UCLCHEM User Guide

August 9, 2018

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 uclspeciesbasic.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 (freeze out) are in the custom reaction file. If you add species, ensure there is at least a freeze out reaction for each species included in the grain 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 src directory.

2.2 Setup and Compiling

- Open the makefile and check the compiler path at the top refers to your preferred compiler. Add any preferred flags here too.
- 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

- Run UCLCHEM as "./uclchem example.inp"
- example.inp should be a list of any variables from defaultparameters.f90 that you wish to change

3 Inputs and Outputs

All major inputs for the code can be found in defaultparameters.f90, this file contains explanations of each variable in the comments and has a default value. UCLCHEM takes an input file which should contain a list of any variable in defaulparameters.f90 that the user wishes to change from the default. Examples of these input files are provided in the examples directory along with the outputs that should be obtained when running them with the cloud.f90 model.

Variables in defaultparamters.f90 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 two major outputs. The first is the "full" output including the major gas properties and the fractional abundance of every single species in the network. This is written at every timestep. Optionally, a more machine readable output can be created. If a list of species is given, a columnated file of time, density and temperature along with the abundances of those species will be written. For those using the full output, a set of python functions to read and plot the abundances from the full output file are provided in the scripts/ directory.

3.1 Phases

Many models use multiple "phases" and there are three inputs relating to this. The code must be given some starting abundances for every species and this is a choice on the part of the user. It is typical therefore, in UCLCHEM publications, to use a two phase approach. In the first phase, a gas of purely atomic elements is modelled and in the second, the science model is run starting from the abundances acquired in phase 1. Thus, to model a process in a molecular cloud,

a phase 1 is run from a density of 100/cm3 with collapse=1 until the density reaches the required value for the molecular. This produces a set of abundances that is consistent with the network that can be read in as the starting point for the science model (phase 2).

The parameter choice to achieve this is to set phase=1,readAbunds=0 and abundFile to an output file that can be used to start the next model. The parameter "phase" in most physics modules will turn off the physics that the module was written for and run a simple cloud instead (ie in the C-shock module, phase=2 allows the shock to run and phase=1 does not). readAbunds determines whether abundFile is read to get starting abundances (readAbunds=1) or atomic elements are assumed (readAbunds=0). If readAbunds=0 and abundFile is set, then the final timestep of the model will be written to abundFile so it can be used as a starting point for other models.

4 Making a Network

Makerates is a python script that will produce network.f90 and odes.f90 as well as species.csv and reactions.csv. The f90 files provide the necessary arrays to describe the network and the code to produce the rate of change of the abundances for the ODEs. These are required for UCLCHEM to run and the code must be recompiled whenever they are updated. The csv files are simply lists of every species and reaction along with important information such as the reaction rate coefficients. These can be inspected by the user or used to make tables.

The input files for Makerates are in Makerates/inputFiles/ and include a list of species, a gas phase network (generally UMIST 12) and a further list of any user defined reactions. Examples of these are included, the inputFiles folder contains uclspeciesbasic.csv and uclgrainbasic.csv which along with UMIST12 produces a large gas phase network with a minimal surface network. For every species, the grain reaction file should contain at least one freeze out route. The species list should be a list of every species, along with their mass. The other values in the species file are only required for surface species, the most important of which are the binding energy and enthalpy.

The grain surface network should in the most minimal case contain a freeze out route for every species. The user may wish to define multiple freeze out routes, this allows for processes such as hydrogenation to be accounted for. For example, the freeze out of CO may follow two routes:

$$CO + Freeze \rightarrow \#CO \tag{1}$$

$$CO + Freeze \rightarrow \#HCO \tag{2}$$

so that some CO immediately hydrogenates. If we assume that 90% of CO to freezes 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 the assumptions.

More complicated surface chemistry can be obtained in two ways. Firstly, Makerates and UCLCHEM will treat any reaction with two surface reactants exactly the same way as a gas-phase two body reaction. It may be possible to parameterize the rate of a surface reaction using the alpha, beta and gamma values of the Kooji-Arrhenius equation. Secondly, Quénard et al. [2018] describes a modification to UCLCHEM to include a diffusion formalism for the reaction of surface species. These are recognized by the code when a third reactant called either "CHEMDES" or "DIFF" is included in a surface reaction.

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 [Viti et al., 2004], cshock.f90 [Jiménez-Serra et al., 2008], turbfrag.f90 [Holdship and Viti, 2015], collapse.f90 [Priestley et al., 2018] 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 Viti et al. [2004] 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 Jiménez-Serra et al. [2008]. 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_sat 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 Jiménez-Serra et al. [2008] for a full technical discussion of the model as well as Viti et al. [2011] and Holdship et al. [2016] for its combination and application with UCLCHEM.

5.3 Collapse

This is the module produced for the work done in Priestley et al. [2018]. By setting collapse to a value of 2 or higher, different parameterizations of a collapsing core can be accessed. These are described in the comments at the top of the module code. This code is best run using a single point phase 1 where a simple freefall collapse (collapse=1) is done to produce gas of the same density as the initial core centre and a multi-point phase 2 where collapse is set to some higher value. The multi-point is required as the radius of the core (rout) is divided and the chemistry is sampled at each point. The code does take into account the reduction in rout as the core collapses and each parcel migrates inwards with the collapse.

5.4 Turbulent Fragmentation

This is the module for the work done in Holdship and Viti [2015]. In phase 2, the density of the gas is increased according to an isothermal, non-magnetic shock.

5.5 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.

Bibliography

Jonathan Holdship and Serena Viti. Chemical tracers of pre-brown dwarf cores formed through turbulent fragmentation. *Monthly Notices of the Royal Astronomical Society*, 455(3):1–10, nov 2015. ISSN 0035-8711. doi: 10.1093/mnras/stv2460. URL http://arxiv.org/abs/1510.06576.

Jonathan Holdship, Serena Viti, Izaskun Jimenez-Serra, Antonios Makrymallis, and Felix Priestley. Chemistry in C-shocks with UCLCHEM. *ApJ*, ?(?):?, ? 2016. URL http://jonholdship.github.io/uclchem.

I. Jiménez-Serra, Paola Caselli, J Martín-Pintado, and T.W. Hartquist. Parametrization of C-shocks. Evolution of the sputtering of grains. *Astronomy & Astrophysics*, 482(2):549–559, 2008.

F. D. Priestley, S. Viti, and D. A. Williams. An Efficient Method for Determining the Chemical Evolution of Gravitationally Collapsing Prestellar Cores. *AJ*, 156:51, Aug 2018. doi: 10.3847/1538-3881/aac957.

D. Quénard, I. Jiménez-Serra, S. Viti, J. Holdship, and A. Coutens. Chemical modelling of complex organic molecules with peptide-like bonds in star-forming regions. *MNRAS*, 474:2796–2812, feb 2018. doi: 10.1093/mnras/stx2960.

Serena Viti, Mark P. Collings, John W Dever, Martin R S McCoustra, and David A Williams. Evaporation of ices near massive stars: models based on laboratory temperature programmed desorption data. *Monthly Notices of the Royal Astronomical Society*, 354(4):1141–1145, 2004.

Serena Viti, I. Jiménez-Serra, J A Yates, C. Codella, M Vasta, Paola Caselli, Bertrand Lefloch, and Cecilia Ceccarelli. L1157-B1: Water and ammonia as diagnostics of shock temperature. *The Astrophysical Journal Letters*, 740(1): L3, 2011.