CHIMES

October 24, 2016

1 Introduction

The CHIMES (CHemistry of the Interstellar Medium and Extragalactic Sources) package is designed to replace the existing cooling routines in a hydrodynamic simulation code. For each gas cell/particle, when the simulation code would normally evaluate the gas cooling over the current timestep, replace this with the chimes_network(gasVariables, globalVariables) routine. This will evolve both the chemical abundances and the temperature of the gas cell/particle over the current hydrodynamic timestep. This routine takes two arguments: firstly, a pointer to the 'gasVariables' structure which contains information on the gas cell (e.g. temperature, chemical abundances etc.). Every gas cell will need its own instance of this structure, and these variables will need to be updated every hydrodynamic timestep. Also, after the chimes_network routine has been called, the temperature used by the hydrodynamics solver (or equivalently the thermal energy, or whichever variable the hydrodynamics code tracks) will need to be updated with the new temperature from the gasVariables structure. Secondly, the routine takes a pointer to the 'globalVariables' structure, which contains global parameters and information that affects the behaviour of the thermo-chemistry solver. These will need to be read in from the parameter file. These two structures are declared in the file 'allvars.h' and will also need to be declared within the main simulation code itself. If the code is written in C, allvars.h can simply be included as a header file in the code, but if it is written in another language, e.g. Fortran, you will need to declare these structures yourself. The variables they contain are descibed in more detail below.

Additionally, allvars.h also contains function prototypes for several routines from the chemical model that you might want to use in the main simulation code. calculate_mean_molecular_weight(gasVariables) takes a pointer to a gasVariables structure and uses the abundances in this structure to calculate the mean molecular weight. allocate_gas_abundances_memory(gasVariables, globalVariables) allocates memory to the abundance array in the gasVariables structure. initialise_gas_abundances(gasVariables, globalVariables) sets the initial abundances of each species. The ionisation states that it starts in are determined by the InitIonState parameter in globalVariables (see below).

2 Global Variables

The variables in this structure control the behaviour of the thermo-chemistry solver and need to be read in from the parameter file.

BenTablesPath - the path to the directory containing Ben's tables (maximum 500 characters).

PhotoIonTablePath - the path (maximum 500 characters) to a text file that specifies the photoionisation cross section tables. Each line in this text file gives the full path to an HDF5 table. If you include multiple UV spectra (e.g. a UV background and an AGN), you need to give one table for each spectrum. The number of tables given in this text file needs to match the 'N_spectra' parameter (see below), otherwise CHIMES will exit with an error message. Each HDF5 table contains the photoionisation cross sections, average photon excess energies (for photoheating) and cosmic ray ionisation rates relative to the HI rate, along with the 'dust_G_parameter' and 'H2_dissocJ' parameters for the given UV spectrum (see below). We also tabulate the shielding factors (i.e. the factor by which the photoionisation rate is suppressed) due to HI, H_2, HeI and HeII, and the factor by which shielding increases the average photon excess energies. These are tabulated as functions of the 'effective' H and He column densities (see Richings, Schaye & Oppenheimer, 2014). Note that for species with an ionisation energy < 1 Ryd we only consider dust shielding, which suppresses photoionisation by $\exp(-\gamma_d A_V)$ at an extinction of A_V. For these species, the factor γ_d is given as the first element in the shielding factors table, with the remaining elements set to 1.

EqAbundanceTablePath - path to the hdf5 table that contains the equilibrium abundances of each species as a function of temperature, density and metallicity. We use these for particles that lie on the equation of state, since the temperature of the gas cell/particle in this case will be artificial, so we just set them to equilibrium rather than integrating the chemistry for them. Note that these tables are calculated assuming a constant UV radiation field and no shielding column density. In the future we may want to extend these, but for now they are just used to get approximate abundances on the equation of state. Also, note that if you remove elements from the network you will need to create a new equilibrium abundances table that also excludes these elements.

MolecularTablePath - path to the molecular cooling tables, for CO, H_2O and OH.

AdditionalRatesTablesPath - path to the hdf5 table containing the rate coefficients of the reactions that I have added in addition to those from Ben's model (for those that depend only on temperature).

ThermEvolOn - If this is set to '1' the temperature is evolved along with the chemistry by chimes_network(). If this is '0' then the temperature is held fixed - use this if you want chimes_network() to only evolve the chemical abundances, and not the temperature.

reductionOn - If this is set to '1' the thermo-chemistry solver will remove species from the network that it considers to be more unimportant. For atomic metal ions, it determines the most abundant ion of each element and includes all ionisation states within some number of this most abundant state (determined by the n_ions_low, n_ions_med and n_ions_high parameters - see below), while molecules will be excluded at temperatures above the T_mol parameter (see below). If this is '0' the full network is used. I have not fully tested this yet, so for now you should just set this to '0'.

updatePhotonFluxOn - once we couple the chemistry solver to radiative transfer we will need to update the photon flux along with the chemistry. This switch will be used to control whether or not the photon flux is updated by the chemistry solver, but we have not yet implemented this, so for now just set it to '0'.

cellSelfShieldingOn - if this is '1' the chemistry solver will take the size of the gas cell and its density to give the column density of the cell, which it then uses to shield the gas from the UV radiation, using its abundances to calculate shielding from HI, HeI, H_2 and CO, plus dust. If this is set to '2', the column densities of each species are evaluated at the beginning of the hydrodynamic timestep, but are then held fixed, i.e. they are not updated as the abundances of individual are evolved (this makes the code run much faster). If this is set to '0' shielding will be ignored.

N_spectra - the number of UV spectra to be included.

InitIonState - this flag is used by the initialise_gas_abundances() routine, and determines in what ionisation states the gas starts. The possible values are:

0 - Fully atomic, neutral.
1 - Fully atomic, singly ionised.
2 - Fully atomic, double ionised.
etc., up to '26' (all elements fully ionised)

staticMolCooling - the molecular cooling rate from CO and H_2O can be affected by line trapping, which depends on the divergence of the velocity of the gas. However, if this option is set to '1' the divergence of the velocity is ignored and only thermal line broadening is considered.

T_EqThresh - If we are using the 'ForceEqOn' flag to force all abundances to be in chemical equilibrium (see next section), this parameter will determine how often we need to update the equilibrium abundances as we evolve the temperature in chemical equilibrium. This parameter is the relative change in temperature that we require before the abundances are updated - we use a default value of 0.05. Note that this will only be important when the gas is being heated off the equation of state, since it cannot cool below this. This parameter is only used when both 'ForceEqOn' and 'ThermEvolOn' are '1'.

time_tolerance - If the reductionOn switch is '1' we may need to update the reduced network before the end of the current hydrodynamic timestep. We calculate a subcycle timestep based on how quickly the most abundant ionisation states are evolving and how quickly the temperature is approaching the T_mol threshold. This time_tolerance parameter determines the tolerance when calculating this subcycle timestep - we use a default value of 1.0. If reductionOn is '0' this parameter does not matter.

min_subcyclestep - this is the smallest subcycle timestep that we allow, in seconds. We use a default value of 100 s, but it probably doesn't need to be this small.

T_mol - the maximum temperature for which we include molecules. Above this temperature all molecules are removed from the network. Note that this is always the case, regardless of whether the 'reductionOn' switch is '1' or '0'. By default we have set this to 10^5 K, but this can probably be set a bit lower, e.g. $\sim 2 - 3 \times 10^4$ K.

n_ions_low - the next three parameters are used in our reduction algorithm, as described above for the 'reductionOn' parameter. We specify the number of ions of an element included in the reduced network separately for the metal elements with small atomic number, medium atomic number and high atomic number. This is useful because for example iron contains many more ions than carbon, so we might want to include a wider range of ionisation states of iron than carbon. n_ions_low is used for C, N and O - we use a default value of 3.

n_ions_med - as above, but for Ne, Mg and Si - we use a default value of 4.

n_ions_high - as above, but for S, Ca and Fe - we use a default value of 5.

grain_temperature - the dust temperature, which is held constant. We take a default value of 10 K, but the results of the chemical model are insensitive to dust temperatures in the range 10 K to 50 K.

cmb_temperature - the temperature of the CMB. This is only used in calculating the compton cooling of gas from the CMB, which is likely to only be important at high redshifts.

relativeTolerance - the relative tolerance used by the CVode solver - we use 10^{-6}.

absoluteTolerance - the absolute tolerance used by the CVode solver for the chemical abundances - we use 10^{-13}.

thermalAbsoluteTolerance - the absolute tolerance used by the CVode solver for integrating the temperature (note that it is the internal energy that we integrate, in cgs units). We use 10^{-40}.

scale_metal_tolerances - if this flag is set to 1, the absolute tolerances used by the metal species (and He) will be multiplied by the corresponding element abundance. For molecules containing both C and O, the smaller of those two element abundances will be used. This is important in simulations that include metal enrichment, as the first enrichment events in the simulation can produce gas particles/cells with very low metal abundances. If the total abundance of a given metal is less than the absolute tolerance, then the integration of that metal's ionisation states can become unstable, which can cause the chemistry solver to crash. If this flag is set to 0, all species use the same absolute tolerance.

element_included[9] - an array of integers that flag which metals we will include in the chemical network, in the order: C, N, O, Ne, Mg, Si, S, Fe. A '1' indicates that the element is included, a '0' indicates that it is not included. Note that H and He will always be included. Also, if you want the full molecular chemistry you will need to include C and O.

speciesIndices[TOTSIZE] - this contains the array indices for the abundances array in the gasVariables structure (see next section). TOTSIZE is the total number of species included in the full network (currently 157, defined in allvars.h). The species appear in this array in the order given in the enumerated list given at the end of allvars.h. If a species is not included in the network, because it contains an element that has been excluded, then its value in this array will be -1, otherwise this array gives its index in the abundances array. See the next section for how to use this. Note that the speciesIndices array will be set by the init_chimes() routine, so the user does not need to worry about how to set it up, but it appears in globalVariables so that the user can use it when accessing the abundances array.

totalNumberOfSpecies - the total number of chemical species to be included in the network. This is calculated by the init_chimes() routine using the element_included array - it does not need to be specified separately.

3 Gas Variables

Each gas cell/particle will need an instance of the gasVariables structure, which contains the following variables:

***abundances** - This array contains the abundance of each species in the chemical network, expressed as the number density of that species divided by the total hydrogen density. The size of this array depends on how many elements have been included in the network. The order in which species will appear in this array is given by the enumerated list at the end of allvars.h. If all elements are included then, for a given species, its number in this list will also be its index in the abundances array, however this is no longer true once we start removing elements. To get the correct index of a species in the abundances array, use the speciesIndices array in the globalVariables structure (see previous section). For example, to get the abundance of HII you would use:

```
abundances[globalVariables->speciesIndices[HII]]
```

If a gas cell/particle has been enriched (e.g. from a supernova) since the last time chimes_network() was called on it, in addition to updating the total element abundances (see below) you will also need to update the abundances array. This is necessary because the abundances are defined relative to hydrogen, so if a total element abundance changes it will no longer match the sum of individual species abundances that contain that element. To do this you will need to make an assumption about which ionisation state(s) the injected material is in. I simply assume it is all injected as neutral, atomic gas, but there are alternative approaches you could take.

element_abundances[10] - This array gives the total abundances of each element in the network, from He to Fe in order of increasing atomic number [1]. These are again given as number density of the element divided by total hydrogen number density.

nH_tot - The total hydrogen number density, in cm^{-3}.

temperature - Gas temperature in K.

TempFloor - The minimum temperature below which the gas is not allowed to cool.

divVel - Velocity divergence, in units of s^{-1}. This is only used by the molecular cooling from CO and H_2O, but there is an option to use 'StaticMolecularCooling', i.e. to neglect the velocity divergence in the molecular cooling (see the previous section). If this option is used then it does not matter what you use for divVel.

doppler_broad - The doppler broadening parameter b, in units km s^{-1}, which is used in calculating the H_2 self shielding. There are a number of ways you can approximate this (see section 2.4.1 of Glover & Jappsen (2007) for more details), but for now you can simply assume b = 7.1 km s^{-1}, corresponding to a turbulent velocity dispersion of 5 km s^{-1}. Note that if cellSelfShielding (see previous section) is switched off this parameter does not matter, but do not let it be zero, otherwise you will end up dividing by zero in the photodissociation rate of H_2.

***isotropic_photon_density** - An array containing the number density of hydrogen-ionising photons (i.e. with energies > 1 Ryd), in units cm^{-3}, for each UV spectrum. The order in this array should match the order that the cross section tables are specified in the 'PhotoIonTablePath' file (see above). If you are coupling the thermo-chemistry solver with a radiative transfer code, note that this variable is NOT currently updated through the course of the chemical evolution. We will address this in the future.

***directed_flux_magnitude** - This will be used by RamsesRT, but we haven't yet got the chemistry solver working self consistently with radiative transfer, so ignore this for now (each element in this array is automatically set to zero, and is never used by the code).

***dust_G_parameter** - is used to convert the isotropic_photon_density of each UV radiation field to G_0, which is defined as the energy density of the radiation field from 6 eV to 13.6 eV in units of the Habing (1968) field:

$$G_0 \equiv \frac{u_{6-13.6\mathrm{eV}}}{5.29 \times 10^{-14}\,\mathrm{erg\ cm}^{-3}}. \tag{1}$$

The dust_G_parameter is defined as:

$$\mathrm{dust\_G\_parameter} = \frac{G_0}{\mathrm{isotropic\_photon\_density} * c}, \tag{2}$$

where the isotropic_photon_density is the number density of hydrogen-ionising photons (see next section), and c is the speed of light. dust_G_parameter is stored as an array, one value for each UV spectrum, and is read in from the corresponding photoionisation cross section table specified

[1]He, C, N, O, Ne, Mg, Si, S, Ca, Fe

in the ‘PhotoIonTablePath’ file (see above). Note that the init_chimes() routine reads the values from the tables into an external array, and not into the array within gasVariables (since there may be many gas particles). The array in gasVariables will then need to be set from the external array after the tables have been read in. The dust_G_parameter in cgs units is 5.282989×10^{-7} for the redshift zero Haardt & Madau (2001) extragalactic UV background and 1.075972×10^{-7} for the Black (1987) interstellar radiation field.

***H2_dissocJ** - the number density of photons in the energy range $12.24\,\mathrm{eV} < h\nu < 13.51\,\mathrm{eV}$ divided by the speed of light times the number density of hydrogen-ionising photons (in cgs units). This is used in calculating the photodissociation rate of molecular hydrogen. H2_dissocJ is stored as an array, one value for each UV spectrum, and is read in from the corresponding photoionisation cross section table specified in the ‘PhotoIonTablePath’ file (see above). The values from the tables are read into an external array by the init_chimes() routine. The array within gasVariables will then need to be set from this external array once the tables have been read in. H2_dissocJ = 1.924×10^{-11} for the Haardt & Madau (2001) UV background, and 2.047×10^{-11} for the Black (1987) interstellar radiation field.

cr_rate - The ionisation rate of neutral hydrogen due to cosmic rays, in units of s^{-1}. We use a default value of $2.5 \times 10^{-17}\,\mathrm{s}^{-1}$, from Williams et al. (1998).

metallicity - In solar units, i.e. $Z/Z_{\odot}$, where we use CLOUDY’s solar abundances, with $Z_{\odot} = 0.0129$.

cell_size - This is used when cellSelfShielding (see previous section) is switched on to evaluate the column density of the cell. You can use either the physical size of the grid cell/Sph kernel smoothing length, or you can use the Sobolev length (I think the latter is more physical).

hydro_timestep - The length of the current hydrodynamic timestep, in seconds, over which you want to evolve the thermo-chemistry of the gas cell/particle.

ForceEqOn - There are several options to set the chemical abundances to equilibrium, rather than following the non-equilibrium evolution. The equilibrium abundances can be taken from the ’EqAbundancesTable’, or they can be computed by setting the rate equations to zero and then solving to find the equilibrium abundances. If thermal evolution is on, the temperature will be evolved using the implicit cooling scheme of EagleThe possible combinations are set using the ForceEqOn parameter as follows:
0 - Solve the full non-equilibrium chemical network.
1 - Equilibrium abundances from tables; evolve temperature using Eagle’s implicit cooling scheme.
2 - Solve equilibrium abundances from rate equations; evolve temperature using Eagle’s implicit cooling scheme.

constant_heating_rate - a constant heating rate, in units of ergs $\mathrm{s}^{-1}\ \mathrm{cm}^{-3}$, that is added to the net cooling rate (positive for heating, negative for cooling). This can be used to add a constant heating term, for example from hydro heating.

References

Black, J. H. 1987, ASSL, 134, 731

Glover, S. C. O., & Jappsen, A.-K. 2007, ApJ, 666, 1

Haardt, F., & Madau, P. 2001, in Neumann D. M., Tran J. T. V., eds, XXIst Moriond Astrophys.

Meeting, Clusters of Galaxies and the High Redshift Universe Observed in X-rays Editions Frontieres, Paris, p. 6

Habing, H. J. 1968, Bull. Astron. Inst. Netherlands, 19, 421

Richings, A. J., Schaye, J., & Oppenheimer, B. D. 2014, MNRAS, 442, 2780

Williams, J. P., Bergin, E. A., Caselli, P., Myers, P. C., & Plume, R. 1998, ApJ, 503, 689