Emily Ng
ASTR496 Computing in Astronomy
Assignment 3 / Final Assignment
Chemistry Solver Write-Up

All material referenced in this write-up is accessible through a Github repository:
https://github.com/emilyng/Chemistry-Solver

The chemistry solver program tracks the evolution of nine species of elements. The nine species involved are: HI, HII, HeI, HeII, HeIII, H2I, H2II, H-, and e- .

The reaction equations are taken from Table 3 and Table 4 of Smith et al., 2016 paper

Table 3. Chemical reactions in the six species network

Reaction			Reference Data	Reference Fit
$H + e^-$	→	$H^+ + e^- + e^-$	1	2
$H^+ + e^-$	→	$H + \gamma$	3	2, 4
$He + e^-$	→	$He^+ + e^- + e^-$	1	2
$He^+ + e^-$	→	$He + \gamma$	5, 6	4, 6, 7
$He^+ + e^-$	→	$He^{++} + e^- + e^-$	1	2
$He^{++} + e^-$	→	$He^+ + \gamma$	3, 8	4, 9
$H + H$	→	$H^+ + e^- + H$	10	11
$H + He$	→	$H^+ + e^- + He$	12	11
$H + \gamma$	→	$H^+ + e^-$	13	—
$He + \gamma$	→	$He^+ + e^-$	13	—
$He^+ + \gamma$	→	$He^{++} + e^-$	13	—

Key: 1 – Janev et al. (1987); 2 – Abel et al. (1997); 3 – Ferland et al. (1992); 4 – Hui & Gnedin (1997); 5 – Burgess & Seaton (1960); 6 – Aldrovandi & Pequignot (1973); 7 – Black (1981); 8 – Spitzer (1978); 9 – Cen (1992); 10 – Gealy & van Zyl (1987); 11 – Lenzuni et al. (1991); 12 – van Zyl et al. (1981); 13 – see Section 5

Table 4. Chemical reactions in the nine species network

Reaction			Reference Data	Reference Fit
$H + e^-$	→	$H^- + \gamma$	1	2
$H^- + H$	→	$H_2 + e^-$	3	3
$H + H^+$	→	$H_2^+ + \gamma$	4	5
$H_2^+ + H$	→	$H_2 + H^+$	6	6
$H_2 + H^+$	→	$H_2^+ + H$	7	8
$H_2 + e^-$	→	$H + H + e^-$	9	9
$H_2 + H$	→	$H + H + H$	10	10
$H^- + e^-$	→	$H + e^- + e^-$	11	12
$H^- + H$	→	$H + e^- + H$	11	12
$H^- + H^+$	→	$H + H$	13	14
$H^- + H^+$	→	$H_2^+ + e^-$	15	12, 16
$H_2^+ + e^-$	→	$H + H$	17	12
$H_2^+ + H^-$	→	$H_2 + H$	18	18
$H + H + H$	→	$H_2 + H$	19	19
$H + H + H_2$	→	$H_2 + H_2$	20, 21	22
$H^- + \gamma$	→	$H + e^-$	23	—
$H_2^+ + \gamma$	→	$H + H^+$	23	—
$H_2 + \gamma$	→	$H_2^+ + e^-$	23	—
$H_2^+ + \gamma$	→	$H^+ + H^+ + e^-$	23	—
$H_2 + \gamma$	→	$H + H$	23	—
$H + H + grain$	→	$H_2 + grain$	—	24*

Note: the nine species network also includes all of the reactions listed in Table 3.
Key: 1 – Wishart (1979); 2 – Stancil et al. (1998); 3 – Kreckel et al. (2010); 4 – Ramaker & Peek (1976); 5 – Latif et al. (2015); 6 – Karpas et al. (1979); 7 – Krstić (2002); 8 — Savin et al. (2004a,b); 9 – Trevisan & Tennyson (2002); 10 – Martin et al. (1996); 11 – Janev et al. (1987); 12 – Abel et al. (1997); 13 – Fussen & Kubach (1986); 14 – Croft et al. (1999); 15 – Poulaert et al. (1978); 16 – Shapiro & Kang (1987); 17 – Schneider et al. (1994); 18 – Dalgarno & Lepp (1987); 19 – See text; 20 – Sutton (1962); 21 – Ham et al. (1970); 22 – Cohen & Westberg (1983); 23 – see Section 5; 24 – Tielens & Hollenbach (1985); Omukai (2000); * - This reaction included as an additional option when metals are present.

Each species' number density rate of change is solved for and updated in the program to evolve each species number density at a later time in the evolution. The species' number density rate of change equations can be generalized in the following form:

dn_S/dt = k1C – k2D

where n_S is the number density of a species S, k1 and k2 are reaction rates, C is the construction term of the species, and D is the destruction term of the species. For example, taking only HI and HII, our reaction are:

HI + $\gamma \rightarrow$ HII + e- , k1
HII + e- $\rightarrow$ HI, k2

Using only these 2 reactions, the number density rates of change for each species would be:

dn_HI/dt = k1*n_HII*N_e- - k2*N_HI
dn_HII/dt = k2*N_HI - k1*n_HII*N_e-
dn_e-/dt = k2*N_HI - k1*n_HII*N_e-

For nine species, the number density rates of change equations for each species would be a lot more complicated than just this. All reaction equations are recorded and documented in the file: RHS_eqs.py

RHS_eqs.py:
In this file, up to 19 chemical reaction equations are recorded. This is because only up to 19 reaction rates were located from Abel et al., 1996. These reaction equations hold all chemical species “HI”, “HII”, “HeI”, “HeII”, etc. as sympy symbols using sympy.simpify. These equations are contained as a list of the reactants, products, and reaction rate of the reaction and are used to form our rate of density changes. Rate equations are formulated from all reactions in getRHS.py . Rates equations will be used as the right hand side function used in scipy.integrate.ode(), which will be featured later in this write-up.

getRHS.py:
getRHS.py hold 3 functions: find_formation(), find_destruction(), and get_rhs(). Only get_rhs() needs to be called explicitly since find_formation() and find_destruction is called implicitly through get_rhs(). get_rhs() is a function that helps formulate our RHS equations used as input for the scipy.integrate.ode() function. The get_rhs() function takes in a species symbol (ie. HI, HII, HeI, etc) defined in RHS_eqs.py and forms a density rate of change equation.

RHS outputs from get_rhs() function:

In[10]: get_rhs(HI)
Out[10]: H2I*HII*k11 + H2I*de*k12 - H2II*HI*k10 + H2II*HM*k19 + H2II*de*k18 - HI*HII*k9 - HI*HM*k8 - HI*de*k1 - HI*de*k7 + HII*HM*k16 + HII*de*k2 + HM*de*k14

In[11]: get_rhs(HII)
Out[11]: -H2I*HII*k11 + H2II*HI*k10 - HI*HII*k9 + HI*de*k1 - HII*HM*k16 - HII*HM*k17 - HII*de*k2

In[12]: get_rhs(HeI)
Out[12]: -HeI*de*k3 + HeII*de*k4

In[13]: get_rhs(HeII)
Out[13]: HeI*de*k3 - HeII*de*k4 - HeII*de*k5 + HeIII*de*k6

In[14]: get_rhs(HeIII)
Out[14]: HeII*de*k5 - HeIII*de*k6

In[15]: get_rhs(H2I)
Out[15]: -H2I*HI*k13 - H2I*HII*k11 - H2I*de*k12 + H2II*HI*k10 + H2II*HM*k19 + HI*HM*k8

In[16]: get_rhs(H2II)
Out[16]: H2I*HII*k11 - H2II*HI*k10 - H2II*HM*k19 - H2II*de*k18 + HI*HII*k9 + HII*HM*k17

In[17]: get_rhs(HM)
Out[17]: -H2II*HM*k19 - HI*HM*k15 - HI*HM*k8 + HI*de*k7 - HII*HM*k16 - HII*HM*k17 - HM*de*k14

In[18]: get_rhs(de)
Out[18]: -H2II*de*k18 + HI*HM*k15 + HI*HM*k8 - HI*de*k7 + HII*HM*k17 - HII*de*k2 - HeII*de*k4 - HeIII*de*k6

evolve.py:
The solving of the RHS equations for number densities of each species over time is done in evolve.py using the evolve() function. Within evolve.py there are 2 functions: rhs() and evolve(). These 2 functions are jointly dependent on each other as rhs() takes in a state vector defined in evolve() and evolve() takes in and solves new rhs equations through scipy.integrate.ode() that are redefined through each time step.

RHS Reaction Rates:
The first thing needed to make the reaction functions are the reaction rates. These reaction rates k1, k2, k3, …, etc. are taken from Abel et al., 1996 and Smith et al. 2016 Grackle Fortran source code coll_rates_g.F (available at https://bitbucket.org/grackle/grackle/overview). These reaction rates are dependent on the temperature of the system. In rate_coeff.py, each k-value is defined as a function of temperature. These functions are called upon when using the rhs() function in evolve.py .

RHS equations:
rhs() function takes in a state vector that is an array for the number densities for each species as well as the temperature of the system. The way the state vector is organized is this:
state_vector = (n_HI, n_HII, n_HeI, n_HeII, n_HeIII, n_H2I, n_H2II, n_HM, n_em, T)

The rhs equations were inputted manually through inspection and correlating symbols in rhs equations outputs from get_rhs (shown above) with its respective indices in the state vector (ie. HI index = 0, HII index = 1, …)

I acknowledge this could have been done using sympy.substitute() function, however I was unable to appropriately implement it this way because of a problem with internal data typing through the use of this method... Additionally, the rhs equations in rhs() for electrons (dnemdt) and temperature (dTdt) are both set to 0 as these quantities are evaluated and updated internally through the time stepping considering charge and (assumed) internal energy conservation.

evolve():
The evolve() function integrates and tracks the ionization and recombination of atomic Hydrogen, Helium, and molecular Hydrogen. It updates the number density of each species after each time step and produces a plot of the change in these quantities over time. It takes in initial neutral Hydrogen, Helium, and molecular Hydrogen fractions as inputs as well as initial density (here we assumed that density stays constant), time to evolve to. Parameters that can be changed in addition to these are: temperature T, safety_factor, and e_frac.

Caveats:
I had tried to revise the evolve() function to take in a type of integrator for scipy.integrate.ode.set_integrator() function, however when I tried to do that I get an error telling me there is an "'Unexpected istate=%s' % istate". So I decided to keep a default integrator of 'lsoda' with method 'bdf'.

I had also tried implementing the dynamic time stepping where the length of the time step is redefined after each rep based on the rate of change in the species densities at the time before. I had tried setting it up as:

```
S = integrator.y
dS = 10**(-4) + rhs(integrator.t, integrator.y)
dt = safety_factor * np.min(S/dS)
```

However, when I try running it with this time stepping, the function runs eternally as never finishes evaluating.

I had also attempted to modify evolve.py to be able to run through the command line using argparse. One problem that I had ran into was that it was giving me an ModuleNotFound error, saying that there was 'no module named 'numpy''. I resolved to using ipywdigets interaction bars in JupyterNotebooks to visualize changes in phase space when giving in different arguments for evolve(). I have made the following parameters configurable:

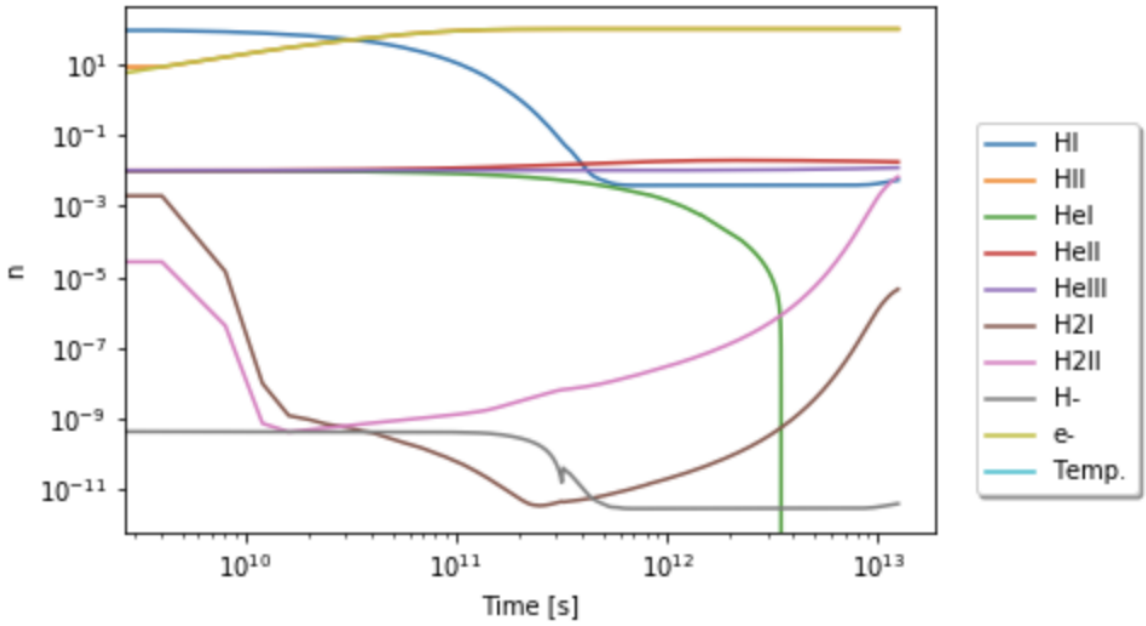
- n_total
- H_frac
- He_frac
- H2_frac

- e_frac
- rho
- T
- final_t
- safety_factor

Results:
When I time the integration and compare computing time of when the system starts with initial conditions already close to equilibrium versus with initial conditions far from equilibrium, I get that it takes less time to run when the system was already near equilibrium. It takes longer when it is far from equilibrium.

I have some preliminary results from before I had added in the temperature change in the evolution and was able to produce a result. This preliminary results code is contained in Prelim_Chemistry Solver.ipynb . Results from this draft program is also available through the progess report write-up.

One of the outputs obtained from this is attached below:

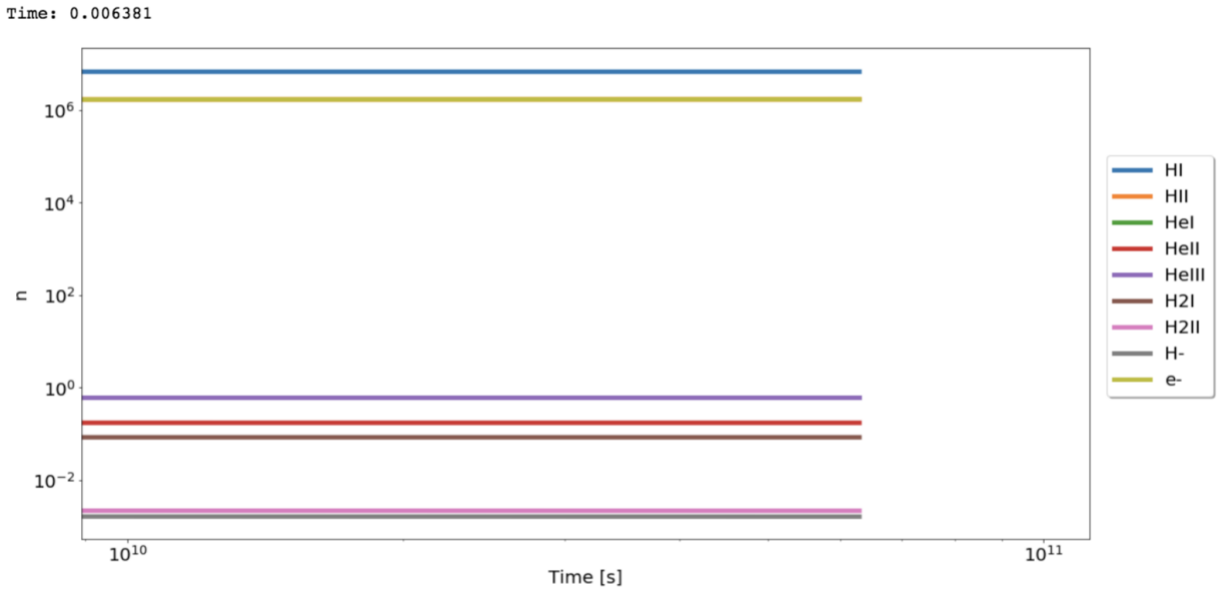


I found this plot to be interesting because the number densities of non-ionized molecular hydrogen (H2I) and singly ionized molecular hydrogen (H2II) decreases initially and then starts to increase later on in the evolution. It seems as though initially as things get ionized and the number density of neutral hydrogen decreases, which makes sense. The number densities of neutral and ionized molecular hydrogen also decreases at first. Neutral molecular hydrogen decreases until about at t = 1.5e10 seconds until it increases until the end of the evolution time. Ionized molecular hydrogen decreases until about at t = 2.5e11 seconds until it starts to increase until the rest of the evolution time. As neutral atomic hydrogen decreases the number of free

electrons increase until it flattens at about t = 1e11 seconds. About the time where the lines for HI and e- crosses, H2II roughly begins to increase.

Another strong feature is that about t = 4e11 seconds, neutral hydrogen begins to reach equilibrium and e- is already at equilibrium. This seems to cause HeI to decrease rapidly as H2I and H2II continues to increase rapidly. At around the same point in time, H- also seems to start to drop while HeII and HeIII shows slight increases. The above plot is consistent with charge conservation! However, one thing that puzzles me is why molecular hydrogen would start to increase. I would have thought that it would decrease thinking in terms of cloud collapse and star formation. This could be due to the design of this preliminary program in that it did not yet take into account the changing temperature of the system. I think this kind of system would have to increase in temperature as things become ionized, it would be more likely that hydrogen molecules would break up into atomic hydrogen particles and later contribute to number of electrons as more of it gets ionized.

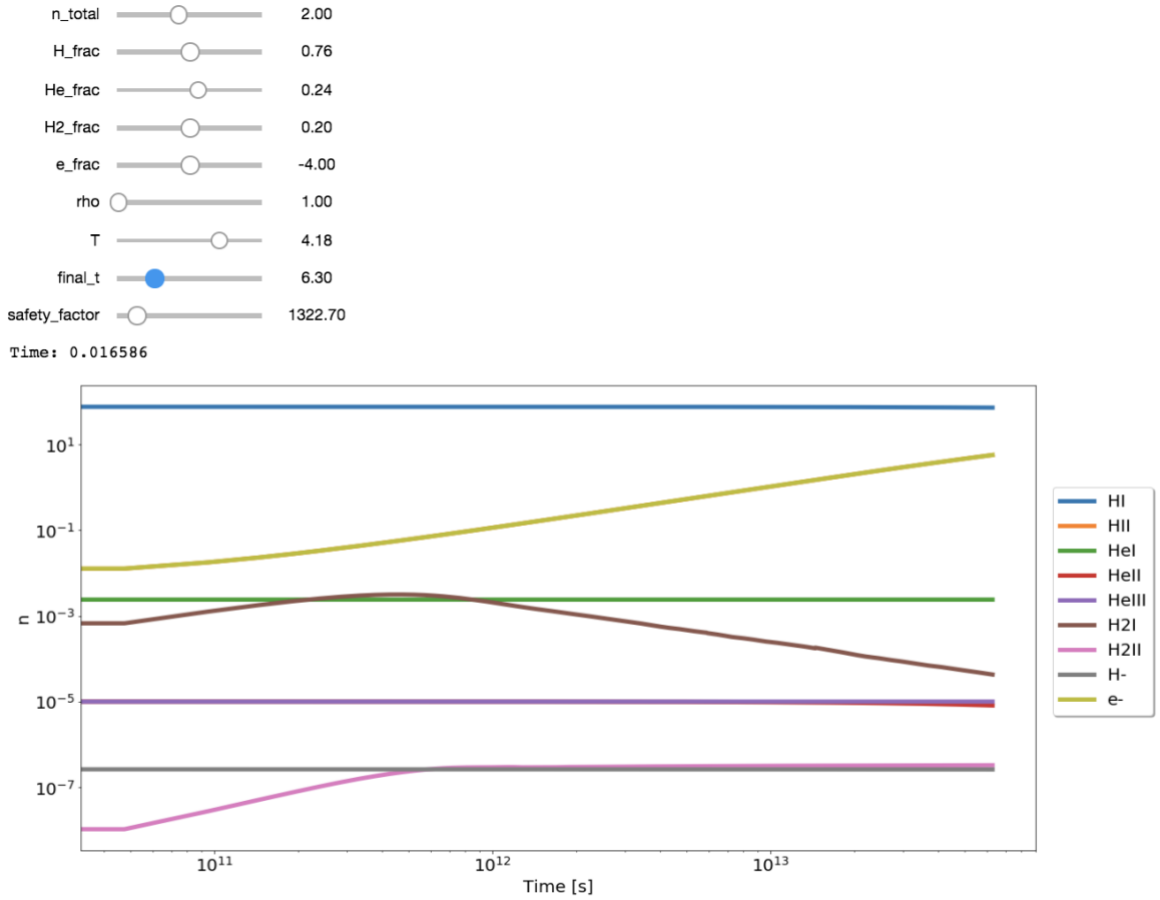
The final program is when I added in a changing temperature time step based on a conserved internal energy and density. Code and results are contained in Chemistry_Solver.ipynb . The first thing I noticed about my program is that the safety_factor had to be set really high in order for the evolution lines to progress through to the final_t. Otherwise, it stops abruptly before the final t. Below is an example of having set the safety_factor = 0.1:



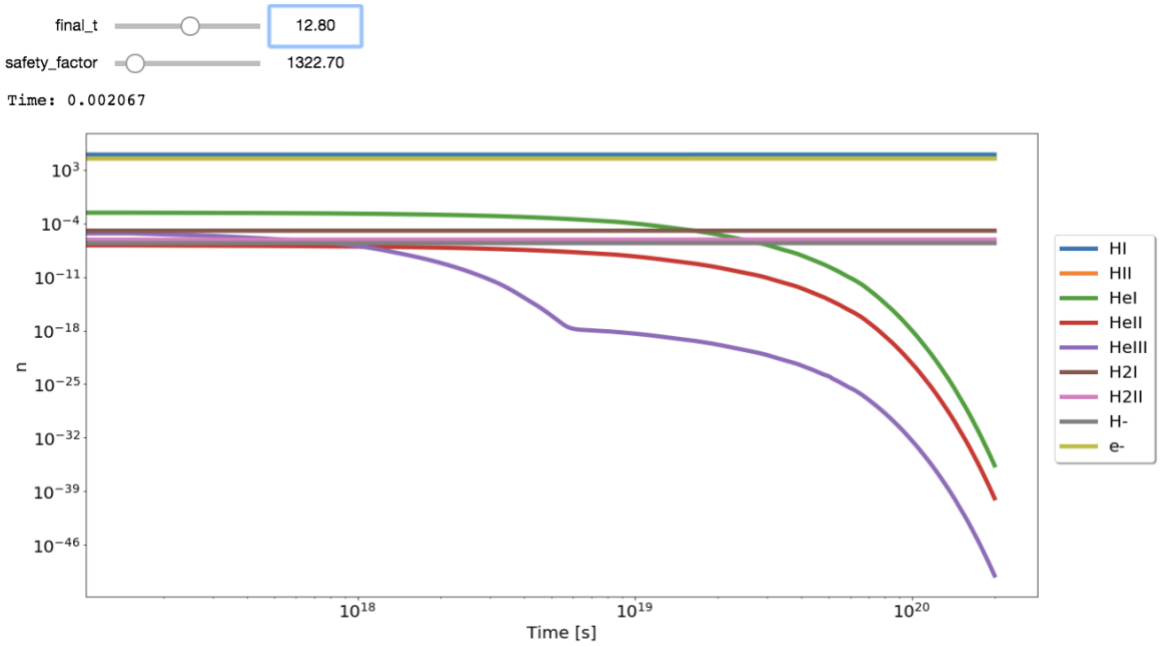
I am not sure why this would be the case, but I think it might have something to do with my inability to successfully implement the dynamic time stepping through the evolution.

Another significant problem I had with implementing this part of the assignment was that I am certain I am getting inaccurate results. Upon browsing through different initial conditions one consistent result I get is a rapid decrease in all He number densities. I would have expected the ionized helium densities to increase! Another unusual result was that the number densities of e- increases from the very beginning of the evolution while the H2I and H2II increases about until t

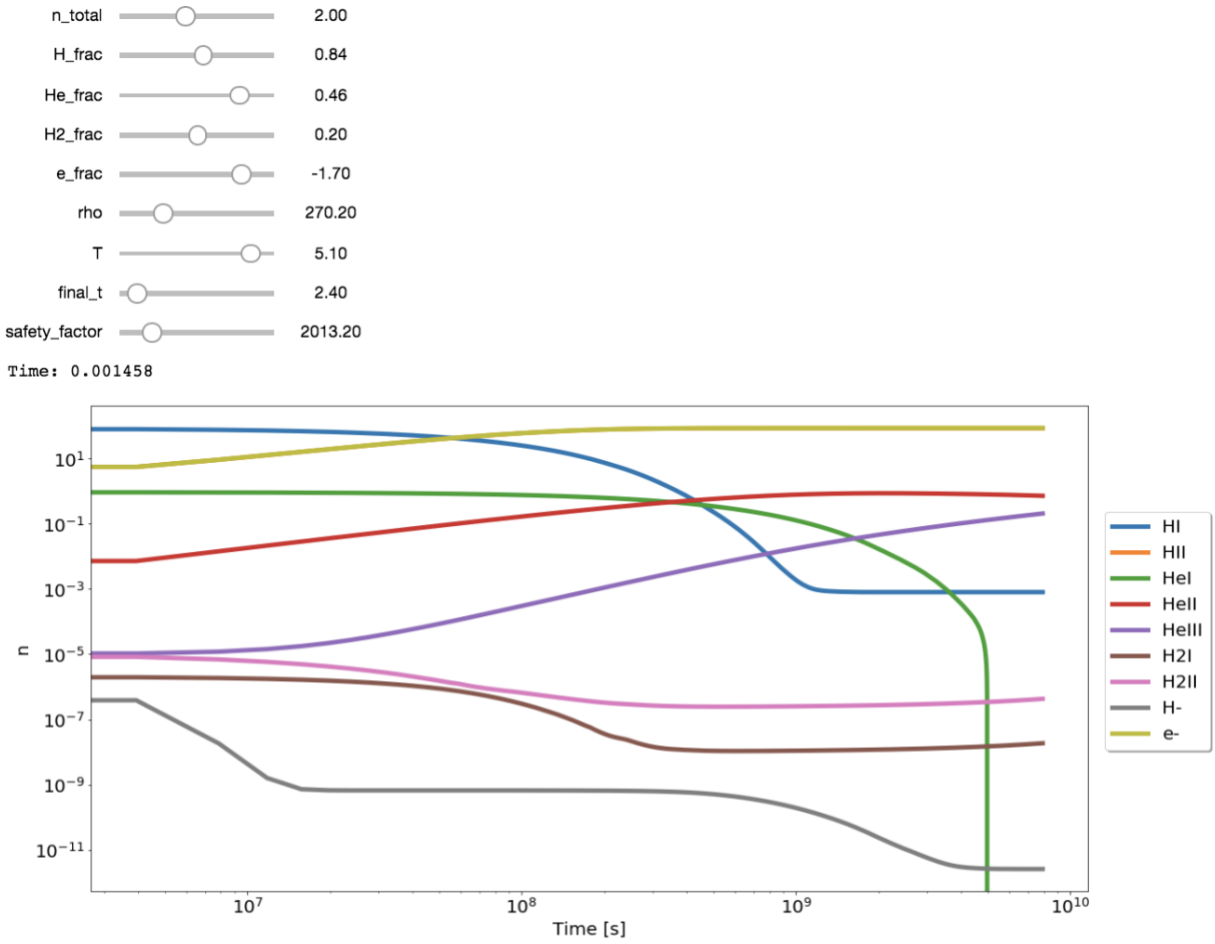
= 5e11 seconds, where H2II comes into equilibrium and H2I begins to drop. HeII begins to decrease slightly beginning at t = 10e13 seconds. However, the number density of electrons continues to rapidly increase:



When final_t is increased, I can begin to see that HI also begins to increase, HeII continues to decrease, and H2II begins to increase again. When I increase final_t yet again, HI and e- begins to level off to equilibrium and at an even later time, all He (neutral and ionized) begins to decrease rapidly! I feel that these results are highly inaccurate, and it probably has to do with the missing implementation of the change time stepping or that the initial conditions of species may have been set wrongly.



I had managed to tweak the parameters enough to be able to get the following result:



The initial arguments passed into evolve() function ended up mostly hydrogen with some helium and even less molecular hydrogen. The e_frac had been set to -1.70, along the way I had also noticed the e_frac parameter influenced the plots the most.

This final plot shows increasing number densities for e-, HeII, and HeIII, while the number densities for HI, HeI, H2I, H2II, and HM decreases. I would have expected this to happen because through conservation of charge, as the electrons and ionized species increase, the neutral species should decrease to compensate for charge interactions. What is surprising is that HeI shows an extremely strong decrease in number density at t = 5e9 seconds, but I am not sure why this might occur.

Assignment Discussion:
I found that the hardest part of this assignment was trying to figure out how each part of the problem comes into play when construction the evolution of particles. I found it hard working with the concept of mean molecular weights and particularly with how it would change with the mixing of several different species through the evolution.

Another challenging part of the assignment, but could be attributed to coding problems in general is that I find it hard to be able to find problems within my code or being able to debug. Often times I find myself running into a problem that I can't seem to identify the origin of.

I think the easy part was developing the right-hand side equations, although it as tedious to input the number densities of each species in the equations by hand according to its indices in the state vector, it was generally pretty straightforward. The harder part of that was using sympy to do it without having to do it manually.