1 Installation

1. Download source code from `https://github.com/uclchem/uclpdr`, extract into new folder.

2. The Github version compiles using `ifort`, and also requires OpenMP and Sundials to run. On a UCL machine, you might need to make sure it knows where to find the compiler and OpenMP libraries before doing anything else. Edit `.login` in your home directory and add
`setenv PATH $PATH':/opt/intel/Compiler/11.1/046/bin/intel64/'`
and
`setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH':/opt/intel/Compiler/11.1/046/lib/intel64/'`
then run `source .login` on the command line to update the paths.

3. Go into `uclpdr/Source` and edit the INCLUDES and LIBRARIES lines of the Makefile to point to a Sundials installation. On a UCL machine you can use
`INCLUDES = -I/home/fdp/sundials/include`
`LIBRARIES = -L/home/fdp/sundials/lib`
and avoid installing it yourself.

4. Type `make` and hopefully the code should compile without anything crashing.

2 Running UCL_PDR

2.1 Chemical Network

The code needs a chemical network to run, and has to be recompiled if you want to change it. The Chemical-Networks folder has a bunch of examples, and `makerates.py` which you need to use to create new ODE and Jacobian files if you change anything. You need four files for each network: `species.dat` is a list of the chemical species and their initial abundances, `rates.dat` is a list of chemical reactions and their temperature dependences, `odes.c` and `jacobian.c` are source files the code needs to calculate the chemical abundances. To change to network `blah`, copy `blah-species.dat` and `blah-rates.dat` to `Datafiles/Chemical-Network/species.dat` and `Datafiles/Chemical-Network/rates.dat`, and `blah-odes.c` and `blah-jacobian.c` to `Source/odes.c` and `Source/jacobian.c` and then `make` again. If you want to change the values (initial abundances, reaction rates) you can just edit the numbers in `Datafiles/Chemical-Network` without having to recompile - if you want to add or remove species or reactions, you need to run `makerates.py` on the new species and rates files to generate the ODE network again.

2.2 Model parameters

Most of the other input parameters are read from `Input/model-parameters.dat`. The folder also contains example cloud and radiation field files. The size and density of the model cloud are given in the cloud file, the incident UV and X-ray flux are given in the radiation file. Other important parameters are the output prefix (the name of the output files), cosmic ray ionization rate, A_V/N_H conversion rate (determines how the density/size of the cloud converts to extinction), and the coolant data files (which species the code calculates cooling rates/line emissivities for). Available cooling species are in `Datafiles/Collisional-Rates` - anything in there can be added, and additional species are available from `http://home.strw.leidenuniv.nl/~moldata/`. Setting a minimum allowed temperature can also sometimes be useful.

2.3 Other inputs

Some effects are hardcoded into the source files so can only be changed there. Possibly important ones are:

X-rays: The X-ray spectrum is by default a blackbody. The temperature and energy range are defined in `xray_cross_sections.f90` in the `CALCULATE_XRAY_PROPERTIES` subroutine. The shape of the spectrum

can be changed by editing the `XRAY_FLUX` function. The `TOTAL_XRAY_CROSS_SECTION` function calcualtes the absorption cross-section from the elemental abundances, so this might need changing for highly non-solar metallicities. If you want to use X-rays make sure the chemical network actually includes X-ray reactions.

H_2 formation rate: Defined in `h2_formation_rate.f90`. There are a bunch of rates from the literature defined at the end of the file - uncomment whichever one you feel like using.

Heating rates: In `heating_rates.f90`. The total heating rate is defined at the end of the subroutine - there are three photoelectric heating rates available, uncomment whichever you want to use. The standard cosmic ray heating rate is determined by the H_2 density so won't be accurate if the gas is mostly atomic.

Dust: The dust abundance and properties are defined in a bunch of different places, annoyingly. Grain radius (defined in `model-parameters.dat`) and dust density (defined as a dust-to-gas ratio in `read_particles.f90`) are used to calculate the gas-grain collisional heating/cooling. The radius also affects freeze-out (if included), and the density affects the calculation of line emissivities (due to thermal dust emission). Some of the H_2 formation rates depend on the total dust surface area per hydrogen nucleus. There are also the extinction properties and albedo in `model-parameters.dat`.

3 Output files

The code creates a bunch of output files named `<modelname>.xxxx.out` where `<modelname>` is the one you define in `model-parameters.dat`. The important ones are:

prop.out: Has the main properties like density, gas and dust temperature and radiation field strength for each point in the cloud.

av.out: Physical distance and extinction into the cloud.

abun.out: Chemical abundances of species in the reaction network.

emis.out: Emissivity of each included emission line in erg cm^{-3} s^{-1}.

cool.out: Total cooling rate for each species in erg cm^{-3} s^{-1}.

heat.out: Total heating rate for each heating mechanism in erg cm^{-3} s^{-1}. Gas-grain 'heating' will generally be negative so is actually a cooling mechanism.

4 Common issues

The code's error messages are usually quite helpful. Anything about a Gauss-Jordan solver means you've probably got a coolant with two energy levels the same (the OH^+ file from the database has this issue). You also need to update the 'number of coolants' parameter whenever you change them. First thing to check is usually that `odes.c/jacobian.c` match the `species.dat` and `rates.dat` files in `Datafiles/Chemical-Network`. If you put particularly insane values for the X-ray flux in it can stop the chemistry solver from ever converging, especially if you reduce the minimum energy.