I give permission for public access to my Honors paper and for any copying or digitization to be done at the discretion of the College Archivist and/or the College Librarian.

Full Name: 段雨彤 Duan Yutong

Signature: ______________________

Date: ______________________
Numerical Simulation of Na I D Absorption-Line Profiles Formed By Galactic Outflows

段雨彤 Duan Yutong

Department of Physics
Rhodes College
Memphis, Tennessee

2013

Submitted in partial fulfillment of the requirements for the Bachelor of Science degree with Honors in Physics
This Honors paper by 段雨彤 Duan Yutong has been read and approved for Honors in Physics.

Dr. David S. N. Rupke

Project Advisor

Dr. Rachel Dunwell

Extra-Departmental Reader

Dr. Brent Hoffmeister

Department Chair
# Contents

<table>
<thead>
<tr>
<th>Permission to Copy</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Page</td>
<td>ii</td>
</tr>
<tr>
<td>Signature Page</td>
<td>iii</td>
</tr>
<tr>
<td>Contents</td>
<td>iv</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>Abstract</td>
<td>ix</td>
</tr>
</tbody>
</table>

## 1 Introduction

1.1 Motivation: Galaxy Evolution ............... 1
1.2 Computational Approach: Simulation ......... 3

## 2 Background

2.1 Basic Astronomical Spectroscopy ............ 5
  2.1.1 Absorption Cross-Section ................ 5
  2.1.2 Column Density .......................... 6
  2.1.3 Optical Depth ........................... 7
  2.1.4 Equivalent Width ......................... 8
  2.1.5 Absorption-Line Broadening Processes ..... 9
  2.1.6 Curve of Growth .......................... 12
2.2 Na I D Absorption Feature ................... 13
2.3 Velocity Distribution Functions ............. 15
  2.3.1 Gaussian Distribution ................... 16
## 3 Wind Model

### 3.1 Geometry of Model in Euclidean Space
- 3.1.1 Cartesian Coordinate System
- 3.1.2 Spherical Coordinate System

### 3.2 Characterization of Outflow Gas
- 3.2.1 Number Density of Na Atoms
- 3.2.2 Temperature of Gas
- 3.2.3 Bulk Velocity of Gas
- 3.2.4 Turbulence Profile of Gas
- 3.2.5 Regular Wind Structures

### 3.3 Input Parameters
- 3.3.1 Fundamental and Atomic Constants
- 3.3.2 Galaxy Parameters
- 3.3.3 Wind Parameters
- 3.3.4 Plotting Parameters

### 3.4 Constructing Lattice
- 3.4.1 Grid Structure
- 3.4.2 Galactic Disk Plane
- 3.4.3 Criteria for Wind Shapes
- 3.4.4 Perturbation of Galactic Center

### 3.5 Integration along Line of Sight
- 3.5.1 Calculation of Probability Density
- 3.5.2 Calculation of Observable Quantities

### 3.6 Graph Generation
- 3.6.1 Spectra
- 3.6.2 Plots
- 3.6.3 Colormaps

## 4 Simulation Results

### 4.1 Spherical Winds
- 4.1.1 $v_{\text{bulk}} \propto r^0$
B.10 partvelvec.pro ............................................. 163
B.11 temp_spherical.pro ........................................ 169
B.12 v_add.pro ..................................................... 169
B.13 v_bulk_spherical.pro ....................................... 170
B.14 v2lambda.pro ................................................. 170
B.15 v2nu.pro ..................................................... 171

Bibliography 172
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>HST Image of M82</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>Equivalent Width</td>
<td>8</td>
</tr>
<tr>
<td>2.2</td>
<td>Line Profiles Produced Line-Broadening Processes</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>General Curve of Growth for the Sun</td>
<td>13</td>
</tr>
<tr>
<td>2.4</td>
<td>Na i D Absorption-line Profiles</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Cartesian Coordinate System of Model</td>
<td>21</td>
</tr>
<tr>
<td>3.2</td>
<td>Spherical Coordinate System of Model</td>
<td>23</td>
</tr>
<tr>
<td>3.3</td>
<td>Lattice Space</td>
<td>37</td>
</tr>
<tr>
<td>4.1</td>
<td>Simulation Graphs for Spherical Winds $v \propto r^0$</td>
<td>46</td>
</tr>
<tr>
<td>4.2</td>
<td>Simulation Graphs for Spherical Winds $v \propto r^1$</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Simulation Graphs for Spherical Winds $v \propto r^2$</td>
<td>56</td>
</tr>
<tr>
<td>4.4</td>
<td>Simulation Graphs for Biconical Winds $v \propto r^0$</td>
<td>61</td>
</tr>
<tr>
<td>4.5</td>
<td>Simulation Graphs for Biconical Winds $v \propto r^1$</td>
<td>66</td>
</tr>
<tr>
<td>4.6</td>
<td>Simulation Graphs for Biconical Winds $v \propto r^2$</td>
<td>71</td>
</tr>
<tr>
<td>4.7</td>
<td>Simulation Graphs for Unfilled Biconical Winds $v \propto r^0$</td>
<td>76</td>
</tr>
<tr>
<td>4.8</td>
<td>Simulation Graphs for Unfilled Biconical Winds $v \propto r^1$</td>
<td>81</td>
</tr>
<tr>
<td>4.9</td>
<td>Simulation Graphs for Unfilled Biconical Winds $v \propto r^2$</td>
<td>86</td>
</tr>
<tr>
<td>4.10</td>
<td>Simulation Graphs for Bipolar Bubble Winds $v \propto r^0$</td>
<td>91</td>
</tr>
<tr>
<td>4.11</td>
<td>Simulation Graphs for Bipolar Bubble Winds $v \propto r^1$</td>
<td>96</td>
</tr>
<tr>
<td>4.12</td>
<td>Simulation Graphs for Bipolar Bubble Winds $v \propto r^2$</td>
<td>101</td>
</tr>
</tbody>
</table>
ABSTRACT

Numerical Simulation of Na I D Absorption-Line Profiles Formed By Galactic Outflows

by

Duan Yutong

Galactic outflows may help explain important aspects of galaxy evolution: the regulation of star formation and black hole activity, the energetics of intergalactic medium, the elemental composition of galaxies, and the missing baryons on galaxy scales. The Na I D doublet absorption feature (\(\lambda\lambda \, 5890, \, 5896\)) in the visible spectrum is ideal for studying galactic outflows due to a number of reasons: its presence in the visible range after being redshifted (up to \(z \sim 0.5\)), the relatively high abundance of Na in the interstellar medium, and its resolution of a degeneracy between optical depth and covering fraction. We developed a computational model to simulate the Na I D absorption.

The program is written in the Interactive Data Language (IDL). The program simulates light that is emitted from the galactic disk, passes through a galactic wind of a given structure, and is absorbed in two upward atomic transitions, resulting in Na I D. The galaxy and outflow gas are characterized by predefined physical parameters. Doppler broadening as the primary physical process re-
sponsible for the linewidth is considered along with the turbulence of the gas flow, whereas natural broadening, pressure and collisional broadening, and other radiative transfer processes are neglected.

Line-of-sight synthetic spectra of regular wind structures, including spherical, filled biconical, unfilled biconical, and bipolar bubble winds, are generated as well as column density, equivalent-width, and mean bulk velocity maps. By exploring the dependence of simulation results on wind structure and relating theoretical simulations to observational data, we hope to gain insights into the physical properties of galactic outflows.
Chapter 1

Introduction

1.1 Motivation: Galaxy Evolution

A galaxy primarily consists of stars, gas, and dust, and dark matter that is gravitationally bounded. Stars are formed as interstellar gas undergoes gravitational collapse and becomes dense. The stars and supernova explosions, especially in starburst regions and in high redshift galaxies, provide the radiation and mechanical pressure to push the gas out of the galactic disk. When a considerable amount of gas is ejected from the galaxy, it is expected to decrease the star formation rate in the galaxy. As the distribution and temperature of the gas influences star formation, galactic outflow is related to regulation of star formation rate (SFR)\[1\].

There is a supermassive black hole at the center of each galaxy which can also provide the driving force for galactic outflows\[2\]. The black hole, by its strong gravitational field, attracts nearly stars and gas. A black hole, as it absorbs infalling matter, produces a heated accretion disk and jets of plasma at relativistic speeds. The energy released during accretion may power galactic outflows by
Figure 1.1: An Hubble Space Telescope image of M82, about 10 million light-years from Earth, shows crimson, incandescent gas flowing from its core out of the galaxy on both sides of the galactic disk. (NASA/JPL-Caltech/STScI/CXC/UofA/ESA/AURA/JHU)
radiative transfer processes, thermal expansion, and magnetic fields associated with the relativistic jets. The galactic outflows may be powerful enough to provide negative feedback, regulating both the star formation rate and the growth of the black hole.

Heavy elements are primarily formed through nucleosynthesis in stars and supernova explosions. During supernovae, metals are ejected into interstellar space, carried away by the outflowing interstellar gas. If this gas escapes the galaxy, galactic winds change the elemental composition of the interstellar medium, which is crucial to star formation.

The so-called ”missing baryons” on galaxy scales are another phenomenon for which galactic outflows may be responsible. Observation has shown that rich galaxy clusters retain their baryons in agreement with theoretical predictions, whereas poor galaxy clusters and all galaxies are missing 50 – 95% of their baryons, with dark matter taken into account. One possible explanation is that the missing baryons are ejected by galactic winds into intergalactic space. Observational studies of the intergalactic medium have yet to confirm this theory.

The above phenomena are just a few of many in which galactic winds may play a role. Studies of the feedback mechanisms to star formation and black hole activity, such as the structure and the power sources of winds, the energetics and elemental composition of interstellar and intergalactic media, gas dynamics, and plasma magnetohydrodynamics are key to understanding galaxy evolution.

1.2 Computational Approach: Simulation

Astronomy research has always relied on observation. Interpretation of observational data remains a challenge, as it is not always obvious how physical
processes give rise to spectra and images. Numerical simulation is a good method when a system is too complicated to solve analytically and there are many free parameters. Given sufficient computational power, we are able to simulate the behavior and observable properties of a complex galaxy with outflowing gas to a good approximation by making educated assumptions.

The model built in this project is an approximation of galactic winds. It mainly considers atomic absorption of Na atoms, but does not include radiative transfer processes. The purpose of the model is to simulate Na I D absorption-line profiles for a given galactic wind. These experimental data are commonly obtained from spectroscopy. Being able to simulate absorption spectra based on given input parameters means the ability to infer and recover the physical conditions that produced such spectra. Finding the properties of the gas itself is an essential step toward understanding the mechanisms involving galactic outflow that are key to galaxy evolution.

The resolution of simulation results is constrained by the hardware and algorithms used, but it is comparable to the resolution of typical observational data. Comparing simulation results with observational data will yield insights into the physical properties of galactic outflows.
Chapter 2

Background

2.1 Basic Astronomical Spectroscopy

2.1.1 Absorption Cross-Section

Absorption cross-section is a measure of the probability of an absorption process. For an arbitrary distribution function in frequency space, the normalization condition is

$$\int_0^\infty \phi(\nu) d\nu = 1 \quad (2.1)$$

The integrated absorption cross-section over all frequencies per atom without correction for stimulated emission is

$$s_u = \int_0^\infty s_{jk,\nu} d\nu = \frac{h\nu_{jk}}{c} B_{jk} = \frac{e^2}{4\varepsilon_0 m_e c} f_{jk} \quad (2.2)$$

where $B_{jk}$ is the Einstein coefficient for photo absorption and $f_{jk}$ is the oscillator strength of the transition. As the final expression for $s_u$ is derived from integrating over all frequencies, it is only valid in frequency space. Integrating over
wavelengths does not yield the same expression. From the definition of integral, it follows that multiplying the integrated cross-section \( s_u \) by the frequency distribution function \( \phi(\nu) \) gives the absorption cross-section \( s_{jk,\nu} \) at each frequency, i.e.

\[
s_{jk,\nu} = s_u \phi(\nu) \tag{2.3}
\]

With the correction for stimulated emission, the integrated absorption cross-section over all frequencies per atom is

\[
s_{jk} = s_u (1 - \frac{b_k}{b_j} e^{-\frac{\hbar \nu_{jk}}{k_B T}}) \tag{2.4}
\]

where \( \frac{b_k}{b_j} \) is the ratio of departure coefficients of the two energy levels in the transition[6]. The departure coefficients and the Boltzmann factor correct for stimulated emission by relating actual level populations to thermal equilibrium populations. When \( \hbar \nu_{jk} \gg k_B T \), \( s_{jk} \approx s_u \); absorption dominates and stimulated emission is negligible. When \( \hbar \nu_{jk} \ll k_B T \), stimulated emission becomes important. For Na I D, \( \frac{\hbar \nu_{jk}}{k_B T} \approx 5 \), so it is important to account for stimulated emission.

### 2.1.2 Column Density

Let \( n \) be the number density of atoms. Suppose an observer looks through a tube of a cross section of 1 m\(^2\). Column density refers to the number of absorbing atoms in the tube. It is defined as the number of atoms per unit area. To find the column density, simply integrate the number density of the atoms over distance

\[
N = \int n \, dl \tag{2.5}
\]
where \( l \) is the line of sight distance. \( N \) has units of \( \text{m}^{-2} \).

### 2.1.3 Optical Depth

Optical depth \( \tau \) is a measure of how opaque a medium is to radiation passing through it. It is defined back along a light ray in the following differential form,

\[
d\tau_\nu = -n s_\nu \, dl
\]

where \( s_\nu \) is the absorption cross-section of an atom at a particular frequency as in Equation 2.3 and \( l \) is the path of light. The observer looks back along the path traveled by the photons. The differential form is a more rigorous definition than the integral form that involves column density, since the absorption cross-section may vary along the column of gas due to inhomogeneous temperature profile.

At any frequency, the observed intensity can be calculated from the optical depth by

\[
I_\nu = I_{\nu 0} e^{-\tau}
\]

where \( I_{\nu 0} \) is the original intensity of the light ray. To normalize the difference in the original intensity, the ratio of the two intensities is taken as

\[
\frac{I_\nu}{I_{\nu 0}} = e^{-\tau}
\]

The normalized intensity is more commonly used as it is only dependent on optical depth \( \tau \).
2.1.4 Equivalent Width

Equivalent width of an absorption-line profile is a measure of the strength of absorption, i.e. the amount of light absorbed. Figure 2.1 is a plot of normalized intensity versus wavelength. On the vertical axis, the intensity, or flux, is normalized by taking the observed intensity as a fraction of the original intensity, \( \frac{I_{\lambda}}{I_{\lambda_0}} \). In the left plot, the core of the absorption line is near the central wavelength, where the curve reaches its minimum. The sides sweeping upward to the continuum are the wings. If no light is absorbed, the normalized intensity would be the horizontal line \( y = 1.0 \), which is the continuum. If there is absorption centered at a certain wavelength, it results in the lower bold curve on the left, which is a typical absorption line. The height difference between these two curves, \( \frac{I_{\lambda_0} - I_{\lambda}}{I_{\lambda_0}} \), is referred to as the depth of the absorption line. The shaded area between the two curves represents the amount of light absorbed.

A shaded rectangle is drawn on the right with the same area as the left

![Image](http://ircamera.as.arizona.edu/astr_250/Lectures/Lec_15sml.htm)

Figure 2.1: The shaded area on the left represents the amount of light absorbed, and the shaded rectangle on the right has the same area. The area is numerically equal to the width of the rectangle.
shaded area, and its height is obviously 1 in the plot. Therefore the width of the rectangle is numerically equal to its area, which is also equal to the shaded area on the left. This way, astronomers can use a single quantity that has the unit of length (usually in nm or Å) to quantitatively describe and compare the strength of absorption.

As integrating the depth of absorption line over wavelength yields the shaded area, the mathematical definition of equivalent width is

\[ W_{jk} = \int \frac{I_{\lambda_0} - I_{\lambda}}{I_{\lambda_0}} \, d\lambda \]  

(2.9)

where \( I_{\nu} \) is the observed intensity at a certain wavelength, and \( I_{\nu_0} \) is the original continuum intensity at that wavelength. This area is equal to the width of the rectangle.

2.1.5 Absorption-Line Broadening Processes

Absorption lines cannot be a delta function that is infinitely sharp with no width, even for motionless, isolated atoms. There are three processes that mainly contribute to the spectral linewidth, and they produce different line profiles.

Natural Broadening

According to Heisenberg’s uncertainty principle, the energy-time uncertainty relation is

\[ \Delta E \Delta t \approx \frac{\hbar}{2} \]  

(2.10)

An electron in an excited state occupies the orbital for only a brief moment. As \( \Delta t \) decreases, \( \Delta E \) increases. This means the atomic energy levels are not
exact, which means that the frequency and wavelength are not exact either. This intrinsic uncertainty contributes to the linewidth by about $10^{-5}$ nm, which is negligible compared to other processes.

**Doppler Broadening**

When an atom is not at rest relative to the observer, the relative motion between them shifts the wavelength of a photon absorbed or emitted due to the Doppler effect. The atoms that absorb photons are moving at a range of veloc-
ities, and consequently their absorption lines have different central wavelengths. Although galactic outflows travel at non-relativistic speeds, they can reach as high as about 0.01c. The formula for relativistic Doppler effect is

$$\frac{\nu_{jk}}{\nu} = \frac{\lambda}{\lambda_{jk}} = \sqrt{\frac{1 + \beta}{1 - \beta}},$$

where $\lambda_{jk}$ is the rest wavelength of the transition, $\lambda$ is the observed wavelength, and $\beta \equiv \frac{v}{c}$.

The relative velocity between the atoms and the observer most commonly arise from thermal motion. For this reason, Doppler broadening is sometimes referred to as thermal broadening, which is a subset of Doppler broadening strictly speaking. The Doppler width due to thermal motions is roughly 10$^{-2}$ nm which is three orders of magnitude greater than for natural broadening. Apart from thermal motion, other factors such as turbulence can also introduce velocity dispersion of atoms, causing Doppler broadening. These other factors can lead to even higher linewidths.

**Pressure and Collisional Broadening**

When the number density is high and atoms are close together, the electric fields of large number of closely packed ions greatly reduce upper state lifetimes, and hence broaden the linewidth. This is termed pressure broadening. The orbitals of an atom can also be perturbed in a collision with a neutral atom or by a close encounter involving the electric field of an ion. The results of individual collisions are collisional broadening.

As the number density increases, pressure and collisional broadening become comparable to natural broadening. For hydrogen atoms at a number density
of $1.5 \times 10^{23} \text{m}^{-3}$, pressure and collisional broadening is about $10^{-5} \text{nm}$, still much less than the effect of Doppler broadening. The density of interstellar Na in galactic outflows is about $10^{-1}$ to $10^4 \text{m}^{-3}$, so pressure and collisional broadening can be safely neglected.

### 2.1.6 Curve of Growth

The curve of growth is a logarithmic plot showing how the equivalent width of an absorption line increases with the number of atoms absorbing the light. It can be divided into three parts: the linear part, the flat part, and the square-root part.

At the beginning of the curve, column density is low and the gas is optically thin. In the weak absorption line, doppler broadening dominates, and the equivalent width of the line increases linearly with column density, $W \propto N$. This is the linear part of the curve.

As more atoms are added, the depth of the absorption line increases. The central core of the line then approaches zero intensity and becomes saturated, while the wings of the line continue to deepen, causing little change in the equivalent width. This produces the flat part of the curve of growth where $W \propto \sqrt{\ln N}$.

As even more atoms are added, the gas becomes optically thick. The core of the absorption line is well saturated and does not increase any more. The wings both widen and deepen, making the equivalent width increase more rapidly. The damping wings in the pressure and collisional profile dominates, with $W \propto \sqrt{N}$ for the total line profile.
Figure 2.3: A general curve of growth for the sun. Both axes are scaled to plot the equivalent widths of several lines formed by transitions from the same initial orbital on a single curve of growth.

2.2 Na I D Absorption Feature

The Na I D absorption feature is known as the sodium doublet, the center wavelength of which (589.29 nm) is given the designation letter “D”, and the roman numeral I means ground state. It consists of two absorption lines, the D₁ (5896 Å) and D₂ (5890 Å), which are the fine splitting of the same transition due to different upper level spins. Both lines result from the transition from lower level 2p⁶3s to upper level 2p⁶3p. For the D₁ line, the electron spin changes from $\frac{1}{2}$ to $\frac{3}{2}$, while for the D₂ line the electron spin stays $\frac{1}{2}$. Several characteristics make
Na $\text{i}$ D stand out as a good choice for spectroscopic study of galactic outflows.

Sodium is relatively abundant in interstellar medium in galaxies. As light passes through galactic winds, this ensures a decent absorption strength which then can be used for studying galactic winds. The number density of sodium is related to that of hydrogen, the most abundant element in interstellar medium. An empirical relation that corrects for ionization, depletion, and abundance is

Figure 2.4: The Na $\text{i}$ D absorption-line profiles in spectra of two galaxies, (a) F15549+4201 and (b) F10378+1108.
given by Rupke\cite{rupke2004} as

\[
N(H) = N(\text{Na} \, \text{i})(1 - y)^{-1}10^{-(a+b)}
\]

where \( y \equiv 1 - \frac{N(\text{Na} \, \text{i})}{N(\text{Na})} \) is the ionization fraction, \( a = \log \left( \frac{N(\text{Na})}{N(H)} \right)_{\text{gas}} \) is the Na abundance in the galaxy of interest, and \( b = \log \frac{N(\text{Na})}{N(H)} - \log \left( \frac{N(\text{Na})}{N(H)} \right)_{\text{gas}} \) is the depletion onto dust. Typical values are \( y = 0.9 \), \( a = -5.69 \), and \( b = -0.95 \). This gives \( N(\text{Na} \, \text{i}) \sim N(H) \times 10^{-7} \), which is still relatively abundant compared to other minor species.

Na i D remains in the visible range of the radiation spectrum until \( z \sim 0.5 \), which corresponds to a lookback time of about 5 billion years and age of the universe of about 8.7 billion years. Therefore, low-redshift galaxies up to \( z \sim 0.5 \) can be studied with ground-based optical telescopes, which are more available than non-optical or space telescopes in terms of number and cost.

By using the Na i D doublet, we are also able to resolve degeneracies in optical depth, covering factor, and direction of motion of the gas\cite{rupke2004,landi2009}. The two absorption lines can be fitted simultaneously. As the oscillator strength of Na i D_2 is twice that of Na i D_1, a simultaneous fit usually well restricts degeneracies in solutions. This is a significant advantage over other options that only consist of a single absorption line.

\section{Velocity Distribution Functions}

A probability density function, also referred to as a distribution function, describes the relative likelihood for a continuous random variable to take on a given value. As we have seen, \( \phi(\nu) \) Equation 2.1 is a probability density function.
The statistical random motion of particles are described in terms of a velocity distribution.

2.3.1 Gaussian Distribution

The Gaussian distribution, also known as the normal distribution, is a continuous probability density function that describe a normally distributed random variable. A normalized Gaussian distribution function has the general form

\[ \phi_{\text{Gaussian}}(v, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{v - \mu}{\sigma} \right)^2} \]  

(2.13)

where \( \mu \) is the mean and \( \sigma \) is the standard deviation of the random variable.

A well known result about the Gaussian distribution is that the sum of two independent normal random variables is still a normal variable\[11\], meaning that the sum of Gaussian distribution functions is still a Gaussian. This important property of Gaussian distribution enables us to reduce more than one Gaussian distributions that describe different physical processes to a single Gaussian distribution. The Gaussian distribution has a wide range of applications. In particular, it can be used to describe velocity dispersion in interstellar gas.

2.3.2 Maxwell-Boltzmann Distribution

In classical kinetic theory, the speed of ideal gas particles in 3-dimensional space is given by the Maxwell-Boltzmann velocity distribution

\[ P(v) = 4\pi \left( \frac{m}{2\pi k_B T} \right)^{\frac{3}{2}} v^2 e^{-\frac{mv^2}{2k_BT}} \]  

(2.14)
where $v$ is the magnitude of thermal velocity, or speed. $P(v)$ is the probability density function, and the quantity $P(v)dv$ is the probability that a particle has its speed in $dv$ at $v$.

In astrophysics, the line-of-sight component of velocity is responsible for the redshift in spectral lines, so we are more interested in the component velocity in a single direction. This velocity component is given by the Maxwell-Boltzmann velocity distribution for a single direction

$$P(v_i) = \sqrt{\frac{m}{2\pi k_B T}} e^{-\frac{mv_i^2}{2k_B T}} \tag{2.15}$$

This is a special case of the Gaussian distribution where $\mu = 0$ and $\sigma = \sqrt{\frac{k_B T}{m}}$.

### 2.4 Transformation of Distribution Function

A velocity distribution function describes an ensemble of particles moving at different velocities. For each atom, the central frequency or wavelength of the transition is related to its velocity by the Doppler effect formula as in Subsection 2.1.5

$$\frac{\nu_{jk}}{\nu} = \frac{\lambda}{\lambda_{jk}} = \sqrt{\frac{1 + \beta}{1 - \beta}} \tag{2.11}$$

For a given transition centered at $\nu_{jk}$, each velocity corresponds to a shifted central frequency. The Doppler effect equation relates velocity to frequency, or wavelength. This way, the velocity distribution can be converted into a distribution in frequency or wavelength space. Despite any transformation, the distribution will always be normalized.

To convert velocity distribution to frequency distribution, first from Equation
and \( \beta \equiv \frac{v}{c} \), the derivative can be taken as

\[
\frac{dv}{d\nu} = \frac{-4 \nu_{jk}^2 \nu}{(\nu_{jk}^2 + \nu^2)^2 c}\tag{2.16}
\]

Then note that both distribution functions are normalized

\[
\int_{-\infty}^{\infty} \phi(v) \, dv = \int_{0}^{\infty} \phi(\nu) \, d\nu = 1 \tag{2.17}
\]

which means the probability of finding a particle in a differential interval is conserved under transformation

\[
\phi(v) dv = \phi(\nu) d\nu \tag{2.18}
\]

Rearrange terms to get the frequency distribution

\[
\phi(\nu) = \phi(v) \frac{dv}{d\nu} = \phi(v) \frac{-4 \nu_{jk}^2 \nu}{(\nu_{jk}^2 + \nu^2)^2 c} \tag{2.19}
\]

Similarly, the velocity distribution can be converted to wavelength distribution. Take the derivative

\[
\frac{dv}{d\lambda} = \frac{4 \lambda_{jk}^2 \lambda}{(\lambda_{jk}^2 + \lambda^2)^2 c} \tag{2.20}
\]
Then the transformation follows as

\[
\phi(v)dv = \phi(\lambda)d\lambda
\]

\[
\phi(\lambda) = \phi(v) \frac{dv}{d\lambda}
\]

\[
= \phi(v) \frac{4\lambda_{jk}^2 \lambda}{(\lambda_{jk}^2 + \lambda^2)^2 c}
\]

(2.21)
Chapter 3

Wind Model

3.1 Geometry of Model in Euclidean Space

3.1.1 Cartesian Coordinate System

In this model, we assume a disk galaxy. A simple circular disk with a certain thickness is used to model the galaxy. For elliptical galaxies it is more complicated. As a galaxy is a 3-dimensional object, 3-dimensional arrays are used to model the Euclidean space where a galaxy and its galactic wind are located. A galaxy sits at the center of the space, and the galactic center is the origin in 3-dimensional Cartesian coordinate system. Galactic wind is additionally superimposed around the galaxy according to the wind shape specified.

In observation when images are acquired, a 3-dimensional galaxy is projected onto a 2-dimensional plane perpendicular to the line of sight. Each galactic disk observed has an inclination angle, $I$, with respect to the line of sight. If this projection plane, which, in fact, is the celestial sphere, is defined as $I = \frac{\pi}{2}$, some galaxies we see face-on ($I = \frac{\pi}{2}$), some edge-on ($I = 0$), and some in between
\( (I \in (0, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)) \). The definition used in the model may differ slightly from conventional definitions. We use a range of \((0, \pi)\) to be able to specify any orientation. Since azimuthal symmetry of the galaxy is assumed, there is virtually only one free parameter. Experimentally, 2-dimensional images can be rotated and consistently aligned within the image plane to eliminate the extra degree of freedom, so any orientations of a galaxy can be characterized solely by the inclination angle \( I \).

![Cartesian coordinate system](image)

Figure 3.1: The Cartesian coordinate system used in the model: The \( x \)-axis points into the page, so the coordinate system is canonical, right-handed. The line of sight is in the \(-z\) direction, and the image observed is a projection onto a plane parallel to \( xy \)-plane. The line of nodes is always on the \( x \)-axis. Inclination is defined as the angle in the \( yz \)-plane relative to \(+z\) direction.

In Cartesian coordinates, the line of sight is fixed along the \(-z\) direction. That is, the observer looks at the galaxy around the origin from the \(+z\) direction toward the \(-z\) direction, which means the image plane is perpendicular to the \( z \)-
axis, parallel to the $xy$ plane. The line of nodes is the intersection of the object’s orbital plane with the plane of reference. The rotational degree of freedom of the galaxy parallel to the $xy$ plane is eliminated by forcing the line of nodes to horizontally overlap the $x$-axis. Therefore in the model only one parameter, $I$, is used to specify the orientation of the galactic disk as seen by the observer. For future comparison to observational data, the rotation in the image plane is expected to have been corrected.

In Euclidean space, a plane can be specified by four coefficients and have the general form

$$C_1x + C_2y + C_3y + C_4 = 0. \quad (3.1)$$

The normal vector to the plane is

$$\mathbf{n} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \end{bmatrix} \quad (3.2)$$

The galactic disk passes through the origin, so $C_4 = 0$. Pick a point $(x_0, y_0, z_0)$ on the plane, and the coordinates satisfy the plane equation

$$C_1x_0 + C_2y_0 + C_3z_0 = 0 \quad (3.3)$$

The vector $\mathbf{u}$ from $(x_0, y_0, z_0)$ to any position $(x, y, z)$ is

$$\mathbf{u} = \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} \quad (3.4)$$
The distance from any position \((x, y, z)\) to the plane is then the projection of \(u\) onto the perpendicular direction, or the normal vector direction. Take the dot product of \(u\) and the unit normal vector

\[
d = \mathbf{u} \cdot \frac{\mathbf{n}}{|\mathbf{n}|} = \frac{C_1(x - x_0) + C_2(y - y_0) + C_3(z - z_0)}{\sqrt{C_1^2 + C_2^2 + C_3^2}} = \frac{C_1 x + C_2 y + C_3 z}{\sqrt{C_1^2 + C_2^2 + C_3^2}} \tag{3.5}
\]

### 3.1.2 Spherical Coordinate System

Figure 3.2: The spherical coordinate system used in the model: The polar axis is perpendicular to the galactic disk, which is the \(\theta = \frac{\pi}{2}\) plane in the spherical coordinate system. As the galaxy is tilted from the \(xz\)-plane (horizontal), the line of nodes is always the \(x\)-axis.
Besides the Cartesian coordinate system, a spherical coordinate system with the same origin is established, which is fixed to the galaxy and the wind. The galaxy is modeled as a circular disk with a given thickness, which provides the background radiation continuum. The spherical coordinate system is set up such that the polar axis aligns with the normal vector of the disk plane. This way, the galactic disk lies in the $\theta = \frac{\pi}{2}$ plane, and the galaxy is assumed to be azimuthally symmetric. For a given galaxy disk plane, its normal vector gives the polar axis direction, which is useful for calculating the polar angle coordinate of each location in space.

### 3.2 Characterization of Outflow Gas

The outflow gas is mainly characterized by four quantities: number density, temperature, bulk velocity, and turbulence profile. The dependencies of these quantities on radial distance $r$ and polar angle $\theta$ are assumed to have predefined functional forms.

#### 3.2.1 Number Density of Na Atoms

The number density profile of outflow gas is constrained by conservation of mass, or particle number, in an expanding spherical shell in the case of spherical wind. Suppose there are a total of $N_{textNa}$ Na atoms at the beginning of the expansion at the galactic center. The number of Na atoms in a spherical shell

$$dN_{textNa} = n \cdot 4\pi r^2 dr$$

(3.6)
is constant as it expands outwards. Therefore the number density $n$ necessarily goes as $n \propto r^{-2}$ per the conservation law. This is also true for filled and unfilled biconical winds. For bipolar wind bubbles, the same relation is used for simplicity. As $n \propto r^{-2}$, the functional form of number density is

$$n(r, \theta) = C_{n1} r^{C_{n2}} f(\theta) \quad (3.7)$$

where $C_{n2} = -2$, and $C_{n1}$, the number_density_coefficient, is determined by boundary conditions, namely the number density at the galactic center ($r = 0$) and outer boundary ($r = r_{\text{galaxy}}$). A problem arises here when calculating the number density at the galactic center, as plugging in $r = 0$ yields $n(0) = 0$, which is nonphysical. To resolve this, perturbative treatment is given to the origin cell that represents galactic center, as further discussed in Subsection 3.4.4. The $\theta$-dependence is fully modifiable and optional. For example, $f(\theta) = \cos(\theta)$ or $f(\theta) = \cos^2(\theta)$ can be used. The models uses $f(\theta) = 1$ for spherically symmetric winds.

### 3.2.2 Temperature of Gas

The temperature of the outflowing gas does not have a well-constrained profile, and is assumed to have the general form

$$T(r) = T_0 + C_{T1} r^{C_{T2}} \quad (3.8)$$

where $C_{T1}$, the temp_coefficient is calculated from boundary conditions. $T_0$, the core temperature, and $C_{T2}$, the temp_exp, are user-defined. $\theta$-dependence of temperature can be added but was not used in this project for simplicity. It
is known that the temperature must drop as radius increases, so the exponent \( C_{T2} \in \{-3, -2, -1\} \) for simulation purposes.

Note that the temperature of outflows is expected to have little effect on the simulation results, because thermal broadening is negligible compared to bulk velocity and turbulence contributions, as we will see in the results in Chapter 4.

### 3.2.3 Bulk Velocity of Gas

Though there are some measurements, the velocity profile of outflow gas is not well-understood either, and is assumed to have the general form

\[
v_{\text{bulk}}(r) = v_{\text{bulk}0} + C_{v_{\text{bulk}1}} r^{C_{v_{\text{bulk}2}}} \tag{3.9}
\]

where \( C_{v_{\text{bulk}1}} \), the \texttt{v\_bulk\_coefficient} is calculated from boundary conditions. Similarly, \( v_{\text{bulk}0} \), the initial velocity, and \( C_{v_{\text{bulk}2}} \), the \texttt{v\_bulk\_exp}, are user-defined. Again, \( \theta \)-dependence of temperature can be added but was not used in this project for simplicity. Previous measurements have shown that the exponent is around 2.5\([13, 14]\), so we assume the exponent \( C_{v_{\text{bulk}2}} \in \{0, 1, 2\} \) for simulation purposes.

### 3.2.4 Turbulence Profile of Gas

Recall that for interstellar Na in galactic outflows, Doppler broadening dominates as previously discussed (Subsection 2.1.5). Natural broadening and pressure and collisional broadening are orders of magnitude less and can be neglected. To simulate the line-broadening process and give rise to the linewidth, the model only considers Doppler broadening, which is a valid assumption. In the case of galactic outflows, thermal motion and turbulence are two main factors that cause
the velocity dispersion in Doppler broadening.

Given the geometry of the gas model in Section 3.1, the line-of-sight velocity is the $z$-component of the absolute velocity of atoms. The absolute velocity of the gas in each lattice cell is the sum of three velocities

$$v_{\text{abs}} = v_{\text{thermal}} + v_{\text{turbulence}} + v_H \tag{3.10}$$

where $v_H$ is the recession velocity due to Hubble expansion of spacetime. The $z$-component of thermal motion is the 1-dimensional Maxwell-Boltzmann speed distribution, and the $z$-component of the turbulence is a Gaussian distribution. Thanks to the result in Subsection 2.3.1, a single Gaussian distribution function can be used to describe both the $z$-component of bulk velocity and the turbulence of gas flow in the line-of-sight direction. For simplicity, this Gaussian distribution function is referred to as the turbulence profile in the model.

As the formula of a Gaussian distribution is given in Section 2.3, the turbulence profile used in the model as a probability density function for velocity distribution is

$$\phi(v, \mu, \sigma) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2} \left( \frac{v-\mu}{\sigma} \right)^2} \tag{3.11}$$

where $\mu$ is the mean velocity and $\sigma$ is the standard deviation. This Gaussian turbulence profile is the velocity distribution that accounts for both thermal motion and turbulence effects.

The model also provides the option of solely using the Maxwell-Boltzmann velocity distribution, which only accounts for the thermal motion and as a function of temperature $T$. From Subsection 2.3.2, the $z$-component velocity distribution
is given by

$$P(v_z, T) = \sqrt{\frac{m}{2\pi k_B T}} e^{-\frac{mv_z^2}{2k_BT}}$$  \hspace{1cm} (3.12)

### 3.2.5 Regular Wind Structures

We consider four geometrically regular wind structures to begin the modeling with: spherical, filled biconical, unfilled biconical, and bipolar bubble winds.

For the spherical wind, the wind shape is dependent on only one parameter, the radius of the wind $r_{\text{wind}}$ as given in Equation 3.40.

For the filled biconical wind, apart from $r_{\text{wind}}$, the bicone is defined by the half opening angle of the cone with respect to the polar axis in the spherical coordinate system.

For the unfilled biconical wind, the wind is additionally defined by a thickness parameter in terms of the polar angle interval.

For bipolar bubble wind, there are two ellipsoid shells on both sides of the galactic disk tangent to the galactic disk. To describe this, we establish a secondary Cartesian coordinate system in the same orientation as the spherical coordinate system. The conversion of coordinates is

$$x = r \sin \theta \cos \varphi$$  \hspace{1cm} (3.13)

$$y = r \sin \theta \cos \varphi$$  \hspace{1cm} (3.14)

$$z = r \cos \theta$$  \hspace{1cm} (3.15)

The general form for an ellipsoid surface in Cartesian coordinates is

$$\frac{(x - x_0)^2}{R_x^2} + \frac{(y - y_0)^2}{R_y^2} + \frac{(z - z_0)^2}{R_z^2} = 1$$  \hspace{1cm} (3.16)
Because the ellipsoids are tangent to the galactic disk, $z_0 = R_z$. With $x_0 = y_0 = 0$, the origins of the two ellipsoids are $(0, 0, \pm R_z)$ in the secondary Cartesian coordinate system. Assume azimuthal symmetry $R_x = R_y = R_{xy}$. Now Equation 3.16 becomes

$$\frac{x^2}{R_x^2} + \frac{y^2}{R_y^2} + \frac{(z \mp R_z)^2}{R_z^2} = 1$$  \hspace{1cm} (3.17)

$$\frac{x^2 + y^2}{R_{xy}^2} + \frac{(z \mp R_z)^2}{R_z^2} = 1$$  \hspace{1cm} (3.18)

Convert to spherical coordinates and we have

$$\frac{(r \sin \theta)^2}{R_{xy}^2} + \frac{(r \cos \theta \mp R_z)^2}{R_z^2} = 1$$  \hspace{1cm} (3.19)

Here the outer surface of the ellipsoids are defined by two radius variables, one in the polar axis direction, one in the perpendicular plane assuming azimuthal symmetry. To get the bubble shell, we can define the inner surface by simply restricting the value of the left-hand side of Equation 3.19 to $(0, 1)$.

### 3.3 Input Parameters

#### 3.3.1 Fundamental and Atomic Constants

First it should be noted that all data and calculation in this model is done in standard SI (metric) units for consistency. Unit conversion usually only takes place at the last steps, for example, when a graph is plotted in units that are more commonly recognizable.

Fundamental constants such as elementary charge $e$, Planck constant $h$, and
speed of light $c$ are involved in many aspects of the model calculations, from calculating energy level populations to converting distribution functions through relativistic Doppler shift. A library file was created to store their values for easier loading. The following is a list of fundamental constants used.

\[
\begin{align*}
  e & = 1.60217656535 \times 10^{-19} \text{ C} & \text{elementary charge (3.20)} \\
  h & = 6.6260695729 \times 10^{-34} \text{ J} \cdot \text{s} & \text{Planck constant (3.21)} \\
  c & = 299792458 \text{ m} \cdot \text{s}^{-1} & \text{speed of light (3.22)} \\
  \varepsilon_0 & = 8.85418782 \times 10^{-12} \text{ C}^2 \cdot \text{m}^{-2} \cdot \text{N}^{-1} & \text{permittivity of free space (3.23)} \\
  k_B & = 1.380648813 \times 10^{-23} \text{ J} \cdot \text{K}^{-1} & \text{Boltzmann constant (3.24)} \\
  m_e & = 9.1093821545 \times 10^{-31} \text{ kg} & \text{electron rest mass (3.25)} \\
  m_u & = 1.66053892173 \times 10^{-27} \text{ kg} & \text{unified atomic mass unit (3.26)} \\
  pc & = 3.08567758 \times 10^{16} \text{ m} & \text{size of a parsec (3.27)}
\end{align*}
\]

Calculation of the integrated absorption cross-section for a particular transition requires the atomic mass, oscillator strength and rest wavelength of the transition. The rest wavelengths of the two transitions observed in vacuum were chosen as opposed to the observed wavelengths in air or the Ritz wavelength in air. This is because the model is only concerned with Na I D absorption, and assumes that the interstellar space is vacuum besides where the outflows gas is present. The Na I D data were taken from the NIST Atomic Spectra Database as well as relevant papers\cite{15, 16}. The values of two Na I D transitions used in the
model are as follows.

\[ m_{Na} = 22.989769282 \, m_u \]  

rest atomic mass of Na \hspace{1cm} (3.28)

\[ f_1 = 0.319913 \]  

oscillator strength of Na \( \text{i} \) D\textsubscript{1} \hspace{1cm} (3.29)

\[ f_2 = 0.640511 \]  

oscillator strength of Na \( \text{i} \) D\textsubscript{2} \hspace{1cm} (3.30)

\[ \lambda_1 = 5897.56661715 \times 10^{-10} \, \text{m} \]  

wavelength of Na \( \text{i} \) D\textsubscript{1} in vacuum \hspace{1cm} (3.31)

\[ \lambda_2 = 5891.58326415 \times 10^{-10} \, \text{m} \]  

wavelength of Na \( \text{i} \) D\textsubscript{2} in vacuum \hspace{1cm} (3.32)

### 3.3.2 Galaxy Parameters

The size of the galaxy is specified by \texttt{r\_galaxy} in meters. The 3-dimensional resolution of the lattice is controlled by \texttt{grid\_radius}, which specifies the half number of lattice cells along each dimension. As there is an origin at the center, the resulting resolution is \((2 \times \texttt{grid\_radius} + 1)^3\). By default the radius of the grid space corresponds to the radius of the galaxy, as a space of size \texttt{r\_galaxy}\(^3\) is usually sufficient to accommodate the galactic outflows. The side length of each lattice cell is therefore \( \frac{\texttt{r\_galaxy}}{\texttt{grid\_radius}} \).

The thickness of the galactic disk is controlled by \texttt{distance\_offset} in unit of cells. For example, if \texttt{distance\_offset} = 1 then the thickness of the disk plane is 3 pixel lengths. The inclination of the galactic disk is specified by \texttt{inclination} in radians with respect to the projection plane, as described in Section 3.1. The Hubble velocity causes cosmological redshift of the absorption lines observed, and is included as \texttt{v\_hubble}.

When seen from the \( +z \)-axis, since the line of nodes is in alignment with the horizontal \( x \)-axis, the measured vertical radius is the actual galaxy radius projected onto the celestial sphere, and the inclination can be alternatively cal-
culated by $\frac{r_{\text{proj}}}{r_{\text{galaxy}}}$, if the projected radius is input instead of inclination angle. The continuum intensity $I_0$ is assumed to be constant, wavelength-independent.

A range of inclinations from about $0^\circ$ to $180^\circ$ was used. Although the model is symmetric, i.e. $(0, \frac{\pi}{2})$ and $(\frac{\pi}{2}, \pi)$ produce vertically inverted images, the entire $\pi$ range is used for completeness. Because when $\text{inclination} = 0$ or $\pi$ produces no absorption and meaningful results, $2^\circ$ and $178^\circ$ are used instead. A small perturbation is given to the galactic center to avoid null values of physical quantities. The distance shift is defined by $\text{center}_\text{perturb}$. In the model, a half of the cell length is used.

The values used in the model simulation are as follows.

$$r_{\text{galaxy}} = 5 \text{ kpc}$$  \hspace{1cm} (3.33)

$$\text{grid}_\text{radius} = 50$$  \hspace{1cm} (3.34)

$$\text{distance}_\text{offset} = 1$$  \hspace{1cm} (3.35)

$$\text{inclination} \in \{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\}$$  \hspace{1cm} (3.36)

$$I_{\lambda 0} = 1$$  \hspace{1cm} (3.37)

$$\text{center}_\text{perturb} = \frac{1}{2} \frac{r_{\text{galaxy}}}{\text{grid}_\text{radius}}$$  \hspace{1cm} (3.38)

### 3.3.3 Wind Parameters

#### General Wind Parameters

The model supports four built-in wind types of regular structures: spherical, filled biconical, unfilled biconical, and bipolar bubble winds. The $\text{windtype}$ keyword specifies the wind type. The $\text{r}_\text{wind}$ defines the maximum radius of the wind. The $\text{vdt}$ variable, velocity distribution type, controls whether Maxwell-
Boltzmann velocity distribution for thermal motion (1) or a Gaussian function for thermal motion and turbulence (2) is used for generating the speed distribution function of gas in the \(z\)-direction in each lattice cell.

Values used:

\[
\text{windtype} \in \{1, 2, 3, 4\} \tag{3.39}
\]
\[
r_{\text{wind}} = r_{\text{galaxy}} \times 0.7 \tag{3.40}
\]
\[
v_{dt} = 2 \tag{3.41}
\]

**Coefficients**

The functional forms of number density, temperature, and bulk velocity of the gas are given in Section 3.2. Some coefficients take user-defined values; some are calculated based on boundary conditions. The range variables are 2-element arrays defining the boundary conditions at the center and boundary of the wind.

User input values:

\[
\text{number\_density\_outer} = 10^{-5} \text{ m}^{-3} \tag{3.42}
\]
\[
\text{number\_density\_exp} = -2 \tag{3.43}
\]
\[
\text{temp\_range} = [10^4, 10^3] \text{ K} \tag{3.44}
\]
\[
\text{temp\_exp} \in \{-2, -1, 0\} \tag{3.45}
\]
\[
\text{v\_bulk\_range} = [0, 500] \text{ km} \cdot \text{s}^{-1} \tag{3.46}
\]
\[
\text{v\_bulk\_exp} \in \{0, 1, 2\} \tag{3.47}
\]

The other coefficients are calculated within the program. For number density,
according to Equation 3.7 we have

\[ C_{n1} = \frac{n_{outer}}{r_{\text{wind}}^{C_{n2}}} \]  

(3.48)

For temperature, from Equation 3.8, the coefficient is

\[ C_{T1} = \frac{T_2 - T_1}{r_2^{C_{T2}} - r_1^{C_{T2}}} \]  

(3.49)

and the intercept \( T_0 \) is

\[ T_0 = T_2 - C_{T1} r_2^{C_{T2}} \]  

(3.50)

For bulk velocity, from Equation 3.9 similarly the coefficient is

\[ C_{v_{\text{bulk}1}} = \frac{v_{\text{bulk}2} - v_{\text{bulk}1}}{r_2^{C_{v_{\text{bulk}2}}} - r_1^{C_{v_{\text{bulk}2}}}} \]  

(3.51)

And the intercept \( v_{\text{bulk}0} \) is

\[ v_{\text{bulk}0} = v_{\text{bulk}2} - C_{v_{\text{bulk}1}} r_2^{C_{v_{\text{bulk}2}}} \]  

(3.52)

Note that for temperature and bulk velocity, when the exponent is assumed to be 0, the quantity is constant. The intercept needs to be calculated differently from above.
Velocity Distribution Parameters

The turbulence profile is a Gaussian function as in Subsection 3.2.4. The mean velocity $\mu$ and standard deviation $\sigma$ are defined by the following variables.

$$\mu = 0 \quad (3.53)$$

$$\sigma = 100 \ \text{km} \cdot \text{s}^{-1} \quad (3.54)$$

Wind Shape Parameters

For $\text{windtype} = 1$, the spherical wind, the wind shape is dependent on only one parameter, the radius of the wind $r_{\text{wind}}$ as given in Equation 3.40.

For $\text{windtype} = 2$, the filled biconical wind, apart from $r_{\text{wind}}$, the bicone is defined by $\text{bicone\_angle}$, the half opening angle with respect to the polar axis in the spherical coordinate system as in Subsection 3.2.5.

For $\text{windtype} = 3$, the unfilled biconical wind, both cones are assumed to share the same shape and parameters, and are additionally defined by $\text{bicone\_thickness}$, a thickness parameter in terms of the polar angle interval. It is half of the actual thickness of the cones.

For $\text{windtype} = 4$, the bipolar bubble wind, in secondary Cartesian coordinates, the radius in $xy$-plane is defined by $\text{bipolar\_rxy}$, and the radius in $z$-direction by $\text{bipolar\_rz}$. As discussed in Subsection 3.2.5, the inner surface of the bubble shells is defined by $\text{bipolar\_lowerbound}$, which specifies the lower bound value of the left-hand side of Equation 3.19.
CHAPTER 3

Values used:

\[ \text{bicone_angle} = \frac{\pi}{4} \]  \hspace{1cm} (3.55)
\[ \text{bicone_thickness} = \frac{\pi}{13} \]  \hspace{1cm} (3.56)
\[ \text{bipolar_rxy} = 2.5 \text{ kpc} \]  \hspace{1cm} (3.57)
\[ \text{bipolar_rz} = 2.3 \text{ kpc} \]  \hspace{1cm} (3.58)
\[ \text{bipolar_lowerbound} = 0.3 \]  \hspace{1cm} (3.59)

3.3.4 Plotting Parameters

The following variables are for plotting purposes. The wavelength range to be observed can be specified by \( \lambda_{\text{min}} \) and \( \lambda_{\text{max}} \), and the spacing by \( \Delta\lambda \). The velocity range for the Gaussian turbulence profile is specified by the absolute value of maximum speed \( v_{\text{gaussian max}} \), and the spacing by \( \Delta v_{\text{gaussian}} \).

Values used:

\[ \lambda_{\text{min}} = 5800 \text{ Å} \]  \hspace{1cm} (3.60)
\[ \lambda_{\text{max}} = 5950 \text{ Å} \]  \hspace{1cm} (3.61)
\[ \Delta\lambda = 0.10 \text{ Å} \]  \hspace{1cm} (3.62)
\[ v_{\text{gaussian max}} = 800 \text{ km} \cdot \text{s}^{-1} \]  \hspace{1cm} (3.63)
\[ \Delta v_{\text{gaussian}} = 0.05 \text{ km} \cdot \text{s}^{-1} \]  \hspace{1cm} (3.64)
3.4 Constructing Lattice

3.4.1 Grid Structure

Figure 3.3: An illustration to help visualize the lattice space.

The model uses grid, a structure variable to store the physical parameters of the outflows gas at each lattice site. Each field of grid is a physical quantity as a 3-dimensional array. The grid occupies a space of dimension $(2 \times \text{grid} \_\text{radius} + 1)^3$. The normal vector to the galactic disk is calculated from the inclination
angle. The grid structure consists of the following fields.

\[
\text{grid} = \{ \\
    x, \quad \text{index of x coordinate} \\
y, \quad \text{index of y coordinate} \\
z, \quad \text{index of z coordinate} \\
r, \quad \text{radial coordinate} \\
\text{theta}, \quad \text{polar angle} \\
\text{number\_density}, \quad \text{number density} \\
\text{temp}, \quad \text{temperature} \\
v\_\text{bulk}, \quad \text{magnitude of bulk velocity} \\
v\_\text{bulk\_x}, \quad \text{x component of bulk velocity} \\
v\_\text{bulk\_y}, \quad \text{y component of bulk velocity} \\
v\_\text{bulk\_z}, \quad \text{z component of bulk velocity} \\
\text{plane}, \quad \text{if the cell lies in the galactic disk} \\
\}
\]

Then the program starts looping over all lattice sites and selectively filling in the above physical parameters.

### 3.4.2 Galactic Disk Plane

The determination of whether a cell lies in the galactic plane or not is important, as it is the galactic disk that serves as the light source and produces the background continuum. If there is no “plane cell” in the line of sight, the flux is 0, and if there is, the integration along line of sight stops at the galactic disk, as we cannot see through it.
In Subsection 3.1.1, Equation 3.5 gives the distance from any point to a given plane. This way, the program in the loop calculates the distance for each lattice cell. If the distance to plane is no greater than \( \text{distance\_offset} \), and if the distance to the origin is no greater than \( r\_\text{wind} \), then the cell is labeled as a plane cell.

### 3.4.3 Criteria for Wind Shapes

For different wind structures, the program uses different sets of criteria to determine if the lattice cell has gas in it or not.

**Spherical Wind**

When \( \text{windtype} = 1 \), the relation between \( \text{grid\_r}(i,j,k) \) and \( r\_\text{wind} \) is the only criterion. The condition for a cell being outside the wind is

\[
\text{grid\_r}(i,j,k) > r\_\text{wind}
\]

(3.66)

If this is true, the number density \( \text{grid\_number\_density}(i,j,k) \) is set to 0 and the program moves on. Otherwise the program calculates all relevant physical parameters. This condition is used in the first three wind types.

**Filled Biconical Wind**

When \( \text{windtype} = 2 \), there is a second condition that makes the cell out of the wind besides Equation 3.66. The two bicones are centered around \( \theta = 0 \) and \( \theta = \pi \), so defining the condition of being outside the wind by comparing to the
galactic disk \( (\theta = \frac{\pi}{2}) \) is simpler. The cell is outside the wind if

\[
|\text{grid.theta}(i,j,k) - \frac{\pi}{2}| < \frac{\pi}{2} - \text{bicone_angle} \tag{3.67}
\]

is true where \( \text{bicone_angle} \) is the half opening angle of each cone.

**Unfilled Biconical Wind**

When \( \text{windtype}=3 \), it is easier to specify the condition for inside the wind. The opposite of Equation \( 3.66 \) is used

\[
\text{grid.r}(i,j,k) \leq r_{\text{wind}} \tag{3.68}
\]

In addition, the cell must lie in one of the polar angle intervals defined as

\[
|\text{grid.theta}(i,j,k) - \text{bicone_angle}| < \text{bicone_thickness} \tag{3.69}
\]

\[
|\text{grid.theta}(i,j,k) - (\pi - \text{bicone_angle})| < \text{bicone_thickness} \tag{3.70}
\]

If Equation \( 3.68 \) and one of Equations \( 3.69 \) and \( 3.70 \) are true, then the cell is in the wind. Otherwise, the program sets the number density \( \text{grid.number_density}(i,j,k) \) to 0.

**Bipolar Bubble Wind**

When \( \text{windtype}=4 \), the bipolar bubble wind is different from others as it involves two centers that are not the origin. We use the values of the left-hand
side of Equation 3.19 as in Subsection 3.2.5 to constrain the wind.

\[
\text{bipolar}_{\text{lowerbound}} \leq \frac{(r \sin \theta)^2}{R_{xy}^2} + \frac{(r \cos \theta \mp R_z)^2}{R_z^2} \leq 1 \tag{3.71}
\]

The value of \text{bipolar}_{\text{lowerbound}} was chosen to be 0.3 for proper thickness of the bubble shell.

### 3.4.4 Perturbation of Galactic Center

The functional form of number density of Na assumed in Subsection 3.2.1 does not work for the galactic center where \( r = 0 \). Therefore the galactic center is shifted by a half of a cell length as discussed in Subsection 3.3.2. Define the shift distance

\[
\text{center}_{\text{perturb}} = \frac{1}{2} \frac{r_{\text{galaxy}}}{\text{grid}_{\text{radius}}} \tag{3.72}
\]

This shift is applied to the radial coordinate of the origin cell

\[
\text{grid}.r(\text{grid}_{\text{radius}}, \text{grid}_{\text{radius}}, \text{grid}_{\text{radius}}) = \text{center}_{\text{perturb}} \tag{3.73}
\]

Then the program finishes calculating the physical parameters for the origin.

### 3.5 Integration along Line of Sight

For each pixel observed on the 2-dimensional projection plane, the telescope is able to detect the light intensities at a wide range of wavelengths. Each pixel has its associated spectrum, and is also referred to as a spaxel. The program simulates the synthetic spectrum for each spaxel, and loop over all spaxels thereby producing 2-dimensional maps comparable to observational data. As line of sight
is fixed along the \( z \)-direction, the model integrates backwards along the light ray from \(+z\) to \(-z\)-direction, until it hits the galactic disk which can be detected from the value of \texttt{grid.plane}.

### 3.5.1 Calculation of Probability Density

For a given spaxel \((i, j)\), the program integrates over \( k \) toward \(-z\)-direction. As it gets to each lattice site, the physical properties of the gas in the cell is given by the corresponding values in the \texttt{grid} structure. The integrated absorption cross-section is calculated from Equation 2.2, which depends on the temperature at the site.

Then the program matches the wavelength range of interest with the velocity range in the rest frame of the gas by vector-adding the Gaussian turbulence profile and the Hubble recession velocity. Using a given velocity distribution function, either Maxwell-Boltzmann distribution or Gaussian distribution, the program transforms the velocity distribution to frequency distribution and calculates the probability density at each frequency by Equation 2.19 as shown in Section 2.4.

### 3.5.2 Calculation of Observable Quantities

From Equation 2.3, the column density within each cell can be calculated. Summing over all cells up to the galactic disk yields the total column density along the line of sight.

From Equation 2.3, the absorption cross-section at each wavelength in a cell is calculated. From Equation 2.6, the optical depth contribution of the cell is calculated. By integrating along the line of sight, the total optical depth and the resulting intensities at each wavelength are calculated from Equation 2.8.
The equivalent width for each spaxel is then obtained from Equation 2.9. A manual integration function was written based on the classical trapezoidal rule as in Appendix B.2. The program also calculates the mean bulk velocity of the gas along line of sight for plotting.

### 3.6 Graph Generation

There are mainly three ways used in the model to render the simulation results: synthetic spectra showing the absorption-line profiles, data plots of assumed physical dependencies, and colormaps.

#### 3.6.1 Spectra

Viewing synthetic spectra is helpful to validate the simulation, as spectra are the most common observational data. The resolution of 2-dimensional projected images can range from $40 \times 40$ to $100 \times 100$. Plotting the spectra for all spaxels is impractical. For this reason, the program only plots the central spaxel, which is the galactic center. However, this is often not where the absorption is the greatest, as we can see from the equivalent width map results.

#### 3.6.2 Plots

Plots of physical quantities as functions of radial distance are a good way to view all the assumptions regarding the physical properties of the gas. The number density, temperature, and bulk velocity are plotted against radial distance. The turbulence profile as a Gaussian distribution is also plotted.

Note that the data used for plotting are not extracted from the grid structure,
because in the synthetic model many cells are empty, making it difficult to see the analytical relations to radial distance. As such, an array of radial coordinates is used to feed the functions and provide visually continuous plots.

### 3.6.3 Colormaps

Colormaps are the most straightforward to help visualize the data. Since the line of nodes is on the $x$-axis and the line of sight is in the $z$-direction, we chose the slice of the $yz$-plane to produce colormaps that best show wind structures with respect to the galactic disk. The bulk velocity field is overplotted with color representing magnitude and arrows representing direction of flow.

Column density, equivalent width, and mean bulk velocity, which have been calculated for each spaxel, are also plotted as colormaps. These colormaps involve no slice selection, and represent what would be observed from Earth directly.
Chapter 4

Simulation Results

Graphs are grouped by wind types and velocity dependencies. Results of each simulation run are displayed as a central spaxel plot, physical parameters plots, and colormaps. The resolution of all colormaps is $100 \times 100$. 
4.1 Spherical Winds

4.1.1 \( v_{\text{bulk}} \propto r^0 \)

Figure 4.1: Spherical winds \( v_{\text{bulk}} \propto r^0 \): The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section (yz-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \( \{2^\circ, 30^\circ, 45^\circ, 60, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\} \).
4.1.2 $v_{\text{bulk}} \propto r^{1}$

Figure 4.2: Spherical winds $v_{\text{bulk}} \propto r^{1}$: The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting color maps observed. The velocity field map shows a cross section ($yz$-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are $\{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\}$. 
4.1.3 \( v_{\text{bulk}} \propto r^2 \)

Figure 4.3: Spherical winds \( v_{\text{bulk}} \propto r^2 \): The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section (\(yz\)-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \( \{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\} \).
4.2 Filled Biconical Winds

4.2.1 \( v_{\text{bulk}} \propto r^0 \)

Figure 4.4: Filled biconical winds \( v_{\text{bulk}} \propto r^0 \): The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section \((yz\text{-plane})\) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \(\{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\}\).
4.2.2 \( v_{\text{bulk}} \propto r^1 \)

Figure 4.5: Filled biconical winds \( v_{\text{bulk}} \propto r^1 \): The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section (\( yz \)-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \( \{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\} \).
CHAPTER 4

DUAN 70
4.2.3 $v_{\text{bulk}} \propto r^2$

Figure 4.6: Filled biconical winds $v_{\text{bulk}} \propto r^2$: The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section ($yz$-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are $\{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\}$. 
4.3 Unfilled Biconical Winds

4.3.1 $v_{\text{bulk}} \propto r^0$

Figure 4.7: Unfilled biconical winds $v_{\text{bulk}} \propto r^0$: The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section ($yz$-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are $\{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\}$.
4.3.2 $v_{\text{bulk}} \propto r^1$

Figure 4.8: Unfilled biconical winds $v_{\text{bulk}} \propto r^1$: The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section ($yz$-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \{2°, 30°, 45°, 60°, 90°, 120°, 135°, 150°, 178°\}. 
4.3.3 \( v_{\text{bulk}} \propto r^2 \)

Figure 4.9: Unfilled biconical winds \( v_{\text{bulk}} \propto r^2 \): The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section (yz-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \( \{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\} \).
4.4 Bipolar Bubble Winds

4.4.1 $v_{\text{bulk}} \propto r^0$

Figure 4.10: Bipolar bubble winds $v_{\text{bulk}} \propto r^0$: The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section ($yz$-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \{2°, 30°, 45°, 60°, 90°, 120°, 135°, 150°, 178°\}. 
4.4.2 \( v_{\text{bulk}} \propto r^1 \)

Figure 4.11: Bipolar bubble winds \( v_{\text{bulk}} \propto r^1 \): The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section \((yz\text{-plane})\) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \(\{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\}\).
4.4.3 \( v_{\text{bulk}} \propto r^2 \)

Figure 4.12: Bipolar bubble winds \( v_{\text{bulk}} \propto r^2 \): The top graph shows the spectrum of the central spaxel. The second row is plots of assumed physical parameters, including number density, temperature, bulk velocity, and turbulence profile. The third row is the resulting colormaps observed. The velocity field map shows a cross section (yz-plane) of the space, where color shows the magnitude of the flow, arrows show the direction of the flow, and the red line shows the galactic disk. The inclinations shown in this subsection are \( \{2^\circ, 30^\circ, 45^\circ, 60^\circ, 90^\circ, 120^\circ, 135^\circ, 150^\circ, 178^\circ\} \).
Chapter 5

Discussion

In all the spaxel plots, the absorption line of the lower wavelength (Na I D$_2$) is plotted in blue, and the absorption line of the high wavelength (Na I D$_1$) in red. The simulated absorption strengths of the two absorption lines agree with the fact that the oscillator strength of Na I D$_2$ is two times that of Na I D$_1$. In some cases there is no central spectrum in the spaxel plot, as there is too little absorption due to the wind configuration.

In all colormaps, the variation of bulk velocity dependence is reflected in the velocity field maps, where the color at each location represents the magnitude of the bulk velocity. If the exponent is 0, the velocity is constant in the wind region. If the exponent is 1 or 2, the change in color with radial distance behaves as expected.

Note that the number density distribution plays a major role in affecting the column density and the equivalent width maps. Take the $135^\circ$ inclination results in Figure 4.1 as an example. From the velocity field map, it is clear that $y = 3$ kpc seems to pass through more volume gas than $y = 0$. However, in the column density map next to it, the $(0,3)$ kpc position has a much lower column
density than (0,0). This is because the line of sight at (0,3) kpc does not reach close enough to the galactic center to pass through any grid cells of high number density, whereas the line of sight at (0,0) accumulates very high column density with lower volume due to the high number density close to the galactic center.

The spherical winds results are somewhat less interesting, as the wind shape is the most regular. The boundaries are all smooth curves. The column density and the equivalent width maps change with inclination, and the maximum absorption position is always at the center spaxel. In the mean bulk velocity maps, the small maximum region shifts a little as orientation changes. When the bulk velocity exponent gets to 2 \( (v \propto r^2) \), the mean bulk velocity in the face-on case \((90^\circ\text{inclination})\) in Figure 4.3 peaks in a ring around the center. This is the first hint of a recurring pattern in other wind types. In other cases, the change of mean bulk velocity outward from maximum to minimum is monotonic.

The filled biconical winds result in distinct shapes observed in the colormaps. The boundaries are not smooth any more due to the sharp cones. As the galaxy gets close to edge-on, it shows a circular sector or triangular shape. The maximum absorption region is slightly shifted by changes in inclination. The mean bulk velocity map in the \( v \propto r^2 \) face-on case also shows the ring-shaped maximum. What is interesting is that for all three bulk velocity dependencies, the column density and the equivalent width maps in the face-on cases show a noticeable cross pattern around the center, unlike the in spherical cases where they are uniformly circular. This cross pattern is similar to the diffraction spikes in images of bright stars caused by secondary mirrors in telescopes. This is a numerical artifact related to the finite square grid and does not represent physical appearance of the winds.
The unfilled biconical winds have their distinct shape, too - the pattern in the colormaps looks like a big circular sector minus a smaller one, resulting in a thick arc shape for inclinations close to 0 or \(\pi\). The maximum absorption region is also slightly shifted by changes in inclination. The ring shape of the mean bulk velocity in the edge-on cases is more than obvious now, as the ring grows much larger. The cross pattern is more noticeable at the center in the column density and the equivalent width maps when seen face-on.

The bipolar bubble winds show fairly regular patterns on the colormaps. The maximum absorption region is slightly shifted by changes in inclination. The mean bulk velocity maps still show a clear ring pattern, although now the mean velocity at the center region is much closer to maximum compared to previous ones. The effect of the galactic disk on the result colormaps is the most significant for bipolar bubble winds. In the 30° and 150° cases there is a clear cut on the side of the wind away from the center, which is where the projection of the galactic disk ends.

Finally, the offset of the equivalent width peak appears to be a feature of filled and unfilled biconical winds and bipolar bubble winds. One could distinguish between spherical, filled and unfilled biconical winds, but distinguishing unfilled biconical winds and bipolar bubble winds is more difficulty as their profiles look similar.
Chapter 6

Future Directions

By the end of the project, we have primarily varied the wind shape and bulk velocity dependence, since they are the most significant factors in the formation of absorption-line profiles. But there are many other free parameters supported by the model that can be altered within reasonable ranges. In the future, we will make more runs with different sets of parameters, explore the consequences of variations of other parameters, and see how theoretically they should affect our observation.

What this model mainly lacks is other relevant radiative transfer processes. The model only considers absorption and corrects for stimulated emission, leaving out other emission and scattering processes, which are also important if not as considerable. The model as of now is able to provide reliable qualitative information and good quantitative approximations, but it will be more accurate if radiative transfer processes are comprehensively implemented.

Due to the nature of the physical calculations involved in this astrophysics project, determining wind structure conditions and integrating along line of sight make loops hard to avoid. The program has been optimized to an extent such
that computing time is reasonably acceptable for common grid resolutions, but it surely will be helpful to look for better computation algorithms and structural logics of the program.

We would also like to seek parallel computing solutions. Currently on an average Mac with a single process, each run at a resolution of $50 \times 50$ takes an hour, and at $100 \times 100$ it takes about half a day. In comparison, the results presented in this report were computed entirely within 3 hours by 4 IDL instances running in parallel on a PC with 8 CPU threads overclocked to 4.3 GHz. When single thread speed is limited by hardware, parallel computing can easily multiply the computation efficiency. In the lab this can be achieved by distributing the task over network to other workstations.

Finally, the ultimate goal of numerical simulation studies is of course to understand galactic outflows. This requires a further comparison between synthetic simulations and observational data, which we have yet to work on.
Appendix A

Model Program

A.1 gas_model_v1.pro

```fortran
function number_density1, r
  ; number of absorbing atoms in unit volume. can be a function of l
  number_density = r^0
  return, number_density
end

pro gas_model_v1, VDT, SUPPRESS_GFX=suppress_gfx
  ; simple model for a parcel of gas
  ;
  @fc ; load fundamental constants
  ; define input variables
  ; velocity distribution type. 1 for maxwellian, 2 for gaussian
  ; var for maxwellian dist: m_atomic, T
  ; var for gaussian dist: mu, sigma
  ; VDT = 1
  density_coefficient = 1d
  mu = -1d3 ; mean of gaussian velocity distribution
  sigma = 150d3 ; standard deviation of gaussian velocity distribution
  lambda_min = 5870d-10 ; lower limit of line profile domain
  lambda_max = 5920d-10 ; upper limit of line profile domain
```
delta_lambda = 0.01d-10 ; increment of wavelength

v_min = -1000d3 ; lower limit of velocity distribution
v_max = +1000d3 ; upper limit of velocity distribution
delta_v = 1d3 ; increment of velocity
T = 1d4 ; temperature of the gas
I0 = 1d ; original intensity of light beam
I_lambda0 = I0 ; use a function instead for a radiation profile that depends on wavelength, eg. blackbody
r_min = 0d ; lower limit of line of sight
r_max = 1d-7*pc ; upper limit of line of sight

; for transition 1
f1 = 0.640511d ; oscillator strength
bkbj1 = 1d ; the ratio b_k/b_j is 1 for bound-bound transition
lambda1 = 5889.950954d-10 ; central wavelength for transition 1
m_atomic1 = m_atomic_Na ; atomic mass of atom 1

; for transition 2
f2 = 0.319913d ; oscillator strength
bkbj2 = 1d ; the ratio b_k/b_j is 1 for bound-bound transition
lambda2 = 5895.924237d-10 ; central wavelength for transition 2
m_atomic2 = m_atomic_Na ; atomic mass of atom 2

lambda_index = dindgen(ceil((lambda_max - lambda_min)/delta_lambda))
lambda = [lambda_min + lambda_index*delta_lambda, lambda_max]
lambda_n = size(lambda, /n_elements)

; compute integrated absorption coefficient
s1 = acs_int(f1, bkbj1, lambda1, T)
s2 = acs_int(f2, bkbj2, lambda2, T)

; wavelength distribution

; maxwell distribution

1: begin
  v_index = dindgen(ceil((v_max - v_min)/delta_v))
v = [v_min + v_index*delta_v, v_max]
v_n = size(v, /n_elements)
P_v1 = P_v_maxwell(v, m_atomic1, T)
P_v2 = P_v_maxwell(v, m_atomic2, T)
\[
\phi_1 = P_v2\phi(\lambda, \lambda_1, \text{ATOMICMASS}=m_{\text{atomic}1}, \text{TEMP}=T)
\]
\[
\phi_2 = P_v2\phi(\lambda, \lambda_2, \text{ATOMICMASS}=m_{\text{atomic}2}, \text{TEMP}=T)
\]
\]
; gaussian distribution
2: begin

\[
v\_\text{index} = \text{dindgen}(\text{ceil}((v\_\text{max} - v\_\text{min})/\text{delta}_{v}))
\]
\[
v = [v\_\text{min} + \text{v\_index}*\text{delta}_{v}, v\_\text{max}]
\]
\[
v\_n = \text{size}(v, /\text{n\_elements})
\]
\[
P\_v1 = P\_v\text{gaussian}(v, \mu, \sigma)
\]
\[
P\_v2 = P\_v\text{gaussian}(v, \mu, \sigma)
\]
\[
\phi_1 = P_v2\phi(\lambda, \lambda_1, \text{MEAN} = \mu, \text{SD} = \sigma)
\]
\[
\phi_2 = P_v2\phi(\lambda, \lambda_2, \text{MEAN} = \mu, \text{SD} = \sigma)
\]
endendcase

; restrict range of \lambda to avoid double peak due to
; negative velocities and such
\lambda\_\text{lowerlimit}1 = \text{min}(v2\lambda(\lambda, \lambda_1))
\lambda\_\text{upperlimit}1 = \text{max}(v2\lambda(\lambda, \lambda_1))
\text{print}, 'limits\_of\_\lambda\_\text{as\_defined\_by\_the\_velocity\_distribution' + $'
for\_transition1\_is' + $'
+ '[' + \text{string}(\lambda\_\text{lowerlimit}1) + ',' + $'
\text{string}(\lambda\_\text{upperlimit}1) + ']' + $'
\lambda\_\text{lowerlimit}2 = \text{min}(v2\lambda(\lambda, \lambda_2))
\lambda\_\text{upperlimit}2 = \text{max}(v2\lambda(\lambda, \lambda_2))
\text{print}, 'limits\_of\_\lambda\_\text{as\_defined\_by\_the\_velocity\_distribution' + $'
for\_transition2\_is' + $'
+ '[' + \text{string}(\lambda\_\text{lowerlimit}2) + ',' + $'
\text{string}(\lambda\_\text{upperlimit}2) + ']' + $'
\lambda\_\text{lowerlimit}1\_\text{index} = \text{value\_locate}(\lambda, \lambda\_\text{lowerlimit}1)
\lambda\_\text{upperlimit}1\_\text{index} = \text{value\_locate}(\lambda, \lambda\_\text{upperlimit}1)
\lambda\_\text{lowerlimit}2\_\text{index} = \text{value\_locate}(\lambda, \lambda\_\text{lowerlimit}2)
\lambda\_\text{upperlimit}2\_\text{index} = \text{value\_locate}(\lambda, \lambda\_\text{upperlimit}2)
\text{if} \lambda\_\text{lowerlimit}1 \text{lt} \lambda\_\text{min} \text{then begin}
\text{print}, 'decrease\_\lambda\_\text{min\_to\_display\_all\_data\_\_\_execution\_halted'
\text{stop}; \lambda\_\text{lowerlimit}1\_outside\_the\_\lambda\_interval\_to\_be\_\text{plotted'
\text{endif}
\text{if} \lambda\_\text{upperlimit}1 \text{gt} \lambda\_\text{max} \text{then begin
print, 'increase lambda_max to display all data, execution halted'
stop ; lambda_upperlimit on the right of the interval
endif
if lambda_lowerlimit2 lt lambda_min then begin
print, 'decrease lambda_min to display all data, execution halted'
stop ; lambda_lowerlimit outside the lambda interval to be plotted
endif
if lambda_upperlimit2 gt lambda_max then begin
print, 'increase lambda_max to display all data, execution halted'
stop ; lambda_upperlimit on the right of the interval
endif
phi1(0:lambda_lowerlimit1_index)=0 ; null meaningless data outside
; what's defined by the velocity distribution
phi1(lambda_upperlimit1_index: size(phi1, /n_elements)-1)=0
phi2(0:lambda_lowerlimit2_index)=0 ; null meaningless data outside
; what's defined by the velocity distribution
phi2(lambda_upperlimit2_index: size(phi2, /n_elements)-1)=0

; absorption cross section for a specific wavelength, lambda
s1_lambda = s1*phi1
s2_lambda = s2*phi2
s0_lambda = s1_lambda + s2_lambda
; column density
column_density = density_coefficient*qsimp('number_density', $ r_min, r_max, /double)
; optical depth
tau1_lambda = s1_lambda*column_density ; optical depth for a specific wavelength, lambda
tau2_lambda = s2_lambda*column_density
tau0_lambda = s0_lambda*column_density
; resulting intensity at a specific wavelength
I1_lambda = I_lambda0*exp(-tau1_lambda)
I2_lambda = I_lambda0*exp(-tau2_lambda)
I0_lambda = I_lambda0*exp(-tau0_lambda)

; equivalent width
equivalent_width1 = discrete_area(lambda, 1-I1_lambda/I_lambda0)
equivalent_width2 = discrete_area(lambda, 1-I2_lambda/I_lambda0)
equivalent_width0 = discrete_area(lambda, 1-I0_lambda/I_lambda0)

; plots
if keyword_set(suppress_gfx) then begin
endif else begin
w=0 ; window index
cgplot, v, P_v1, background='white',color='red',axiscolor='black'
; for transitions of two types of atoms with
; different veolocity distributions
; cgplot, v, P_v2, background='white', color='red', axiscolor='black'
window, w++ ; optical depth
cgplot, lambda, phi1, background='white',color='red',axiscolor='black'
window, w++ ; absorption line profile for transition1
cgplot, lambda, I1_lambda/I_lambda0, $
window, w++ ; absorption line profile for transition2
cgplot, lambda, I2_lambda/I_lambda0, $
window, w++ ; absorption line profile for doublet
cgplot, lambda, I0_lambda/I_lambda0, $
endelse
; save cache; debug
save, /variables, filename='var.sav'
stop
end

A.2  gas_model_v2.pro

function v_bulk, r, coefficient, exp
; bulk velocity of gas as a function of radius
; $v \sim r^2$
v_bulk = -coefficient*r^(exp)
return, v_bulk
end

function model2_T, r
    ; temperature as a function of radius
    T = 1d4*(r/(1d3*pc))^(1d)
    return, T
end

pro gas_model_v2, SUPPRESS_GFX=suppress gfx
    ; load fundamental constants
    common ndshare, number_density2_exp, number_density2_coefficient
    ; define input variables
    ; var for maxwellian dist: m_atomic, T
    ; var for guassian dist: mu, sigma
    VDT = 1d               ; velocity distribution type
    r_min = 10*pc          ; lower limit of radial distance
    r_max = 10d3*pc        ; upper limit of radial distance
    v_max = 1d6
    delta_r = 100*pc       ; increment of radial distance
    number_density2_exp = -1 ; [-3, 0]
    number_density2_coefficient = 1d6
    v_bulk_exp = 1          ; [0, 3]
    v_bulk_coefficient = v_max/(r_max)^v_bulk_exp ; automatically set
    ; according to contraint: v_max = 1000 km/s at r_max
    lambda_min = 5750d-10   ; lower limit of line profile domain
    lambda_max = 5950d-10   ; upper limit of line profile domain
    delta_lambda = 0.20d10  ; increment of wavelength
    v_thermal_min = -10d3   ; lower limit of velocity distribution
    v_thermal_max = +10d3   ; upper limit of velocity distrubition
    delta_v_thermal = 0.01d3 ; increment of velocity
    I0 = 1d                 ; original intensity of light beam
    I_lambda0 = I0          ; use a function instead for a radiation
    ; profile that depends on wavelength.
    ; eg. blackbody
    f1 = 0.640511d          ; oscillator strength


APPENDIX A

bkbj1 = 1d ; the ratio b_k/b_j is 1 for bound–bound transition
lambda1 = 5889.950954d–10 ; central wavelength for transition 1
m_atomic1 = m_atomic_Na ; atomic mass of atom 1

; for transition 2

f2 = 0.319913d ; oscillator strength
bkbj2 = 1d ; the ratio b_k/b_j is 1 for bound–bound transition
lambda2 = 5895.924237d–10 ; central wavelength for transition 2
m_atomic2 = m_atomic_Na ; atomic mass of atom 2

lambda_index = dindgen( ceil ((lambda_max - lambda_min)/delta_lambda))
lambda = [lambda_min + lambda_index*delta_lambda, lambda_max]

; set initial values for optical depth

tau1_lambda = 0
tau2_lambda = 0
tau0_lambda = 0

; divide the line of sight into infinitesimal segments

for r=r_min, r_max-delta_r, delta_r do begin

sigma = 150d3 ; standard deviation of gaussian velocity distribution
mu = -1d3 ; mean of gaussian velocity distribution
T = 1d4 ; temperature of the gas

; to be used for calculating acs_int & p_v_maxwell
l_min = 0d ; lower limit of line of sight
l_max = 1d–7*pc ; upper limit of line of sight I0 = 1d

density_coefficient = 1d

; velocity and wavelength distribution
r_mean = r + delta_r/2d
v_bulk = v_bulk(r_mean, v.bulk_coefficient, v.bulk_exp)

T = model2_T(r_mean)

; compute integrated absorption coefficient
s1 = acs_int(f1, bkbj1, lambda1, T)
s2 = acs_int(f2, bkbj2, lambda2, T)

case VDT of

1: begin

v_thermal_index = dindgen(ceil((v_thermal_max - v_thermal_min)/$
delta_v_thermal))

v_thermal = [v_thermal_min + v_thermal_index*delta_v_thermal, $
  v_thermal_max]

P_v1 = P_v_maxwell(v_thermal, m_atomic1, T)

P_v2 = P_v_maxwell(v_thermal, m_atomic2, T)

phi1 = P_v2phi(v2lambda(lambda2v(lambda, lambda1) - v_bulk, $
  lambda1), $

  lambda1, ATOMICMASS=m_atomic1, TEMP=T)

phi2 = P_v2phi(v2lambda(lambda2v(lambda, lambda2) - v_bulk, $
  lambda2), $

  lambda2, ATOMICMASS=m_atomic2, TEMP=T)

end

; gaussian distribution

2: begin

v_thermal_index = dindgen( ceil ((v_thermal_max
  - v_thermal_min)/$
    delta_v_thermal))

v_thermal = [v_thermal_min + v_thermal_index*delta_v_thermal, $
  v_thermal_max]

P_v1 = P_v_gaussian(v_thermal, mu, sigma)

P_v2 = P_v_gaussian(v_thermal, mu, sigma)

phi1 = P_v2phi(lambda, lambda1, MEAN=mu, SD=sigma)

phi2 = P_v2phi(lambda, lambda2, MEAN=mu, SD=sigma)

end

endcase

v = v_add(v_thermal, v_bulk)

v_n = size(v, /n_elements)

; restrict range of lambda to avoid double peak due to
; negative velocities and such

lambda_lowerlimit1 = min(v2lambda(v,lambda1))

lambda_upperlimit1 = max(v2lambda(v,lambda1))

print, 'limits_of_lambda_as_defined_by_the_v_thermal+v_bulk'+$
  'for_transition1$is' $
  + '][' + string(lambda_lowerlimit1) + '],' + $
    string( lambda_upperlimit1) + ']'$

lambda_lowerlimit2 = min(v2lambda(v,lambda2))

lambda_upperlimit2 = max(v2lambda(v,lambda2))

print, 'limits_of_lambda_as_defined_by_the_v_thermal+v_bulk'+$
120 'for transition2', is $''$
121 + '[' + string(lambda_lowerlimit2) + ' , ' + 
122 string(lambda_upperlimit2) + ']
123 lambda_lowerlimit1_index = value_locate(lambda, lambda_lowerlimit1)
124 lambda_upperlimit1_index = value_locate(lambda, lambda_upperlimit1)
125 lambda_lowerlimit2_index = value_locate(lambda, lambda_lowerlimit2)
126 lambda_upperlimit2_index = value_locate(lambda, lambda_upperlimit2)
127 if lambda_lowerlimit1 lt lambda_min then begin
128 print, 'decrease lambda_min to display all data execution halted'
129 stop ; lambda_lowerlimit outside the lambda interval to be plotted
129 endif
130 if lambda_upperlimit1 gt lambda_max then begin
131 print, 'increase lambda_max to display all data execution halted'
132 stop ; lambda_upperlimit on the right of the interval
132 endif
133 if lambda_lowerlimit2 lt lambda_min then begin
134 print, 'decrease lambda_min to display all data execution halted'
135 stop ; lambda_lowerlimit outside the lambda interval to be plotted
135 endif
136 if lambda_upperlimit2 gt lambda_max then begin
137 print, 'increase lambda_max to display all data execution halted'
138 stop ; lambda_upperlimit on the right of the interval
138 endif
139 phi1(0:lambda_lowerlimit1_index)=0 ; null meaningless data outside
140 ; what's defined by the velocity
141 ; distribution
142 phi1(lambda_upperlimit1_index: size(phi1, /n_elements)-1)=0
143 phi2(0:lambda_lowerlimit2_index)=0 ; null meaningless data outside
144 ; what's defined by the velocity
145 ; distribution
146 phi2(lambda_upperlimit2_index: size(phi2, /n_elements)-1)=0
147 ; absorption cross section for a specific wavelength, lambda
148 s1_lambda = s1*phi1
149 s2_lambda = s2*phi2
150 s0_lambda = s1_lambda + s2_lambda
151 ; column density
152 column_density = qsimp('number_density2', r, r+delta_r, /double, eps=1d3)
; optical depth
tau1_lambda = tau1_lambda + s1_lambda*column_density ; optical depth
; for a specific wavelength
 tau2_lambda = tau2_lambda + s2_lambda*column_density
 tau0_lambda = tau0_lambda + s0_lambda*column_density
endfor

; resulting intensity at a specific wavelength
I1_lambda = I_lambda0*exp(-tau1_lambda)
I2_lambda = I_lambda0*exp(-tau2_lambda)
I0_lambda = I_lambda0*exp(-tau0_lambda)

; equivalent width
equivalent_width1 = discrete_area(lambda, 1-I1_lambda/I_lambda0)
equivalent_width2 = discrete_area(lambda, 1-I2_lambda/I_lambda0)
equivalent_width0 = discrete_area(lambda, 1-I0_lambda/I_lambda0)

; plots
r_index = dindgen(ceil((r_max-r_min)/delta_r))
r_array = [r_min + r_index*delta_r, r_max]
r_n = size(r, /n_elements)
if keyword_set(suppress_gfx) then begin
endif else begin

; w=0 ; window index
; window, w+ ; velocity distribution of Na
ps_start, file='\users\duan\Documents\research\model2/n^'+STRTRIM(number_density2_exp, 2)+'v^'+STRTRIM(v_bulk_exp,2)'+.eps', $
/encap
!p.multi=[0, 3, 2, 0, 0]
margin = [2, 2, 1, 0]
; graph 1
cgplot, r_array/pc, number_density2(r_array)/1d6, $
  background='white',color='red',axiscolor='black',$
  title = 'Number Density of Na', $
  xtitle = 'Radius/pc', ytitle = 'Number density/\text{cm}^{-3}'+textoidl('cm\raise-.5ex\text{\^{}}-3')$
; graph 2
cgplot, r_array/pc, 
  v_bulk(r_array, v__bulk_coefficient, v_bulk_exp)/1d3, $
  background='white', color='red', axiscolor='black', $
APPENDIX A

```
title = 'Bulk Velocity', $
xtile = 'Radius/pc',$
ytitle = 'Velocity/\text{km/s}',$

; graph 3
cgplot, r_array/pc, model2_T(r_array),$
  background='white'
color='red'
axiscolor='black'
title = 'Temperature,$
xtitle = 'Radius/pc',
ytitle = 'Temperature/K'

; graph 4

cgplot, v_thermal/1d3, P_v_maxwell(v_thermal, m_atomic1, mean(T)),$
  background='white'
color='red'
axiscolor='black'
title = 'Maxwellian Distribution (Thermal),$ xtitle = 'Velocity/\text{km/s}',
ytitle = 'Probability density'$

; graph 5

cgplot, lambda*1d10, tau1_lambda, /ylog,$
  background='white'
color='red'
axiscolor='black'
charthick = 1, thick = 2,
title = 'Optical depth,$
xtitle = 'Wavelength/\text{Angstrom}',
ytitle = 'Optical depth,$
xr = [5850d, 5925d]$

; graph 6

cgplot, lambda*1d10, I1_lambda/I_lambda0,$
  background='white'
color='red'
axiscolor='black'
charthick = 1, thick = 2,
title = 'Relative intensity,$
xtitle = 'Wavelength/\text{Angstrom}',
ytitle = 'Relative intensity,$
xr = [5850d, 5925d]$
```

A.3 gas_model_v3.pro

; for best view set tab size to 4 characters. in vim run :set tabstop=4
pro gas_model_v3, windtype_input, inclination_input, SUPPRESS_GFX=suppress_gfx
@fc ; load fundamental constants
common shared_parameters, number_density_range, $
number_density_coefficient, number_density_exp, $
temp_coefficient, temp_exp, temp0, $
v_bulk_coefficient, v_bulk_exp, v_bulk0
; share parameters across several functions
; DO NOT CHANGE as order of these variables MATTERS as IDL is dumb
; define input variables
; variables for MAXWELLIAN distribution : m_atomic, T
; variables for gaussian distribution : mu, sigma
; Geometry assumptions: observing along z direction
; major axis of the ellipse observed always
; is the x-axis
APPENDIX A

DUAN 123

16;      galactic center is at (0, 0) on x-y plane
17;  --------------------- model options ---------------------
18;  spot = [0, 0] ; location of the pixel observed on x-y plane
19  windtype = windtype_input ; 1, spherical
20  ; 2, filled biconical
21  ; 3, unfilled biconical
22;  windtype = 2d
23;  grid_radius = 10d ; number of pixels along radius
24;  ; half length of each dimension
25  vdt = 2 ; velocity distribution type:  1 for maxwellian
26;  2 for gaussian
27  distance_offset = 1d ; distance from galactic plane. for plotting purpose
28  bicone_angle = !pi/6d ; angle between cone and galactic plane (theta = pi/2)
29  bicone_thickness = !pi/13d ; for windtype 3
30;  ---------------------- observables ----------------------
31;  r_proj = 0.5*r_galaxy ; projected radius. measured vertically
32  inclination = inclination_input ; we can measure original & projected
33;  ; area or radius
34;  inclination = !pi*3/4 ; inclination angle between galactic
35;  ; disc and z+ axis. range [0, pi)
36;  used !pi/2 (vertical) and !pi*3/4 (45 degrees) for senior presentation
37  r_galaxy = 10d3*pc ; actual radius of the galaxy. measured horizontally
38  r_wind = r_galaxy*0.7 ; radius of the gas
39  center_perturb = r_galaxy/grid_radius/2 ; amount of perturbation for center
40;  ; location to calculate quantities
41  v_hubble = 200d3 ; velocity due to hubble expansion
42;  ; typically 200km/s (M82)
43;  ----------------------- dependences -----------------------
44  sigma = 100d3 ; standard deviation for gaussian velocity distribution
45  mu = 0d ; mean for gaussian
46  number_density_range = [3d-7, 1d-8] ; from center to outer boundary
47  number_density_exp = -2 ; [-3, 0]
48  temp_exp = -1 ; [-3, 0]
49  temp_range = [1d4,1d3] ; for maxwell velocity distribution
50  v_bulk_exp = 1 ; [ 0, 3]
51  v_bulk_range = [50d3,500d3]
52;  ------------------- plotting settings ---------------------
53;  without considering velocity due to hubble expansion
\begin{verbatim}
lambda_min = 5800d-10 ; lower limit of line profile domain in the
plot and all calculations
lambda_max = 5950d-10 ; upper limit of line profile domain
delta_lambda = 0.10d ; increment of wavelength
gaussian_max = 800d3 ; upper limit of gaussian velocity
10 = 1d ; original intensity of light beam
I_lambda0 = 10 ; use a function insted for a radiation
profile that depends on wavelength.
; eg. blackbody
Na I D1 transition parameters
f1 = 0.319913d ; oscillator strength
bkbj1 = 1d ; b_k/b_j is 1 for bound–bound transition
lambda1 = 5897.5661715d-10 ; central wavelength for transition 1,
5890A in our case
m_atomic1 = m_atomic_Na ; atomic mass of atom 1

Na I D2 transition parameters
f2 = 0.640511d ; oscillator strength
bkbj2 = 1d ; b_k/b_j is 1 for bound–bound transition
lambda2 = 5891.58326415d-10 ; central wavelength for transition 2,
5896A in our case
m_atomic2 = m_atomic_Na ; atomic mass of atom 2

; set coefficients of dependence relations
number_density_coefficient = 1/r_galaxy* 
( 
    number_density_range(1) 
    / 
    number_density_range(0) 
) ^ (1/number_density_exp)
temp_coefficient = (temp_range(1)-temp_range(0)) 
/ 
( r_galaxy^temp_exp – center_perturb^temp_exp)
v_bulk_coefficient = (v_bulk_range(1)-v_bulk_range(0)) 
/ 
( r_galaxy^v_bulk_exp – center_perturb^v_bulk_exp)
\end{verbatim}
temp0 = temp_range(1) - temp_coefficient * r_galaxy^temp_exp
v_bulk0 = v_bulk_range(1) - v_bulk_coefficient * r_galaxy^v_bulk_exp

; construct lattice

; basic stuff

side = 2*grid_radius+1 ; total number of pixels along each dimension
xhat = [1, 0, 0] ; unit vectors in cartesian space
yhat = [0, 1, 0]
zhat = [0, 0, 1]

normal = [0, sin(inclination-!pi/2), cos(inclination-!pi/2)] ; normal vector

plane_co1 = normal(0) ; coefficients to specify galactic plane
plane_co2 = normal(1)
plane_co3 = normal(2)

allocate memory

occupy = dblarr(side, side, side) ; just an empty template

grid = {
  x: occupy, $ ; index of x coordinate
  y: occupy, $ ; index of y coordinate
  z: occupy, $ ; index of z coordinate
  r: occupy, $ ; radial coordinate
  theta: occupy, $ ; polar angle with respect to
          $ ; the normal direction
          $ ; which is the polar axis
  number_density: occupy, $ 
  temp: occupy, $ 
  v_bulk: occupy, $ 
  v_bulk_x: occupy, $ 
  v_bulk_y: occupy, $ 
  v_bulk_z: occupy, $ 
  plane: occupy $ ; as logical matrix, value is 1
          $ ; if the plane lies in the block
}

; loop for each cell in lattice

; assign index number with respect to central origin, (i−grid_radius), to
; every slice
for \( i = 0, \text{side} - 1 \) do begin

\[
\text{occupy} = \text{replicate}(i - \text{grid}\_\text{radius}, \text{side}, \text{side})
\]

\[
\text{grid}\.\text{x}(i,*,*) = \text{occupy}
\]

\[
\text{grid}\.\text{y}(*,i,*) = \text{occupy}
\]

\[
\text{grid}\.\text{z}(*,*,i) = \text{occupy}
\]

endfor

for \( i = 0, \text{side} - 1 \) do begin

for \( j = 0, \text{side} - 1 \) do begin

for \( k = 0, \text{side} - 1 \) do begin

\[
\text{position} = [ \ 
\text{grid}\.\text{x}(i,j,k), \ 
\text{grid}\.\text{y}(i,j,k), \ 
\text{grid}\.\text{z}(i,j,k) \ ]
\]

; dimensionless index numbers

; in the cartesian grid

\[
\text{grid}\.\text{r}(i,j,k) = \text{norm(position)}*\text{r}\_\text{galaxy}/\text{grid}\_\text{radius}
\]

; radial distance

\[
\text{grid}\.\text{theta}(i,j,k) = \text{acos} \left( \frac{\text{total(position)*normal}}{\text{norm(position)*norm(normal)}} \right) ; \text{angle between two vectors}
\]

\[
\text{position}\_\text{spherical} = [\text{grid}\.\text{r}(i,j,k), \text{grid}\.\text{theta}(i,j,k)]
\]

; assume azimuthally symmetric wind

; outside the wind

if \( \text{grid}\.\text{r}(i,j,k) \gt \text{r}\_\text{wind} \) then begin

\[
\text{grid}\.\text{number}\_\text{density}(i,j,k) = 0d \quad \text{set density to 0}
\]

; and we are done

; ------- determine if block is a plane block -------

; los_logic = grid\.theta(i,*) \gt \pi/2

; plane_index = SEARCH\_ARRAY(los_logic, 0)

; plane_index = plane_index(0)

; grid\.plane(i,j,plane_index) = 1

\[
\text{distance} = \text{abs} \left( \ 
\text{plane}\.\text{co1}*\text{grid}\.\text{x}(i,j,k)+ \ 
\text{plane}\.\text{co2}*\text{grid}\.\text{y}(i,j,k)+ \ 
\text{plane}\.\text{co3}*\text{grid}\.\text{z}(i,j,k) \ 
\right)
\]

/ sqrt(\text{plane}\.\text{co1}^2+\text{plane}\.\text{co2}^2+\text{plane}\.\text{co3}^2)
; distance to the galactic disk
if distance le distance_offset then begin
  grid.plane(i,j,k)=1
endif $
else begin
  grid.plane(i,j,k)=0
endelse
endif $

; ------------ inside the wind ------------
else begin
  ; ------- determine if block is a plane block -------
  ; los_logic = grid.theta(i,j,*) gt !pi/2
  ; plane_index = SEARCH_ARRAY(los_logic, 0)
  ; plane_index = plane_index(0)
  ; grid.plane(i,j,plane_index) = 1
  distance = abs ( plane_co1*grid.x(i,j,k)+
                  plane_co2*grid.y(i,j,k)+
                  plane_co3*grid.z(i,j,k) )
                 / sqrt(plane_co1^2+plane_co2^2+plane_co3^2)
  ; distance to the galactic disk
if distance le distance_offset then begin
  grid.plane(i,j,k)=1
endif $
else begin
  grid.plane(i,j,k)=0
endelse

; -------- fill in physical parameters --------
case windtype of
1: begin ; spherial wind
  grid.number_density(i,j,k) = $ number_density_spherical(position_spherical)
  grid.temp(i,j,k) = $ temp_spherical(position_spherical)
  v_bulk = position / norm(position) * $ abs ( v_bulk_spherical ) $
position_spherical $
) $
)

; the function v_bulk_spherical
; returns velocities from given
; spherical coordinates (r, theta)
; in cartesian vector form

grid.v_bulk(i,j,k) = norm(v.bulk)
grid.v_bulk_x(i,j,k) = v.bulk(0)
grid.v_bulk_y(i,j,k) = v.bulk(1)
grid.v_bulk_z(i,j,k) = v.bulk(2)

end

2: begin ; filled biconical wind
if abs(grid.theta(i,j,k)-pi/2) lt bicone_angle $
then begin
grid.number_density(i,j,k)=0d
endif $
else begin
grid.number_density(i,j,k) = $ number_density_spherical ( $ position_spherical $
)
grid.temp(i,j,k) = temp_spherical ( $ position_spherical $ )

v.bulk = position / norm(position) * $ abs ( $
    v.bulk_spherical ( $ position_spherical $ ) $ )

; the function v_bulk_spherical
; returns velocities from given
; spherical coordinates (r, theta)
; in cartesian vector form

grid.v.bulk(i,j,k) = norm(v.bulk)
grid.v.bulk_x(i,j,k) = v.bulk(0)
grid.v.bulk_y(i,j,k) = v.bulk(1)
grid.v_bulk_x(i,j,k) = v_bulk(2)
endelse
end

3: begin ; unfilled biconical wind

if $(abs(grid.theta(i,j,k)−bicone_angle) le bicone_thickness)$
else $(abs(grid.theta(i,j,k)−(pi−bicone_angle)) le bicone_thickness)$
then begin ; on either side of galactic disk
grid.number_density(i,j,k) = number_density_spherical(position_spherical)
grid.temp(i,j,k) = temp_spherical(position_spherical)
v_bulk = position/norm(position)*abs(v_bulk_spherical(position_spherical))
; the function v_bulk_spherical
; returns velocities from given
; spherical coordinates (r, theta)
; in cartesian vector form
; velocities are relative to galaxy
grid.v_bulk(i,j,k) = norm(vBulk)
grid.v_bulk_x(i,j,k) = v_bulk(0)
grid.v_bulk_y(i,j,k) = v_bulk(1)
grid.v_bulk_z(i,j,k) = v_bulk(2)
endif $

else begin
grid.number_density(i,j,k)=0d
endelse
end ; finish windtype 3
endcase ; finish windtype cases
endelse ; finish inside the wind condition
endfor
endfor
print, 'finished constructing plane x = ', + string(i)
endfor

; ----------------- fill galactic center by perturbation -----------------
; as some functions yield null values when r=0 or theta=0
position = [0,center_perturb,0] ; index for shifted core position
grid.r(grid_radius,grid_radius,grid_radius) = center_perturb
grid.theta(grid_radius,grid_radius,grid_radius) = $
acos (  
  total(position*normal)  $ ; dot product
  / $  
  (norm(position)*norm(normal))  $ ; product of norms 
)
grid.plane(grid_radius,grid_radius,grid_radius) = 1
position_spherical = [center_perturb, 1d3]
grid.number_density(grid_radius,grid_radius,grid_radius) = $  
  number_density_spherical(position_spherical) 
grid.temp(grid_radius,grid_radius,grid_radius) = $  
  temp_spherical(position_spherical) 
v_bulk = position/norm(position)*abs(v_bulk_spherical(position_spherical))
grid.v_bulk(grid_radius,grid_radius,grid_radius) = norm(v_bulk)
grid.v_bulk_x(grid_radius,grid_radius,grid_radius) = v_bulk(0)
grid.v_bulk_y(grid_radius,grid_radius,grid_radius) = v_bulk(1)
grid.v_bulk_z(grid_radius,grid_radius,grid_radius) = v_bulk(2)

; ------------------------ calculate spaxels ------------------------
; create wavelength array
lambda_index = dindgen( ceil ((lambda_max - lambda_min)/delta_lambda))
lambda = [lambda_min + lambda_index*delta_lambda, lambda_max]
lambda_n = size(lambda, /n_elements)
; first two dimensions for pixel position, third dimension for transition
; selection, forth dimension for storing spectrum
spaxel = dblarr(side,side,3,lambda_n) ; spaxel
tau = dblarr(side,side,3,lambda_n) ; optical depth
column_density = dblarr(side,side)
equivalent_width = dblarr(side,side,3)
v_gaussian_index = dindgen(ceil((2*v_gaussian_max/delta_v_gaussian)))
v_gaussian = [ $
v_gaussian_max + v_gaussian_index*delta_v_gaussian,

v_gaussian_max

; gaussian spread

; integrate along line of sight

for i = 0, side−1 do begin
  for j = 0, side−1 do begin
    ; loop over all spaxels
    ; i = spot(0)+grid_radius ; for a specific spaxel
    ; j = spot(1)+grid_radius
    ; if line of sight passes through galactic disk plane=1 for some cells
    ; the sum of plane is greater than 0
    if total(grid.plane(i,j,*)) gt 0 then begin
      for k = side−1, 0, −1 do begin
        ; integrate along −z direction
        ; stop when hitting galactic plane
        if grid.plane(i,j,k) eq 1 then break
        s1 = acs_int(f1, bkbj1, lambda1, grid.temp(i,j,k))
        s2 = acs_int(f2, bkbj2, lambda2, grid.temp(i,j,k))
        ; determine wavelengths of interest
        ; info_lambda1_min = min(v2lambda(v,lambda1))
        ; info_lambda2_min = min(v2lambda(v,lambda2))
        ; info_lambda1_max = max(v2lambda(v,lambda1))
        ; info_lambda2_max = max(v2lambda(v,lambda2))
        ; velocity and wavelength distribution
        ; array of absolute velocities relative to observer
        v = v_add ( grid.v_bulk_z(i,j,k),
                    v_add(v_gaussian, v_hubble) )
        ; v_n = size(v, /n_elements)
        ; target wavelength space to velocity space
        ; convert to the grid cell rest frame
        ; for the first transition
        v_rest1 = v_add ( lambda2v(lambda, lambda1),
                         v_add ( grid.v_bulk_z(i,j,k),
                                −v_hubble ) )
\text{for the second transition}
\begin{verbatim}
358 )
359
360 v_rest2 = v_add ( 
361 lambda2v(lambda, lambda2), $ 
362 v_add ( 
363 grid.v_bulk_z(i, j, k), $ 
364 -v_hubble $ 
365 ) $ 
366 )
367 case vdt of
368 1: begin ; maxwell distribution
369 T = grid.temp(i, j, k)
370 phi1 = 
371 P_v2phi
372 ( 
373 v2lambda(v_rest1, lambda1), $ 
374 lambda1, $ 
375 ATOMICMASS=m_atomic1, $ 
376 TEMP=T $ 
377 )
378 phi2 = 
379 P_v2phi
380 ( 
381 v2lambda(v_rest2, lambda2), $ 
382 lambda2, $ 
383 ATOMICMASS=m_atomic2, $ 
384 TEMP=T $ 
385 )
386 end
387 2: begin ; gaussian distribution
388 phi1 = P_v2phi ( 
389 v2lambda ( 
390 v_rest1, lambda1 $ 
391 ), $ 
392 lambda1, $ 
393 MEAN=mu, $ 
394 SD=sigma $ 
395 ) $
) ; assume same gaussian for all locations

\[
\phi_2 = P_{v2\phi} ( \phi_2 \lambda ( v_{\text{rest}2}, \lambda_2 ) , \lambda_1, \text{MEAN} = \mu, \text{SD} = \sigma )
\]

end
endcase

; -------- absorption cross section for each specific wavelength --------

\[
s_1_\lambda = s_1 \phi_1 \\
\]
\[
s_2_\lambda = s_2 \phi_2 \\
\]
\[
s_0_\lambda = s_1_\lambda + s_2_\lambda
\]

; column density
\[
column\_density\_block = \text{grid\_number\_density}(i,j,k) \times r_{\text{galaxy}}/\text{grid\_radius}
\]

; optical depth
\[
\tau(i,j,0,*) = \tau(i,j,0,*) + s_0_\lambda \times column\_density\_block \\
\tau(i,j,1,*) = \tau(i,j,1,*) + s_1_\lambda \times column\_density\_block \\
\tau(i,j,2,*) = \tau(i,j,2,*) + s_2_\lambda \times column\_density\_block \\
column\_density(i,j) = column\_density(i,j) + column\_density\_block
\]
endfor
endif

; ----------------- resulting intensity -----------------

\[
\text{spaxel}(i,j,0,*) = I_{\lambda 0} \times \exp(-\tau(i,j,0,*)) \\
\text{spaxel}(i,j,1,*) = I_{\lambda 0} \times \exp(-\tau(i,j,1,*)) \\
\text{spaxel}(i,j,2,*) = I_{\lambda 0} \times \exp(-\tau(i,j,2,*))
\]

; ------------------ equivalent width -----------------

\[
\text{equivalent\_width}(i,j,0) = \text{discrete\_area}\times (\lambda, 1-\text{spaxel}(i,j,0,*)/I_{\lambda 0}) \\
\text{equivalent\_width}(i,j,1) = \text{discrete\_area}\times (\lambda, 1-\text{spaxel}(i,j,1,*)/I_{\lambda 0}) \\
\text{equivalent\_width}(i,j,2) = \text{discrete\_area}\times (\lambda, 1-\text{spaxel}(i,j,2,*)/I_{\lambda 0})
\]
endfor
print, 'finished integrating over spaxal column x=', + string(i)
endfor

; save cache; debug
save, /all, /comm, /variables, $
   file='/users/duan/Documents/research/model3/variables_wt'+'$
   strtrim(floor(windtype),2)+'_res'+'$
   strtrim(floor(grid_radius)*2,2)+'_incl'+'$
   strtrim(inclination,2)+'$
   '.sav'

; ========================== plots =========================

if keyword_set(suppress_gfx) then begin
endif else begin

; ------------------------ spaxel_display ------------------------

ps_start, file='/users/duan/documents/research/model3/spaxel_wt'$
   +strtrim(floor(windtype),2)+'_res' '$
   +strtrim(floor(grid_radius)*2,2)+'_incl' $'
   +strtrim(inclination,2) $'
   '.eps', /encap, /nomatch
x = grid_radius+1 ; set center coordinates
y = grid_radius+1
spec0 = spaxel(x,y,0,*)
spec1 = spaxel(x,y,1,*)
spec2 = spaxel(x,y,2,*)
margin = [2, 2, 2, 2]
cgplot, lambda*1d10, spec0/I_lambda0, $
   background='white', color='black', axiscolor='black', $
   xtitle = 'Wavelength/\AA' +cgsymbol('Angstrom'), $
   ytitle = 'Relative Flux' $
   , xr = [5870d, 5930d], xsty=1, charthick = 1, thick = 2
cgplot, lambda*1d10, spec1/I_lambda0, $
   background='white', color='red', axiscolor='black', $
   charthick = 1, thick = 2
cgplot, lambda*1d10, spec2/I_lambda0, $
   background='white', color='blue', axiscolor='black', $
   charthick = 1, thick = 2
charthick = 1, thick = 2

ps_end

; colormaps
ps_start, file='/users/duan/documents/research/model3/colormap_wt^' $ +strtrim(floor(windtype),2)+'_res' $ +strtrim(floor(grid_radius)*2,2)+'_incl' $ +strtrim(inclination,2) $ +'.eps', /encap

!p.multi=[0, 2, 3, 0, 0]
margin = [2, 2, 1, 0]

; "n-r$ W_eq$v_mean"
; "v-r$ CD$ P.O."
; -----------------------------------------
; 1st graph v-r
;
;
;
;
; -----------------------------------------
; 1st graph equivalent width
;
; Set up variables for the contour plot. Normally, these values
; would be passed into the program as positional and keyword parameters.
image = equivalent_width(*,*,0)*1D9 ; convert m to mm
nLevels = 10
position = [0.125, 0.125, 0.9, 0.800]
cbposition = [0.125, 0.865, 0.9, 0.895]

; Set up a "window" for the plot. The PostScript output will have
; the same aspect ratio as the graphics window on the display.
; cgDisplay, 600, 600, Title='Image Plot with Contours'

; Set up colors for contour plot.
cgLoadCT, 33

cgImage, image, Stretch=1, MinValue=MIN(IMAGE), MaxValue=MAX(IMAGE), $ /Axes, XTitle='X, Distance in kpc', YTitle='Y, Distance in kpc', $
Position=position, $
XRange=[-r\_galaxy/pc/1d3, r\_galaxy/pc/1d3], $
YRange=[-r\_galaxy/pc/1d3, r\_galaxy/pc/1d3], $
/Keep\_Aspect, title='Equivalent Width/\text{nm}'

; Draw the color bar. Fit it to the location of the image.
cgColorbar, Position=cbposition, Range=[min(image), max(image)], /Fit

; ______________________ 2nd graph column density ______________________

; Set up variables for the contour plot. Normally, these values
; would be passed into the program as positional and keyword parameters.
image = column\_density/1d14
nLevels = 10
position = [0.125, 0.125, 0.9, 0.800]
cbposition = [0.125, 0.865, 0.9, 0.895]

; Set up a "window" for the plot. The PostScript output will have
; the same aspect ratio as the graphics window on the display.
cgDisplay, 600, 600, Title='Image Plot with Contours'

; Set up colors for contour plot.
cgLoadCT, 33

cgimage, image, stretch=1, minvalue=min(image), maxvalue=max(image), $
/axes, xtitle='X Distance/\text{kpc}', ytitle='Y Distance/\text{kpc}',$
position=position, $
xrange=[-r\_galaxy/pc/1d3, r\_galaxy/pc/1d3],$
yrange=[-r\_galaxy/pc/1d3, r\_galaxy/pc/1d3],$
/keep\_aspect, title='Column\_Density/1E14, '+textoidl('m'^{2})

; draw the color bar. fit it to the location of the image.
cgcolorbar, position=cbposition, range=[min(image), max(image)], /fit

; __________________________ 3rd graph galactic disc and wind _____________

; Set up variables for the contour plot. Normally, these values
; would be passed into the program as positional and keyword parameters.
velocity\_magnitude = sqrt$
(grid.v_bulk_z(grid_radius,*,*)^2 + grid.v_bulk_y(grid_radius,*,*)^2)

image = transpose(velocity_magnitude/1d3)

plane = transpose(grid.plane(grid_radius,*,*))*max(image)

nLevels = 10

dm_position = [0.125, 0.125, 0.9, 0.800]
cb_position = [0.125, 0.865, 0.9, 0.895]

; Set up a "window" for the plot. The PostScript output will have
; the same aspect ratio as the graphics window on the display.
; cgDisplay, 600, 600, Title='Image Plot with Contours'

; Set up colors for contour plot.

cgLoadCT, 33

; Display the image on the display. Keep its aspect ratio.

cgimage, image+plane, stretch=1,$

minvalue=min(image), maxvalue=max(image),$

/axes, xtitle='Z_Distance_{Line_of_Sight}/kpc',$

ytitle='Y_Distance/kpc', position=position, $

xrange=[-r_galaxy/pc/1d3, r_galaxy/pc/1d3],$

yrange=[-r_galaxy/pc/1d3, r_galaxy/pc/1d3],$

/keep_aspect,$

title='Velocity_Field/\text{km}\cdot s^{{-1}}'$

partvelvec, grid.v_bulk_z(grid_radius,*,*) ,$

grid.v_bulk_y(grid_radius,*,*) ,$

r_galaxy/pc/1d3/grid_radius*grid.z(grid_radius,*,*) ,$

r_galaxy/pc/1d3/grid_radius*grid.y(grid_radius,*,*) ,$

/over, fraction=0.3

; Draw the color bar. Fit it to the location of the image.

cgColorbar, Position=cb_position, Range=[min(image), max(image)], /Fit

p.multi=0

ps_end

; r_index = dindgen(ceil((r_max - r_min)/delta_r))

; r_array = [r_min + r_index*delta_r, r_max]

; r_n = size(r, /n_elements)

; w=0 ; window index

; window, w++ ; velocity distribution of Na

; i = spot(0)+grid_radius
; j = spot(1)+grid_radius
; z = grid.z(i,j,*)*r_galaxy/grid_radius
endelse
stop
end

A.4 gas_model_v4.pro

; for best view set tab size to 4 characters. in vim run :set tabstop=4
pro gas_model_v4, windtype_input, v_bulk_exp_input, inclination_input, SUPPRESS_GFX=suppress_gfx
@fc ; load fundamental constants
common shared_parameters, number_density_range, number_density_coefficient, number_density_exp, temp_coefficient, temp_exp, temp0, v_bulk_coefficient, v_bulk_exp, v_bulk0
; share parameters across several functions
; DO NOT CHANGE order of these variables as IDL is dumb

; define input variables

; variables for MAXWELLIAN distribution : m_atomic, T
; variables for guassian distribution : mu, sigma

; Geometry assumptions: observing along z direction
; major axis of the ellipse observed always
; is the x-axis
; galactic center is at (0,0) on x–y plane

; model options

; spot = [0, 0] ; location of the pixel observed on x–y plane
windtype = windtype_input ; 1, sperical
; 2, filled biconical
; 3, unfilled biconical

; windtype = 2d
grid_radius = 50d ; number of pixels along radius
; half length of each dimension
vdt = 2 ; velocity distribution type: 1 for maxwellian
; 2 for gaussian
distance_offset = 1d ; distance offset for plotting galactic plane
bicone_angle = !pi/4d ; half opening angle of bicone with respect to
bicone_thickness = 0.6 * pi/13d ; for windtype 3
bipolar_lowerbound = 0.3 ; for windtype 4
bipolar_rxy = 2.5d3*pc ; ellipsoid parameter
bipolar_rz = 2.3d3*pc ; ellipsoid parameter

; polar axis in spherical coordinates

r_proj = 0.5*r_galaxy ; projected radius, measured vertically
inclination = inclination_input ; we can measure original & projected

; area or radius
inclination = !pi*3/4 ; inclination angle between galactic
disc and z+ axis. range [0, pi)

; used !pi/2 (vertical) and !pi*3/4 (45 degrees) for senior presentation
r_galaxy = 5d3*pc ; actual radius of the galaxy, measured horizontally
r_wind = r_galaxy*0.7 ; radius of the gas
center_perturb = r_galaxy/grid_radius/2 ; amount of perturbation for center

v_hubble = 200d3 ; velocity due to hubble expansion
typically 200km/s (M82)<D>F12>

sigma = 100d3 ; standard deviation for gaussian velocity distribution
mu = 0d ; mean for gaussian

number_density_outer = 1d–5 ; from center to outer boundary
number_density_exp = -2 ; fixed by conservation of mass
temp_exp = -1d ; [-3, 0]
temp_range = [1d4,1d3] ; for maxwell velocity distribution
v.bulk_exp = double[v.bulk_exp_input] ; [0, 2]
v.bulk_range = [0,500d3] ; force 0 intercept (initial velocity)

; without considering velocity due to hubble expansion

lambda_min = 5800d–10 ; lower limit of line profile domain in the
; plot and all calculations
lambda_max = 5950d–10 ; upper limit of line profile domain
delta_lambda = 0.10d–10 ; increment of wavelength
v.gaussian_max = 800d3 ; upper limit of gaussian velocity
delta_v.gaussian = 0.05d3 ; increment of thermal velocity
I0 = 1d ; original intensity of light beam
I lambda0 = 10 ; use a function insted for a radiation

; profile that depends on wavelength.
; eg. blackbody

; ---- Na I D1 transition parameters ----
f1 = 0.319913d ; oscillator strength
bbkj1 = 1d ; b_k/b_j is 1 for bound–bound transition
lambda1 = 5897.5661715d–10 ; central wavelength for transition 1,
      ; 5890A in our case
m_atomic1 = m_atomic_Na ; atomic mass of atom 1

; ---- Na I D2 ----
f2 = 0.640511d ; oscillator strength
bbkj2 = 1d ; b_k/b_j is 1 for bound–bound transition
lambda2 = 5891.58326415d–10 ; central wavelength for transition 2,
      ; 5896A in our case
m_atomic2 = m_atomic_Na ; atomic mass of atom 2

; set coefficients of dependence relations

number_density_coefficient = number_density_outer / r_wind^number_density_exp

temp_coefficient = (temp_range(1)–temp_range(0)) / (r_wind^temp_exp − center_perturb^temp_exp)

if v_bulk_exp lt 1d then begin
v_bulk_coefficient = mean(v_bulk_range)
endif $

else begin
v_bulk_coefficient = (v_bulk_range(1)–v_bulk_range(0)) / (r_wind^v_bulk_exp − center_perturb^v_bulk_exp)
endelse

temp0 = temp_range(1) − temp_coefficient * r_wind^temp_exp
v_bulk0 = v_bulk_range(1) − v_bulk_coefficient * r_wind^v_bulk_exp

; construct lattice

side = 2*grid_radius+1 ; total number of pixels along each dimension
xhat = [1d, 0d, 0d] ; unit vectors in cartesian space
yhat = [0d, 1d, 0d]
zhat = [0d, 0d, 1d]
normal = [ $ ; normal vector
0d, $ ; with respect to galactic disc
sin(inclination−pi/2), $ ; always in the y–z plane
cos(inclination−pi/2) ] $plane_col = normal(0) ; coefficients to specify galactic plane
plane_co2 = normal(1)
plane_co3 = normal(2) ; allocate memory
occupy = dblarr(side,side,side) ; just an empty template
grid = { $ x: occupy, $ ; index of x coordinate
y: occupy, $ ; index of y coordinate
z: occupy, $ ; index of z coordinate
r: occupy, $ ; radial coordinate
theta: occupy, $ ; polar angle with respect to
$ ; the normal direction
$ ; which is the polar axis
number_density: occupy, $ temp: occupy, $ v_bulk: occupy, $ v_bulk_x: occupy, $ v_bulk_y: occupy, $ v_bulk_z: occupy, $ plane: occupy $ ; as logical matrix. value is 1
$ ; if the plane lies in the block
} ; loop for each cell in lattice
; assign index number with respect to central origin, (i−grid_radius), to
every slice
for i = 0, side−1 do begin
occupy = replicate(i−grid_radius,side,side)
grid.x(i,*,*), occup
grid.y(*,i,*), occup
grid.z(*,*), occup
endfor
for i = 0, side−1 do begin
for j = 0, side−1 do begin
for k = 0, side-1 do begin

    position = [ grid.x(i,j,k),
                grid.y(i,j,k),
                grid.z(i,j,k) ] ; dimensionless index numbers

    ; in the cartesian grid

    ; spherical r and theta coordinate with respect to the galactic disk

    grid.r(i,j,k) = norm(position)*r_galaxy/grid_radius ; radial distance

    grid.theta(i,j,k) = acos ( total(position*normal) /
                                (norm(position)*norm(normal)) ) ; angle between two vectors

    ; assume azimuthally symmetric wind

    ; determine if block is a plane block

    los_logic = grid.theta(i,j,* ) > !pi/2

    ; plane_index = SEARCH_ARRAY(los_logic , 0)

    ; plane_index = plane_index(0)

    ; grid.plane(i,j,plane_index) = 1

    distance = abs ( plane_co1*grid.x(i,j,k)+
                     plane_co2*grid.y(i,j,k)+
                     plane_co3*grid.z(i,j,k) )

    / sqrt(plane_co1^2+plane_co2^2+plane_co3^2) ; distance to the galactic disk

    if ( distance le distance_offset )
        then begin
            grid.plane(i,j,k)=1
        endif

    else begin
        grid.plane(i,j,k)=0
    endelse

; fill in physical parameters
APPENDIX A

case windtype of
  1: begin ; spherical wind
     ; outside the wind
     if grid.r(i,j,k) gt r_wind then begin
       grid.number_density(i,j,k) = 0d ; set density to 0
       ; and we are done
     endif
     ; inside the wind
     else begin
       grid.number_density(i,j,k) = number_density_spherical( $ 
       grid.r(i,j,k), $ 
       grid.theta(i,j,k) $ 
     )
     grid.temp(i,j,k) = temp_spherical ( $ 
       grid.r(i,j,k), $ 
       grid.theta(i,j,k) $ 
     )
     v_bulk = position / norm(position) * $ 
     abs ( $ 
       v_bulk_spherical ( $ 
         grid.r(i,j,k), $ 
         grid.theta(i,j,k) $ 
       ) $ 
     )
     ; the function v_bulk_spherical
     ; returns velocities from given
     ; spherical coordinates (r, theta)
     ; in cartesian vector form
     grid.v_bulk(i,j,k) = norm(v_bulk)
     grid.v_bulk_x(i,j,k) = v_bulk(0)
     grid.v_bulk_y(i,j,k) = v_bulk(1)
     grid.v_bulk_z(i,j,k) = v_bulk(2)
   endelse
  2: begin ; filled biconical wind
     theta = grid.theta(i,j,k)
     angle = bicone_angle
end
thickness = bicone_thickness

; outside the wind

if (grid.r(i,j,k) gt r_wind) $
  || 
  (abs(theta-!pi/2) lt (!pi/2-angle)) $
  then begin
    grid.number_density(i,j,k)=0d
  endif

; inside the wind

else begin
  grid.number_density(i,j,k) = 
    number_density_spherical( grid.r(i,j,k), grid.theta(i,j,k) )
  grid.temp(i,j,k) = temp_spherical ( grid.r(i,j,k), grid.theta(i,j,k) )
  v_bulk = position / norm(position) * 
    abs ( 
      v_bulk_spherical( grid.r(i,j,k), grid.theta(i,j,k) )
    )

endelse

3: begin ; unfilled biconical wind

theta = grid.theta(i,j,k)
angle = bicone_angle
thickness = bicone_thickness

; inside the wind
if (grid.r(i,j,k) le r_wind)
   &&
   (abs((theta-angle)) le thickness)
   ||
   (abs((theta-(pi-angle))) le thickness)
then begin ; on either side of galactic disk
grid.number_density(i,j,k) = 
   number_density_spherical(
   grid.r(i,j,k),
   grid.theta(i,j,k)
)
grid.temp(i,j,k) = temp_spherical ( 
   grid.r(i,j,k),
   grid.theta(i,j,k)
)
v_bulk = position / norm(position) * 
   abs ( 
   v_bulk_spherical( 
   grid.r(i,j,k),
   grid.theta(i,j,k)
   )
) ; the function v_bulk_spherical;
; returns velocities from given;
; spherical coordinates (r, theta);
; in cartesian vector form
grid.v_bulk(i,j,k) = norm(v_bulk)
grid.v_bulk_x(i,j,k) = v_bulk(0)
grid.v_bulk_y(i,j,k) = v_bulk(1)
grid.v_bulk_z(i,j,k) = v_bulk(2)
endif

; outside the wind
else begin
grid.number_density(i,j,k)=0d
endelse
end ; finish windtype 3

4: begin ; bipolar wind bubble shell
r = grid.r(i,j,k)
theta = grid.theta(i,j,k)
rxy = bipolar_rxy
rz = bipolar_rz
; two bubbles on both sides with different centers
bipolar_lhs1 = (r*sin(theta))^2/rxy^2 $ + (r*cos(theta)-rz)^2/rz^2
bipolar_lhs2 = (r*sin(theta))^2/rxy^2 $ + (r*cos(theta)+rz)^2/rz^2
; inside either bubble
; restrict value of LHS within lowerbound and 1
if ( $ (bipolar_lowerbound le bipolar_lhs1) $ & & $ (bipolar_lhs1 le 1) $ ) $ || $ ( $ (bipolar_lowerbound le bipolar_lhs2) $ & & $ (bipolar_lhs2 le 1) $ ) $ then begin
grid.number_density(i,j,k) = $ number_density_spherical( $ grid.r(i,j,k), $ grid.theta(i,j,k) $ )
grid.temp(i,j,k) = temp_spherical ( $ grid.r(i,j,k), $ grid.theta(i,j,k) $ )
v_bulk = position / norm(position) * $ abs ( $
v_bulk_spherical( grid.r(i,j,k), grid.theta(i,j,k) )

; the function v_bulk_spherical
; returns velocities from given spherical coordinates (r, theta)
; in cartesian vector form

grid.v_bulk(i,j,k) = norm(v_bulk)
grid.v_bulk_x(i,j,k) = v_bulk(0)
grid.v_bulk_y(i,j,k) = v_bulk(1)
grid.v_bulk_z(i,j,k) = v_bulk(2)

else begin
  grid.number_density(i,j,k) = 0d
endelse

endfor
endfor

; fill galactic center by perturbation

position = [0, center_perturb, 0]  ; index for shifted core position
grid.r(grid_radius, grid_radius, grid_radius) = center_perturb
grid.theta(grid_radius, grid_radius, grid_radius) = acos(
  total(position*normal)  ; dot product
  /
  (norm(position)*norm(normal))  ; product of norms
)
grid.plane(grid_radius, grid_radius, grid_radius) = 1
r_center = center_perturb
theta_center = 1d-3
grid.number_density(grid_radius, grid_radius, grid_radius) =
number_density_spherical(r_center, theta_center)

grid.temp(grid_radius, grid_radius, grid_radius) = $temp_spherical(r_center, theta_center)$

v_bulk = position/norm(position) $ *abs(v_bulk_spherical(r_center, theta_center))$

grid.v_bulk(grid_radius, grid_radius, grid_radius) = norm(v_bulk)

grid.v_bulk_x(grid_radius, grid_radius, grid_radius) = v_bulk(0)

grid.v_bulk_y(grid_radius, grid_radius, grid_radius) = v_bulk(1)

grid.v_bulk_z(grid_radius, grid_radius, grid_radius) = v_bulk(2)

; calculate spaxels

; create wavelength array

lambda_index = dindgen(ceil((lambda_max - lambda_min)/delta_lambda))

lambda = [lambda_min + lambda_index*delta_lambda, lambda_max]

lambda_n = size(lambda, /n_elements)

; first two dimensions for pixel position, third dimension for transition

; selection, forth dimension for storing spectrum

spaxel = dblarr(side, side, 3, lambda_n) ; spaxel
tau = dblarr(side, side, 3, lambda_n) ; optical depth
column_density = dblarr(side, side)
equivalent_width = dblarr(side, side, 3)
mean_velocity = dblarr(side, side)
v_gaussian_index = dindgen(ceil((2*v_gaussian_max/delta_v_gaussian)))

v_gaussian = [ 
- v_gaussian_max + v_gaussian_index*delta_v_gaussian, $v_gaussian_max$
] ; gaussian spread relative to cell rest frame

; integrate along line of sight

for i = 0, side-1 do begin
  for j = 0, side-1 do begin ; loop over all spaxels
    ; i = spot(0)+grid_radius ; for a specific spaxel
    ; j = spot(1)+grid_radius
    ; if line of sight passes through galactic disk plane=1 for some cells
    ; the sum of plane is greater than 0
    v_spaxel = []
    if total(grid.plane(i,j,*)) gt 0 then begin
      for k = side-1, 0, -1 do begin ; integrate along -z direction
        ; stop when hitting galactic plane
if grid.plane(i,j,k) eq 1 then break

; ------ compute integrated absorption coefficient ------
sl = acs_int(f1, bkbj1, lambda2nu(lambda1), grid.temp(i,j,k))
s2 = acs_int(f2, bkbj2, lambda2nu(lambda2), grid.temp(i,j,k))

; ------ velocity and wavelength distribution ------

; array of absolute velocities relative to observer
v = v_add ( 
    grid.v_bulk_z(i,j,k),
    v_add ( 
        v_gaussian ,
        v_hubble
    )
)

; target wavelength space to velocity space
; convert to the grid cell rest frame
; for the first transition
v_rest1 = v_add ( 
    lambda2v(lambda, lambda1),
    v_add ( 
        grid.v_bulk_z(i,j,k),
        -v_hubble
    )
)

; for the second transition
v_rest2 = v_add ( 
    lambda2v(lambda, lambda2),
    v_add ( 
        grid.v_bulk_z(i,j,k),
        -v_hubble
    )
)

case vdt of
    1: begin ; maxwell distribution
        T = grid.temp(i,j,k)
        phi1 = P_v2phi_nu ( 
            v2nu(v_rest1, lambda1),
            lambda1,
            ATOMICMASS=m_atomic1, 
        )
    end
APPENDIX A

448 phi2 = P_v2phi_nu ( v2nu(v_rest2, lambda2), lambda2, ATOMICMASS=m_atomic2, TEMP=T )

456 end

457 2: begin ; gaussian distribution
458 ; assume same gaussian for all locations
459 phi1 = P_v2phi_nu ( v2nu(v_rest1, lambda2nu(lambda1) ), lambda2nu(lambda1), MEAN=nu, SD=sigma )
467 phi2 = P_v2phi_nu ( v2nu(v_rest2, lambda2nu(lambda2) ), lambda2nu(lambda2), MEAN=nu, SD=sigma )

475 endcase

477 ; -------- absorption cross section for each wavelength --------
478 s1_lambda = s1*phi1
479 s2_lambda = s2*phi2
480 s0_lambda = s1_lambda + s2_lambda

482 column_density_block = grid_number_density(i,j,k) *r_galaxy/grid_radius

485 tau(i,j,0,*) = tau(i,j,0,*) - s0_lambda*column_density_block
\[
\text{tau}(i,j,1,*) = \text{tau}(i,j,1,*) - \text{s}_1\text{lambda} \cdot \text{column\_density\_block}
\]
\[
\text{tau}(i,j,2,*) = \text{tau}(i,j,2,*) - \text{s}_2\text{lambda} \cdot \text{column\_density\_block}
\]
\[
\text{column\_density}(i,j) = \text{column\_density}(i,j) + \text{column\_density\_block}
\]
\[
; \text{bulk\ velocity\ array}
\]
\[
\text{v\_spaxel} = [\text{grid\_v\_bulk\_z}(i,j,k), \text{v\_spaxel}]
\]
\[
\text{endfor}
\]
\[
\text{endif}
\]
\[
\text{else\ begin}
\]
\[
\text{v\_spaxel} = 0
\]
\[
\text{endelse}
\]
\[
; \text{__________________________ resultin\_g\ intensity \_________________________}
\]
\[
\text{spaxel}(i,j,0,*) = \text{l}\_\text{lambda0} \cdot \exp(-\text{tau}(i,j,0,*))
\]
\[
\text{spaxel}(i,j,1,*) = \text{l}\_\text{lambda0} \cdot \exp(-\text{tau}(i,j,1,*))
\]
\[
\text{spaxel}(i,j,2,*) = \text{l}\_\text{lambda0} \cdot \exp(-\text{tau}(i,j,2,*))
\]
\[
; \text{__________________________ equivalent\ width \_________________________}
\]
\[
\text{equivalent\_width}(i,j,0) = \text{discrete\_area} (\text{lambda}, 1 - \text{spaxel}(i,j,0,*) / \text{l}\_\text{lambda0})
\]
\[
\text{equivalent\_width}(i,j,1) = \text{discrete\_area} (\text{lambda}, 1 - \text{spaxel}(i,j,1,*) / \text{l}\_\text{lambda0})
\]
\[
\text{equivalent\_width}(i,j,2) = \text{discrete\_area} (\text{lambda}, 1 - \text{spaxel}(i,j,2,*) / \text{l}\_\text{lambda0})
\]
\[
\text{if n\_elements(v\_spaxel) eq 0 then begin}
\]
\[
\text{mean\_velocity}(i,j) = 0
\]
\[
\text{endif}
\]
\[
\text{else\ begin}
\]
\[
\text{mean\_velocity}(i,j) = \text{mean}(v\_spaxel)
\]
\[
\text{endelse}
\]
\[
\text{endfor}
\]
\[
\text{print}, \ 'finished\_integrating\_over\_spaxal\_column\_x=\_', + \text{string}(i)
\]
\[
\text{endfor}
\]
\[
; \text{__________________________ save\ cache\;\ debug \_________________________}
\]
\[
; \text{save\, /all},
\]
file = 'E:\honors\results\spaxel_wt' + $
strtrim(floor(windtype),2)+'_v'+'$ 
strtrim(floor(v_bulk_exp),2)+'_res'+'$ 
strtrim(floor(grid_radius),2)+'incl'+'$ 
'.sav

; =============== plots ========================
if keyword_set(suppress_gfx) then begin
endif $
else begin

; =========== spaxel display ===========
ps_start, /encap, /nomatch,$
file = 'E:\honors\results\spaxel_wt' + $
strtrim(floor(windtype),2)+'_v'+'$ 
strtrim(floor(v Bulk_exp),2)+'_res'+'$ 
strtrim(floor(grid_radius),2)+'incl'+'$ 
'.eps

x = grid_radius+1 ; set center coordinates
y = grid_radius+1
spec0 = spaxel(x,y,0,*)
spec1 = spaxel(x,y,1,*)
spec2 = spaxel(x,y,2,*)
margin = [2,2,2,2]
cgplot , lambda*1d10, spec0/l_lambda0,$
background='white', color='black', axiscolor='black',$
xtitle = 'Wavelength/\ang'+cgsymbol('Angstrom'),$
ytitle = 'Relative,Flux',$
xr = [5870d, 5930d],$
xsty=1, charthick = 1, thick = 2
cgoplot ,lambda*1d10, spec1/l_lambda0,$
background='white', color='red', axiscolor='black',$
charthick = 1, thick = 2
cgoplot ,lambda*1d10, spec2/l_lambda0,$
background='white', color='blue', axiscolor='black',$
charthick = 1, thick = 2
ps_end

; =============== plots ========================
\begin{verbatim}
; /--------------------------------------------------/
; / n-r T-r v_bulk-r gaussian /
; /--------------------------------------------------/

ps_start, /encap, /nomatch, xsize = 10, ysize = 2.3,
file='E:\honors\results\plots_wt '+
strtrim(floor(windtype),2)+'_v^'+
strtrim(floor(v_bulk_exp),2)+'_res'+
strtrim(floor(grid_radius)*2,2)+'_incl'+
strtrim(inclination,2)+
'.eps'
; !p.multi=[0, 4, 2, 0, 0]
; margin = [2, 2, 1, 0]
; generate radial array
; not extracting data from grid because of undetermined empty cells
r_index = dindgen(grid_radius)
reverse(r_index), r_index
r_plot = r_index/grid_radius*r_galaxy
theta_plot = r_plot*0

; ~~~~~~~~~~~~~~~~~~~ plot 1: n-r ~~~~~~~~~~~~~~~~~~~
n_plot = number_density_spherical(r_plot,theta_plot)
cgplot, r_plot/pc/1d3, n_plot/1d6, /ylog,
layout = [4,1,1], aspect = 1,
background='white', color='red', axiscolor='black',
title = 'Number Density',
xtile = 'Radial Distance/\,kpc',
ytitle = 'Number Density/\,\,cm^\{-3\}',
+textoidl('cm\{-3\}')

; ~~~~~~~~~~~~~~~~~~~ plot 2: T-r ~~~~~~~~~~~~~~~~~~~
T_plot = temp_spherical(abs(r_plot),theta_plot)
cgplot, r_plot/pc/1d3, T_plot, /ylog,
layout = [4,1,2], aspect = 1,
background='white', color='red', axiscolor='black',
title = 'Temperature',
xtile = 'Radial Distance/\,/kpc',
ytitle = 'Temperature/\,K'

; ~~~~~~~~~~~~~~~~~~~ plot 3: v_bulk-r ~~~~~~~~~~~~~~~~~~~
v_bulk_plot = v_bulk_spherical(abs(r_plot),theta_plot)
v_bulk_plot = (v_bulk_plot\,1e\,1d4)*1d4+v_bulk_plot
\end{verbatim}
APPENDIX A

```plaintext
600  cgplot, r_plot/pc/1d3, v_bulk_plot/1d3, /ylog, $ 601      layout = [4,1,3], aspect = 1, $ 602      background='white', color='red', axiscolor='black', $ 603      title = 'Bulk/Velocities', $ 604      xtitle = 'Radial Distance../kpc', $ 605      ytitle = 'Bulk Velocity../u' $ 606      ; textoidl('km\cdot s^{-1}') $ 607      plot 4: gaussian distribution $ 608      cgplot, v_gaussian/1d3, P_v_gaussian(v_gaussian, mu, sigma), $ 609      layout = [4,1,4], aspect = 1, $ 610      background='white', color='red', axiscolor='black', $ 611      title = 'Gaussian Speed Distribution', $ 612      xtitle = 'Speed Along Line of Sight../u', $ 613      ytitle = 'Probability Density' $ 614      ps_end

; ----------------- colormaps ------------------
  /encap, /nomatch, xsize = 10, ysize = 3, $ 620      file='E:\honors\results\colormap_wt'+ 621      strtrim(floor(windtype),2)+'_v^'+ 622      strtrim(floor(v_bulk_exp),2)+'_res '+ 623      strtrim(floor(grid_radius)*2,2)+'_incl '+ 624      strtrim(inclination,2)+' .eps' 626      ; colormap 1: v field, wind shape and plane orientation ----
  ; Set up variables for the contour plot. Normally, these values would 629      ; be passed into the program as positional and keyword parameters. 633      velocity_magnitude=sqrt ( 634        grid.v_bulk_z(grid_radius,*,* )^2 635        + grid.v_bulk_y(grid_radius,*,* )^2 636      ) 638      image = transpose(velocity_magnitude)/1d3 ; convert to km/s 640      plane = transpose(grid.plane(grid_radius,*,* ))*max(image) 642      nLevels = 10 644      imgposition = [0.044, 0.020, 0.236, 0.870]
```

The code snippets represent contour plots and colormap generation for various data sets, including bulk velocity distributions and Gaussian speed distributions. It also includes calculations for velocity magnitude and image processing for visualization.
cbposition = [0.044, 0.900, 0.236, 0.950]

; Set up a "window" for the plot. The PostScript output will have
; the same aspect ratio as the graphics window on the display.
; cgDisplay, 600, 600, Title='Image Plot with Contours'
; Set up colors for contour plot.
cgLoadCT, 33

; Display the image on the display. Keep its aspect ratio.
cgimage, image+plane, /axes, /keep_aspect, layout = [4,1,1],
position = imgposition, charsize = 1.4, stretch=1,
minvalue = min(image), maxvalue = max(image),
title = 'Velocity Field',
+textoidl('km\cdot s^{-1}'),
xtitle = 'Z Distance (Line of Sight)/kpc',
ytitle = 'Y Distance/kpc',
xrange = [-r_galaxy/pc/1d3, r_galaxy/pc/1d3],
yrange = [-r_galaxy/pc/1d3, r_galaxy/pc/1d3]

; overplot velocity field
partvelvec, grid.v_bulz(grid_radius,*),
grid.v_bulk_y(grid_radius,*),
r_galaxy/pc/1d3/grid_radius*grid.z(grid_radius,*),
r_galaxy/pc/1d3/grid_radius*grid.y(grid_radius,*),
/over, fraction=0.3

; Draw the color bar. Fit it to the location of the image.
cgColorbar, position=cbposition, charsize = 0.6,
Range = [min(image), max(image)]

; -------------- colormap 2: column density --------------
; Set up variables for the contour plot. Normally, these values would
; be passed into the program as positional and keyword parameters.
image = column_density/1d14/1d4 ; scaling factor
nLevels = 10

imgposition = [0.294, 0.020, 0.486, 0.870]

; Set up a "window" for the plot. The PostScript output will have
; the same aspect ratio as the graphics window on the display.
; cgDisplay, 600, 600, Title='Image Plot with Contours'
; Set up colors for contour plot.
cgLoadCT, 33
; Display the image on the display. Keep its aspect ratio.
cgimage, image, /axis, /keep_aspect,$
  layout = [4,1,2],$
  position = imgposition, chsize = 1.4, stretch=1,$
  minvalue=min(image), maxvalue=max(image),$
  title='Column_Density/\,\,10^{14}\,\,cm^{-2}',$
  xtitle='X_Distance/\,\,kpc',$
  ytitle='Y_Distance/\,\,kpc',$
  xrange=[r_galaxy/pc/1d3, r_galaxy/pc/1d3],$
  yrange=[r_galaxy/pc/1d3, r_galaxy/pc/1d3]$
; Draw the color bar. Fit it to the location of the image.
cgColorbar, position=cbposition, chsize = 0.6,$
  Range=[min(image), max(image)]$
; Set up variables for the contour plot. Normally, these values would
; be passed into the program as positional and keyword parameters.
image = equivalent_width(*,*0)*1d9 ; convert to nm
nLevels = 10
; imgposition = [0.125, 0.125, 0.9, 0.800]
imgposition = [0.544, 0.020, 0.736, 0.870]
; Set up a "window" for the plot. The PostScript output will have
; the same aspect ratio as the graphics window on the display.
cgDisplay, 600, 600, Title='Image Plot with Contours'$
; Set up colors for contour plot.
cgLoadCT, 33$
; Display the image on the display. Keep its aspect ratio.
cgImage, image, /axes, /keep_aspect,$
  layout = [4,1,3],$
  position = imgposition, chsize = 1.4, stretch=1,$
  minvalue=min(image), maxvalue=max(image),$
  title='Equivalent_Width/\,\,nm',$
  xtitle='X_Distance/\,\,kpc',$
  ytitle='Y_Distance/\,\,kpc',$
  xrange=[-r_galaxy/pc/1d3, r_galaxy/pc/1d3],$
  yrange=[-r_galaxy/pc/1d3, r_galaxy/pc/1d3]$
; Draw the color bar. Fit it to the location of the image.
cgColorbar, position=cbposition, chsize = 0.6,
Range=[min(image), max(image)]

; Set up variables for the contour plot. Normally, these values would
; be passed into the program as positional and keyword parameters.
image = mean_velocity/1d3

nLevels = 10

imgposition = [0.794, 0.020, 0.986, 0.870]
cbposition = [0.794, 0.900, 0.986, 0.950]

; Set up a "window" for the plot. The PostScript output will have
; the same aspect ratio as the graphics window on the display.
cgDisplay, 600, 600, Title='Image Plot with Contours'

; Set up colors for contour plot.
cgLoadCT, 33

cgImage, image, /axes, /keep_aspect,
   layout = [4,1,4],
   position = imgposition, charsize = 1.4, stretch=1,
   minvalue=min(image), maxvalue=max(image),
   title= 'Mean_Bulk_Velocity/,,'
      +textoidl('km\text{\cdot}s^{-1})',
   xtitle='X\ Distance/,,kpc',
   ytitle='Y\ Distance/,,kpc',
   xrange=[-r_galaxy/pc/1d3, r_galaxy/pc/1d3],
   yrange=[-r_galaxy/pc/1d3, r_galaxy/pc/1d3]

; Draw the color bar. Fit it to the location of the image.
cgColorbar, position=cbposition, charsize = 0.6,
   Range=[min(image), max(image)]

ps_end

; w=0 ; window index
; window, w++ ; velocity distribution of Na
i = spot(0)+grid_radius
j = spot(1)+grid_radius
z = grid.z(i,j)*r_galaxy/grid_radius
endelse

; stop
end
Appendix B

Function Files

B.1 acs_int.pro

function acs_int, f, bkbj, nu, T

;calculate integrated absorption cross section
;for a given upward oscillator strength
;
;bkbj : the ratio b_k/b_j is 1 for bound-bound transition
;f : upward oscillator strength
;lambda : central wavelength for a certain transition
;T : temperature

@fc

s_u = (pi*e^2)/(m_e*c)*f/epsilon ; uncorrected for stimulated emission
s = s_u*(1-bkbj*exp(-(h*nu)/(k*T))) ; corrected for stimulated emission

return, s

end

B.2 discrete_area.pro

function discrete_area, xdata, ydata
B.3  lambda2nu.pro

```fortran
function lambda2nu, lambda
  @fc
  nu = c/lambda
  return, nu
end
```

B.4  lambda2v.pro

```fortran
function lambda2v, lambda, lambda0
  @fc
  v = (lambda^2d - lambda0^2d)/(lambda^2d + lambda0^2d)*c
  return, v
end
```
B.5 number_density_spherical.pro

function number_density_spherical, r, theta
; number of absorbing atoms in unit volume along the radial direction
; input: r, theta
; rho = r^-2
common shared_parameters, number_density_range, $
    number_density_coefficient, number_density_exp, $
    temp_coefficient, temp_exp, temp0, $
    v_bulk_coefficient, v_bulk_exp, v_bulk0
; share parameters across several functions
number_density = number_density_coefficient* $ 
    r^(number_density_exp) $ 
    *(abs(cos(theta)))^0
return, number_density
end

B.6 p_v_gaussian.pro

function P_v_gaussian, v, mu, sigma
; compute the probability DENSITY of a gaussian distribution
; for a given velocity (normalized)
; v : velocity
; mu : mean/expectation value of velocity
; sigma : standard deviation
;@
@fc ; load fundamental constants
P_v_gaussian = 1d/(sigma*sqrt(2d*!pi))*exp(-1d/2d)*((v-mu)/sigma)^2d)
return, P_v_gaussian
end

B.7 p_v_maxwell.pro
function P_v_maxwell, v, m_atomic, T
; maxwell–boltzmann velocity distribution in one dimension
; compute the probability DENSITY for a given velocity (normalized)
; m_atomic : atomic mass
; m=22.989769282 for sodium
; T : temperature of gas
; v : velocity

@fc ; load fundamental constants

m=m_u*m_atomic ; compute actual mass of the atom
P_v_maxwell = sqrt(m/(2*pi*k*T))*exp((m*v^2)/(2*k*T))
return, P_v_maxwell
end

B.8  p_v2phi_lambda.pro

function P_v2phi_lambda, lambda, lambda0, $
ATOMICMASS = atomicmass, TEMP = temp, $
; for maxwell distribution
MEAN = mean, SD = sd
; for gaussian distribution

; convert maxwell–boltzmann velocity distribution to wavenlength distribution
; for a certain trasition and given wavelength to be observed

@fc ; load fundamental constants

if keyword_set(TEMP) then begin
m_atomic=atomicmass
T=temp;
phi = $
P_v_maxwell((lambda^2-lambda0^2)/(lambda^2+lambda0^2)*c, m_atomic, T)$
*(4*lambda0^2*lambda)/(lambda0^2+lambda^2)^2*c
endif
if keyword_set(SD) then begin
mu=mean
sigma=sd
phi = 
\[ \text{P}_v\_\text{gaussian}\left( \frac{\lambda^2 - \lambda_0^2}{\lambda^2 + \lambda_0^2} \right) \cdot c, \mu, \sigma \] 
* \left( \frac{4 \lambda_0^2 \lambda}{\lambda_0^2 + \lambda^2} \right)^2 \cdot c 
endif
return, phi
end

B.9  \text{p}_v\_\text{phi}_\text{nu}.pro

function P\_v\_phi\_nu, \nu, \nu_0, ATOMICMASS = atomicmass, TEMP = temp, MEAN = mean, SD = sd
; for maxwell distribution
; for gaussian distribution

@fc
; load fundamental constants

if keyword_set(TEM) then begin
m_\text{atomic}=\text{atomicmass}
T=\text{temp};
phi = P\_v\_\text{maxwell}\left( \frac{\nu_0^2 - \nu^2}{\nu_0^2 + \nu^2} \right) \cdot c, \mu, \sigma \right) 
* \left( \frac{4 \nu_0^2 \nu}{\nu_0^2 + \nu^2} \right)^2 \cdot c 
endif
if keyword_set(SD) then begin
mu=mean
sigma=sd
phi = P\_v\_\text{gaussian}\left( \frac{\nu_0^2 - \nu^2}{\nu_0^2 + \nu^2} \right) \cdot c, \mu, \sigma \right) 
* \left( \frac{4 \nu_0^2 \nu}{\nu_0^2 + \nu^2} \right)^2 \cdot c 
endif
return, phi
end
B.10 partvelvec.pro

;+  
; NAME:  
; PARTVELVEC  
;  
; PURPOSE:  
; Plot the velocity vectors of particles at their positions  
;  
; EXPLANATION:  
; This procedure plots the velocity vectors of particles (at the  
; positions of the particles).  
;  
; CATEGORY:  
; Plotting, Two-dimensional.  
;  
; CALLING SEQUENCE:  
; PARTVELVEC, VELX, VELY, POSX, POSY [, X, Y]  
;  
; INPUTS:  
; VELX: An array of any dimension, containing the x-components  
; of the particle velocities.  
; VELY: An array of the same dimension as velx, containing the  
; y-components of the particle velocities.  
; POSX: An array of the same dimension as velx, containing the  
; x-components of the particle positions.  
; POSY: An array of the same dimension as velx, containing the  
; y-components of the particle positions.  
;  
; OPTIONAL INPUTS:  
; X: Optional abscissa values. X must be a vector.  
; Y: Optional ordinate values. Y must be a vector. If only X  
; is specified, then Y is taken equal to be equal to X.  
;  
; OPTIONAL INPUT KEYWORD PARAMETERS:  
; FRACTION: The fraction of the vectors to plot. They are  
; taken at random from the complete sample. Default is  
; FRACTION = 1.0, use all vectors  
;
; LENGTH: The maximum vectorlength relative to the plot data window. Default = 0.08
; COLOR: Color for the vectors, axes and titles by string name or number (see cgCOLOR). Note that if VECCOLORS is supplied, then the COLOR keyword still specifies the color of the axes and title. Default = 'Opposite'
; OVER: Plot over the previous plot
; VECCOLORS: The vector colors. Must be either a scalar, or a vector (numeric or string) the same size as VELX. Set to COLOR by default.
; WINDOW – Set this keyword to plot to a resizeable graphics window
; Plot All other keywords available to cgPlot (e.g. AXISCOLOR,
; Keywords: LINESTYLE, XRANGE) are available (via _EXTRA)

; OUTPUTS:
; This procedure plots the velocity vectors (VELX, VELY) at the positions of the particles, (POSX, POSY). If X and Y are not specified, then the size of the plot is such that all vectors just fit within the plot data window.

; SIDE EFFECTS:
; Plotting on the current device is performed.

; EXAMPLE:
; Generate some particle positions and velocities.
; POSX=RANDOMU(seed,200)
; POSY=RANDOMU(seed,200)
; VELX=RANDOMU(seed,200)–0.5
; VELY=RANDOMU(seed,200)–0.5
; Plot the particle velocities.
; PARTVELVEC, VELX, VELY, POSX, POSY
Example using vector colors.

`POSX= RANDOMU(seed,200)`
`POSY= RANDOMU(seed,200)`
`VELX= RANDOMU(seed,200) - 0.5`
`VELY= RANDOMU(seed,200) - 0.5`
`magnitude = SQRT(velx^2 + vely^2)`
`LOADCT, 5, NCOLORS=254, BOTTOM=1 ; Load vector colors`
`colors = BytScl(magnitude, Top=254) + 1B`
`PARTVELVEC, VELX, VELY, POSX, POSY, COLOR='green', VECCOLORS=colors`

MODIFICATION HISTORY:

Written by: Joop Schaye (jschaye@astro.rug.nl), Sep 1996.
Added /OVER keyword Theo Brauers (th.brauers@fz-juelich.de) Jul 2002
Added VECCOLORS keyword. David Fanning (david@dfanning.com) March, 2005
Incorporate the Coyote Graphics (cg) plot programs WL January 2011

`; Various settings, modify these to customize`

`c = {customize, $
length: 0.08, $ ; Maximum vector length relative to plot region. (*)
lengtharrow: 0.3, $ ; Length of arrowhead legs relative to vectorlength.
angle: 22.5 } ; 1/2 times the angle between the arrowhead legs.

; (*) Not used if keyword LENGTH is present`

; Some error handling`
on_error, 2 ; Return to caller if an error occurs.

nparams = n_params()

IF nparams NE 4 THEN BEGIN
  IF nparams NE 5 AND nparams NE 6) THEN BEGIN
    message, 'Wrong number of parameters!' / continue
    message, 'Syntax: PARTVELVEC, VELX, VELY, POSX, POSY, [ , X, Y ] ' , $
    / noname, / noprefix
  ENDIF

  IF nparams EQ 5 THEN y=x
  sizex = size(x)
  sizey = size(y)
  IF (sizex[0] NE 1 || sizey[0] NE 1) THEN $
    message, 'X and Y must be vectors!
  ENDIF

  sizevelx = size(velx)
  sizevely = size(vely)
  sizeposx = size(posx)
  sizeposy = size(posy)

  IF (total(sizevelx[0:sizevelx[0]] - sizevely[0:sizevelx[0]]) NE 0 $
    || total(sizevelx[0:sizevelx[0]] - sizeposx[0:sizevelx[0]]) NE 0 $
    || total(sizevelx[0:sizevelx[0]] - sizeposy[0:sizevelx[0]]) NE 0) THEN $
    message, 'All arguments must have the same dimension and size!
  ENDIF

  IF n_elements(fraction) GT 0 THEN $
    IF (fraction LT 0.0 || fraction GT 1.0) THEN $
      message, 'Fraction has to be between 0.0 and 1.0.
    ENDIF

nvecs = n_elements(velx) ; Number of particles.
vel = sqrt(v px^2 + v py^2) ; Total velocity.
maxvel = max(vel) ; Maximum velocity.

; Compute maximum length of vectors.
IF n_elements(length) LE 0 THEN length = c.length
minposx = min(posx)
maxposx = max(posx)
minposy = min(posy)
maxposy = max(posy)
length = length * ((maxposx - minposx) > (maxposy - minposy))

; Convert velocities.
v x = length * v px / maxvel
v y = length * v py / maxvel
vel = length * temporary(vel) / maxvel

; Make sure no vectors extend beyond the plot data window.
x 1 = posx + v x ; End of vector.
y 1 = posy + v y
IF nparams EQ 4 THEN BEGIN
minposx = min(x 1) < minposx
maxposx = max(x 1) > maxposx
minposy = min(y 1) < minposy
maxposy = max(y 1) > maxposy
ENDIF

angle = c. angle * ! dtor ; Convert from degrees to radians.
sinangle = sin(angle) ; Need these.
cosangle = cos(angle)

if N_elements(color) EQ 0 then color = cgcolor('opposite')
IF n_elements(veccolors) EQ 0 THEN BEGIN
veccolors = Replicate(cgcolor('opposite'), nvecs)
nvc = N_Elements(veccolors)
CASE nvc OF
1: veccolors = Replicate(veccolors, nvecs)
ELSE: Message, 'Vector_color_array_VECCOLORS_must_be_same_size_as_VELX.'
ENDCASE
ENDIF ELSE BEGIN
187 IF n_elements(over) EQ 0 THEN BEGIN
188 IF nparms EQ 4 THEN $
189 cgPlot,[minposx,maxposx],[minposy,maxposy],axiscolor=color,$
190 /nodata,/xstyle,/ystyle,COLOR=color,window=window,_EXTRA=extra $
191 ELSE cgPlot,x,y,/nodata,/xstyle,/ystyle,COLOR=color,window=window,_EXTRA=extra
192 ENDIF
193 IF keyword_set(window) then cgcontrol,execute=0
194 ; Plot vectors
195 ;
196 ENDIF ELSE BEGIN
197 IF (n_elements(fraction) GT 0) && (fraction NE 1.0) THEN BEGIN
198 nrgood=long(fraction*nvecs); # of vectors to plot.
199 IF nrgood EQ 0 THEN return
; Compute indices of vectors to plot. I use two lines to get more
; random "random numbers".
200 good=long(randomu(seed,nrgood+1)*(nvecs-1.0))
201 good=good[1:*
202 vx = temporary(vx[good])
203 vy = temporary(vy[good])
204 px = posx[good] ; Can’t use temporary if we want to keep the data.
205 py = posy[good]
206 x1 = temporary(x1[good])
207 y1 = temporary(y1[good])
208 nvecs=nrgood
209 ENDIF ELSE BEGIN
210 px=posx
211 py=posy
212 ENDIF
FOR i=0l,nvecs-ll DO BEGIN ; Loop over particles.
    ; Note that we cannot put the next three lines outside the loop,
    ; because we want the arrow size to be relative to the vector length.
    r = c.lengtharrow*vel[i] ; Length of arrow head.
    r = r*sinangle
    r = r*cosangle
    ; Draw basis, arrow leg, same arrow leg, other arrow leg.
    ; One arrow leg is drawn twice, because we need to return to the end
    ; of the vector to draw the other leg.
    cgPlots,[px[i],x1[i],x1[i]-(vx[i]*rcos+vy[i]*rsin)/vel[i], $
    x1[i],x1[i]-(vx[i]*rcos-vy[i]*rsin)/vel[i]], $
    [py[i],y1[i],y1[i]-(vy[i]*rcos-vx[i]*rsin)/vel[i], $
    y1[i],y1[i]-(vy[i]*rcos+vx[i]*rsin)/vel[i]],COLOR=veccolors[i],$
    ADDCMD = window
ENDFOR
if keyword_set(window) then cgcontrol,execute=1
return
END ; End of procedure PARTVELVEC.

B.11 temp_spherical.pro

function temp_spherical , r , theta
    ; temperature as a function of radius
    ; input: r , theta
    common shared_parameters , number_density_range, $
    number_density_coefficient , number_density_exp, $
    temp_coefficient , temp_exp, temp0, $
    v_bulk_coefficient , v_bulk_exp, v_bulk0
    ; share parameters across several functions
    T = temp0 + temp_coefficient*r^temp_exp*(abs(cos(theta)))^temp_coefficient
    return , T
end

B.12 v_add.pro
function v_add, v0, u

; accurate velocity addition according special relativity
; input parameters
; v0: original velocity in rest frame
; u: velocity of moving frame with respect to rest frame

v = (u + v0)/(1 + u*v0/c^2)
return v
end

B.13  v_bulk_spherical.pro

function v_bulk_spherical, r, theta

; bulk velocity of gas as a function of radius
; input: r, theta
; v = r^2

common shared_parameters, number_density_range, number_density_coefficient, number_density_exp,
temp_coefficient, temp_exp, temp0,
v_bulk_coefficient, v_bulk_exp, v_bulk0
; share parameters across several functions
v_bulk = v_bulk0 + v_bulk_coefficient*r^(v_bulk_exp)
return v_bulk
end

B.14  v2lambda.pro

function v2lambda, v, lambda0

@fc
beta = v/c
lambda= lambda0*sqrt((1+beta)/(1-beta))
return lambda
B.15  v2nu.pro

function v2nu, v, nu0

@fc  ; load fundamental constants

beta = v/c
nu = nu0*sqrt((1 - beta)/(1 + beta))

return, nu

end
Bibliography


