Krome School 2014: Exercises

17-19 September

KROME Team

Problem 1

Simple chemical network and heating function

The aim of this exercise is to introduce the participant to the use of the chemical network file that represents the input of KROME. We strongly recommend to use the flag **-pedantic** (which enable a pedantic Makefile) during the whole exercise in order to avoid bad surprises. The pedantic Makefile enables a -O0 compilation and includes compiler flags to trap common bugs. Add **-useN** to run in number density.

Part 1: Preparing the chemical network
We want to study the evolution of a simple chemical network with a few reactions, namely

1. $H_2 + CR \rightarrow H_2^+ + e^-$
2. $H_2 + H_2^+ \rightarrow H_3^+ + H$
3. $H_3^+ + CO \rightarrow HCO^+ + H_2$

The rate coefficient for the first reaction is 7×10^{-17} s^{-1}, while for the other two reactions is 10^{-9} cm^3 s^{-1}. Prepare the chemical network using different @format tokens to avoid blank parts.

Part 1 A: Writing the code
Use the code template test.f90 provided and build the rest.
Initial conditions are: $n_{H_2} = 10^4$ cm^{-3}, $n_{CO} = 10^{-4}\ n_{H_2}$ cm^{-3}, and all the other species are 10^{-40} cm^{-3}. Since the reaction rate coefficients are temperature independent, T_{gas} can be any temperature, we have adopt 300 K (it will be useful later). Evolve the system for 10^{10} yr, with increasing time-step (e.g. from dt= 10^{-6} yr increasing by 1.1 times every loop, i.e. dt = dt*1.1).

Note: a useful common variable is spy=krome_seconds_per_year.

Part 1 B: plot the results
Plot the time evolution of CO, H_2^+, H_3^+, and H_2. Gnuplot users should employ the variables contained in the species.gps file.

Part 1 C: Print the flux
Print the flux of the reactions at approximately t = 10^6 yr using the krome_print_best_flux subroutine (if needed, see suggested pseudocode below). Employ the list_user_functions.py utility in the build folder to obtain the arguments of the interface of the function.

[Type python list_user_function.py and grep some text from the function.]

Solution The results at 10^6 yr should look like this:

```
2 H2 + H2+ -> H3+ + H      0.69691245E-12
1 H2 -> H2+ + E            0.69691245E-12
```

```
1: if t>=10^6*krome_seconds_per_year and boolean then
2:     print flux
3:     boolean = .false.
4: end if
```

Algorithm 1: print flux pseudo-code

Part 2: Expand the network
Add now an additional reaction

$HCO^+ + H_2 \rightarrow H_3^+ + CO$

with a rate coefficient of $\alpha \times 3.88 \times 10^{-10}$ cm^3 s^{-1}, where α is a free parameter.
Check what happen to the evolution when α changes to 10^{-5}, 10^{-2}, and 1 using the token @common.
You have to re-run the KROME pre-processor, so the option **-noExample** should be useful to avoid a replacement of test.f90.

To initialize the variable use the subroutine krome_set_user_alpha(argument). Plot the same species as in the Part 1.

Part 3: Heating from CRs
Add heating from cosmic rays, considering that the first reaction is a CR ionization. Use the tokens @CR_begin and @CR_end.

Problem 2

Advanced cooling functions and LAMDA database

This exercise shows how to use the custom cooling functions. We strongly recommend to use the flag **- pedantic** during the whole exercise in order to avoid bad surprises. Moreover, add **-useN** to run in number density.

Part 1: Preparing the chemical network
The backbone of the network (network_start) is provided. The main problem here is to add a custom function for CO from the equations in the file CO_appendix.ps in the exercise folder. Take a look at the CO rotational cooling considering the collisions with H_2 as described in eqn.A25 (page 146) in the attached file. For the sake of simplicity we provide the equations translated in F90 (see provided file code_block):

```
ntot = get_Hnuclei(n(:))
na = 0.5d0*([total amount of H]+1.414d0*[total amount of H2])
ncr = 3.3d6*(Tgas*1d-3)**0.75

speed = sqrt(kvgas_erg/2d0)
factCO = 1d0 + na/ncr + 1.5d0*(na/ncr)**0.5
sigma = 3d-16*(Tgas*1d-3)**(-0.25)

!total cooling
ntot*(boltzmann_erg*Tgas)*sigma*speed*[total amount of CO]/factCO
```

Note that some parts between square brackets should be replaced with an appropriate expression. To do this it is important to remark that the internal array for the species is n(:), and the internal indexes for species are without the krome_ prefix, so for example HCO has idx_HCO instead of krome_idx_HCO. Note also that the variables should be indicated with the token @var, and that the total cooling uses @cooling token. Do not forget to define the block using @cooling_start and @cooling_stop. For instance

```
@cooling_start
@var: your variable
@cooling: your cooling function
@cooling_stop
```

Part 1 A: Writing the code
In this exercise the output of the code should be the amount of cooling in erg cm^{-3} s^{-1} at different temperatures considering the chemical equilibrium. To do this you should write a loop on T_{gas} in the range 10-10^4 K, with jmax=100 logarithmic steps, for example

```
do j=1,jmax
        Tgas = 1d1**((j-1)*(4d0-1d0)/(jmax-1)+1d0)
        ...
end do
```

The total density of H is 10^6 cm^{-3}, while the other species are defined through the metallicity $Z = -1$ by employing the function krome_scale_Z. To check its arguments run the list_user_functions.py utility (you already know how to use it).

Problem 2 continued on next page...

Now that you have rescaled the metals you should find chemical equilibrium. For this you should turn off the thermal evaluation by using

call krome_thermo_OFF()

that needs the **-useThermoToggle** flag when you run KROME. This feature switches off the cooling and solves only the differential equations for the chemical species: in this way you will find the equilibrium at constant temperature.

Then you can integrate KROME for a long time, let's say dt=10^8 yrs

call krome(x(:),Tgas,dt)

This is enough time for this network to reach a steady state equilibrium.

Finally you want to write the cooling function into a file. For this purpose use the krome_get_cooling_array that returns an array of size krome_ncools, containing the cooling functions. The position in the array is krome_idx_cool_custom. Since this chemical network contains reactions involving cosmic-rays, somewhere in the test.f90 (before the call to KROME) you need to add the cosmic-rays initialization (to $\zeta_{CR} = 1.3\times 10^{-17}$ s^{-1}), namely

call krome_set_user_crate(1.3d-17)

Part 1 B: Running KROME
Now it is time to run KROME. Use the appropriate flags indicated above and marked in boldface.
Use **-noExample** to avoid to replace the test.f90 file just prepared. Once compiled and run, display the results in a plot showing the cooling function vs $T_{\rm gas}$.

Part 1 C: Plot chemistry
Plot the chemical abundances of CO, H, H_2, and compare them to the results obtained from Part 1 B (i.e., the cooling function). This is useful to understand the importance of the single cooling functions over time.

Part 1 D: Compare cooling functions
Compare the cooling from CO employed above with the cooling from H_2. You should re-run KROME with the option which enables H_2 cooling (**-noExample** is your friend, you don't want to re-write your test.f90!).

Note: krome_idx_cool_H2 is the position of the cooling function in the array.

Part 2: Compare the cooling with LAMDA data (optional)
We now want to compare the approximate CO cooling employed in the previous test with the results obtained by using the cooling from the LAMDA database. To do this add the following option

-coolFile="tools/coolCO.dat"

to indicate that KROME should take into account the molecular transitions data from that file. If you want to check the availability of the cooling function just add

-cooling=?

Problem 2 continued on next page...

and run KROME. A list of the cooling functions will be provided, including the new one found in the data file above. Replace "?" with CO. To reduce the computational cost (and the compiling time) reduce the number of CO rotational lines using **-coolLevels=32** and run KROME again (**-noExample** is your friend again).

Now with the same method used in Part 1 compare the two cooling functions, remembering that in KROME any cooling from a data file has an index equal to the integer variable krome_idx_cool_Z.

Problem 3

The aim of this exercise is to explain how bin-based photochemistry works in KROME. In order to do this we model a 1D Strömgren sphere, i.e. the evolution of a ionization front. As usual do not forget to put **-pedantic** for a safer test, as well as the **-useN** option. This test will take about one minute to run, if you want to speed-up remove the **-pedantic** option.

Part 1: The physics
We want to understand what happens when a stellar source is embedded in a cloud of gas with constant density $n_{\rm tot} = 10^{-3}$ cm^{-3}. The temperature of the star is 30,000 K to mimic a B0 star, while its radius is taken to be $R_\star$=66.1 $R_\odot$ ($R_\odot$=7×10^{10} cm) to ensure an emission of ∼5×10^{48} photons per second. The emission is isotropic and follows a black-body (BB) spectrum. To ensure the correct number of photons use an energy range between 13.6 and 13.8 eV. The box of the simulation is $L = 6.6$ kpc (1 pc=3.085×10^{18} cm) and we will divide it into a linearly equally-spaced grid with ngrid=300 points, e.g.

```
r = (i-1)*(rmax-rmin)/(ngrid-1) + rmin
```

You can set the first point at 1 pc. This grid is represented by a 2-dimensional array that contains the chemical species at each grid point, e.g. xall(ngrid, krome_nmols). The initial conditions are $T_{\rm gas}$=10^4 K, and $n_{\rm H} = n_{\rm tot}$, $n_{\rm H^+} = n_{\rm e^-} = 1.2\times10^{-3}\ n_{\rm tot}$.
To simplify the exercise we use an explicit solver and provide a test.f90 template. The backbone of the solver is a loop on grid points nested in a loop on time as:

```
dt = 0.1 yr
LOOP time
  dt = dt * 1.01
  t = t + dt
  LOOP grid for i
    x(:) = xall(i,:)
    call KROME(x(:),Tgas,dt)
    xall(i,:) = x(:)
  END LOOP grid
  if(t>tmax) break loop on time
END LOOP time
```

Part 1 A: The chemical network
The chemical network is initially based only on the recombination rate:

$$\rm H^+ + e^- \rightarrow H + \gamma$$

with the rate coefficient $2.59\times10^{-13}(10^4/T_{\rm gas})^{0.7}$ cm^{3}s^{-1}. Without any radiation the gas will recombine, verify it plotting the radial profile of H and H$^+$ at t ∼tmax = 30 Myr.

Part 1 B: Adding photochemistry
Now we want to add the radiation and the photoionization process

$$\rm H + \gamma \rightarrow H^+ + e^-$$

In this case KROME can take the cross section from its internal database. Just specify "auto" as the reaction rate in the network file. Do not forget to indicate that the rate is a photorate, using the token @photo_start and @photo_end.

Problem 3 continued on next page...

As an example

```
@photo_start
1,R,P,P,auto
@photo_end
```

To add the photochemistry to KROME first run it using **-photoBins=1** to specify that you are interested in just one bin of monochromatic radiation. KROME will enable all the machinery for handling photochemistry. In the test.f90 file you should consider an emitting BB source at $T_{bb} = T_\star$. KROME has several functions in the module krome_user. For a BB you can use the subroutine krome_set_photoBin_BBlog. Check the list of functions (you should know how to do) to see which arguments you need. Decide a narrow energy interval around 13.6 eV to mimic monochrome radiation at the Lyman edge. In this way you are telling KROME what energy you are using.
It is now time to rescale the energy according to the distance r, using krome_photoBin_scale. The scaling factor is $\frac{1}{4\pi}\pi(\frac{R_\star}{r})^2$. The variable $\frac{1}{4\pi}$ is used to cope with the internal 4π factor that assume isotropic radiation to compute the photoionization cross sections.

We are considering here an optically thin case! Run the model and plot the radial profile (linear-log).

Part 2: Optically thick case
Now the tricky part.

When a photon reaches the grid point N it has a non-zero probability of being stopped by the previous $N-1$ volumes of gas. Namely: the optical depth. For this reason at each grid point you should take into account the "history" of the radiation from the star's surface to the grid point N.
The function krome_get_opacity_size returns an array of size krome_nmols that contains the optical depth (i.e. τ) for each frequency bin. The arguments include the size of the crossed gas volume (i.e. the size of the grid cell, $r_i - r_{i-1}$). For each grid point, store this in a bi-dimensional array (that should be defined!) as

```
opt(ngrid,krome_nPhotoBins)
```

The optical depth for the i-th grid point is then

```
do j=1,krome_nPhotoBins
  op(j) = product(exp(-opt(1:i,j)))
end do
```

Now you can rescale the flux by

```
op(:)*(1/4*pi)*pi*(Rstar/r)**2
```

using this time the function krome_photoBin_scale_array instead of krome_photoBin_scale (since you want to do it bin by bin).

Plot the radial profile at t_{end} ∼30 Myr and t_{end} ∼500 Myr. Compare your results with the figure below (i.e. Fig. 8 of Iliev+2006).

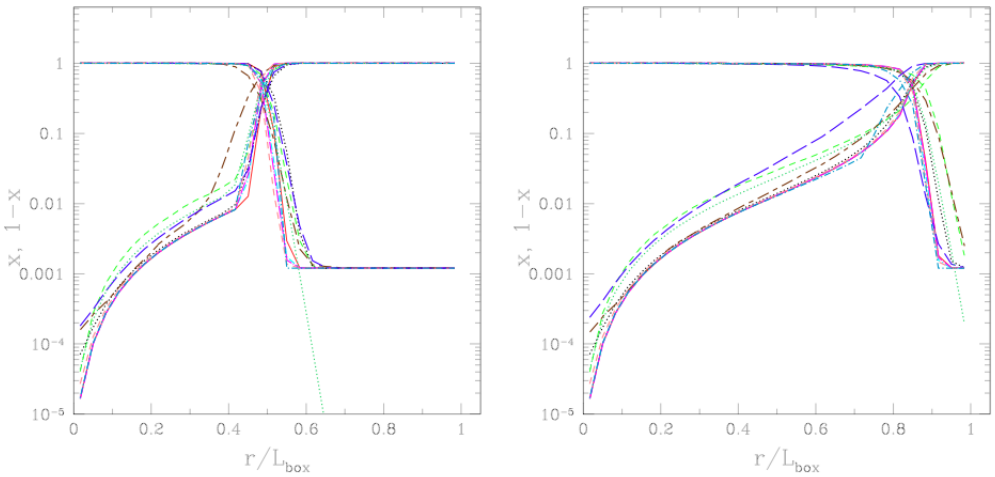

Figure 8. Test 1 (H II region expansion in an uniform gas at fixed temperature): Spherically-averaged profiles for ionized fractions x and neutral fractions $x_{\rm HI} = 1 - x$ at times $t = 30$ Myr and 500 Myr vs. dimensionless radius (in units of the box size).

Problem 4

Surface reactions with size-dependent dust modelling

The aim of this exercise is to understand how the size-bin-based dust distribution works in KROME. To do this we will implement a chemical network on the dust surface. The difficulty of this exercise is that some of the reaction rates depend on the size of the dust, and hence you need to write them according to the way KROME handles the dust bins. The equations sketched here below are from Dulieu+2013 (http://adsabs.harvard.edu/abs/2013NatSR...3E1338D) and Hocuk+2014 (http://arxiv.org/abs/1408.5029).

Recap on dust variables:
`boltzmann_erg`, Boltzmann constant, erg/K (included in `krome_constants` module)
`ndust`, size of the dust array
`xdust(:)`, dust abundances (array of size `ndust`, cm^{-3})
`krome_dust_asize(:)`, size of the dust (array of size `ndust`, cm)
`krome_dust_asize2(:)`, square of the size (array of size `ndust`, cm^2)
`krome_dust_T(:)`, dust temperature (array of size `ndust`, K)

Part 1 A: The chemical network
The initial network here is rather simple. It is based on 2 reactions. The first is a gas→gas process, while the second is a gas→dust process (adsorption, or shortly process A):

1. $\mathrm{O} + \mathrm{O} \rightarrow \mathrm{O}_2$ $\quad \mathrm{rate}_1 = 4.9\times10^{-20}(T_{\mathrm{gas}}/300)^{1.58}$
2. $\mathrm{O} \rightarrow \mathrm{O}_{\mathrm{dust}}$ $\quad \mathrm{rate}_2$=see below

The first rate coefficient is indicated in the list above, while the second for the jth reactant (in our case just oxygen) is

$$k_{\mathrm{ads},j} = v_j \sum_i x_{d,i}\pi a_i^2 \, S(T_{\mathrm{gas}}, T_{\mathrm{dust},i}) \,, \tag{1}$$

where $v_j = \sqrt{8k_b T_{\mathrm{gas}}/\pi/m_j}$, $x_{d,i}$ is the amount of dust in the ith bin, a_i the size of the ith bin, and S is the sticking coefficient defined by

$$S(T_{\mathrm{gas}}, T_{\mathrm{dust},i}) = \left(1 + 0.4\left(\frac{T_{\mathrm{gas}} + T_{\mathrm{dust},i}}{100}\right)^{0.5} + 0.2\frac{T_{\mathrm{gas}}}{100} + 0.08\left(\frac{T_{\mathrm{gas}}}{100}\right)^2\right)^{-1} . \tag{2}$$

In the network file the processes A ($\mathrm{O}\rightarrow\mathrm{O}_{\mathrm{dust}}$) should look like this:

```
2,O,O_dust,fA(m(idx_O),Tgas)
```

where `fA` is a custom function and `m(idx_O)` is the mass of oxygen in grams. The custom function `fA` will be prepared later.
To get the array containing the masses of the species in grams write in your network file the following statement

```
@var:[nspec] m = get_mass()
```

where `nspec` (i.e. the total number of species) is the size of the array `m(:)`.

Part 1 A: Run KROME
Once you wrote the two-reactions chemical network, run ./krome with `-useN` option for the number density interface, `-pedantic` to trap problems, and then set 10 dust bins, together with the dust type (carbon-based).

To do this use `-dust` option (check the help, `./krome -h`, to understand how).

Part 1 B: Preparing the custom function
We did not write the custom function `fA` yet. We suggest to use the `krome_user_commons` module to provide this function. For example the function for adsorption could be something like the following pseudocode:

```
function fA(mass,Tgas)
 use krome_commons
 use krome_constants
 rate = 0
 vx = compute thermal speed using the mass variable
 LOOP on dust bins
  stick = equation for stick using krome_dust_T array
  rate = rate + expression that depends on krome_dust_asize2 and xdust arrays
 END LOOP
 fA = rate
end function
```

Some useful information:

1. constants stored in the module `krome_constants` are without the `krome_` prefix (e.g. `pi` instead of `krome_pi`).
2. the index for a gas species, e.g. CO is `krome_idx_CO`, while for its counterpart on surface we have `krome_idx_CO_dust`.
3. eqn.(2) depends on the $T_{\rm dust,i}$, which depends on the bin size.

Part 1 C: The test.f90
This one-zone model (test.f90 file provided) should evolve for 10^9 years and the initial conditions are

- $n_{\rm tot} = 10^6$ cm^{-3}
- $n_{\rm H} = n_{\rm tot}$
- $n_{\rm O_2} = 10^{-4}\ n_{\rm H}$
- $n_{\rm O_{dust}} = 3 \times 10^{-4}\ n_{\rm H}$
- $T_{\rm gas} = 300$ K
- $T_{\rm dust} = 30$ K

All the other species are set to 10^{-40}. The dust-to-gas ratio is 10^{-2}.
The only thing you need to do now is to add the dust initialization routines by replacing the comments in the test.f90 file: use `krome_set_dust_distribution` and `krome_scale_dust_gas_ratio`. After doing this compile, run, and plot the time evolution (log-log) for O(gas) and O(dust).

Part 2: Evaporation: introduction

After adsorption we add evaporation (E), which is the opposite process.

This process needs another reaction in our chemical network

- $\rm O_{dust} \rightarrow O$

Before discussing the rate coefficients it is better to understand the equations. The machinery is similar to Part 1 above. The evaporation rate is given by

$$k_{evap,j} = \nu_0 \sum_i \left(\mathcal{F}_{bare,i} \exp\left(-\frac{E_{bare,j}}{T_{d,i}}\right) + \mathcal{F}_{ice,i} \exp\left(-\frac{E_{ice,j}}{T_{d,i}}\right)\right) \quad \mathrm{s}^{-1}, \tag{3}$$

where the sum runs over the dust bins, and j is the species index (in this case only oxygen). The other parameters are $E_{bare,j} = E_{ice,j} = 1.7 \times 10^3$ K, $\nu_0 = 10^{12}$ Hz, and

$$\mathcal{F}_{ice,i} = \min\left[\frac{\mathrm{n_{H_2Odust}}}{\phi_\mathrm{i}}, 1\right]. \tag{4}$$

The bare fraction of the dust is obtained by

$$\mathcal{F}_{bare,i} = 1 - \mathcal{F}_{ice,i} \tag{5}$$

and

$$\phi_i = \pi a_i^2 \frac{4}{a_\mathrm{pp}^2} \mathrm{xdust_i}\,, \tag{6}$$

where $a_\mathrm{pp} = 3{\times}10^{-8}$ cm. Note here that Eqn.(6) depends on a_i, which is bin-based.

Part 2 A: Preparing the network and the custom reaction
In the network add evaporation ($\mathrm{O_{dust}} \rightarrow \mathrm{O}$) as

```
3,O_dust,O,fevap(1700d0, 1700d0, n(idx_H2O_dust))
```

For this exercise we assume that the numerator of Eqn.(4) - density of water on dust - is constant (nH2O_dust = 1), hence set `x(krome_idx_H2O_dust) = 1d0` in the test.f90 file. In the `krome_user_commons` module add the code using this pseudocode as template:

```
function fevap(Ebare,Eice,nH2O_dust)
 use krome_commons
 use krome_constants
 rate = 0e0
 LOOP on dust bins
  phi = expression that depends on krome_dust_size
  Fice = min(nH2O_dust / phi, 1)
  Fbare = 1 - Fice
  rate = rate + expression that depends on krome_dust_T array
 END LOOP
 fevap=rate
end function
```