Plane parallel steady state flow from blackboard notes

William Henney

September 12, 2016

Contents

- Initial equations
  - $\rho v = \Phi_0 \equiv \rho_1 v_1$
  - $\rho\,(a^2 + v^2) = \Pi_0 \equiv \rho_1 a_1^2\,(1 + M_1^2)$
  - $\frac{5}{2}\rho v a^2\,(1 + \frac{1}{5}M^2) = \mathcal{E}_0 - \int L\,dx$
    - where $\mathcal{E}_0 \equiv \frac{5}{2}\rho_1 v_1 a_1^2\,(1 + \frac{1}{5}M_1^2)$
- Can be boiled down to
  1. $(1 + M^2)\,a^2/v = \Pi_0/\Phi_0 = (1 + M_1^2)\,a_1^2/v_1 = (1 + M_0^{-2})\,v_0$
     - This is how velocity varies with soundspeed
     - For subsonic limit ($M^2 \ll 1$) it is effectively $v \propto a^2$. If the particle mass is not changing (constant ionization) then this is $v \propto T$
  2. $a^2\,(1 + \frac{1}{5}M^2) = a_1^2\left(1 + \frac{1}{5}M_1^2 - \frac{3}{2}\int \mathcal{L}\,ds\right)$

hello, this is some annotated text

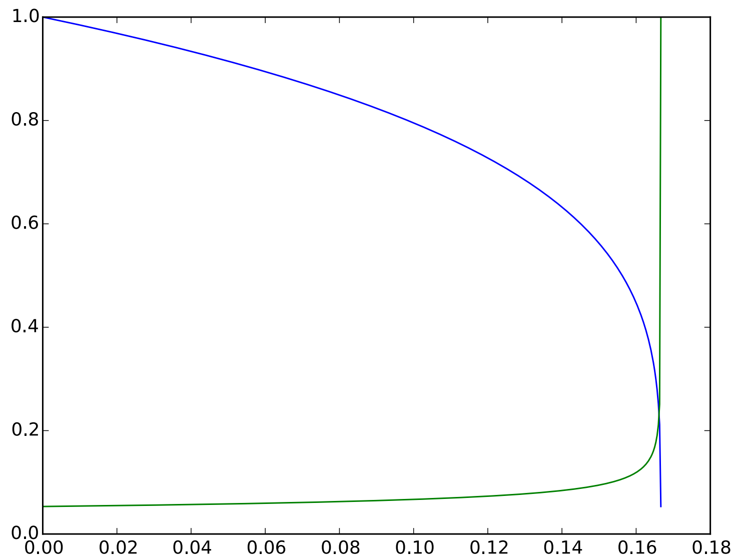
- This is how the sound speed (or Temperature) varies with distance
- Where $\mathcal{L} = L/L_1$ is dimensionless cooling function
- $s = x/h$ is dimensionless distance in terms of the cooling length: $h = \frac{3}{5}\rho_1 a_1^2 v_1 / L_1$
- And the immediate post-shock cooling function is $L_1 = n_1^2 \Lambda(T_1)$

1 Try to solve the subsonic-limit case and with power law cooling func

- Assume $\Lambda = \Lambda_1(T/T_1)^a$, where $a \approx -1$ for 10^5 to 10^6 K
- So first equation gives $v/v_1 = T/T_1$ and $n/n_1 = T_1/T$
  - => $\mathcal{L} = (n/n_1)^2(T/T_1)^a = (T/T_1)^{a-2}$
- And second equation gives
  - $\tau = 1 - 1.5 \int \tau^{a-2}\, ds$
  - where $\tau \equiv T/T_1$ is the dimensionless temperature
  - Differentiating: $d\tau/ds = -1.5\tau^{a-2}$
    - => $\int_1^\tau \tau^{2-a}\, d\tau = -1.5 \int_0^s ds$
    - => $(\tau^{3-a} - 1)/(3-a) = -1.5s$
    - => $\tau = (1 - 1.5(3-a)s)^{1/(3-a)}$
  - For example, with $a = -1$
    - $\tau = (1 - 6s)^{1/4}$
  - For example, with $a = +2$
    - $\tau = 1 - 1.5s$

```
####+BEGIN_SRC python :results file :return pltfile
import numpy as np
from matplotlib import pyplot as plt
pltfile = 'cooling-shell.pdf'
fig, ax = plt.subplots(1, 1)
s = np.linspace(0, 0.167, 500)
a = -1
tau = (1.0 - 1.5*(3 - a)*s)**(1./(3 - a))
print(tau)
```

```
rho = np.nanmin(tau)/tau
print(rho)
ax.plot(s, tau)
ax.plot(s, rho)
ax.set_ylim(0, 1)
fig.savefig(pltfile)
```



2 Use real cooling function instead

:ID: 05FA6299-E408-4935-8237-194ECCA91844

- This is an attempt to reconstruct this from memory since I had an emacs disaster last night and lost all my work for the last two days
- First equation is the same
- Second equation is $T/T_1 = 1 - 1.5 \int (\Lambda/\Lambda_1)(T_1^2/T^2)\, ds$
  - Differentiating: $(1/T_1) dT/ds = -1.5(\Lambda/\Lambda_1)(T_1^2/T^2)$
  - => $s = \frac{2}{3}(\Lambda_1/T_1^3)\ \int_T^{T_1} (T^2/\Lambda)\, dT$
- Note that the following needs to be run in python 3

```
import os
import numpy as np
from scipy import interpolate, optimize, integrate
from astropy.table import Table
from matplotlib import pyplot as plt
import seaborn as sns

def get_cooltable(logphi=10.0, logn=0.0, star='wr136', cwd='.'):
    cooldir = os.path.join('JaneCloudy', star.upper() + 'COOL/')
    templ = 'coolfunc-photo-{}-phi{:.2f}-ngc6888-n{:.2f}.dat'
    coolfile = templ.format(star, logphi, logn)
    return Table.read(os.path.join(cwd, cooldir, coolfile),
                      format='ascii.commented_header', delimiter='\t')

# Set up cooling function
tab = get_cooltable()
T_tab = tab['Temperature']
Lambda_tab = (tab['L (erg/cm3/s)'] - tab['H (erg/cm3/s)'])/(tab['Np']*tab['Ne'])
fLambda = interpolate.interp1d(T_tab, Lambda_tab)

# Calculate integral on finer grid
integrand_tab = T_tab**2 / Lambda_tab
fIntegrand = interpolate.interp1d(T_tab, integrand_tab)

# Equilibrium T where heating = cooling
Teq = optimize.fsolve(fLambda, 1e4)
# Go up to 1e6 K
logThi = 6.0
# And down to just above equilibrium T
logTlo = np.log10(1.001*Teq)
ngrid = 50
T_grid = np.logspace(logTlo, logThi, ngrid)

Lambda_grid = fLambda(T_grid)
# integrand_grid = fIntegrand(T_grid)

# Don't interpolate the integrand - rather recalculate it from the
# interpolated T and Lambda
integrand_grid = T_grid**2 / Lambda_grid
integral_grid = integrate.cumtrapz(integrand_grid, T_grid, initial=0.0)
```

```
fIntegral = interpolate.interp1d(T_grid, integral_grid)

# Set up graph for temperature and density
pltfile = 'cooling-shell-new-n100.pdf'
fig, (axtop, axbot) = plt.subplots(2, 1, sharex=True)

# Loop over all the shock velocities
for row in models:
    M0, u0, v1, n0, n1, N2, T1, dcool, tcool = [float(x) for x in row]
    label = 'Vs = {:.0f} km/s'.format(u0)
    mask = T_grid < T1
    T = T_grid[mask][::-1]
    s = (2./3.)*(fLambda(T1)/T1**3)*(fIntegral(T1) - integral_grid[mask][::-1])
    x = np.hstack([[-0.05, 0.0, 0.0], dcool*s])
    axtop.semilogy(x, np.hstack([[Teq, Teq, T1], T]))
    den = n1*T1/T
    axbot.semilogy(x, np.hstack([[n0, n0, n1], den]), label=label)

axtop.set_ylim(9000, 1.1e6)
axbot.set_ylim(0.3, 200.0)
axbot.set_xlabel('Distance, pc')
axbot.set_ylabel('Density, pcc')
axtop.set_ylabel('Temperature, K')
axbot.legend(ncol=3, fontsize='x-small', loc='upper center')
fig.savefig(pltfile)

#return list(zip(T_grid, Lambda_grid, integrand_grid, integral_grid))
```

3 [O III] and Ha emissivities

3.1 Equilibrium emissivity in the cooling zoone

3.2 Non equilibrium emissivity in the shock

4 Relation of isothermal sound speed and temperature:

- $\rho\ \mathrm{a}^2 = \mathrm{n_{tot}\ k\ T}$
- $\rho = \mathrm{m_p\ n_H\ (1 + 4\ y_{He})}$

- $n_{tot} = n_H \, (1 + x_H + y_{He} \, (1 + x_{He} + 2 \, x_{HeII}))$
- $=> a^2 = (k \, / \, \mu \, m_p) \, T$
  - where $\mu = (1 + 4 \, y_{He}) \, / \, (1 + x_H + y_{He} \, (1 + x_{He} + 2 \, x_{HeII}))$
- Table of μ values

y_{He}	x_H	x_{He}	μ
0.1	0.0	0.0	1.27
0.1	1.0	0.0	0.67
0.1	1.0	1.0	0.64
0.162	0.0	0.0	1.42
0.162	1.0	0.0	0.76
0.162	1.0	1.0	0.71