Problem 3: Strömgren sphere

This problem will show you how to use the main photochemistry commands in KROME. It consists of a 1D radial model of a Strömgren sphere and a star as a central source of ionizing radiation. The evolution will be calculated using a very simple operator splitting method (very inefficient!) to maintain focus on the chemistry.

Part 1: Basic chemical network
The chemical network here is pretty simple, being composed of only two reactions, namely H recombination and photoionization:

1. $H^+ + e^- \to H + \gamma$
2. $H + \gamma \to H^+ + e^-$

As a first step, we have to indicate to KROME that the second reaction is photochemical. To do this, you should wrap the reaction in a block delimited by the tokens `@photo_start` and `@photo_end`. When you indicate to KROME that a reaction is photochemical, the "rate" you put in the network file is actually a cross-section that KROME will internally convert to a rate coefficient for you. The cross-section for the photoionization reaction used here will be approximated by $\sigma_H(E) = 6.3 \times 10^{-18}\,[E/(13.6\,\text{eV})]^{-3}$ cm^2. Do not forget that KROME's photochemistry works in eV and the KROME variable for E is `energy_eV`. The recombination rate can be indicated with *auto*, as in previous exercises.

Part 2: Using photobins
"Photobins" are what you get when you split a spectral energy distribution (SED) into a finite number of discrete bins. In this case, the SED we are splitting belongs to the central radiation source. To use photobins, you should add the option `-photoBins NBINS`, where `NBINS` represents the number of photobins used to discretize the radiation spectrum. However, we are only interested in a narrow region around 13.6 eV, and for this reason we set `-photoBins=1`. After running the KROME python, you should find the pre-processed files in the `build/` folder, including the `test.f90` file that will be edited in the next section.

Part 3: Main file (no photochemistry)
We wish to understand what happens when a stellar source is embedded in a gas cloud of constant density, $n_\text{tot} = 10^{-3}$ cm^{-3}. The extent of the 1D domain will be $L = 6.6$ kpc (1 pc $= 3.085 \times 10^{18}$ cm), represented by a linearly and equally spaced grid of `ngrid=300` points. The position of the ith grid point, r, is given by:

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

where `rmin=1 pc` and `rmax=L`, i.e., the first point is at 1 pc and we assume that the size of the cell, Δr, corresponding to this grid point is also 1 pc. Model results are represented by a 2-dimensional array that contains the chemical species at each grid point: `xall(ngrid, krome_nmols)`. The initial conditions are $T_\text{gas} = 10^4$ K, $n_\text{H} = n_\text{tot}$, and $n_{\text{H}^+} = n_{\text{e}^-} = 1.2 \times 10^{-3}\, n_\text{tot}$.
To simplify the exercise, we use an explicit solver and supply a `test.f90` template file that you can build on. The backbone of the solver is a loop over spatial grid points nested in a loop over time steps:

```
initialize chemistry for all grid points
dt = 0.1 yr                   // default time-step
LOOP on time points
  dt = dt * 1.01              // increase time-step
  t = t + dt                  // advance total time
  LOOP on grid points using loop variable i
    x(:) = xall(i,:)          // store species array for ith cell
    call KROME(x(:),Tgas,dt)  // advance chemical evolution by a time-step
    xall(i,:) = x(:)          // copy evolved local array to global array
  END LOOP on grid points
```

Problem 3 continued on next page...

```
  if(t>tmax) break loop on time // exit loop when end time exceeded
END LOOP time

//save data as a function of radius
LOOP on grid points using loop variable i
    write xall(i,:) to file
END LOOP on grid points
```

where $t_{max} = 3 \times 10^7$ yr.

Part 4: Add photochemistry (optically thin)
The pseudo-code in the previous step calculates the chemical evolution at each grid point without any radiation. Now we want to add a stellar source at $r = 0$ given by a black-body with $T_\star = 3 \times 10^4$ K (mimicking a B0 star) and radius $R_\star$=66.1 $R_\odot$ ($R_\odot = 7 \times 10^{10}$ cm), corresponding to an emission of $\sim 5 \times 10^{48}$ photons/s. The emission is assumed isotropic. The energy range we are interested in is between 13.6 and 13.8 eV. To make KROME aware of this, you can use the subroutine `krome_set_photoBin_BBlog` (as usual, more information is available from `list_user_functions.py`) to initialize a black-body spectrum with logarithmically spaced photobins in a range indicated by the arguments (it does not actually matter here since we are using just one bin).
The stellar radiation is emitted by a star of radius $R_\star$ and has a geometrical dilution factor that is proportional to $1/r^2$, and thus a rescaling factor $\eta_g = 4\pi^2 R_\star^2/(4\pi r^2)$. However, KROME internally already multiplies the radiation by a factor 4π (it assumes isotropy), so that we actually need $\eta_g = \pi R_\star^2/(4\pi r^2) = R_\star^2/(4r^2)$.
To rescale the stellar flux, use the subroutine `krome_photoBin_scale(xscale)`, which multiplies the spectrum by a factor `xscale`, at every grid point. In this case, `xscale` is η_g. The rescaling is not progressive (it depends on r, which is an absolute quantity), and so, at each grid point, you need to "restore" the initialized spectrum before scaling it. In KROME, photochemical initialization functions automatically store the spectrum and, more importantly, it can be restored by using the subroutine `krome_photoBin_restore()`.

To recap, the algorithm including the new parts is as follows:

```
initialize black-body spectrum
LOOP on time points
  LOOP on grid points using loop variable i
     restore spectrum
     calculate eta_g
     scale spectrum by eta_g
     call KROME
  END LOOP on grid points
END LOOP time
```

If you compile with `Makefile` and run the executable, you should obtain Fig. 7 (left).

Part 5: Add photochemistry (optically thick)
In the optically thick case, the photochemistry at lower radii reduces the flux of photons at the next grid point (i.e., at larger radii). This reduction in flux is represented by the opacity. You can determine this quantity for a grid cell of size Δr by using the `krome_get_opacity_size` function, which returns an array of size `krome_nPhotoBins` containing the opacity $\tau(E)$ for each photobin. You should apply this reducing factor at every grid point progressively, i.e., the ith point will have an energy-dependent scaling factor $\eta_i(E) = \prod_{j=0}^{i} \exp[-\tau_i(E)]$, where the product runs from the first to the ith cells.
As the scaling is energy dependent, you should replace the function `krome_photoBin_scale(xscale)`, used in part 4, with `krome_photoBin_scale_array(xscale(:))` that accepts an array of doubles of size

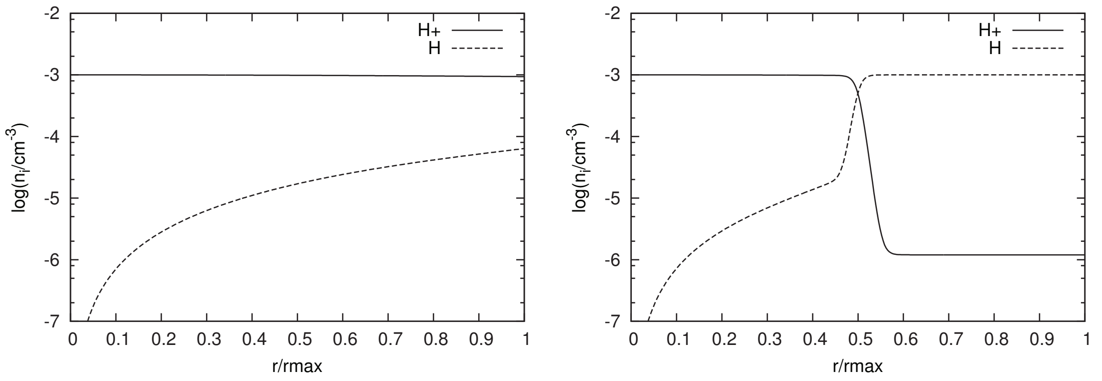

Figure 7 – Expected results for problem 3. *Left:* part 4; *right:* part 5.

krome_nPhotoBins as an argument which, in this case, is $\eta_g \times \eta_i(E)$.

Again, to recap, the algorithm for the optically thick case is as follows:

```
initialize black-body spectrum
LOOP on time points
  eta_i(:) = 1.
  LOOP on grid points using loop variable i
     restore spectrum
     eta_i(:) = eta_i(:)*exp(-opacity(:))
     calculate eta_g
     scale spectrum by eta_g*eta_i
     call KROME
  END LOOP on grid points
END LOOP time
```

If you compile with Makefile and run the executable, you should obtain Fig. 7 (right).