{\rm e}n_{\rm H^+}$, $n_{\rm e}n_{\rm He^+}$, $n_{\rm e}n_{\rm He^{++}}$ (Black 1981; Ferland et al. 1992; Hui & Gnedin 1997);

(iv) Bremsstrahlung cooling for all ionized species (Black 1981);

(v) Compton cooling/heating off the CMB (Peebles 1971), and

(vi) Photoionization heating for H, He and He$^+$, depending on the ionizing radiation field – see section 5 for more details.

In addition to the sources referenced above, we note that most of these rates were tabulated in Appendix B of Anninos et al. (1997).

For the nine species version (`primordial_chemistry = 2`), we add H$_2$ cooling. Our default cooling rate is as follows. At high densities, where the level populations are in LTE and hence depend only on temperature, we use the high-density rate from Galli & Palla (1998). For low densities, we computed the cooling rate due to collisions with H, H$_2$, He, H$^+$, and e$^-$ as described in section 2.3 of Glover & Abel (2008). The exception is that for collisions with e$^-$, we use revised rates from Yoon et al. (2008) and for H$^+$, we adopt rates from Honvault et al. (2011) and Honvault et al. (2012). For intermediate densities, we use the smooth density-dependent switch from Galli & Palla (1998). In addition to this cooling function, the code can also (depending on a compile time switch) use an older rate from Lepp & Shull (1983). At very high densities, `Grackle` can also account for the decrease in the H$_2$ cooling rate that comes about once the H$_2$ lines becomes optically thick. This is treated using a simple density-dependent opacity correction term introduced by Ripamonti & Abel (2004). This option is enabled by setting `h2_optical_depth_approximation = 1`.

In addition to H$_2$ radiative cooling, we include the impact of chemical heating or cooling due to the formation or destruction of molecular hydrogen. Following Omukai (2000), we add $4.48(1 + n_{\rm cr}/n)^{-1}$ eV for H$_2$ formation by the three-body reaction, and $0.2+4.1(1+n_{\rm cr}/n)^{-1}$ eV for H$_2$ formation on dust grain surfaces. The critical density $n_{\rm cr}$ is given by eq. (23) of Omukai (2000). For H$_2$ destruction, we remove 4.48 eV per H$_2$ molecule dissociated.

Finally, in twelve species mode (`primordial_chemistry = 3`), which adds deuterium chemistry, the code includes (radiative) cooling from HD. This is a combination of a fit from Coppola et al. (2011) for the high-density limit, and Wrathmall et al. (2007) for the low density limit.

There are a number of other optional heating and cooling terms that the code includes, some of which are not strictly primordial, but are included as part of this cooling package. These include:

(i) Collisionally induced excitation of H$_2$ at high densities, with rates as described in Ripamonti & Abel (2004).

(ii) X-ray Compton heating (or cooling) using eq. (4) and (11) of Madau & Efstathiou (1999).

(iii) A photoelectric heating rate, equal to $\Gamma_{\rm eff} n_{\rm H}$, where $\Gamma_{\rm eff}$ is a fixed input parameter. Although not strictly primordial, we include this rate here as it is distinct from the dust model described later.

4.1.2 Equilibrium (Tabulated)

In the other (simpler) mode, the cooling and heating due to the primordial elements can be calculated using tables of pre-computed values under the assumption of ionization equilibrium. If there is no incident radiation, then we have simple collisional ionization equilibrium (CIE), and the cooling rate (per hydrogen atom) is solely a function of temperature. This means we can look up the cooling rate using a simple one-dimensional table. If radiation is present, the cooling rate under ionization equilibrium for a fixed spectral shape and intensity is a function of density and temperature, resulting in a two-dimensional table look-up. The process by which these tables are created is discussed in Section 4.2.1. For the primordial elements, `Grackle` provides pre-computed tables for the cooling rate, Λ; the heating rate, Γ; and the mean molecular weight, μ, of the gas as a function of temperature and density, if required. All rates are computed using linear interpolation in log-space.

Since simulation codes typically solve for the internal energy of the gas instead of the temperature, it is necessary to convert one to the other via

$$e = \frac{kT}{(\gamma - 1)\,\mu m_\mathrm{H}}, \qquad (12)$$

where k is the Boltzmann constant, T is the gas temperature, e is the specific internal energy, γ is the adiabatic index of an ideal gas, and m_H is the mass of a hydrogen atom. Since the mean molecular weight, μ, is also a function of temperature, we solve Equation 12 iteratively with an initial guess of $\mu = 1$. The temperature calculated using the initial guess for μ is then used as an input to the table of $\mu(T, ...)$, from which a new value of μ is calculated via linear interpolation. To prevent the solution of μ and T from oscillating, we apply a dampener such that the new value of μ is the average of the old value and the value from the table. For iteration, i, of this procedure, μ_i is then given by

$$\mu_i = \frac{1}{2}(\mu_{i-1} + \mu(T_{i-1}, ...)). \qquad (13)$$

To account for the presence of metals, we then apply an additional correction such that the value with metals included, $\mu_{i,Z}$ is

$$\frac{\rho}{\mu_{i,Z}} = \frac{\rho}{\mu_i} + \frac{\rho_Z}{\mu_Z}, \qquad (14)$$

where ρ is the total gas density, ρ_Z is the metal density, and $\mu_Z \equiv 16$, which is consistent with a Solar abundance pattern. In practice, this process arrives at a solution for μ and T that converges to within 1% in just a few iterations.

Once the gas temperature has been calculated, the cooling and heating due to the primordial species is then computed via interpolating over the multidimensional tables. In the most commonly used mode, heating comes from a model UV background, which is spatially uniform and varies as a function of redshift. In this case, the tables for Λ, Γ, and μ have dimensions of z, ρ, and T. The effects of the UV background models and their implementation within the code are discussed further in section 5.

In a cosmological simulation, the CMB acts as a temperature floor, below which the gas cannot cool radiatively. We approximate this effect by subtracting the cooling rate at the CMB temperature, T_CMB, from the calculated cooling rate such that the final cooling rate that is applied is given by

$$\Lambda_\mathrm{final}(T) = \Lambda(T) - \Lambda(T_\mathrm{CMB}). \qquad (15)$$

This allows the cooling rate to smoothly approach zero as the temperature approaches T_CMB and also for the CMB to heat the gas when $T < T_\mathrm{CMB}$. We take the same approach when calculating the cooling from metals.

4.2 Metal Heating and Cooling

Next we turn to the impact of metals on the thermal evolution of the gas. Solving for the cooling from metals using a non-equilibrium network akin to that discussed in section 3.1 is computationally challenging since the number of species and reactions that must be considered rises steeply with each additional element. For this reason, `Grackle` computes the impact from metals using tables of heating and cooling rates in a way analogous to that discussed in section 4.1.2. This method was first described by Smith et al. (2008). Other packages, such as `KROME` (Grassi et al. 2014), offer the ability to perform non-equilibrium chemistry calculations including metals, but at the proportional computational cost.

The cooling or heating from metals can be added to the rate from the primordial species as calculated by either the non-equilibrium network (§4.1.1) or the tabulated solver (§4.1.2). As in the tabulated primordial cooling, the heating and cooling tables have three dimensions (z, ρ, and T) to account for the effects of the UV background models. The values stored in the tables correspond to those of Solar metallicity and the cooling rate applied to the gas is scaled by the local metallicity. All of the available metal cooling tables assume a Solar abundance pattern and consider all elements heavier than He up to atomic number 30 (Zn).

4.2.1 Constructing Cooling Tables

The `Grackle` library comes with three different model input files that can be used to calculated the tabulated cooling from primordial species and metals under different conditions. The three available models are the UV background model of Faucher-Giguère et al. (2009), that of Haardt & Madau (2012), and a model assuming no incident radiation, i.e., collisional ionization only. For the two UV background models, the input files also contain tables of photo-ionization, photo-dissociation, photo-detachment, and photo-heating rates as a function of redshift for various atomic and molecular H/He species. These are used in conjunction with the non-equilibrium primordial chemistry solver.

The cooling tables are created using the method originally described by Smith et al. (2008). Cooling, heating, and mean molecular weight values are computed using the photoionization simulation code, `Cloudy`[1] (Ferland et al. 2013). We use the `CIALoop`[2] code of Smith et al. (2008) to loop over the appropriate parameter space, call `Cloudy`, and collate the results. To expedite this process, `CIALoop` runs in

---

[1] http://nublado.org/
[2] https://bitbucket.org/brittonsmith/cloudy_cooling_tools

parallel by managing multiple instances of `Cloudy` simultaneously. To calculate the cooling and heating contribution from metals, we run each of the above models twice, once with the full complement of elements and once with only H and He. For every point in each version of the model, we extract all cooling/heating components contributing at least 10^{-10} of the total rate. We then remove all components that appear in both the full and H/He models, leaving only the contributions of the metals. All of the data are organized in `HDF5` files. The structure and discoverability of `HDF5` files allows the data to be easily used for other applications.

4.3 Dust Heating and Cooling

Dust grains transfer heat to and from a gas through collisions with the atoms and molecules in that gas. The surfaces of dust grains also provide a site for efficient formation of molecules, particularly H_2. Both heat transfer and molecular formation rates depend very sensitively on the dust temperature, T_{gr}. The dust temperature is determined by balancing the relevant heating and cooling terms. Dust grains are heated by incident radiation and cool through emission of thermal radiation. Heat flows between gas and dust in the direction of whichever has the lower temperature. The implementation of dust-related chemistry here follows very closely the work of Omukai (2000) and Omukai et al. (2005). As in these works, we currently assume that heating radiation comes only from the CMB. Future versions of the code will allow for the inclusion of additional radiative heating terms. This heat balance equation is, therefore, given by

$$4\sigma T_{gr}^4 \kappa_{gr} = \Lambda_{gas/grain} + 4\sigma T_{rad}^4 \kappa_{gr}, \qquad (16)$$

where σ is the Stefan-Boltzmann constant, T_{rad} is the radiation temperature, and κ_{gr} is the grain opacity. The left-hand side of Equation 16 represents cooling by thermal radiation and the second term on the right-hand side represents the incident radiation field characterized by a radiation temperature, T_{rad}. The dust/gas heat transfer rate per unit dust mass, $\Lambda_{gas/grain}$, is given by

$$\begin{aligned} \Lambda_{gas/grain} &= 1.2 \times 10^{-31} \; \frac{n_H^2}{\rho_{gr}} \left(\frac{T}{1000K}\right)^{1/2} (1 - 0.8e^{-75/T}) \\ &\quad (T - T_{gr}) \; \text{erg s}^{-1} \; \text{g}^{-1}, \end{aligned} \qquad (17)$$

(Hollenbach & McKee 1989), where T is the gas temperature and ρ_{gr} is the dust mass density. Currently, the dust to gas mass ratio is assumed to scale with metallicity, i.e., the dust to metal mass ratio is constant. As in Omukai (2000), we use the grain composition model of Pollack et al. (1994) where the grain mass fraction is 9.34×10^{-3} at Solar metallicity. In the future, the user will have the option to provide the dust density independently of the metal density. For the grain opacity, we use the piece-wise polynomial of Dopcke et al. (2011), which is given by

$$\kappa(T_{gr}) \propto \begin{cases} T_{gr}^2 & , T_{gr} < 200 \text{ K}, \\ \text{constant} & , 200 \text{ K} < T_{gr} < 1500 \text{ K}, \\ T_{gr}^{-12} & , T_{gr} > 1500 \text{ K}, \end{cases} \qquad (18)$$

with a normalization of $\kappa_{gr}(T_{gr} = 200 \, K) = 16 \text{ cm}^2 \text{ g}^{-1}$ (Pollack et al. 1994; Omukai 2000). The steep power-law for $T > 1500$ K is designed to mimic the effect of grains sublimating. The timescale for dust to reach thermal equilibrium is extremely short and, thus, we assume it to be in instantaneous equilibrium. We calculate the dust temperature by solving Equation 16 for T_{gr} using Newton's method. From this solution, a corresponding heating/cooling term is added to the gas following Equation 17. The dust temperature is then used to calculate the rate coefficient for H_2 formation on dust, which is a function of both the gas and dust temperatures as well as the number density of grains. The exact form of this rate is given by Omukai (2000), who derive it from that of Tielens & Hollenbach (1985). This reaction can be extremely efficient and the heating resulting from molecule formation can significantly heat the gas if the total H_2 binding energy is returned to the gas. For this reason, we also include the appropriate chemical heating term, as described in Section 4.1.1. We note that the amount of H_2 binding energy that goes toward heating the gas is highly uncertain (see e.g., Islam et al. 2010, for a brief review of theoretical and experimental efforts to determine the fraction of energy going into heating) and so we adopt this option as it is the most commonly employed. We do not account for heating of the dust grains due to H_2 formation on their surfaces, as this effect is minor compared to the other terms in Equation 16.

4.4 Constant Heating Rates

Additionally, the user has the option to supply arrays of constant heating rates that will be added to the total heating/cooling rate of each computational element due to the processes described above. These heating rates can be either volumetric (units of erg/s/cm^3) to mimic heat input from a radiation field or specific (erg/s/g), corresponding to a uniform temperature change that is independent of density.

5 RADIATION BACKGROUNDS

The Universe was reionized during the epoch of $z \sim 6 - 10$ by the buildup of radiation from stars and active galactic nuclei (AGN). This radiation heated the intergalactic medium (IGM) to $\sim 2 \times 10^4$ K (e.g. Schaye et al. 2000), inhibiting the collapse of halos with virial temperatures below this. Reproducing these effects directly in simulations requires large box sizes, extremely high resolution, as well as radiative transfer, and is, thus, prohibitively expensive. A simpler approach is to make use of a spatially uniform, redshift dependent model for the evolution of UV background radiation, such as those introduced by Haardt & Madau (1996). These models produce time/redshift dependent spectra from which photoheating and photo-chemical reaction rates can be derived. These rates can then be used in the reactions shown in Tables 3 and 4. More simply, the spectra from UV background models can be used as inputs to photoionization codes, like `Cloudy`, to calculate the heating/cooling rates as a function of density, temperature, and redshift.

As discussed in Section 4.2.1, `Grackle` makes use of data files which store tables of all relevant chemistry and cooling rates as a function of redshift for each UV background model. For the six and nine species primordial chemistry networks, we store photo-ionization heating rates for H, He, and He$^+$; photo-ionization rates for H, He, He$^+$, and H_2; photo-dissociation rates for H_2 and H_2^+; and the photo-detachment rate for H$^-$. For the tabulated cooling method, we store the

total heating and cooling rates for the primordial and metal species as well as the mean molecular weight. These tables are also functions of density and temperature as they are created under the assumption of ionization equilibrium.

Currently, two UV background models are available for use with `Grackle`. These are the models of Faucher-Giguère et al. (2009) and Haardt & Madau (2012). Data tables for new models can be created following the method described in Section 4.2.1.

5.1 Approximate Self-shielding of the UV Background

Self-shielding against the UV photoionizing background can be important in many applications. However, accounting for this effect directly requires full radiative transfer, which is often computationally infeasible. In many cosmological simulations, the UV background is commonly taken to be optically thin everywhere, which may not always be an appropriate assumption. If desired, the user may include one of three analytic self-shielding prescriptions which operate independently on each computational element. Each method stems from the analytic fits to radiative transfer simulations from Rahmati et al. (2013). They found that H self-shielding occurs at densities

$$n_{\text{H,SSh}} \sim 6.73 \times 10^{-3} \text{cm}^{-3} \left(\frac{\bar{\sigma}_\nu}{2.49 \times 10^{-18} \text{cm}^2} \right)^{-2/3}$$
$$\times T_4^{0.17} \Gamma_{-12}^{2/3} \left(\frac{f_g}{0.17} \right)^{-1/3}, \quad (19)$$

where $\bar{\sigma}_\nu$ is the gray (spectrum-averaged) absorption cross-section, $T_4 = T/10^4$ K, Γ_{-12} is the photoionization rate in units of 10^{-12} s^{-1}, and f_g is the absorber baryon fraction, which we take as $f_g = 0.17$ for simplicity. Both the ionization and photoheating rates are then attenuated due to self-shielding by a factor:

$$\frac{\Gamma_{\text{shield}}}{\Gamma_{\text{UVB}}} = 0.98(1 + x^{1.64})^{-2.28} + 0.02(1 + x)^{-0.84}, \quad (20)$$

where $x = n_H/n_{\text{H,SSh}}$.

All three available methods are various applications of equations 19 and 20. The first includes self-shielding in H only by applying these equations, leaving He and He$^+$ optically thin. The second includes self-shielding in both neutral H and neutral He using these equations, leaving He$^+$ optically thin. Finally, the third applies these equations for both neutral H and He as before, while ignoring He$^+$ photoionization/photoheating from the UV background entirely. (In other words, when accounting for self-shielding, leaving He$^+$ optically thin to the UV background may be much worse than ignoring it entirely). The latter is a common simplifying assumption in radiative transfer simulations for the H reionization epoch (but not during He reionization!) that is generally found to be a reasonable approximation (Osterbrock & Ferland 2006; McQuinn & Switzer 2010; Friedrich et al. 2012; Rahmati et al. 2013).

By default self-shielding is off; these methods should be used with care, as these equations may not be applicable in all situations. This is particularly true in regimes where the ionization rate becomes dominated by collisional ionization (Rahmati et al. 2013), as is the case at high densities or for low UV background ionizing rates. Finally, we note that no

shielding correction is applied to the metals, which can cause large errors in the cooling due to the very different predicted electron fractions (in the non-equilibrium calculation vs. the `Cloudy` metal tables). Great care must therefore be taken when using this feature with metal cooling. We note that it is, in principle, possible to address this shortcoming by generating `Cloudy` tables including shielding effects.

While equations 19 and 20 are redshift independent fitting formulae, the gray-averaged cross section, $\bar{\sigma}_\nu$, depends on the evolving spectrum of the UV background. Included in the `Grackle` data files are the pre-computed $\bar{\sigma}_\nu$ for H, He, and He$^+$ at each redshift for both the Faucher-Giguère et al. (2009) and Haardt & Madau (2012) UV background models using the frequency dependent photoionization cross sections from Verner et al. (1996).[3]

5.2 Ionization and Heating from Radiative Transfer Simulations

Although `Grackle` itself does not perform radiative transfer, it can be used with a simulation code that does. In particular, we allow for the optional inclusion of (spatially-varying) arrays of H, He, and He$^+$ photo-ionization rates, as well as a H$_2$ photo-dissociation rate, from radiative transfer calculations. Associated heating from these processes is handled through a single heating rate array. These rates are included for each computational element and are tied to the overall heating/cooling and chemistry rates. This allows the user to couple radiative transfer solutions self-consistently with the chemical reaction network. This could be done as a post-processing step, usually for cosmic reionization calculations (e.g. Iliev et al. 2014; Mesinger et al. 2016), or coupled with the hydrodynamics, which is becoming more commonplace in galaxy and star formation simulations (e.g. Wise et al. 2014; Rosdahl et al. 2015; Ocvirk et al. 2015; Baczynski et al. 2015; Pawlik et al. 2016; Rosen et al. 2016). Our current interface only connects to primordial rates, but in the future, additional connections to radiative transfer models could include a more accurate computation of (i) the photoelectric effect, or (ii) the heating and cooling rates from metals, where the local UV/X-ray flux would be an additional interpolation variable in the lookup table (e.g. Aykutalp et al. 2013).

6 IMPLEMENTATION

In this section, we discuss details of the code itself, including the available application program interfaces (APIs), code layout, and how the existing models and networks may be extended. For more detailed information, users should consult online documentation available at `https://grackle.readthedocs.org`. In addition to the documentation, the `Grackle` community maintains a mailing list where users can post questions or comments and receive help from other users and developers. More information can be found on the front page of the documentation.

---

[3] Source code containing the analytic fits given in Verner et al. (1996) was obtained from `http://www.pa.uky.edu/~verner/photo.html`

6.1 Simulation Code API

The `Grackle` library provides five main functions to the user for use in simulation codes through C or Fortran bindings: solving the chemistry and cooling (i.e., updating the chemical species and internal energy) and calculating the cooling time, temperature, pressure, and ratio of specific heats. Before these functions can be called, the code must be initialized with various user-specified settings. This initialization process is also responsible for loading data from external files and calculating the chemistry and cooling rate tables used by the solvers. All `Grackle` run-time parameters are stored within a C `struct` of type `chemistry_data`. The user initializes this structure by calling the function `set_default_chemistry_parameters` and supplying a pointer to a `chemistry_data` structure. The `chemistry_data` pointer is then attached to a globally viewable pointer called `grackle_data`, allowing all run-time parameters to be accessible without having to store the struct manually. Once all parameters are properly set, the user must call `initialize_chemistry_data` to finalize the initialization process. An example of this procedure is shown below:

```
int rval;
chemistry_data my_pars;
rval =
  set_default_chemistry_parameters(&my_pars);

grackle_data.use_grackle = 1;
grackle_data.with_radiative_cooling = 1;
grackle_data.primordial_chemistry = 3;
grackle_data.metal_cooling = 1;
grackle_data.UVbackground = 1;
grackle_data.grackle_data_file =
  "CloudyData_UVB=HM2012.h5";

rval = initialize_chemistry_data(&my_units);
```

In the above example, the variable `my_units` is a C `struct` that holds unit conversions from internal code units to the CGS unit system for quantities such as density, length, and time. These are required in order to set up the internal unit system for the chemistry and cooling rate tables.

Once `Grackle` has been initialized, the functionality described above can be called by the simulation code. The available functions are: 1) `solve_chemistry` to integrate chemistry and cooling equations over a specified time step, 2) `calculate_cooling_time` to calculate the cooling time ($e/(de/dt)$) for each computational element, 3) `calculate_temperature` to calculate the temperature from the internal energy and chemical species densities, 4) `calculate_pressure` to calculate the gas pressure, and 5) `calculate_gamma` to calculate the ratio of specific heats. In `Grackle`, the ratio of specific heats is only altered from that of a monatomic gas by the presence of H_2.

For efficiency, `Grackle`'s functions are designed to operate on multiple computational elements simultaneously. The user provides arrays of the required fields to `Grackle` and their values are updated by the chemistry and cooling solvers. Because the number of required fields depends on the specific solver being used, `Grackle` makes use of another C `struct` as a means of passing field arrays to the `Grackle` functions. The `grackle_field_data` struct contains pointers to which can be attached the arrays of density, internal energy, and any optional fields, such as individual species densities or the arrays of constant heating rates (Section 4.4). Since arrays of the optional fields are only accessed based on run-time parameter settings, the user has the option of only providing the fields they wish to use. The field arrays can be one, two, or three-dimensional, allowing both Lagrangian particle-based codes and Eulerian mesh-based codes to provide fields in their native layout. The `grackle_field_data` struct contains entries to specify the field dimensionality as well as to flag certain array elements as boundary cells which are to be ignored. An example of calling the main chemistry solver function on a 10^3 grid with 3 boundary zones on each side is shown below:

```
grackle_field_data my_fields;

my_fields.grid_rank = 3;
my_fields.grid_dimension = new int[3];
my_fields.grid_start = new int[3];
my_fields.grid_end = new int[3];
for (int i = 0;i < 3;i++) {
  my_fields.grid_dimension[i] = 10;
  my_fields.grid_start[i] = 3;
  my_fields.grid_end[i] = 12;
}

my_fields.density = density_array;
my_fields.internal_energy = energy_array;
my_fields.HI_density = HI_array;
...

// 1 Myr in internal units
double dt = 3.15e7 * 1e6 /
  my_units.time_units;

rval =
  solve_chemistry(&my_units, &my_fields, dt);
```

An added benefit of this approach is that adding new features which use additional fields will not require a change in the function signature. This will, in theory, allow `Grackle` to maintain backward compatibility indefinitely. We note that versions of `Grackle` prior to 3.0 did not make use of the `grackle_field_data` struct and instead required all fields to be provided as individual arguments. We acknowledge that the release of `Grackle` 3.0 constitutes a significant change to the API, but one that will ultimately provide more stability moving forward.

6.2 Pygrackle - Grackle's Python API

As described above, `Grackle`'s native API is provided through C and Fortran bindings. This is particularly useful for simulation codes, as they are typically written in either C or Fortran compatible languages, but it provides a barrier to entry for experimental work and testing of `Grackle` functionality. Python is a high-level, interpreted language, increasingly used in scientific computing both as "glue" code as well as a mechanism for authoring production scientific

codes. For instance, in 2016, one of the Gordon Bell Prize nominees utilized the Python code PyFR to demonstrate extreme scaling of finite element calculations (Vincent et al. 2016).

To facilitate access to `Grackle` functionality, we provide Python bindings to it, designed to make interacting with chemical rates and rate equations available to researchers at all stages. This enables rapid iteration over different rate coefficients, different initial state vectors, and over arbitrary time periods. The bindings are written in Cython (Behnel et al. 2011) with minimal overhead from Python operations. Below, we describe two particular aspects of `Pygrackle`.

6.2.1 *Fluid Container*

`Pygrackle` provides an object called a "fluid container". When data is passed to `Grackle` during the course of a simulation, it may be sent as a single zone, as multiple zones that are organized in a 3D array, or as a 1D "pencil beam" of data. The fluid container object is designed to mimic this, enabling individuals to "create" a collection of fluids either by reading them from disk or constructing them in-memory from NumPy arrays (van der Walt et al. 2011). This enables calculations of cooling time, chemical evolution, etc., from analytically-defined gas parcels and distributions. While this will not take into account hydrodynamics (unless a package spiritually similar to `Grackle`, but for hydrodynamics, is released) it can provide useful and scientifically relevant information.

The fluid container provides several high-level functions, such as computing the cooling time, the pressure, and so forth, most of which are usually used only internally in `Grackle`. Additionally, this object provides compatibility with `yt` (Turk et al. 2011a), enabling individuals to load data with `yt` and then evolve it through `Grackle`. One possible use case for this is to load in a dataset and use `Pygrackle` to compute different cooling times for a collection of gas identified in `yt` based on different metallicity assumptions, different chemical rate coefficients, and different radiation backgrounds.

6.2.2 *Evolution Models*

In addition to providing access to `Grackle`'s primary functionality, `Pygrackle` also features a set of convenience functions to evolve a fluid container forward in time following simple models. These functions return a dictionary of arrays of all fluid quantities (i.e., species densities, internal energy, temperature, etc.) for time values of the evolution. These functions can be used in semi-analytic models that require knowledge of the thermal evolution of gas under different conditions. Examples of use are discussed in Section 7.1.

The simplest of these evolves a fluid container assuming a constant density model. The `evolve_constant_density` function takes an initialized fluid container and repeatedly calls `Grackle`'s `solve_chemistry` function until a specified stopping time or temperature has been reached. For each iteration, the timestep is taken to be a fraction of the local cooling time, specifiable by the user and defaulting to 0.01.

The second of these functions models the evolution of a parcel of gas collapsing due to self-gravity. The `evolve_freefall` function closely follows the one-zone free-fall collapse model introduced by Omukai (2000) and modified by Omukai et al. (2005) to include the effects of thermal pressure support. The gas density evolves following the modified collapse model of Omukai et al. (2005), given by

$$\frac{d\rho}{dt} = \frac{\rho}{t_{\text{col}}}, \tag{21}$$

where t_{col} is the collapse time-scale expressed as

$$t_{\text{col}} = \frac{t_{\text{ff}}}{\sqrt{1-f}}, \tag{22}$$

and the free-fall time is given by

$$t_{\text{ff}} = \sqrt{\frac{3\pi}{32G\rho}}. \tag{23}$$

Thermal pressure forces, which act to slow the collapse of the cloud, are modeled by the factor f, which is expressed as

$$f = \begin{cases} 0, & \gamma < 0.83, \\ 0.6 + 2.5(\gamma - 1) - 6.0(\gamma - 1)^2, & 0.83 < \gamma < 1, \\ 1.0 + 0.2(\gamma - 4/3) - 2.9(\gamma - 4/3)^2, & \gamma > 1, \end{cases} \tag{24}$$

where the effective adiabatic index, γ, is

$$\gamma \equiv \frac{\partial \ln p}{\partial \ln \rho}. \tag{25}$$

As the density increases, the internal energy is altered by a combination of adiabatic compression and radiative cooling, computed by `Grackle`, and is given by

$$\frac{de}{dt} = -p\frac{d}{dt}\frac{1}{\rho} - \Lambda, \tag{26}$$

where the pressure is given by

$$p = \frac{\rho kT}{\mu m_{\text{H}}}, \tag{27}$$

the specific internal energy is given by Equation 12, and Λ is the radiative cooling rate. For each iteration of this function, the timestep is taken to be a fraction of the local free-fall time, specifiable by the user and defaulting to 0.01. An example use of this function is shown in Section 7.1.5.

6.3 Adding New Models/Rates

Adding new rates to `Grackle` involves modifying the values stored in the rate coefficients and additionally defining a new network for both the chemistry and if desired also the cooling. As it stands, some direct modification of the code structure is required (although see §8.3 for future improvements in this area), and we give more details of that in this section.

The structure of the current code base is set up to realize either an equilibrium chemistry model using cooling tables derived from `Cloudy`, or a non-equilibrium cooling model based on a six, nine or twelve species chemistry model. To implement a different equilibrium cooling model, the process is relatively straightforward: `Grackle` reads the `Cloudy` cooling and interpolates the data along one, two or three dimensions as appropriate. By substituting in a different cooling table, with the correct format, the behavior of the cooling for a given species can be easily changed.

To modify the non-equilibrium cooling behavior, more work is required. `Grackle` assumes that non-equilibrium chemistry models are hierarchical, with the nine species model composed of the six species model plus three additional species and likewise for the twelve species model. To expand the network to include, say, a fourth model containing the twelve species model as a subset then the procedure is straightforward. However, if a different chemical network is envisaged containing some species already in the twelve species model and some not, then this will require modifying the existing hierarchical structure. The next sections provide some details of this process.

6.3.1   *Updating the rate coefficients*

The rates are allocated and declared in the `initialize_chemistry` function. To update the rate coefficients (e.g., based on more recent experimental data), simply make use of the already existing rate coefficient arrays (`k` array) as declared in `initialize_chemistry`. The rates are populated in `calc_rates_g`. Within this function the rates are fully described. For example, `k1` is the rate coefficient for the collisional ionization of neutral hydrogen by electrons (`k1`: $H + e^- \rightarrow H^+ + 2e^-$). If a significant number of new rate coefficients are needed then the most expedient approach would be to insert a preprocessor directive into `calc_rates_g` in which the appropriate function call can be inserted. For any additional rate coefficients that are required, the corresponding `k` array needs to be declared and allocated in `initialize_chemistry`. Furthermore, the interpolation of the new rates will need to be added in the function `lookup_cool_rates1d`.

6.3.2   *Updating the chemistry network*

To update the chemical network to either include or exclude reactions, a new rate network will be required. As a template, the function `step_rate` can be used. If the new network is simply an addition to the existing network (e.g., a 15 species model) then the easiest option is to simply augment this network with the three extra species using the appropriate interactions. The species will then be evolved until they converge. More complex additions would require creating a new network update routine using `step_rate` as a template.

6.3.3   *Updating the cooling model*

Apart from the chemical network, the cooling model may also be modified. As discussed previously, if the intention is to implement a new cooling table then the changes are straightforward. For changes to the equilibrium network (say, to modify the cooling rate due to, for example, emission line cooling from a given species), this is handled in `cool1d_multi_g`. As discussed in §4, line emission cooling is determined using collisional excitation, collisional ionization and recombination rates. If the intention is to update/modify existing rates then the cooling rates are also set in `calc_rates_g` and can be modified there. If new cooling rates are required from another species whose cooling properties are important for the network then the rates can be added there also. Any new arrays required in

this case will also need to be declared and initialized in `initialize_chemistry` in a way similar to the rate coefficients. The rates can then be interpolated to the required temperature values in `calc_rates_g`, following the examples there. Finally, once the new cooling rate has been determined its values needs to be added to the `edot` array, which tracks the total cooling/heating rate.

7   PROFILING AND TESTING

7.1   Testing Framework

Testing a library like `Grackle`, with its mix of `Fortran`, and `C++` code, can be difficult. In the interest of ultimately improving test coverage and making it easier to prototype tests, `Grackle` employs a Python-based testing framework that makes heavy use of the `Pygrackle` Python wrapper for `Grackle`. The tests are orchestrated using the `py.test`[4] package.

Currently there are two different types of tests in the `Grackle` test suite: unit tests and answer tests. A unit test in `Grackle` compares the results of a calculation using `Grackle` to some set of "correct" answers that are known *a priori*. The unit tests currently implemented in `Grackle` check that the unit system is behaving correctly (see Section 7.1.1) and that the ionization equilibrium for a primordial gas agrees with the analytic prediction using the rates implemented in `Grackle` (see Section 7.1.2).

The answer tests consist of a set of example calculations where each calculation writes out a summary plot as well as an `HDF5` dataset that is loadable by the `yt` package. Known "correct" answers for the summary plot and `yt`-loadable dataset are saved in the repository so that code changes in `Grackle` can be tested to ensure that answers produced by the library do not change. This process does not prevent *incorrect* answers from being generated initially, but it does notify the developer if answers change as a result of a code modification. Incorrect answers are prevented by manually inspecting test answers when the test is first added to the codebase. If subsequently a bug is discovered (or an enhancement to the code is made) and the test output changes, then the test answer must also be manually updated. Currently, `Grackle` contains stored answers for: the instantaneous cooling rate (see Section 7.1.3) at a constant density, the temperature evolution of a uniform-density cloud (see Section 7.1.4), and the density and temperature evolution of a gas cloud undergoing free-fall collapse (see Section 7.1.5). The answer tests are run several times using different input physics to ensure `Grackle`'s solvers are well-exercised by the tests. The answer tests are presented as sample scripts that can be run manually by the user, producing a figure. For each answer test, we show the corresponding figure exactly as produced by each script.

In addition to the unit and answer tests, which monitor functionality of the solvers, we also employ tests to ensure that all Python source code conforms to PEP 8 style standards[5] and that all instructional sample codes compile and run without producing errors.

---

[4] http://pytest.org/
[5] https://www.python.org/dev/peps/pep-0008/

7.1.1 Unit Test: Unit Systems

For a given set of physical conditions, (i.e., densities and internal energies), the results of `Grackle`-related calculations should be independent of the choice of reference frame (comoving or proper) and internal unit system. However, because chemistry and cooling calculations involve numerical values that span many orders of magnitude, round-off error will eventually lead to significant differences when the internal unit systems are varied beyond a certain degree. The unit systems unit tests set up two fluid container objects with the same physical conditions but different internal unit systems. In each instance, the chemical species fractions are evolved until ionization equilibrium has been reached (see §6.2.1), after which time the cooling time is calculated. The tests assert that the cooling time values agree to within 4 decimal places between the two unit systems.

Three variants of this unit test exist. In the first two, a comoving-frame unit system appropriate for a cosmological simulation is compared with a proper-frame unit system drawn from a random number generator that allows the density, time, and length units to vary by 4 orders of magnitude. A cosmologically appropriate unit system is roughly defined as one with density units equal to the average comoving matter density of the Universe, $\bar{\rho}_m$; time units proportional to $1/\sqrt{G\,\bar{\rho}_m}$; and length units on the scale of Mpc. One of the two of these type is performed with the non-equilibrium chemistry solver and the other with the fully tabulated solver. In both cases, we compare a randomly generated proper-frame unit system with a comoving-frame unit system at $z = 0$ and $z > 0$, where the proper and comoving frames differ. In these tests, a UV background model is also used as the radiative heating rate is proportional to ρ, whereas collisional heating/cooling rates are proportional to ρ^2 (or ρ^3 for three-body reactions). Including heating/cooling terms with different density scaling is useful for exposing errors in adjusting between comoving and proper reference frames.

The final variant of the unit system test compares two randomly generated proper-frame unit systems whose density units differ by as much as possible while maintaining equivalency to 4 decimal places. In practice, we find that the density units can differ by roughly 31 orders of magnitude before the threshold level of accuracy is lost. With this in mind, we allow the two randomly generated unit systems to span 2 orders of magnitude with the center of each random distribution chosen such that the unit system will differ by 27-31 orders of magnitude. By comparing unit systems that differ by the maximally allowed amount, we are able to measure the degree to which new terms added to the network suffer from round-off error.

7.1.2 Unit Test: Collisional Ionization Equilibrium

The equilibrium ionization state of a gas is determined solely by its temperature when only collisional ionization is considered, (i.e., when photo-ionization is neglected). Thus, a density-independent equilibrium solution can be calculated for any ion by equating the creation terms (recombination from a higher ionization state and ionization from a lower state) with the destruction terms (recombination into a lower state and ionization into a higher state). For exam-
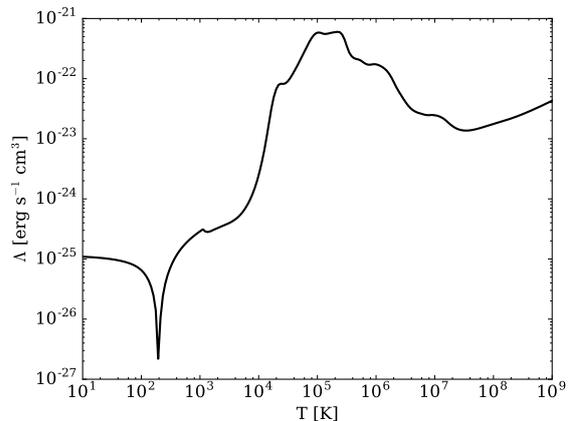
Figure 2. Figure output by the default configuration of the cooling rate answer test, described in Section 7.1.3, showing the equilibrium cooling rate as a function of temperature for a Solar metallicity gas at a density of 1 amu/cm^3 with an incident radiation field described by the Haardt & Madau (2012) model at z = 0. The absolute value of the cooling rate is shown (in order to use a log scale) because below ~ 200 K, the radiation field induces a net heating rate.

ple, this yields a CIE solution for neutral H given by

$$f_{\mathrm{H,CIE}} = \frac{\alpha_{\mathrm{H^+}}(T)}{\alpha_{\mathrm{H^+}}(T) + \Gamma_{\mathrm{H}}(T)}, \tag{28}$$

where $\alpha_{\mathrm{H^+}}$ is the recombination rate of H$^+$ and Γ_{H} is the collisional ionization rate of H.

In order to test that `Grackle` arrives at the correct CIE solution for the atomic primordial network, we initialize a fluid container at a constant density with temperature varying smoothly in log-space from 10^4 K to 10^9 K. The gas is initialized in a fully neutral state and the chemistry solver is called repeatedly with cooling processes deactivated (to keep each cell at its original temperature) until convergence has been reached in all cells. These values are then compared to the analytical solutions (as in Equation 28) for the ionization states of all H and He species and the total electron fraction.

7.1.3 Answer test: cooling rate test

Similar to the CIE unit test (Section 7.1.2), the cooling rate answer test initializes a fluid container with a constant density and smoothly varying temperature from 10 K to 10^9 K, then iterates the chemistry solver until all species have reached equilibrium. Unlike the CIE unit test, metal cooling and a Haardt & Madau (2012) UV background at $z = 0$ are also included. After reaching equilibrium, the total cooling rate is calculated and compared with stored answers, as described in Section 7.1 above. This test is performed for all versions of the primordial solver as well as the fully tabulated cooling solver. The figure produced by the default configuration of this test (H, D, He non-equilibrium solver) is shown in Figure 2.
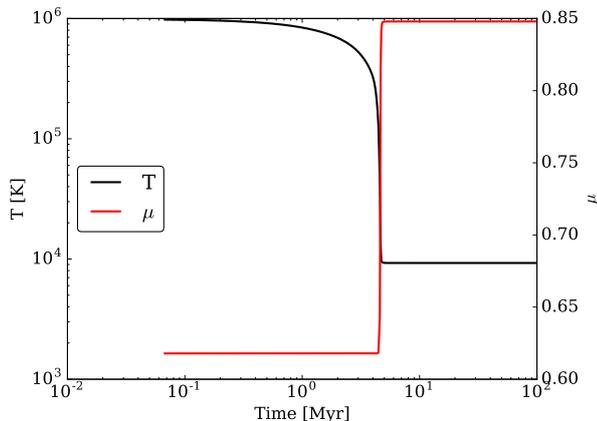
Figure 3. Figure output by the uniform cooling answer test, described in Section 7.1.4, showing the temperature (black) and mean molecular weight (red) as a function of time for a parcel of gas cooling at constant density.
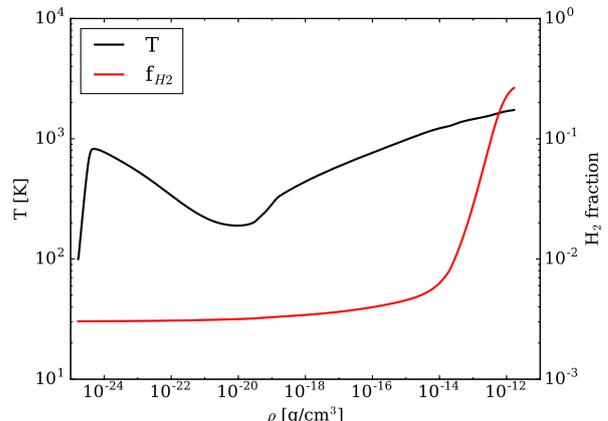

Figure 4. Figure output by the default configuration of the free-fall answer test, described in Section 7.1.5, showing the temperature (black) and H$_2$ fraction (red) as a function of density for a free-fall collapse model of a metal-free gas.

7.1.4   Answer test: constant density cooling test

The uniform cooling answer test simulates the thermal evolution of a parcel of gas cooling at constant density. This test ensures that the solver properly evolves the internal energy over a period of time. The test initializes a single-cell fluid container with a density of 0.1 amu/cm^3 and a temperature of 10^6 K. The cell is evolved for 100 Myr using the `evolve_constant_density` convenience function with timesteps of 1% of the cooling time. The resulting evolution is shown in Figure 3.

7.1.5   Answer test: free-fall collapse test

The free-fall collapse answer test simulates the thermal evolution of a cloud of gas collapsing due to self-gravity. This test is useful for ensuring that the non-equilibrium chemistry solver is functioning properly over a large range in density. The test creates a single-cell fluid container with an initial density of 0.1 amu/cm^3 and an initial temperature of 50,000 K. Before beginning the free-fall collapse phase, the cloud is allowed to cool at a constant density to a temperature of 100 K using the `evolve_constant_density` function described in Section 6.2.2. This allows the gas to settle into an ionization state that is roughly appropriate for the temperature. From there, we evolve the density of the cloud using the `evolve_freefall` function, also discussed in Section 6.2.2. This test is performed using the full non-equilibrium chemistry solver, once at zero metallicity (Figure 4) and once with a metallicity of 10^{-3} Z$_\odot$.

7.2   Performance

7.2.1   Optimization Strategy

Our optimization strategy in the `Grackle` has two components related to serial and parallel execution. We begin with single processor optimization.

The ordinary differential equations that describe chemical and thermal evolution do not use spatial information and so each discretization point (particle or cell) can be evolved independently of the others. Because of this, the `Grackle` can be used with a wide variety of codes and applications, and optimization is relatively straightforward. Our primary technique for single processor optimization is to make good use of cache and (single processor) vectorization. The API is built around the idea of "fields" of points (fluid containers) rather than a single point for this reason. The field can be a single-dimensional contiguous list as might be used for particle-based codes, or a three dimensional grid with inactive ("ghost") points as would be appropriate for grid-based codes. By taking an entire field, and operating on the field in an order corresponding to the way it is laid out in memory, the code tries to minimize cache misses. In particular, multi-dimensional arrays are assumed to be Fortran-ordered (column-major) and operations are performed in loops over the most rapidly varying index. Loops are generally also written in a way which facilitates unrolling so that compilers can easily make use of vector operations and prefetching. Much of the computationally intensive part of the code is written in Fortran (in part for historical reasons but in part to take advantage of well-tested Fortran compiler loop optimizations).

The second optimization involves taking advantage of parallel computation. `Grackle` itself requires no communication (and is completely thread-safe after the initialisation step) and so can easily work as part of a code that uses MPI or some other message passing library to achieve distributed parallelization. In addition, `Grackle` supports OpenMP parallelization and thus can easily work with applications adopting a hybrid MPI/OpenMP model. The OpenMP is implemented by parallelizing over outer j,k loops and giving a thread an i "slice" to operate on. This is a natural model for structured grid-based codes, but some work may be required to get good performance this way with unstructured or particle based codes (e.g., by artificially splitting a 1D list of particle quantities into a 2D grid).
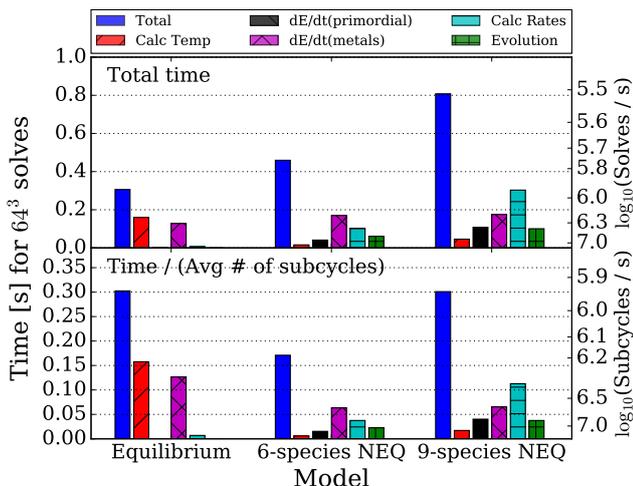
Figure 5. Top panel: Total time to compute the cooling rate test (§7.1.3) in 64^3 fluid containers, using the tabulated equilibrium cooling model (left), six species non-equilibrium model (middle), and nine species non-equilibrium model (right). The different bars show the time needed for the complete solve (blue), the temperature calculation (red), the $\dot{e}$ computation due to primordial chemistry (black) and metals (magneta), the interpolation of rate coefficients (cyan), and the update of the species fractions with backward differencing. Bottom panel: Total time but normalized by the average number of subcycles per cell, which demonstrates the performance of a single iteration in each solver mode.

7.2.2 Serial Performance

We utilize the cooling rate test (§7.1.3) to assess the performance of Grackle. We set up the test with 64^3 fluid containers with hydrogen number density n_H, temperature T, and metallicity Z varying in each dimension. These quantities are equally log-spaced in the range $\log(n_H/\mathrm{cm}^{-3}) = [-1,3]$, $\log(T/\mathrm{K}) = [1,8]$, and $\log(Z/Z_\odot) = [-4,0]$. All of the fluid containers are initially neutral. We run each test with the tabulated solver in ionization equilibrium and the non-equilibrium solver with the six species and nine species networks on a single core of a desktop computer with dual Intel Xeon "Westmere" E5645 CPUs at 2.4 GHz, each of which has six cores. The tests are evolved for 500 yr. In most cases, this is shorter than the cooling time, but it provides an ample test for the performance of Grackle. Because the fluid containers are not initialized in ionization equilibrium, the first timestep in the non-equilibrium solvers requires hundreds of subcycles for the system to converge. Due to the fact that the non-equilibrium solver performance is directly tied to the number of subcycles, we do not include the first three cycles of the tests (which are not representative of typical use conditions).

The top panel of Figure 5 shows the total time (blue bars) required for this 64^3 performance test in each solver mode. This is further divided into the time spent in each major component of Grackle: the calculation of the temperature (red), $\dot{e}$ from primordial (black) and metal (magenta) species, rate coefficient interpolation (cyan), and the update of the species fractions (green). From a total performance standpoint, the non-equilibrium solver in the six species and nine species primordial models requires 50% and 164% more time than the equilibrium solver, respectively. In

this test, Grackle can update 8.6×10^5, 5.7×10^5, and 3.2×10^5 fluid containers per second for the equilibrium, six species and nine species solvers, respectively. The computational expense in the equilibrium solver is almost evenly split between the equilibrium temperature calculation and metal cooling rates. The metal cooling rate and rate coefficient interpolation consume the most compute cycles in the six species and nine species solvers, respectively. The temperature calculation in the non-equilibrium solver takes relatively little computation because it is simply calculated from the pressure and total number density with no iterative processes.

The cooling rate test represents a fluid in many different chemo-thermal states, which converge to some equilibrium. However in production simulations, there are many "difficult" situations, such as high densities, strong shocks, and strong radiation fields, in which the equations become stiff and require many subcycles to complete an entire solve. Therefore to better gauge the performance of a single iteration, we show the average time elapsed *per subcycle* for the same test in the bottom panel of Figure 5. The equilibrium and non-equilibrium solvers take an average of 1.01 and 2.67 subcycles per solve, respectively. Grackle can perform 8.7×10^5, 1.5×10^6, and 8.7×10^5 subcycles per second for the equilibrium, six species and nine species solvers, respectively. Here we see that the equilibrium solver actually requires 75% more time per subcycle than the six species non-equilibrium solver and is equivalent to the performance (per subcycle) of the nine species non-equilibrium solver. In practice, if any cells require many subcycles to converge to a solution, the call to Grackle will require more time per cell than in this ideal test, because the total performance is entirely dependent on the total number of subcycles performed in one solve, not the number of cells.

7.2.3 OpenMP Performance

In addition to the single-processor performance just described, we characterize the threaded performance of the Grackle. Figure 6 shows an OpenMP performance benchmark for both the non-equilibrium and tabulated functionality, where parallel efficiency is defined as the ratio of multi-thread to single-thread performance. We conduct this benchmark on the Campus Cluster of the University of Illinois at Urbana-Champaign using 20 threads on two Intel Xeon E5-2670 v2 CPUs at 2.50 GHz, each of which has ten cores. We compile Grackle using version 15.0.0 of the Intel compilers with "-O3" optimization. For all time-consuming routines (i.e., calculating cooling, cooling time, and temperature with the tabulated solver, and calculating chemistry, cooling, and cooling time with the non-equilibrium chemistry solver), the parallel efficiency reaches $\sim 60\%-90\%$ for 16^3 cells and $\sim 80\%-90\%$ for 64^3 cells. For other computationally cheap routines, such as calculating pressure, the parallel efficiency is relatively low. This is not an issue since they take negligible time compared to other computationally expensive routines.
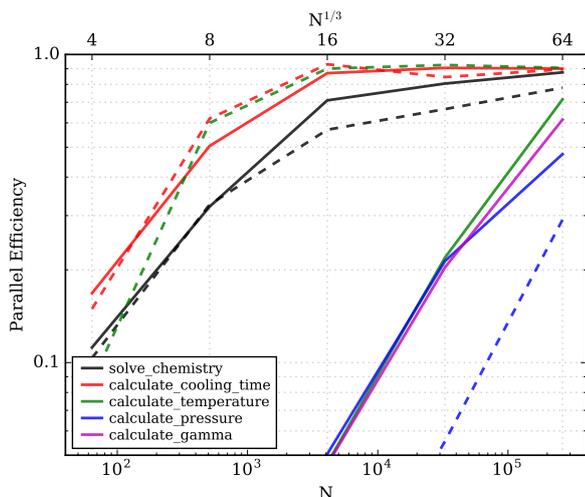
Figure 6. OpenMP parallel efficiency using 20 threads as a function of the size of the input array. The solid lines show the use of the non-equilibrium solver with `primordial_chemistry =` 3 and the dashed lines show the analogous functions using the tabulated solver. For all time-consuming routines, the parallel efficiency reaches ~60% to 90% for 16^3 cells and ~80% to 90% for 64^3 cells.

8 DISCUSSION

8.1 Applicability and Limitations

It is important to remember that the range of physical conditions over which `Grackle` can be considered valid is not unlimited. In Figure 7, we show the rough density range over which different components of the solver are valid. The high density limit of the non-equilibrium solver is set roughly by the reactions present in the network and the range over which their rate coefficients are trusted. The limits on the cooling tables correspond to the density range over which they were calculated. Users are especially cautioned against exceeding the upper density limit of the tabulated cooling solver. The critical density above which energy levels are populated according to LTE exceeds the upper limit of the tables for many metal coolants (Smith et al. 2008). Thus, these tables do not capture the NLTE to LTE transition where the cooling rates change from scaling as ρ^2 to ρ. Hence, extrapolation beyond the upper limit may result in vast over-prediction of the cooling rate. If the use-case requires exceeding this limit, then the high density metal table should be used in conjunction with the non-equilibrium solver. In all cases, the valid temperature range is roughly 1 K to 10^9 K. The tables computed with `Cloudy` are defined over this temperature range. For the primordial chemistry, we note that the reaction rates are defined over this temperature range, if not explicitly valid. However, at all temperatures, the rates describing the dominant processes are valid. It is also extremely important to remember that all `Grackle` calculations are based on the assumption that the medium is optically thin. In practice, the length scale of optical thickness will become very short as density increases.

The addition of radiative cooling to a simulation creates another relevant length scale which must be kept in mind. The cooling length, defined as the product of the local sound speed and the cooling time, sets the approximate size of ob-
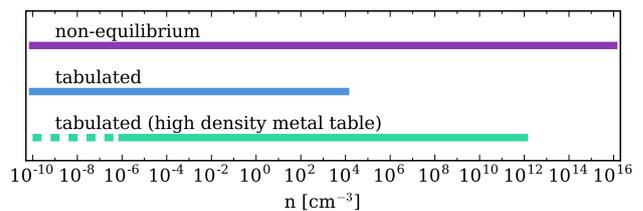
Figure 7. The appropriate density range for different versions of the `Grackle` solver. The high density metal cooling table (bottom) explicitly spans the density range, 10^{-6} cm$^{-3} < n < 10^{12}$ cm^{-3}, but extrapolation down to $n = 10^{-10}$ cm^{-3} is still valid, as indicated by the dashed line. For each of these, the valid temperature range is roughly 1 K to 10^9 K.

jects as they cool and condense (Iwasaki & Tsuribe 2009). The cooling length is inversely proportional to density, effectively setting a density limit for any given spatial resolution. When this scale becomes unresolved, radiative losses will be overpredicted, leading to unphysically high densities and further exacerbating the resolution problem in a runaway cycle. This effect is likely related to the over-cooling problem that has classically plagued cosmological simulations (e.g. Katz et al. 1996; Balogh et al. 2001). In Figure 8, we show an estimate of the cooling length for the scenario of a gas at Solar metallicity in a Haardt & Madau (2012) radiation background at z = 0, noting how quickly the cooling length drops below 1 kpc, and even 1 pc, for densities relevant to galaxy formation simulations. This length scale should be taken into consideration when choosing the density threshold above which sub-grid models are applied.

8.2 Simulation codes with Grackle

To date, the following codes are known to have `Grackle` implemented:

- AREPO (Springel 2010)
- ART-I (Kravtsov 1999; Kravtsov et al. 2002)
- ART-II (Rudd et al. 2008)
- CHANGA (Wadsley et al. 2004; Stinson et al. 2006)
- Cosmos++ (Anninos et al. 2003, 2005)
- Enzo (Bryan et al. 2014)
- Gadget-3 (Springel 2005)
- GAMER (Schive et al. 2010)
- GASOLINE (Wadsley et al. 2004)
- Gear (Revaz & Jablonka 2012b,a)
- Gizmo (Hopkins 2015)
- RAMSES (Teyssier 2002)
- SPHS (Read & Hayfield 2012)
- SWIFT (Gonnet et al. 2013; Schaller et al. 2016)

8.3 Future Directions

8.3.1 Including new rates and models in Grackle

The current code structure is highly integrated. This makes introducing new rates for the chemical network or cooling function a rather intricate task requiring multiple changes throughout the code. Apart from the fact that this is more time consuming it is also much more error prone. In a future
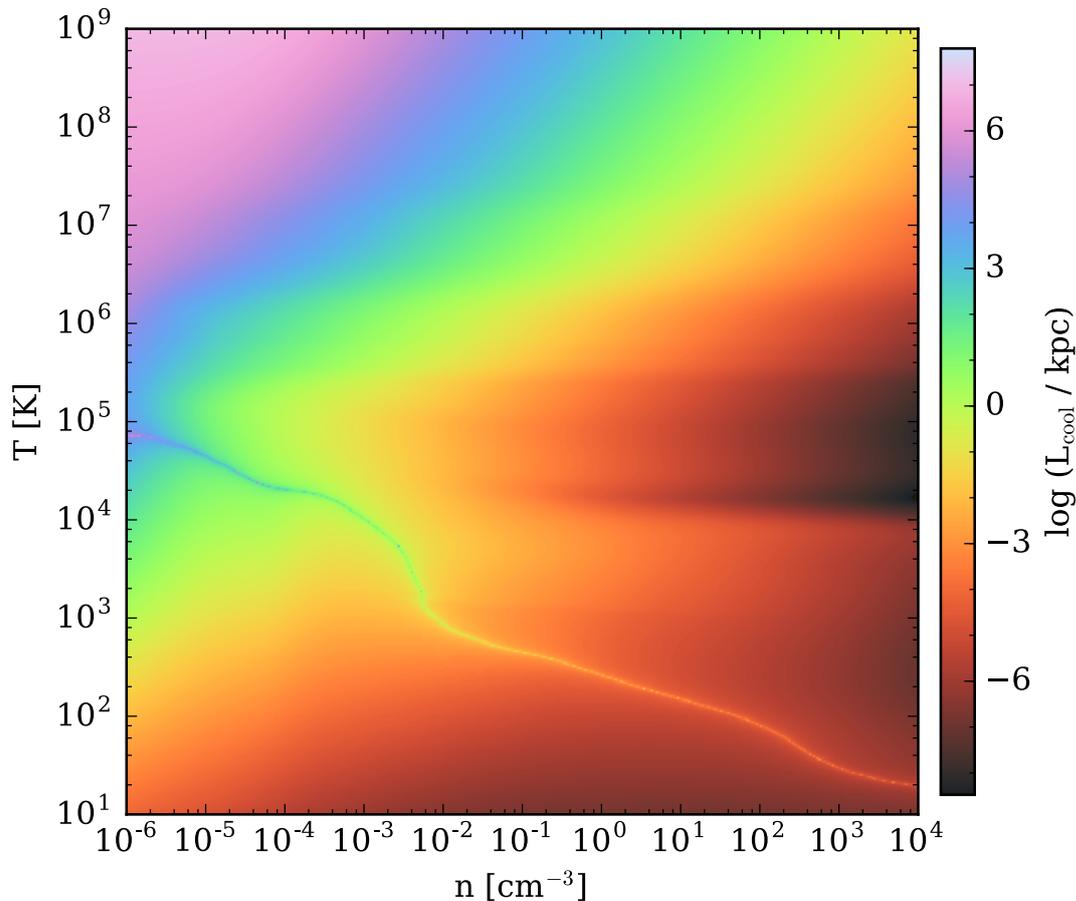
Figure 8. The cooling length, defined as the product of the sound speed and the cooling time, as a function of number density and temperature for a gas with Solar metallicity exposed to a radiation field defined by the model of Haardt & Madau (2012) at z = 0. The narrow line extending from the middle, left to the bottom, right represents the temperature where heating and cooling are balanced. Above this line, the gas is being cooled while below the line it is being heated.

release of the code the modularity of the code will be greatly increased. There will be a function to populate the species rate coefficients and a function to populate the cooling rate coefficients. Separate template files can then be updated by a developer wishing to use their own rates. This file can then be included in the build and a flag set to indicate that the new rates should be used in place of the old rates. Furthermore, a similar method will be implemented for solving the network. A template network solver will be available which the developer can use to implement a new network with a developer-determined number of species. The developer will be responsible for updating only three files to achieve a solution to their own chemical network.

8.3.2   Multiple element cooling

Currently, `Grackle` only considers a single metallicity field for the calculation of the cooling due to heavy elements. However, more sophisticated feedback models now consider feedback from multiple sources, like Type Ia and Type II supernova and winds from AGB stars, each of which produce distinct abundance patterns. In the future, we will look to create additional cooling tables that consider varying

abundance patterns. As an intermediate step before creating cooling tables for every metal species, as in Wiersma et al. (2009), we will likely begin with a two-element model distinguishing between type Ia and II supernovae, such as that of De Rijcke et al. (2013).

8.4   Summary

In this paper, we have described an open-source chemistry and radiative cooling/heating library suitable for use in numerical astrophysical simulations. `Grackle` includes a number of non-equilibrium chemistry and cooling models involving H, D and He, including H_2 formation and a simple dust model. In addition, the library has the ability to compute equilibrium cooling/heating rates for primordial and metal-enriched gas, with a number of radiative backgrounds. The sophistication of the primordial chemistry network makes `Grackle` ideally suited for detailed studies of the chemistry of metal-free gas. Although `Grackle` does not explicitly follow chemical reactions for elements heavier than He, the tabulated metal cooling rates allow the code to be employed in all situations where only the total cooling rate is needed. The library has an API suitable for calling from C, C++,

FORTRAN and Python. This paper describes the physical processes included, the implementation of the models, as well as our open development and testing framework which allows users/developers to add to the code in a scalable way that is also intended to minimize new errors. We describe the optimization and parallelization strategy, along with performance benchmarks. `Grackle` is well-tested and already used in a substantial number of high-performance numerical simulation codes.

REFERENCES

Abel T., Anninos P., Zhang Y., Norman M. L., 1997, New Astron., 2, 181

Abel T., Bryan G. L., Norman M. L., 2002, Science, 295, 93
Aldrovandi S. M. V., Pequignot D., 1973, A&A, 25, 137
Anninos P., Zhang Y., Abel T., Norman M. L., 1997, New Astron., 2, 209
Anninos P., Fragile P. C., Murray S. D., 2003, ApJS, 147, 177
Anninos P., Fragile P. C., Salmonson J. D., 2005, ApJ, 635, 723
Aykutalp A., Wise J. H., Meijerink R., Spaans M., 2013, ApJ, 771, 50
Baczynski C., Glover S. C. O., Klessen R. S., 2015, MNRAS, 454, 380
Balogh M. L., Pearce F. R., Bower R. G., Kay S. T., 2001, MNRAS, 326, 1228
Barrett P., Hunter J., Miller J. T., Hsu J.-C., Greenfield P., 2005, in Shopbell P., Britton M., Ebert R., eds, Astronomical Society of the Pacific Conference Series Vol. 347, Astronomical Data Analysis Software and Systems XIV. p. 91
Behnel S., Bradshaw R., Citro C., Dalcin L., Seljebotn D., Smith K., 2011, Computing in Science Engineering, 13, 31
Black J. H., 1981, MNRAS, 197, 553
Bryan G. L., et al., 2014, ApJS, 211, 19
Burgess A., Seaton M. J., 1960, MNRAS, 121, 471
Cen R., 1992, ApJS, 78, 341
Cen R., Fang T., 2006, ApJ, 650, 573
Clark P. C., Glover S. C. O., Klessen R. S., Bromm V., 2011, ApJ, 727, 110
Cohen N., Westberg K. R., 1983, Journal of Physical and Chemical Reference Data, 12, 531
Coppola C. M., Lodi L., Tennyson J., 2011, MNRAS, 415, 487
Croft H., Dickinson A. S., Gadea F. X., 1999, MNRAS, 304, 327
Dalgarno A., Lepp S., 1987, in Vardya M. S., Tarafdar S. P., eds, IAU Symposium Vol. 120, Astrochemistry. pp 109–118
De Rijcke S., Schroyen J., Vandenbroucke B., Jachowicz N., Decroos J., Cloet-Osselaer A., Koleva M., 2013, MNRAS, 433, 3005
Dopcke G., Glover S. C. O., Clark P. C., Klessen R. S., 2011, ApJ, 729, L3
Faucher-Giguère C.-A., Lidz A., Zaldarriaga M., Hernquist L., 2009, ApJ, 703, 1416
Ferland G. J., Peterson B. M., Horne K., Welsh W. F., Nahar S. N., 1992, ApJ, 387, 95
Ferland G. J., et al., 2013, Rev. Mex. Astron. Astrofis., 49, 137
Flower D. R., Harris G. J., 2007, MNRAS, 377, 705
Forrey R. C., 2013, ApJ, 773, L25
Friedrich M. M., Mellema G., Iliev I. T., Shapiro P. R., 2012, MNRAS, 421, 2232
Fussen D., Kubach C., 1986, Journal of Physics B Atomic Molecular Physics, 19, L31
Galli D., Palla F., 1998, A&A, 335, 403
Galli D., Palla F., 2002, Planet. Space Sci., 50, 1197
Gealy M. W., van Zyl B., 1987, Phys. Rev. A, 36, 3100
Gerlich D., 1982, in Lindinger W., Howorka F., Maerk T. D., Egger F., eds, Symposium on Atomic and Surface Phys., 1982 Contrib.. pp 304–309
Glover S., 2008, in O'Shea B. W., Heger A., eds, American Institute of Physics Conference Series Vol. 990, First Stars III. pp 25–29 (arXiv:0708.3086), doi:10.1063/1.2905558
Glover S. C. O., 2015, MNRAS, 451, 2082
Glover S. C. O., Abel T., 2008, MNRAS, 388, 1627
Gonnet P., Schaller M., Theuns T., Chalk A. B. G., 2013, ArXiv e-prints:1309.3783,
Grassi T., Bovino S., Schleicher D. R. G., Prieto J., Seifried D., Simoncini E., Gianturco F. A., 2014, MNRAS, 439, 2386
Haardt F., Madau P., 1996, ApJ, 461, 20
Haardt F., Madau P., 2012, ApJ, 746, 125
Ham D. O., Trainor D. W., Kaufman F., 1970, J. Chem. Phys., 53, 4395
Hindmarsh A. C., 1983, in Stepleman R. S., et al., eds, Scientific Computing. North-Holland, Amsterdam, pp 55–64

Hollenbach D., McKee C. F., 1979, ApJS, 41, 555

Hollenbach D., McKee C. F., 1989, ApJ, 342, 306

Honvault P., Jorfi M., González-Lezana T., Faure A., Pagani L., 2011, Physical Review Letters, 107, 023201

Honvault P., Jorfi M., González-Lezana T., Faure A., Pagani L., 2012, Physical Review Letters, 108, 109903

Hopkins P. F., 2015, MNRAS, 450, 53

Hui L., Gnedin N. Y., 1997, MNRAS, 292, 27

Hummels C. B., Bryan G. L., Smith B. D., Turk M. J., 2013, MNRAS, 430, 1548

Igarashi A., Lin C. D., 1999, Physical Review Letters, 83, 4041

Iliev I. T., Mellema G., Ahn K., Shapiro P. R., Mao Y., Pen U.-L., 2014, MNRAS, 439, 725

Islam F., Cecchi-Pestellini C., Viti S., Casu S., 2010, ApJ, 725, 1111

Iwasaki K., Tsuribe T., 2009, A&A, 508, 725

Jacobs T. A., Giedt R. R., Cohen N., 1967, J. Chem. Phys., 47, 54

Janev R. K., Langer W. D., Evans K., 1987, Elementary processes in Hydrogen-Helium plasmas - Cross sections and reaction rate coefficients. Springer

Johnson J. L., Bromm V., 2006, MNRAS, 366, 247

Karpas Z., Anicich V., Huntress W. T., 1979, J. Chem. Phys., 70, 2877

Katz N., Weinberg D. H., Hernquist L., 1996, ApJS, 105, 19

Kim J.-h., et al., 2014, ApJS, 210, 14

Kravtsov A. V., 1999, PhD thesis, NEW MEXICO STATE UNIVERSITY

Kravtsov A. V., Klypin A., Hoffman Y., 2002, ApJ, 571, 563

Kreckel H., Bruhns H., Čížek M., Glover S. C. O., Miller K. A., Urbain X., Savin D. W., 2010, Science, 329, 69

Krstić P. S., 2002, Phys. Rev. A, 66, 042717

Latif M. A., Bovino S., Van Borm C., Grassi T., Schleicher D. R. G., Spaans M., 2014, MNRAS, 443, 1979

Latif M. A., Bovino S., Grassi T., Schleicher D. R. G., Spaans M., 2015, MNRAS, 446, 3163

Lenzuni P., Chernoff D. F., Salpeter E. E., 1991, ApJS, 76, 759

Lepp S., Shull J. M., 1983, ApJ, 270, 578

Madau P., Efstathiou G., 1999, ApJ, 517, L9

Martin P. G., Schwarz D. H., Mandy M. E., 1996, ApJ, 461, 265

McGreer I. D., Bryan G. L., 2008, ApJ, 685, 8

McKee C. F., Ostriker J. P., 1977, ApJ, 218, 148

McQuinn M., Switzer E. R., 2010, MNRAS, 408, 1945

Mesinger A., Greig B., Sobacchi E., 2016, MNRAS, 459, 2342

Mielke S. L., Peterson K. A., Schwenke D. W., Garrett B. C., Truhlar D. G., Michael J. V., Su M.-C., Sutherland J. W., 2003, Physical Review Letters, 91, 063201

Miller K. A., et al., 2012, Phys. Rev. A, 86, 032714

Ocvirk P., et al., 2015, preprint, (arXiv:1511.00011)

Omukai K., 2000, ApJ, 534, 809

Omukai K., Nishi R., 1998, ApJ, 508, 141

Omukai K., Tsuribe T., Schneider R., Ferrara A., 2005, ApJ, 626, 627

Oppenheimer B. D., Schaye J., 2013, MNRAS, 434, 1043

Orel A. E., 1987, J. Chem. Phys., 87, 314

Osterbrock D. E., 1989, Astrophysics of gaseous nebulae and active galactic nuclei. University Science Books

Osterbrock D. E., Ferland G. J., 2006, Astrophysics of gaseous nebulae and active galactic nuclei

Palla F., Salpeter E. E., Stahler S. W., 1983, ApJ, 271, 632

Pawlik A. H., Rahmati A., Schaye J., Jeon M., Dalla Vecchia C., 2016, preprint, (arXiv:1603.00034)

Peebles P. J. E., 1971, Physical cosmology

Pollack J. B., Hollenbach D., Beckwith S., Simonelli D. P., Roush T., Fong W., 1994, ApJ, 421, 615

Poulaert G., Brouillard F., Claeys W., McGowan J. W., Van Wassenhove G., 1978, Journal of Physics B Atomic Molecular Physics, 11, L671

Rahmati A., Pawlik A. H., Raičeviè M., Schaye J., 2013, MNRAS, 430, 2427

Ramaker D. E., Peek J. M., 1976, Phys. Rev. A, 13, 58

Read J. I., Hayfield T., 2012, MNRAS, 422, 3037

Rees M. J., Ostriker J. P., 1977, MNRAS, 179, 541

Revaz Y., Jablonka P., 2012a, in Capuzzo-Dolcetta R., Limongi M., Tornambè A., eds, Astronomical Society of the Pacific Conference Series Vol. 453, Advances in Computational Astrophysics: Methods, Tools, and Outcome. p. 141

Revaz Y., Jablonka P., 2012b, A&A, 538, A82

Ripamonti E., 2007, MNRAS, 376, 709

Ripamonti E., Abel T., 2004, MNRAS, 348, 1019

Rosdahl J., Schaye J., Teyssier R., Agertz O., 2015, MNRAS, 451, 34

Rosen A. L., Krumholz M. R., McKee C. F., Klein R. I., 2016, preprint, (arXiv:1607.03117)

Rudd D. H., Zentner A. R., Kravtsov A. V., 2008, ApJ, 672, 19

Savin D. W., 2002, ApJ, 566, 599

Savin D. W., Krstić P. S., Haiman Z., Stancil P. C., 2004a, ApJ, 606, L167

Savin D. W., Krstić P. S., Haiman Z., Stancil P. C., 2004b, ApJ, 607, L147

Schaller M., Gonnet P., Chalk A. B. G., Draper P. W., 2016, ArXiv e-prints:1606.02738,

Schaye J., Theuns T., Rauch M., Efstathiou G., Sargent W. L. W., 2000, MNRAS, 318, 817

Schive H.-Y., Tsai Y.-C., Chiueh T., 2010, ApJS, 186, 457

Schneider I. F., Dulieu O., Giusti-Suzor A., Roueff E., 1994, ApJ, 424, 983

Shakura N. I., Sunyaev R. A., 1973, A&A, 24, 337

Shapiro P. R., Kang H., 1987, ApJ, 318, 32

Shavitt I., 1959, J. Chem. Phys., 31, 1359

Shull J. M., Danforth C. W., Tilton E. M., 2014, ApJ, 796, 49

Smith B., Sigurdsson S., Abel T., 2008, MNRAS, 385, 1443

Smith B. D., Hallman E. J., Shull J. M., O'Shea B. W., 2011, ApJ, 731, 6

Spitzer L., 1978, Physical processes in the interstellar medium. Wiley-Interscience

Springel V., 2005, MNRAS, 364, 1105

Springel V., 2010, MNRAS, 401, 791

Stancil P. C., Lepp S., Dalgarno A., 1998, ApJ, 509, 1

Stinson G., Seth A., Katz N., Wadsley J., Governato F., Quinn T., 2006, MNRAS, 373, 1074

Sutton E. A., 1962, J. Chem. Phys., 36, 2923

Teyssier R., 2002, A&A, 385, 337

Tielens A. G. G. M., Hollenbach D., 1985, ApJ, 291, 722

Trevisan C. S., Tennyson J., 2002, Plasma Physics and Controlled Fusion, 44, 1263

Turk M. J., Abel T., O'Shea B., 2009, Science, 325, 601

Turk M. J., Smith B. D., Oishi J. S., Skory S., Skillman S. W., Abel T., Norman M. L., 2011a, ApJS, 192, 9

Turk M. J., Clark P., Glover S. C. O., Greif T. H., Abel T., Klessen R., Bromm V., 2011b, ApJ, 726, 55

Verner D. A., Ferland G. J., Korista K. T., Yakovlev D. G., 1996, ApJ, 465, 487

Vincent P., Witherdon F., Vermeire B., Park J., Arvind I., 2016, Towards Green Aviation with Python at Petascale, Supercomputing 2016, http://sc16.supercomputing.org/presentation/?id=gb101&sess=sess174

Wadsley J. W., Stadel J., Quinn T., 2004, New Astron., 9, 137

White S. D. M., Frenk C. S., 1991, ApJ, 379, 52

Wiersma R. P. C., Schaye J., Smith B. D., 2009, MNRAS, 393, 99

Wise J. H., Demchenko V. G., Halicek M. T., Norman M. L., Turk M. J., Abel T., Smith B. D., 2014, MNRAS, 442, 2560

Wishart A. W., 1979, MNRAS, 187, 59P

Wrathmall S. A., Gusdorf A., Flower D. R., 2007, MNRAS, 382, 133

Yoon J.-S., Song M.-Y., Han J.-M., Hwang S. H., Chang W.-S., Lee B., Itikawa Y., 2008, Journal of Physical and Chemical Reference Data, 37, 913

Zhao Z. X., Igarashi A., Lin C. D., 2000, Phys. Rev. A, 62, 042706

van Zyl B., Le T. Q., Amme R. C., 1981, J. Chem. Phys., 74, 314

van der Walt S., Colbert S. C., Varoquaux G., 2011, Computing in Science Engineering, 13, 22