ASTRO 5470 Final Project Proposal

Chase Funkhouser

March 26, 2024

1 Introduction

In the study of stellar winds it can be necessary to include the radiative acceleration due to spectral lines from the corona when solving for the velocity of the wind. In my own research concerning the atmospheres of hot Jupiter exoplanets, the same problem comes up. Specifically, we want to determine the velocity profile of a spherical outflowing wind that is pushed both by gas pressure gradients and a line-driving force (in my case, from the Lyman-α line). This velocity is a necessary part of the equation for the mass loss rate, and is thus an integral part of our understanding of atmospheric mass loss and the importance of line-driving in outflows from hot Jupiters.

While a simple isothermal wind solution has been known for many years now thanks to Parker (1958), the inclusion of a line-driven acceleration term can greatly complicate the solution. To see why this is the case, consider the following derivation which draws from Castor, Abbott and Klein (1975). The momentum equation for a steady-state radial line-driven wind is given by:

$$v\frac{\mathrm{d}v}{\mathrm{d}r} = -\frac{GM}{r^2} + \frac{1}{\rho}\frac{\mathrm{d}P}{\mathrm{d}r} + g_l \tag{1}$$

Where g_l is the radiative acceleration due to the line. This could be a general (smooth) function of r as $g_l(r)$, and might not be monotonic. The second equation we use is the law of mass conservation:

$$\dot{M} = 4\pi r^2 v\rho = \text{constant} \tag{2}$$

If we assume the wind is isothermal with sound speed c_s that satisfies the equation of state $P = \rho c_s^2$, then we can combine this with the above mass conservation equation to find:

$$\frac{1}{\rho}\frac{\mathrm{d}P}{\mathrm{d}r} = -\frac{c_s^2}{v}\frac{\mathrm{d}v}{\mathrm{d}r} - \frac{2c_s^2}{r} + \frac{\mathrm{d}c_s^2}{\mathrm{d}r}$$

If our atmosphere is isothermal such that c_s is constant, then we can neglect the final term above. Note that for an ideal gas we have $c_s = \sqrt{\gamma kT/m}$ where γ is the adiabatic index, T the temperature and m the mass of the particles in the gas. This gives the momentum equation as:

$$v\frac{\mathrm{d}v}{\mathrm{d}r} = -\frac{GM}{r^2} + \frac{c_s^2}{v}\frac{\mathrm{d}v}{\mathrm{d}r} + \frac{2c_s^2}{r} + g_l$$

We can group the derivative terms together to obtain the standard form:

$$\frac{\mathrm{d}v}{\mathrm{d}r} = \frac{-GM + 2c_s^2 r + g_l r^2}{r^2(v^2 - c_s^2)/v} \tag{3}$$

Since we want a monotonically increasing velocity law, we require a regularity condition at the sonic point where $v = c_s$. Here, the denominator vanishes, and so too must the numerator at this specific radius satisfying:

$$-GM + 2c_s^2 r_s + g_l r_s^2 = 0$$

Note that in the case of no line acceleration $g_l = 0$, we recover the Parker solution with sonic point $r_s = GM/2c_s^2$. However, in the case where $g_l(r)$ is a general function of r, it is more difficult to determine where the sonic point is, which complicates our numerical integration scheme. In the end, we can calculate the mass loss rate $\dot{M}$ as:

$$\dot{M} = 4\pi r^2 v\rho$$

And should find it to be roughly constant by the mass conservation equation.

2 Methods

Computing the solution for the wind profile $v(r)$ requires solving the above nonlinear differential equation in Eq. (3). To do so we will use the shooting method, which starts the solution at a point with a known value before "shooting" outwards until it reaches a boundary. In the original Parker wind derivation, the initial condition is chosen at the base of the atmosphere, with a given velocity. We will do the same, starting at the base of the atmosphere r_b with velocity v_b and shooting outward.

The main difficulty arrises in dealing with the regularity condition at the sonic point, since we don't know recisely where the sonic point actually is ahead of time. In this case, we guess a value for the sonic point r_s satisfying our regularity condition, then numerically integrate the differential equation inwards to the boundary at the base r_b with velocity v_b, and outwards to the boundary at r_e with velocity v_e. Most likely our guess is wrong, and our solution will fail to behave nicely. In accordance with the Parker wind solution, we expect the solution with a radiative acceleration term to be subsonic at small radii, pass smoothly through the sonic point and become supersonic at large radii. So we could try again with a new point, guided by how our solution failed to behave nicely.

Another method for performing the shooting algorithm with a free internal boundary condition, is given in *Numerical Recipes in Fortran 77*, section 17.6. Here, we can modify the independent variable and recover the usual second boundary condition required by the shooting method. This is the method I will try to implement in the code.

3 Implementation

The main body of the program will be written in Fortran 90 for the speed it will provide an for my own familiarity with the language. Scripts for plotting and visualization of outputs will be written in Python to take advantage of the wealth of available plotting tools.

The Fortran section of the code will include a main program and four modules. One module will contain the code for reading in the input parameters and the imput data for the radiative acceleration, and computing the sound speed c_s and GM factor. Another will set up and run the shooting algorithm to numerically integrate the differential equation to within the desired accuracy. A third module will dump the solution into an output file containing the velocity v and mass loss $\dot{M}$. The fourth module will run the three tests to ensure that the code is working properly.

The inputs to the code will be a namelist with all the relevant input parameters for the model, namely: the surface boundary condition for radius r_b with velocity v_b; the final boundary at the edge to be integrated to r_e; the mass of the object being considered M; an adiabatic index γ, temperature T, and particle mass m used to calculate the sound speed c_s; and a table (.tab) file with the values for a radial grid and the radiative acceleration g_l at each point on that grid. This data will be used to interpolate the value for g_l everywhere within the bounds of the radial grid given; the code will not extrapolate beyond these bounds, so the boundaries given must lie within the boundaries of this grid.

Outputs will include the data for the solution $v(r)$ at each point on the grid of r chosen in a table format (.tab), and from a plotting script in Python these can be used to plot the velocity as a function of r. If the test for mass loss rate (described below) is used, the code will also compute and output the mass loss rate and plot to visually check that it is roughly constant.

4 Code Tests

This code will have three built-in tests that can be run from a separate module. They are as follows:

1. **Parker Wind Solution:** For the first test we will set the radiative acceleration to zero and run the shooting algorithm. Without the radiative acceleration term, the momentum equation should be simple to solve. We can find the sonic point simply from:

$$r_s = \frac{GM}{2c_s^2}$$

   Then shoot both inwards and outwards from there with initial value $v(r_s) = c_s$ until we reach the inner and outer radius boundaries r_b and r_e. The result can then be compared to the analytic Parker wind solution.

2. **Constant Acceleration:** In the case where g_l is a nonzero constant, the sonic point can be found from the quadratic:

$$g_l r_s^2 + 2c_s^2 r_s - GM = 0 \implies r_s = \frac{-2c_s^2 + \sqrt{4c_s^4 + 4g_l GM}}{2g_l}$$

This then provides a test of the code's ability to determine where the sonic point is located, and an analytic form from Castor, Abbott and Klein (1975) can be used to check the validity of the solution.

3. **Constant Mass Loss Rate:** Per the law of mass conservation, the mass loss rate within the wind should be constant, and is given by:

$$\dot{M} = 4\pi r^2 v \rho = \text{constant}$$

To test this, we can simply check the mass loss rate at every radius r once we have found solution for $v(r)$ to make sure that it is roughly constant. Note that this test should work no matter the form for g_l supplied, and thus it could be done every time the code is run.

References

Castor, Abbott, and Klein 1975 "Radiation-Driven Winds in Of Stars", ApJ 195:157
Parker, E. N. 1958 "Dynamics of the Interplanetary Gas and Magnetic Fields", ApJ 128:664
Press, Teukolsky, Vetterling, and Flannery 1986 *Numerical Recipes in Fortran 77: The Art of Scientific Computing*, Cambridge University Press.
Lamers and Cassinelli 1999 *Introduction to Stellar Winds*, Cambridge University Press.