UCLCHEM User Guide

January 3, 2017

J. Holdship

1 Introduction

UCLCHEM is a gas-grain chemical code written in FORTRAN 95. It solves a couple system of ODEs that give the abundance of every species from a user defined set of species and reactions. As well as the user defined reactions it implements optional processes such as thermal and non-thermal desorption and self-shielding of H2 and CO. This document explains the code generally and explains how to use it. Detailed information and references for individual processes in the code can be found in the comments. In Section 2, the basic process of setting up and running the code is explained and Section 3 gives further explanation of the inputs and outputs of the code. The later sections then go into depth about the main components of the code so that the physical models can be well understood. At the end of the document, Section 7 gives troubleshooting advice for problems encountered by first time users.

2 Quick Start Guide

Follow the instructions in this section to set up UCLCHEM and start running models. When more detail may be necessary, we provide links to later sections where each process is fully explained.

2.1 1. Prepare your network

- In the MakeRates/inputFiles directory, there should be a UMIST database file, a custom reaction file and a species file (UMIST12, uclgrainbasic.csv and basicspecies.csv are the default). Alter these as required. Only species in the species file will be included in your network, reactions not involving them are dropped. UMIST is gas phase only, all grain processes (including freeze out) are in the custom reaction file.
- Check MakeRates.py refers to your input files and from the MakeRates/ directory, run "python MakeRates.py"
- Copy the contents of MakeRates/outputFiles into your UCLCHEM directory.

2.2 Setup and Compiling

- If you renamed the output files in MakeRates, check parameters.f90 refers to the correct species and reaction files. Default is species.csv and reactions.csv

– Open the makefile and check the compiler path at the top refers to your preferred compiler.
– The makefile also contains a variable called physics. Here you specify the f90 file that contains the physics module. The repository contains cloud.f90,cshock.f90,hydro.f90 and turbfrag.f90. This choice depends entirely on the object being modelled, see Section 5.
– simply run "make" to compile UCLCHEM.

2.3 Running the Models

– For each run of the model, parameters.f90 needs to be altered to set the values of variables and the names of output files. For large grid you may wish to override this with a simple read in. See Section 6.
– After each change to parameters.f90, run "make". Unlike the full build, this takes negligible time.
– Run "./uclchem" to run UCLCHEM with your parameters.

3 Inputs and Outputs

All major inputs for the code can be found in parameters.f90, this is a list of every variable that one may wish to change and its value. Variables are logically grouped and highly commented so their purpose can be easily understood. The file broadly follows the pattern of: Physical properties of the gas, flags to set processes on and off, input and output files, and finally detailed parameters the user is unlikely to change.

UCLCHEM has three major outputs. The first two are the chemical abundances calculated by the code. At every timestep specified in the physics module, the code writes out the major gas properties and the fractional abundance of every single species in the network. A more machine readable output is also created, a columnated file of time, density, temperature along with the abundances of a list of species defined by the user. The length of this list is set by '*nout*' in chem.f90, and the species in the list are set in parameters.f90. Each species is referred to by it's position in the array containing all species in the code, starting from 1 (FORTRAN does not count arrays from 0). This number can be easily found from the input species file described in Section 4, by subtracting 1 from the number of the line in which the species appears (line 1 is taken by the number of species so species 1 appears on line 2). This output is extremely useful for grids ran to explore the abundances of a small number of species of interest. For those using the full output, a set of python functions to read and plot the abundances from the full output file are provided as is in the scripts/ directory.

The other major output is the analysis file. For every species appearing in the columnated output, the relative rates of the reactions contributing to its formation and destruction are calculated and written to this file. In this way, the user can explore which reactions are most important to the behaviour they observe in a given species.

4 Networks and MakeRates

Uclchem requires four files relating to the network to run: species.csv, reactions.csv, odes.f90 and evaplists.csv. Each of these is produced by MakeRates from user defined files. species.csv and reactions.csv are columnated lists of species, reactions and various related values such as masses and rate constants. odes.f90 is an automatically generated Fortran file containing every ODE in the network and evaplists.csv is a list of mantle species and the fraction evaporated in each type of desorption event. It is important to note that whilst the csv files are read in, odes.f90 is a compiled file. It is included in chem.f90 as part of the F subroutine. Therefore, the code must be recompiled via make if odes.f90 is changed. Small changes to the other files (for example, changing some alphas in reactions.csv) do not require recompilng the code as long as the number of species/reactions or the odes do not change.

The required input files are a species list, a grain/custom reaction file and a UMIST file for gas-phase reactions. For every species, the grain reaction file should contain freeze out routes and desorption reactions, see MakeRates/inputFiles/uclgrainbasic.csv for examples. The species list should be a list of every species, along with their mass and the fraction desorbed in each evaporation event if it is a mantle species. There is, unfortunately, no way to automate this. Once the files are prepared, check MakeRates.py refers to the correct input files and run ”python MakeRates.py” from the MakeRates/ directory. MakeRates will place all required files in the outputFiles/ directory. It will also print important information to the terminal. This includes notes of possible duplicate reactions and a list of species that freeze out along multiple reaction pathways.

It is important to ensure all the α values for the freeze out reactions of a single species sum to 1. For example, if there are two freeze out reactions for CO,

$$CO + Freeze \rightarrow \#CO \quad (1)$$
$$CO + Freeze \rightarrow \#HCO \quad (2)$$

and you wish 90% of CO to freeze as itself (#CO) and 10% to #HCO, the alphas should be 0.9 and 0.1 respectively. This will make the overall freeze out rate of CO correct for the model, whilst setting the proportions according to your assumptions.

5 Physics

uclchem has grown out of a all purpose chemical code used in the UCL astrophysics group with users typically making large edits to the code to model different types of objects. In an effort to make the code more accessible and reusable, uclchem is now split into two modules. The chemistry module (chem.f90) is the core code, it sets up and solves the ODEs for the chemical abundances and models processes like evaporation. The physics module controls the gas properties. This is primarily the density, temperature and visual extinction but other variables are accessible from this module.

The average user will have no need to change chem.f90 and will be able to use UCLCHEM by simply selecting the physics module that applies best to the object they wish to model. For more specific purposes, physics-template.f90 is included, which gives the bare skeleton of what must be included if a user wishes to create their own physics module. UCLCHEM is provided with 4 physics modules: cloud.f90 (**?**), cshock.f90 (**?**), turbfrag.f90 (**?**) and hydro.f90. All are described in more detail below.

5.1 Cloud

The cloud model (cloud.f90) is the 'standard' UCLCHEM model. It follows a single parcel of gas or multiple parcel distributed evenly over a line from the centre of a cloud of gas to the edge. It can be used to study gas in steady state or to follow the evolution of the chemistry as a cloud collapses and a star is formed. Cloud is typically run in two phases.

In phase 1, the parcel is initialised with an extremely low density and allowed to collapse. The gas starts in purely atomic form. The model is ran until a chosen time or density is reached. Whilst the chemical abundances are written out at every time step, the main result is the abundances at the final time step. These are written into a separate file (default is startabund.dat) and are used as the initial abundances for phase 2. Whilst phase 1 can be used to study steady state gases or look at collapse modes, it's primary purpose is to create a starting point for phase 2 without assuming initial abundances. Instead, phase 1 gives abundances that are consistent with the network.

In phase 2, collapse is generally turned off, though this is optional. The temperature profiles for cores surrounding protostars of a number of different masses are detailed in parameters.f90 and so the effects of warming the gas around a protostar can be studied. In this phase, evaporation is also introduced. At pre-determined temperatures, evaporation flags are activated so that the chemistry module will remove species from the grains. See **?** for more information on the temperature profiles and evaporation events.

5.2 C-shock

The C-shock module is the combination of UCLCHEM with the C-shock parameterization of **?**. Phase 1 can be run in the same way as the cloud model. However, in phase 2 the gas properties are entirely determined by the shock model. Evaporation occurs as a complete ejection of all mantle species into the gas phase at a time t$_{sat}$, the time when there is no further desorption in full C-shock models. The time from desorption beginning to t$_s at$ is short compared to the timescales in the model and so it implemented as a single event. The C-shock model outputs the temperature, density and neutral and ion velocities of the gas as well as the usual UCLCHEM outputs. We encourage the user to read **?** for a full technical discussion of the model as well as **?** and **?** for its combination and application with UCLCHEM.

5.3 Turbulent Fragmentation

5.4 Hydro

Hydro.f90 is a simple physics module that reads in temperature, density and time values from a file and interpolates between them. This module can be used to post-process hydrodynamic simulations to explore the chemistry. The user will almost certainly have to alter either hydro.f90 or the output of their simulations to allow for formatting differences. To build up a multi-dimensional picture of the chemistry, the user can produce files containing the time, density and temperature values for multiple positions or tracer particles and run each as separate single-point UCLCHEM instances.

6 Customization

It is impossible to predict all the changes one may wish to make to the code. However, there are a few common changes one may wish to consider.

Firstly, the parameters file is a useful way to set up a single model and to understand the variables that a user may wish to edit. It avoids the awkwardness of FORTRAN's read/write functions and the possible errors that arises from misread variables. However, for a large grid of models running simultaneously, it is cumbersome. A simple edit is to insert a read input line in main.f90 after "include parameters.f90"; an example is commented out in main.f90. This way, if a grid of models is being run varying a small number of parameters, the code can be run as,

"./uclchem < variable1 variable2 ... variableN".

Eliminating the need to edit parameters.f90 programmatically, make the executable and run for each model.

The other obvious edit is the physics, physics-template.f90 is provided to encourage this. If none of the physics modules included in the repository suit the objects you wish to modle, then physics-template.f90 provides a base from which you can produce a new module. Ultimately, it needs to set the density, temperature and visual extinction of the gas as well as setting flags for the chemistry module to know that it should evaporate the mantle species. Anything else is entirely optional. We would like to encourage users who create physics modules to share them in the main repository once they've reached a usable state.

7 Troubleshooting

Bad Parameters If UCLCHEM crashes on running, its often due to incorrect values in the parameter file. In particular, if firstRun=0 when running

UCLCHEM for the first time, the program will crash as it will not find a file with initial abundances. Equally, if the reactions or species files are not named correctly in parameters.f90, the program will fail.

Convergence Failure UCLCHEM uses DVODE to integrate the ODEs for the species' abundances. Variables controlling DVODE can be found in paramters.f90, set to values that are optimal for simple cloud models in phase 1. Most DVODE related errors and warnings can be fixed by adjusting these parameters. In particular, abstol and reltol control the integration accuracy. The default values function well for a variety of standard model runs, achieving fast runs without sacrificing accuracy compared to smaller tolerances. However, for larger networks or models where variables change more quickly, lowering these tolerances will improve stability.