


การแจกแจงความเร็วของอะตอมไฮโดรเจนตามแนวโซลาร์เอเพกซ์



นายชาญเรืองฤทธิ์ จันทร์นอก

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาฟิสิกส์ ภาควิชาฟิสิกส์

คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย

ปีการศึกษา 2543

ISBN 974-13-0236-3

ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

VELOCITY DISTRIBUTION OF HYDROGEN ATOMS ALONG THE SOLAR APEX

MR.CHANRUANGRITH CHANNOK

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Physics
Department of Physics
Faculty of Science
Chulalongkorn University
Academic Year 2000
ISBN 974-13-2036-3

Thesis Title VELOCITY DISTRIBUTION OF HYDROGEN ATOMS TOWARD THE
SOLAR APEX
By Mr. Chanruangrith Channok
Field of Study Physics
Thesis Advisor Associate Profesor David Ruffolo, Ph.D.

Accepted by the Faculty of Science, Chulalongkorn University in partial
Fulfillment of the Requirement for the Master's Degree

..... Dean of Faculty of Science
(Associate Professor Wanchai Phothiphichitr, Ph.D.)

THESIS COMMITTEE

..... Chairman
(Assistant Professor Pirapat Sirisomboonlarp, Ph.D.)

..... Thesis Advisor
(Associate Professor David Ruffolo, Ph.D.)

..... Member
(Assistant Professor Kajornyod Yoodee, Ph.D.)

..... Member
(Ahpisit Ungkitchanukit, Ph.D.)

นายชาญเรืองฤทธิ์ จันทน์นอก : การแจกแจงความเร็วของอะตอมไฮโดรเจนตามแนวโซลาร์เอเพกซ์
(VELOCITY DISTRIBUTION OF HYDROGEN ATOMS TOWARD THE SOLAR APEX)

อ. ที่ปรึกษา : รองศาสตราจารย์ ดร. เดวิด รุฟโฟโล, 112 หน้า. ISBN 974-13-0236-3.

ได้ทำการคำนวณอันตรกิริยาของสสารตัวกลางระหว่างดาว(ส่วนใหญ่เป็นอะตอมไฮโดรเจน) ที่ไหลเข้ามาสู่ระบบสุริยะ ซึ่งมีอันตรกิริยากับลมสุริยะ(ส่วนใหญ่เป็นอนุภาคโปรตอน) ที่พุ่งจากดวงอาทิตย์ ทั้งในและนอกระบบสุริยะ โดยใช้แนวคิดจากประยุกต์สมการขนส่งอนุภาคของโบลทซ์มานน์และผลของการชนกันโดยการแลกเปลี่ยนประจุของไฮโดรเจนกับโปรตอนเพื่อหาฟังก์ชันการกระจายตัวของอะตอมไฮโดรเจน $f(r,v,t)$ ในเฟสสเปซโดยพิจารณาฟังก์ชันการกระจายตัวของไฮโดรเจนในหนึ่งมิติตามแนวโซลาร์เอเพกซ์และในสองมิติของความเร็ว $f_H(z,v_z,v_y,t)$ โดยวิธีการเชิงตัวเลข จากผลการจำลองด้วยคอมพิวเตอร์เราสามารถศึกษาได้ถึง 3 แหล่งกำเนิดของอะตอมไฮโดรเจน ซึ่งเป็นผลมาจากการไหลของตัวกลางระหว่างดาว หรือจากการแลกเปลี่ยนประจุกับโปรตอนใน 3 บริเวณที่ต่างกัน ผลของเรานี้สามารถใช้ปรับปรุงการจำลองในรูปแบบของไหลของโครงสร้างภายนอกของระบบสุริยะได้เป็นอย่างดี

ภาควิชา.....ฟิสิกส์.....ลายมือชื่อนิสิต.....
สาขาวิชา.....ฟิสิกส์.....ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา.....2543.....

##4172270123 : MAJOR PHYSICS

KEY WORD: SPACE PHYSICS/ INTERSTELLAR MATTER/ SOLAR WIND/ CHARGE
EXCHANGE INTERACTION/ BOLTZMANN TRANSPORT EQUATION

CHANRUANGRITH CHANNOK : VELOCITY DISTRIBUTION OF
HYDROGEN ATOMS TOWARD THE SOLAR APEX

THESIS ADVISOR: ASSOCIATE PROFESSOR DAVID RUFFOLO, Ph.D.
112 pp. ISBN 974-13-0236-3.

We have calculated the interaction between interstellar matter (mostly hydrogen atoms) flowing into the solar system with the solar wind (mostly protons) emitted from the Sun inside and outside the solar system. We have applied the Boltzmann transport equation and a model of charge exchange collisions between hydrogen and protons in order to determine the distribution function $f(r, v, t)$ of hydrogen atoms in phase space. We calculate the distribution function of hydrogen in one dimension along the solar apex and in two dimensions of velocity space, $f_H(z, v_z, v_y, t)$, by numerical methods. From our computer simulation results, we are able to study 3 populations of hydrogen atoms, resulting from inflow of the interstellar medium or from charge exchange with protons in 3 distinct regions. Our results can be used to improve full fluid models of the structure of the outer solar system.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Acknowledgement

The author wishes to express his sincere appreciation and gratitude to his supervisor, Assoc. Prof. Dr. David Ruffolo, for great advice, guidance and encouragement given throughout the course of the investigation.

He would like to thank the members of committee, Asst. Prof. Dr. Pirapat Sirisomboonlarp, Asst. Prof. Dr. Kajornyod Yoodee, and Dr. Ahpisit Ung kitchanukit for their reading and criticizing the manuscript.

He would like to thank the members of the computational astrophysics research laboratory of Chulalongkorn University especially for Dr. Tanin Nutaro, Dr. Udomsilp Pinsook, Mr. Paisan Tooprakai, Mr. Kittipat Malakit, Miss Panita Boonma, and Miss Thiranee Khumlumlert.

He would like to thank his lovely friends: Anucha Yangthaisong (A+), Anuson Niyompan (Pongo), Khomsorn Lomthaisong (Eat), Paisan Boonhan, Sripajak Kongsuk (Jak), Teerachai Chamnanmor, Sam Srisuro, Tanomsak Prom prasit. Many thanks go to Chittakorn Polyon, Saktavorn Pongvanitchaya, and Thanittha Saravit. Finally, he would like to thank his family.

This research was supported by a Chulalongkorn University research grant and the King Prajathipok and Queen Rambhai Barni Memorial Foundation.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Table of Contents

Abstract in Thai	iv
Abstract in English	v
Acknowledgement	vi
Contents	vii
List of Figures	ix
Chapter 1 Introduction	1
Chapter 2 Theoretical Background	5
2.1 Space Physics and Heliospheric Physics	5
2.1.1 The solar wind	5
2.1.2 The interstellar wind	7
2.2 Boltzmann Transport Equation	9
2.2.1 Streaming process	11
2.2.2 Collision process	12
2.2.3 Ionization process	15
2.3 Maxwellian Distribution Function	16
Chapter 3 Numerical Method	18
3.1 Finite Differencing	18
3.2 Operator Splitting	23
3.3 Advection Process and TVD Method	25

3.4 Interpolation.....	27
3.4.1 Linear interpolation.....	28
3.4.2 Bilinear interpolation.....	29
3.4.3 Geometric interpolation.....	30
3.5 Numerical Procedure.....	31
3.5.1 Mathematical formulation.....	31
3.5.2 Numerical technique.....	35
3.5.3 Numerical simulation program.....	38
Chapter 4 Results	41
Chapter 5 Discussion and Summary	59
5.1 Comparison with Previous Results.....	61
5.2 Summary.....	64
References	66
Appendices.....	70
Appendix A Cross Section of Charge Transfer	
between H and H+	71
Appendix B Solar Wind Proton Distribution	75
Appendix C Source Code	83
Curriculum Vitae	112

List of Figures

Figure 1.1	Schematic of the heliosphere.	2
Figure 2.1	Schematic of the galaxy and the part of galaxy within 10 light-years of the Sun.	6
Figure 2.2	The solar wind velocity distribution around the Sun from the ULYSSES spacecraft.	8
Figure 2.3	The photoionization process for neutral hydrogen atoms.	10
Figure 2.4	The collision of two particles.	13
Figure 3.1	Discretization of $x \in [a, b]$	19
Figure 3.2	Discretization in two-dimensional x - y space.	22
Figure 3.3	Linear interpolation in one dimension.	28
Figure 3.4	Bilinear interpolation in two dimensions.	29
Figure 3.5	Charge exchange between a proton and a hydrogen atom in the center of mass frame.	32
Figure 3.6	Grid points in velocity space in our simulation model (scale in km/s).	38
Figure 3.7	Flow chart of the simulation program.	40
Figure 4.1	Distribution function in velocity space of hydrogen atoms that flow in through the boundary at 200 AU. In this figure and subsequent figures, f_H is in units of $\text{cm}^{-3}(\text{eV}/c)^{-3}$	42
Figure 4.2	Distribution function in velocity space of solar wind protons at 1 AU.	47
Figure 4.3	Distribution function in velocity space of solar wind protons at 100 AU.	48

Figure 4.4	Distribution function in velocity space of solar wind protons at 160 AU.	49
Figure 4.5	Distribution function of hydrogen atoms in velocity space at $z = 20$ AU.	50
Figure 4.6	Distribution function of hydrogen atoms in velocity space at $z = 40$ AU.	51
Figure 4.7	Distribution function of hydrogen atoms in velocity space at $z = 60$ AU.	52
Figure 4.8	Distribution function of hydrogen atoms in velocity space at $z = 80$ AU.	53
Figure 4.9	Distribution function of hydrogen atoms in velocity space at $z = 100$ AU.	54
Figure 4.10	Distribution function of hydrogen atoms in velocity space at $z = 120$ AU.	55
Figure 4.11	Distribution function of hydrogen atoms in velocity space at $z = 140$ AU.	56
Figure 4.12	Distribution function of hydrogen atoms in velocity space at $z = 160$ AU.	57
Figure 4.13	Distribution function of hydrogen atoms in velocity space at $z = 180$ AU.	58
Figure 5.1	Summary of results for the hydrogen distribution in velocity space at various distances (20 to 160 AU) along the solar apex, inside and outside the solar system.	60

Figure 5.2	Results of the hydrogen distribution in velocity space at 10 AU (top) and 30 AU (bottom) from Boonma (2000). Contours indicate $f_H = 10^{-21.0}, 10^{-20.5}, 10^{-20.0}, 10^{-19.5},$ and $10^{-19.0} \text{ cm}^{-3}(\text{eV}/c)^{-3}$ for a charge exchange duration of 1 day.	62
Figure 5.3	Results of the hydrogen distribution in velocity space at 1 AU, 100 AU, 200 AU, and 300 AU from Müller et al. (2000) ..	64
Figure A.1	The elastic, momentum transfer, viscosity, and charge transfer cross sections for proton-hydrogen charge exchange. We performed fits to the charge transfer cross section (simulation results from Schultz et al. 1995).	73
Figure B.1	Graph of proton density, velocity, and temperature versus distance from the Sun (z).	82

Chapter 1

Introduction

The study of our solar system in the past thousands of years had yielded many important discoveries. In the past, many things about the solar system environment were not clearly understood. Nowadays, we can study many phenomena about the Sun and the solar system better than in the past with data from spacecraft, satellites, space telescopes, etc. that were developed by scientists. In this work we focus on effects in the inner and outer solar system (the heliosphere; see Figure 1.1). We consider the interstellar wind (mostly neutral hydrogen) that flows into our solar system, which collides with the solar wind (almost entirely a charged plasma of protons and electrons) flowing out from the Sun. More details about the heliosphere's structure are presented in Chapter 2.

The study of the interaction of the interstellar wind (interstellar medium) with the solar wind is a basic topic in heliospheric physics research. At present, many research groups are working on this problem. Research to date has not completely explained this interaction, because of the complexity of the Sun and environment of the solar system.

The overall size and structure of the heliosphere result from the interaction of the interstellar wind (mostly hydrogen atoms with low velocity) and

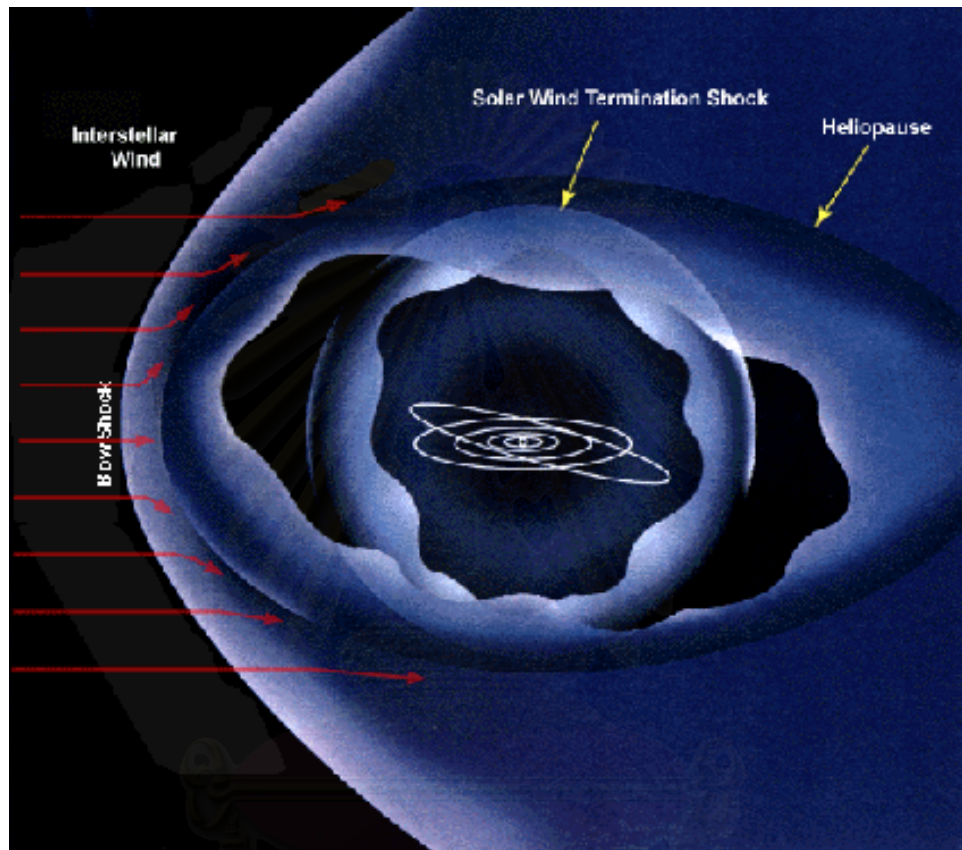


Figure 1.1: Schematic model of the heliosphere (bordered by the heliopause).

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

solar wind (with a mass flow dominated by proton of high, supersonic velocity). Our model calculates the distribution function of hydrogen from the interstellar wind which interacts with solar wind protons at each distance along the solar apex, i.e., the direction to the left of the Sun in Figure 1.1, toward the incoming interstellar wind. In this work we use numerical modeling as a tool to calculate the important physical effects.

The aims of this work are:

1. To study in detail the effect of the distribution of hydrogen atoms from the interstellar medium that flow into our solar system and their interaction with solar wind protons.
2. Calculate the distribution of hydrogen atoms along the solar apex in velocity space, $f_H(z, \vec{v}, t)$, and interpret the results physically.

We present results that are new and unique when compared with similar research in this field by research groups around the world. Our research work can explain the distribution function in velocity space in greater detail than that of other groups (e.g., Zank et al. 1999, Müller et al. 1999, 2000).

Advantages of our work compared with of other research groups can be explained as follows:

- We use grid points in two-dimensional velocity space which show distinct details for the velocity distribution function along the solar apex.
- Even though we consider the position in one-dimension along the solar apex from 1 to 200 AU, we use fine spatial position steps. That gives detailed results for each position.

The other chapters in this thesis are organized as follows: The basic knowledge, theoretical model and mathematical formulation are given in Chapter 2. The numerical model for solving the mathematical formulation is described in Chapter 3. The results of numerical calculations are presented in detail in Chapter 4. The discussion and conclusions of this work are presented in Chapter 5.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Chapter 2

Theoretical Background

The environment around the solar system is matter and gas of the local interstellar medium. When observed on the large scale of our galaxy, it is found that the Sun and solar system orbit around the center of the galaxy (see Figure 2.1). Because of the Sun's movement through the interstellar medium, it seems that matter flows toward the solar system. This can be called the local interstellar wind. The physics of these phenomena is the field of heliospheric physics, which can be used to explain and understand the solar system and its environment.

2.1 Space Physics and Heliospheric Physics

The region called the heliosphere includes space influenced by the Sun and solar wind. The heliosphere and solar system are in some sense synonymous. When studying the structure of the solar system on a large scale, it is found to be quite complex and full of detail.

2.1.1 The solar wind

The solar wind is the plasma of charged particles (protons, electrons, and heavier ions) coming out of the Sun in all directions at very high velocities from 300 to

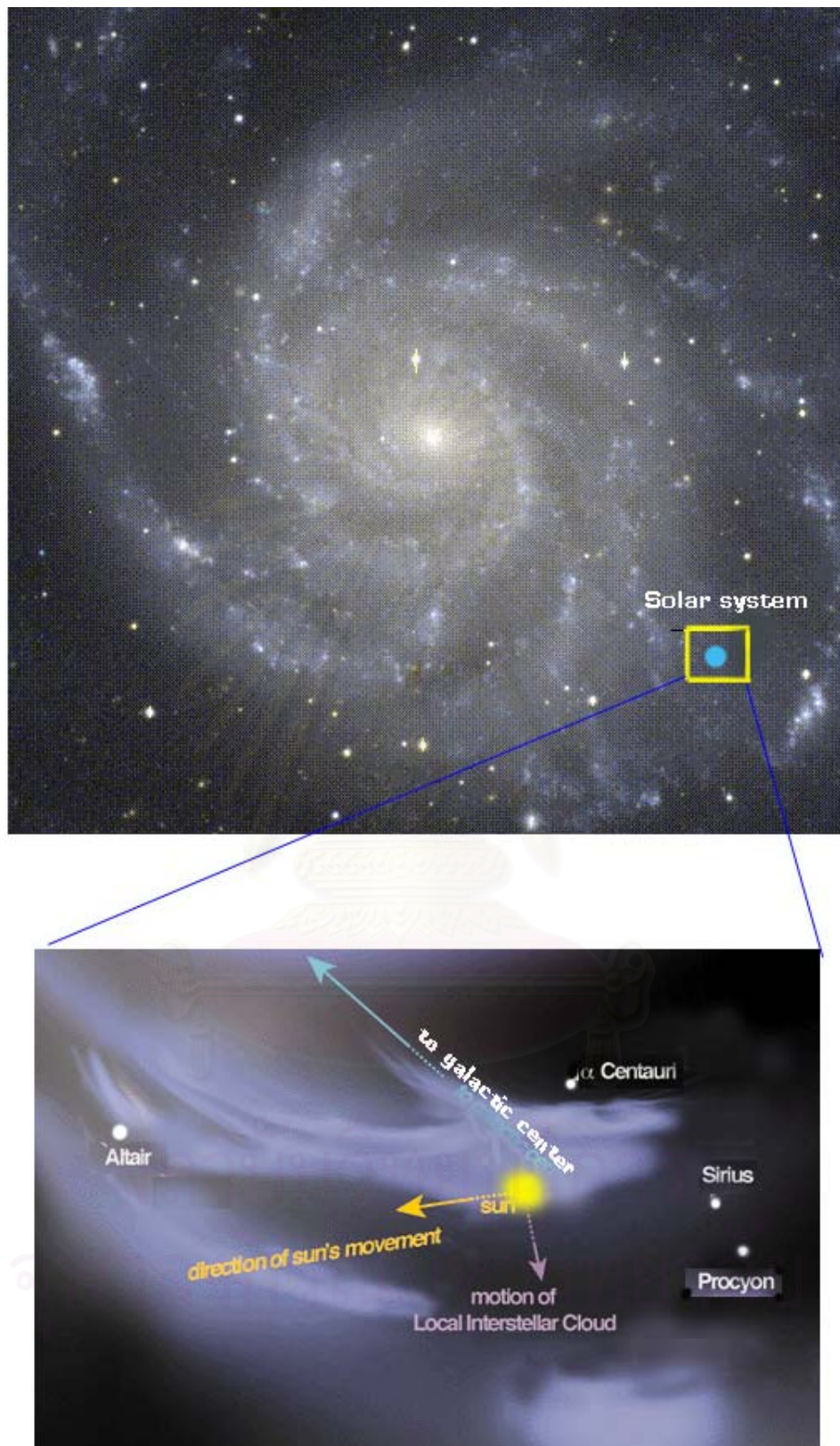


Figure 2.1: Schematic of the galaxy and the part of galaxy within 10 light-years of the Sun.

800 km/s (see in Figure 2.2). The average equatorial velocity is about 400 km/s or about 1.4 million kilometers per hour. The composition of the solar wind reflects the composition of the solar corona, modified by solar wind processes. Anyway, the exact mechanism of solar wind formation is not known in detail (NASA, 1997).

At 1 AU the average speed of the solar wind is about 400 km/s, and the average density is about 5 protons/cm³ with large variations (Frisch 2000, Zank 1999, Cravens 1997). When the solar wind expands in space far from the Sun, its density decreases as the inverse square of its distance from the Sun. At some far distance from the Sun (in a region known as the heliopause), the solar wind interacts with plasmas of the local interstellar medium and the solar wind slows down from nearly 400 km/s to perhaps 20 km/s. The location of this transition region is unknown at the present time.

2.1.2 The interstellar wind

The first discovery of interstellar matter within the solar system was made in the 1960s by a spacecraft observing the Earth's geocorona, a layer of neutral hydrogen atoms that forms in the outermost part of the planet's atmosphere. The spacecraft detected a weak fluorescent glow of Lyman-alpha (Ly α) ultraviolet radiation. (A Ly α photon is emitted when an electron in a neutral hydrogen atom falls from the first excited energy level to the ground state.)

From this discovery in 1960, many scientists have studied about the interstellar matter that flows into the solar system, mostly comprising neutral hydrogen atoms. The interstellar medium can be observed by **electromagnetic radiation** as follows:

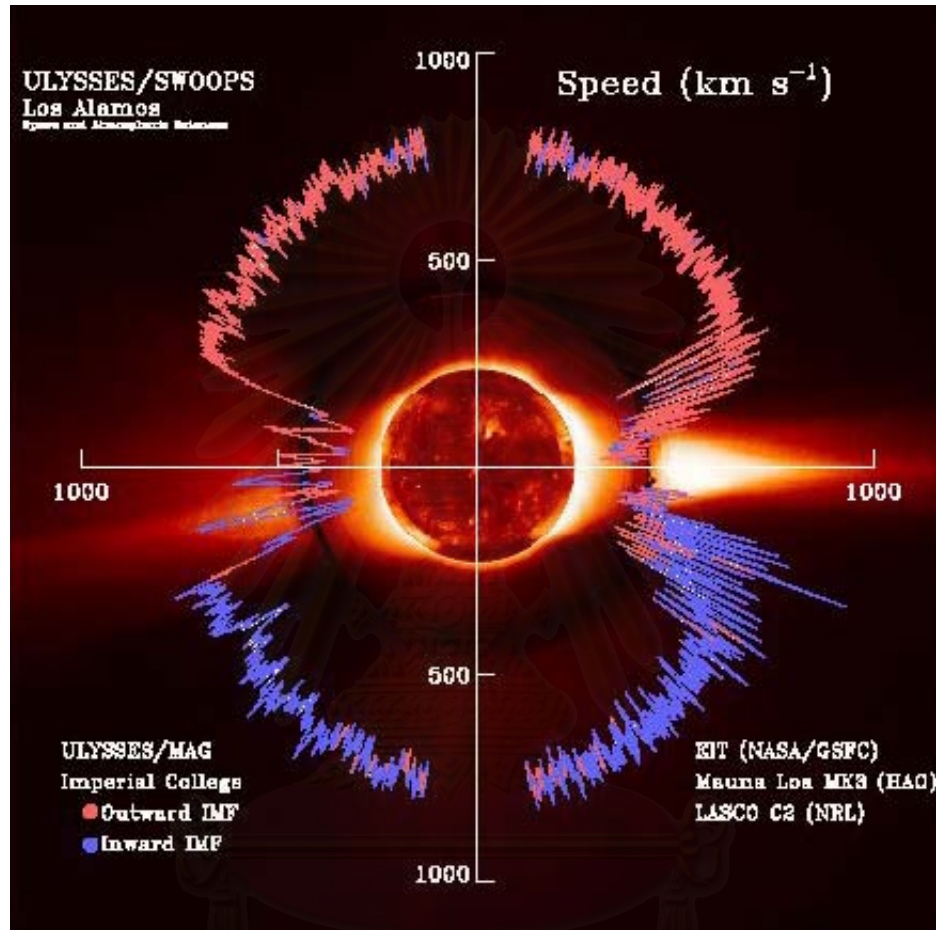


Figure 2.2: The solar wind velocity distribution around the Sun from the ULYSSES spacecraft.

☒ RADIO: Thermal radiation, bremsstrahlung and non-thermal radiation by dust.

☒ INFRARED: Thermal emission of dust, line spectra of atoms and molecules.

☒ VISIBLE: Extinction of starlight, absorption/emission lines of nebulae.

☒ ULTRAVIOLET: Absorption lines, neutral gas and fluorescent emission by highly ionized gas.

☒ X-RAYS: Emission and absorption of the hot interstellar medium.

☒ γ -RAYS: Cosmic ray interactions with interstellar nuclei or cosmic ray bremsstrahlung emission.

In the solar system, the effects from solar radiation and the solar wind almost completely ionize the interstellar hydrogen atoms within several AU of the Sun, partly by photoionization (see Figure 2.3) and partly by charge exchange interactions with the solar wind.

2.2 The Boltzmann Transport Equation

To study the flow of interstellar neutrals into the solar system and their interactions with the solar wind protons, we use the mathematical formulation of the Boltzmann transport equation to describe in detail the phase distribution of interstellar neutral hydrogen inside and outside the solar system.

Consider the motion of particle of mass m . We are not interested in the motion of each particle in detail; instead we are interested in the phase space distribution function $f(\vec{r}, \vec{v}, t)$,

$$f(\vec{r}, \vec{v}, t) = \frac{d^6 N}{d^3 r d^3 v}. \quad (2.1)$$

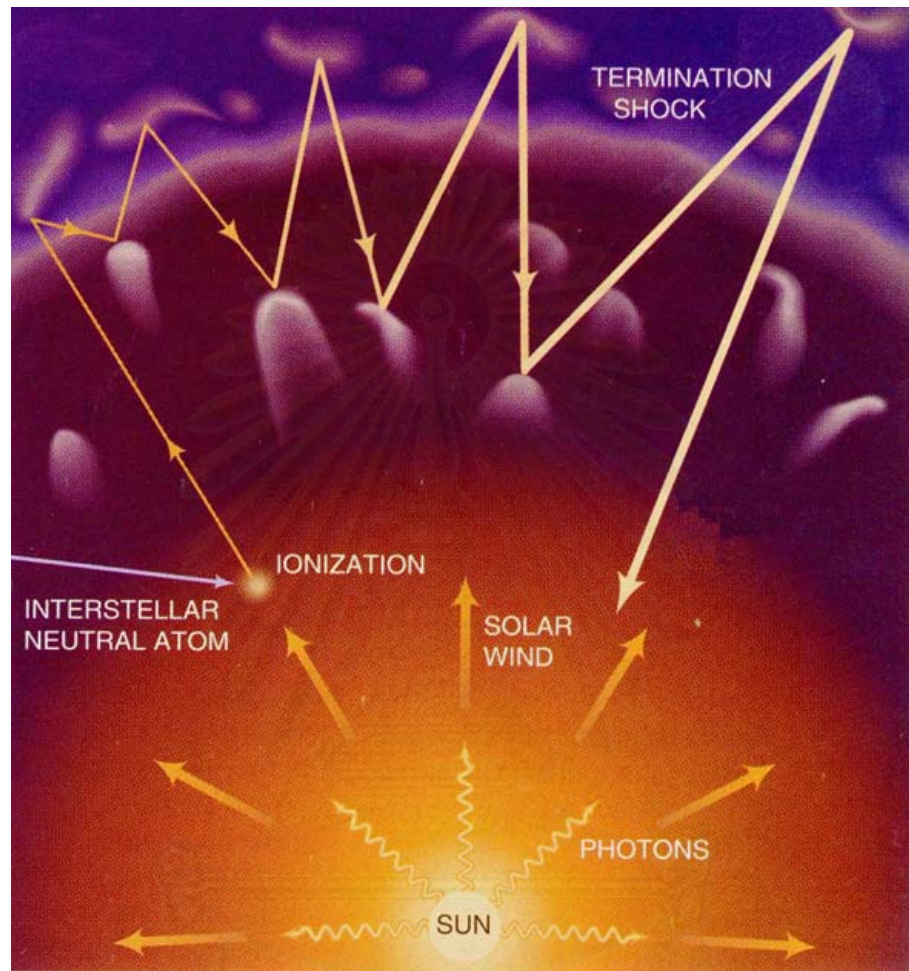


Figure 2.3: The photoionization process for neutral hydrogen atoms. After ionization, the so-called pickup ions can travel to the termination shock, undergo scattering, and be accelerated to become so-called anomalous cosmic rays.

จุฬาลงกรณ์มหาวิทยาลัย

where phase space is a six-dimensional space with coordinates (x, y, z, v_x, v_y, v_z) .

We define $f(\vec{r}, \vec{v}, t)d^3rd^3v$ as the expected number of particles at time t with positions within a volume element d^3r about \vec{r} and velocities within a velocity space element d^3v about \vec{v} .

The distribution function $f(\vec{r}, \vec{v}, t)$ is a function of seven independent variables in the form $f(x, y, z, v_x, v_y, v_z, t)$. The total time derivative of f is

$$\begin{aligned} \frac{df}{dt} &= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} \frac{dx}{dt} + \frac{\partial f}{\partial y} \frac{dy}{dt} + \frac{\partial f}{\partial z} \frac{dz}{dt} + \frac{\partial f}{\partial v_x} \frac{dv_x}{dt} + \frac{\partial f}{\partial v_y} \frac{dv_y}{dt} + \frac{\partial f}{\partial v_z} \frac{dv_z}{dt} \\ \frac{df}{dt} &= \frac{\partial f}{\partial t} + \dot{x} \frac{\partial f}{\partial x} + \dot{y} \frac{\partial f}{\partial y} + \dot{z} \frac{\partial f}{\partial z} + \dot{v}_x \frac{\partial f}{\partial v_x} + \dot{v}_y \frac{\partial f}{\partial v_y} + \dot{v}_z \frac{\partial f}{\partial v_z}, \end{aligned} \quad (2.2)$$

which can be written in the form

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \dot{\vec{r}} \cdot \frac{\partial f}{\partial \vec{r}} + \dot{\vec{v}} \cdot \frac{\partial f}{\partial \vec{v}}. \quad (2.3)$$

Consider the evolution of f , describing the physical behavior of particles in (\vec{r}, \vec{v}) . In the absence of collisions, each particle's orbit would describe a continuous curve and the function $f(\vec{r}, \vec{v}, t)$ would obey the continuity equation (Liouville equation) in this form:

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \dot{\vec{r}} \cdot \frac{\partial f}{\partial \vec{r}} + \dot{\vec{v}} \cdot \frac{\partial f}{\partial \vec{v}} = 0 \quad (2.4)$$

2.2.1 Streaming process

From equation (2.4) we derive the collisionless Boltzmann transport equation, which we write in the form

$$\begin{aligned} \frac{\partial f}{\partial t} &= -\dot{\vec{r}} \cdot \frac{\partial f}{\partial \vec{r}} - \dot{\vec{v}} \cdot \frac{\partial f}{\partial \vec{v}} \\ &= -\vec{v} \cdot \frac{\partial f}{\partial \vec{r}} - \vec{a} \cdot \frac{\partial f}{\partial \vec{v}} \\ &= -\vec{v} \cdot \frac{\partial f}{\partial \vec{r}} - \frac{\vec{F}}{m} \cdot \frac{\partial f}{\partial \vec{v}}. \end{aligned}$$

We consider that no external force acts on particles in the system ($\vec{F} = 0$); therefore we get an equation in the form

$$\frac{\partial f}{\partial t} = -\vec{v} \cdot \frac{\partial f}{\partial \vec{r}}. \quad (2.5)$$

The term $-\vec{v} \cdot \partial f / \partial \vec{r}$ in equation (2.5) represents the streaming process of particles in the system which move with velocity \vec{v} . We can call equation (2.5) an advection equation that describes the flow of particles in the system.

2.2.2 Collision process

The physical processes of particles moving in the system include their collisions. Therefore equation (2.5) should have a collision term in the form $(\partial f / \partial t)_{coll}$, in order to describe a real physical system:

$$\frac{\partial f}{\partial t} = -\vec{v} \frac{\partial f}{\partial \vec{r}} + \left(\frac{\partial f}{\partial t} \right)_{coll} \quad (2.6)$$

(Huang 1987). The calculation starts with

$$R = n\sigma v_{rel} = n\sigma |\vec{v} - \vec{v}_1|,$$

where the collision rate, R , depends on the number density of target particles, n , the cross section, σ , and the relative velocity of the target particle and projectile particle. The number density of particles is given by

$$n = \frac{N}{V} = \int f(\vec{r}, \vec{v}, t) d^3v,$$

and

$$\sigma = \int \frac{d\sigma}{d\Omega'} d\Omega',$$

where $d\sigma/d\Omega'$ is the differential cross section and $d\Omega'$ is an element of solid angle in the center of mass frame.

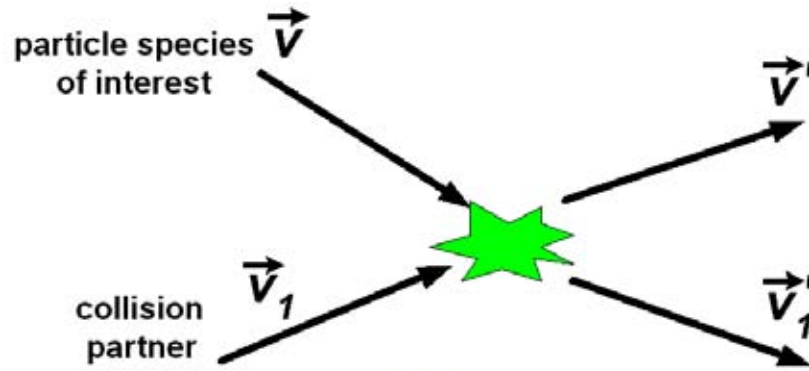


Figure 2.4: The collision of two particles.

Consider the collision of two particles of the system which have the same position but have different velocities as in Figure 2.4. Here

\vec{v} is the velocity of the particle of interest,

\vec{v}_1 is the velocity of the particle collision partner,

\vec{v}' is the velocity of the particle of interest after the collision, and

\vec{v}'_1 is the velocity of the particle collision partner after the collision.

For collisions of two particles we can write the change in f in the form

$$\left(\frac{\partial f}{\partial t}\right)_{coll} d^3r d^3v = -C_{out} + C_{in}, \quad (2.7)$$

where C_{out} and C_{in} are rates at which particles leave and enter the elementary volume $d^3r d^3v$ of interest in phase space due to collisions.

Consider a beam of particles of number density n_1 and velocity \vec{v}_1 colliding with another beam of particles of number density n and velocity \vec{v} . A particle in the second beam experiences a flux $I = n_1 |\vec{v} - \vec{v}_1|$ of particles from the first beam. We consider the number of collisions per unit volume per unit time, δn_c , which deflect particles from the other beam into $d\Omega'$, an element of solid angle in the center of mass frame. This number is proportional to the number density

n of particles in the other beam, proportional to the flux I these particles are exposed to and proportional to the solid angle. We can write

$$\delta n_c = n n_1 |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega'. \quad (2.8)$$

Consider collisions between the stream of particles having a velocity within d^3v of \vec{v} and the stream of particles having velocities within d^3v_1 of \vec{v}_1 . The first particles make up a beam with number density $n = f(\vec{r}, \vec{v}, t) d^3v$ and velocity \vec{v} , whereas the partner particles constitute a beam with number density $n_1 = f(\vec{r}, \vec{v}_1, t) d^3v_1$ and velocity \vec{v}_1 , so the collision rate is

$$\delta n_c = f(\vec{r}, \vec{v}, t) d^3v f(\vec{r}, \vec{v}_1, t) d^3v_1 |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega'. \quad (2.9)$$

Therefore we write C_{out} in the form

$$C_{out} = \int_{\vec{v}_1} \int_{\Omega'} f(\vec{r}, \vec{v}, t) f(\vec{r}, \vec{v}_1, t) |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega' (d^3v d^3v_1) d^3r. \quad (2.10)$$

To evaluate C_{in} , we consider the reverse collision between particles of velocity within d^3v' of \vec{v}' and particles of velocity within $d^3v'_1$ of \vec{v}'_1 such that velocities after the collision lie within d^3v of \vec{v} and d^3v_1 of \vec{v}_1 . As in (2.9), the number of such collisions per unit volume per unit time is

$$\delta n'_c = f(\vec{r}, \vec{v}', t) d^3v' f(\vec{r}, \vec{v}'_1, t) d^3v'_1 |\vec{v}' - \vec{v}'_1| \frac{d\sigma}{d\Omega'} d\Omega'. \quad (2.11)$$

If we consider only elastic collisions we have $|\vec{v} - \vec{v}_1| = |\vec{v}' - \vec{v}'_1|$ and the phase space of the forward and reverse collisions are equal: $d^3v d^3v_1 = d^3v' d^3v'_1$. Then we get

$$\delta n'_c = f(\vec{r}, \vec{v}', t) d^3v' f(\vec{r}, \vec{v}'_1, t) d^3v_1 |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega'. \quad (2.12)$$

and

$$C_{in} = \int_{\vec{v}_1} \int_{\Omega'} f(\vec{r}, \vec{v}', t) f(\vec{r}, \vec{v}'_1, t) |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega' (d^3v d^3v_1) d^3r. \quad (2.13)$$

From (2.10) and (2.13) substituted into (2.7),

$$\begin{aligned}
\left(\frac{\partial f}{\partial t}\right)_{coll} d^3 r d^3 v &= \int_{\vec{v}_1} \int_{\Omega'} f(\vec{r}, \vec{v}, t) f(\vec{r}, \vec{v}_1, t) |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega' d^3 v d^3 v_1 d^3 r \\
&- \int_{\vec{v}_1} \int_{\Omega'} f(\vec{r}, \vec{v}', t) f(\vec{r}, \vec{v}'_1, t) |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega' d^3 v d^3 v_1 d^3 r \\
\left(\frac{\partial f}{\partial t}\right)_{coll} &= \int_{\vec{v}_1} \int_{\Omega'} f(\vec{r}, \vec{v}, t) f(\vec{r}, \vec{v}_1, t) |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega' d^3 v_1 \\
&- \int_{\vec{v}_1} \int_{\Omega'} f(\vec{r}, \vec{v}', t) f(\vec{r}, \vec{v}'_1, t) |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} d\Omega' d^3 v_1.
\end{aligned}$$

For the distribution function of particles, we write $f = f(\vec{r}, \vec{v}, t)$ and $f_1 = f(\vec{r}, \vec{v}_1, t)$ (before the collision), and $f' = f(\vec{r}, \vec{v}', t)$ and $f'_1 = f(\vec{r}, \vec{v}'_1, t)$ (after the collision). Then

$$\left(\frac{\partial f}{\partial t}\right)_{coll} = \int_{\vec{v}_1} \int_{\Omega'} |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} (f' f'_1 - f f_1) d\Omega' d^3 v_1. \quad (2.14)$$

From (2.6) and (2.14) we get

$$\begin{aligned}
\frac{\partial f}{\partial t} = & \quad -\vec{v} \cdot \frac{\partial f}{\partial \vec{r}} \quad + \int_{\vec{v}_1} \int_{\Omega'} |\vec{v}_1 - \vec{v}| \frac{d\sigma}{d\Omega'} (f' f'_1 - f f_1) d\Omega' d^3 v_1. \quad (2.15) \\
& \text{(streaming term)} \qquad \qquad \qquad \text{(collision term)}
\end{aligned}$$

2.3 Ionization Process

The interstellar medium that flows into the solar system interacts with photons from the Sun by the photo-ionization process (see Figure 2.3). The photo-ionization by solar extreme ultraviolet (EUV) radiation has a direct effect on the number density of interstellar neutral atoms in the inner solar system.

For interstellar hydrogen atoms the rate at which neutral hydrogen is photo-ionized is inversely proportional to the square of the heliocentric distance r :

$$\nu = \nu_0 \left(\frac{r_0^2}{r^2} \right), \quad (2.16)$$

where ν is the rate of photo-ionization per hydrogen atom at any distance r , and $\nu_0 = 0.9 \times 10^{-7} \text{ s}^{-1}$ is the rate of photo-ionization per hydrogen atom at $r_0 = 1$ AU (Möbius, 1993; Whang 1996, 1998).

2.4 Maxwellian Distribution Function

The phase space distribution of particles in thermal equilibrium, in a fluid at rest, is given by a Maxwellian distribution. For particles of type i , this is given by the expression

$$f_i(\vec{r}, \vec{v}, t) = n_i(\vec{r}, t) \left(\frac{m_i}{2\pi k_B T_i} \right)^{3/2} \exp \left[-\frac{\frac{1}{2} m_i v^2}{k_B T_i} \right], \quad (2.17)$$

where n_i is the number density as a function of position and time which can be found by integration of f_i over all velocity space, $v^2 = v_x^2 + v_y^2 + v_z^2$, the temperature T_i is a function of position, $T_i = T_i(\vec{r})$, and m_i is the mass of particles of type i . The distribution function f_i in equation (2.17) falls off more rapidly for low temperature than for high temperature. The probability of finding a particle of type i decreases exponentially with increasing v^2 for a Maxwellian distribution (or equivalently with increasing kinetic energy).

The Maxwellian distribution is isotropic; that is, f_i depends on the magnitude of the velocity vector \vec{v} and not on its direction. A distribution that is closely related to the Maxwellian is the “drifting Maxwellian” distribution for which the particles have an additional velocity, \vec{v}_0 , in some direction, which is

the average or bulk velocity. In this case, the distribution function looks like the Maxwellian distribution in equation (2.17) with v^2 replaced by $|\vec{v} - \vec{v}_0|^2$.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Chapter 3

Numerical Method

Numerical methods for computational physics have the advantage of allowing us to study physical systems that are complicated and cannot be fully examined by theory or experiment. Computational physics is now widely accepted as a third branch of physics in addition to theoretical and experimental physics. In fact, the majority of physics research projects use numerical methods instead of or in addition to traditional experimental and analytic methods. Using simulations of a physical system can have the advantage of approaching the core of the problem, mathematical formulation, and nature of the system of interest. In this chapter we will explain the mathematical formulation of our problem and the basic methods for we use for our numerical calculations.

3.1 Finite Differencing

There is a wide range of methods for solving problems involving differential equations. An important class of numerical methods are the finite difference methods, using differencing formulae in place of a differential equation. A particular advantage of finite difference methods is that they are often closely related to the

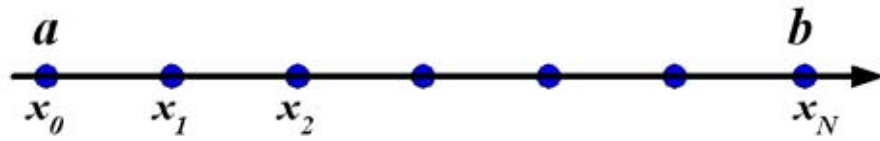


Figure 3.1: Discretization of $x \in [a, b]$.

physical processes. We can use a Taylor series approach to verify and evaluate the error of difference formulae.

Consider a problem in one independent variable, x . We divide the region of interest along the x -axis by using N points (see Figure 3.1). For $x \in [a, b]$, the continuous spatial domain can be discretized into an equally spaced grid of discrete points as illustrated in Figure 3.1, where the subscript i denotes a particular spatial location, with f_i as an approximation to $f(x_i)$ (Hoffman 1993).

Difference formulae for $f'(x_i)$, $f''(x_i)$, etc. can be developed from the Taylor series of $f(x)$ in the form

$$f(x) = f(x_0) + f'(x_0)\Delta x + \frac{1}{2}f''(x_0)\Delta x^2 + \dots + \frac{1}{n!}f^{(n)}(x_0)\Delta x^n + \dots \quad (3.18)$$

For a function of more than one variable, such as $f(x, t)$, difference formulae can be developed from the Taylor series for the function $f(x, t)$:

$$\begin{aligned} f(x, t) = & f(x_0, t_0) + f_x(x_0, t_0)\Delta x + f_t(x_0, t_0)\Delta t \\ & + \frac{1}{2}[f_{xx}(x_0, t_0)\Delta x^2 + 2f_{xt}(x_0, t_0)\Delta x\Delta t + f_{tt}(x_0, t_0)\Delta t^2] + \dots \\ & + \frac{1}{n!}[f_{(n)x}(x_0, t_0)\Delta x^n + \dots + f_{(n)t}(x_0, t_0)\Delta t^n] + \dots, \end{aligned} \quad (3.19)$$

where $f_{(n)x}$ denotes $\partial^n f / \partial x^n$. The continuous domain $D(x, t)$ can be discretized into an orthogonal and equally spaced grid of discrete points (x_i, t_n) (given i as the spatial index and n as the time index). Values such as f_i^n , which approximates $f(x_i, t_n)$, can be combined to obtain difference formulae for f_x , f_t , etc.

If we want to derive differencing formulae in multidimensions, we can start by considering those in one dimension. Indeed, when considering the partial derivative of $f(x, t)$ with respect to x , and given $t = t_0 = \text{constant}$, (3.2) becomes

$$\begin{aligned} f(x, t_0) &= f(x_0, t_0) + f_x(x_0, t_0)\Delta x + \frac{1}{2}f_{xx}(x_0, t_0)\Delta x^2 + \dots \\ &\quad + \frac{1}{n!}f_{(n)x}(x_0, t_0)\Delta x^n + \dots \end{aligned} \quad (3.20)$$

Equation (3.3) is identical in form to equation (3.1), where $f'(x_0)$ corresponds to $f_x(x_0)$, etc. The partial derivative $f_x(x_0, t_0)$ of the function $f(x, t)$ will be obtained from (3.3) in exactly the same manner as the total derivative $f'(x_0)$ of the function $f(x)$ is obtained from equation (3.1). Since equations (3.1) and (3.3) are identical in form, the difference formulae for $f'(x_0)$ and $f_x(x_0, t_0)$ are identical if the same discrete grid points are used to develop the formulae. Consequently, difference formulae for partial derivatives of a function of several variables can be derived from the Taylor series for a function of a single variable.

Similarly, the partial derivative of $f(x, t)$ with respect to t (when $x = x_0 = \text{constant}$) can be obtained from

$$\begin{aligned} f(x_0, t) &= f(x_0, t_0) + f_t(x_0, t_0)\Delta t + \frac{1}{2}f_{tt}(x_0, t_0)\Delta t^2 + \dots \\ &\quad + \frac{1}{n!}f_{(n)t}(x_0, t_0)\Delta t^n + \dots \end{aligned} \quad (3.21)$$

Returning to equation (3.1), this can be written as a finite Taylor series in $f(x)$:

$$\begin{aligned} f(x) &= f(x_0) + f_x(x_0)\Delta x + \frac{1}{2}f_{xx}(x_0)\Delta x^2 + \dots + \frac{1}{n!}f^{(n)x}(x_0)\Delta x^n \\ &\quad + \frac{1}{(n+1)!}f^{(n+1)x}(\xi)\Delta x^{n+1}, \end{aligned} \quad (3.22)$$

where we give the final term of (3.5) as the remainder term,

$$R_{n+1} = \frac{1}{(n+1)!}f^{(n+1)x}(\xi)\Delta x^{n+1}$$

for some ξ such that $x_0 \leq \xi \leq x_0 + \Delta x$.

Therefore, the error incurred by truncating the infinite Taylor series after the n th derivative is exactly the remainder term of the n^{th} -order Taylor formula. Truncating the Taylor series is equivalent to dropping the R_{n+1} term of the finite Taylor series. The finite difference approximation of exact derivatives can be obtained by solving exactly from either the infinite or finite Taylor series, and then either truncating the Taylor series or dropping the remainder term R_{n+1} , respectively.

The terms that are truncated from the infinite Taylor series, which are identical to the R_{n+1} term of the Taylor formula, are called the “truncation error” of the finite difference approximation of the exact derivative. An important point is that the truncation error approaches to zero as $\Delta x \rightarrow 0$. The order of the truncation error, i.e., the R_{n+1} term, can be written $O(\Delta x^{(n+1)})$, where $O(\Delta x^{(n+1)})$ means that the leading-order term is of order $(\Delta x)^{(n+1)}$.

We develop the formulation of a finite difference equation, which is adapted from a differential equation by differencing formulae.

We start in one dimension by considering the equally spaced discrete finite difference grid shown in Figure 3.1, using point i as the base point and writing the Taylor series of f_{i+1} and f_{i-1} as

$$f_{i+1} = f_i + f_x(x_i)\Delta x + \frac{1}{2}f_{xx}(x_i)\Delta x^2 + \frac{1}{6}f_{xxx}(x_i)\Delta x^3 + \frac{1}{24}f_{xxxx}(x_i)\Delta x^4 + \dots \quad (3.23)$$

$$f_{i-1} = f_i - f_x(x_i)\Delta x + \frac{1}{2}f_{xx}(x_i)\Delta x^2 - \frac{1}{6}f_{xxx}(x_i)\Delta x^3 + \frac{1}{24}f_{xxxx}(x_i)\Delta x^4 + \dots \quad (3.24)$$

Subtracting f_{i-1} from f_{i+1} ,

$$f_{i+1} - f_{i-1} = 2f_x(x_i)\Delta x + \frac{1}{3}f_{xxx}(x_i)\Delta x^3 + \dots, \quad (3.25)$$

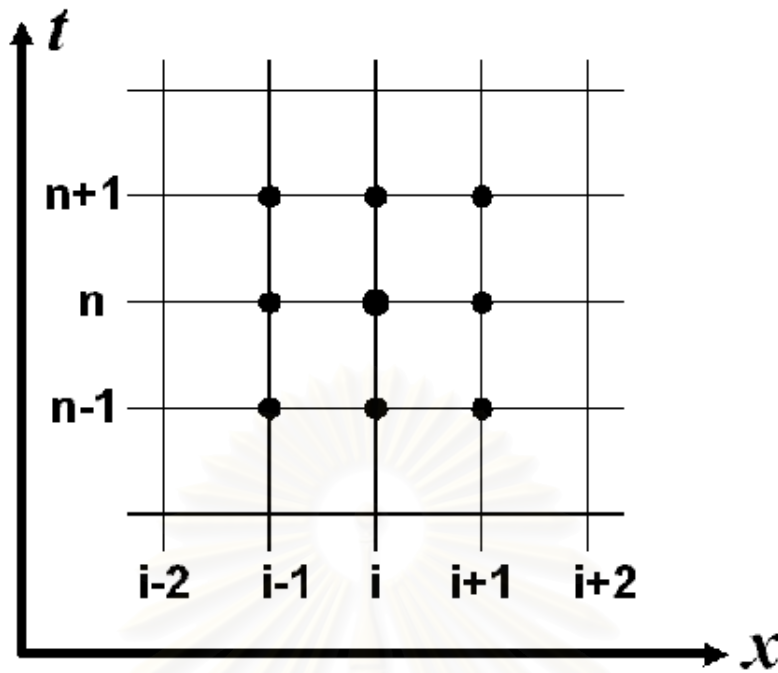


Figure 3.2: Discretization in two-dimensional $x - t$ space.

we can derive a formula for $f_x(x_i)$, converting the $f_{xxx}(x_i)$ term to be a “remainder term”:

$$f_x(x_i) = \frac{f_{i+1} - f_{i-1}}{2\Delta x} - \frac{1}{6}f_{xxx}(\xi)\Delta x^2, \quad (3.26)$$

where $x_{i-1} \leq \xi \leq x_{i+1}$. Equation (3.9) gives a good approximation for $f_x(x_i)$ if the remainder term is dropped:

$$f_x(x_i) \approx \frac{f_{i+1} - f_{i-1}}{2\Delta x}. \quad (3.27)$$

Adding f_{i+1} and f_{i-1} from (3.6) and (3.7) gives

$$f_{i+1} + f_{i-1} = 2f_i + f_{xx}(x_i)\Delta x^2 + \frac{1}{12}f_{xxxx}(x_i)\Delta x^4 + \dots, \quad (3.28)$$

where here the $f_{xxxx}(x_i)$ term is taken to be the “remainder term”:

$$f_{xx}(x_i) = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} - \frac{1}{2}f_{xxxx}(\xi)\Delta x^2 \quad (3.29)$$

or

$$f_{xx}(x_i) \approx \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2}. \quad (3.30)$$

Equations (3.10) and (3.13) are central-difference formulae, which are inherently more accurate than corresponding one-sided difference formulae.

When considering a problem with more than one independent variable, e.g., in terms of a function of two variables, $f(x, t)$, as shown in Figure 3.2, we approximate the true solution $f(x_i, t_n)$ by f_i^n . Difference formulae for t derivatives can be developed similarly to those for x .

3.2 Operator Splitting

Partial differential equations can sometimes have many variables, or be complicated, large, or hard to solve analytically. Even when using a numerical method such as finite differencing, it may not be easy to solve. We can solve the finite difference equations for a large problem in multiple variables by sequentially updating for each term or set of terms in the differential equation. This technique for solving step by step is the “operator splitting” technique.

The operator splitting technique (Press et al. 1992) is also called time splitting or the method of fractional steps. Suppose we have a partial differential equation for a function $u(t, x_1, x_2, \dots, x_k)$ in the form

$$\frac{\partial u}{\partial t} = Lu, \quad (3.31)$$

where L is a differential operator. While L is not necessarily linear, suppose that it can at least be written as a linear sum of k variables, which act additively on u ,

$$Lu = L_1u + L_2u + \dots + L_ku,$$

so that

$$\frac{\partial u}{\partial t} = L_1 u + L_2 u + \dots + L_k u. \quad (3.32)$$

Finally, suppose that for each L_i , we already know a differencing scheme for updating the variable u from timestep n to timestep $n + 1$, valid if that piece of the operator were the only one on the right-hand side. We will write such updating symbolically as

$$\begin{aligned} \frac{\partial u}{\partial t} = L_1 u &\quad \rightarrow \quad u^{n+1} = u_1(u^n, \Delta t) \\ \frac{\partial u}{\partial t} = L_2 u &\quad \rightarrow \quad u^{n+1} = u_2(u^n, \Delta t) \\ &\quad \dots \\ \frac{\partial u}{\partial t} = L_k u &\quad \rightarrow \quad u^{n+1} = u_k(u^n, \Delta t). \end{aligned} \quad (3.33)$$

One form of operator splitting would be to get from u^n to u^{n+1} for the full equation (3.15) by the following sequence of updating:

$$\begin{aligned} u^{n+(1/k)} &= u_1(u^n, \Delta t) \\ u^{n+(2/k)} &= u_2(u^{n+(1/k)}, \Delta t) \\ &\quad \dots \\ u^{n+1} &= u_k(u^{n+(k-1)/k}, \Delta t). \end{aligned} \quad (3.34)$$

For example, a numerical method including operator splitting was used (Ruffolo 1995) to solve a problem with a transport equation for the function $f(t, \mu, z, p)$. The procedure can be physically interpreted as having particles alternately undergo changes in μ , p , and z during each time step. As the time step is shortened, this sequence provides a more accurate approximation. In that work, the procedure for updating $f(t, \mu, z, p)$ from t to $t + \Delta t$ is as follows:

1. Update f for μ -changing processes over a time $\Delta t/2$.
2. Update f for z -changing processes (streaming) over a time Δt .
3. Update f for p -changing processes over a time Δt .
4. Update f for μ -changing processes over a time $\Delta t/2$.

Note that μ -changing processes are treated for $\Delta t/2$ each at the beginning and end for better (second-order) accuracy.

3.3 Advection Process and TVD Method

The advection term of the transport equation (2.5) is

$$\frac{\partial f}{\partial t} = -\vec{v} \cdot \frac{\partial f}{\partial \vec{r}}, \quad (3.35)$$

or in one dimension,

$$\frac{\partial f}{\partial t} = -v_z \frac{\partial f}{\partial z}. \quad (3.36)$$

This systematic, advection process can also be referred to as streaming. We employ a type of method for solving this part of the problem known as a “Total Variation Diminishing” (TVD) method (Harten 1983) as generalized by Nutaro et al. (2001). The advantages of this method are that it can accurately solve advection problems as well as avoiding artificial numerical diffusion of particles moving from one position to another.

The total variation (TV) of a function $f(x)$ is defined by

$$TV = \int \left| \frac{\partial f}{\partial x} \right| dx. \quad (3.37)$$

We can write this in discrete form as

$$TV(f) = \sum_i |f_{i+1} - f_i|, \quad (3.38)$$

where f_i approximates $f(x_i)$. Considering a function of x which depends on time t , too, we can approximate $f(x_i, t^n)$ by f_i^n , where i is the spatial index and n is the time index. Equation (3.21) at any time can be written as

$$TV(f^n) = \sum_i |f_{i+1}^n - f_i^n|. \quad (3.39)$$

The principal requirement of a total variation diminishing method is that

$$TV(f^{n+1}) \leq TV(f^n). \quad (3.40)$$

The motivation of this requirement is to avoid creating new maxima or minima of the function in each time step.

The TVD scheme that we employ is based on Roe's superbee limiter (Roe, 1983), which gives remarkably constant shape profiles for the linear advection equation. We define the Courant number as $\gamma = v\Delta t/\Delta z$, and in usual TVD methods $0 \leq \gamma \leq 1$. For our application, we want to be able to set $\gamma > 1$ for greater flexibility. Thus we use a developed generalization of a TVD method (Nutaro et al., 2001) to allow a general value of γ which can also vary with position, z .

In the generalized TVD technique, the advection of F in the z -direction first uses an integral number of steps g , where g is obtained by rounding γ downward. For example, if $\gamma=4.6$, then F is moved forward by 4 z -grid points and the remainder $\gamma' = \gamma - g = 4.6 - 4 = 0.6$. We see that $0 \leq \gamma' \leq 1$. Then by the usual TVD differencing,

$$F_l \leftarrow F_{l-g} - \frac{\Delta t}{\Delta z} S'_{l+\frac{1}{2}} + \frac{\Delta t}{\Delta z} S'_{l-\frac{1}{2}}, \quad (3.41)$$

where l is the index of the z -cell, and $S_{l+1/2}$ is the flux of particles from z -cell (l) to ($l+1$) due to γ' , which is calculated from

$$S_{l+\frac{1}{2}} = v'_{l+\frac{1}{2}} F_{l-g} + \frac{1}{2} v'_{l+\frac{1}{2}} (1 - \gamma'_{l+\frac{1}{2}}) (F_{l-g+1} - F_{l-g}) \phi_{l-g}, \quad (3.42)$$

where $\gamma'_{l+1/2} = \gamma'(z_l + \Delta z/2)$ and $v'_{l+1/2} = \gamma'_{l+1/2} \Delta z / \Delta t$, the index $l + \frac{1}{2}$ referring to the position of the cell boundary at $z_l + \Delta z/2$. Here ϕ_l is Roe's superbee limiter (Roe, 1983), given by

$$\phi_l = \begin{cases} 0 & r_l \leq 0 \\ 2r_l & 0 \leq r_l \leq 0.5 \\ 1 & 0.5 \leq r_l \leq 1 \\ r_l & 1 \leq r_l \leq 2 \\ 2 & r_l > 2. \end{cases}, \quad (3.43)$$

where r_l in equation (3.26) is

$$r_l = \frac{F_l - F_{l-1}}{F_{l+1} - F_l}. \quad (3.44)$$

The TVD method will sacrifice convergence speed (sometimes converging only to first order in Δz) in order to guarantee equation (3.25) that no new minima or maxima are introduced.

Next, we should consider a variation in v and hence γ as a function of z , in which case g itself can change from one grid point the next. To take this possibility into account, F over a time step Δt is updated by

$$F_l \leftarrow \sum_{m=l-g_-}^{m=l-g_+} F_m - \frac{\Delta t}{\Delta z} (S'_{l+\frac{1}{2}} - S'_{l-\frac{1}{2}}) \quad (3.45)$$

where g_+ is the rounded-down integer corresponding to $\gamma_{l+\frac{1}{2}}$ and g_- corresponds to $\gamma_{l-\frac{1}{2}}$. This formula is subject to the constraint that $g_+ \leq g_- + 1$, and if $g_+ = g_- + 1$ we interpret the sum to be zero. In practice that constraint can often be avoided by reducing Δt , and in physical situations where $v(z)$ is discontinuous a different technique is needed.

3.4 Interpolation Method

Interpolation, an important topic in numerical analysis, is the approximation of the value at a point of interest given a set of known values of any function, or

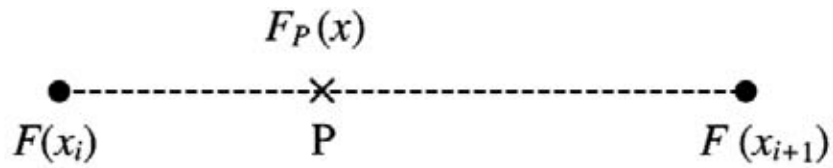


Figure 3.3: Linear interpolation in one dimension.

data from an experiment. Many famous mathematicians such as Gauss, Newton, Bessel, and Stirling are associated with procedures for interpolation. The need to interpolate began with the early studies of astronomy when the motion of heavy bodies was determined from periodic observations (Gerald and Wheatley, 1994).

3.4.1 Linear interpolation

Linear interpolation is the simplest method to determine the value at a point of interest between two points with known values of data or a function in one dimension (Figure 3.3).

From Figure 3.3 we have known values at two points, $F(x_i)$ and $F(x_{i+1})$. We can calculate an approximate value of $F(x)$ at the point P by the formula:

$$F_P(x) = \frac{x_{i+1} - x}{x_{i+1} - x_i} F(x_i) + \frac{x - x_i}{x_{i+1} - x_i} F(x_{i+1}). \quad (3.46)$$

We can also write this in the form

$$F_P(x) = (1 - f_x)F(x_i) + f_x F(x_{i+1}), \quad (3.47)$$

where f_x is the fractional distance of P from x_i to x_{i+1} ,

$$f_x = \frac{x - x_i}{x_{i+1} - x_i},$$

and the value of f_x should be between 0 and 1 for interpolation.

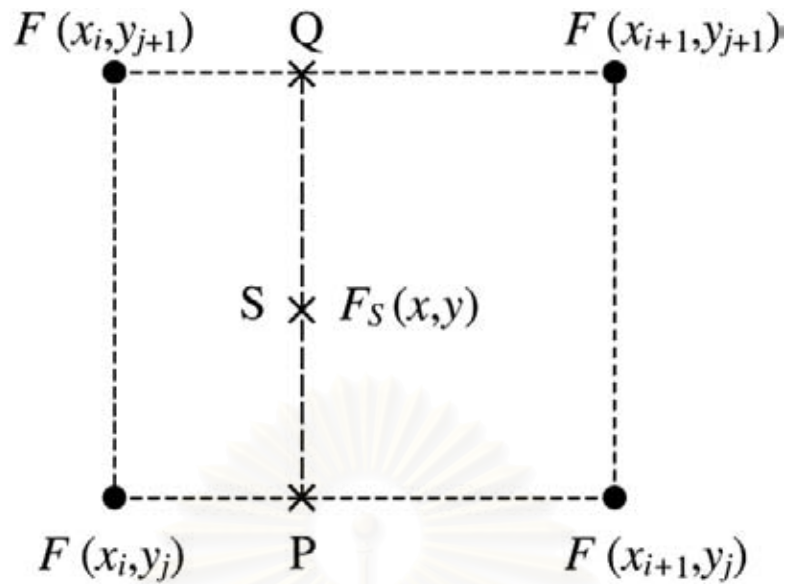


Figure 3.4: Bilinear interpolation in two dimensions.

3.4.2 Bilinear interpolation

For our work, we use bilinear interpolation to determine the value of a function in two dimensions, $F(x, y)$, which is useful for creating contour plots of our results. The bilinear interpolation is adapted from basic linear interpolation.

Figure 3.4 shows the known values for four points, $F(x_i, y_j)$, $F(x_{i+1}, y_j)$, $F(x_i, y_{j+1})$, and $F(x_{i+1}, y_{j+1})$. We want to estimate the unknown value at point S, which can be estimated by projection from the points P and Q by using linear interpolation to find F_P and F_Q (Tooprakai 1999).

- At point P:

$$\begin{aligned}
 F_P(x, y_j) &= \frac{x_{i+1} - x}{x_{i+1} - x_i} F(x_i, y_j) + \frac{x - x_i}{x_{i+1} - x_i} F(x_{i+1}, y_j) \\
 &= (1 - f_x) F(x_i, y_j) + f_x F(x_{i+1}, y_j).
 \end{aligned} \tag{3.48}$$

- At point Q:

$$\begin{aligned} F_Q(x, y_{j+1}) &= \frac{x_{i+1} - x}{x_{i+1} - x_i} F(x_i, y_{j+1}) + \frac{x - x_i}{x_{i+1} - x_i} F(x_{i+1}, y_{j+1}) \\ &= (1 - f_x) F(x_i, y_{j+1}) + f_x F(x_{i+1}, y_{j+1}), \end{aligned} \quad (3.49)$$

where $f_x = (x - x_i)/(x_{i+1} - x_i)$. After we know the values of F_P and F_Q , next we will calculate the value at point S:

$$\begin{aligned} F_S(x, y) &= \frac{y_{j+1} - y}{y_{i+1} - y_i} F_P(x, y_j) + \frac{y - y_i}{y_{i+1} - y_i} F_Q(x, y_{j+1}) \\ &= (1 - f_y) F_P(x, y_j) + f_y F_Q(x, y_{j+1}), \end{aligned} \quad (3.50)$$

where $f_y = (y - y_j)/(y_{j+1} - y_j)$, so the approximate value at point S can be written as

$$\begin{aligned} F_S(x, y) &= (1 - f_y)(1 - f_x) \cdot F(x_i, y_j) + (1 - f_y)f_x \cdot F(x_{i+1}, y_j) \\ &\quad + f_y(1 - f_x) \cdot F(x_i, y_{j+1}) + f_y f_x \cdot F(x_{i+1}, y_{j+1}). \end{aligned} \quad (3.51)$$

3.4.3 Geometric interpolation

In this work, in addition to using bilinear interpolation, we use the geometric interpolation technique, too. We use geometric interpolation to find values of a function that roughly increases or decreases as an exponential in x and y . We can determine the value of $F(x, y)$ by geometric interpolation by a process similar to the bilinear interpolation technique of Section 3.4.2 (Figure 3.4) as follows:

- At point P:

$$\log F_P(x, y_j) = (1 - f_x) \times \log F(x_i, y_j) + f_x \times \log F(x_{i+1}, y_j).$$

- At point Q:

$$\log F_Q(x, y_{j+1}) = (1 - f_x) \times \log F(x_i, y_{j+1}) + f_x \times \log F(x_{i+1}, y_{j+1}).$$

Here f_x is defined by

$$f_x = \frac{\log x - \log x_i}{\log x_{i+1} - \log x_i}.$$

From the values at points P and Q, we then determine the value at point S by the approximation formula

$$F_S(x, y) = (1 - f_y) \times \log F_P(x, y_j) + f_y \times \log F_Q(x, y_{j+1}), \quad (3.52)$$

in which f_y is defined by

$$f_y = \frac{\log y - \log y_j}{\log y_{j+1} - \log y_j}.$$

3.5 Numerical Procedure

In our problem we investigate the distribution of hydrogen atoms at each heliocentric distance in the solar system by considering the physical effects of streaming, charge-exchange collisions, and photo-ionization. We show the numerical procedure and mathematical formulation for taking these effects into account.

3.5.1 Mathematical formulation

We consider hydrogen atoms as the major neutral constituent of interstellar matter in the galaxy (Axford and Suess 1994, Audouze and Israël 1996). We would like to determine the distribution function of hydrogen atoms: $f_H(\vec{r}, \vec{v}, t)$. We apply equation (2.15) written in the form

$$\frac{\partial f_H}{\partial t} = -\vec{v} \cdot \frac{\partial f_H}{\partial \vec{r}} + \int_{\vec{v}_1} \int_{\Omega'} |\vec{v}_1 - \vec{v}| \frac{d\sigma}{d\Omega'} (f'_H f'_1 - f_H f_1) d\Omega' d^3 v_1 \quad (3.53)$$

(neglecting photoionization for the time being).

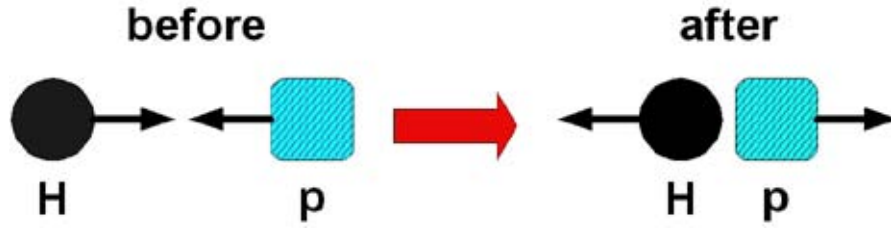


Figure 3.5: Charge exchange between a proton and a hydrogen atom in the center of mass frame.

For the collision term, the second term of the right side in (3.36), we consider elastic collisions between hydrogen and protons, so we replace f_1 with $f_p(\vec{r}, \vec{v}, t)$:

$$\frac{\partial f_H}{\partial t} = -\vec{v} \cdot \frac{\partial f_H}{\partial \vec{r}} + \int_{\vec{v}_1} \int_{\Omega'} |\vec{v}_1 - \vec{v}| \frac{d\sigma}{d\Omega'} [f_H(\vec{v}') f_p(\vec{v}'_1) - f_H(\vec{v}) f_p(\vec{v}_1)] d\Omega' d^3 v_1.$$

For the collisions of particles in the system of interest, we consider **charge exchange interactions**. It is easiest to work in the center of mass frame (see Figure 3.5).

In the case of charge exchange collisions, we can approximate the differential cross section by

$$\frac{d\sigma}{d\Omega'}(\theta', \vec{v}_1, \vec{v}) = \frac{1}{2\pi} \sigma(|\vec{v}_1 - \vec{v}|) \delta(\theta' - \pi). \quad (3.54)$$

Physically, the reason why the charge exchange differential cross section is concentrated at $\theta' \simeq \pi$ is because the hydrogen atom and proton basically continue moving in the same direction after exchanging an electron, so the result is that in the center of mass frame, the product hydrogen atom moves with an angle $\theta' \simeq \pi$ relative to the motion of the original hydrogen atom before the collision.

From Figure 3.5, conservation of momentum implies that after the collision, $\vec{p}' = \vec{p}_1$ and $\vec{p}'_1 = \vec{p}$. The mass of hydrogen can be assumed to be equal to

the mass of a proton, so the velocities after the collision are $\vec{v}' = \vec{v}_1$ and $\vec{v}'_1 = \vec{v}$.

Thus we have the greatly simplified equation,

$$\begin{aligned} \frac{\partial f_H}{\partial t} &= -\vec{v} \cdot \frac{\partial f_H}{\partial \vec{r}} \\ &+ \int_{\vec{v}_1} \int_{\Omega'} |\vec{v}_1 - \vec{v}| \frac{1}{2\pi} \sigma(|\vec{v} - \vec{v}_1|) \delta(\theta' - \pi) [f_H(\vec{v}') f_p(\vec{v}'_1) - f_H(\vec{v}) f_p(\vec{v}_1)] d\Omega' d^3 v_1, \end{aligned}$$

where $d\Omega'$ is an element of solid angle in the center of mass frame, $d\Omega' = \sin \theta' d\theta' d\phi'$. We integrate from $\theta' = 0 \rightarrow \pi$ and $\phi' = 0 \rightarrow 2\pi$:

$$\begin{aligned} \frac{\partial f_H}{\partial t} &= -\vec{v} \cdot \frac{\partial f_H}{\partial \vec{r}} \\ &+ \int_{\vec{v}_1} \int_{\phi'=0}^{2\pi} \int_{\theta'=0}^{\pi} |\vec{v}_1 - \vec{v}| \frac{1}{2\pi} \sigma(|\vec{v} - \vec{v}_1|) \delta(\theta' - \pi) [f_H(\vec{v}') f_p(\vec{v}'_1) - f_H(\vec{v}) f_p(\vec{v}_1)] d\theta' d\phi' d^3 v_1 \\ &= -\vec{v} \cdot \frac{\partial f_H}{\partial \vec{r}} \\ &+ \int_{\vec{v}_1} 2\pi \int_{\theta'=0}^{\pi} |\vec{v}_1 - \vec{v}| \frac{1}{2\pi} \sigma(|\vec{v} - \vec{v}_1|) \delta(\theta' - \pi) [f_H(\vec{v}') f_p(\vec{v}'_1) f_H(\vec{v}) f_p(\vec{v}_1)] d\theta' d^3 v_1 \end{aligned}$$

$$\frac{\partial f_H}{\partial t} = -\vec{v} \cdot \frac{\partial f_H}{\partial \vec{r}} + \int_{\vec{v}_1} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) [f_H(\vec{v}_1) f_p(\vec{v}) - f_H(\vec{v}) f_p(\vec{v}_1)] d^3 v_1. \quad (3.55)$$

We would like to solve this equation by considering the position \vec{r} in one dimension along the solar apex (we use the z -axis) and considering the velocity \vec{v} in two dimensions (v_z is the velocity along the solar apex, while v_y is perpendicular to the solar apex):

$$\frac{\partial f_H}{\partial t} = -v_z \frac{\partial f_H}{\partial z} + \int_{\vec{v}_1} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) [f_H(\vec{v}_1) f_p(\vec{v}) - f_H(\vec{v}) f_p(\vec{v}_1)] d^3 v_1. \quad (3.56)$$

For collisions of particles in the system, we use the momentum $\vec{p} = m\vec{v}$ instead of the velocity \vec{v} , to permit an easier generalization to relativistic particles.

Thus in the program we consider momentum space in two dimensions, p_z and

p_y , and assume axisymmetry in momentum space about the p_z -axis. The angle between \vec{p} and \vec{z} is θ , and we define $\mu = \cos \theta$, which gives $v_z = v \cos \theta = \mu v$.

Thus in our work, we would like to determine the distribution function as $f_H(z, p, \mu, t)$, using

$$\frac{\partial f_H}{\partial t} = -\mu v \frac{\partial f_H}{\partial z} + \int_{\vec{p}_1} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) [f_H(\vec{p}_1) f_p(\vec{p}) - f_H(\vec{p}) f_p(\vec{p}_1)] d^3 p_1.$$

We consider momentum space $d^3 p_1$ in spherical coordinates (p_1, θ_1, ϕ_1) :

$$\begin{aligned} \frac{\partial f_H}{\partial t} &= -\mu v \frac{\partial f_H}{\partial z} \\ &+ \int_{p_1} \int_{\theta_1} \int_{\phi_1} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) [f_H(\vec{p}_1) f_p(\vec{p}) - f_H(\vec{p}) f_p(\vec{p}_1)] d\phi_1 \sin \theta_1 d\theta_1 p_1^2 dp_1, \end{aligned}$$

which we integrate over the ranges $\phi_1 = 0$ to 2π and $\theta_1 = 0$ to π ($\mu_1 \equiv \cos \theta_1 = -1$ to 1):

$$\begin{aligned} \frac{\partial f_H}{\partial t} &= -\mu v \frac{\partial f_H}{\partial z} \\ &+ \int_{p_1} \int_{\mu_1=-1}^1 \int_{\phi_1=0}^{2\pi} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) [f_H(\vec{p}_1) f_p(\vec{p}) - f_H(\vec{p}) f_p(\vec{p}_1)] d\phi_1 d\mu_1 p_1^2 dp_1. \end{aligned}$$

Because of symmetry around the z -axis, the distribution functions do not depend on ϕ_1 and we can isolate the integral over the angle ϕ_1 :

$$\begin{aligned} \frac{\partial f_H}{\partial t} &= -\mu v \frac{\partial f_H}{\partial z} \\ &+ \int_{p_1} \int_{\mu_1} \left[\int_0^{2\pi} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) d\phi_1 \right] [f_H(\vec{p}_1) f_p(\vec{p}) - f_H(\vec{p}) f_p(\vec{p}_1)] d\mu_1 p_1^2 dp_1. \end{aligned} \tag{3.57}$$

We now apply the effect of the photo-ionization process from (2.16) into (3.40):

$$\begin{aligned}
\frac{\partial f_H}{\partial t} &= -\mu v \frac{\partial f_H}{\partial z} && \text{(streaming)} \\
&+ \nu_0 \left(\frac{z_0^2}{z^2} \right) f_H && \text{(photo-ionization)} \\
&+ \int_{p_1}^1 \int_{-1}^1 \left[\int_0^{2\pi} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) d\phi_1 \right] \left[f_H(\vec{p}_1) f_p(\vec{p}) - f_H(\vec{p}) f_p(\vec{p}_1) \right] d\mu_1 p_1^2 dp_1 && \\
&&& \text{(charge exchange interaction)}. \tag{3.58}
\end{aligned}$$

3.5.2 Numerical techniques

From equation (3.41), we see that term of charge exchange interactions cannot be solved analytically for general f_H and f_p . Therefore, we have used an approximation in momentum space (volume element $d\mu p^2 dp$) in two dimensions by considering discrete grid spacings ($d\mu \rightarrow \Delta\mu$, and $p^2 dp = d(p^3)/3 \rightarrow \Delta(p^3)/3$). We consider that each point in two-dimensional momentum space, (p_z, p_y) or (p, μ) , represents a “torus-like volume” in three-dimensional momentum space, a volume that covers a range of Δp in p , $\Delta\mu$ in μ , and 2π in ϕ , because of symmetry in the ϕ angle. Finally the momentum volume element is $\Delta V = 2\pi \Delta(p^3/3)\Delta\mu$. The key point is the numerical technique for calculating the last term of the equation by changing the integral in momentum space to use a summation over discrete momentum points and their associated volumes. This approximation scheme can be written as

$$\begin{aligned}
\frac{\partial f_H}{\partial t} &\approx \mu v \frac{\partial f_H}{\partial z} + \nu_0 \left(\frac{z_0^2}{z^2} \right) f_H \\
&+ \sum_{\vec{p}_1 \text{ points in 2D}} \left[\int_0^{2\pi} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) d\phi_1 \right] \left[f_H(\vec{p}_1) f_p(\vec{p}) - f_H(\vec{p}) f_p(\vec{p}_1) \right] \\
&\quad \times \left[\Delta\mu_1 \frac{\Delta(p_1^3)}{3} \right]
\end{aligned}$$

or

$$\begin{aligned}
\frac{\partial f_H}{\partial t} &\approx \mu v \frac{\partial f_H}{\partial z} + \nu_0 \left(\frac{z_0^2}{z^2} \right) f_H \\
&+ \sum_{\vec{p}_1 \text{ points in 2D}} \left[\frac{1}{2\pi} \int_0^{2\pi} |\vec{v}_1 - \vec{v}| \sigma(|\vec{v} - \vec{v}_1|) d\phi_1 \right] \left[f_H(\vec{p}_1) f_p(\vec{p}) - f_H(\vec{p}) f_p(\vec{p}_1) \right] \\
&\qquad \qquad \qquad \text{term A} \qquad \qquad \qquad \text{term B} \\
&\qquad \qquad \qquad \times \left[\Delta\mu_1 \frac{2\pi}{3} \Delta(p_1^3) \right]. \\
&\qquad \qquad \qquad \text{term C}
\end{aligned} \tag{3.59}$$

We solve (3.42) for the distribution function $f_H(p, z, \mu, t)$ by a finite difference method using forward finite differencing for the time-derivative term on the left side of equation (3.42). In the computer code, we use the array `f[w][l][u]` for $f(p_w, z_l, \mu_u)$. We use a generalized total variation diminishing method to solve the streaming term (Nutaro et al., 2001). For the charge exchange cross section in term A, we use a fit to $\sigma(|\vec{v} - \vec{v}_1|)$ data from Schultz et al. (1995) (see details in Appendix A). We calculate the complete charge exchange interaction term (by multiplying in terms A, B, and C) by our numerical program's `chex.c` routine (see Appendix C). We calculate all terms on the right side and then update over the time step by using the technique of operator splitting.

We usually define the initial condition of the distribution function of hydrogen atoms $f_H(\vec{v})$ to be zero. We then assume an inflow boundary condition very far from the Sun as in equation (2.17):

$$f_H(\vec{v}) = \frac{n}{(2\pi kT/m)^{3/2}} \exp \left[\frac{-m(\vec{v} - \langle \vec{v}_0 \rangle)^2}{2kT} \right]$$

or in terms of $f_H(\vec{p})$,

$$f_H(\vec{p}) = \frac{n}{(2\pi mkT)^{3/2}} \exp \left[\frac{-(\vec{p} - \langle \vec{p}_0 \rangle)^2}{2mkT} \right].$$

We set the outer boundary of the simulation region at the far distance of 200 AU (inflow boundary), and use conditions for interstellar medium hydrogen atoms of density $n=0.1 \text{ cm}^{-3}$, average velocity $\langle \vec{v}_0 \rangle = -26 \text{ km/s}$ ($\langle \vec{p} \rangle = m\langle \vec{v} \rangle$), and temperature $T = 10900 \text{ K}$ (Axford and Suess 1994, Zank et al., 1996, Müller et al., 1999).

For the proton distribution function f_p (which we use to calculate term B of the charge exchange interaction), we use simulation data for solar wind protons from 2-D hydrodynamical calculations performed by expert scientists (G. Zank and L. Pauls, private communication, 1997), with values of the proton density, velocity, and temperature depending on distance (z) from the Sun, from $z = 1 \text{ AU}$ to $z = 200 \text{ AU}$ (see table in Appendix B). We set the distribution function of protons depending on distance from the Sun in the form

$$f_p(z, \vec{p}) = \frac{n(z)}{[2\pi mkT(z)]^{3/2}} \exp \left[\frac{-(\vec{p} - \langle \vec{p}(z) \rangle)^2}{2mkT(z)} \right].$$

In our simulation program, we express the results for the distribution function of hydrogen atoms $f_H(t, p, \mu, z)$ by converting (p, μ) into velocity space (v_z, v_y) . We simulate over the spatial region $z=1$ to 200 AU by $\Delta z=1 \text{ AU}$, momentum $p = mv$ for $v = 0$ to 500 km/s, and $\mu = \cos \theta$ for $\theta=0$ to 180 degrees.

A strong point of our approach is that we have specially designed a simulation grid in velocity space (or momentum space) in two dimensions, in order to achieve sufficient detail in regions where a substantial hydrogen density is expected. There is a total of 226 points in momentum space (coarse grid: 181 points, fine grid: 45 points). For term C of equation (3.42), each grid point is assigned a surrounding volume.

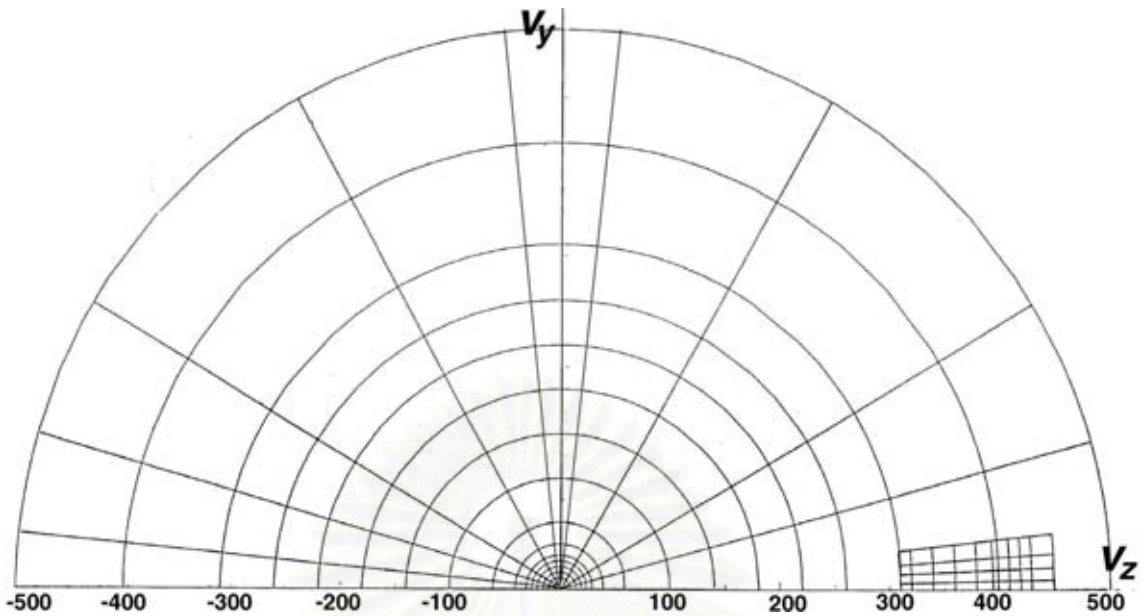


Figure 3.6: Grid points in velocity space in our simulation model (scale in km/s).

3.5.3 Numerical simulation program

We have written a C-language program for simulations in this work in 6 files as follows:

1. **boltz.c**: This is the main program in our calculation of $f_H(t, z, p, \mu)$.
2. **chex.c**: This is a subroutine for calculation of the charge exchange interaction term in equation (3.42) as a process of elastic collisions between hydrogen atoms and solar wind protons.
3. **initial.c**: This is a subroutine for defining the distribution function of hydrogen atoms f_H at the initial time of the simulation. We can set this to a Maxwellian distribution, or for simulation results shown in the next chapter, we used an initial f_H of zero.

4. **streaming.c**: This is a subroutine for calculation of the advection term for the streaming process of flowing of hydrogen atoms, in which we applied a generalized total variation diminishing code (Nutaro et al., 2001).

5. **nrutil.c**: This contains subroutines for supporting techniques of computation for allocating and deallocating memory for vectors and matrices, which we modified from Numerical Recipes in C (Press et al., 1998).

6. **printout.c**: This is a subroutine for output of computational results of f_H , which we have calculated by interpolation in velocity space (v_z, v_y) .

The flow chart structure of our simulation program is shown in Figure 3.7.



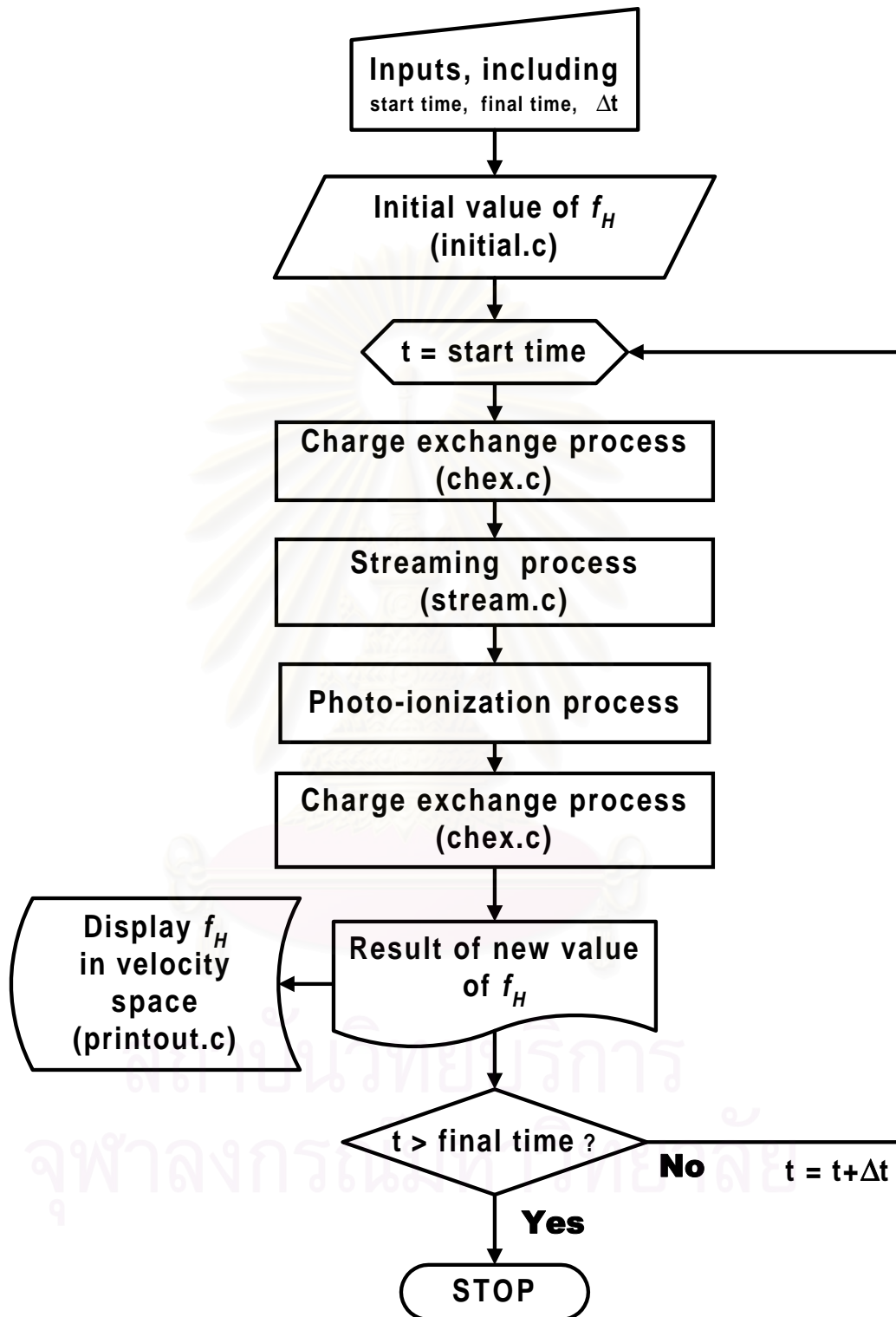


Figure 3.7: Flow chart of the simulation program.

Chapter 4

Results

We use an initial condition of $f_H = 0$ throughout, and an inflow boundary condition (at 200 AU from the Sun) for interstellar hydrogen atoms of density $n_H = 0.1 \text{ cm}^{-3}$, kinetic temperature $T = 10900 \text{ K}$, and average velocity $\langle v_0 \rangle = -26 \text{ km/s}$ (following Zank and Pauls, 1996). We plot the initial distribution function of interstellar hydrogen in velocity space before its flow into the solar system in Figure 4.1.

In simulating the time evolution of the distribution function f_H according to equation (3.42), we use a time step $\Delta t = 5$ days and continue the simulation until f_H relaxes to the steady-state neutral distribution, which we found from results that used a simulation time of 60000 days. [Such a long time is needed for relaxation from an initial condition of $f_H = 0$. The relaxation time after a minor perturbation may well be much shorter.] The actual CPU time used on a PC Linux system computer was 42.5 seconds per time step.

The values of f_H in units of $\text{cm}^{-3}(\text{eV}/c)^{-3}$ are plotted in two-dimensional velocity space (v_z - v_y space) in which v_z is the velocity in the direction of flow from the Sun along the solar apex (v_z is positive for particles moving outward from the Sun, whereas v_z is negative for particles flowing toward the Sun), and v_y

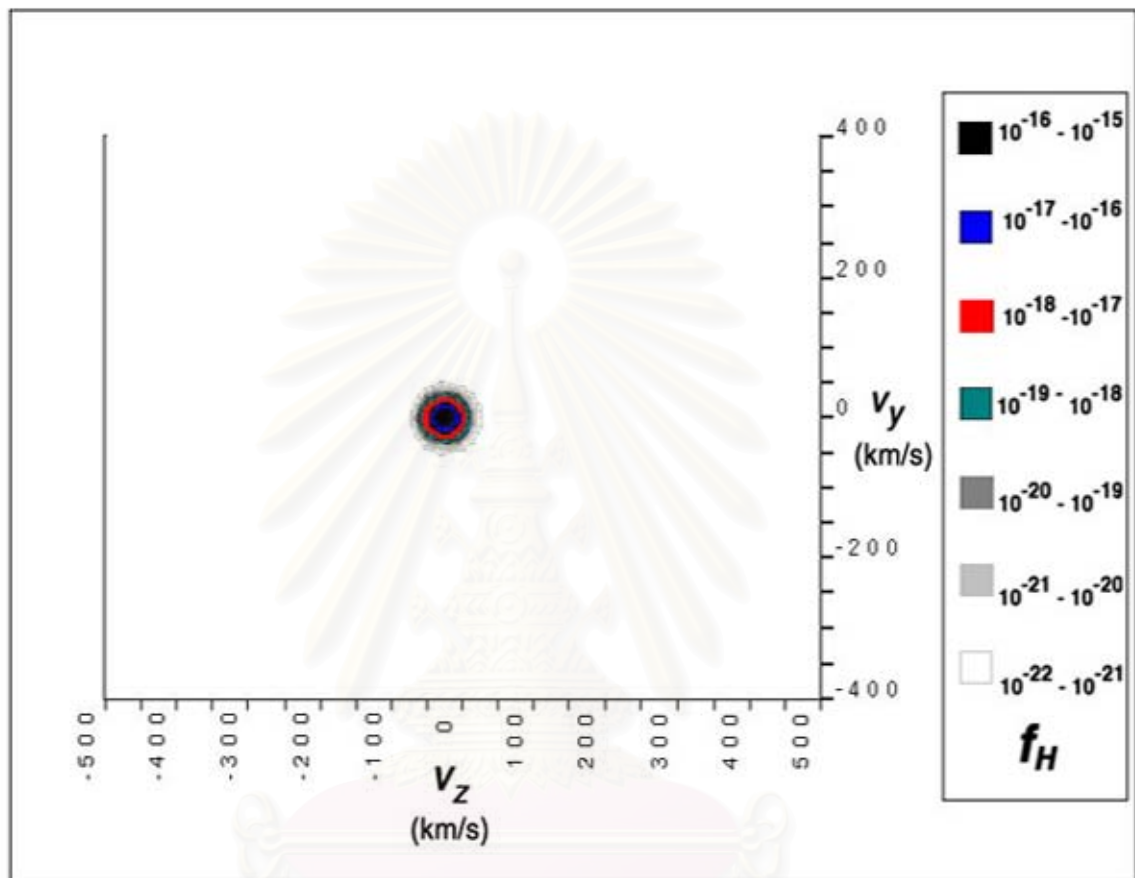


Figure 4.1: Distribution function in velocity space of hydrogen atoms that flow in through the boundary at 200 AU. In this figure and subsequent figures, f_H is in units of $\text{cm}^{-3}(\text{eV}/c)^{-3}$.

is the velocity in a direction perpendicular to v_z (assuming cylindrical symmetry about the z -axis).

Before illustrating simulation results, we show f_p , the velocity distributions of plasma protons (using data from Appendix B), in three distinct regions as: (1) supersonic solar wind protons in the inner solar system, (2) solar wind plasma in the inner, shock-heated heliosheath, and (3) plasma protons outside the heliopause (Figures 4.2, 4.3, and 4.4, respectively). These plasma protons at each location in distance z have a direct effect on the distribution of neutral hydrogen that flows into the solar system after charge exchange interactions. Because H- p charge exchange causes H and p to switch locations in velocity space, charge exchange in each region effectively injects a distribution of hydrogen (f_H) like the f_p distribution.

We illustrate the results from our simulation in Figures 4.5 to 4.13 for different locations along the solar apex (20 to 180 AU from the Sun).

From the results in Figures 4.5 to 4.13, we found the steady-state distribution of neutral hydrogen in velocity space. The dominant population of neutral hydrogen (at a high density of about 10^{-16} to 10^{-15} $\text{cm}^{-3}(\text{eV}/c)^{-3}$, which we call the “low-velocity peak”) appears near the initial interstellar hydrogen velocity ($v \approx -26$ km/s), and part of the hydrogen (at a peak about 100 times lower than that of the low-velocity peak) occurs at high velocity ($v \approx 400$ km/s) in the opposite direction of the interstellar hydrogen flow. The low-velocity peak includes the original interstellar hydrogen atoms which flow from the outer boundary into the solar system, and the high-velocity peak (near the velocity of solar wind protons) results from the charge exchange interaction of hydrogen with solar wind protons ($\text{H}+p \rightarrow p+\text{H}$). Thus we see that some hydrogen atoms flow toward the

Sun and some flow outward. The low-velocity peak overlies a highly anisotropic pedestal distribution, which is broader than the high-velocity peak, over the entire simulation range. We can explain the results in detail as follows:

- For distances ≤ 80 AU (Figures 4.5 to 4.8), we found low-velocity and high-velocity peaks, with f_H values only weakly dependent on distance. The low-velocity distribution peak appears at negative v_z only, indicating hydrogen flow into the solar system. The peak distribution is apparently a truncated, narrow Maxwellian due to interstellar hydrogen (Figure 4.4) superimposed on a much broader half-Maxwellian due to charge exchange between hydrogen atoms with the high-temperature (broad) proton distribution of the inner heliosheath (Figure 4.3). Both distributions are truncated for $v_z < 0$ because only inflowing H from these sources is found in the inner solar system. The high-velocity peak due to charge exchange interactions within the inner solar system is similar to the distribution in Figure 4.2 and has a Maxwellian-like distribution. The density of the high-velocity peak (a “beam-like” distribution) can be quantitatively understood in terms of a Lagrangian integral along characteristics:

$$f_H = \frac{1}{v_z} \int_0^z \left(\frac{\partial f}{\partial t} \right)_{coll} dz. \quad (v_z > 0)$$

Here $(\partial f / \partial t)_{coll}$ is dominated by the production term close to the Sun, and the integral of the production term alone provides a good approximation to the simulation results.

- For distances of 100-120 AU (Figures 4.9 and 4.10), in the inner heliosheath, we found low-velocity and high-velocity peaks like those at 1-80 AU, except that the broad distribution due to charge exchange in the heliosheath is quite different.

That distribution has expanded in the v_y -direction. Plasma protons in this region have a very high temperature due to shock heating of the solar wind (Zank et al. 1996), so charge exchange interactions in this region produce hydrogen atoms of a distribution like that shown in Figure 4.3. We can understand the elongation of this component along the v_y -axis from the integral along characteristics:

$$f_H = \frac{1}{v_z} \int_{86 \text{ AU}}^z \left(\frac{\partial f}{\partial t} \right)_{coll} dz, \quad (v_z > 0)$$

or

$$f_H = \frac{1}{|v_z|} \int_z^{131 \text{ AU}} \left(\frac{\partial f}{\partial t} \right)_{coll} dz, \quad (v_z < 0)$$

where 86 AU and 131 AU are the boundaries of the inner heliosheath region of hot plasma. [Note that there is no divergence as $v_z \rightarrow 0$, where the integral is better expressed as an integral over time, and at late times $(\partial f / \partial t)_{coll} \rightarrow 0$ as an equilibrium is reached between production and destruction by charge exchange.] The increase at $v_z \approx 0$ appears as an elongation along the v_y -axis. Thus we found hydrogen from all three sources in this region (100-120 AU): interstellar medium neutral hydrogen which flows into this region, hydrogen locally produced by charge exchange, and a high velocity peak of hydrogen atoms occurring through charge exchange from the inner region and flowing outward from the Sun at high velocity.

- For distances of 140-180 AU (Figures 4.10 to 4.13), outside the heliopause, the velocity distributions have a similar pattern, with hydrogen found in the velocity range of -50 to 500 km/s, except that hydrogen from inner sources appears only at $v_z > 0$. The low-velocity peak is dominant at negative and low velocity because charge exchange interactions with local plasma protons have low velocity and low

temperature (see the distribution of f_p in Figure 4.4). Thus we found hydrogen atoms occurring from three sources: (1) from this region with a velocity peak around low velocity plus interstellar medium neutral hydrogen which flows into this region, (2) a high-velocity peak of hydrogen atoms occurring through charge exchange inside the termination shock and flowing outward from the Sun at high velocity, and (3) from some hydrogen atoms with a broad distribution, created by charge exchange in the inner heliosheath and flowing outward to this region.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

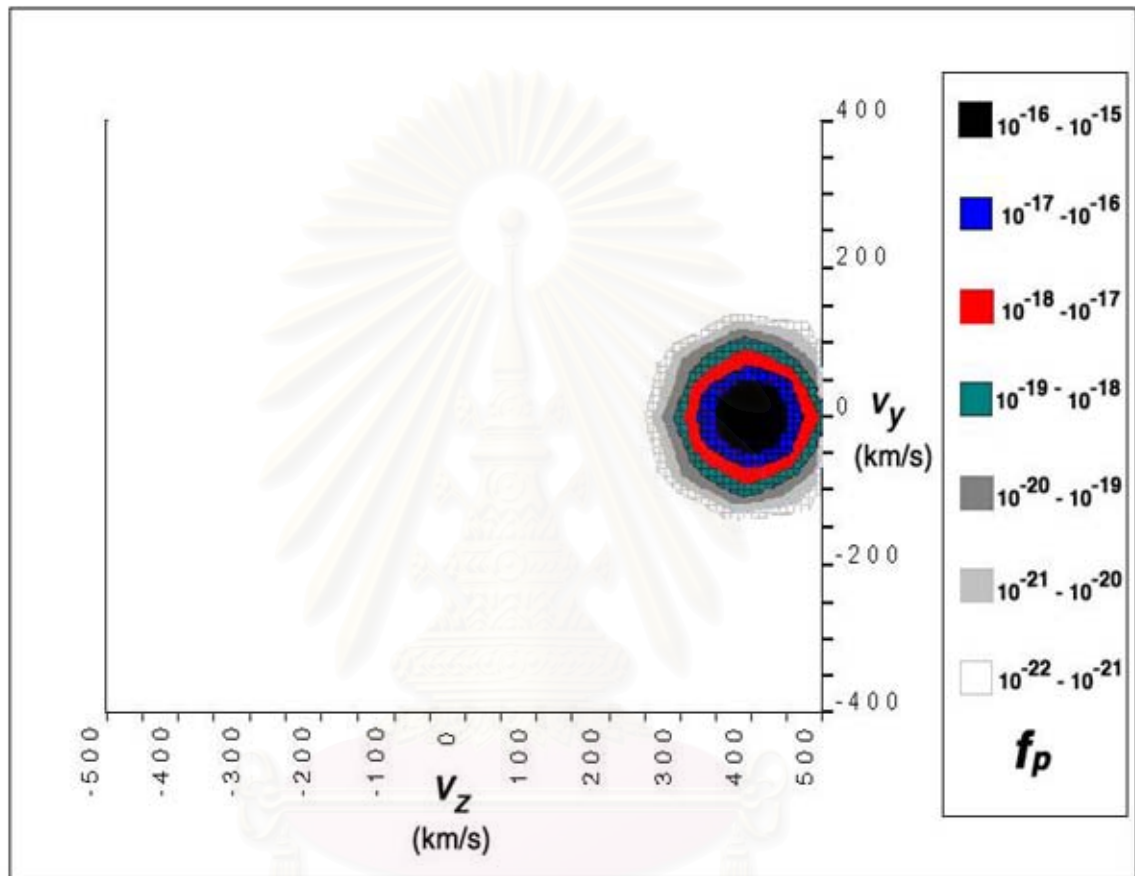


Figure 4.2: Distribution function in velocity space of solar wind protons at 1 AU.

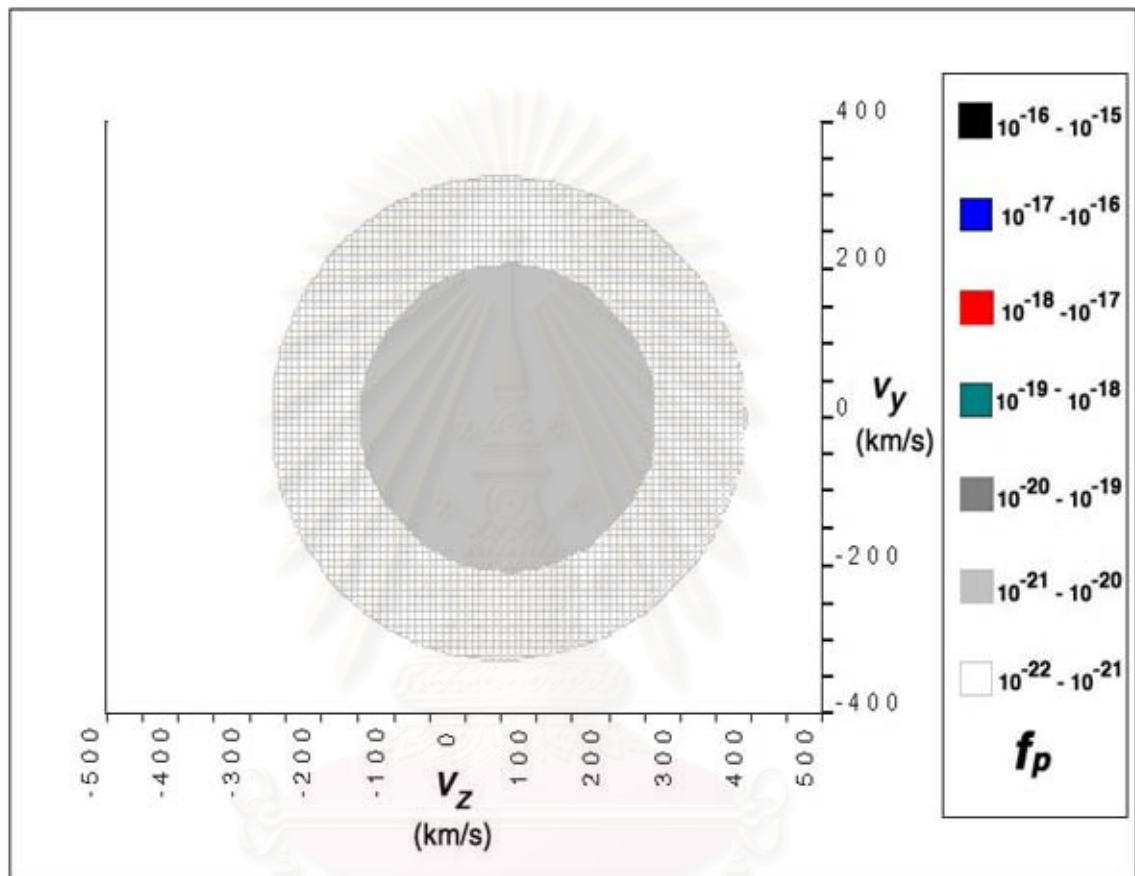


Figure 4.3: Distribution function in velocity space of solar wind protons at 100 AU.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

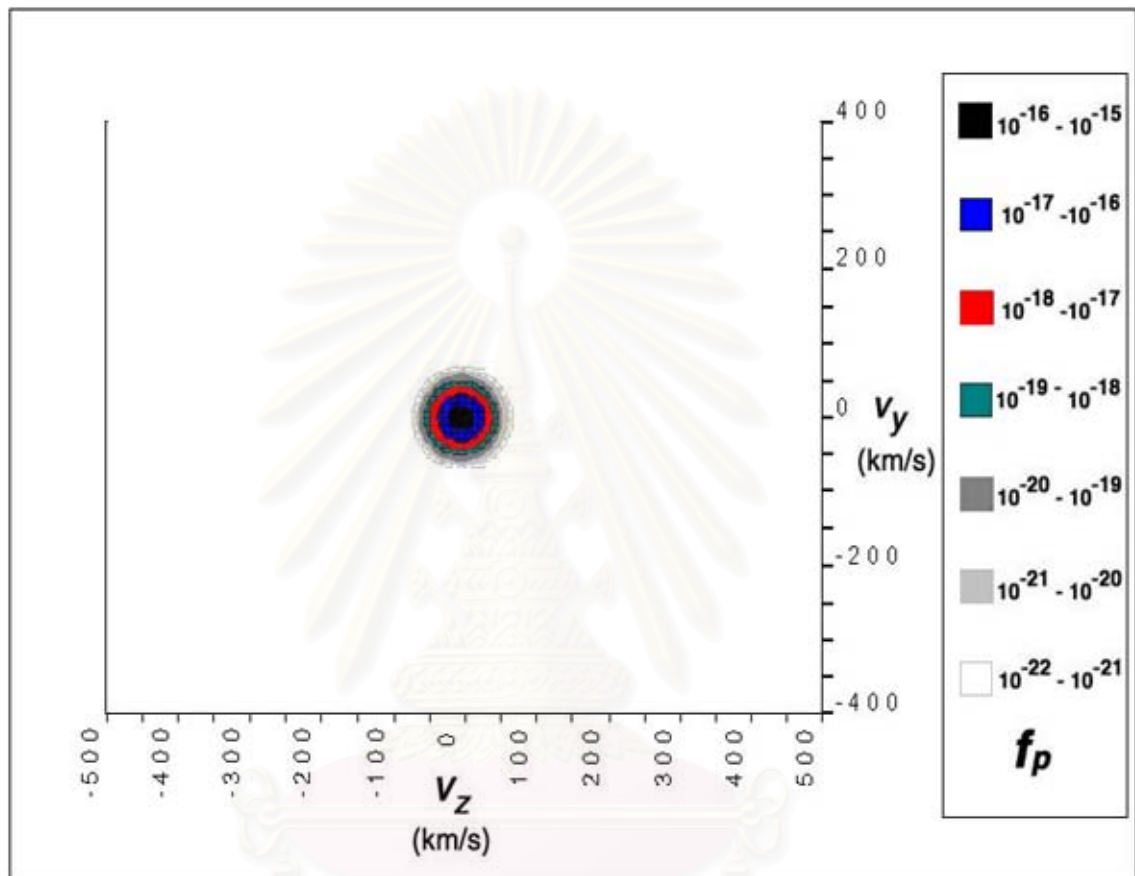


Figure 4.4: Distribution function in velocity space of solar wind protons at 160 AU.

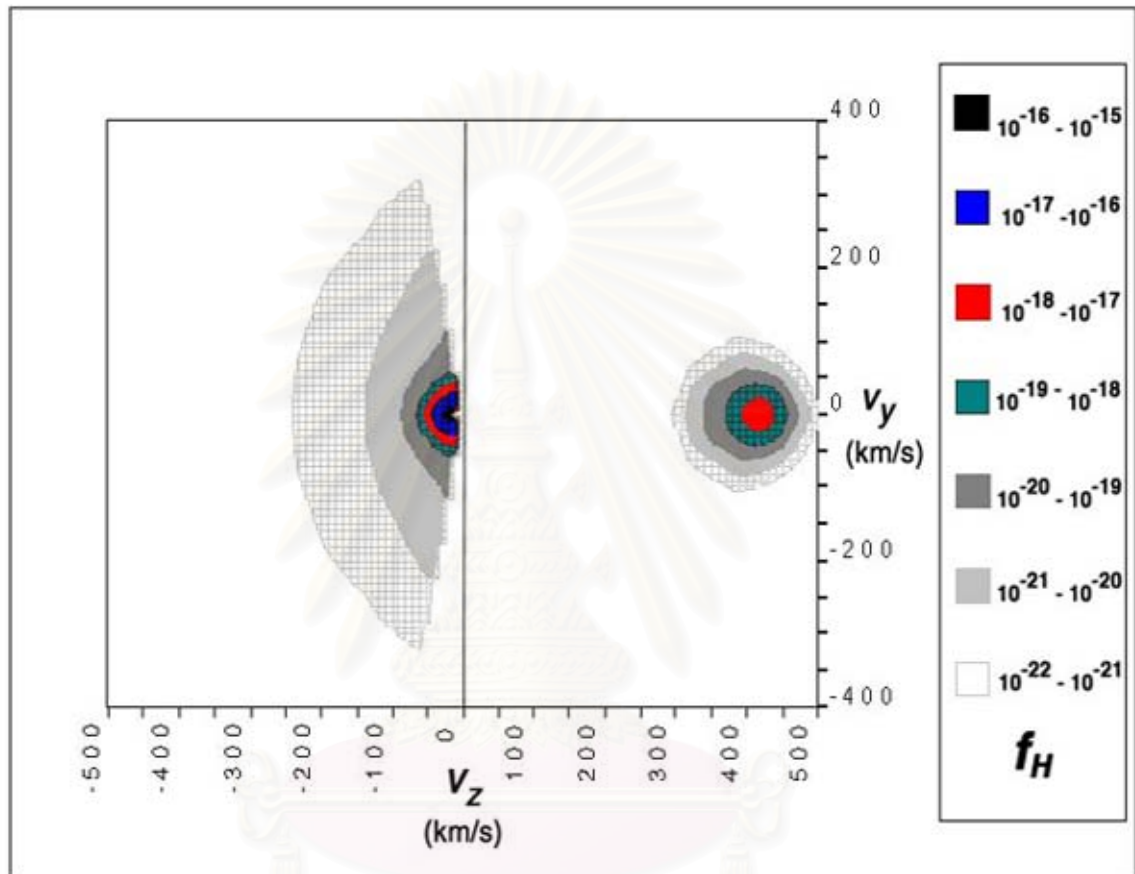


Figure 4.5: Distribution function of hydrogen atoms in v_z - v_y space at $z = 20$ AU.

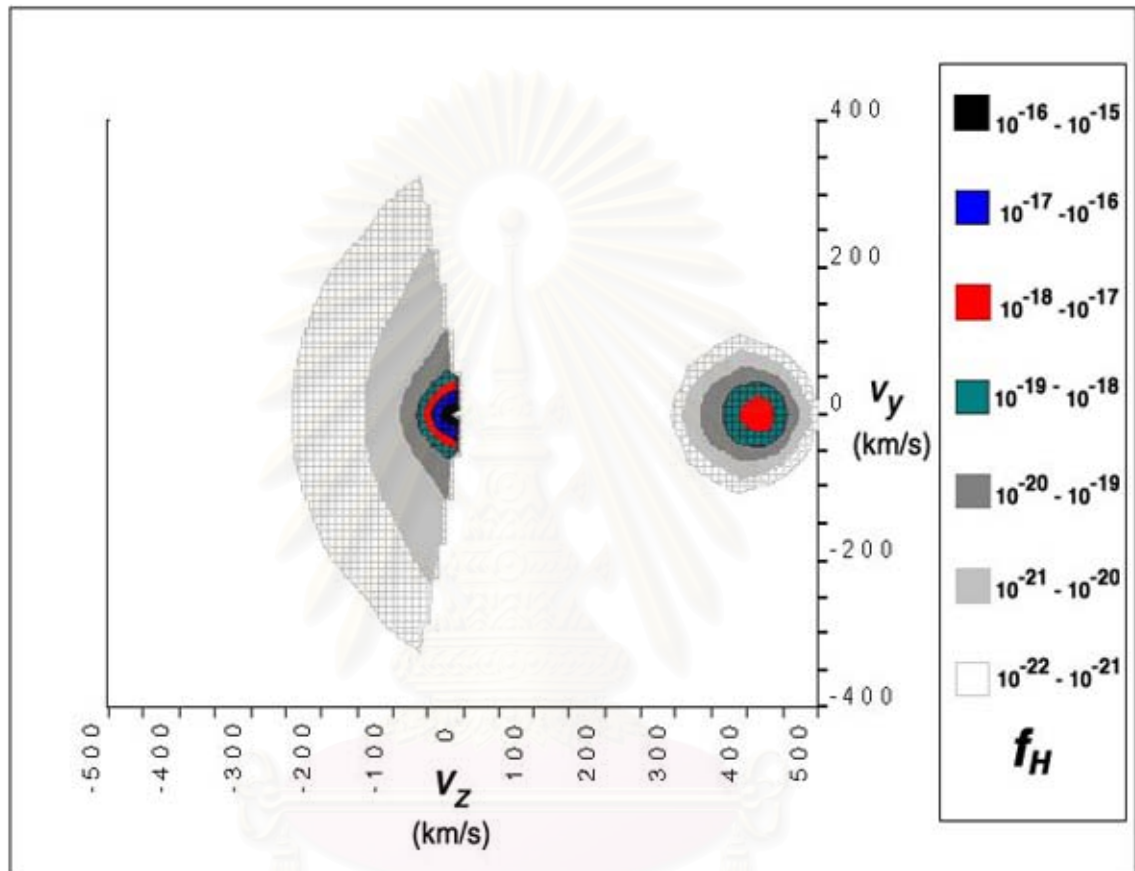


Figure 4.6: Distribution function of hydrogen atoms in v_z - v_y space at $z = 40$ AU.

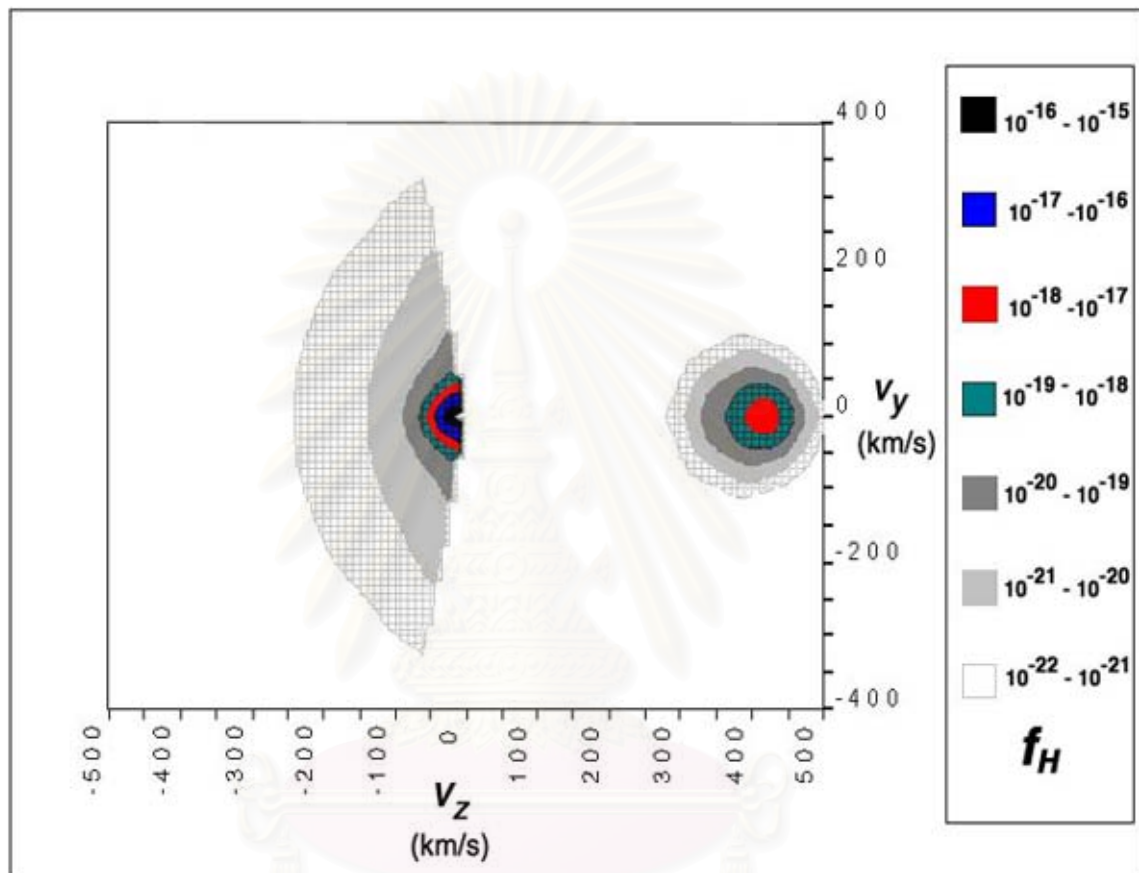


Figure 4.7: Distribution function of hydrogen atoms in v_z - v_y space at $z = 60$ AU.

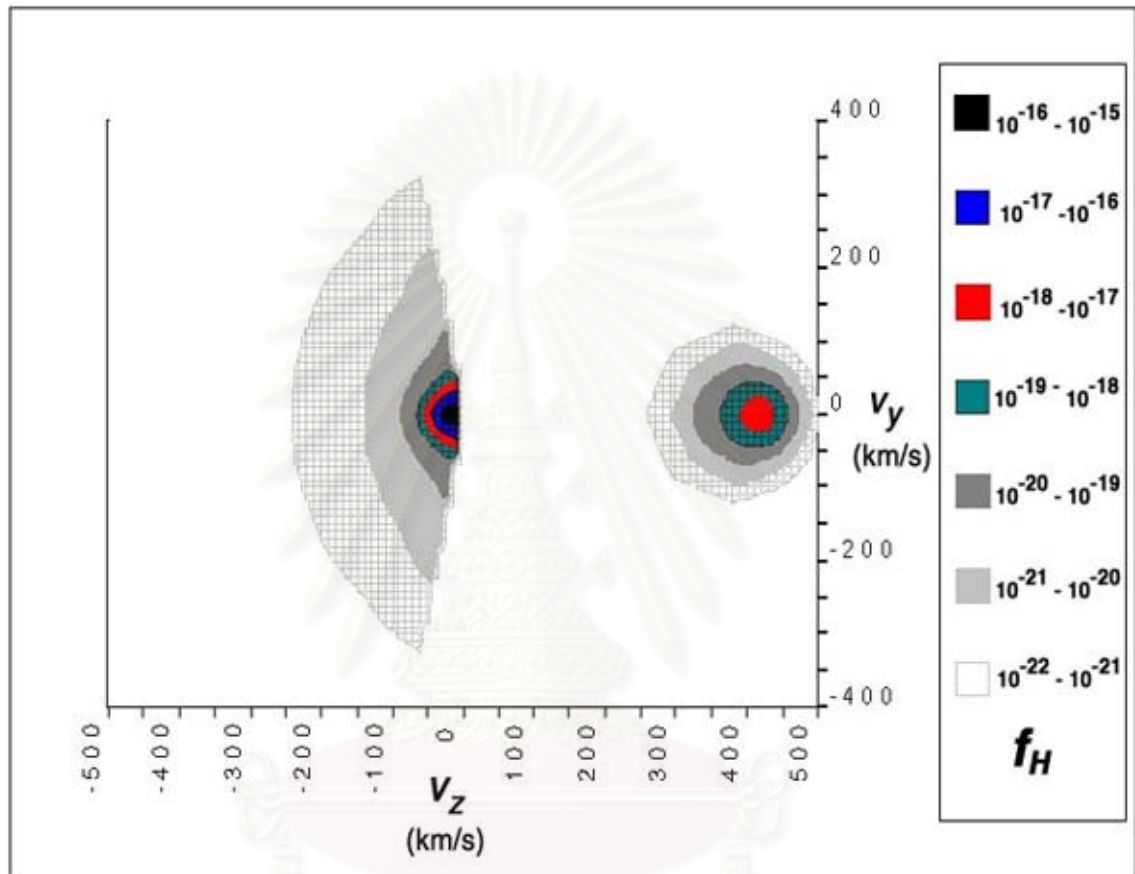


Figure 4.8: Distribution function of hydrogen atoms in v_z - v_y space at $z = 80$ AU.

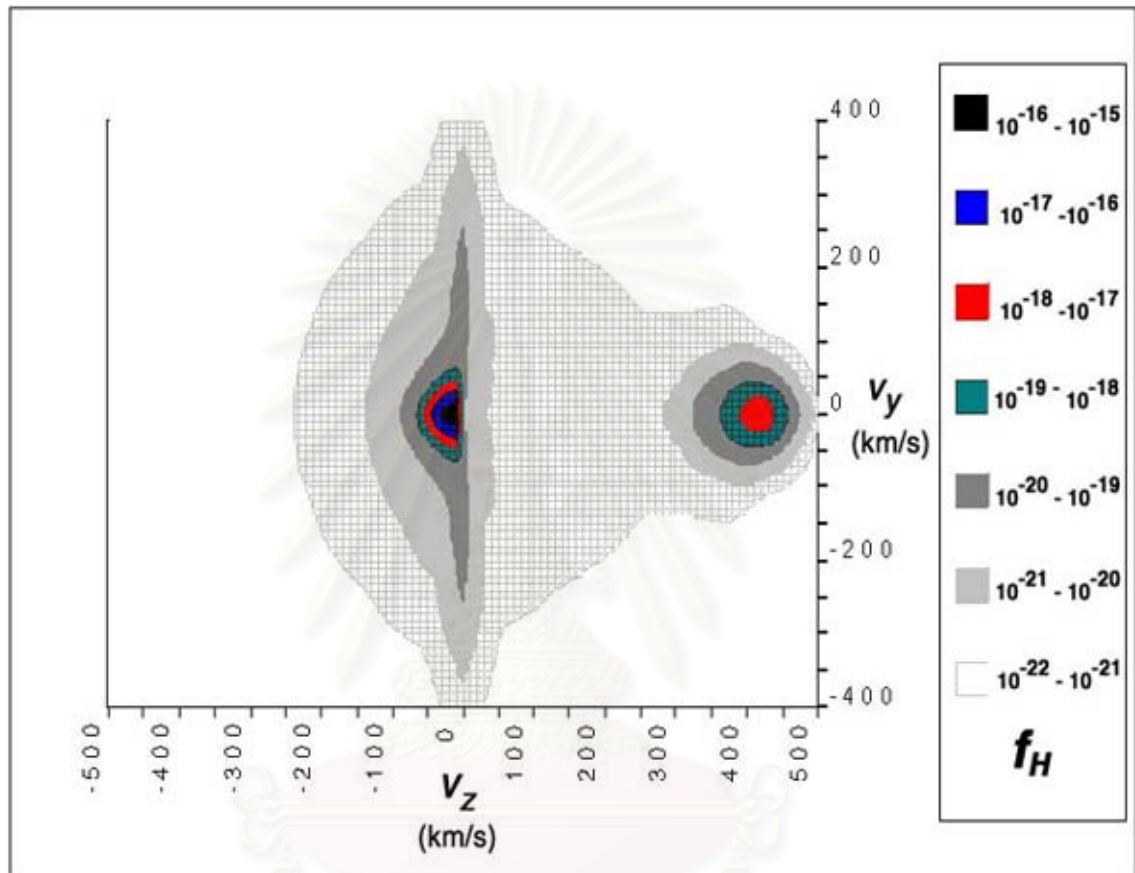


Figure 4.9: Distribution function of hydrogen atoms in v_z - v_y space at $z = 100$ AU.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

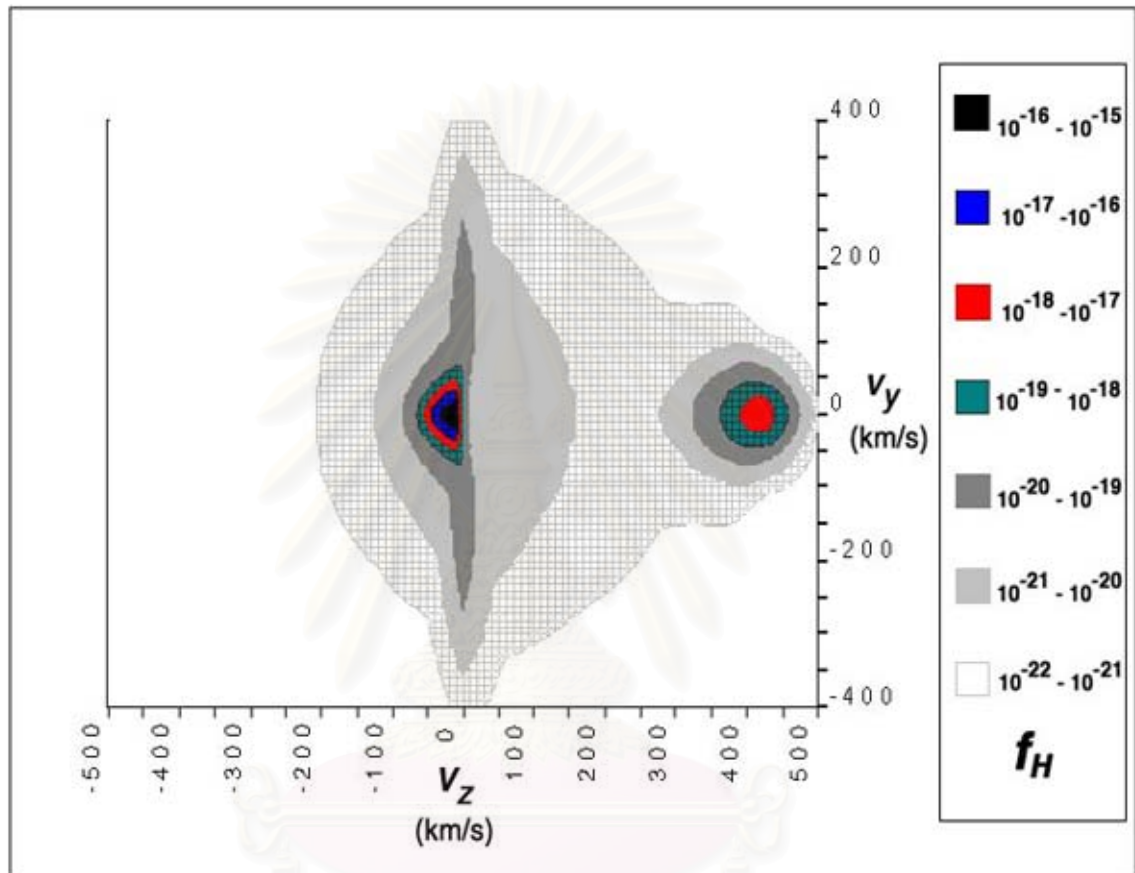


Figure 4.10: Distribution function of hydrogen atoms in v_z - v_y space at $z = 120$ AU.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

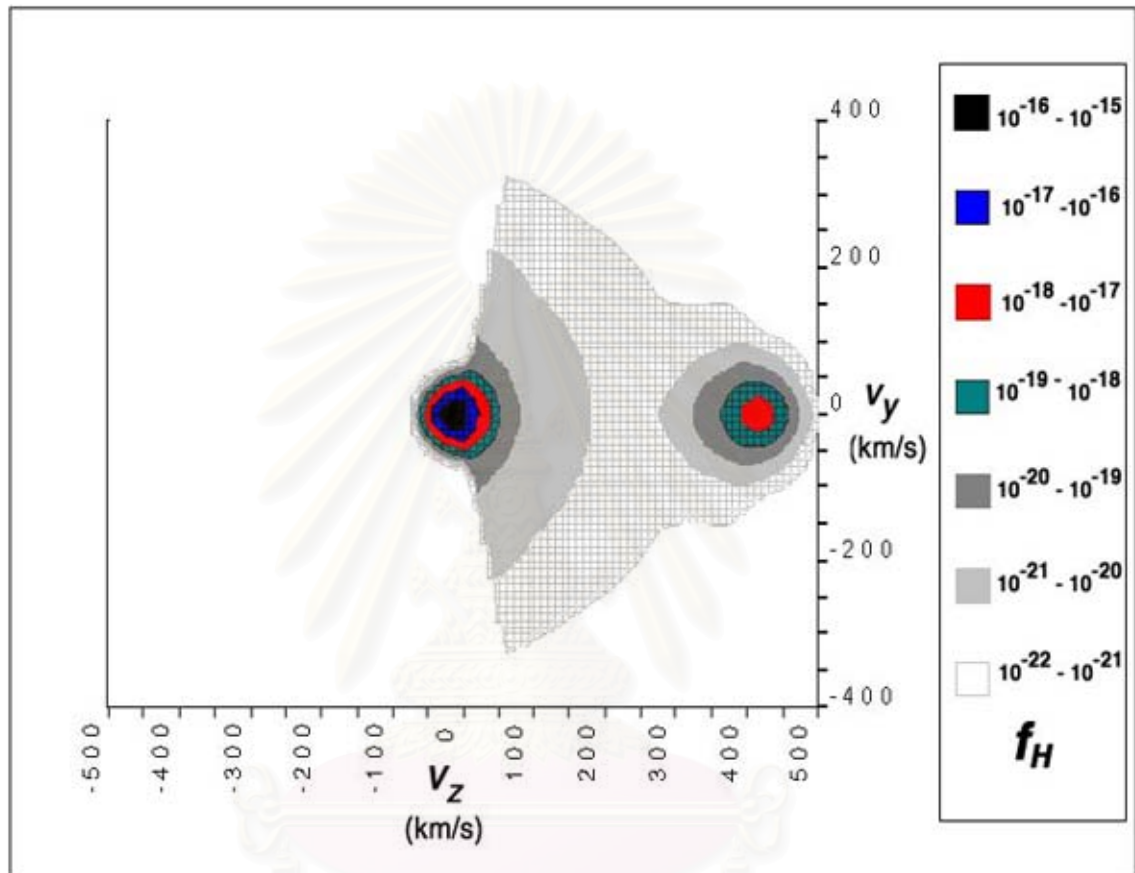


Figure 4.11: Distribution function of hydrogen atoms in v_z - v_y space at $z = 140$ AU.

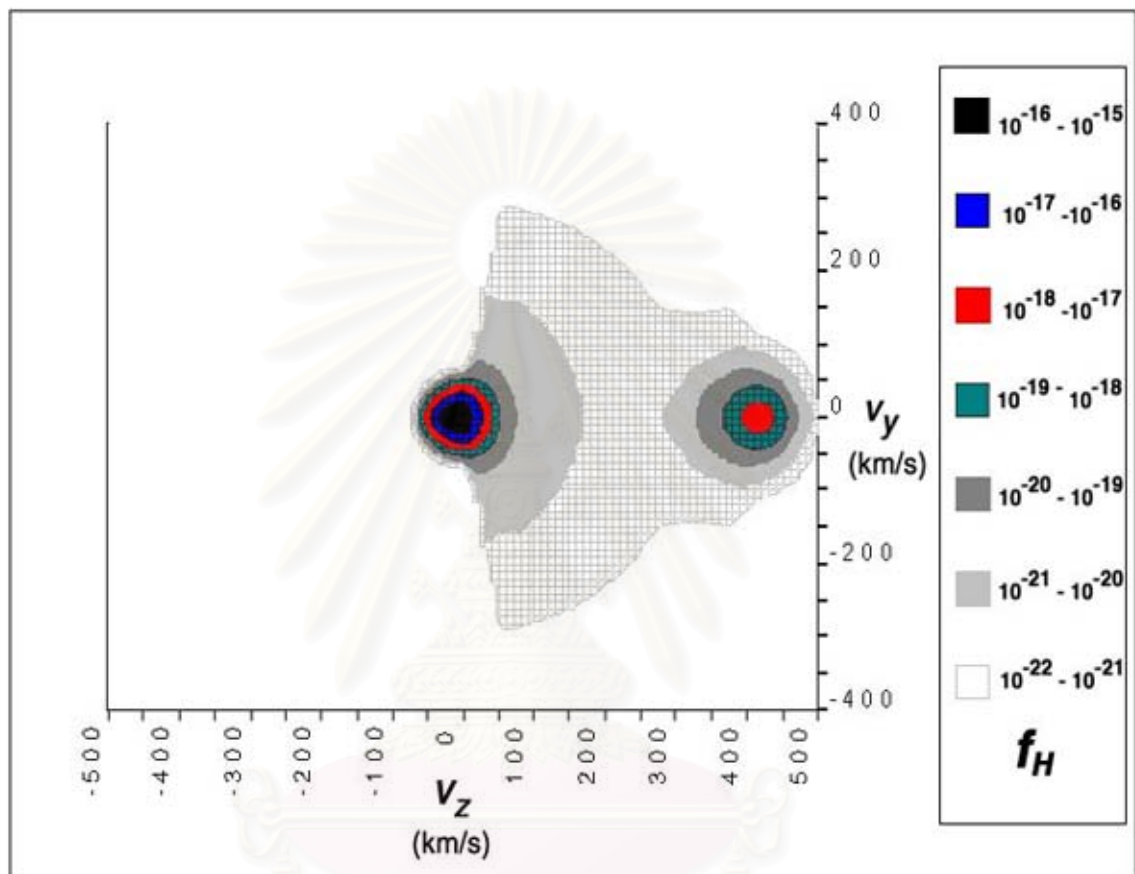


Figure 4.12: Distribution function of hydrogen atoms in v_z - v_y space at $z = 160$ AU.

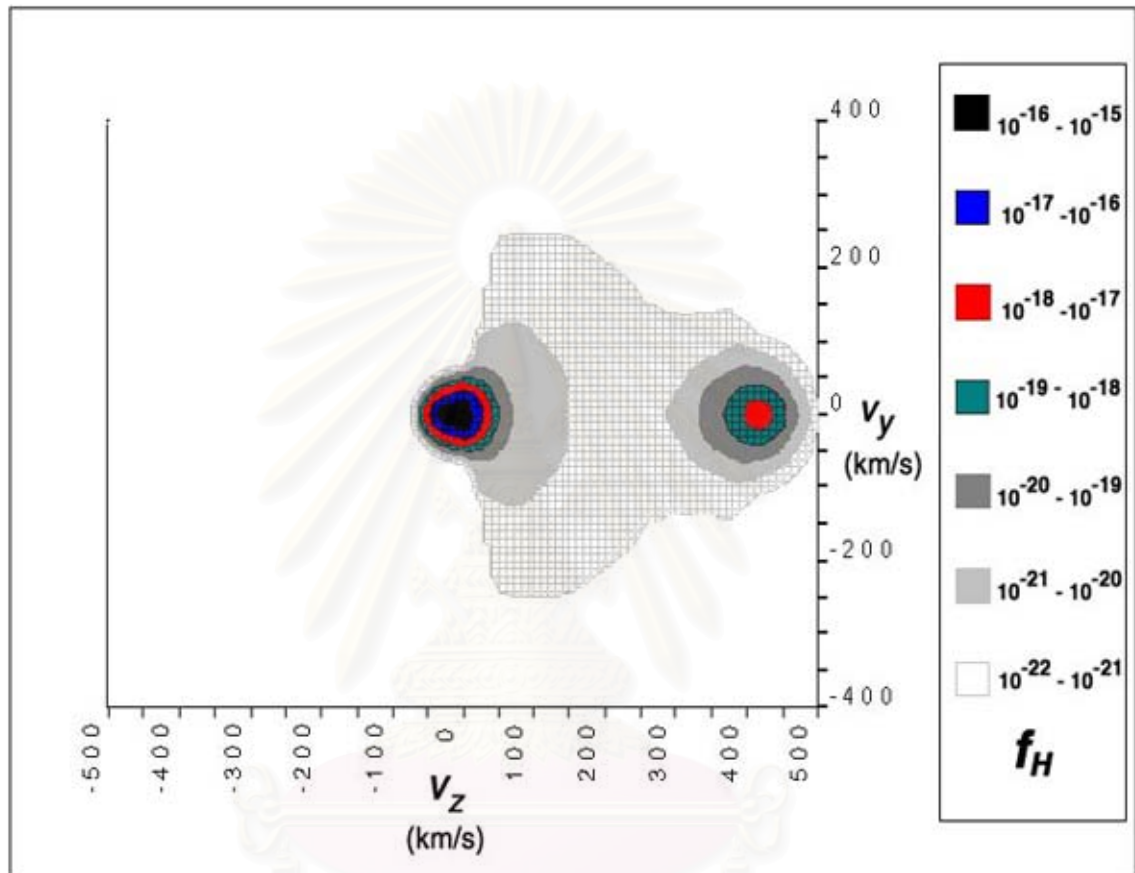


Figure 4.13: Distribution function of hydrogen atoms in v_z - v_y space at $z = 180$ AU.

Chapter 5

Discussion and Summary

We have developed a simulation model for determining the distribution of hydrogen atoms resulting from neutral interstellar matter which flows into the solar system. We numerically solve a Boltzmann transport equation that includes the processes of streaming, photo-ionization, and charge exchange interaction given a distribution of protons, based on previous research results kindly provided by Zank and Pauls (private communication, 1997). Our research focuses on the neutral hydrogen velocity distribution functions, clarifying the role of charge exchange in the evolution and response of the distribution to solar wind protons along the solar apex. The resulting neutral hydrogen distribution functions clearly show the importance of effects of the charge exchange interaction and streaming. Our results provide more detailed velocity-space hydrogen distributions than those obtained in previous research.

We obtained simulation results determining the steady-state distribution in velocity space vs. distance along the solar apex (Figures 4.5 to 4.13), which for clarity are summarized in Figure 5.1, and we analyze physical processes which we consider in detail from simulation results to describe neutral matter and heliospheric structure. Note that charge exchange tends to switch the locations of

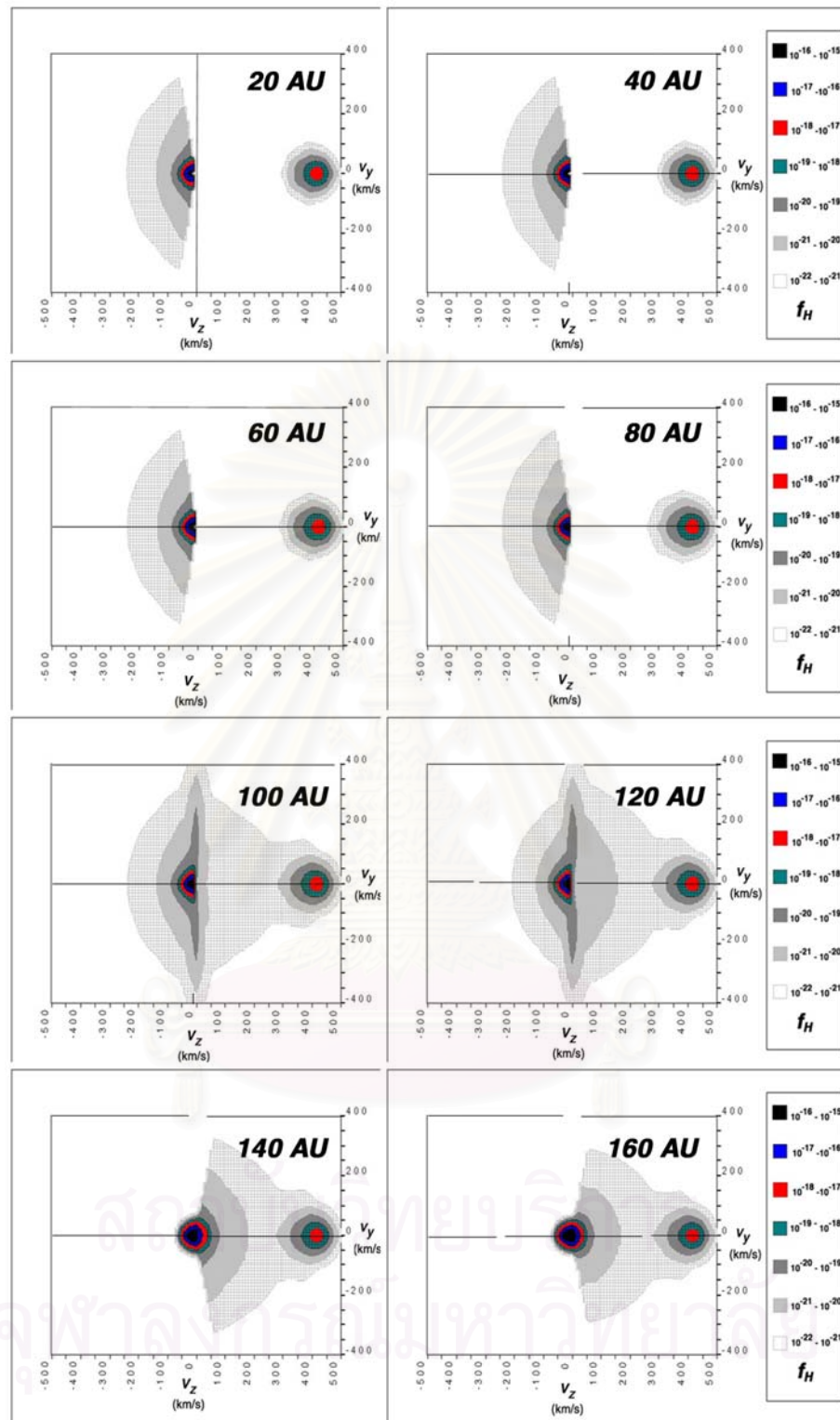


Figure 5.1: Summary of results for the hydrogen distribution in velocity space at various distances (20 to 160 AU) along the solar apex, inside and outside the solar system.

H and p in velocity space, so at a given distance, f_H will appear in the region where f_p is large (see graphs of f_p in Figure 4.2 to 4.4). The origin of the neutral hydrogen can be explained as follows:

(1) Neutral hydrogen occurred from the original interstellar medium that flows into the solar system from a very far distance from the Sun. This component is found in all regions considered.

(2) Some neutral hydrogen is found at high velocity (and positive v_z) at all distances in our simulation results. These hydrogen atoms occurred from charge exchange interactions with the cold, supersonic solar wind inside the termination shock (within 80 AU), so these neutral hydrogen atoms have a direction outward from the Sun. We also see this component in each region along the solar apex from 1 to 200 AU.

(3) Some neutral hydrogen results from charge exchange interactions with hot plasma protons in the shock-heated solar wind of the inner heliosheath region (in this work, 86-131 AU). Some of this neutral hydrogen, with $v_z \approx 0$, remains in this region; that with $v_z > 0$ is found outside this region; and that with $v_z < 0$ is found in inner regions.

5.1 Comparison with Previous Results

Our numerical results have been compared with previous work of the Computational Astrophysics Research Group at Chulalongkorn University Thailand (Boonma 2000) and our collaborative group at the Bartol Research Institute of the University of Delaware, USA (Müller et al. 2000).

► Comparison with results of Boonma (2000) (Figure 5.2) - We consider the streaming process, charge exchange interaction, and photo-ionization process in

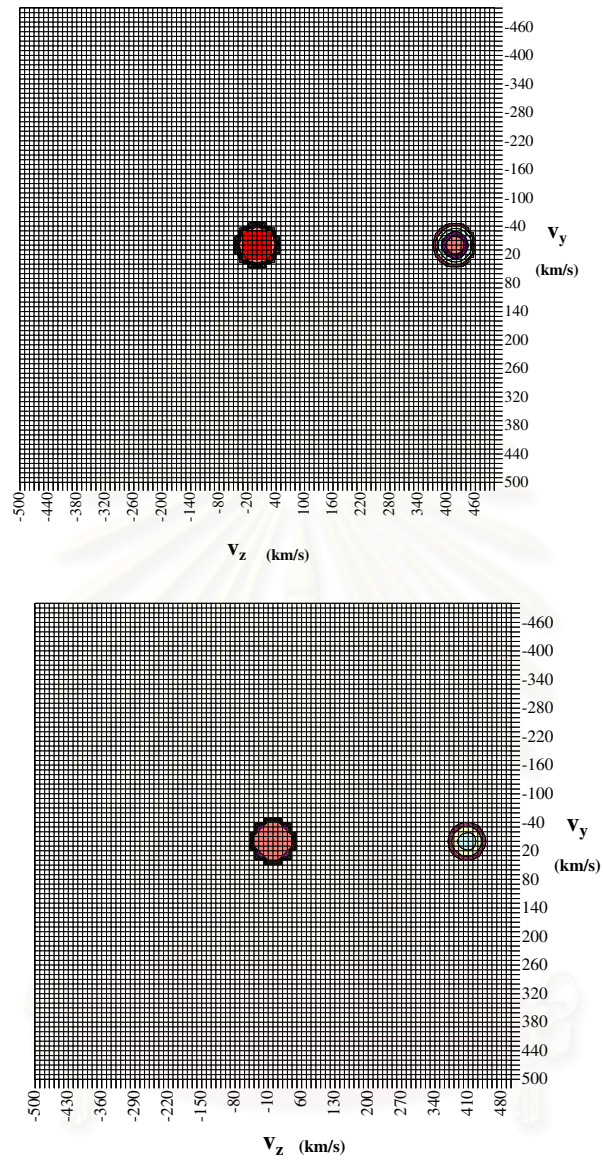


Figure 5.2: Results of the hydrogen distribution in velocity space at 10 AU (top) and 30 AU (bottom) from Boonma (2000). Contours indicate $f_H = 10^{-21.0}, 10^{-20.5}, 10^{-20.0}, 10^{-19.5},$ and $10^{-19.0} \text{ cm}^{-3}(\text{eV}/c)^{-3}$ for a charge exchange duration of 1 day.

our simulation model, whereas Boonma used elastic collisions only in a Boltzmann transport equation. Thus effects of the flow of hydrogen did not appear in those results (Figure 5.2). Since Boonma did not consider interstellar flow into the solar system, our results describe physical effects of the velocity distribution along the solar apex more clearly. Furthermore, Boonma was not able to consider a steady state. Our results also show the different distributions at 100-120 AU better than Boonma (2000), who only considered distances inside 60 AU. Anyway, Boonma employed a good concept for detailed consideration of elastic collisions of hydrogen atoms and protons.

► Comparison with results of Müller et al. (2000) (Figure 5.3) - Müller et al. used self-consistent hybrid simulation of the interaction of hydrogen with plasma protons based on the particle mesh method to solve the Boltzmann equation. Results for the hydrogen distribution in velocity space are qualitatively similar to our results along the solar apex (to z of about 300 AU, while our results are for z up to about 200 AU).

In Figure 5.3 from Müller et al. (2000), the hydrogen distributions at 1 AU and 100 AU have large statistical variations. The hydrogen distribution at 1 AU, with a circle at around 400 km/s, many arise because of an artifact of integrating over a volume including $r = 0$.

In addition to calculating the velocity space distribution of hydrogen atoms along the solar apex, Müller et al. calculated many other aspects of heliospheric structure and the interaction between the solar wind and the interstellar medium.

At the suggestion of one of those authors (G. Zank), we have considered the photo-ionization process in our model, which Müller et al. did not consider.

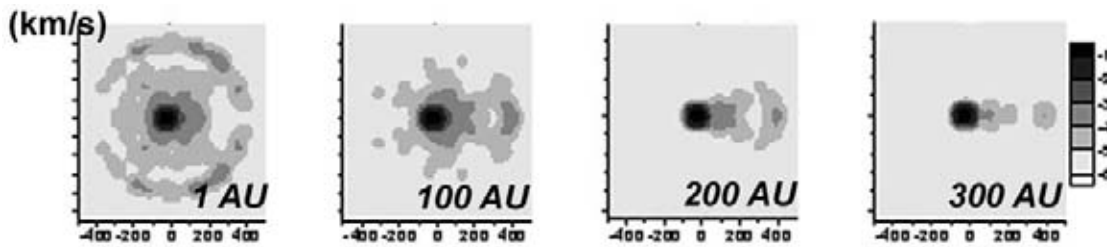


Figure 5.3: Results of the hydrogen distribution in velocity space at 1 AU, 100 AU, 200 AU, and 300 AU from Müller et al. (2000).

Since our simulations are specifically designed for a more limited objective, our finite difference method calculates the neutral hydrogen velocity distribution in more detail, avoiding the statistical uncertainty of the particle method. However, this detail comes at the expense of approximating the true heliospheric geometry by a plane-parallel geometry, which limits the application of our results to a distance of about 200 AU. Also, our results are not self-consistent in that we do not calculate the effect of our derived hydrogen distribution on the proton distribution, as Müller et al. do. The simulations of Müller et al. are much more computation-intensive and involve a solution of hydrodynamic equations of motion.

5.2 Summary

We summarize the conclusions of this work as follows:

- We have developed and adapted a mathematical formulation and simulation model of the distribution of the neutral interstellar medium (ISM) in the heliosphere, and calculate the hydrogen atom distribution function f_H in velocity space along the solar apex.

- We can find neutral hydrogen atoms in three regions of velocity space, which come from three sources: 1) the original ISM, 2) neutral hydrogen produced from charge exchange interactions with cold, supersonic solar wind inside the termination shock, and 3) neutral hydrogen produced from charge exchange interactions with the solar wind shock heated in the inner heliosheath (between the termination shock and heliopause).

- The hydrogen distribution in the solar system is clearly different in three regions: the inner solar system (within 86 AU), beyond the heliopause (at or beyond 131 AU), and the inner heliosheath (86 – 131 AU).

Our results can be used to improve the multi-fluid model (Zank et al. 1996) of the structure of our solar system. Zank et al. used a sum over all three components, approximated as Maxwellian distributions. In this work we find that H distributions from the ISM or due to charge exchange with inner heliosheath protons are highly non-Maxwellian due to streaming effects. Anyway, our model cannot examine structure outside the heliosphere, such as the possible bow shock of interstellar plasma, examined in the two-shock model of Müller et al. (2000).

For future work we can use this model to simulate the time dependence of the neutral hydrogen distribution based on time-dependent solar wind data, taking into account the expansion or contraction of heliospheric structure due to solar cycle fluctuations (Wang and Belcher, 1999; Richardson et al. 1995). Our novel computational techniques would be especially useful when applied in conjunction with hydrodynamic simulations such as those of Müller et al. (2000).

References

- Axford, W. I. and Suess, S. T., "The heliosphere," *EOS*, December (1994).
- Boonma, P. and Ruffolo, D., "Solution of the Boltzmann equation in one spatial dimension for hydrogen atoms in the heliosphere," *ANSCSE '99*, Bangkok, Thailand (1999): 154-158.
- Boonma, P., "Solution of the Boltzmann equation in one spatial dimension for hydrogen atoms in the heliosphere," *M.Sc. Thesis*, Department of Mathematics, Chulalongkorn University, Bangkok, Thailand (2000).
- Channok, C., Boonma, P., and Ruffolo, D. "Simulation of Boltzmann transport equation for charge exchange interaction between hydrogen atoms and protons in the heliosphere," *ANSCSE '2000*, Bangkok, Thailand (2000): 50-55.
- Cravens, T. E., *Physics of Solar System Plasmas*, 1st ed., Cambridge University Press, (1997): 9-263.
- Fahr, H.J., "The galactic environment of the Sun," *Astron. Astrophys.*, **241**, (1991): 251-259.
- Frisch, P. C., "Local interstellar oxygen in the heliosphere: Its analytic representation and observational consequences," *American Scientist*, Volume 88 (2000): 52-59.

- Gerald, C. F. and Wheatley, P. O., *Applied Numerical Analysis*, 5th ed., Addison-Wesley Publ. Co., (1994): 210–286.
- Harten, A., “High resolution schemes for hyperbolic conservation laws,” *J. Comput. Phys.*, **49**, (1983): 375.
- Hoffman, J. D., *Numerical Methods for Engineers and Scientists*, McGraw-Hill Press, (1993).
- Huang, K., *Statistical Mechanics*, 2nd ed., John Wiley & Sons, Inc. (1987): 52–84.
- Jokipii, J.R. and McDonald, F.B., *Scientific American*, April (1995): 58–63.
- Möbius, E.D., in *London–Bornstein Tables of Physics*, ed. H.H. Voigt (Berlin: Springer), 186 (1993).
- Müller, H.R., and Zank G.P., “Self-consistent hybrid simulations of the interaction of the heliosphere with the local interstellar medium,” *Solar Wind Nine*, edited by S. R. Habbal et al., AIP Conf. Proc. **417**, 819, (1999).
- Müller, H. R., Zank G. P., and Lipatov, A. S., “Self-consistent hybrid simulations of the interaction of the heliosphere with the local interstellar medium,” *J. Geophys. Res.*, **105**, (2000): 27419–27438.
- NASA, *Advanced Composition Explorer Report*, NP-1997(07)-021-GSFC, August (1997).
- Nutaro, T., Riyavong, S., and Ruffolo, D., “Application of a generalized total variation diminishing algorithm to cosmic ray transport and acceleration,” *Computer Physics Communications*, **134**, (2001): 209-222.

- Parks, G. K., *Physics of Space Plasmas*, Redwood City, CA: Addison-Wesley Publ. Co., (1991): 1–40.
- Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes in C*, New York: Cambridge University Press, (1988).
- Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., *Numerical Recipes in C*, 2nd ed., New York: Cambridge University Press, (1992).
- Richardson, J. W. et al., “Plasma in the outer heliosphere”, in *Solar Wind Eight*, edited by D. Winterhalter et al., AIP Conf. Proc. **382**, (1995): 51-55.
- Ruffolo, D., *Astrophys. J.*, **442**, (1995): 861.
- Roe, P. L., “Some contributions to the modeling of discontinuous flows,” Proc. AMS/SIAM *Sum. Sem. on Large-Scale Comp. in Fluid Mech.*, 1983, edited by B. E. Engquist et al., Providence, RI: Am. Math. Soc., (1995), 63.
- Schultz, D.R., Ovchinnikov, S.Yu., and Passovets, S.V., “Elastic and related cross sections for low-energy collisions among hydrogen and helium ions, neutrals, and isotopes,” *Atomic and Molecular Processes in Fusion Edge Plasma*, edited by Janev, R.K., New York: Plenum Press, (1995): 279–307.
- Tooprakai, P., “Modeling of magnetic loops on the west solar limb observed during the total solar eclipse of October 24, 1995”, *M.Sc. Thesis*, Department of Physics, Chulalongkorn University, Bangkok, Thailand (1999).
- Wang, C. and Belcher, W., “Detecting the termination shock and beyond”, in *Solar Wind Nine*, edited by S. R. Habbal et al., AIP Conf. Proc. **417**, (1999): 775–778.

Wang, Y.C., *J. Geophys. Res.*, **103**, (1998): 17419–17428.

Wang, Y.C., *Astrophys. J.*, **468**, (1996): 974–954.

Zank, G.P., “The Dynamical Heliosphere”, in *Solar Wind Nine*, edited by S. R. Habbal et al., AIP Conf. Proc. **417**, (1999): 783-786.

Zank, G. P., Lipanov, A.S., and Müller, H. R., “The interaction of heavy interstellar atoms with the heliosphere”, in *Solar Wind Nine*, edited by S. R. Habbal et al. AIP Conf. Proc. **417**, (1999): 811-814.

Zank G. P., Pauls, H.L., Williams L.L, and Hall, D.T., *J. Geophys. Res.*, **101**, (1996): 21693–21655.

<http://helios.gsfc.nasa.gov/heliosphere.html>

[http://www.afit.af.mil/ENGgraphics/annobib/STAGE/gewillia/
/joki95-00.gew.html](http://www.afit.af.mil/ENGgraphics/annobib/STAGE/gewillia/joki95-00.gew.html)

[http://web.mit.edu/afs/athena.mit.edu/org/s/space/www/helio.review/
/axford.suess.html](http://web.mit.edu/afs/athena.mit.edu/org/s/space/www/helio.review/axford.suess.html)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย



Appendices

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Appendix A

Cross Section of Charge Transfer between H and H⁺

When two particles a and b collide without a change of state, we term the event elastic scattering. A familiar example is the Rutherford scattering of two pointlike charges (e.g., proton-proton scattering), for which the differential cross section is

$$\frac{d\sigma}{d\Omega_{cm}}(E_{cm}, \theta_{cm}) = \left[\frac{q_a q_b}{4E_{cm} \sin^2 \theta_{cm}} \right]^2$$

(Schultz et al. 1995). Here θ_{cm} , Ω_{cm} , and E_{cm} are the center-of-mass (cm) scattering angle, solid angle, and collision energy, respectively, and q_a and q_b are the charges of the two ions. The non-relativistic relationship between the center-of-mass and laboratory collision energies is

$$E_{cm} = \frac{m_b}{m_a + m_b} E_{lab},$$

where in the laboratory a is the projectile and b is the target. The total elastic cross section can be written as

$$\sigma_{el} = \int \frac{d\sigma_{el}}{d\Omega} d\Omega = \int_0^{2\pi} \int_0^\pi \frac{d\sigma_{el}}{d\Omega} \sin \theta d\theta d\phi = 2\pi \int_0^\pi \frac{d\sigma_{el}}{d\Omega} \sin \theta d\theta.$$

In the case when either or both of the two particles have electronic structure [e.g., proton-hydrogen atom scattering (H⁺- H)], there is no exact analytic classical or quantum mechanical expression for the differential cross section. In fact, rather than being governed by a single interaction potential, the collision

system may have an adiabatic potential energy curve, representing the open electronic transition channels. Even at very low energies where not even the first excitation threshold can be reached, charge transfer between two identical or nearly identical nuclei can take place with a probability almost as large as that for elastic scattering. Thus, above the first excitation threshold, elastic scattering is but one of the competing channels in a possibly complicated multichannel scattering problem, while even below this energy, it may be necessary to solve at least a two-channel problem.

Knowledge of the elastic total and differential cross sections, computed by theoretical means or measured experimentally, is critical to modeling the transport of ions and neutrals in cool dense gas or plasma. However, other related cross sections are actually of greater practical use. They may be measured through various parameters of the gas or plasma transport or calculated from the elastic differential cross section.

Figure A.1 presents results from Schultz et al. (1995) for the following cross sections: elastic (σ_{el}), momentum transfer (σ_{mt}), viscosity (σ_{vi}), and charge transfer (= charge exchange; σ_{ct}). The center-of-mass collision energy is in the range $0.001 \leq E_{cm} \leq 100$ eV. This range accommodates the fact that often the cross section must be averaged over Maxwellian distributions so that knowledge of the cross section over a slightly larger range is useful. In addition, at the low end of this range comparisons can be made with data of astrophysical interest, thus providing an additional benchmark.

Energy values in Figure A.1 can be written in terms of the relative velocity of particles a and b from the relation $E_{cm} = \frac{1}{4}M|\vec{v}_a - \vec{v}_b|^2$, giving $|\vec{v}_a - \vec{v}_b|^2 = 4E_{cm}/M$.

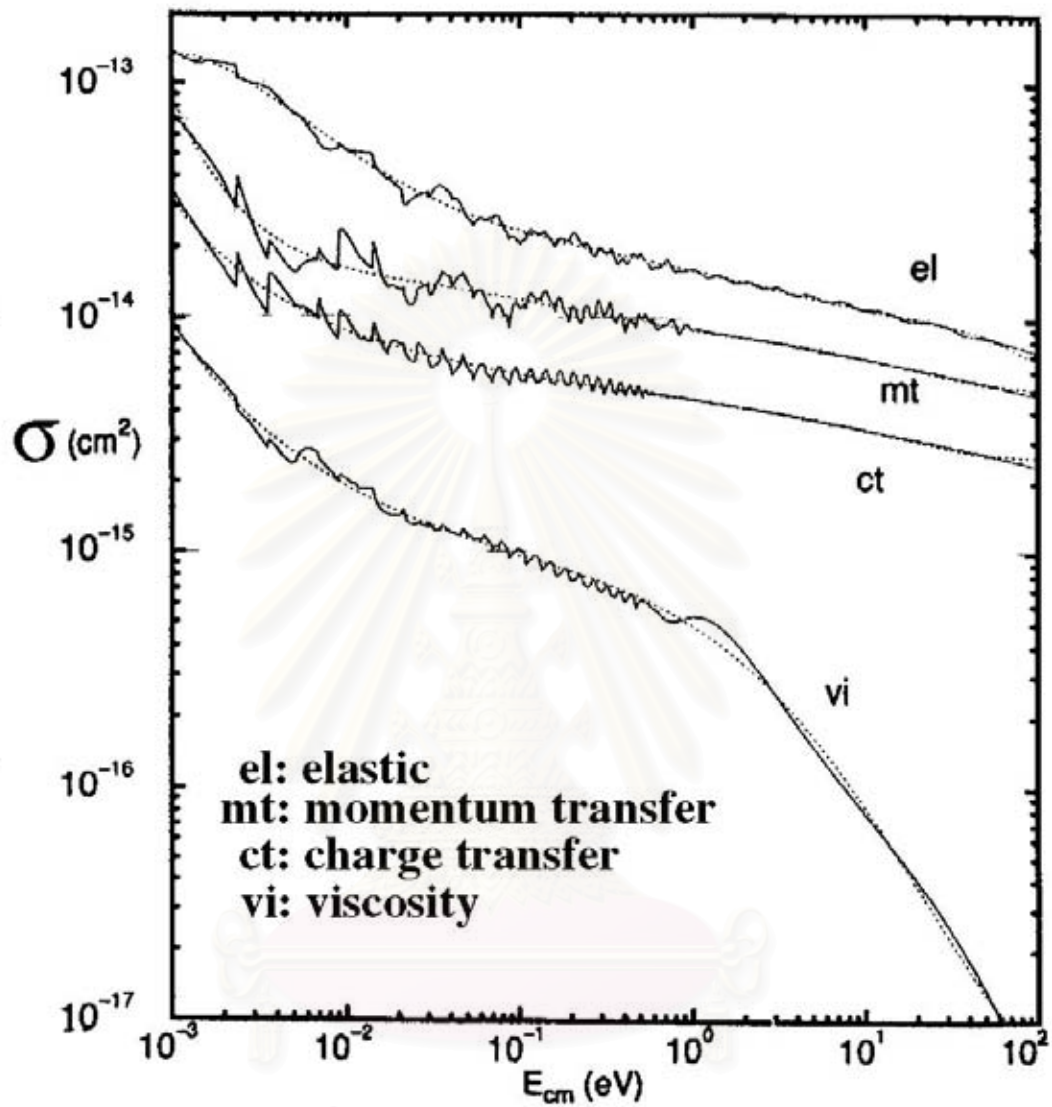


Figure A.1: The elastic, momentum transfer, viscosity, and charge transfer cross sections for proton-hydrogen charge exchange. We performed fits to the charge transfer cross section (simulation results from Schultz et al. 1995).

We fit the simulation data in Figure A.1 in terms of the relative velocity as

$$\sigma(|\vec{v}_a - \vec{v}_b|) \approx B + \beta \log(|\vec{v}_a - \vec{v}_b|), \quad (\text{A.1})$$

where B and β are constant coefficients fit to curve **ct** in Figure A.1, depending on energy in the center of mass frame (E_{cm}):

$$\text{when } E_{cm} \leq 10 \text{ eV: } \quad B = -13.09, \quad \beta = -0.2000$$

$$\text{and } E_{cm} > 10 \text{ eV: } \quad B = -12.63, \quad \beta = -0.2717.$$



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Appendix B

Solar Wind Proton Distribution

The solar wind protons distribution at each distance from 1 AU to 200 AU can be derived from previous hydrodynamic simulation results (G. Zank and H.L. Pauls, private communication, 1997) in terms of a Maxwellian distribution of varying density, velocity, and temperature as follows:

distance from the Sun (AU)	density (cm^{-3})	velocity (km/s)	temperature (K)
1.000	5.216E+00	400.000	1.07E+05
1.208	3.693E+00	403.000	8.48E+04
1.418	2.747E+00	406.900	6.96E+04
1.629	2.120E+00	409.600	5.86E+04
1.843	1.683E+00	411.600	5.04E+04
2.058	1.366E+00	413.000	4.41E+04
2.276	1.130E+00	414.100	3.91E+04
2.495	9.485E-01	415.000	3.51E+04
2.716	8.066E-01	415.600	3.19E+04
2.939	6.935E-01	416.200	2.92E+04
3.164	6.019E-01	416.600	2.71E+04
3.391	5.268E-01	416.900	2.53E+04
3.620	4.645E-01	417.200	2.38E+04
3.851	4.122E-01	417.400	2.25E+04
4.084	3.679E-01	417.600	2.15E+04
4.319	3.301E-01	417.800	2.07E+04
4.557	2.976E-01	417.900	2.00E+04
4.796	2.694E-01	418.000	1.95E+04
5.037	2.449E-01	418.000	1.91E+04
5.281	2.234E-01	418.100	1.87E+04
5.526	2.045E-01	418.100	1.85E+04
5.774	1.877E-01	418.100	1.84E+04
6.024	1.728E-01	418.100	1.83E+04
6.276	1.595E-01	418.100	1.84E+04
6.531	1.476E-01	418.100	1.84E+04

distance from the Sun (AU)	density (cm^{-3})	velocity (km/s)	temperature (K)
6.787	1.369E-01	418.000	1.86E+04
7.046	1.273E-01	418.000	1.87E+04
7.307	1.185E-01	417.900	1.90E+04
7.571	1.106E-01	417.900	1.92E+04
7.837	1.034E-01	417.800	1.95E+04
8.105	9.676E-02	417.700	1.99E+04
8.375	9.073E-02	417.600	2.03E+04
8.648	8.521E-02	417.600	2.07E+04
8.924	8.013E-02	417.500	2.12E+04
9.201	7.546E-02	417.400	2.16E+04
9.481	7.115E-02	417.200	2.22E+04
9.764	6.717E-02	417.100	2.27E+04
10.050	6.348E-02	417.000	2.33E+04
10.340	6.006E-02	416.900	2.39E+04
10.630	5.689E-02	416.800	2.45E+04
10.920	5.393E-02	416.600	2.51E+04
11.210	5.118E-02	416.500	2.58E+04
11.510	4.862E-02	416.300	2.64E+04
11.810	4.622E-02	416.200	2.71E+04
12.120	4.398E-02	416.100	2.79E+04
12.420	4.188E-02	415.900	2.86E+04
12.730	3.991E-02	415.700	2.94E+04
13.040	3.806E-02	415.600	3.02E+04
13.360	3.632E-02	415.400	3.09E+04
13.670	3.469E-02	415.200	3.18E+04
13.990	3.315E-02	415.100	3.26E+04
14.310	3.170E-02	414.900	3.34E+04
14.640	3.033E-02	414.700	3.43E+04
14.970	2.904E-02	414.500	3.52E+04
15.300	2.782E-02	414.400	3.61E+04
15.630	2.667E-02	414.200	3.70E+04
15.970	2.558E-02	414.000	3.79E+04
16.310	2.454E-02	413.800	3.89E+04
16.650	2.356E-02	413.600	3.98E+04
17.000	2.263E-02	413.400	4.08E+04
17.350	2.175E-02	413.200	4.18E+04
17.700	2.091E-02	413.000	4.28E+04
18.050	2.011E-02	412.800	4.38E+04
18.410	1.935E-02	412.500	4.48E+04
18.770	1.863E-02	412.300	4.59E+04
19.140	1.794E-02	412.100	4.69E+04
19.500	1.728E-02	411.900	4.80E+04
19.870	1.665E-02	411.700	4.91E+04

distance from the Sun (AU)	density (cm^{-3})	velocity (km/s)	temperature (K)
20.250	1.606E-02	411.400	5.02E+04
20.630	1.549E-02	411.200	5.13E+04
21.010	1.494E-02	411.000	5.24E+04
21.390	1.442E-02	410.700	5.35E+04
21.780	1.392E-02	410.500	5.47E+04
22.170	1.345E-02	410.200	5.58E+04
22.560	1.299E-02	410.000	5.70E+04
22.960	1.255E-02	409.700	5.82E+04
23.360	1.214E-02	409.500	5.94E+04
23.770	1.173E-02	409.200	6.06E+04
24.180	1.135E-02	409.000	6.18E+04
24.590	1.098E-02	408.700	6.31E+04
25.000	1.063E-02	408.400	6.43E+04
25.420	1.029E-02	408.200	6.56E+04
25.840	9.962E-03	407.900	6.68E+04
26.270	9.649E-03	407.600	6.81E+04
26.700	9.347E-03	407.400	6.94E+04
27.140	9.058E-03	407.100	7.07E+04
27.570	8.779E-03	406.800	7.21E+04
28.020	8.511E-03	406.500	7.34E+04
28.460	8.253E-03	406.200	7.48E+04
28.910	8.005E-03	405.900	7.61E+04
29.360	7.766E-03	405.600	7.75E+04
29.820	7.536E-03	405.300	7.89E+04
30.280	7.314E-03	405.000	8.03E+04
30.750	7.100E-03	404.700	8.17E+04
31.220	6.893E-03	404.400	8.31E+04
31.690	6.694E-03	404.100	8.45E+04
32.170	6.502E-03	403.800	8.59E+04
32.650	6.317E-03	403.500	8.74E+04
33.140	6.138E-03	403.200	8.88E+04
33.630	5.965E-03	402.800	9.03E+04
34.130	5.798E-03	402.500	9.18E+04
34.630	5.637E-03	402.200	9.33E+04
35.130	5.481E-03	401.900	9.48E+04
35.640	5.330E-03	401.500	9.63E+04
36.150	5.185E-03	401.200	9.78E+04
36.670	5.044E-03	400.900	9.93E+04
37.190	4.908E-03	400.500	1.01E+05
37.720	4.776E-03	400.200	1.02E+05
38.250	4.648E-03	399.800	1.04E+05
38.780	4.525E-03	399.500	1.06E+05

distance from the Sun (AU)	density (cm^{-3})	velocity (km/s)	temperature (K)
39.320	4.405E-03	399.100	1.07E+05
39.870	4.290E-03	398.800	1.09E+05
40.420	4.178E-03	398.400	1.10E+05
40.970	4.069E-03	398.100	1.12E+05
41.530	3.964E-03	397.700	1.14E+05
42.100	3.862E-03	397.300	1.15E+05
42.670	3.763E-03	397.000	1.17E+05
43.240	3.667E-03	396.600	1.19E+05
43.820	3.574E-03	396.200	1.20E+05
44.410	3.484E-03	395.800	1.22E+05
45.000	3.397E-03	395.400	1.24E+05
45.590	3.312E-03	395.000	1.25E+05
46.190	3.230E-03	394.600	1.27E+05
46.800	3.150E-03	394.200	1.29E+05
47.410	3.073E-03	393.800	1.31E+05
48.030	2.997E-03	393.400	1.33E+05
48.650	2.924E-03	393.000	1.34E+05
49.270	2.854E-03	392.600	1.36E+05
49.910	2.785E-03	392.200	1.38E+05
50.550	2.718E-03	391.800	1.40E+05
51.190	2.653E-03	391.300	1.42E+05
51.840	2.590E-03	390.900	1.44E+05
52.490	2.528E-03	390.500	1.46E+05
53.160	2.469E-03	390.000	1.48E+05
53.820	2.411E-03	389.600	1.50E+05
54.500	2.355E-03	389.100	1.52E+05
55.170	2.300E-03	388.600	1.54E+05
55.860	2.246E-03	388.200	1.56E+05
56.550	2.195E-03	387.700	1.58E+05
57.250	2.144E-03	387.200	1.60E+05
57.950	2.095E-03	386.700	1.62E+05
58.660	2.048E-03	386.200	1.64E+05
59.370	2.001E-03	385.700	1.66E+05
60.100	1.956E-03	385.200	1.68E+05
60.820	1.912E-03	384.700	1.71E+05
61.560	1.870E-03	384.200	1.73E+05
62.300	1.828E-03	383.600	1.75E+05
63.050	1.787E-03	383.100	1.77E+05
63.800	1.748E-03	382.500	1.80E+05
64.560	1.710E-03	382.000	1.82E+05
65.330	1.672E-03	381.400	1.84E+05

distance from the Sun (AU)	density (cm^{-3})	velocity (km/s)	temperature (K)
66.100	1.636E-03	380.800	1.87E+05
66.880	1.600E-03	380.300	1.89E+05
67.670	1.566E-03	379.700	1.91E+05
68.470	1.532E-03	379.100	1.94E+05
69.270	1.499E-03	378.500	1.96E+05
70.080	1.467E-03	377.900	1.99E+05
70.890	1.436E-03	377.200	2.02E+05
71.710	1.406E-03	376.600	2.04E+05
72.550	1.376E-03	376.000	2.07E+05
73.380	1.347E-03	375.300	2.10E+05
74.230	1.319E-03	374.700	2.12E+05
75.080	1.291E-03	374.000	2.15E+05
75.940	1.265E-03	373.300	2.18E+05
76.810	1.239E-03	372.700	2.20E+05
77.680	1.213E-03	372.000	2.23E+05
78.570	1.188E-03	371.300	2.26E+05
79.460	1.164E-03	370.600	2.29E+05
80.350	1.140E-03	369.900	2.32E+05
81.260	1.117E-03	369.100	2.35E+05
82.170	1.094E-03	368.400	2.37E+05
83.100	1.073E-03	367.700	2.40E+05
84.030	1.048E-03	367.000	2.43E+05
84.970	1.037E-03	366.300	2.47E+05
85.910	1.100E-03	360.600	2.67E+05
86.870	1.359E-03	322.900	4.37E+05
87.830	2.012E-03	237.900	9.63E+05
88.800	3.098E-03	147.200	1.57E+06
89.780	3.440E-03	101.000	1.74E+06
90.770	3.470E-03	93.770	1.74E+06
91.770	3.492E-03	89.910	1.74E+06
92.780	3.517E-03	85.980	1.74E+06
93.790	3.542E-03	82.030	1.75E+06
94.820	3.567E-03	78.210	1.75E+06
95.850	3.592E-03	74.520	1.75E+06
96.900	3.615E-03	70.930	1.75E+06
97.950	3.638E-03	67.440	1.75E+06
99.010	3.659E-03	64.050	1.74E+06
100.100	3.682E-03	60.750	1.74E+06
101.200	3.705E-03	57.530	1.74E+06
102.200	3.728E-03	54.390	1.73E+06
103.300	3.752E-03	51.340	1.73E+06

distance from the Sun (AU)	density (cm^{-3})	velocity (km/s)	temperature (K)
104.500	3.776E-03	48.350	1.72E+06
105.600	3.801E-03	45.440	1.72E+06
106.700	3.828E-03	42.590	1.71E+06
107.800	3.857E-03	39.810	1.70E+06
109.000	3.887E-03	37.090	1.69E+06
110.100	3.920E-03	34.420	1.68E+06
111.300	3.957E-03	31.820	1.66E+06
112.500	3.996E-03	29.270	1.65E+06
113.700	4.041E-03	26.790	1.63E+06
114.900	4.091E-03	24.360	1.61E+06
116.100	4.148E-03	22.000	1.59E+06
117.300	4.213E-03	19.690	1.57E+06
118.500	4.290E-03	17.450	1.54E+06
119.800	4.380E-03	15.270	1.51E+06
121.000	4.490E-03	13.170	1.47E+06
122.300	4.625E-03	11.140	1.43E+06
123.600	4.795E-03	9.195	1.37E+06
124.900	5.019E-03	7.341	1.31E+06
126.200	5.330E-03	5.588	1.23E+06
127.500	5.806E-03	3.942	1.13E+06
128.800	6.713E-03	2.414	9.78E+05
130.100	9.444E-03	1.146	6.94E+05
131.500	1.011E-01	0.239	6.46E+04
132.800	1.854E-01	-0.367	3.49E+04
134.200	1.969E-01	-0.795	3.25E+04
135.600	2.050E-01	-1.182	3.08E+04
137.000	2.111E-01	-1.537	2.96E+04
138.400	2.159E-01	-1.868	2.86E+04
139.800	2.198E-01	-2.177	2.77E+04
141.200	2.229E-01	-2.468	2.70E+04
142.700	2.254E-01	-2.746	2.64E+04
144.100	2.275E-01	-3.010	2.58E+04
145.600	2.292E-01	-3.265	2.53E+04
147.100	2.307E-01	-3.510	2.49E+04
148.600	2.318E-01	-3.747	2.45E+04
150.100	2.328E-01	-3.977	2.41E+04
151.600	2.336E-01	-4.200	2.38E+04
153.100	2.343E-01	-4.418	2.34E+04
154.700	2.347E-01	-4.631	2.31E+04
156.200	2.352E-01	-4.838	2.28E+04
157.800	2.355E-01	-5.040	2.26E+04

distance from the Sun (AU)	density (cm^{-3})	velocity (km/s)	temperature (K)
159.400	2.358E-01	-5.238	2.23E+04
161.000	2.360E-01	-5.433	2.21E+04
162.600	2.361E-01	-5.623	2.19E+04
164.200	2.362E-01	-5.810	2.17E+04
165.900	2.363E-01	-5.994	2.15E+04
167.500	2.363E-01	-6.175	2.13E+04
169.200	2.363E-01	-6.353	2.11E+04
170.900	2.363E-01	-6.528	2.09E+04
172.600	2.363E-01	-6.700	2.08E+04
174.300	2.363E-01	-6.870	2.06E+04
176.100	2.362E-01	-7.038	2.05E+04
177.800	2.362E-01	-7.202	2.03E+04
179.600	2.361E-01	-7.366	2.02E+04
181.300	2.361E-01	-7.527	2.00E+04
183.100	2.360E-01	-7.687	1.99E+04
185.000	2.359E-01	-7.845	1.98E+04
186.800	2.358E-01	-8.002	1.97E+04
188.600	2.356E-01	-8.158	1.95E+04
190.500	2.355E-01	-8.313	1.94E+04
192.400	2.354E-01	-8.466	1.93E+04
194.200	2.353E-01	-8.618	1.91E+04
196.200	2.352E-01	-8.770	1.90E+04
198.100	2.351E-01	-8.920	1.89E+04
200.000	2.350E-01	-9.068	1.87E+04

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

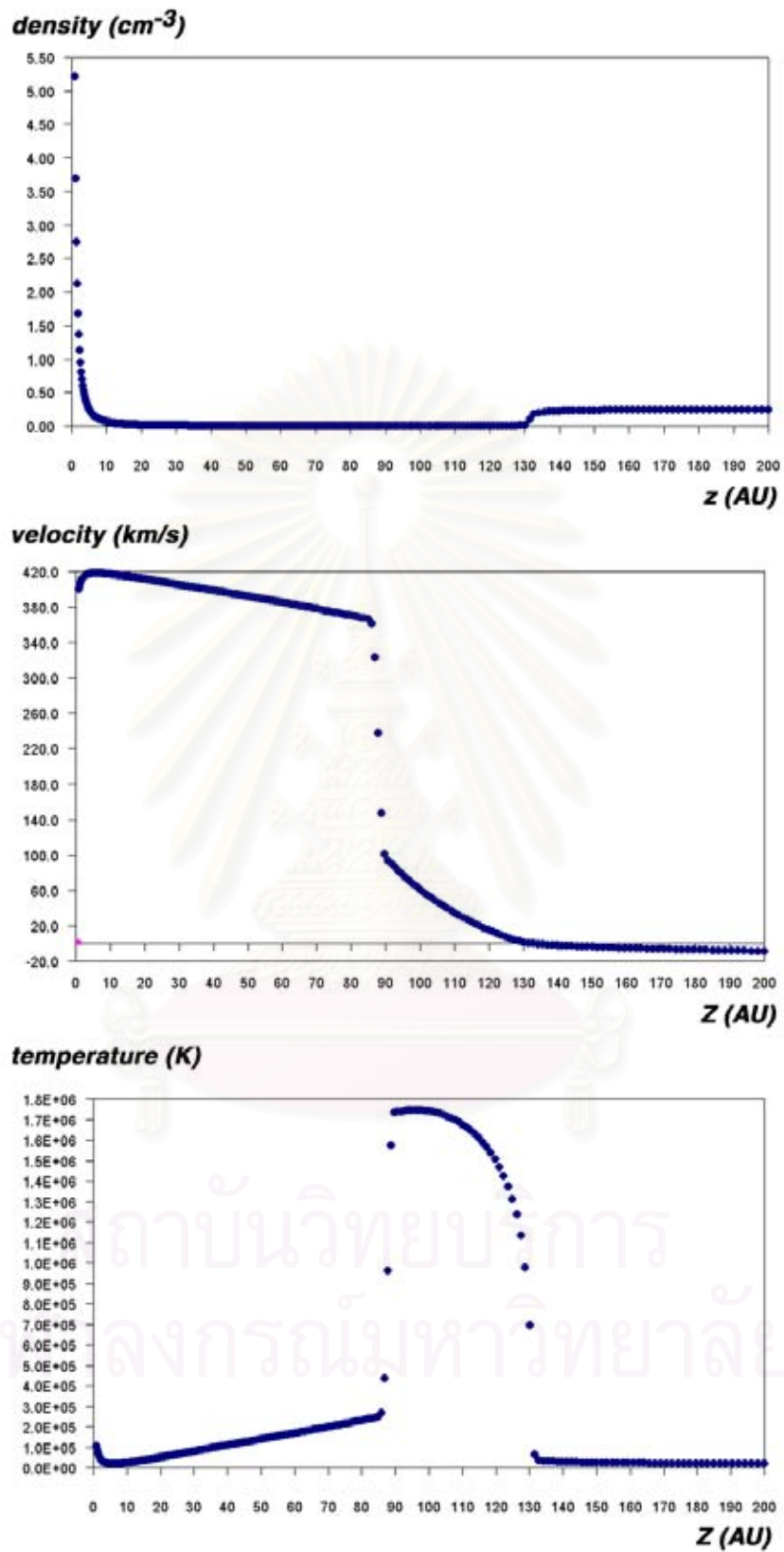


Figure B.1: Graph of proton density, velocity, and temperature versus distance from the Sun (z).

Appendix C

Source Code

Boltz.c

```

/* adapt on interpolate case on Oct 10, 2000

New Lastest update for interpolate case

boltz.c -- November 25, 2000

Modified from wind.c (t) of February 1, 1997 to solve the
Boltzmann equation for neutral Hydrogen in the outer heliosphere.

The processes we consider at this point are:
Necessary files and subroutines:

    boltz.c    main, arctan, grid_t, grid_p, interpo, sub_interpo,
               polint, interpo_t, interpo_t, interpo_f
    chex.c     chex, protondata, fact, doublefact, alphafact, velocity_term
    initial.c  initial
    nrutil.c   nrerror, dvector, ivector, dmatrix, darray,
               free_dvector, free_ivector, free_dmatrix, free_darray
    printout.c printout
    stream.c   stream, tvdinflow, gengam

Variables input from the user:

    starttime  Initial value of time   (days)
    stoptime   Final value of time     (days)
    timestep   Time step                (days)
    printtime  Printing interval       (days)
    length     Length of simulation region (AU)
    zstep      Spatial step slong solar apex (AU)
    printextra Print extra diagnostic information? (0/1)

-----
Chanruangrit Channok, Panita Boonma, Worachate Boonplod and David Ruffolo
Department of Physics
Faculty of Science
Chulalongkorn University
Bangkok 10330, THAILAND
-----
*/

#include <math.h>
#include <stdio.h>

#define M 938780000.0 /* Mass of a hydrogen atom (eV/c^2) */
#define DUMP 1 /* (0 = DON'T DUMP), (1 = DUMP) */

double ***f, ***gam;
double *dens, *px, *temp;
double *p, *mu, *mu0, *mu1, *theta, *zstep;

```

```

int      *fine, np, nz, nmu0, nmu1;

main()
{
  FILE    *f_dump;
  double  starttime, stoptime, printtime, timestep, time, nextprint;
  double  *beta, *ke, length;
  int     nmu, printextra, u, w, l;

  double  ***darray2(), *dvector();
  void    free_darray2(), free_dvector(), nrerror();
  void    initial(), printout(), gen_gam(), stream(), chex(), photoionization(),
  print_fp(), test_f(), grid_t(), grid_p(), protondata();

  /* Input parameters from the user */
  printf(" ----- \n");
  printf(" Hello! Welcome to Boltzmann Transport Simulation Project. \n");
  printf(" ----- \n");
  printf(" \n Please input the following parameters:\n");

  printf("\n Starting value of time (days): ");
  scanf("%lf",&starttime);
  printf("\n Final value of time (days): ");
  scanf("%lf",&stoptime);
  printf("\n Time step (days): ");
  scanf("%lf",&timestep);
  printf("\n Time interval after which to print out data (days): ");
  scanf("%lf",&printtime);
  printf("\n Length in the z-direction (AU): ");
  scanf("%lf",&length);
  printf("\n Step in the z-direction (AU): ");
  scanf("%lf",&zstep);

  /* find p[w] */
  grid_p(); /* set np */
  printf("\n number of momentum grid (np) = %d ",np);
  beta = dvector(0,np);
  ke = dvector(0,np);
  for(w=0;w<=np;w++)
    printf("\n p[w=%2d] = %12.4lf eV/C   fine[w=%2d] = %d",w,p[w],w,fine[w]);

  printf("\n Do you want to print extra diagnostic information ? ");
  printf("\n   Enter 1 for <Yes>, 0 for <No> ");
  scanf("%d",&printextra);

  /* Calculating kinetic energy[w] and beta[w]. */
  ke[0]=0.0;  beta[0]=0.0;
  for (w=0;w<=np;w++) {
    ke[w] = sqrt(p[w]*p[w] + M*M) - M;
    beta[w] = p[w]/(ke[w] + M);
    if (printextra)
      printf("\n boltz.c: ke[%2d]=%13.5lf, beta[%2d]=%11.8lf",w,ke[w],w,beta[w]);
  }
  printf("\n");

  /* Calculating nz, zstep; length is now rounded to be integral multiple of zstep. */

  nz=(length/zstep)+0.5;
  if(nz==0) nrerror("boltz:nz=0");
  printf("\n input : nz(double)=%lf, nz(int)=%d, length=%lf",length/zstep,nz,nz*zstep);
  length = nz*zstep;

```

```

printf("\n nz=%d : for use in array temp[1...nz] dens[1...nz] px[1...nz] \n",nz);
temp = dvector(1,nz);
dens = dvector(1,nz);
px = dvector(1,nz);

grid_t(); /* set nmu0 & nmu1 */
for(u=0;u<=nmu0+nmu1;u++) printf(" theta[%2d] = %6.2lf \n",u,theta[u]);

/* Echoing */
printf("\n Your input the following parameters:\n");
printf("\n Starting time (days) : %12lf",starttime);
printf("\n Final time (days) : %12lf",stoptime);
printf("\n Time step (days) : %12lf",timestep);
printf("\n Print interval (days) : %12lf",printtime);
printf("\n Length (AU) : %12lf",length);
printf("\n Step in z-direction (AU) : %12lf",zstep);
printf("\n Printextra (0 & 1) : %5d\n",printextra);

/* Idiot Proofing */
if (stoptime < starttime) nrerror("boltz: stoptime < starttime");
if (timestep <= 0) nrerror("boltz: timestep <= 0");
if (printtime < timestep) nrerror("boltz: printtime < timestep");
if (length <= 0) nrerror("boltz: length <= 0");
if (zstep <= 0) nrerror("boltz: zstep <= 0");

/* Defining array for f */
nmu=nmu0>nmu1 ? nmu0:nmu1;
f = darray2(0,np,1,nz,0,nmu);
printf("\n boltz.c: array of f(0-%d,1-%d,0-%d)",np,nz,nmu);

if (printextra) printf("\n Now we are here step ONE \n");

/* Start Programing Calculation */
initial(); /* initial for f[w][l][u] */
printf("\n Start for print test f from initial() \n");
test_f(); /* print value f[w][l][u] for test */
printout(printtime,starttime);

protondata(nz,zstep);
gen_gam(timestep,beta);

/* FOR EACH TIME STEP:
- Inject new flux (IF NECESSARY).
- Calculate f at the new time step, according to the transport equation
Print out the data (IF NECESSARY).
*/

nextprint = starttime + printtime;

for (time=starttime;time+timestep<=stoptime+timestep/2;time+=timestep) {
printf("\n Loop of time t = %lf \n",time);

chex(timestep/2,beta);

/* ## No need for streaming when [w=0] because p=0 and therefore v_z=0. */
printf(" \n Streaming Process \n");
for(w=1;w<=np;w++) stream(time,timestep,w);

photoionization(timestep);

chex(timestep/2.,beta);

```



```

    if (time+timestep >= nextprint-0.01*timestep) {
        printf("\n\n time = %.2lf days \n",time+timestep);

        printf("\n ***** test_f after change of time ***** \n");
        test_f();
        printout(printtime,time+timestep);
        nextprint += printtime;
    }
}

/* "DUMP" out f, SO THAT THE RUN MAY BE CONTINUED. < 0 or 1 > */
if(DUMP) {
    f_dump = fopen("dump.dat","w");
    printf("\n\n Now dumping f(p[w],z[l],mu[u])
        for a later run... time final = %.2lf day \n",time);

    /* w=0 */
    for (l=1;l<=nz;l++) fprintf(f_dump,"\n f[0][%3d][0]=%12le ",l,f[0][l][0]);
    /* w>0 */
    for(w=1;w<=np;w++) {
        if(fine[w]==1) nmu=nmul;
        else nmu=nmu0;
        for(l=1;l<=nz;l++) {
            for(u=0;u<=nmu;u++) {
                fprintf(f_dump,"\n f[%2d][%3d][%2d]=%12le ",w,l,u,f[w][l][u]);
            }
        }
    }
    fclose(f_dump);
}

free_dvector(theta,0);
free_dvector(mu0,0);
free_dvector(mu1,0);
free_dvector(p,0);
free_dvector(ke,0);
free_dvector(beta,0);
free_ivector(fine,0);
free_darray2(f,0,np,1,nz,0);
free_dvector(dens,1);
free_dvector(px,1);
free_dvector(temp,1);

printf("\n ----- :o                      ** E N D **                      :o ---- \n");
printf("\n ----- Finish Boltz.c ----- \n");
}

/* test_f : for print test data of f[w][l][u] */

void test_f()
{
    int u, w, l, nmu;

    /* for w=0 */
    for(l=1;l<=nz;l++) {
        /* printf("\n f[w=0][l=%3d][u=0] = %1e ",l,f[0][l][0]); */
    }

    /* for w>=1 */
    for(w=1;w<=np;w++) {
        if(fine[w]==0) nmu=nmu0;
        else nmu=nmul;
    }
}

```

```

        for(l=1;l<=nz;l++) {
            for(u=0;u<=nmu;u++)
                printf("\n f[w=%2d][l=%3d][u=%2d] = %1e ",w,l,u,f[w][l][u]);
        }
    }
}

```

```

/* This function discretizes theta[0.....13] from 0 - 180 degree
   and discretize mu0[0....11] & mu1[0....4]
*/

```

```

void grid_t()

```

```

{
    double pi, *dvector();
    pi=4.0*atan(1.0);
    nmu0 = 11; nmu1 = 4;
    mu0 =dvector(0,nmu0);
    mu1 =dvector(0,nmu1);
    theta=dvector(0,nmu0+nmu1);

    theta[0] = 0.0;
    theta[1] = 1.0;
    theta[2] = 2.0;
    theta[3] = 4.0;
    theta[4] = 6.0;
    theta[5] = 15.0;
    theta[6] = 30.0;
    theta[7] = 60.0;
    theta[8] = 80.0; /* add new grid */
    theta[9] = 90.0;
    theta[10] = 100.0; /* add new grid */
    theta[11] = 120.0;
    theta[12] = 150.0;
    theta[13] = 165.0;
    theta[14] = 175.0;
    theta[15] = 180.0;

    mu0[0]=cos( theta[0]*pi/180.);
    mu0[1]=cos( theta[5]*pi/180.);
    mu0[2]=cos( theta[6]*pi/180.);
    mu0[3]=cos( theta[7]*pi/180.);
    mu0[4]=cos( theta[8]*pi/180.);
    mu0[5]=cos( theta[9]*pi/180.);
    mu0[6]=cos(theta[10]*pi/180.);
    mu0[7]=cos(theta[11]*pi/180.);
    mu0[8]=cos(theta[12]*pi/180.);
    mu0[9]=cos(theta[13]*pi/180.);
    mu0[10]=cos(theta[14]*pi/180.);
    mu0[11]=cos(theta[15]*pi/180.);

    mu1[0]=cos(theta[0]*pi/180.);
    mu1[1]=cos(theta[1]*pi/180.);
    mu1[2]=cos(theta[2]*pi/180.);
    mu1[3]=cos(theta[3]*pi/180.);
    mu1[4]=cos(theta[4]*pi/180.);
}

```

```

/* This function discretizes momentum p[0.....24] */

```

```

void grid_p()

```

```

{

```

```

double a, *dvector();
int    *ivector();

np = 24;
p = dvector(0,np);
fine = ivector(0,np);
/*-----fine[...] = 0 for course grid-----*/
/*-----fine[...] = 1 for fine grid-----*/

a=3136.125; /* factor for change velocity(km/s) to momentum(eV/C) */

p[0] = (0.*a);    fine[0] = 0;
p[1] = (10.*a);   fine[1] = 0;
p[2] = (15.*a);   fine[2] = 0;
p[3] = (20.*a);   fine[3] = 0;
p[4] = (25.*a);   fine[4] = 0;
p[5] = (30.*a);   fine[5] = 0;
p[6] = (40.*a);   fine[6] = 0;
p[7] = (60.*a);   fine[7] = 0;
p[8] = (100.*a);  fine[8] = 0;
p[9] = (140.*a);  fine[9] = 0;
p[10] = (180.*a); fine[10] = 0;
p[11] = (220.*a); fine[11] = 0;
p[12] = (260.*a); fine[12] = 0;
p[13] = (310.*a); fine[13] = 0;
p[14] = (320.*a); fine[14] = 1;
p[15] = (340.*a); fine[15] = 1;
p[16] = (360.*a); fine[16] = 1;
p[17] = (380.*a); fine[17] = 1;
p[18] = (395.*a); fine[18] = 1;
p[19] = (400.*a); fine[19] = 0;
p[20] = (410.*a); fine[20] = 1;
p[21] = (420.*a); fine[21] = 1;
p[22] = (430.*a); fine[22] = 1;
p[23] = (450.*a); fine[23] = 1;
p[24] = (500.*a); fine[24] = 0;
}

/* Program interpo.c 15 Feb 2000

The output is the *log base 10* of f for momentum pp and angle theta.
An output of -999 means that f is zero.
Note: we must have a value of theta[] (the last one) as 180.0;
*/

double interpo(pp,l,th)
int l;
double pp,th;
{
double lf00, lf01, lf10, lf11, lfp0, lfp1, lftH, lftL, output;
double frac, f00, f01, f10, f11;
int w, w1, u, u0, u1, angle0, angle1;

double fourpt();
void nrrerror();

if (th > 180.0) nrrerror("interpo: bad th");

/*----- case 1: coarse grid, p near 0, use log-linear p-interpolation-----*/
if (pp>=0 && pp<=p[1]) {
if (th <= theta[5]) {

```

```

    u0=0;      u1=1;
    angle0=0; angle1=5;
} else {
    for (u0=1;theta[u0+5]<th;u0++);
    u1=u0+1;
    angle0=u0+4; angle1=u0+5;
}

/* First, log-linear interpolation in theta at p[1]. */

if (f[1][1][u0]*f[1][1][u1]*f[0][1][0] == 0) return -999.;
else {
    frac = (th-theta[angle0])/(theta[angle1]-theta[angle0]);
    lfp1 = (1-frac)*log10(f[1][1][u0]) + frac*log10(f[1][1][u1]);

    /* Now, log-linear interpolation in p, from p[0]=0 to p[1]. */

    frac = pp/p[1];
    output = (1-frac)*log10(f[0][1][0]) + frac*lfp1;
    return output;
}
}

/*----- case 2 -----*/
else if((pp>p[1]&&pp<=p[13])) {
    for (w=1;p[w+1]<pp;w++);
    if (w<1 || w>12) nrerror("interpo: bad w in case 2");
    if (th <= theta[5]) {
        u0=0;      u1=1;
        angle0=0; angle1=5;
    } else {
        for (u0=1;theta[5+u0]<th;u0++);
        u1=u0+1;
        angle0=u0+4; angle1=u0+5;
    }
    if (f[w][1][u0]*f[w][1][u1]*f[w+1][1][u0]*f[w+1][1][u1] == 0) {
        return -999.;
    } else {
        lf00 = log10(f[w][1][u0]);
        lf01 = log10(f[w][1][u1]);
        lf10 = log10(f[w+1][1][u0]);
        lf11 = log10(f[w+1][1][u1]);
        return fourpt(p[w],p[w+1],pp,theta[angle0],theta[angle1],th,lf00,lf01,lf10,lf11);
    }
}

/*----- case 3 -----*/
else if((pp>p[14]&&pp<=p[23]&&th<=theta[4]&&!(pp>p[18]&&pp<=p[20]&&th<=theta[1])) {
    if(pp>p[18] && pp<=p[20]) {
        w=18; w1=20;
        for (u=0;theta[u+1]<th;u++);
        if (f[w][1][u]*f[w][1][u+1]*f[w1][1][u]*f[w1][1][u+1] == 0) {
            return -999.;
        } else {
            lf00 = log10(f[w][1][u]);
            lf01 = log10(f[w][1][u+1]);
            lf10 = log10(f[w1][1][u]);
            lf11 = log10(f[w1][1][u+1]);
            return fourpt(p[w],p[w1],pp,theta[u],theta[u+1],th,lf00,lf01,lf10,lf11);
        }
    } else {
        for(w=14;p[w+1]<pp;w++);
    }
}

```

```

for (u=0;theta[u+1]<th;u++);
if (f[w][l][u]*f[w][l][u+1]*f[w+1][l][u]*f[w+1][l][u+1] == 0) {
    return -999.;
} else {
    lf00 = log10(f[w][l][u]);
    lf01 = log10(f[w][l][u+1]);
    lf10 = log10(f[w+1][l][u]);
    lf11 = log10(f[w+1][l][u+1]);
    return fourpt(p[w],p[w+1],pp,theta[u],theta[u+1],th,lf00,lf01,lf10,lf11);
}
}

/*-----case 4-----*/
else if((pp>p[13])&&(pp<=p[19]) && (th>theta[5])) {
    w=13; w1=19;
    for (u=1;theta[u+5]<th;u++);
    if (f[w][l][u]*f[w][l][u+1]*f[w1][l][u]*f[w1][l][u+1] == 0) {
        return -999;
    } else {
        lf00 = log10(f[w][l][u]);
        lf01 = log10(f[w][l][u+1]);
        lf10 = log10(f[w1][l][u]);
        lf11 = log10(f[w1][l][u+1]);
        return fourpt(p[w],p[w1],pp,theta[u+4],theta[u+5],th,lf00,lf01,lf10,lf11);
    }
}

/*-----case 5-----*/
else if((pp>p[19])&&(pp<=p[24]) && (th>theta[5])) {
    w=19; w1=24;
    for (u=1;theta[u+5]<th;u++);
    if (f[w][l][u]*f[w][l][u+1]*f[w1][l][u]*f[w1][l][u+1] == 0) {
        return -999;
    } else {
        lf00 = log10(f[w][l][u]);
        lf01 = log10(f[w][l][u+1]);
        lf10 = log10(f[w1][l][u]);
        lf11 = log10(f[w1][l][u+1]);
        return fourpt(p[w],p[w1],pp,theta[u+4],theta[u+5],th,lf00,lf01,lf10,lf11);
    }
}

/*-----case 6-----*/
else if((pp>p[13])&&(pp<=p[14]) && (th<=theta[4])) {
    w=13; w1=14;
    for (u=0;theta[u+1]<th;u++);
    if (f[w][l][0]*f[w][l][1]*f[w1][l][u]*f[w1][l][u+1] == 0) {
        return -999;
    } else {
        /* First, interpolate in theta at w. */
        frac = th/theta[5];
        lfp0 = (1-frac)*log10(f[w][l][0]) + frac*log10(f[w][l][1]);

        /* Next, interpolate in theta at w1. */
        frac = (th-theta[u])/(theta[u+1]-theta[u]);
        lfp1 = (1-frac)*log10(f[w1][l][u]) + frac*log10(f[w1][l][u+1]);

        /* Now, log-log interpolation in p. */
        frac = log10(pp/p[w])/log10(p[w1]/p[w]);
        output = (1-frac)*lfp0 + frac*lfp1;
    }
}

```

```

        return output;
    }

}

/*-----case 7-----*/
else if((pp>p[23])&&(pp<=p[24]) && (th<=theta[4])) {
    w=23; w1=24;
    for (u=0;theta[u+1]<th;u++);
    if (f[w][1][u]*f[w][1][u+1]*f[w1][1][0]*f[w1][1][1] == 0) {
        return -999;
    } else {

        /* First, interpolate in theta at w1. */
        frac = th/theta[5];
        lfp1 = (1-frac)*log10(f[w1][1][0]) + frac*log10(f[w1][1][1]);

        /* Next, interpolate in theta at w. */
        frac = (th-theta[u])/(theta[u+1]-theta[u]);
        lfp0 = (1-frac)*log10(f[w][1][u]) + frac*log10(f[w][1][u+1]);

        /* Now, log-log interpolation in p. */
        frac = log10(pp/p[w])/log10(p[w1]/p[w]);
        output = (1-frac)*lfp0 + frac*lfp1;
        return output;
    }
}

/*-----case 8-----*/
else if((pp>p[13])&&(pp<=p[19]) && (th>theta[4])&&(th<=theta[5])) {
    if (f[13][1][1]*f[19][1][1] == 0) {
        return -999;
    } else {
        frac = log10(pp/p[13])/log10(p[19]/p[13]);
        lftH = (1-frac)*log10(f[13][1][1]) + frac*log10(f[19][1][1]);
    }
}
if(pp<p[14]) {
    if (f[13][1][0]*f[14][1][4] == 0) {
        return -999;
    }
    else {
        frac = th/theta[5];
        lft0 = (1-frac)*log10(f[13][1][0]) + frac*log10(f[13][1][1]);

        frac = log10(pp/p[13])/log10(p[14]/p[13]);
        lftL = (1-frac)*lft0 + frac*log10(f[14][1][4]);

        /* Second, interpolate in theta (use linear) */
        frac = (th-theta[4])/(theta[5]-theta[4]);
        output = (1-frac)*lftL + frac*lftH;
        return output;
    }
}
else if(pp>=p[14] && pp<=p[18]) {
    for (w=14;p[w+1]<pp;w++);
    if (f[w][1][4]*f[w+1][1][4] == 0) {
        return -999;
    }
    else {
        frac = log10(pp/p[w])/log10(p[w+1]/p[w]);
        lftL = (1-frac)*log10(f[w][1][4]) + frac*log10(f[w+1][1][4]);
    }
}

```

```

/* Second, interpolate in theta (use linear) */
frac = (th-theta[4])/(theta[5]-theta[4]);
output = (1-frac)*lftL + frac*lftH;
return output;
}
}
else if(pp>p[18]){
if (f[18][1][4]*f[20][1][4] == 0) {
return -999;
}
else {
frac = log10(pp/p[18])/log10(p[20]/p[18]);
lftL = (1-frac)*log10(f[18][1][4]) + frac*log10(f[20][1][4]);

/* Second, interpolate in theta (use linear) */
frac = (th-theta[4])/(theta[5]-theta[4]);
output = (1-frac)*lftL + frac*lftH;
return output;
}
}
}

/*-----case 9-----*/
else if((pp>p[19])&&(pp<p[24]) && (th>theta[4])&&(th<=theta[5])) {
if (f[19][1][1]*f[24][1][1] == 0) {
return -999;
} else {
frac = log10(pp/p[19])/log10(p[24]/p[19]);
lftH = (1-frac)*log10(f[19][1][1]) + frac*log10(f[24][1][1]);
}
}
if(pp<p[20]) {
if (f[18][1][4]*f[20][1][4] == 0) {
return -999;
}
else {
frac = log10(pp/p[18])/log10(p[20]/p[18]);
lftL = (1-frac)*log10(f[18][1][4]) + frac*log10(f[20][1][4]);

/* Second, interpolate in theta (use linear) */
frac = (th-theta[4])/(theta[5]-theta[4]);
output = (1-frac)*lftL + frac*lftH;
return output;
}
}
}
else if(pp>=p[20] && pp<=p[23]) {
for (w=20;p[w+1]<pp;w++){
if (f[w][1][4]*f[w+1][1][4] == 0) {
return -999;
}
else {
frac = log10(pp/p[w])/log10(p[w+1]/p[w]);
lftL = (1-frac)*log10(f[w][1][4]) + frac*log10(f[w+1][1][4]);

/* Second, interpolate in theta (use linear) */
frac = (th-theta[4])/(theta[5]-theta[4]);
output = (1-frac)*lftL + frac*lftH;
return output;
}
}
}
else if(pp>p[23]){
if (f[24][1][0]*f[24][1][1] == 0) {
return -999;
}
}
}

```

```

    }
    else {
        frac = theta[4]/theta[5];
        lft10 = (1-frac)*log10(f[24][1][0]) + frac*log10(f[24][1][1]);

        frac = log10(pp/p[23])/log10(p[24]/p[23]);
        lftL = (1-frac)*log10(f[23][1][4]) + frac*lft10;

        /* Second, interpolate in theta (use linear) */
        frac = (th-theta[4])/(theta[5]-theta[4]);
        output = (1-frac)*lftL + frac*lftH;
        return output;
    }
}

}

/*-----case 10-----*/
else if (pp>p[18]&&pp<=p[20]&&th<=theta[1]) {
    if (pp>p[18]&&pp<=p[19]) {
if (f[18][1][0]*f[19][1][0]*f[18][1][1]*f[20][1][1] == 0) {
    return -999;
} else {
        frac = log10(p[19]/p[18])/log10(p[20]/p[18]);
        lft11 = (1-frac)*log10(f[18][1][1]) + frac*log10(f[20][1][1]);

        lft00 = log10(f[18][1][0]);
        lft01 = log10(f[18][1][1]);
        lft10 = log10(f[19][1][0]);
        return fourpt(p[18],p[19],pp,theta[0],theta[1],th,lft00,lft01,lft10,lft11);
    }
}
else {
if (f[19][1][0]*f[20][1][0]*f[18][1][1]*f[20][1][1] == 0) {
    return -999;
} else {
        frac = log10(p[19]/p[18])/log10(p[20]/p[18]);
        lft01 = (1-frac)*log10(f[18][1][1]) + frac*log10(f[20][1][1]);

        lft00 = log10(f[19][1][0]);
        lft10 = log10(f[20][1][0]);
        lft11 = log10(f[20][1][1]);
        return fourpt(p[19],p[20],pp,theta[0],theta[1],th,lft00,lft01,lft10,lft11);
    }
}
}

}

/*-----case 11 -----*/
else if(pp>p[24]) return -999;
else printf(" Out of Case on interpolate \n");
}

/* fourpt - four-point interpolation

First we use log-log interpolation of f in the p direction, and then
log-linear interpolation of f in the theta direction.

inputs:
p0 -> lower p
p1 -> higher p
pp -> p where log10(f) is desired
th0 -> lower theta
th1 -> higher theta
th -> theta where log10(f) is desired

```



```

lf00 -> log10(f) at the lower p, lower theta
lf01 -> log10(f) at the lower p, higher theta
lf10 -> log10(f) at the higher p, lower theta
lf11 -> log10(f) at the higher p, higher theta

variables in the subroutine:
lfint0 -> log10(f) interpolated in (log10)p, for lower theta
lfint1 -> log10(f) interpolated in (log10)p, for higher theta
output -> final, interpolated log10(f)
*/

double  fourpt(p0,p1,pp,th0,th1,th,lf00,lf01,lf10,lf11)
double  p0, p1, pp, th0, th1, th, lf00, lf01, lf10, lf11;
{
    double  lfint0, lfint1, lp0, lp1, lpp, frac,  output;
    void    nrerror();

    if (p0 <= 0 || p1 <= 0) {
        printf("p0=%le, p1=%le\n");
        nrerror("fourpt: bad p0 or p1");
    }
    lp0 = log10(p0);
    lp1 = log10(p1);
    lpp = log10(pp);
    frac = (lpp-lp0)/(lp1-lp0);
    lfint0 = (1-frac)*lf00 + frac*lf10;
    lfint1 = (1-frac)*lf01 + frac*lf11;

    frac = (th-th0)/(th1-th0);
    output = (1-frac)*lfint0 + frac*lfint1;

    return output;
}

```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Initial.c

```

/* Define initial distribution of hydrogen atoms */

#include <stdio.h>
#include <math.h>
#define M 938780000.0 /* Mass of a hydrogen atom (eV/c^2) */
#define k 8.617384e-05 /* Boltzmann constant (eV/K) */
#define T 10900 /* Temperature (K) */
#define VAVE -26 /* average velocity of maxwellian initial distribution (km/s) */
#define L 200 /* define position of z[l] for initial Hydrogen flow */

extern double ***f;
extern double *p, *mu0, *mu1;
extern int np,nz,*fine,nmu0,nmu1;

void initial()
{
    int u, w, l, nmu;
    double b, d, g, mu, pi, pave;

    pi=4.0*atan(1.0);
    pave=VAVE*3136.125; /* change velocity to momentum */
    b=(2.0*pi*M*k*T);

    for(w=0;w<=np;w++) {
        if(w==0) nmu=0;
        else if(fine[w]==0) nmu=nmu0;
        else nmu=nmu1;
        for(l=1;l<=nz;l++) {
            for(u=0;u<=nmu;u++) {
                if(fine[w]==0) mu=mu0[u];
                else mu=mu1[u];

                d=(p[w]*p[w])-(2.*mu*p[w]*pave)+(pave*pave);
                g=d/(2.*M*k*T);
                if(l==L) f[w][l][u]=(1./pow(b,1.5))*exp(-g);
                else f[w][l][u]=0.0;
            }
        }
    }

    /* consider lower Hydrogen distribution by if <very small> given to ZERO */
    for(w=0;w<=np;w++) {
        if(w==0) nmu=0;
        else if(fine[w]==0) nmu=nmu0;
        else nmu=nmu1;
        for(l=1;l<=nz;l++) {
            for(u=0;u<=nmu;u++) {
                if(f[w][l][u]<1.0e-100) f[w][l][u]=0.0;
            }
        }
    }
}

```

Printout.c

```

/* Print output for v_\perp vs. v_\parallel ---> Contour plots of distribution .

We have f as a function of p and mu = p_z/p.
Now we want to output f for a rectangular grid of
v_\parallel and v_\perp values, using linear interpolation
in mu and geometric interpolation in p.
(Note that : v_\parallel = v_z = mu*v, and v_\perp = sqrt(1-mu*mu)*v. )

First step: what (pp) and (angle) correspond to our desired vz and vy ?
*/
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

#define VRANGE 500. /* in (km/s) */
#define YVRANGE 400. /* in (km/s) */
#define VSTEP 10. /* in (km/s) */
#define C 299790. /* in (km/s) */
#define M 938780000.0 /* Mass of a hydrogen atom (eV/c^2) */
#define SMALL 0.000000000001

extern double ***f;
extern double *p,*mu0,*mu1,*theta;
extern int np,nz,zstep,*fine,nmu0,nmu1;

void printout(printtime,time)
double printtime, time;
{
FILE *fp_f, *fp_v, *fopen();
static char fn_f[]="mu_z_f.dat", fn_v[]="vplot0000.dat";
double gamma, lff, angle;
double pp, v, vy, vz;
double pi, mu;
double interpo(),*dvector();
int u, w, nmu, l;

pi= 4.0*atan(1.0);
printf("\n-----begin printout.c-----\n");

for(l=20;l<=200;l+=20) {

fn_v[8]++;
if(fn_v[8]==':') {
fn_v[8]='0';
fn_v[7]++;
if(fn_v[7]==':') {
fn_v[7]='0';
fn_v[6]++;
if(fn_v[6]==':') {
fn_v[6]='0';
fn_v[5]++;
}
}
}
}

/* print value of velocity distribution : vplot of vy and vz into file */
fp_v = fopen(fn_v,"w");
printf(" Test printout at time = %.21f (days) at l = %d \n",time,l);

```

```

fprintf(fp_v, " printout at time = %.2lf (days) at l = %d \n",time,l);
fprintf(fp_v, " ");
for(vz=-VRANGE;vz<=VRANGE+0.1*VSTEP;vz+=VSTEP) fprintf(fp_v, "%8.2lf ",vz);
fprintf(fp_v, "\n");

for(vy=-YVRANGE;vy<=YVRANGE+0.1*VSTEP;vy+=VSTEP) {
    fprintf(fp_v, " %8.2lf ",vy);
    for(vz=-VRANGE;vz<=VRANGE+0.1*VSTEP;vz+=VSTEP) {
        v = sqrt((vy*vy)+(vz*vz));
        if(v<SMALL) {
            if(f[0][1][0]>0.0) lff=log10(f[0][1][0]);
            else lff = -999.;
        }
        else {
            gamma = 1.0/sqrt(1.0-(v*v)/(C*C));
            pp = gamma*M*v/C; /* calcalte momentum (eV/C) */
            /* calculate angle [0 to 180 Degree] */
            if (atan2(vy,vz)<0.0) angle = -180.0*atan2(vy,vz)/pi;
            else angle = 180.0*atan2(vy,vz)/pi;

            /* In most cases, will use linear interpolation of log10(f) in terms of log10(p). */

            lff = interpo(pp,l,angle); /* interpolate part from function in boltz.c */
        }
        fprintf(fp_v, " %8.2lf ",lff);
    }
    fprintf(fp_v, "\n");
}
fclose(fp_v);
}

/* ----- print value of z_step, mu and f[w][l][u] into file ----- */
/* fp_f=fopen(fn_f,"a");
fprintf(fp_f, "\n time = %lf \n",time);
w=1;
if(fine[w]==0) nmu=nmu0;
else nmu=nmu1;
for(u=0;u<=nmu;u++) {
    if(fine[w]==0) mu=mu0[u];
    else mu=mu1[u];
    for(l=1;l<=nz;l++) {
        fprintf(fp_f, "\n mu=%8.4lf z=%8.5lf %9.51e ",mu,l*zstep,f[w][l][u]);
    }
}
fclose(fp_f);
*/

printf("\n----- Exit printout.c -----");
}

```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Stream.c

```

/*
Update by used Tanin TVD original concept
September 4, 2000

change routine of New TVD from Tanin : tvd -> tvdinflow()
and adapt for inflow process in boltzmann case

s_inflow.c -- March 21, 2000

Adapting TVD for inflow boundary conditions, so tvd() -> tvdinflow().
Also improving modularity of tvdinflow().

s_cont.c -- February 15, 2000

Adapted for wind (stvd) code. gen_gam() now receives sstep as an
argument and calculates timestep inside. nz and zstep are not
arrays any more. w is an argument, no w loop inside.

s_cont.c -- November 23, 1999

Adapted for hybwind code.

Fixed a bug in tvd(), discovered with varwind code, which caused
segmentation faults when the Courant number was less than -1.
Also clarified some of logic in tvd().

April 7, 1999
Modified to incorporate the latest version of tvd(). ltrue is back.

-- February 2, 1999   @@@ only temporary version @@@
modified tvd() by eliminates "ltrue", by only using
l to control the whole spatial movement;
-- January 25, 1999
-- January 15, 1999
Modified for compatibility with varwind.c. Instead of vel,
use gam, i.e., gamma = v_z * Delta t / Delta z.

*/

#include<stdio.h>
#include<math.h>

#define SMALL 1.0e-6
#define TINY 1.0e-10
#define C 173.1 /* units of AU/day */

#define M 938780000.0 /* Mass of a hydrogen atom (eV/C^2) */
#define k 8.617384e-05 /* Boltzmann constant (eV/K) */
#define T 10900. /* Temperature ( K ) */
#define pi 4*atan(1.0)
#define VAVE -26

extern double ***f, ***gam;
extern double *p, *mu0, *mu1, zstep;
extern int np, nz, nmu0, nmu1, *fine;

double *dvector2(), ***darray();

```

```

/* ***** */
void stream(time,timestep,w)
/* ***** */

double  time, timestep;
int     w;

{
  double  delta_t_z, *ff, *Fj, *ga, *lsp, finu;
  double  b, d, g, mu, pave;
  int     nmu, u;
  long    l;
  void    tvdinflow(), free_dvector();

  ff = dvector2(1,nz+2);
  ga = dvector2(0,nz);
  lsp = dvector2(1,nz);

  delta_t_z = timestep/zstep;
  pave=VAVE*3136.125;

  printf("\n in routine of Streaming  w = %2d ",w);

  if (w==0) nmu=0;
  else if(fine[w]==0) nmu=nmu0;
  else nmu=nmu1;

  for (u=0;u<=nmu;u++) {
    if(fine[w]==0) mu = mu0[u];
    else mu = mu1[u];

    ga[0] = gam[w][0][u];
    for(l=1;l<=nz;l++) {
      ff[l] = f[w][l][u];
      ga[l] = gam[w][l][u];
    }

    /* Setting upper z-boundary condition (finu): Maxwellian distribution. */
    b=2.*pi*M*k*T;
    d=(p[w]*p[w])-(2.0*mu*p[w]*pave)+(pave*pave);
    g=d/(2.0*M*k*T);
    finu=(1./pow(b,1.5))*exp(-g);
    if(finu<1e-100) finu=0.0;

    tvdinflow(ff, finu, ga, lsp, delta_t_z, w, nz, 1);

    for(l=1;l<=nz;l++) f[w][l][u] = ff[l];
  }
  free_dvector(ff,1);
  free_dvector(ga,0);
  free_dvector(lsp,1);
}

/* ***** */
void gen_gam(timestep,beta)
/* ***** */

/* This routine generates the initial velocity at each grid point in mu-z space */

double  timestep, *beta ;

```

```

{
  double  z, mu;
  int     w, u, nmu;
  long    l;
  double  ***darray();

  gam = darray(0,np,0,nz,0,nmu0);

  for (w=0; w<=np; w++) {
    if (w==0)      nmu=0;
    else if(fine[w]==0)  nmu=nmu0;
    else          nmu=nmu1;
    for(l=0; l<=nz; l++){
      z=((double) l)*zstep;
      for(u=0; u<=nmu; u++){
        if(fine[w]==0)  mu = mu0[u];
        else          mu = mu1[u];
        gam[w][l][u] = mu*beta[w]*C*timestep/zstep;
      }
    }
  }
}

void tvdinflow(ff, finu, ga, lsp, delta_t_z, w, znum, lstart)
/* use lstart = 1 on stream function after discussion with Dr.Tanin Nutaro */

double  delta_t_z, *ff, *lsp, *ga, finu;
int     w;
long    znum, lstart;
{
  double  gamma, *fold, *Fj, *rj, *phi, *df_plus, veff, fin;
  long    l, ltrue, lm;
  int     f_ward, fw_old;
  void    free_dvector(), nrerror();

  fold    = dvector2(1,znum);
  Fj      = dvector2(0,znum);
  rj      = dvector2(0,znum);
  phi     = dvector2(0,znum);
  df_plus = dvector2(0,znum); /* df_plus is the forward difference. */

  ff[znum-1]=finu;
  ff[znum]=finu;
  for(l=1;l<=znum;l++) fold[l] = ff[l];

  for(l=0,ltrue=lstart-1; l<=znum; l++,ltrue++) {

    gamma = ga[ltrue];
    f_ward = floor(gamma+SMALL);
    gamma -= (double)f_ward;

/* necessary condition for gamma calculation, because we have added a small number 1.0e-6 */

    if(fabs(gamma) <= SMALL) gamma =0.0;
    if(gamma < 0 || gamma >= 1){
      nrerror("tvd: bad gamma after subtraction");
    }
    lm = l - f_ward;

    if(gamma==0.0 || gamma < TINY ) { Fj[l]=0.0; }
    else { veff = gamma/delta_t_z;

```

```

if(lm == znum || lm == 1) { Fj[l] = veff*fold[lm]; }
else if(lm < znum && lm > 1) {
    df_plus[l] = fold[lm+1] - fold[lm];

    if(fabs(df_plus[l]) < TINY) { Fj[l] = veff*fold[lm]; }
    else {
        rj[l] = (fold[lm]-fold[lm-1])/df_plus[l];

        if(rj[l] <= TINY) { phi[l] = 0.0; }
        else if (rj[l] <= 0.5){ phi[l] = 2*rj[l]; }
        else if (rj[l] <= 1.0){ phi[l] = 1.0; }
        else if (rj[l] < 2.0){ phi[l]= rj[l]; }
        else { phi[l]=2.0; }

        Fj[l] = veff*fold[lm] + 0.5*veff*(1.0 - gamma)*df_plus[l]*phi[l];
    }
}
else { Fj[l]=0.0; }
}
}
/* ----- ending of Flux Fj[l] calculation ----- */

/* Boundary condition: No inflow. Fj = veff*(incoming f). */
fw_old = floor(ga[lstart-1]+SMALL);

/* Now set lsp[l] to be the value of lsp[1] (Fj[l+f_ward]*delta_t_z/ff[l]).
When in doubt set to be gamma.
*/

for(l=1,ltrue=lstart; l<=znum; l++,ltrue++) {
    if (ff[l] > TINY) {
        lsp[l] = Fj[l+f_ward]*delta_t_z/ff[l];
    } else {
        gamma = ga[l];
        f_ward = floor(gamma+SMALL);
        lsp[l] = gamma - f_ward;
    }
}

/* Think backwards: particles come from l-f_ward and l-f_ward-1. */

for (l=1;l<=znum;l++) fold[l] = ff[l];

for (l=1,ltrue=lstart; l<=znum; l++,ltrue++){
    gamma = ga[ltrue];
    f_ward = floor(gamma+SMALL);

    ff[l] = -delta_t_z*(Fj[l]-Fj[l-1]);

    /* Treats general case with different fw_old (g-) and f_ward (g+).
    The only constraint is that f_ward <= fw_old+1.

    BC: assumes that ff from #outside# (l=1 to znum) is zero.
    */

for(lm=l-fw_old>1?l-fw_old:1 ; lm<=(l-f_ward<znum?l-f_ward:znum) ; lm++)
    ff[l] += fold[lm];

if(ff[l] < 0.0){
    printf(" ERROR at ff[%3d] =%.5e\n",l,ff[l]);
}
}
}

```



```
        nrrerror("in tvd :: stream! ff < 0");  
    }  
    fw_old = f_ward;  
}  
  
free_dvector(df_plus,0);  
free_dvector(rj,0);  
free_dvector(Fj,0);  
free_dvector(phi,0);  
free_dvector(fold,1);  
}
```



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Chex.c

```

/*
  Charge Exchange Process.
  ## Nov 24, 2000 : change in Volume P: because add new coarse grid point

  ## Oct 27, 2000 : Update new version in chex2000.c at Term1 & Term 3

  > Modify on Th 2 Dec 99 at term3
  > change for global variable on Jan,27 2000
  by use global variable for Grid
*/

#include <stdio.h>
#include <math.h>
#define C 2997900000.0 /* light velocities (cm/s) */
#define M 938780000.0 /* Mass of a hydrogen atom (eV/C^2) */
#define K 8.617384e-05 /* Boltzmann constant (eV/K) */

double **fp;
extern double ***f;
extern double *dens,*px,*temp;
extern double *p,*mu0,*mu1;
extern int np,nz,zstep,*fine,nmu0,nmu1;

void chex(timestep,beta)
double timestep, *beta;
{
  FILE *fp_n;
  static char fn_n[]="l_plot.dat";
  int u, w, w1, u1, l, nmu, Nmu, Nmu1;
  double **fnew,**volP, *m_mu, mu00, mu11, v, v1;
  double pi, arg, value, term1, term2, term3;
  double **dmatrix2(), *dvector(), Velocity_term();
  void VolumeP();

  printf("\n ----- begin chex.c ----- \n");
  pi = 4.0*atan(1.0);
  fnew = dmatrix2(0,np,0,nmu0);
  fp = dmatrix2(0,np,0,nmu0);
  volP = dmatrix2(0,np,0,nmu0);
  m_mu = dvector(0,nmu0);
  fp_n=fopen(fn_n,"w");

  VolumeP(volP);
  printf("\n in Charge Exchange Processing \n\n");

  for(l=1;l<=nz;l++) {
    // printf("\n loop for chex l=%3d ",l);

    /* ---- define Hydrogen distributon fnew[w][l] = f[w][l][u] ----- */

    fnew[0][0] = f[0][l][0];
    for(w=1;w<=np;w++){
      if(fine[w]==0) nmu=nmu0;
      else nmu=nmu1;
      for(u=0;u<=nmu;u++) fnew[w][u] = f[w][l][u];
    }
  }
}

```

```

/* ---- Let function of proton distribution fp[w=0..np][u=0..nmu] ----- */

arg = -(px[l]*px[l])/(2.0*M*K*temp[l]);
if((arg>-50.0) && (arg<50.0)) {
  fp[0][0] = (dens[l]/pow(2.0*pi*M*K*temp[l],1.5))*exp(arg);
}
else if(arg<=-50.0) fp[0][0] = 0.0;
else nerror("hello boy chex: bad fp[0][0]");

for(w=1;w<=np;w++) {
  if(fine[w]==0) nmu=nmu0;
  else nmu=nmu1;
  for(u=0;u<=nmu;u++) {
    if(fine[w]==0) m_mu[u]=mu0[u];
    else m_mu[u]=mu1[u];

    arg = -((p[w]*p[w])-(2.0*m_mu[u]*p[w]*px[l])+(px[l]*px[l]))/(2.0*M*K*temp[l]);
    if((arg>-50) && (arg<50)) {
      fp[w][u] = (dens[l]/(pow(2.0*pi*M*K*temp[l],1.5)))*exp(arg);
    }
    else if(arg<=-50) fp[w][u] = 0.0;
    else nerror("Error on chex: bad fp[w][u]");
  }
}
}
/* ----- */

/* ----- !!! consider Charge Exchange for [w][u] <----> [w1][u1] !!! ----- */

for(w=0;w<=np;w++) {
  if(w==0) Nmu=0;
  else if(fine[w]==0) Nmu=nmu0;
  else Nmu=nmu1;
  for(u=0;u<=Nmu;u++) {
    if(fine[w]==0) m_mu[u]=mu0[u];
    else m_mu[u]=mu1[u];
    mu00 = m_mu[u]; /* mu for position [w][u] of interest particle */

    /* !! Let [w1][u1] for collision !! */
    if(w==0&&u==0) { w1=1; u1=0; }
    else { w1=w; u1=u+1; }
    for(;w1<=np;w1++) {
      if(w1==0) Nmu1=0;
      else if(fine[w1]==0) Nmu1=nmu0;
      else Nmu1=nmu1;
      for(;u1<=Nmu1;u1++) {
        if(fine[w1]==0) m_mu[u1]=mu0[u1];
        else m_mu[u1]=mu1[u1];
        mu11 = m_mu[u1]; /* mu1 for position [w1][u1] of particle collision */

        /* here only: v is in units of (cm/s) */
        v = beta[w]*C;
        v1 = beta[w1]*C;

        /* velocity term <TERM 1> */
        term1 = Velocity_term(mu00,mu11,v,v1);

        /* transfer term <TERM 2> */
        term2 = (f[w1][l][u1]*fp[w][u])-(f[w][l][u]*fp[w1][u1]);

        /* volume term <TERM 3> */

```

```

        term3 = volP[w1][u1]; /* Call array of volumeP */

/* --- updates f --- */
value = timestep*86400.*term1*term2*term3;
fnew[w][u] += value; /* fnew increase in [w][u] */

        if(!(value > -1000000. && value < 1000000.)) {
            printf("\n value=%le, timestep=%.21f days ",value,timestep);
            printf("\n l=%d: w=%d, u=%d, w1=%d, u1=%d ",l,w,u,w1,u1);
            printf("\n term1=%le , term2=%le , term3=%le",term1,term2,term3);
            printf("\n fnew =%le\n",fnew[w][u]);
            exit(1);
        }

value = timestep*86400.*term1*term2*volP[w][u];
fnew[w1][u1]-=value; /* fnew decrease in [w1][u1] */

        } /* END OF U1 */
        u1=0;
    } /* END OF W1 */
} /* END OF U */
} /* END OF W */

for(w=0;w<=np;w++) {
    if(w==0) nmu=0;
    else if(fine[w]==0) nmu=nmu0;
    else nmu=nmu1;
    for(u=0;u<=nmu;u++) {
        f[w][l][u] = fnew[w][u];
        if(f[w][l][u]<0.0) {
            printf("\n chex: ## f [%2d] [%3d] [%d]<0 --> f = %le",w,l,u,f[w][l][u]);
            exit(1);
        }
    }
}

} /* END OF L */

fclose(fp_n);
free_dmatrix2(fnew,0,np,0);
free_dmatrix2(fp,0,np,0);
free_dmatrix2(volP,0,np,0);
free_dvector(m_mu,0);
printf("\n -----end chex.c----- ");
}

void photoionization(timestep)
double timestep;
{
    int w, l, u, nmu;
    double value;
    printf("\n Photo-ionization Process \n");
    for(w=0;w<=np;w++){
        if(w==0) nmu=0;
        else if(fine[w]==0) nmu=nmu0;
        else nmu = nmu1;
        for(l=1;l<=nz;l++) {
            for(u=0;u<=nmu;u++) {
                value=timestep*86400.*(9.e-08/(1*1))*f[w][l][u];
                f[w][l][u]-=value;
            }
        }
    }
}

```

```

    }
  }
}

/* Arrays of proton data for each z */
void protondata(nz,zstep)
int nz;
double zstep;
{
  /* Read data and computes from file pro97new.dat which is modified from pro_97.dat

     Interpolated because steps in z-direction are NOT constant in proton data file */

  FILE *fp;
  double z_value, z, y, d, vy, p, t, m, frac, px_old, px_new;
  double z_new, d_new, vx_new, t_new, vx;
  int l;

  printf("\n-----begin protondata()-----\n");
  fp=fopen("pro97new.dat","r");
  fscanf(fp,"%le %le %le %le %le %le %le %le",&z_new,&y,&d_new,&vx_new,&vy,&p,&t_new,&m);
  px_new = M*(vx_new*100000.)/C;      /* also changes (km/s) to (cm/s) */

  for(l=1;l<=nz;l++) {
    z_value = l*zstep;

    while(z_value>z_new) {
      z = z_new;
      d = d_new;
      vx = vx_new;    px_old = M*vx*100000.0/C;
      t = t_new;

      fscanf(fp,"%le %le %le %le %le %le %le %le",&z_new,&y,&d_new,&vx_new,&vy,&p,&t_new,&m);
    }
    px_new = M*vx_new*100000.0/C;
    frac = (z_value-z)/(z_new-z);
    dens[l]=(1.0-frac)*d + frac*d_new;
    px[l] = (1.0-frac)*px_old + frac*px_new;
    temp[l]=(1.0-frac)*t + frac*t_new;

  }
  fclose(fp);
  printf("\n\n-----end protondata()-----\n");
}

double doublefact(int n)
{
  int i;
  double value;

  value = 1;
  for(i=1;i<=n;i++) value *= ((2.0*i)-1.0)/(2.0*i);
  return(value);
}

double alphafact(double alpha,int n)
{
  int i;
  double value;

```

```

value=1;
for(i=0;i<=(2*n)-1;i++) value*=(alpha-i)/(i+1);
return(value);
}

/* Velocity Term */
double Velocity_term(mu, mu1, v, v1)
double mu, mu1, v, v1;
{
double logsigma, Ecm, alpha, w2, r, v_rel;
double B, be, deviation, MeanVrel, MeanVrel_old;
int n;

v_rel = (v*v)+(v1*v1)-(2.0*v*v1*mu*mu1);
v_rel = sqrt(v_rel); /* v_rel = |vector(v) - vector(v1)| */

Ecm = 0.25*M*v_rel*v_rel/(C*C);
if(Ecm<0.0) {
nrerror("Velocity_term: bad Ecm (Ecm<0.0)");
exit(1);
}

/*
log10(sigma/cm^2)= B + (beta*log10(v_rel/cm s^-1))
where B and beta are from the paper of Schultz, Ovchinnikov & Passovets 1995
Chapter 11: Elastic & Related Cross Sections for Low-Energy Collisions Hydrogen
----- and Helium ions, Neutrals, and Isotopes
*/

if(Ecm<=10) {
B = -14.3442-(0.1000134*log10(M/(4.*C*C)));
be = -0.1000134*2.;
alpha = (1.+be)/2.;
} else {
B = -14.33676-(0.1358551*log10(M/(4.*C*C)));
be = -0.1358551*2.0;
alpha = (1.+be)/2.0;
}
w2 = (v*v)+(v1*v1)-(2.0*v*v1*mu*mu1);
r = (-2.0*v*v1*sqrt(1.0-(mu*mu))*sqrt(1.0-(mu1*mu1)))/w2;

n=0;
deviation=1.0;
MeanVrel = 0.0;
while(deviation>=0.01) {
MeanVrel += alphafact(alpha,n)*doublefact(n)*pow(r,2.*n);
if(n!=0) deviation = fabs(MeanVrel-MeanVrel_old);
MeanVrel_old = MeanVrel;
n=n+1;
}
MeanVrel = pow(w2,alpha)*MeanVrel;
return(pow(10.0,B)*MeanVrel);
}

void VolumeP(volP)
double **volP;
{

volP[0][0] = 1.615023e+13;
volP[1][0] = 1.010350e+12;

```

```

volP[1][1] = 7.979368e+12;
volP[1][2] = 2.560055e+13;
volP[1][3] = 4.311621e+13;
volP[1][4] = 3.009912e+13;
volP[1][5] = 2.058594e+13;
volP[1][6] = 3.009912e+13;
volP[1][7] = 4.311621e+13;
volP[1][8] = 2.560055e+13;
volP[1][9] = 7.195535e+12;
volP[1][10] = 1.681779e+12;
volP[1][11] = 1.124037e+11;
volP[2][0] = 1.882532e+12;
volP[2][1] = 1.486754e+13;
volP[2][2] = 4.770017e+13;
volP[2][3] = 8.033618e+13;
volP[2][4] = 5.608212e+13;
volP[2][5] = 3.835670e+13;
volP[2][6] = 5.608212e+13;
volP[2][7] = 8.033618e+13;
volP[2][8] = 4.770017e+13;
volP[2][9] = 1.340707e+13;
volP[2][10] = 3.133571e+12;
volP[2][11] = 2.094359e+11;
volP[3][0] = 3.333290e+12;
volP[3][1] = 2.632509e+13;
volP[3][2] = 8.445993e+13;
volP[3][3] = 1.422466e+14;
volP[3][4] = 9.930136e+13;
volP[3][5] = 6.791599e+13;
volP[3][6] = 9.930136e+13;
volP[3][7] = 1.422466e+14;
volP[3][8] = 8.445993e+13;
volP[3][9] = 2.373912e+13;
volP[3][10] = 5.548433e+12;
volP[3][11] = 3.708360e+11;
volP[4][0] = 5.198551e+12;
volP[4][1] = 4.105623e+13;
volP[4][2] = 1.317225e+14;
volP[4][3] = 2.218458e+14;
volP[4][4] = 1.548690e+14;
volP[4][5] = 1.059208e+14;
volP[4][6] = 1.548690e+14;
volP[4][7] = 2.218458e+14;
volP[4][8] = 1.317225e+14;
volP[4][9] = 3.702318e+13;
volP[4][10] = 8.653255e+12;
volP[4][11] = 5.783504e+11;
volP[5][0] = 1.220191e+13;
volP[5][1] = 9.636621e+13;
volP[5][2] = 3.091759e+14;
volP[5][3] = 5.207111e+14;
volP[5][4] = 3.635047e+14;
volP[5][5] = 2.486148e+14;
volP[5][6] = 3.635047e+14;
volP[5][7] = 5.207111e+14;
volP[5][8] = 3.091759e+14;
volP[5][9] = 8.689992e+13;
volP[5][10] = 2.031071e+13;
volP[5][11] = 1.357490e+12;
volP[6][0] = 4.538801e+13;
volP[6][1] = 3.584578e+14;
volP[6][2] = 1.150055e+15;

```



ภาชนะวิทยบริการ
 ภาครณมหาวิทยาลัย

```
volP[6][3] = 1.936913e+15;  
volP[6][4] = 1.352145e+15;  
volP[6][5] = 9.247835e+14;  
volP[6][6] = 1.352145e+15;  
volP[6][7] = 1.936913e+15;  
volP[6][8] = 1.150055e+15;  
volP[6][9] = 3.232456e+14;  
volP[6][10] = 7.555068e+13;  
volP[6][11] = 5.049518e+12;  
volP[7][0] = 2.138832e+14;  
volP[7][1] = 1.689171e+15;  
volP[7][2] = 5.419439e+15;  
volP[7][3] = 9.127369e+15;  
volP[7][4] = 6.371752e+15;  
volP[7][5] = 4.357884e+15;  
volP[7][6] = 6.371752e+15;  
volP[7][7] = 9.127369e+15;  
volP[7][8] = 5.419439e+15;  
volP[7][9] = 1.523239e+15;  
volP[7][10] = 3.560196e+14;  
volP[7][11] = 2.379499e+13;  
volP[8][0] = 6.720465e+14;  
volP[8][1] = 5.307575e+15;  
volP[8][2] = 1.702852e+16;  
volP[8][3] = 2.867928e+16;  
volP[8][4] = 2.002080e+16;  
volP[8][5] = 1.369299e+16;  
volP[8][6] = 2.002080e+16;  
volP[8][7] = 2.867928e+16;  
volP[8][8] = 1.702852e+16;  
volP[8][9] = 4.786199e+15;  
volP[8][10] = 1.118656e+15;  
volP[8][11] = 7.476669e+13;  
volP[9][0] = 1.308722e+15;  
volP[9][1] = 1.033580e+16;  
volP[9][2] = 3.316081e+16;  
volP[9][3] = 5.584912e+16;  
volP[9][4] = 3.898788e+16;  
volP[9][5] = 2.666530e+16;  
volP[9][6] = 3.898788e+16;  
volP[9][7] = 5.584912e+16;  
volP[9][8] = 3.316081e+16;  
volP[9][9] = 9.320493e+15;  
volP[9][10] = 2.178435e+15;  
volP[9][11] = 1.455983e+14;  
volP[10][0] = 2.157623e+15;  
volP[10][1] = 1.704011e+16;  
volP[10][2] = 5.467052e+16;  
volP[10][3] = 9.207558e+16;  
volP[10][4] = 6.427731e+16;  
volP[10][5] = 4.396170e+16;  
volP[10][6] = 6.427731e+16;  
volP[10][7] = 9.207558e+16;  
volP[10][8] = 5.467052e+16;  
volP[10][9] = 1.536622e+16;  
volP[10][10] = 3.591475e+15;  
volP[10][11] = 2.400404e+14;  
volP[11][0] = 3.218749e+15;  
volP[11][1] = 2.542049e+16;  
volP[11][2] = 8.155766e+16;  
volP[11][3] = 1.373586e+17;  
volP[11][4] = 9.588910e+16;
```



บัณฑิตวิทยาลัย
กรมมหาวิทยาลัย


```
volP[11][5] = 6.558221e+16;  
volP[11][6] = 9.588910e+16;  
volP[11][7] = 1.373586e+17;  
volP[11][8] = 8.155766e+16;  
volP[11][9] = 2.292338e+16;  
volP[11][10] = 5.357774e+15;  
volP[11][11] = 3.580931e+14;  
volP[12][0] = 5.153715e+15;  
volP[12][1] = 4.070214e+16;  
volP[12][2] = 1.305864e+17;  
volP[12][3] = 2.199324e+17;  
volP[12][4] = 1.535333e+17;  
volP[12][5] = 1.050073e+17;  
volP[12][6] = 1.535333e+17;  
volP[12][7] = 2.199324e+17;  
volP[12][8] = 1.305864e+17;  
volP[12][9] = 3.670387e+16;  
volP[12][10] = 8.578624e+15;  
volP[12][11] = 5.733624e+14;  
volP[13][0] = 4.480356e+15;  
volP[13][1] = 9.423456e+16;  
volP[13][2] = 3.023368e+17;  
volP[13][3] = 5.091928e+17;  
volP[13][4] = 3.554639e+17;  
volP[13][5] = 2.431153e+17;  
volP[13][6] = 3.554639e+17;  
volP[13][7] = 5.091928e+17;  
volP[13][8] = 3.023368e+17;  
volP[13][9] = 8.497767e+16;  
volP[13][10] = 1.986143e+16;  
volP[13][11] = 1.327462e+15;  
volP[14][0] = 1.151466e+13;  
volP[14][1] = 9.211198e+13;  
volP[14][2] = 3.108089e+14;  
volP[14][3] = 7.363067e+14;  
volP[14][4] = 1.367940e+15;  
volP[15][0] = 1.706613e+13;  
volP[15][1] = 1.365212e+14;  
volP[15][2] = 4.606568e+14;  
volP[15][3] = 1.091297e+15;  
volP[15][4] = 2.027454e+15;  
volP[16][0] = 1.913236e+13;  
volP[16][1] = 1.530502e+14;  
volP[16][2] = 5.164296e+14;  
volP[16][3] = 1.223423e+15;  
volP[16][4] = 2.272923e+15;  
volP[17][0] = 1.852860e+13;  
volP[17][1] = 1.482203e+14;  
volP[17][2] = 5.001324e+14;  
volP[17][3] = 1.184815e+15;  
volP[17][4] = 2.201195e+15;  
volP[18][0] = 1.136907e+13;  
volP[18][1] = 1.144128e+14;  
volP[18][2] = 3.860576e+14;  
volP[18][3] = 9.145710e+14;  
volP[18][4] = 1.699126e+15;  
volP[19][0] = 8.910990e+12;  
volP[19][1] = 2.024653e+17;  
volP[19][2] = 6.495783e+17;  
volP[19][3] = 1.094014e+18;  
volP[19][4] = 7.637232e+17;  
volP[19][5] = 5.223394e+17;
```



บัณฑิตวิทยาลัย
กรมมหาวิทยาลัย

```

volP[19][6] = 7.637232e+17;
volP[19][7] = 1.094014e+18;
volP[19][8] = 6.495783e+17;
volP[19][9] = 1.825767e+17;
volP[19][10]= 4.267279e+16;
volP[19][11]= 2.852086e+15;
volP[20][0] = 1.240541e+13;
volP[20][1] = 1.838305e+13;
volP[20][2] = 1.838305e+13;
volP[20][3] = 1.838305e+13;
volP[20][4] = 1.838305e+13;
volP[21][0] = 1.301790e+13;
volP[21][1] = 1.041373e+14;
volP[21][2] = 3.513853e+14;
volP[21][3] = 8.324322e+14;
volP[21][4] = 1.546525e+15;
volP[22][0] = 2.070756e+13;
volP[22][1] = 1.656510e+14;
volP[22][2] = 5.589481e+14;
volP[22][3] = 1.324149e+15;
volP[22][4] = 2.460056e+15;
volP[23][0] = 2.989155e+13;
volP[23][1] = 2.391188e+14;
volP[23][2] = 8.068466e+14;
volP[23][3] = 1.911421e+15;
volP[23][4] = 3.551113e+15;
volP[24][0] = 3.815577e+16;
volP[24][1] = 3.284499e+17;
volP[24][2] = 1.053780e+18;
volP[24][3] = 1.774766e+18;
volP[24][4] = 1.238952e+18;
volP[24][5] = 8.473664e+17;
volP[24][6] = 1.238952e+18;
volP[24][7] = 1.774766e+18;
volP[24][8] = 1.053780e+18;
volP[24][9] = 2.961854e+17;
volP[24][10]= 6.922604e+16;
volP[24][11]= 4.626804e+15;

```

```

}

```



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Curriculum Vitae

Mr. Chanruangrith Channok was born on July 17, 1974 in Nakornratchasima. He received his B.Sc. degree in physics from Khon Kean University on 1996. He has been a faculty member at the Department of Physics, Faculty of Science, Ubonratchathani University, since 1996.



สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย