

# GRACKLE: a chemistry and cooling library for astrophysics

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

<sup>1</sup>*Institute for Astronomy, University of Edinburgh, Royal Observatory, Edinburgh EH9 3HJ, UK*

<sup>2</sup>*San Diego Supercomputer Center, University of California, San Diego, 10100 Hopkins Drive, La Jolla, CA 92093, USA*

<sup>3</sup>*Department of Astronomy, Columbia University, New York, NY 10025, USA*

<sup>4</sup>*Simons Center for Computational Astrophysics, New York, NY 10010, USA*

<sup>5</sup>*Institut für Theoretische Astrophysik, Zentrum für Astronomie, Universität Heidelberg, Albert-Ueberle-Straße 2, D-69120 Heidelberg, Germany*

<sup>6</sup>*National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign, IL 61820, USA*

<sup>7</sup>*School of Information Sciences, University of Illinois, Urbana-Champaign, IL 61820, USA*

<sup>8</sup>*Department of Astronomy, University of Illinois, Urbana-Champaign, IL 61820, USA*

<sup>9</sup>*Institute for Computational Cosmology, Durham University, South Road, Durham DH1 3LE, UK*

<sup>10</sup>*School of Mathematical Sciences, Dublin City University, Dublin 9, Ireland*

<sup>11</sup>*Center for Relativistic Astrophysics, Georgia Institute of Technology, 837 State Street, Atlanta, GA 30332, USA*

<sup>12</sup>*Kavli Institute for Particle Astrophysics and Cosmology, Stanford University, Menlo Park, CA 94025, USA*

<sup>13</sup>*Department of Physics, Stanford University, Stanford, CA 94305, USA*

<sup>14</sup>*Department of Astrophysics, American Museum of Natural History, New York, NY 10024, USA*

<sup>15</sup>*Department of Physics and Astronomy, Michigan State University, East Lansing, MI 48824, USA*

<sup>16</sup>*Department of Computational Mathematics, Science and Engineering, Michigan State University, East Lansing, MI 48824, USA*

<sup>17</sup>*JINA: Joint Institute for Nuclear Astrophysics, Michigan State University, East Lansing, MI 48824, USA*

<sup>18</sup>*Lawrence Livermore National Laboratory, Livermore, CA 94550, USA*

<sup>19</sup>*California Institute of Technology, Pasadena, CA 91125, USA*

Accepted 2016 December 14. Received 2016 December 13; in original form 2016 October 24

## 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<sub>2</sub> formation on dust grains; tabulated primordial and metal cooling; multiple ultraviolet 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?

Modelling of plasma chemistry and radiative cooling is absolutely essential in a wide range of astrophysical phenomena. At a funda-

mental level, virtually all astrophysical objects begin as a cloud of diffuse plasma in a gravitational potential 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 absence 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

★ E-mail: [brittonsmith@gmail.com](mailto:brittonsmith@gmail.com)

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, Bryan & Norman 2002; Glover 2008; Turk et al. 2011b). At a smaller physical scale, radiative cooling can profoundly affect the structure and behaviour of accretion discs 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 haloes 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 centre of dark matter haloes, 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, Abel & O'Shea 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, Danforth & Tilton 2014).

## 1.2 Why is a multicode 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.

First, 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 library in a simulation paper, researchers can indicate which set of values were used in the

**Table 1.** Important GRACKLE resources.

Source code	<a href="https://bitbucket.org/grackle/grackle">https://bitbucket.org/grackle/grackle</a>
Documentation	<a href="https://grackle.readthedocs.io/">https://grackle.readthedocs.io/</a>
Mailing list	'grackle-cooling-users' on Google Groups

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 multiplatform 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 ultraviolet (UV) radiation backgrounds. In Section 6, we discuss implementation details, including the simulation code application program interface (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 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, Sigurdsson & Abel 2008). In 2012, the Assembling Galaxies Of Resolved Anatomy (AGORA) simulation comparison project (Kim et al. 2014) was first organized with the

**Table 2.** Dates of major GRACKLE RELEASES.

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

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 2012 October 2. The first official release (GRACKLE 1.0) occurred on 2014 January 10, with five additional major releases following over the next two and half years. Integer increments of the version number marked changes to the 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 adaptive mesh-refinement 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 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<sup>+</sup>, He, He<sup>+</sup>, He<sup>++</sup> and e<sup>−</sup> 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<sub>2</sub>), plus the two ions primarily responsible for its formation in primordial gas (H<sup>−</sup> and H<sub>2</sub><sup>+</sup>). Finally, setting `primordial_chemistry = 3` selects the 12 species network, which is an extension of the nine species model that includes D, D<sup>+</sup> 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.

**Table 3.** Chemical reactions in the six species network.

Reaction		Reference	
		Data	Fit
H + e <sup>−</sup>	→ H <sup>+</sup> + e <sup>−</sup> + e <sup>−</sup>	1	2
H <sup>+</sup> + e <sup>−</sup>	→ H + γ	3	2, 4
He + e <sup>−</sup>	→ He <sup>+</sup> + e <sup>−</sup> + e <sup>−</sup>	1	2
He <sup>+</sup> + e <sup>−</sup>	→ He + γ	5, 6	4, 6, 7
He <sup>+</sup> + e <sup>−</sup>	→ He <sup>++</sup> + e <sup>−</sup> + e <sup>−</sup>	1	2
He <sup>++</sup> + e <sup>−</sup>	→ He <sup>+</sup> + γ	3, 8	4, 9
H + H	→ H <sup>+</sup> + e <sup>−</sup> + H	10	11
H + He	→ H <sup>+</sup> + e <sup>−</sup> + He	12	11
H + γ	→ H <sup>+</sup> + e <sup>−</sup>	13	–
He + γ	→ He <sup>+</sup> + e <sup>−</sup>	13	–
He <sup>+</sup> + γ	→ He <sup>++</sup> + e <sup>−</sup>	13	–

Key: 1 – Janev, Langer & Evans (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, Chernoff & Salpeter (1991); 12 – van Zyl, Le & Amme (1981); 13 – see Section 5.

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

(i) By default, the recombination of H<sup>+</sup>, He<sup>+</sup> and He<sup>++</sup> 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<sup>+</sup> can bring about the photoionization of hydrogen (discussed in some detail in Osterbrock 1989) is not accounted for, but in most circumstances this will only lead to a small error in the H<sup>+</sup> and He<sup>+</sup> 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<sub>2</sub> 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<sub>2</sub> level populations approach their local thermodynamical equilibrium (LTE) values. This is important, because H<sub>2</sub> 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 tunnelling as well as direct collisional dissociation; previously, ENZO only accounted for the latter process

**Table 4.** Chemical reactions in the nine species network.

Reaction		Reference Data	Fit
$\text{H} + \text{e}^-$	$\rightarrow \text{H}^- + \gamma$	1	2
$\text{H}^- + \text{H}$	$\rightarrow \text{H}_2 + \text{e}^-$	3	3
$\text{H} + \text{H}^+$	$\rightarrow \text{H}_2^+ + \gamma$	4	5
$\text{H}_2^+ + \text{H}$	$\rightarrow \text{H}_2 + \text{H}^+$	6	6
$\text{H}_2 + \text{H}^+$	$\rightarrow \text{H}_2^+ + \text{H}$	7	8
$\text{H}_2 + \text{e}^-$	$\rightarrow \text{H} + \text{H} + \text{e}^-$	9	9
$\text{H}_2 + \text{H}$	$\rightarrow \text{H} + \text{H} + \text{H}$	10	10
$\text{H}^- + \text{e}^-$	$\rightarrow \text{H} + \text{e}^- + \text{e}^-$	11	12
$\text{H}^- + \text{H}$	$\rightarrow \text{H} + \text{e}^- + \text{H}$	11	12
$\text{H}^- + \text{H}^+$	$\rightarrow \text{H} + \text{H}$	13	14
$\text{H}^- + \text{H}^+$	$\rightarrow \text{H}_2^+ + \text{e}^-$	15	12, 16
$\text{H}_2^+ + \text{e}^-$	$\rightarrow \text{H} + \text{H}$	17	12
$\text{H}_2^+ + \text{H}^-$	$\rightarrow \text{H}_2 + \text{H}$	18	18
$\text{H} + \text{H} + \text{H}$	$\rightarrow \text{H}_2 + \text{H}$	19	19
$\text{H} + \text{H} + \text{H}_2$	$\rightarrow \text{H}_2 + \text{H}_2$	20, 21	22
$\text{H}^- + \gamma$	$\rightarrow \text{H} + \text{e}^-$	23	–
$\text{H}_2^+ + \gamma$	$\rightarrow \text{H} + \text{H}^+$	23	–
$\text{H}_2 + \gamma$	$\rightarrow \text{H}_2^+ + \text{e}^-$	23	–
$\text{H}_2^+ + \gamma$	$\rightarrow \text{H}^+ + \text{H}^+ + \text{e}^-$	23	–
$\text{H}_2 + \gamma$	$\rightarrow \text{H} + \text{H}$	23	–
$\text{H} + \text{H} + \text{grain}$	$\rightarrow \text{H}_2 + \text{grain}$	–	24 <sup>a</sup>

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

Key: 1 – Wishart (1979); 2 – Stancil, Lepp & Dalgarno (1998); 3 – Kreckel et al. (2010); 4 – Ramaker & Peek (1976); 5 – Latif et al. (2015); 6 – Karpas, Anicich & Huntress (1979); 7 – Krstić (2002); 8 – Savin et al. (2004a,b); 9 – Trevisan & Tennyson (2002); 10 – Martin, Schwarz & Mandy (1996); 11 – Janev et al. (1987); 12 – Abel et al. (1997); 13 – Fussen & Kubach (1986); 14 – Croft, Dickinson & Gadea (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, Trainor & Kaufman (1970); 22 – Cohen & Westberg (1983); 23 – see Section 5; 24 – Tielens & Hollenbach (1985), Omukai (2000).

<sup>a</sup>This reaction included as an additional option when metals are present.

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



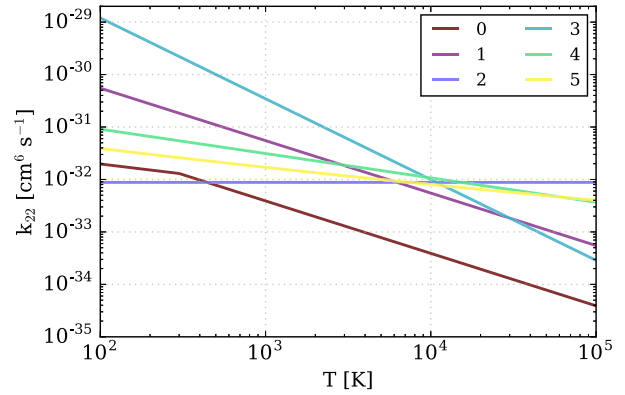
discussed in detail in Glover (2008) and Turk et al. (2011b), GRACKLE provides several different rate coefficients for this process. The user can select which of these rate coefficients to adopt by means of the `three_body_rate` parameter. The options are

**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, Salpeter & Stahler (1983), derived using detailed balance applied to the  $\text{H}_2$  collisional dissociation rate measured by Jacobs, Giedt & Cohen (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  $\text{H}_2$  partition function.



**Figure 1.** The available three-body  $\text{H}_2$  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).

**Table 5.** Additional reactions included in the 12 species network.

Reaction		Reference Data	Fit
$\text{H}^+ + \text{D}$	$\rightarrow \text{H} + \text{D}^+$	1, 2	3
$\text{D}^+ + \text{H}$	$\rightarrow \text{D} + \text{H}^+$	1, 2	3
$\text{H}_2 + \text{D}^+$	$\rightarrow \text{HD} + \text{H}^+$	4	5
$\text{HD} + \text{H}^+$	$\rightarrow \text{H}_2 + \text{D}^+$	4	5
$\text{H}_2 + \text{D}$	$\rightarrow \text{HD} + \text{H}$	6	7
$\text{HD} + \text{H}$	$\rightarrow \text{H}_2 + \text{D}$	8	5, 9
$\text{D} + \text{H}^-$	$\rightarrow \text{HD} + \text{e}^-$	10, 11	10

*Note.* The 12 species network also includes all of the reactions listed in Tables 3 and 4.

Key: 1 – Igarashi & Lin (1999); 2 – Zhao, Igarashi & Lin (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).

**4:** Rate coefficient from Glover (2008), derived from the Martin et al. (1996) high-density  $\text{H}_2$  collisional dissociation rate using detailed balance

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

We plot each of these rates in Fig. 1.

Finally, the 12 species network includes the reactions in Tables 3 and 4, plus a small number of additional reactions involving  $\text{D}^+$ ,  $\text{D}$  and  $\text{HD}$ , listed in Table 5. The intent of the 12 species network is to allow the  $\text{HD}$  abundance of the gas to be tracked accurately, since in cold gas  $\text{HD}$  can become a more effective coolant than  $\text{H}_2$  despite its much lower fractional abundance (see e.g. Johnson & Bromm 2006; McGreer & Bryan 2008). It is therefore necessary to include only a small number of reactions, as the direct conversion of  $\text{H}_2$  to  $\text{HD}$  by collisions with  $\text{D}^+$  and  $\text{D}$ , together with the corresponding inverse reactions generally dominate the production and destruction of  $\text{HD}$ .

Two reactions in the 12 species network require further discussion:

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



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  $\text{H}^-$  by D



is the same as for the corresponding reaction between H and  $\text{H}^-$ , 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



Note that we do not explicitly include this reaction in our network because it would require us to track the abundance of the  $\text{D}^-$  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 time-steps. 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 multidimensional simulations it is useful to have an implementation that 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

$$\frac{\partial n_i}{\partial t} = \sum_j \sum_l k_{jl} n_j n_l + \sum_j I_j n_j, \quad (5)$$

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 equation (5) as

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

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

$$n^{t+\Delta t} = \frac{C^{t+\Delta t} \Delta t + n^t}{1 + D^{t+\Delta t} \Delta t}. \quad (7)$$

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,  $\text{H}^+$ ,  $\text{e}^-$ , He,  $\text{He}^+$ ,  $\text{He}^{++}$ . For the nine species model, we then add  $\text{H}_2$ ,  $\text{H}^-$  and  $\text{H}_2^+$ . The  $\text{H}_2^+$  time-scale is sufficiently short that it can be decoupled and we use instead the equilibrium value:

$$n_{\text{H}_2^+} = \frac{k_{9\text{H}} n_{\text{H}} n_{\text{H}^+} + k_{11\text{H}_2} n_{\text{H}} n_{\text{H}^+} + k_{17\text{H}^-} n_{\text{H}^-} n_{\text{H}^+} + k_{29\text{H}_2} n_{\text{H}_2}}{k_{10\text{H}} n_{\text{H}} + k_{18\text{H}} n_{\text{e}} + k_{19\text{H}^-} n_{\text{H}^-} + k_{28} + k_{30}}. \quad (8)$$

Finally, for 12 species model, we add D,  $\text{D}^+$  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 time-step 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 subcycle the BDF step described above, constraining the chemical time-step such that the H and  $\text{e}^-$  abundances change by no more than 10 per cent in any subcycle step of length  $\Delta t$ :

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

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 time-step), we replace the analytically calculated time derivatives in the above expression with numerical time derivatives (i.e. change from the previous subcycle 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:

$$\frac{de}{dt} = -\dot{e}_{\text{cool}} + \dot{e}_{\text{heat}}. \quad (10)$$

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 subcycled with a time-step constraint

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

If both chemistry and cooling/heating are turned on, these are integrated at the same time, using a time-step that 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, photoheating 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:  $n_e n_{\text{H}}$ ,  $n_e^2 n_{\text{He}^+}$  and  $n_e n_{\text{He}^+}$  (Black 1981; Cen 1992);
- (ii) collisional ionization cooling for  $n_e n_{\text{H}}$ ,  $n_e n_{\text{He}}$ ,  $n_e n_{\text{He}^+}$  and  $n_e^2 n_{\text{He}^+}$  (Shapiro & Kang 1987; Cen 1992; Abel et al. 1997).
- (iii) recombination cooling:  $n_e n_{\text{H}^+}$ ,  $n_e n_{\text{He}^+}$ ,  $n_e n_{\text{He}^{++}}$  (Black 1981; Ferland et al. 1992; Hui & Gnedin 1997);
- (iv) Bremsstrahlung cooling for all ionized species (Black 1981);
- (v) Compton cooling/heating off the cosmic microwave background (CMB) (Peebles 1971) and
- (vi) photoionization heating for H, He and  $\text{He}^+$ , 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  $\text{H}_2$  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,  $\text{H}_2$ , He,  $\text{H}^+$  and  $\text{e}^-$  as described in section 2.3 of Glover & Abel (2008). The exception is that for collisions with  $\text{e}^-$ , we use revised rates from Yoon et al. (2008) and for  $\text{H}^+$ , 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  $\text{H}_2$  cooling rate that comes about once the  $\text{H}_2$  lines become 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  $\text{H}_2$  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  $4.48(1 + n_{\text{cr}}/n)^{-1}$  eV for  $\text{H}_2$  formation by the three-body reaction, and  $0.2 + 4.1(1 + n_{\text{cr}}/n)^{-1}$  eV for  $\text{H}_2$  formation on dust grain surfaces. The critical density  $n_{\text{cr}}$  is given by equation 23 of Omukai (2000). For  $\text{H}_2$  destruction, we remove 4.48 eV per  $\text{H}_2$  molecule dissociated.

Finally, in 12 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, Lodi & Tennyson (2011) for the high-density limit, and Wrathmall, Gusdorf & Flower (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  $\text{H}_2$  at high densities, with rates as described in Ripamonti & Abel (2004).
- (ii) X-ray Compton heating (or cooling) using equations 4 and 11 of Madau & Efstathiou (1999).
- (iii) A photoelectric heating rate, equal to  $\Gamma_{\text{eff}} n_{\text{H}}$ , where  $\Gamma_{\text{eff}}$  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,  $\Lambda$ ; the heating rate,  $\Gamma$ ; and the mean molecular weight,  $\mu$ , 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

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

where  $k$  is the Boltzmann constant,  $T$  is the gas temperature,  $e$  is the specific internal energy,  $\gamma$  is the adiabatic index of an ideal gas and  $m_{\text{H}}$  is the mass of a hydrogen atom. Since the mean molecular weight,  $\mu$ , 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  $\mu$  is then used as an input to the table of  $\mu(T, \dots)$ , from which a new value of  $\mu$  is calculated via linear interpolation. To prevent the solution of  $\mu$  and  $T$  from oscillating, we apply a dampener such that the new value of  $\mu$  is the average of the old value and the value from the table. For iteration,  $i$ , of this procedure,  $\mu_i$  is then given by

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

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

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

where  $\rho$  is the total gas density,  $\rho_Z$  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  $\mu$  and  $T$  that converges to within 1 per cent 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  $\Lambda$ ,  $\Gamma$  and  $\mu$  have dimensions of  $z$ ,  $\rho$  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,  $T_{\text{CMB}}$ , from the calculated cooling rate such that the final cooling rate that is applied is given by

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

This allows the cooling rate to smoothly approach zero as the temperature approaches  $T_{\text{CMB}}$  and also for the CMB to heat the gas when  $T < T_{\text{CMB}}$ . 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 (Section 4.1.1) or the tabulated solver (Section 4.1.2). As in the tabulated primordial cooling, the heating and cooling tables have three dimensions ( $z$ ,  $\rho$  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 calculate 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 photoionization, photodissociation, photodetachment and photoheating 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<sup>1</sup> (Ferland et al. 2013). We use the CIALOOP<sup>2</sup> 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 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 allow 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  $\text{H}_2$ . Both heat transfer and molecular formation rates depend very sensitively on the dust temperature,  $T_{\text{gr}}$ . 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

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

where  $\sigma$  is the Stefan–Boltzmann constant,  $T_{\text{rad}}$  is the radiation temperature and  $\kappa_{\text{gr}}$  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,  $T_{\text{rad}}$ . The dust/gas heat transfer rate per unit dust mass,  $\Lambda_{\text{gas/grain}}$ , is given by

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

(Hollenbach & McKee 1989), where  $T$  is the gas temperature and  $\rho_{\text{gr}}$  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  $9.34 \times 10^{-3}$  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

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

with a normalization of  $\kappa_{\text{gr}}(T_{\text{gr}} = 200 \text{ K}) = 16 \text{ cm}^2 \text{ g}^{-1}$  (Pollack et al. 1994; Omukai 2000). The steep power-law for  $T > 1500 \text{ K}$  is designed to mimic the effect of grains sublimating. The time-scale 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  $T_{\text{gr}}$  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  $\text{H}_2$  formation on dust, which is a function of both the gas and dust temperatures as

<sup>1</sup> <http://nublado.org/>

<sup>2</sup> [https://bitbucket.org/brittonsmith/cloudy\\_cooling\\_tools](https://bitbucket.org/brittonsmith/cloudy_cooling_tools)

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 towards 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  $\text{erg s}^{-1} \text{cm}^{-3}$ ) to mimic heat input from a radiation field or specific ( $\text{erg s}^{-1} \text{g}^{-1}$ ), corresponding to a uniform temperature change that is independent of density.

## 5 RADIATION BACKGROUNDS

The Universe was reionized during the epoch of  $z \sim 6\text{--}10$  by the buildup of radiation from stars and active galactic nuclei (AGNs). This radiation heated the intergalactic medium (IGM) to  $\sim 2 \times 10^4 \text{ K}$  (e.g. Schaye et al. 2000), inhibiting the collapse of haloes 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 photochemical 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 that 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 photoionization heating rates for H, He and  $\text{He}^+$ ; photoionization rates for H, He,  $\text{He}^+$  and  $\text{H}_2$ ; photodissociation rates for  $\text{H}_2$  and  $\text{H}_2^+$ ; and the photodetachment rate for  $\text{H}^-$ . 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

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

where  $\bar{\sigma}_v$  is the grey (spectrum-averaged) absorption cross-section,  $T_4 = T/10^4 \text{ K}$ ,  $\Gamma_{-12}$  is the photoionization rate in units of  $10^{-12} \text{ s}^{-1}$  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  $x = n_{\text{H}}/n_{\text{H,SSH}}$ .

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  $\text{He}^+$  optically thin. The second includes self-shielding in both neutral H and neutral He using these equations, leaving  $\text{He}^+$  optically thin. Finally, the third applies these equations for both neutral H and He as before, while ignoring  $\text{He}^+$  photoionization/photoheating from the UV background entirely. (In other words, when accounting for self-shielding, leaving  $\text{He}^+$  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 versus 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 grey-averaged cross-section,  $\bar{\sigma}_v$ , depends on the evolving spectrum of the UV background. Included in the `GRACKLE` data files are the pre-computed  $\bar{\sigma}_v$  for H, He and  $\text{He}^+$  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).<sup>3</sup>

## 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<sup>+</sup> photoionization rates, as well as a H<sub>2</sub> photodissociation 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, Greig & Sobacchi 2016), or coupled with the hydrodynamics, which is becoming more commonplace in galaxy and star formation simulations (e.g. Wise et al. 2014; Baczynski, Glover & Klessen 2015; Ocvirk et al. 2016; Rosdahl 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. Aykatalp et al. 2013).

## 6 IMPLEMENTATION

In this section, we discuss details of the code itself, including the 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.

### 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<sub>2</sub>.

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 that are to be ignored. An example of calling the main chemistry solver function on a 10<sup>3</sup> grid with three boundary zones on each side is shown below:

```
grackle_field_data my_fields;
```

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

```

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 that 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 `grackles_field_datastruct` 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 are 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 three-dimensional array, or as a one-dimensional ‘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, Colbert & Varoquaux 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 data set 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 time-step 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

$$\frac{d\rho}{dt} = \frac{\rho}{t_{\text{col}}}, \quad (21)$$

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

$$t_{\text{col}} = \frac{t_{\text{ff}}}{\sqrt{1-f}} \quad (22)$$

and the free-fall time is given by

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

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

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

where the effective adiabatic index,  $\gamma$ , is

$$\gamma \equiv \frac{\partial \ln p}{\partial \ln \rho}. \quad (25)$$

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

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

where the pressure is given by

$$p = \frac{\rho k T}{\mu m_H}, \quad (27)$$

the specific internal energy is given by equation (12), and  $\Lambda$  is the radiative cooling rate. For each iteration of this function, the time-step 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 Section 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 6, 9 or 12 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 behaviour of the cooling for a given species can be easily changed.

To modify the non-equilibrium cooling behaviour, more work is required. GRACKLE assumes that non-equilibrium chemistry models are hierarchical, with the 9 species model composed of the 6 species model plus three additional species and likewise for the 12 species model. To expand the network to include, say, a fourth model containing the 12 species model as a subset then the procedure is straightforward. However, if a different chemical network is envisaged containing some species already in the 12 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 ( $\text{H} + \text{e}^- \rightarrow \text{H}^+ + 2\text{e}^-$ ). 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 inter-

polation 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 Section 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 need 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 `PYTEST`<sup>4</sup> 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 data set that is loadable by the `yt` package. Known ‘correct’ answers for the summary plot and `yt`-loadable data set are saved in the

<sup>4</sup> <http://pytest.org/>

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<sup>5</sup> and that all instructional sample codes compile and run without producing errors.

### 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 Section 6.2.1), after which time the cooling time is calculated. The tests assert that the cooling time values agree to within four 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 four 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 types 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  $\rho$ , whereas collisional heating/cooling rates are proportional to  $\rho^2$  (or  $\rho^3$  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 four 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 two orders of magnitude with the centre 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 photoionization 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 example, this yields a CIE solution for neutral H given by

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

where  $\alpha_{\text{H}^+}$  is the recombination rate of  $\text{H}^+$  and  $\Gamma_{\text{H}}$  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  $10^4$  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

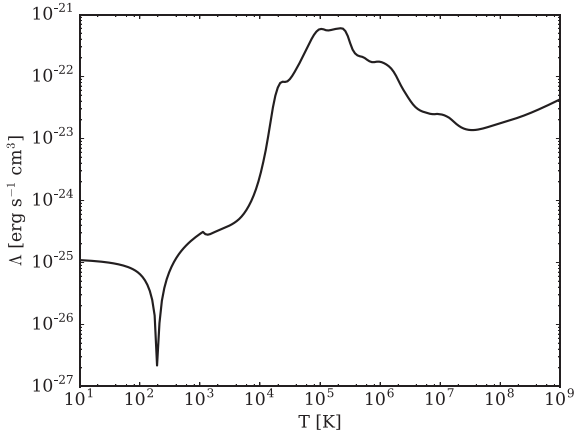
Similar to the CIE unit test (Section 7.1.2), the cooling rate answer test initializes a fluid container with a constant density and smoothly varying temperature from 10 to  $10^9$  K, then iterates the chemistry solver until all species have reached equilibrium. Unlike the CIE unit test, metal cooling 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 Fig. 2.

### 7.1.4 Answer test: constant density cooling test

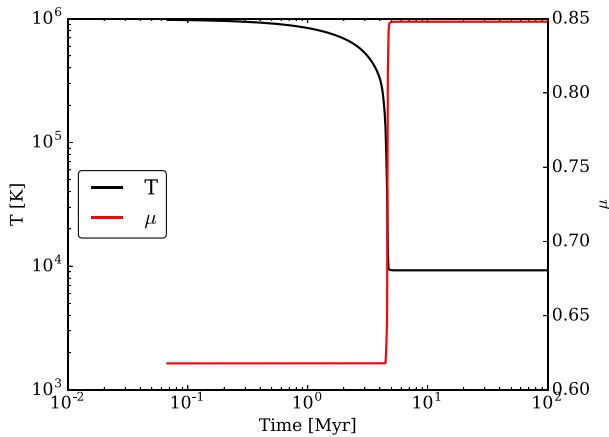
The uniform cooling answer test simulates the thermal evolution of a parcel of gas cooling at constant density. This test ensures that the solver properly evolves the internal energy over a period of time. The test initializes a single-cell fluid container with a density of  $0.1 \text{ amu cm}^{-3}$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

<sup>5</sup> <https://www.python.org/dev/peps/pep-0008/>





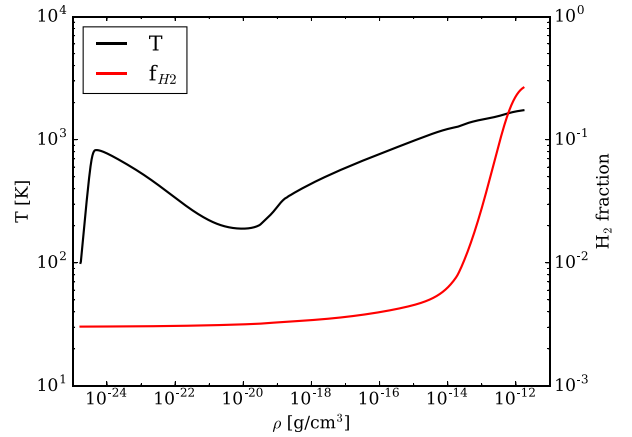
**Figure 2.** Figure output by the default configuration of the cooling rate answer test, described in Section 7.1.3, showing the equilibrium cooling rate as a function of temperature for a Solar metallicity gas at a density of  $1 \text{ amu cm}^{-3}$  with an incident radiation field described by the 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  $\sim 200 \text{ K}$ , the radiation field induces a net heating rate.



**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.

### 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  $0.1 \text{ amu cm}^{-3}$  and an initial temperature of  $50\,000 \text{ K}$ . Before beginning the free-fall collapse phase, the cloud is allowed to cool at a constant density to a temperature of  $100 \text{ K}$  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 (Fig. 4) and once with a metallicity of  $10^{-3} Z_{\odot}$ .



**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  $\text{H}_2$  fraction (red) as a function of density for a free-fall collapse model of a metal-free gas.

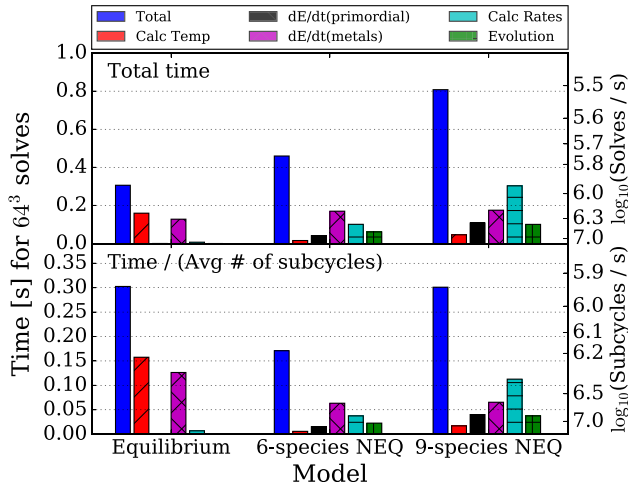
## 7.2 Performance

### 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, multidimensional 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 that 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 initialization 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 one-dimensional list of particle quantities into a two-dimensional grid).

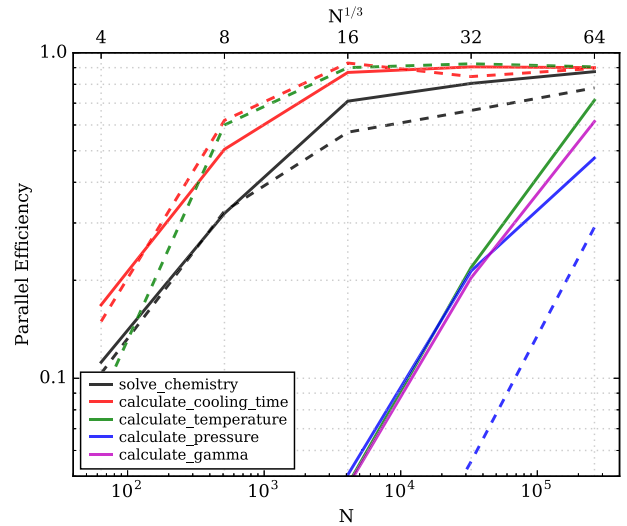


**Figure 5.** Top panel: Total time to compute the cooling rate test (Section 7.1.3) in  $64^3$  fluid containers, using the tabulated equilibrium cooling model (left), six species non-equilibrium model (middle) and nine species non-equilibrium model (right). The different bars show the time needed for the complete solve (blue), the temperature calculation (red), the  $\dot{e}$  computation due to primordial chemistry (black) and metals (magenta), 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 (Section 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  $\log(n_H/\text{cm}^{-3}) = [-1, 3]$ ,  $\log(T/\text{K}) = [1, 8]$  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 time-step in the non-equilibrium solver 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 Fig. 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 per cent and 164 per cent more time than the equilibrium solver, respectively. In this test, GRACKLE can update  $8.6 \times 10^5$ ,  $5.7 \times 10^5$  and  $3.2 \times 10^5$  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



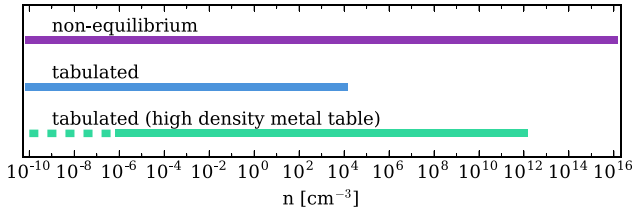
**Figure 6.** OpenMP parallel efficiency using 20 threads as a function of the size of the input array. The solid lines show the use of the non-equilibrium solver with `primordial_chemistry = 3` and the dashed lines show the analogous functions using the tabulated solver. For all time-consuming routines, the parallel efficiency reaches  $\sim 60$  per cent to 90 per cent for  $16^3$  cells and  $\sim 80$  per cent to 90 per cent for  $64^3$  cells.

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 Fig. 5. The equilibrium and non-equilibrium solvers take an average of 1.01 and 2.67 subcycles per solve, respectively. GRACKLE can perform  $8.7 \times 10^5$ ,  $1.5 \times 10^6$  and  $8.7 \times 10^5$  subcycles per second for the equilibrium, six species and nine species solvers, respectively. Here we see that the equilibrium solver actually requires 75 per cent 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. Fig. 6 shows an OpenMP performance benchmark for both the non-equilibrium and tabulated functionality, where parallel efficiency is defined as the ratio of multithread 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 10 cores. We compile GRACKLE



**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,  $10^{-6} \text{ cm}^{-3} < n < 10^{12} \text{ cm}^{-3}$ , but extrapolation down to  $n = 10^{-10} \text{ cm}^{-3}$  is still valid, as indicated by the dashed line. For each of these, the valid temperature range is roughly  $1\text{--}10^9 \text{ K}$ .

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  $\sim 60\text{--}90$  per cent for  $16^3$  cells and  $\sim 80\text{--}90$  per cent 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.

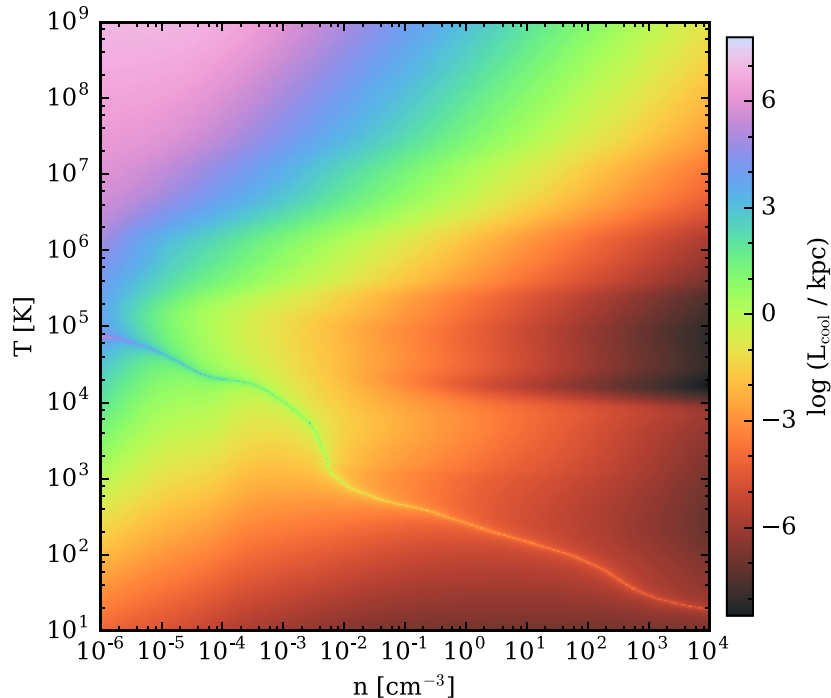
## 8 DISCUSSION

### 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 Fig. 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  $\rho^2$  to  $\rho$ . Hence, extrapolation beyond the upper limit may result in vast overprediction 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  $1\text{--}10^9 \text{ K}$ . 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 objects 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 overcooling problem that has classically plagued cosmological simulations (e.g. Katz, Weinberg & Hernquist 1996; Balogh et al. 2001). In Fig. 8, we



**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.

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:

- (i) AREPO (Springel 2010)
- (ii) ART-I (Kravtsov 1999; Kravtsov, Klypin & Hoffman 2002)
- (iii) ART-II (Rudd, Zentner & Kravtsov 2008)
- (iv) CHANGA (Wadsley, Stadel & Quinn 2004; Stinson et al. 2006)
- (v) COSMOS++ (Anninos, Fragile & Murray 2003; Anninos, Fragile & Salmonson 2005)
- (vi) ENZO (Bryan et al. 2014)
- (vii) GADGET-3 (Springel 2005)
- (viii) GAMER (Schive, Tsai & Chiueh 2010)
- (ix) GASOLINE (Wadsley et al. 2004)
- (x) GEAR (Revaz & Jablonka 2012b,a)
- (xi) GIZMO (Hopkins 2015)
- (xii) RAMSES (Teyssier 2002)
- (xiii) SPHS (Read & Hayfield 2012)
- (xiv) 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 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, Schaye & Smith (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 that 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.

## ACKNOWLEDGEMENTS

We are grateful to the anonymous referee whose comments helped to clarify various points in the manuscript. BDS would like to thank Michael Kuhlen for his work on the code in the early stages of the project; the organizers of the AGORA project, Ji-hoon Kim, Joel Primack and Piero Madau for the original motivation for the GRACKLE project; as well as Nick Gnedin, James Wadsley and Romeel Davé for providing useful suggestions for additional functionality. GLB acknowledges support from NASA grant NNX15AB20G and NSF grant 1312888. SCOG acknowledges support from the Deutsche Forschungsgemeinschaft via SFB 881, ‘The Milky Way System’ (sub-projects B1, B2 and B8) and SPP 1573, ‘Physics of the Interstellar Medium’ (grant number GL 668/2-1), and from the European Research Council under the European Community’s Seventh Framework Programme (FP7/2007-2013) via the ERC Advanced Grant STARLIGHT (project number 339177). NJG and MJT were supported by NSF grant ACI-1535651 and by the Gordon and Betty Moore Foundation’s Data-Driven Discovery Initiative through Grant GBMF4651. JHW is supported by NSF grants AST-1333360 and AST-1614333 and Hubble theory grants HST-AR-13895 and HST-AR-14326. AE is supported by a NSF Graduate Research Fellowship grant No. DGE-16-44869. We also thank the NSF for computational resources provided through the XSEDE program. JAR acknowledges support through the STFC capital grant ST/L00075X/1 and the Marie Curie Research Fund, grant 699941. BWO acknowledges support from NASA through grants NNX12AC98G, NNX15AP39G, and Hubble Theory Grants HST-AR-13261.01-A and HST-AR-14315.001-A. BWO was supported in part by the sabbatical visitor program at the Michigan Institute for Research in Astrophysics (MIRA) at the University of Michigan in Ann Arbor, and gratefully acknowledges their hospitality. Work by PA was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory



under Contract DE-AC52-07NA27344. GRACKLE was also made possible by the open-source projects, HDF,<sup>6</sup> h5py,<sup>7</sup> MATPLOTLIB (Barrett et al. 2005) and NUMPY (van der Walt et al. 2011).

## 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, ASP Conf. Ser. Vol. 347, Astronomical Data Analysis Software and Systems XIV. Astron. Soc. Pac., San Francisco*, p. 91
- Behnel S., Bradshaw R., Citro C., Dalcin L., Seljebotn D., Smith K., 2011, *Comput. Sci. Eng.*, 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, *J. Phys. Chem. Ref. 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, Proc. IAU Symp. 120, Astrochemistry. Reidel, Dordrecht*, p. 109
- 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, *J. Phys. B: At. Mol. Phys.*, 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 Physics. Kluwer, Dordrecht* p. 304
- Glover S., 2008, in *O'Shea B. W., Heger A., eds, AIP Conf. Proc. Vol. 990, First Stars III. Am. Inst. Phys., New York*, p. 25
- 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, preprint ([arXiv:1309.3783](https://arxiv.org/abs/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*, p. 55
- 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, *Phys. Rev. Lett.*, 107, 023201
- Honvault P., Jorfi M., González-Lezana T., Faure A., Pagani L., 2012, *Phys. Rev. Lett.*, 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, *Phys. Rev. Lett.*, 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-Verlag, Berlin*
- 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
- 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
- Madau P., Efsthathiou G., 1999, *ApJ*, 517, L9
- Martin P. G., Schwarz D. H., Mandy M. E., 1996, *ApJ*, 461, 265
- 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, *Phys. Rev. Lett.*, 91, 063201
- Miller K. A. et al., 2012, *Phys. Rev. A*, 86, 032714
- Ocvirk P. et al., 2016, *MNRAS*, 463, 1462
- 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, Mill Valley, CA*
- Osterbrock D. E., Ferland G. J., 2006, *Astrophysics of Gaseous Nebulae and Active Galactic Nuclei. University Science Books, Mill Valley, CA*
- 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](https://arxiv.org/abs/1603.00034))
- Peebles P. J. E., 1971, *Physical Cosmology. Princeton Univ. Press, Princeton, NJ*
- 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, *J. Phys. B: At. Mol. Phys.*, 11, L671
- Rahmati A., Pawlik A. H., Raičević M., Schaye J., 2013, *MNRAS*, 430, 2427

<sup>6</sup> <https://www.hdfgroup.org/>

<sup>7</sup> <http://www.h5py.org/>

- 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, *ASP Conf. Ser. Vol. 453, Advances in Computational Astrophysics: Methods, Tools, and Outcome*. Astron. Soc. Pac., San Francisco, 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, *MNRAS*, 463, 2553
- 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, *Proc. Platform Adv. Scientific Comp. Conf., SWIFT: Using task-based parallelism, fully asynchronous communication, and graph partition-based domain decomposition for strong scaling on more than 100,000 cores*. ACM, New York
- Schaye J., Theuns T., Rauch M., Efstathiou G., Sargent W. L. W., 2000, *MNRAS*, 318, 817
- Shive 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, New York
- 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 Phys. Control. 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
- van der Walt S., Colbert S. C., Varoquaux G., 2011, *Comput. Sci. Eng.*, 13, 22
- van Zyl B., Le T. Q., Amme R. C., 1981, *J. Chem. Phys.*, 74, 314
- 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*. Available at: <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, *J. Phys. Chem. Ref. Data*, 37, 913
- Zhao Z. X., Igarashi A., Lin C. D., 2000, *Phys. Rev. A*, 62, 042706

This paper has been typeset from a  $\text{\LaTeX}$  file prepared by the author.