

Krome School 2015: Exercises

20-24 July

Credits:

T. Grassi, S. Bovino, J. Ramsey, M. Küffmeier

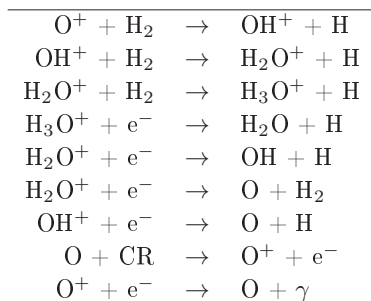


Problem 1: Preparing a chemical network

The aim of this exercise is to understand how to prepare a chemical network and what are the possible caveats related to this. We choose a simple H_2O chemical network to show the evolution and the influence of the different parameters on the evolution of the chemical species.

Part 1: Basic chemical network

In the first part we consider the chemical network listed here:



Before writing any code, draw a sketch of the network to understand how it should work. The suggestion here is to use H_2 (being almost unaffected by the chemistry of oxygen) and e^- as “companion reactants”, as shown in Fig. 1. This step will help you understand the behaviour of the network in the subsequent parts of the exercise.

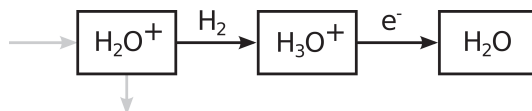


Figure 1

Part 2: Write the chemical network for KROME

In this part, we will prepare the network for KROME. Some of the reactions are already provided in `networkH20.ntw`; some need to be added. As you can see, most of the reaction rate coefficients can be automatically loaded from the KROME database using `auto` instead of writing the rate in F90 style. However, for cosmic ray (CR) ionization, we must explicitly write the rate coefficient as it depends on an external quantity (i.e., the CR flux). In this case, the rate for the CR ionization of oxygen is $k_{CR} = 2.8\xi_{CR}$ where $\xi_{CR} = 1.3 \times 10^{-17} \text{ s}^{-1}$ is a common variable that can be set in the `test.f90` file; more on this later. In KROME, a common variable can be added to a network by using the `@common` “token”. Common variables allow “communication” between the network and your main (`test.f90`) file. More details on `@common` and other tokens can be found in the wiki at bitbucket.org/tgrassi/krome/wiki/quick1.

Note: the photon γ and the cosmic rays CR should be omitted from your network file.

Part 3: The main file

If you now run the KROME python with the option `-n networkH20.ntw`, KROME will raise a warning like “*sinks found, check your network*”; this happens because, as you can see from your sketch (part 1), this network forms water without destroying it! You can avoid this warning with the `-noSinkCheck` option. We also strongly suggest you use the `-pedantic` option to generate a pedantic Makefile in the `build/` folder. If you now run the KROME python again with these additional options, it will automatically create all the necessary files in the `build/` folder, including a pre-made main file called `test.f90`. This file provides the

bare-minimum algorithm that you will modify to suit your problem. In this case, we need a loop that dumps the evolution for every time interval, `dt`.

Here is some example pseudo-code for this exercise:

```

initialize species and temperature
initialize cosmic ray flux with the function in krome_user
t = 0
dt = 1e-2 yr
do
  dt = dt * 1.1
  t = t + dt
  call krome (x(:), Tgas, dt) ! x in cm**-3, Tgas in K, dt in seconds
  write output
  if(t>1e6 yr) break loop
end do

```

The initial conditions for the simulation are $n_{\text{H}} = n_{\text{tot}} = 10^8 \text{ cm}^{-3}$, and $n_{\text{H}_2} = 0.1 \cdot n_{\text{tot}}$. The neutral oxygen abundance is scaled using the `krome_scale_Z` subroutine, with $\log(Z/Z_{\odot}) = 0$ (i.e. solar metallicity). This subroutine is found in the `krome_user` module, which contains several front-end functions. To retrieve more details on these functions, you can invoke the python file `list_user_functions.py` in the `build/` folder (`-h` for details on its usage). The end time for the simulation is $t_{\text{end}} = 10^6 \text{ yr}$, and the temperature is set constant at $T_{\text{gas}} = 50 \text{ K}$ for the entire simulation.

Once you have modified the `test.f90`, you can compile `KROME` via the provided `Makefile` and then run the executable `./krome` in the `build/` folder. If you plot the evolution of water you should obtain something like Fig. 2 (left). How you choose to plot the results is up to you, but `KROME` does provide a Gnuplot template script `plot.gps` in `build/`.


Note that, as discussed in the morning lecture, `KROME` generates a Gnuplot file (`species.gps`) that contains the indices of the species in a Gnuplot-friendly style as well as (`species.log`) that contains the names and indices in a more generic fashion (for non-Gnuplot users).

Part 4: Key reaction rates

When dealing with chemical networks, it is important to understand what is occurring during the evolution. One of the most effective diagnostic tools is to rank the fluxes of the different reactions, namely the product of the rate coefficient with the time-dependent abundances of the reactants (i.e. $k \times n_1(t) \times n_2(t) \times \dots$). In `KROME`, you can use the `krome_print_best_flux` subroutine to print the most efficient reactions in descending rank (`list_user_functions.py` and the `KROME` wiki also help here).

Part 5: Adding a reaction

IMPORTANT: before proceeding, please note that adding the `-noExample` flag to the `KROME` python call will NOT replace your `test.90` file in `build/` that you previously modified.

We will now add a reaction rate from the KIDA database¹. We wish to add a secondary photodissociation reaction from CRs for water, i.e., $\text{H}_2\text{O} + \text{CRP} \rightarrow \text{H} + \text{OH}$. Search the database and add the reaction to the network: the rate coefficient has the same form as the direct CR dissociation, only the pre-factor changes (click on the information icon, , on the KIDA website to display the form of the rate coefficient). Re-run the `KROME` (python) pre-processor, then compile and run the code. This rate will not affect the total amount of water, but will have an impact on the OH abundance. Compare your results with Fig. 2 (right).

¹<http://kida.obs.u-bordeaux1.fr/>

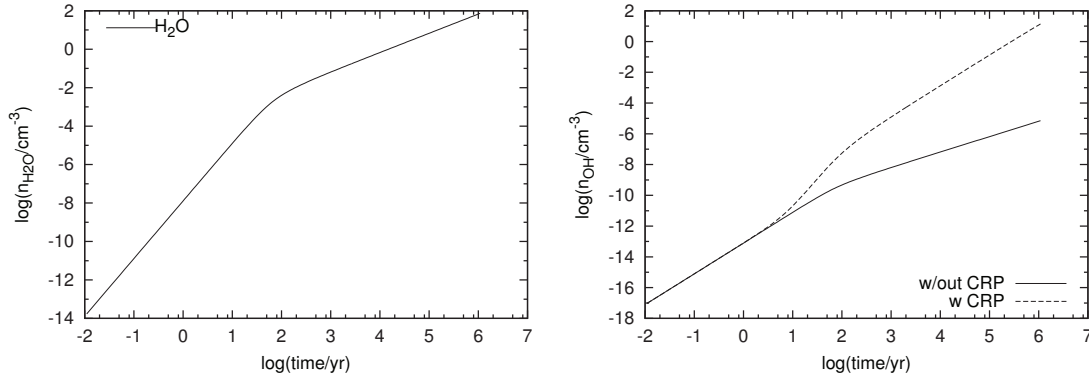
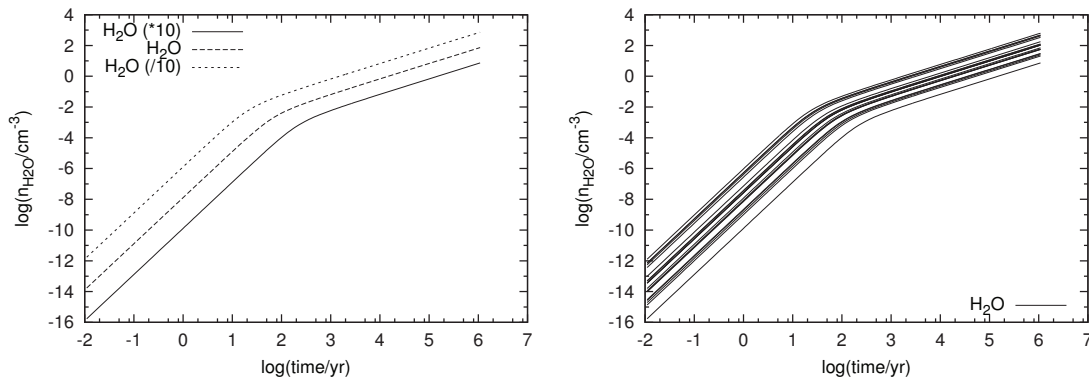
Figure 2 – Expected results for problem 1. *Left*: part 3; *right*: part 5.

Figure 3 – Expected results for problem 1, part 6.

Part 6: Play with the parameters

To understand what happens when you change the flux of CRs, run the simulation again by increasing and decreasing the CR flux by an order of magnitude and plot the evolution of H_2O again. You should obtain Fig. 3 (left).

If we increase the number of independent variables, we can run a sensitivity analysis for the network. As an example, let's do this for the first three reactions by changing the *auto* reaction rate coefficient in the `networkH2O.ntw` to these temperature-independent coefficients:

reaction	rate(cm^3/s)
$\text{O}^+ + \text{H}_2 \rightarrow \text{OH}^+ + \text{H}$	$0.16 \times 10^{-8} \cdot \varphi_1$
$\text{OH}^+ + \text{H}_2 \rightarrow \text{H}_2\text{O}^+ + \text{H}$	$0.11 \times 10^{-8} \cdot \varphi_2$
$\text{H}_2\text{O}^+ + \text{H}_2 \rightarrow \text{H}_3\text{O}^+ + \text{H}$	$0.61 \times 10^{-9} \cdot \varphi_3$

where φ_i are multiplicative factors that allow the variation of the reaction rate intensity (e.g., $\varphi_1 = 0$ will turn off the first reaction rate). These coefficients can be controlled in the same way as the CR flux, i.e., with a `@common` statement. If you randomly increase or decrease these coefficients by one order of magnitude you should obtain Fig. 3 (right), which shows 30 runs where we varied φ_1 , φ_2 , φ_3 , and ξ_{CR} asynchronously and randomly by one order of magnitude. We suggest you build a loop to randomly change the parameters, and do so logarithmically, i.e., $\varphi_i = \text{dex}(2 \cdot \text{rand}() - 1)$, where $0 \leq \text{rand}() \leq 1$.

Problem 2: Cooling and Heating

In this exercise we will learn how to customize cooling and heating in KROME. The chemical network used is a simple atomic network that contains H and Fe up to the first level of ionization (i.e., H^+ and Fe^+). To continue keeping things simple, cooling will only be provided by Fe^+ , while the heating will be from radioactive decay ($^{60}\text{Fe} \rightarrow \text{products}$), including ionization of H via high energy electrons.

Note: the model applied in this exercise has some limitations and is intended only to give an overview on some KROME commands which can be helpful to the user. In particular, (i) the cooling is limited to a single species, (ii) the ionization due to radionuclides acts only on H, even though H_2 ionization is more important, and (iii) the effect of the radionuclides is enhanced to create a visible impact on the evolution. However, the physics employed here *can* be generalized to more realistic models (see Cleeves+ 2013² and Umebayashi & Nakano 2009³ for example).

Part 1: Basic chemical network

The chemical network employed for this problem is rather simple. It consists of the following set of reactions:

1. $\text{H}^+ + \text{e}^- \rightarrow \text{H} + \gamma$
2. $\text{Fe}^+ + \text{e}^- \rightarrow \text{Fe} + \gamma$
3. $\text{H} + \text{e}^- \rightarrow \text{H}^+ + 2\text{e}^-$
4. $\text{Fe} + \text{e}^- \rightarrow \text{Fe}^+ + 2\text{e}^-$

The first step consists of preparing the chemical network (Note: remember that you can omit γ from a reaction). The reactions are already in the KROME database, so you can write *auto* instead of explicitly including a reaction rate coefficient, as in the following example (and Problem 1):

```
@format:idx,R,R,P,rate
1,H+,E,H,auto
...
```

Part 2: Evolution

To test the chemical network, you should write a program that evolves the system from these initial conditions: $n(\text{H}^+) = n_{\text{tot}}$, $T_{\text{gas}} = 10^3$ K, and $n_{\text{tot}} = 10^4 \text{ cm}^{-3}$. To initialize Fe, we again employ the `krome_scale_Z` subroutine with solar metallicity ($\log(Z/Z_{\odot}) = 0$), but note that this function only scales *neutral* metals. *As we want the iron to be initially fully ionized, you should copy the variable `x(krome_idx_Fe)` to `x(krome_idx_Fej)` and set the other to zero.* The electron abundance should be calculated accordingly (using `krome_get_electrons`, for example). Recall that you can use `list_user_functions.py` to get information on the functions contained in the user module (-h for details on its usage). To evolve the system in time, you can modify the `test.f90` auto-generated by KROME as in the following pseudo-code:

```
initialize species and temperature
t = 0
dt = 1e-2 yr
do
  dt = dt * 1.1
  t = t + dt
  call krome
  write output
  if(t>1e8 yr) break loop
end do
```

After compiling and running the code, you will hopefully obtain a plot like Fig. 4 (left).

²<http://adsabs.harvard.edu/abs/2013ApJ...777...28C>; see CABV2013.pdf in the ex2/ folder.

³<http://adsabs.harvard.edu/abs/2009ApJ...690...69U>; see UN2009.pdf in the ex2/ folder.

Part 3: Cooling

In this part, we will add custom cooling for Fe^+ using data from Hollenbach & McKee (1989⁴; hereafter HM89). Note that HM89 provides the relative ΔE_{ul} while KROME uses the absolute energy of the individual level E_u . However, since the energy of the ground level is $E_0 = 0$, $\Delta E_{20} = E_2$, and $\Delta E_{10} = E_1$, and thus you can directly use the data provided by HM89. A skeleton cooling file is provided (`coolX.dat`), and as you can see, it contains information about the atom levels, transitions, and collisional rates. What you have to do is extract the missing data from HM89 Table 8 and add (i) A_{ij} coefficients for the transition $2 \rightarrow 0$, (ii) the excitation coefficients for electron collisions (transition $1 \rightarrow 0$, $2 \rightarrow 0$, and $2 \rightarrow 1$), and energy level 2. The degeneracy of level $n = 2$ is 6. Note that, to include the excitation coefficients, an additional variable is required and can be added at the top of the cooling file with `@var`, namely $T_2 = T_{gas}/(100 \text{ K})$. Once you have added the data to the cooling file you can load it to KROME by using the command `-coolFile=coolX.dat`.

REMINDER: by adding the `-noExample` flag, the `test.90` file that you wrote earlier will NOT be replaced!

Running KROME with the `-coolFile=coolX.dat` option will not add automatically the appropriate cooling function to KROME. In order to do that, we need to do one more thing. Check which cooling functions are available by using `-cooling=?`. This will stop the execution of KROME after listing all of the available cooling functions. You should obtain something like:

```
Cooling FeII available from coolX.dat
Available coolings are: ATOMIC, H2, HD, DH, DUST, FF, H2GP98, COMPTON,
EXPANSION, CIE, CONT, CHEM, DISS, Z, CO, Z_CIE, Z_CIENOUV, FeII, FILE
```

where FeII is included in the list because it has been loaded from the file you prepared. You can now add the cooling by replacing `-cooling=?` with `-cooling=FeII` (case does matter).

Compile and run KROME again to obtain the temperature evolution with time (Fig. 4, right). Note that you can use the same `test.f90` file as before since the KROME pre-processor has automatically replaced all the files, including those used for the thermal evolution.

Part 4: Excited levels

As a continuation of the previous part, we are going to plot the excitation levels for Fe^+ . To do this, use the function `krome_popcool_dump`. Note that the function dumps the data in the following way:

```
FeII   1  0.11000000E-001  0.20310919E+000  0.31422730E+000
FeII   2  0.11000000E-001  0.74648789E-001  0.31422730E+000
FeII   3  0.11000000E-001  0.36469327E-001  0.31422730E+000
...
```

where the columns are (i) atom name, (ii) level number (the ground level is $n = 1$), (iii) independent variable, in your case time, (iv) abundance of the level, and (v) total abundance of all the levels for the given atom. If you plot column (iii) vs. column (iv) divided by column (v) you should get Fig. 5 (left).

Can you comment on the result and explain why the temperature decrease ceases to work around 10^3 yr?

Part 5: Adding an isotope

In this part of the exercise we want to add the decay reaction $^{60}\text{Fe} \rightarrow ^{60}\text{Co} \rightarrow ^{60}\text{Ni}$. For the purposes of our exercise, we consider the process as a single step $^{60}\text{Fe} \rightarrow ^{60}\text{Ni}$ with a half-life of $\tau_h = 1.5 \times 10^6$ yr. Add this reaction to the network keeping in mind that isotopes are written as, e.g., `[60Fe]`. With some algebra you can prove that the reaction rate is $\ln(2)/\tau_h$. The abundance of ^{60}Fe is $n_{60\text{Fe}} = 10^{-6} n_{\text{Fe}^+}$, and assume that,

⁴<http://adsabs.harvard.edu/abs/1989ApJ...342..306H>; a PDF is provided in the `ex2/` folder.

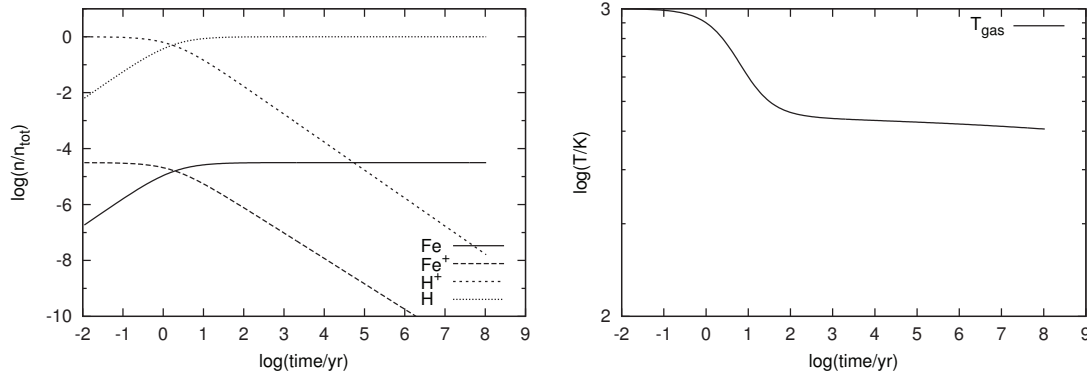


Figure 4 – Expected results for problem 2. *Left*: part 2; *right*: part 3.

given the low concentration, we do not care about the ionization state of the isotope, but instead use the reaction to track its total abundance.

After modifying the network, re-run the KROME python. You will notice a couple of warning messages: (i) the mass in the reaction is unbalanced because of the β decay, (ii) you now have a sink, namely ^{60}Ni . You can ignore both. Next, set the initial isotopic abundance in the `test.f90` file, compile, and run the executable again. You should obtain something like Fig. 5 (right).

Part 6: Ionization from isotope decay

We assume that the products of the decay ionize the hydrogen via the reaction $\text{H} \rightarrow \text{H}^+ + e^-$. The reaction rate depends on the amount of ^{60}Fe present in the gas phase, as well as a coefficient that is a function of the geometry of the problem (see Cleeves+ 2013² and Umebayashi & Nakano 2009³ for further details). We assume this coefficient ξ to be a variable by using the `@common` token (remember to check the wiki, bitbucket.org/tgrassi/krome/wiki/quick1, for details), and the final rate is given by $\xi \cdot n_{60\text{Fe}}$ (the index for the isotope should be called `idx_60Fe`). Add this reaction to the network and run the system using $\xi = 10^{-10}$ and $10^{-10} \text{ cm}^3 \text{ s}^{-1}$. You should obtain Fig. 6 (left). As you can see (in this particular case), the effects of isotope decay (ID) on the ionization of H^+ are negligible.

Part 7: Heating by ionization from isotope decay

Even if the effect on the ionization state is negligible, the heating from the process generates $W = 36 \text{ eV}$ per ionization. The energy released into the gas is then $E = W \xi n_{60\text{Fe}} n_{\text{H}} \text{ erg cm}^{-3} \text{ s}^{-1}$. We can add this heating to the system by using a custom statement in the network file. To do this you should add a custom heating block as sketched here:

```
@heating_start
  @heating:user_flux*n(idx_60Fe) * hydrogen_abundance * (W in erg)
@heating_stop
```

This block will be automatically included into KROME during the python execution. If you compile and run the executable again, you should obtain Fig. 6 (right). In this case, the heating due to ID is *not* negligible at late times.

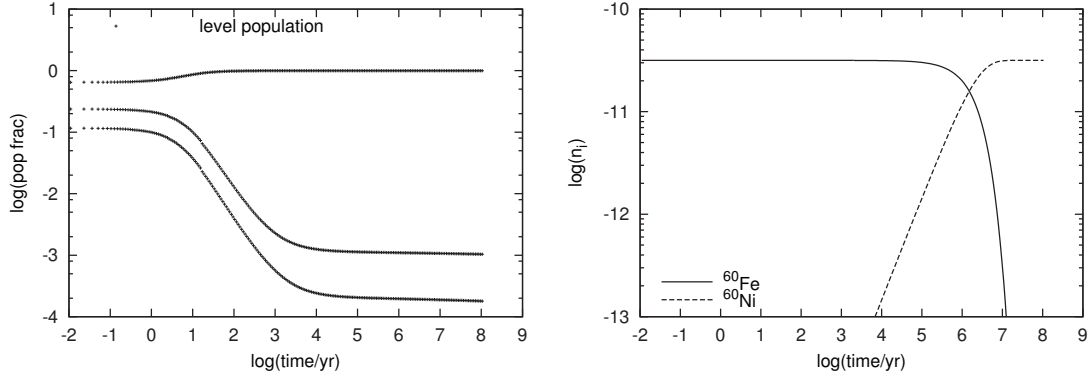


Figure 5 – Expected results for problem 2, continued. *Left*: part 4; *right*: part 5.

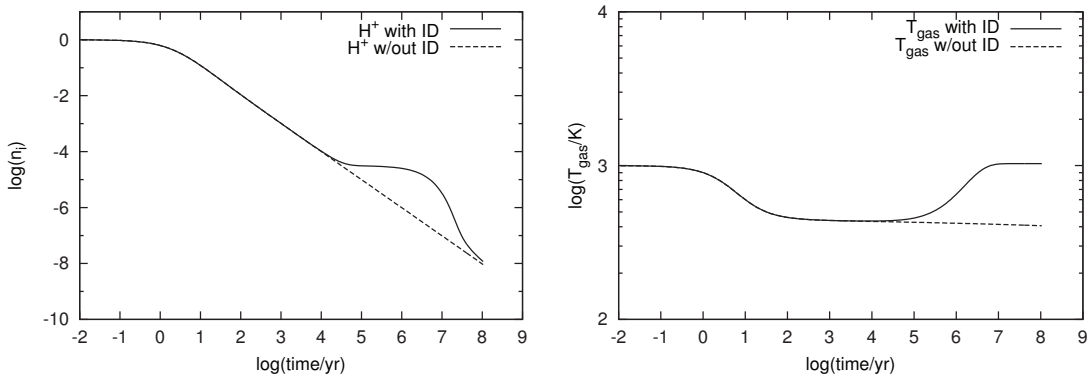


Figure 6 – Expected results for problem 2, part 6 and 7. Solid (dashed) lines represent models that include (omit) ionization/heating due to isotope decay (ID).