การหาคำตอบของสมการโบลต์ซมันน์ในหนึ่งมิติของอวกาศสำหรับอะตอมไฮโดรเจนในระบบสุริยะ

นางสาวปณิตา บุญมา

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาการคณนา ภาควิชาคณิตศาสตร์
คณะวิทยาศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2543

ISBN 974-346-642-8

SOLUTION OF THE BOLTZMANN EQUATION IN ONE SPATIAL DIMENSION FOR HYDROGEN ATOMS IN THE HELIOSPHERE

Miss Panita Boonma

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Computational Science

Department of Mathematics
Faculty of Science
Chulalongkorn University

Academic Year 2000
ISBN 974-346-642-8

Thesis Title	Solution of the Boltzmann Equation in One Spatial Dimension for Hydrogen Atoms in the Heliosphere
By	Miss Panita Boonma
Department	Mathematics
Thesis Advisor	Associate Professor David Ruffolo, Ph. D.

Accept by the Faculty of Science, Chulalongkorn University in Partial Fulfillment of the Requirements for the Master's Degree

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

THESIS COMMITTEE

.....Chairman
(Assistant Professor Pornpote Piumsomboon, Ph.D.)

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

.....Member
(Vimolrat Ngarmaramvarangkool, Ph.D.)

ปณิตา บุญมา : การหาคำตอบของสมการโบลต์ซมันน์ในหนึ่งมิติของอวกาศสำหรับอะตอมไฮโดรเจนในระบบสุริยะ (SOLUTION OF THE BOLTZMANN EQUATION IN ONE SPATIAL DIMENSION FOR HYDROGEN ATOMS IN THE HELIOSPHERE)
อ. ที่ปรึกษา : รศ.ดร. เดวิด รูฟโฟโล, 106 หน้า. ISBN 974-346-642-8.

ดวงอาทิตย์เป็นต้นกำเนิดของกระแสพลาสมาซึ่งเรียกว่าลมสุริยะและโลกโคจรอยู่ท่ามกลางกระแสลมสุริยะที่เคลื่อนที่ด้วยอัตราเร็วประมาณ 400 km/s นอกจากนี้ทั้งระบบสุริยะเคลื่อนที่รอบใจกลางของกาแลคซีทางช้างเผือกหรือจากมุมมองของระบบสุริยะมีกระแสของอะตอมและไอออนจากตัวกลางระหว่างดาว ซึ่งมีอัตราเร็วประมาณ 26 km/s เมื่อกระแสที่มาจากต้นกำเนิดสองชนิดนี้เดินทางมาชนกัน ทำให้อะตอมที่เป็นกลางจากตัวกลางระหว่างดาวเกิดการชนและการแลกเปลี่ยนประจุกับโปรตอนที่มีอยู่ในลมสุริยะมีผลสำคัญต่อการคาด-การณ์โครงสร้างของบริเวณขอบนอกของระบบสุริยะซึ่งยานอวกาศของมนุษย์ยังสำรวจไม่ถึง (ในขณะนี้การคาด-

ภาควิชา	ลายมือชื่อนิสิต
สาขาวิชา	ลายมือชื่ออาจารย์ที่ปรึกษา
ปีการศึกษา	ลายมือชื่ออาจารย์ที่ปรึกษาร่วม

4072297023 : MAJOR MATHEMATICS
KEY WORD : ASTROPHYSICS / HELIOSPHERE / BOLTZMANN EQUATION / CHARGE EXCHANGE / SOLAR WIND / INSTELLAR MEDIUM
PANITA BOONMA : SOLUTION OF THE BOLTZMANN EQUATION IN ONE SPATIAL DIMENSION FOR HYDROGEN ATOMS IN THE HELIOSPHERE.
THESIS ADVISOR : ASSOC. PROF. DAVID RUFFOLO, Ph.D., 106 pp. ISBN 974-346-642-8.

The Sun is the origin of a flow of plasma (ionized gas) called the solar wind, and the Earth orbits the Sun while its magnetosphere is pelted by the solar wind at a speed of about 400 km/s. In addition, the entire solar system orbits the center of our Milky Way Galaxy, and from the point of view of the solar system, there is a flow of atoms and ions from the interstellar medium impacting with a speed of about 26 km/s. Where the two flows collide, neutral atoms from the interstellar medium can undergo charge exchange with protons from the solar wind. This has an important effect on the predicted structure of the region near the outer edge of the solar system, which has not yet been explored by spacecraft. (The solar system is estimated to extend 80-160 AU from the Sun.) In this work we therefore study the evolution of the phase space distribution function of hydrogen atoms due to collisions between atoms and ions by solving the Boltzmann equation.

ภาควิชา ..... ลายมือชื่อนิสิต
สาขาวิชา ..... ลายมือชื่ออาจารย์ที่ปรึกษา
ปีการศึกษา ..... ลายมือชื่ออาจารย์ที่ปรึกษาร่วม

Acknowledgements

I would like to express my deeply felt gratitude to numerous people who have, directly and indirectly, contributed to this work. I am thankful to all of them for their encouragement and support, especially Associate Professor Dr. David Ruffolo, my direct advisor, Paisan Tooprakai and Chanruangrith Channok who helpful during my thesis and my programming work.

I would like to thank the Department of Physics, Faculty of Science of Chulalongkorn University for support computer and a NSTDA/GREC graduate student fellowship. DR acknowledges the kind support of a Basic Research Grant from the Thailand Research Fund.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

The solar wind emits particles and magnetic flux from the Sun into the outer heliosphere at all times. The result of this phenomenon is a low-density hole that is called the heliosphere. At the termination shock the solar wind slows down suddenly and then it contacts the flow from the local interstellar medium (LISM) at the heliopause. However, the interstellar medium not only has charged particles, but also has neutral particles that are not affected by the magnetic field.

Past studies of the effects of charge exchange between hydrogen and protons showed that the result of charge exchange has an impact on the density of particles in the outer heliosphere. The density of particles is a function of the position and the direction of the momentum, and is therefore properly described by a phase space distribution function.

In this thesis we study this effect in detail by considering general elastic collisions changing the particle direction. The resulting interaction between protons and hydrogen shows the effects of charge exchange and other elastic collisions. These results should be useful for understanding some characteristics of the heliosphere.

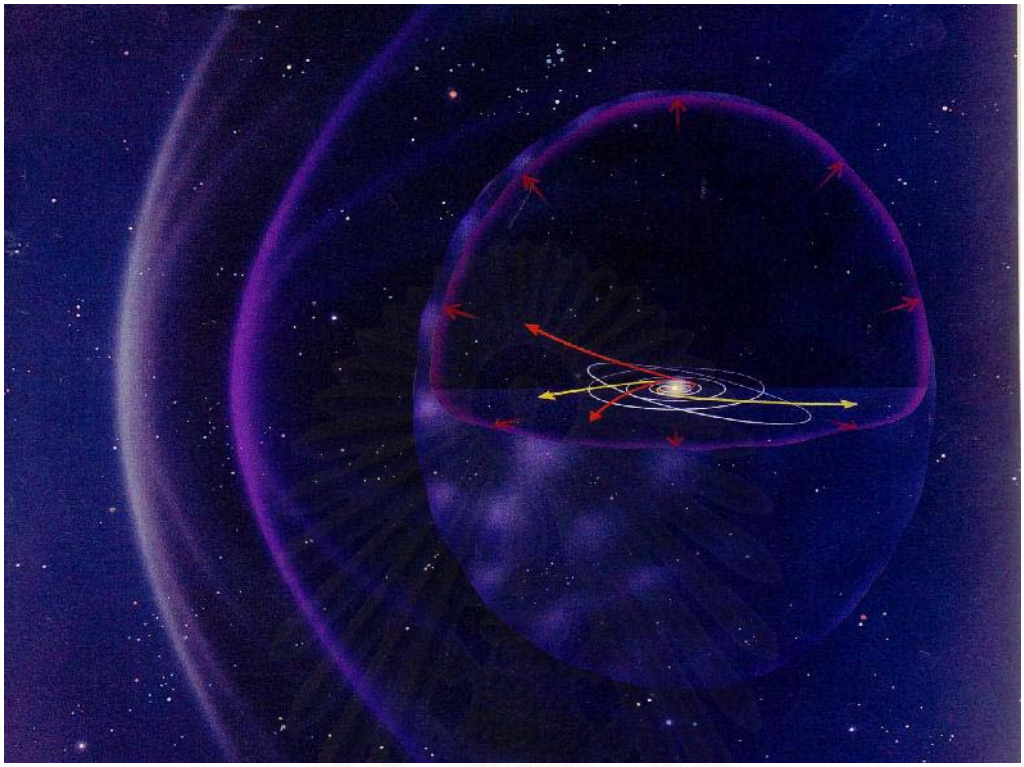

Figure 1.1: Model of the heliosphere (Jokipii and McDonald, 1995).

Chapter 2

Structures of the Heliosphere, the Local Interstellar Medium and the Solar Wind

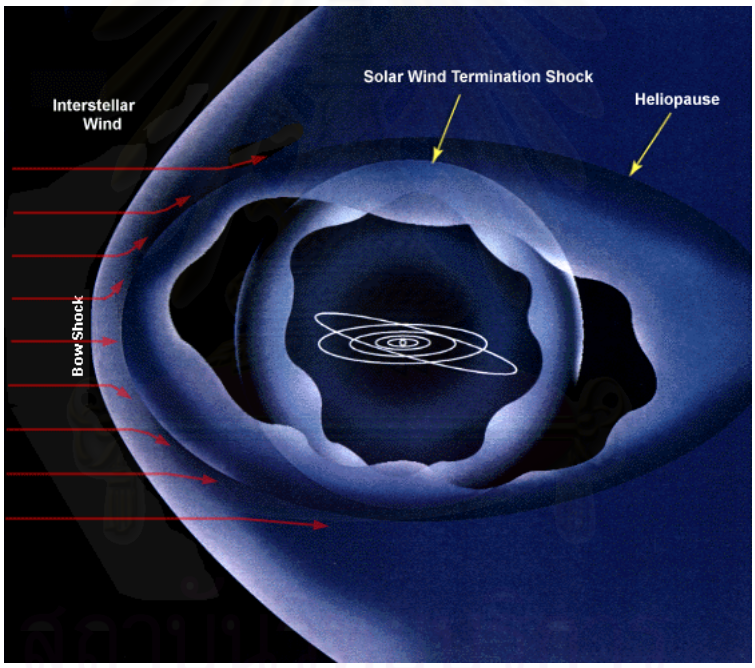

Figure 2.1: Structures of the heliosphere (http://helios.gsfc.nasa.gov/heliosphere.html).

The Sun is the source of the solar wind, which consists of a high-speed outflow of plasma, i.e., an ionized gas. Typically the solar wind has a velocity of about 400 km/s. Both the Sun and the solar wind move with respect to a

denser interstellar medium. The region where the solar wind hits the interstellar medium, including the termination shock, the heliosphere, and the bow shock, is illustrated in Figure 2.1 and the details of this structure are shown in Figure 2.2. The result of the solar wind is a hole (of less dense plasma) that is called the heliosphere. At the termination shock the solar wind suddenly slows down and then encounters the flow from the interstellar medium at the heliopause. However, the interstellar medium not only has charged particles but also has neutral particles which are not affected by the magnetic field. The interaction of protons from the solar wind with the interstellar neutral hydrogen results in charge exchange and elastic collisions. We will further explain these phenomena in the sections below.

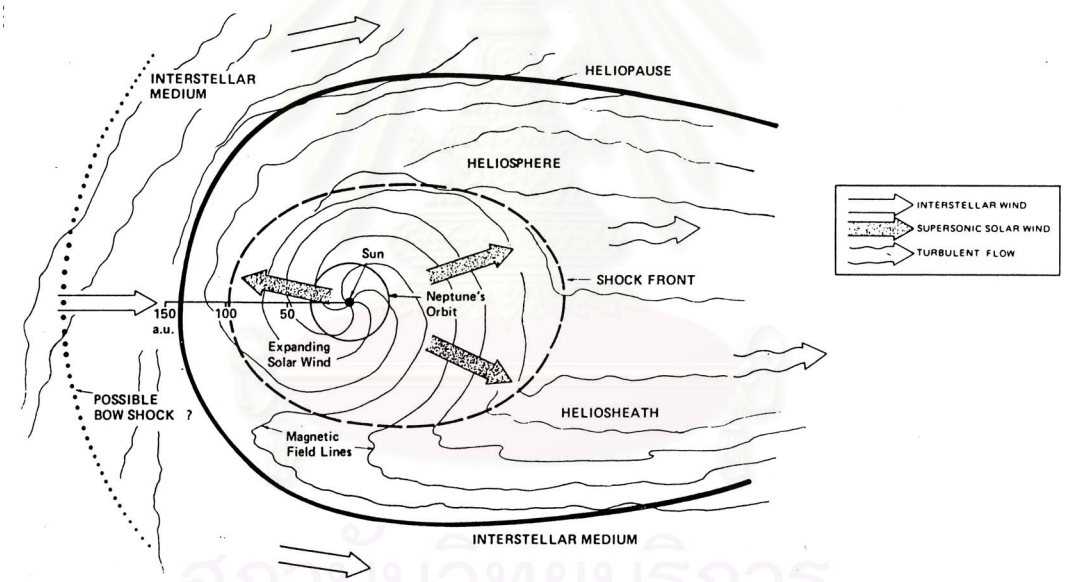

Figure 2.2: Schematic diagram of Figure 2.1. The plane of the figure coincides with the plane of the Sun's equator, which is approximately the general plane of planetary orbits (Foukal, 1990).

2.1 Structures of the Heliosphere

The first suggestions about the nature of the heliosphere were introduced by Davis (1955). A basic element was the so-called "solar corpuscular radiation" would force magnetic flux in the local interstellar medium outward, thereby partially excluding cosmic rays. A simplest expression of this structure is that the solar wind blows a spherical bubble, the "heliosphere," that continually expands over the lifetime of the solar system.

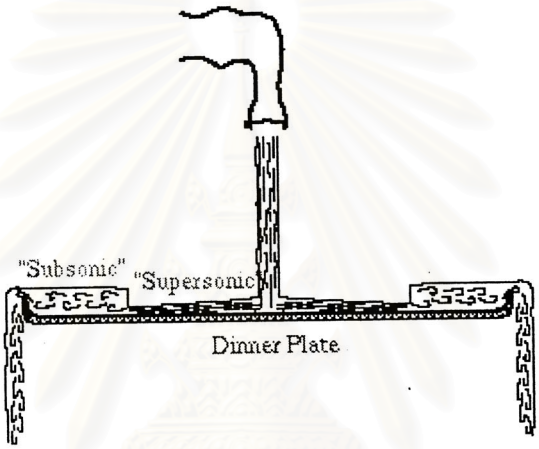

Figure 2.3: Illustration of a shock (http://web.mit.edu/afs/athena/org/s/space/www/helio.review/axford.suess.html).

The termination shock is analogous to a shock in the flow of fluid over a surface under the influence of gravity (Figure 2.1). If water from a tap impacts onto a plate, the water flows radially away from the point of collision at a "supersonic" speed. The term supersonic refers to a flow faster than the speed of sound waves in that fluid. If the rate of the flow is not too large, a shock wave is produced at a position such that the water outside the shock is just deep enough to flow over the raised edge of the plate. It is easy to show that the flow is "subsonic" in the outer region. This is a general feature of shocks: the flow

upstream of the shock is supersonic and that downstream is subsonic. In fact, the only way in which a supersonic flow can slow down to subsonic speeds is by a shock discontinuity. In this analogy, the height, h, of the water indicates the pressure, ρgh, which can be compared with the pressure or the density of the solar wind.

The motion of the Sun and the heliosphere relative to the very local interstellar medium (VLISM) is discussed in the next section. It is natural to assume that the relative velocity might be parallel to the motion of the Sun with respect to the nearer stars, namely from the solar apex with a speed of about 20 km/s. A simplest model of the resulting stagnation point flow is found by assuming incompressibility and neglecting magnetic fields.

The interstellar flow turns at the stagnation point to go around the heliosphere and the subsonic solar wind flow is turned in the region of the "heliosheath," between the termination shock and the heliopause, to flow down the "heliotail." The solar wind inside the termination shock is flowing supersonically and radially (Axford and Suess, http://web.mit.edu/afs/athena/org/s/ space /www/helio.review/axford.suess.html).

2.2 The Interstellar Medium

In this part of the galaxy the composition of the interstellar gas is like that of the gas in our solar system. A cloud of gas can collapse under the influence of its own gravity and create a new star. In a star, the fusion process converts some of the original hydrogen and helium nuclei into nuclei like carbon-12 and oxygen-16. Similarly, carbon, nitrogen, and oxygen nuclei that were previously in the stellar fuel are converted into heavier, neutron-rich nuclei, like neon-22 and magnesium-25 (Christian *et al.*, 1997).

When the quiescent burning has exhausted the nuclear fuel in the core of the star, the star explodes as a supernova. The shock wave generated by the explosion synthesizes addition heavy nuclei and ejects most of the products of nucleosynthesis back into the interstellar gas (Christian *et al.*, 1997).

Some of the nuclei in the gas are accelerated to cosmic ray speeds, maybe by the shock wave from a supernova. The acceleration of cosmic rays could happen directly as the supernova is ejecting matter into interstellar space (Christian *el al.*, 1997).

While the interstellar plasma is kept outside the heliosphere by an interplanetary magnetic field, the interstellar neutral gas flows into the solar system at a speed of 20 km/s. Near the Sun such atoms can lose one electron by photo-ionization or by charge exchange. Photo-ionization occurs when an electron is knocked off by a solar UV photon and charge exchange involves giving up an electron to an ionized solar wind atom. The resulting ions are charged, so the interplanetary magnetic field picks them up and carries them outward to the solar wind termination shock. Spacecraft measurements of these are so-called pickup ions, can determine in the composition, flow and temperature of the neighboring interstellar gas.

Pickup ions can encounter the termination shock, gaining energy in the process of shock drift acceleration which continues until these ions escape from the shock and perhaps diffuse toward the inner heliosphere. These energetic particles are known as anomalous cosmic rays(ACRs).

The anomalus composition of N and O at ~10 MeV per nucleon arises from neutral interstellar atoms passing into the solar cavity by the motion of the Sun through the interstellar medium, which are ionized and accelerated by the solar wind (Fisk *et al.*, 1974). In ACRs there are more helium ions than protons, and much more oxygen than carbon, while in galactic cosmic rays (GCRs) and

solar energetic particles (SEPs) there are many more protons than helium, and roughly equal amounts of oxygen and carbon (Mewaldt, 1994).

2.3 The Solar Wind

The solar wind is a plasma of charged particles such as protons, electrons and heavier ions which flow out of the Sun in all directions at the high speed of about 400 km/s. It is responsible for the anti-sunward tails of comets and the shapes of the magnetospheres around the planets. The composition of the solar wind is much the same as the composition of the solar corona, modified by solar wind processes. The exact mechanisms of formation of the solar wind are not known. Accurately measuring its composition may help in separating the effects of these processes from the basic composition of the corona.

In spite of its low density, the solar wind is strong enough to interact with the planets and their magnetic fields to shape the planets' magnetospheres, because the ions in the solar plasma are charged and are swept around planetary magnetospheres. The shape of the Earth's magnetosphere is the result of being blasted by the solar wind (see illustration of the Earth's magnetosphere in Figure 2.4). The solar wind compresses its sunward side to a distance of only 6 to 10 times the radius of the Earth. A shock wave is created sunward of Earth. This shock wave is called the bow shock. Most of the particles of solar wind are heated and slowed at the bow shock and indirect around the Earth.

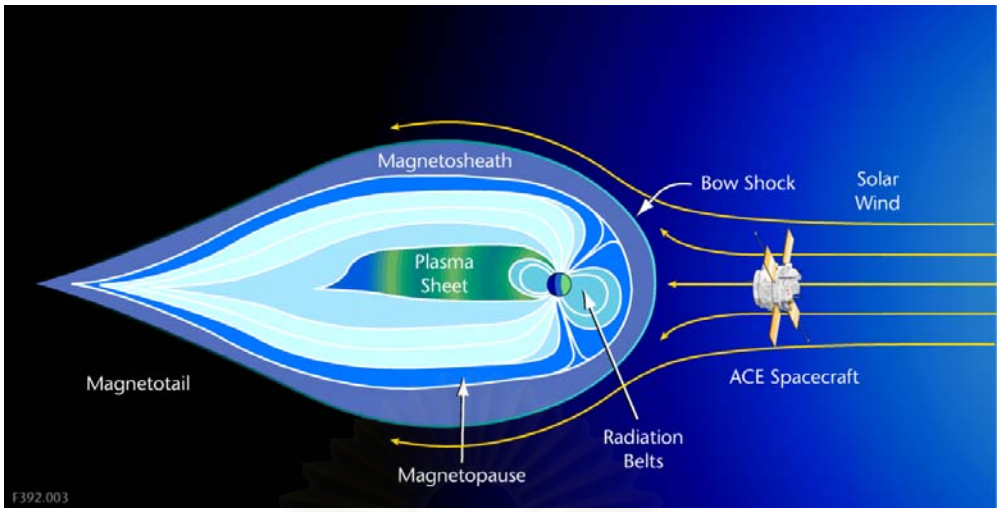

Figure 2.4: The Earth's magnetosphere (http://helios/gsfc/nasa/gov/ace/gallery.html)

The solar wind moves out the night-side magnetosphere to possibly 1,000 times the Earth's radius; no one knows the exact length. This extension of the the magnetosphere is known as the magnetotail. Most of the planets in our solar system have magnetospheres of similar, solar wind influenced shapes (Christian *et al.*, 1997).

Chapter 3

Theoretical Background

3.1 Boltzmann Equation

Consider a particle moving with respect to collision targets with a relative velocity V_{rel} in a small time interval dt. Then the frequency of collisions can be derived in terms of the number of targets within a cylinder of cross sectional area σ and length $V_{rel}dt$ (Figure 3.1). In other words, we have

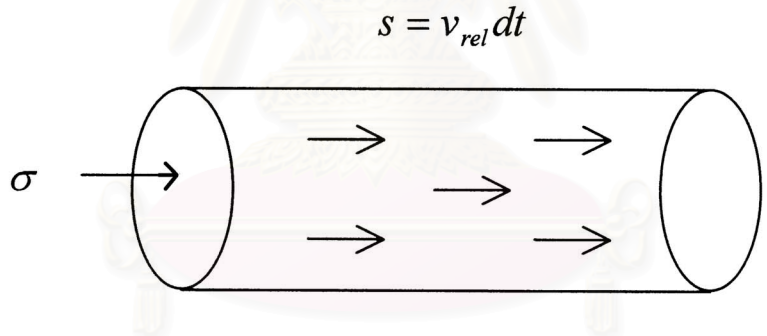

Figure 3.1: The frequency of collisions.

where
$$n = \frac{\text{number of particles}}{\text{volume}}$$
$$\sigma = \text{cross section}$$
$$V_{rel} = \text{relative velocity.}$$

Then

$$\text{number of particle collisions in time } dt = n \times \text{volume}$$

$$
\begin{aligned}
&= n \times V_{rel} dt \times \sigma \\
&= n \times \sigma \times V_{rel} dt \\
\text{rate of collisions} &= n\sigma V_{rel}.
\end{aligned}
$$

Consider particles at time $t + dt$ and with a position and velocity in the range $d\vec{r}\, d\vec{v}$ near $\vec{r}$ and $\vec{v}$. If there are no collisions, particles move to a position around $\vec{r}\,' = \vec{r} + \dot{\vec{r}}dt$ and velocity about $\vec{v}\,' = \vec{v} + \dot{\vec{v}}dt$. If there are collisions,

1. Particles in the considered range can be scattered into the range of interest.

2. Particles in the considered range can be scattered out of it.

Let $D_c f d\vec{r} d\vec{v}$ be the net increase per unit time in the number of particles in the range $d\vec{r} d\vec{v}$ as a result of collisions, so that we can write

$$f(\vec{r} + \dot{\vec{r}}dt, \vec{v} + \dot{\vec{v}}dt, t + dt) d\vec{r} d\vec{v} = f(\vec{r}, \vec{v}, t) + D_c f d\vec{r} d\vec{v} dt, \tag{3.1}$$

where $D_c f$ is the rate of change of f caused by collisions, which can be written as

$$Df = D_c f. \tag{3.2}$$

We call Equation (3.2) the "Boltzmann Equation" (Reif, 1985).

We use knowledge of particle collisions to derive $D_c f$ in the Boltzmann Equation (3.2). By definition (Reif, 1985),

$$Df = \frac{f(\vec{r} + \dot{\vec{r}}dt, \vec{v} + \dot{\vec{v}}dt, t + dt) - f(\vec{r}, \vec{v}, t)}{dt}. \tag{3.3}$$

By a Taylor series we get

$$Df = \frac{\partial f}{\partial t} + \vec{v} \cdot \frac{\partial f}{\partial \vec{r}} + \frac{\vec{F}}{m} \cdot \frac{\partial f}{\partial \vec{v}} \qquad \text{where } \dot{\vec{v}} = \frac{\vec{F}}{m} \tag{3.4}$$

or

$$\frac{\partial f}{\partial t} = -\vec{v} \cdot \frac{\partial f}{\partial \vec{r}} - \frac{\vec{F}}{m} \cdot \frac{\partial f}{\partial \vec{v}} + D_c f. \tag{3.5}$$

Here we assume that there are no effects from external forces ($\vec{F} = 0$), so we have

$$\frac{\partial f}{\partial \vec{t}} = -\vec{v} \cdot \frac{\partial f}{\partial \vec{r}} + D_c f. \tag{3.6}$$

Now consider

$$D_c f = -D_c^{(-)} f + D_c^{(+)} f \tag{3.7}$$

where $-D_c^{(-)} f(\vec{r}, \vec{v}, t) d\vec{r} d\vec{v} dt$ is the decrease in time dt of the density of particles due to scattering out of the range of interest and $D_c^{(+)} f(r, v, t) d\vec{r} d\vec{v} dt$ is the increase in time dt of the number of particles scattered into this velocity range.

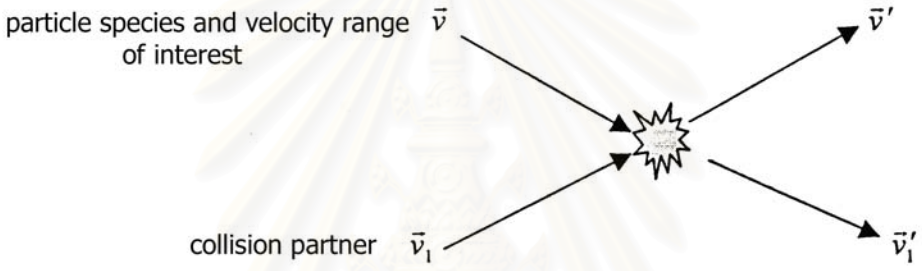

Figure 3.2: An elastic collision of two particles.

To calculate $D_c^{(-)} f$, we consider in the volume element $d\vec{r}$ particles with velocity near $\vec{v}$ (we called them **A** particles) which are scattered out of this velocity range by collisions with other particles (we call them $\mathbf{A}_1$ particles) which are in the same volume element $d\vec{r}$ and which have some velocity $\vec{v}_1$. The scattering cross section of such a collision, where an **A** particle changes its velocity from $\vec{v}$ to $\vec{v}'$ while an $\mathbf{A}_1$ particle changes its velocity from $\vec{v}_1$ and $\vec{v}'_1$, can be written $\sigma'(\vec{v}, \vec{v}_1 \to \vec{v}', \vec{v}'_1) d^3\vec{v}' d^3\vec{v}'_1$. We consider the total decrease $D_c^{(-)} f(\vec{r}, \vec{v}, t) d\vec{r} d\vec{v} dt$ in time dt of the number of particles in $d\vec{r}$ with velocity between $\vec{v}$ and $\vec{v} + d\vec{v}$. We first multiply the cross section $\sigma' d\vec{v}' d\vec{v}'_1$ by the relative flux $|\vec{v} - \vec{v}_1| f(\vec{r}, \vec{v}_1, t) d\vec{v}$

and then multiply this by the number of $\mathbf{A}_1$ particles $f(\vec{r},\vec{v},t)d\vec{r}d\vec{v}$ and the time interval dt.

Next, we sum (integrate) over all possible initial velocities $\vec{v}_1$ of $\mathbf{A}_1$ particles with which $\mathbf{A}$ can collide and over all possible final velocities $\vec{v}\,'$ and $\vec{v}\,'_1$. Thus one obtains:

$D_c^{(-)} f(\vec{r},\vec{v},t)d\vec{r}d\vec{v}dt =$

$$\int_{\vec{v}\,'_1}\int_{\vec{v}\,'}\int_{\vec{v}_1}[|\vec{v}-\vec{v}_1|f(\vec{r},\vec{v}_1,t)d\vec{v}][f(\vec{r},\vec{v},t)d\vec{r}d\vec{v}_1][\sigma'(\vec{v},\vec{r}_1\to\vec{v}\,',\vec{v}\,'_1)d\vec{v}\,'d\vec{v}\,'_1]dt \quad (3.8)$$

and

$D_c^{(+)} f(\vec{r},\vec{v},t)d\vec{r}d\vec{v}dt =$

$$\int_{\vec{v}\,'_1}\int_{\vec{v}\,'}\int_{\vec{v}_1}[|\vec{v}\,'-\vec{v}\,'_1|f(\vec{r},\vec{v}\,',t)d\vec{v}\,'][f(\vec{r},\vec{v}\,'_1,t)d\vec{r}d\vec{v}\,'_1][\sigma'(\vec{v}\,',\vec{v}\,'_1\to\vec{v},\vec{v}_1)d\vec{v}d\vec{v}_1]dt. \quad (3.9)$$

We can find $D_c f$ by subtracting Equation (3.8) from Equation (3.9). Using the principle of detailed balance, the probabilities for inverse collisions are equal,

$$\sigma'(\vec{v}\,',\vec{v}\,'_1\to\vec{v},\vec{v}_1)=\sigma'(\vec{v},\vec{v}_1\to\vec{v}\,'\vec{v}\,'_1), \quad (3.10)$$

and the relative velocities

$$\vec{V}\equiv\vec{v}-\vec{v}_1, \qquad \vec{V}\,'\equiv\vec{v}\,'-\vec{v}\,'_1 \quad (3.11)$$

are equal for elastic collisions. Then we write f in short form:

$$\begin{aligned} f &\equiv f(\vec{r},\vec{v},t),\\ f_1 &\equiv f(\vec{r},\vec{v}_1,t),\\ f' &\equiv f(\vec{r},\vec{v}\,',t),\\ f'_1 &\equiv f(\vec{r},\vec{v}\,'_1,t). \end{aligned}$$

Then Equation (3.7) becomes

$$D_c f = \int_{\vec{v}'_1} \int_{\vec{v}'} \int_{\vec{v}_1} (f' f'_1 - f f_1) V \sigma'(\vec{v}, \vec{v}_1 \to \vec{v}', \vec{v}'_1) d\vec{v}_1 d\vec{v}' d\vec{v}'_1. \tag{3.12}$$

We consider elastic collisions only, so $\vec{v}'$ and $\vec{v}'_1$ depend on the collision angle in the center of mass frame. The Boltzmann equation for $f(\vec{r}, \vec{v}, t)$ can then be written in the form

$$\frac{\partial f}{\partial t} = -v_z \frac{\partial f}{\partial z} + \int_{\vec{v}_1} \int_{\Omega'} |\vec{v}_1 - \vec{v}| \frac{d\sigma}{d\Omega'} (f' f'_1 - f f_1) d\Omega' d\vec{v}_1. \tag{3.13}$$

In this thesis we would like to find the change of the distribution function of hydrogen (H) due to collisions of H and protons (p) in the solar wind, including the effects of charge exchange, and also due to collisions of H and H in the Boltzmann equation. Then we can write

$$\begin{aligned} \frac{\partial f_H}{\partial t} &= -\mu v \frac{\partial f_H}{\partial z} \\ &\quad \text{(streaming)} \\ &\quad + \int_{\vec{v}_1} \int_{\Omega'} |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} [f_H(\vec{p}') f_p(\vec{p}'_1) - f_H(\vec{p}) f_p(\vec{p}_1)] d\Omega' d\vec{p}_1 \\ &\quad \text{(H-p elastic collisions including charge exchange)} \\ &\quad + \int_{\vec{v}_1} \int_{\Omega'} |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} [f_H(\vec{p}') f_H(\vec{p}'_1) - f_H(\vec{p}) f_H(\vec{p}_1)] d\Omega' d\vec{p}_1 \\ &\quad \text{(H-H elastic collisions)} \end{aligned}$$

where μ is $\cos\theta$ and $\vec{p}'$ and $\vec{p}'_1$ are the momenta after collisions. We solve for f_H, the distribution function of H. The distribution function for protons, f_p, is taken from simulations performed elsewhere.

3.2 Initial Condition and Grid Points

We define the initial distribution of hydrogen (f_H) by a Gaussian (thermal) distribution function, distributed as in Figure (3.3):

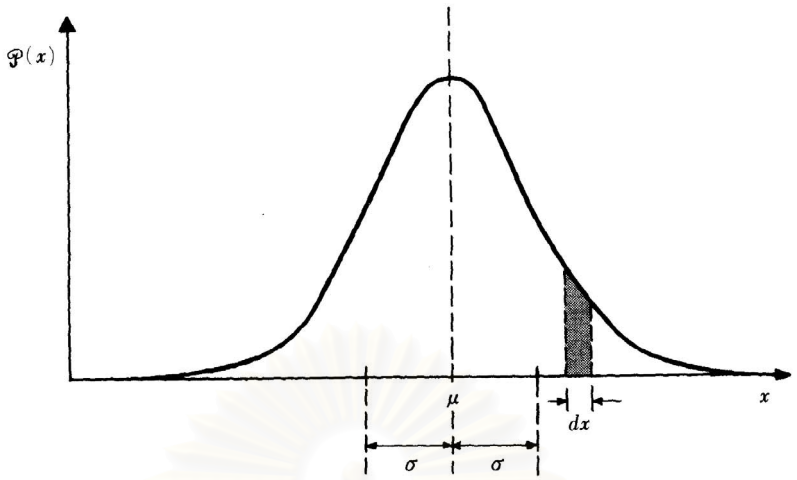

Figure 3.3: The Gaussian distribution.

Here $\mathcal{P}(x)dx$ is the area under the curve in the interval between x and $x + dx$ and the probability that the variable x lies in this range:

Gaussian distribution in 1-D:

$$f(x, v_x) = \frac{n(x)}{[2\pi\sigma^2]^{1/2}} \quad e^{-\left(\frac{{v_x}^2}{2\sigma^2}\right)}, \tag{3.14}$$

Gaussian distribution in 3-D:

$$f(x, \vec{v}) = \frac{n(x)}{[2\pi\sigma^2]^{3/2}} \quad e^{-\left(\frac{v^2}{2\sigma^2}\right)}. \tag{3.15}$$

In this thesis we consider a Gaussian distribution in 3-D and express the velocity $(\vec{v})$ in terms of momentum $(\vec{p})$ and position (x) in terms of z, so that for a distribution centered at $p = \overline{p}$ and $\mu = 0$,

$$f(z, \vec{p}) = \frac{n(z)}{[2\pi\sigma^2]^{3/2}} \quad e^{-\left(\frac{|\Delta\vec{p}|^2}{2\sigma^2}\right)}, \tag{3.16}$$

where $|\Delta\vec{p}|^2 = p^2 - 2\mu p\overline{p} + \overline{p}^2$. For a thermal distribution,

$$\frac{\sigma^2}{2m} = \frac{1}{2}kT \tag{3.17}$$

$$\sigma^2 = mkT. \tag{3.18}$$

We substituted $|\Delta\vec{p}|^2 = p^2 - 2\mu p\overline{p} + \overline{p}^2$ and $\sigma^2 = mkT$ in Equation (3.16):

$$f_H(z,\vec{p}) = \frac{n(z)}{[2\pi mkT(z)]^{3/2}} e^{-\left(\frac{p^2 - 2\mu p\overline{p} + \overline{p}^2}{2mkT(z)}\right)} \tag{3.19}$$

where $n(z)$ is the number density of particles, m is the mass of hydrogen, k is the Boltzmann constant and $T(z)$ is the temperature.

In the previous work by Worachate Boonplod (1997), the momentum grid was scaled from $\log(p) = 4.0$ to $\log(p) = 6.0$ with a step in $\log(p)$ of 0.2 and the grid in $\mu = \cos\theta$ from -1 to 1 with a step in μ of 0.1, where θ is the angle between the momentum and direction outward from the Sun (Figure 3.4).

Figure (3.4) shows that the old system of grids does not accurately sample the Gaussian distribution functions derived from Zank and Pauls (1996). Therefore we use a new system of grids in order to cover distribution functions that we calculate. We need to set up both a coarse grid and fine grid to get more accurate sampling of the region of solar wind distributions (Figure 3.5):

1. For momentum grids (p grids)

We defined p grids as below in order to cover the Gaussian distribution functions and defined fine[w]=0 for coarse grids and fine[w]=1 for fine grids, with $w = 0, .., 24$:

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

where a = 3136.125 in order to change units from km/s to eV/c.

2. We defined the μ grids by $\mu = \cos\theta$ and divided θ to cover distribution functions. We set up mu0[u] and $u = 0, .., 9$ for the coarse grids and mu1[u] for $u = 0, .., 4$ for the fine grids:

$$\text{mu0}[0] = \cos(0 \times \frac{\pi}{180}) \qquad \text{mu1}[0] = \cos(0 \times \frac{\pi}{180})$$

$$\text{mu0}[1] = \cos(15 \times \frac{\pi}{180}) \qquad \text{mu1}[1] = \cos(0 \times \frac{\pi}{180})$$

$$\text{mu0}[2] = \cos(30 \times \frac{\pi}{180}) \qquad \text{mu1}[2] = \cos(2 \times \frac{\pi}{180})$$

$$\text{mu0}[3] = \cos(60 \times \frac{\pi}{180}) \qquad \text{mu1}[3] = \cos(4 \times \frac{\pi}{180})$$

$$\text{mu0}[4] = \cos(90 \times \frac{\pi}{180}) \qquad \text{mu1}[4] = \cos(6 \times \frac{\pi}{180})$$

$$\text{mu0}[5] = \cos(120 \times \frac{\pi}{180})$$

$$\text{mu0}[6] = \cos(150 \times \frac{\pi}{180})$$

$$\text{mu0}[7] = \cos(165 \times \frac{\pi}{180})$$

$$\text{mu0}[8] = \cos(175 \times \frac{\pi}{180})$$

$$\text{mu0}[9] = \cos(180 \times \frac{\pi}{180})$$

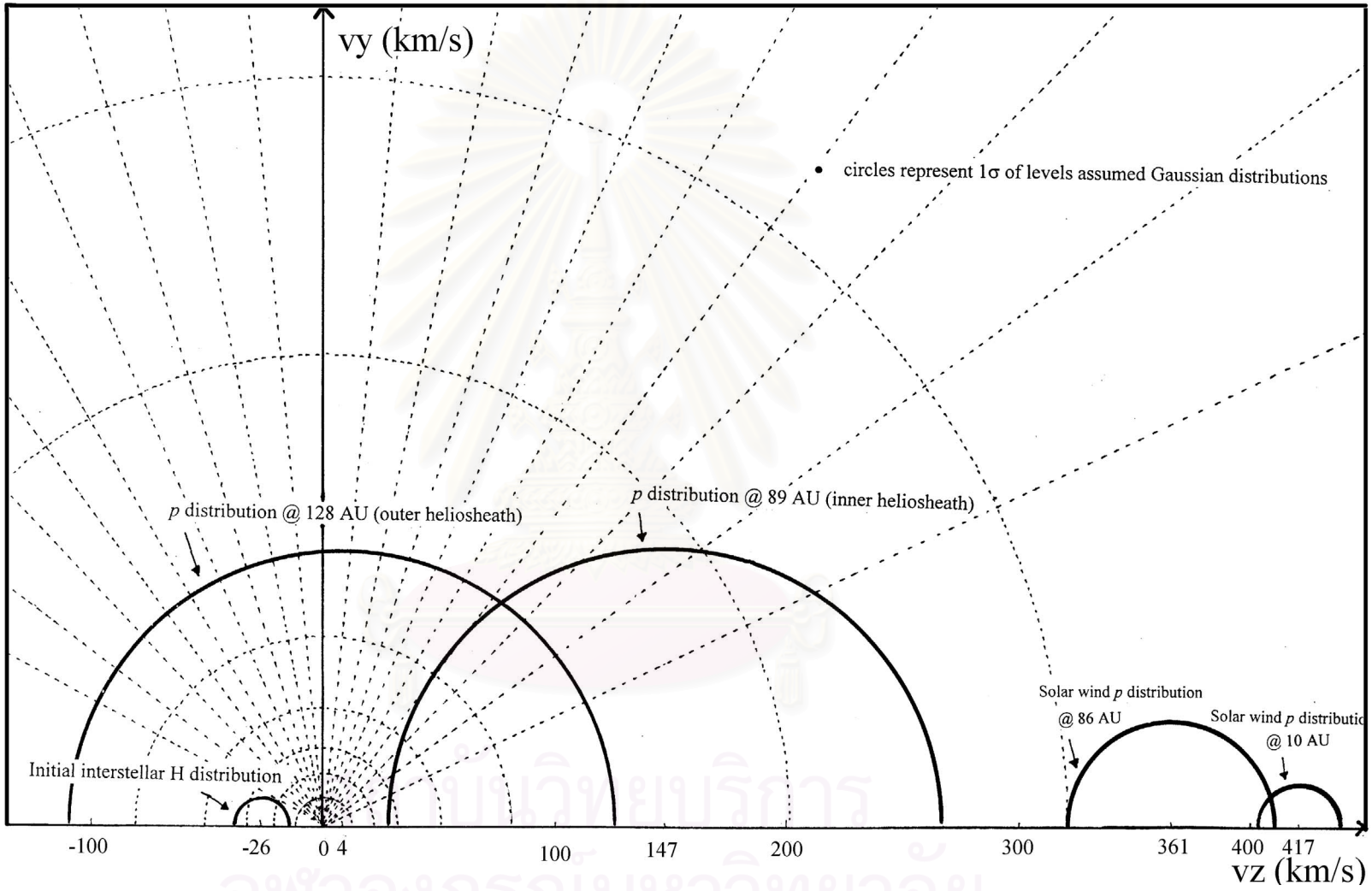

Figure 3.4: The old system of grids.

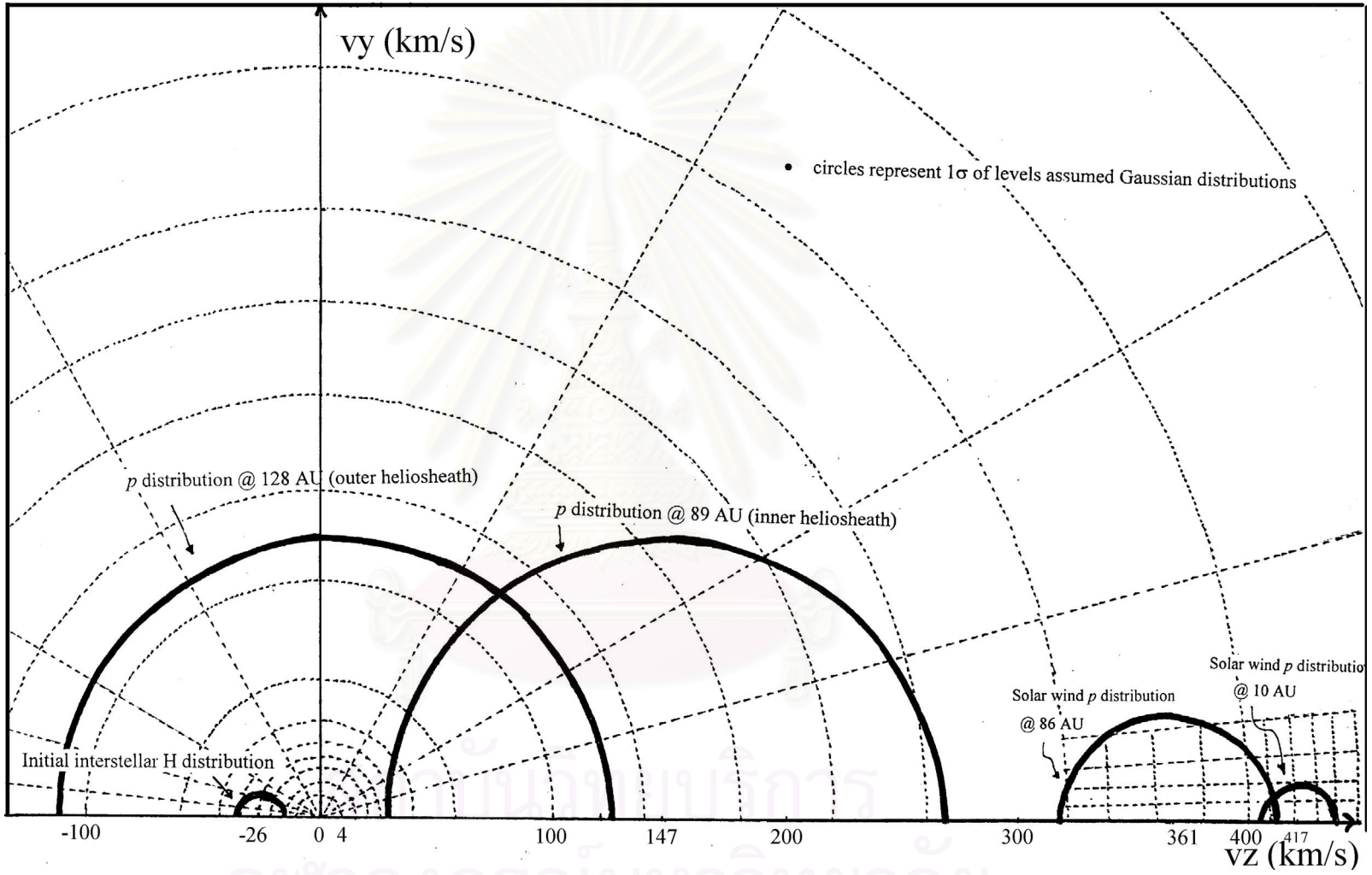

Figure 3.5: The new system of grids.

Chapter 4

Numerical Method

In this chapter we describe the numerical method that we use to solve the equation that we derived in chapter 3, that is

$$\begin{aligned}\frac{\partial f_H}{\partial t} &= -\mu v\frac{\partial f_H}{\partial z} \\ &\quad \text{(streaming)} \\ &\quad + \int_{\vec{v}_1}\int_{\Omega'} |\vec{v}-\vec{v}_1|\frac{d\sigma}{d\Omega'}[f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1)]d\Omega' d\vec{p}_1 \\ &\quad \text{(H-}p\text{ elastic collision including charge exchange)} \\ &\quad + \int_{\vec{v}_1}\int_{\Omega'} |\vec{v}-\vec{v}_1|\frac{d\sigma}{d\Omega'}[f_H(\vec{p}\,')f_H(\vec{p}\,'_1) - f_H(\vec{p})f_H(\vec{p}_1)]d\Omega' d\vec{p}_1, \\ &\quad \text{(H-H elastic collision)} \end{aligned} \tag{4.1}$$

where $\mu = \cos\theta$ and $\vec{p}\,'$ and $\vec{p}\,'_1$ are the momenta after collisions. We solve Equation (4.1) by a finite difference method by calculating f_H as an array, with f[w][l][u] representing $f_H(p_w, Z_l, \mu_u)$ at each timestep t for a specially designed grid over a range of momentum, $p = mv$ for $v = 0 - 500$ km/s, distance $z = 0 - 200$ AU from the Sun, and $\mu = \cos\theta$ for $\theta = 0 - 180°$.

We choose to use Gaussian integration to solve the integral terms. We used Gauss-Laguerre integration for a weight function corresponding to $d\sigma/d\Omega'$, as approximated by an exponential function, which makes the most efficient use

of grid points. Such efficiency is critical for the overall computational efficiency, because the integrals need to be evaluated at each time step and grid point.

We separated Equation (4.1) into 3 terms:

1. Advection (first term)
2. H-p elastic collisions including charge exchange (second term)
   - H-p collisions include elastic collisions for $\theta' \approx 0$
   - H-p collisions also include charge exchange for $\theta' \approx \pi$
3. H-H elastic collisions (third term)
   - H-H collisions include only elastic collisions for $\theta' \approx 0$

4.1 Advection term

Considering only the first term on the right hand side of (4.1), called the streaming term, we obtain an advection equation:

$$\frac{\partial f_H}{\partial t} = -\mu v \frac{\partial f_H}{\partial z} \tag{4.2}$$

Advection problems describe the flow of particles, in which a physical property is carried through space by the motion of the medium occupying that space. In this thesis we neglect the advection term.

4.2 Approximation Formulae for Cross Sections

4.2.1 H-p elastic collisions including charge exchange

We consider

$$\int_{\vec{v}_1}\int_{\Omega'} |\vec{v} - \vec{v}_1| \frac{d\sigma}{d\Omega'} [f_H(\vec{p}')f_p(\vec{p}'_1) - f_H(\vec{p})f_p(\vec{p}_1)] d\Omega' d\vec{p}_1. \tag{4.3}$$

From $\int \ldots d\Omega' = \int_0^{2\pi}\int_0^{\pi} \ldots \sin\theta' d\theta' d\varphi'$, (4.3) can then be written

$$\int_{\vec{p_1}} |\vec{v}-\vec{v}_1| \left\{ \int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta'; |\vec{v}-\vec{v}_1|) \int_0^{2\pi} \left[f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1) \right] d\varphi' \sin\theta' d\theta' \right\} d^3p_1, \tag{4.4}$$

and we approximate (4.4) as a summation,

$$\int_{\vec{p_1}} |\vec{v}-\vec{v}_1| \left\{ \int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta'; |\vec{v}-\vec{v}_1|) \left[\sum_{\varphi' \;\; grid} \left[f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1) \right] \Delta\varphi' \right] \sin\theta' d\theta' \right\} d^3p_1. \tag{4.5}$$

.

We split (4.5) into a product of 3 terms:

$$\int_{\vec{p_1}} \underbrace{|\vec{v}-\vec{v}_1|}_{\text{term 1}} \underbrace{\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta'; |\vec{v}-\vec{v}_1|) \sum_{\varphi' \;\; grid} \left[f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1) \right] \Delta\varphi' \sin\theta' d\theta'}_{\text{term 2}} \underbrace{d^3p_1}_{\text{term 3}} . \tag{4.6}$$

Term 1 is $|\vec{v}-\vec{v}_1|$, which we call the"velocity term."

Term 2 is $\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta'; |\vec{v}-\vec{v}_1|) \sum_{\varphi' \;\; grid} \left[f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1) \right] \Delta\varphi' \sin\theta' d\theta'.$

Term 3 is d^3p_1, which we call the"volume term."

Now we consider in term 2:

$$\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta'; |\vec{v}-\vec{v}_1|) \sum_{\varphi' \;\; grid} \left[f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1) \right] \Delta\varphi' \sin\theta' d\theta'. \tag{4.7}$$

In order to get a simpler form, we define

$$S_1(\theta') = \sum_{\varphi' \;\; grid} \left[f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1) \right] \Delta\varphi', \tag{4.8}$$

so that (4.7) becomes

$$\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta'; |\vec{v}-\vec{v}_1|) S_1(\theta') \sin\theta' d\theta'. \tag{4.9}$$

• $\mathrm{d}\sigma/\mathrm{d}\Omega'$ **for H-p elastic collisions** ($\theta' \approx 0$)

We changed $d\theta'$ into dx, where $x \equiv 1 - \cos\theta'$, so that $dx = \sin\theta' d\theta'$, and

$$\begin{aligned} \theta' = 0 \quad &\rightarrow \quad x = 0 \\ \theta' = \pi \quad &\rightarrow \quad x = 2. \end{aligned}$$

Then (4.9) can be written

$$\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta'; |\vec{v} - \vec{v}_1|) S_1(\theta') \sin\theta' d\theta' \;=\; \int_0^{2} \frac{d\sigma}{d\Omega'}(x; |\vec{v} - \vec{v}_1|) S_1(x) dx. \quad (4.10)$$

We approximate

$$\int_0^{2} \frac{d\sigma}{d\Omega'}(x; |\vec{v} - \vec{v}_1|) S_1(x) dx \approx \int_0^{\infty} A e^{-ax} S_1(x) dx. \quad (4.11)$$

because $d\sigma/d\Omega'$ drops off very rapidly with increasing θ', and (for the example of E_{cm}=100 eV) is well approximated by $e^{14.955 - 80069\theta^2}$, so that at a given $|\vec{v} - \vec{v}_1|$, $d\sigma/d\Omega' \approx Ae^{-ax}$ for some constants A and a. Thus we approximate term 2 by

$$\int_0^{\infty} A e^{-ax} \sum_{\varphi' \;\; grid} \left[f_H(\vec{p}\,') f_p(\vec{p}\,'_1) - f_H(\vec{p}) f_p(\vec{p}_1) \right] \Delta\varphi' dx. \quad (4.12)$$

We calculated A and a as functions of the center of mass collision energy (E_{cm}) by approximating the differential cross section $d\sigma/d\Omega'$ from data in http://www-cfadc.phy.ornl.gov/cgi-bin/kelastic.cgi:

1. We plotted $d\sigma/d\Omega'$ for the elastic collisions (el) vs. x for the center of mass (CM) scattering angle (θ') for H^++H at E_{cm} values of 1 eV, 3.16 eV, 10 eV, 31.6 eV, and 100 eV (Figures 4.1 to 4.5).

2. We plotted $\log(A)$ and $\log(a)$ vs. $\log(E_{cm})$ and added trend lines in the graphs (Figures 4.6 and 4.7).

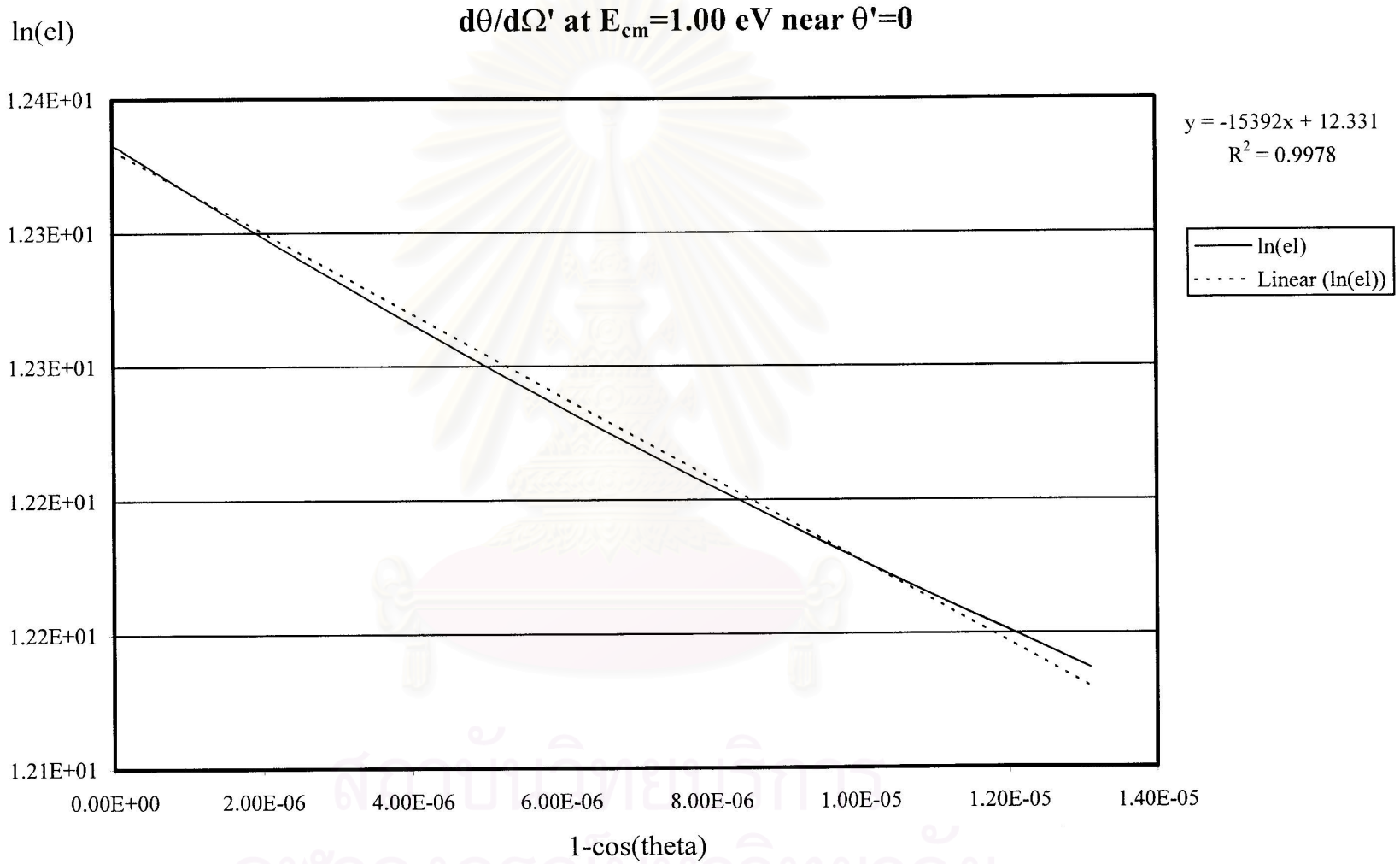

Figure 4.1: Elastic collisions (el, in a.u.) vs. CM scattering angle (theta) at 1.00 eV.

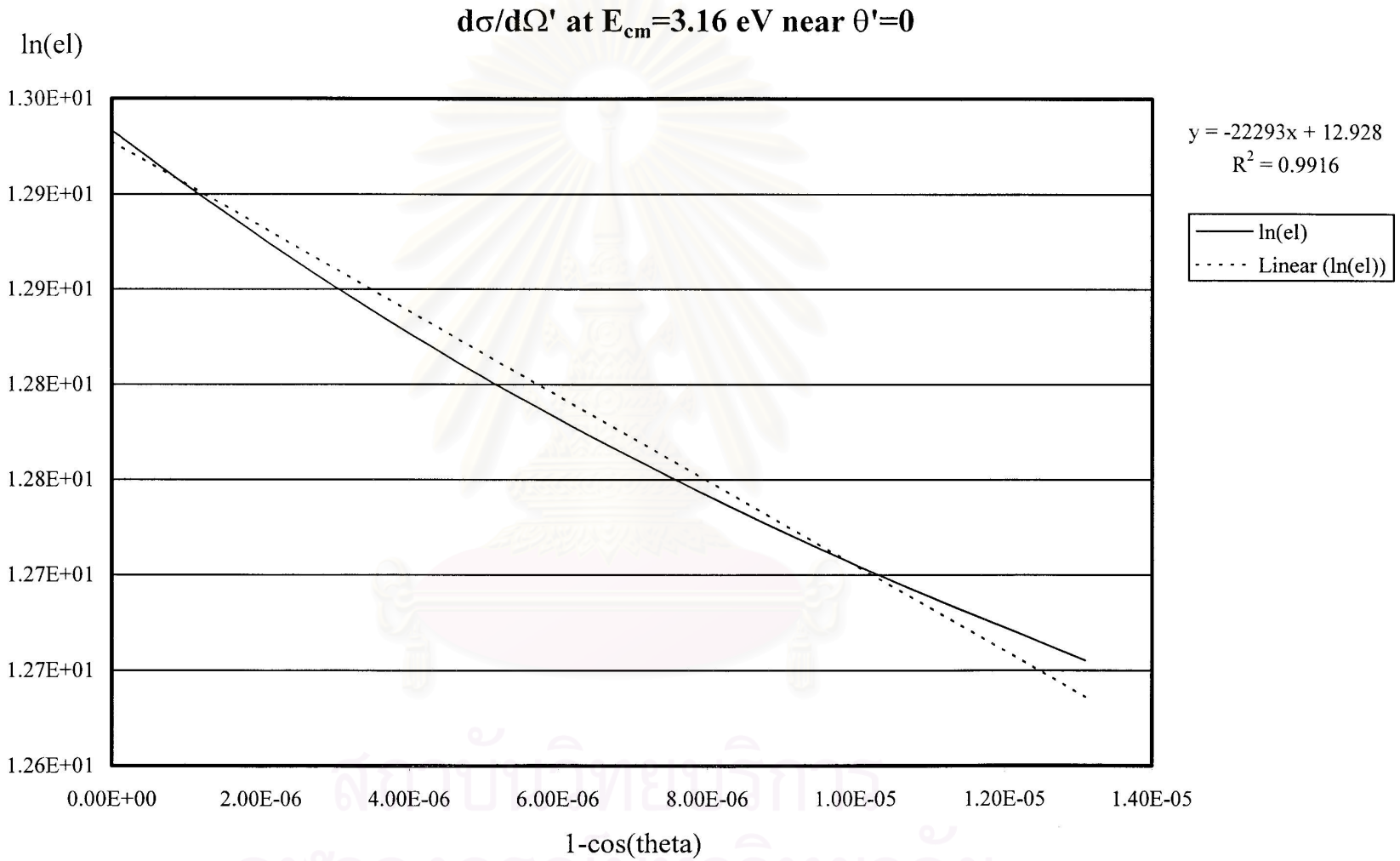

Figure 4.2: Elastic collisions (el, in a.u.) vs. CM scattering angle (theta) at 3.16 eV.

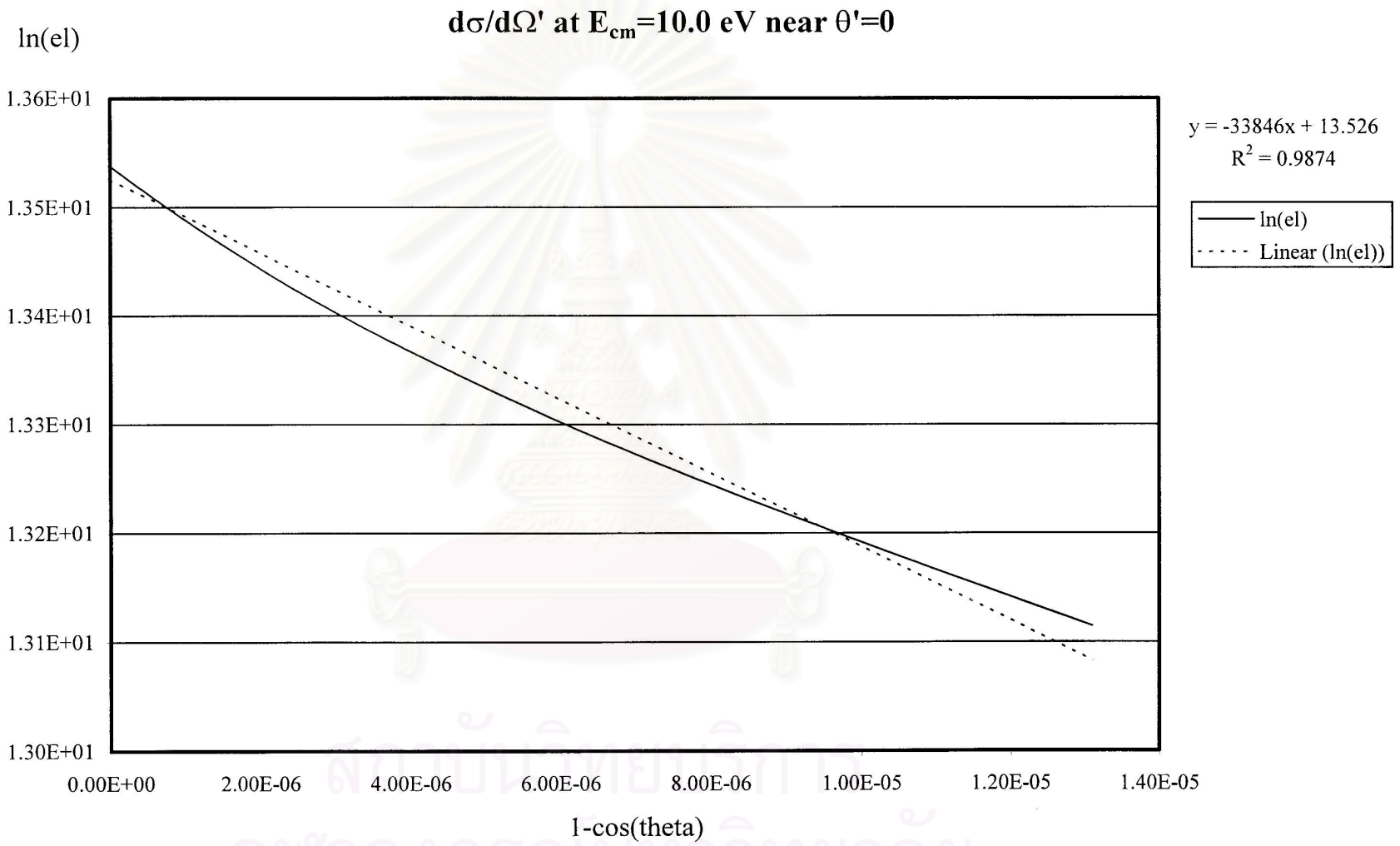

Figure 4.3: Elastic collisions (el, in a.u.) vs. CM scattering angle (theta) at 10.0 eV.

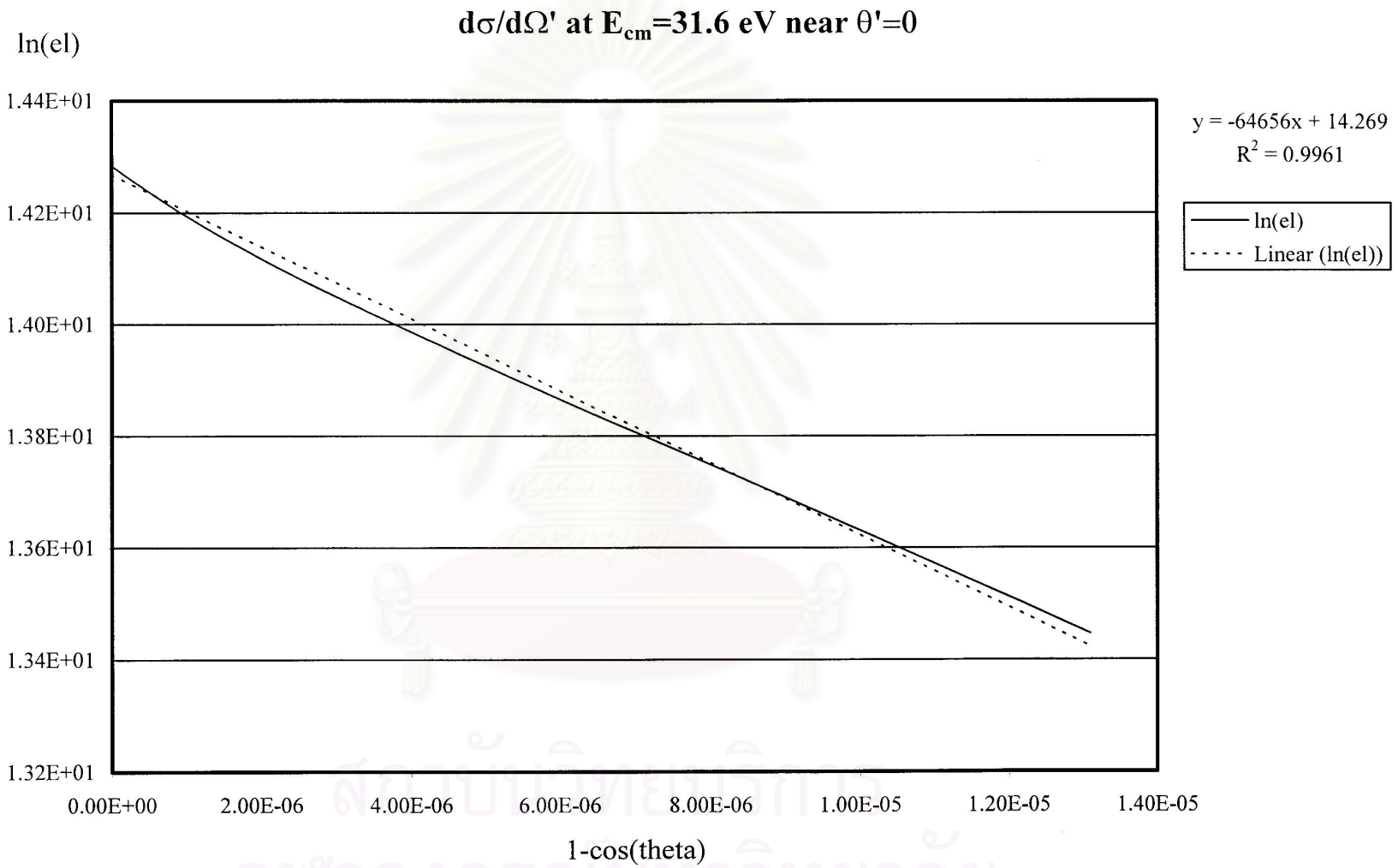

Figure 4.4: Elastic collisions (el, in a.u.) vs. CM scattering angle (theta) at 31.6 eV.

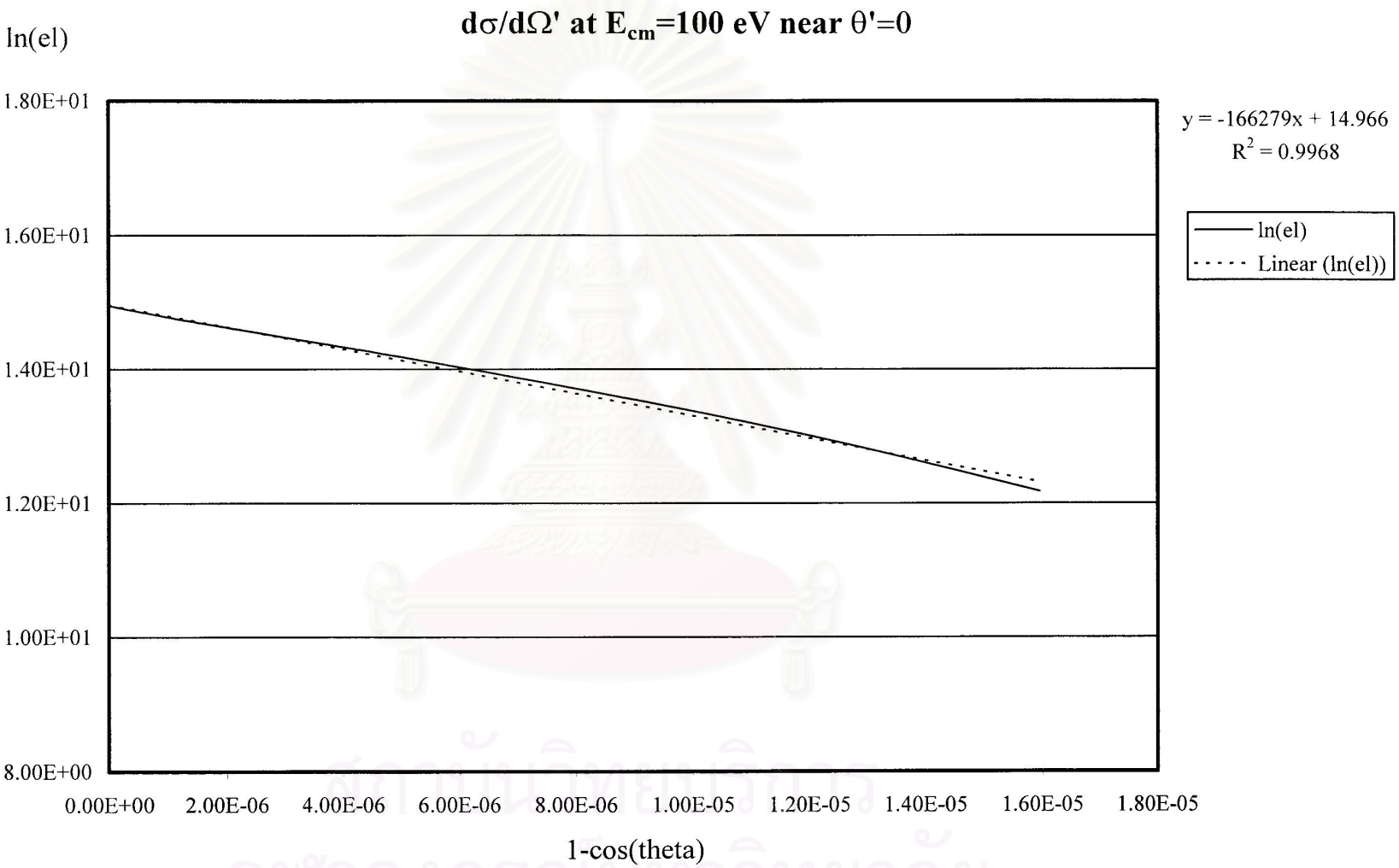

Figure 4.5: Elastic collisions (el, in a.u.) vs. CM scattering angle (theta) at 100 eV.

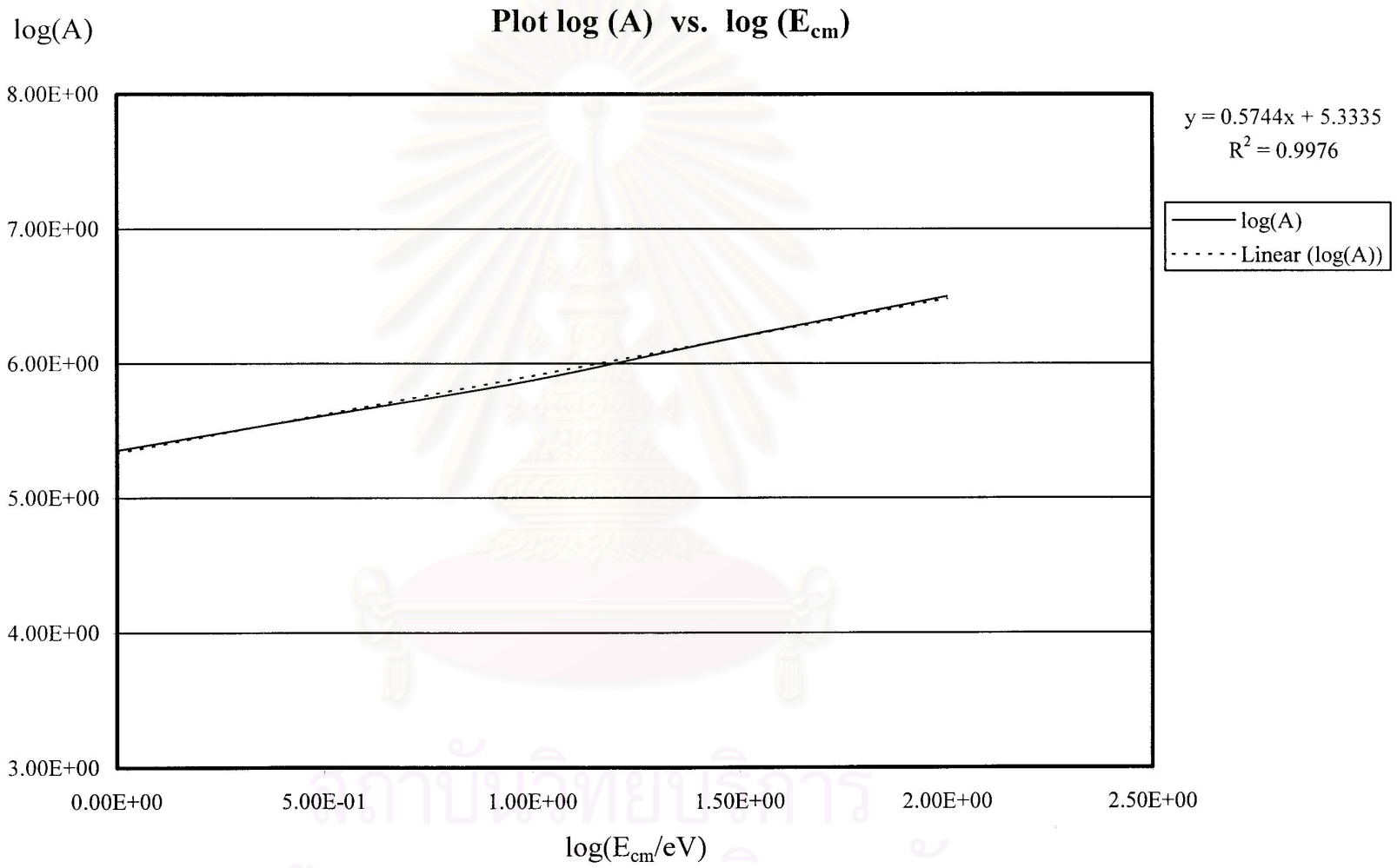

Figure 4.6: Plot of $\log(A)$ (in a.u.) vs. $\log(E_{cm})$ (in eV).

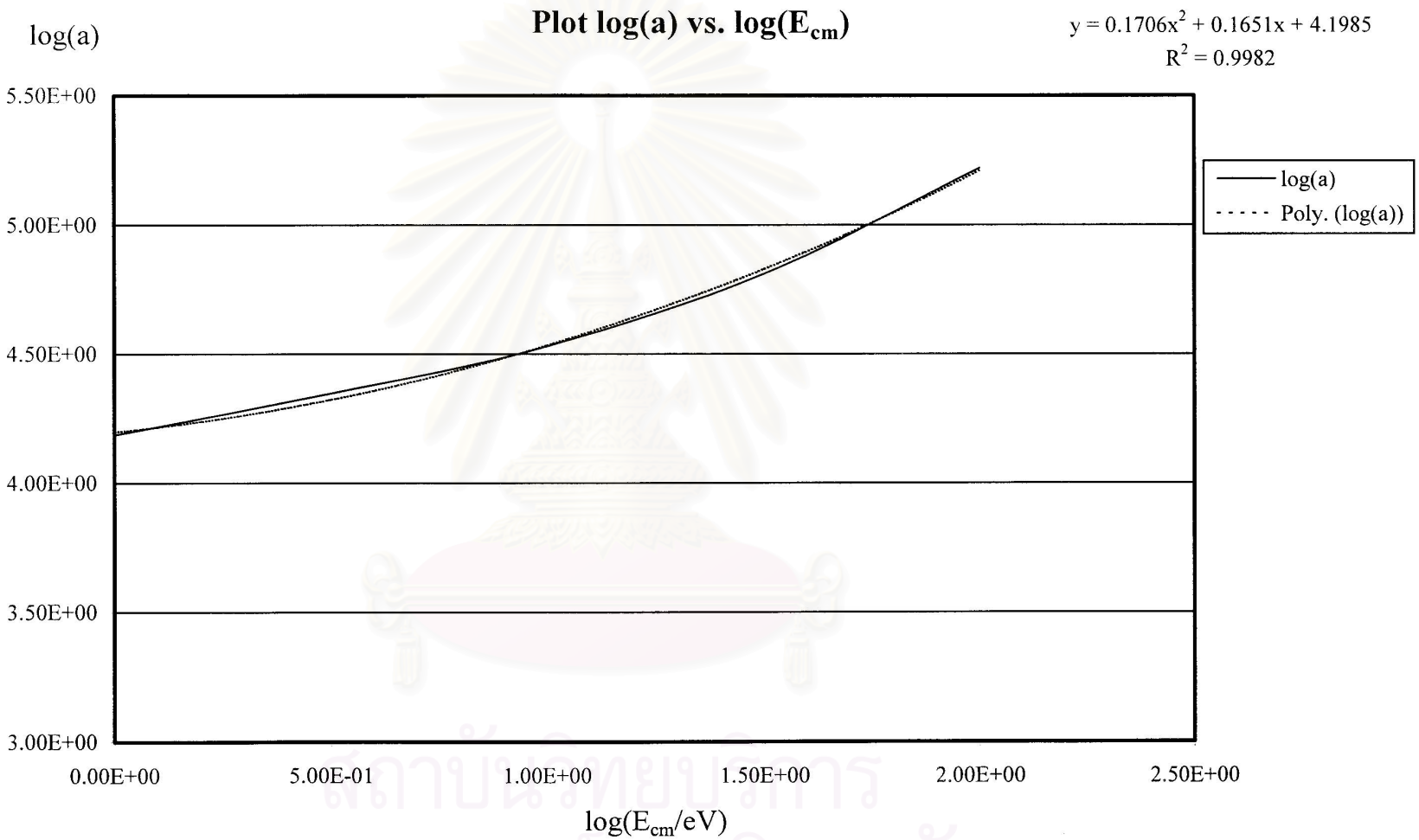

Figure 4.7: Plot of $\log(a)$ (in a.u.) vs. $\log(E_{cm})$ (in eV).

Table 4.1: Trendline equations and values of A and a from Figures 4.1 to 4.5.

E_{cm} (in eV)	Trendline Equation	A	a
1.00	$y = -1.54 \times 10^4 x + 12.3$	$e^{12.3}$	1.54×10^3
3.16	$y = -2.23 \times 10^4 x + 12.9$	$e^{12.9}$	2.23×10^4
10.0	$y = -3.358 \times 10^4 x + 13.5$	$e^{13.5}$	3.38×10^4
31.6	$y = -6.47 \times 10^4 x + 14.3$	$e^{14.3}$	6.47×10^5
100	$y = -1.66 \times 10^5 x + 15.0$	$e^{15.0}$	1.66×10^5

Note that $d\sigma/d\Omega'$ and A are expressed in atomic units (a.u.); the atomic unit for area is 2.8×10^{-18} cm^2. Note also that the formula for E_{cm} is

$$E_{cm} = \frac{1}{4} M V_{rel}^2, \tag{4.13}$$

where M is the mass of a hydrogen atom, about 938786000 eV/c^2, and V_{rel} is the relative collision velocity:

$$\begin{aligned} V_{rel} &= \frac{1}{M}|\vec{p} - \vec{p}_1| \\ &= \frac{\sqrt{(\vec{p} - \vec{p}_1)(\vec{p} - \vec{p}_1)}}{M} \\ &= \frac{\sqrt{p^2 - 2\vec{p} \cdot \vec{p}_1 + {p_1}^2}}{M} \end{aligned} \tag{4.14}$$

Considering $\vec{p} \cdot \vec{p}_1$, we obtain $\vec{p}$ in Cartesian coordinates (p_x, p_y, p_z) from $\vec{p}$ in spherical coordinates (p, θ, φ) by

$$\begin{aligned} p_x &= p \sin\theta \cos\varphi \\ p_y &= p \sin\theta \sin\varphi \\ p_z &= p \cos\theta. \end{aligned}$$

Then

$$\begin{aligned} \vec{p} &= (p \sin\theta \cos\varphi, p \sin\theta \sin\varphi, p \cos\theta) \\ \vec{p}_1 &= (p_1 \sin\theta_1 \cos\varphi_1, p_1 \sin\theta_1 \cos\varphi_1, p_1 \cos\theta_1) \\ \vec{p} \cdot \vec{p}_1 &= p_x p_{1x} + p_y p_{1y} + p_z p_{1z}, \end{aligned}$$

so that

$$\vec{p}\cdot\vec{p}_1 = (p\sin\theta\cos\varphi)(p_1\sin\theta_1\cos\varphi_1) + (p\sin\theta\sin\varphi)(p_1\sin\theta_1\sin\varphi_1) + (p\cos\theta)(p_1\cos\theta_1). \quad (4.15)$$

We assume azimutal symmetry in $\vec{p}$ space, and choose to use $\vec{p}$-grid points along $\varphi = 0$, where $\sin\varphi = 0$ and $\cos\varphi = 1$. Then

$$\vec{p}\cdot\vec{p}_1 = pp_1(\sin\theta\cos\varphi_1\sin\theta_1) + pp_1\cos\theta\cos\theta_1. \quad (4.16)$$

We substitute Equation (4.16) into Equation (4.14), and then Equation (4.14) becomes

$$V_{rel} = \frac{\sqrt{p^2 - 2pp_1(\sin\theta\sin\theta_1\cos\varphi_1 + \cos\theta\cos\theta_1) + p_1^2}}{M} \quad (4.17)$$

so that

$$E_{cm} = \frac{p^2 - 2pp_1(\sin\theta\sin\theta_1\cos\varphi + \cos\theta\cos\theta_1) + p_1^2}{4M}. \quad (4.18)$$

• $d\sigma/d\Omega'$ **for H-p charge exchange** ($\theta' \approx \pi$)

We changed $d\theta'$ into dx, where $x \equiv 1 + \cos\theta'$, so that $dx = -\sin\theta' d\theta'$, and

$$\theta' = 0 \quad \rightarrow \quad x = 2$$
$$\theta' = \pi \quad \rightarrow \quad x = 0.$$

Then (4.9) can be written

$$\int_0^\pi \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)S_1(\theta')\sin\theta' d\theta' = -\int_\pi^0 \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)S_1(\theta')\sin\theta' d\theta'$$
$$= \int_0^2 \frac{d\sigma}{d\Omega'}(x;|\vec{v}-\vec{v}_1|)S_1(x)dx. \quad (4.19)$$

We approximate

$$\int_0^2 \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)S_1(\theta')dx \approx \int_0^\infty Be^{-bx}S_1(x)dx. \quad (4.20)$$

Thus we approximate term 2 by

$$\int_0^\infty Be^{-bx} \sum_{\varphi' \ grid} \left[f_H(\vec{p}\,') f_p(\vec{p}\,'_1) - f_H(\vec{p}) f_p(\vec{p}_1) \right] \Delta\varphi' dx. \tag{4.21}$$

1. We plotted $d\sigma/d\Omega'$ for the charge transfer cross section (ct) vs. x for the center of mass (CM) scattering angle (θ') for H^++H at E_{cm} values of 1 eV, 3.16 eV, 10 eV, 31.6 eV, and 100 eV (Figures 4.8 to 4.12).

2. We plotted $\log(B)$ and $\log(b)$ vs. $\log(E_{cm})$ and added trend lines in the graphs (Figures 4.13 and 4.14).

Table 4.2: Trendline equations and values of B and b from Figures 4.8 to 4.12.

E_{cm} (in eV)	Trendline Equation	B	b
1.00	$y = -6.47 \times 10^3 x + 9.97$	$e^{9.97}$	6.47×10^3
3.16	$y = -1.68 \times 10^4 x + 11.0$	$e^{11.0}$	1.68×10^4
10.0	$y = -4.85 \times 10^4 x + 11.9$	$e^{11.9}$	4.85×10^4
31.6	$y = -1.24 \times 10^5 x + 13.0$	$e^{13.0}$	1.24×10^5
100	$y = -4.17 \times 10^5 x + 14.2$	$e^{14.2}$	4.17×10^5

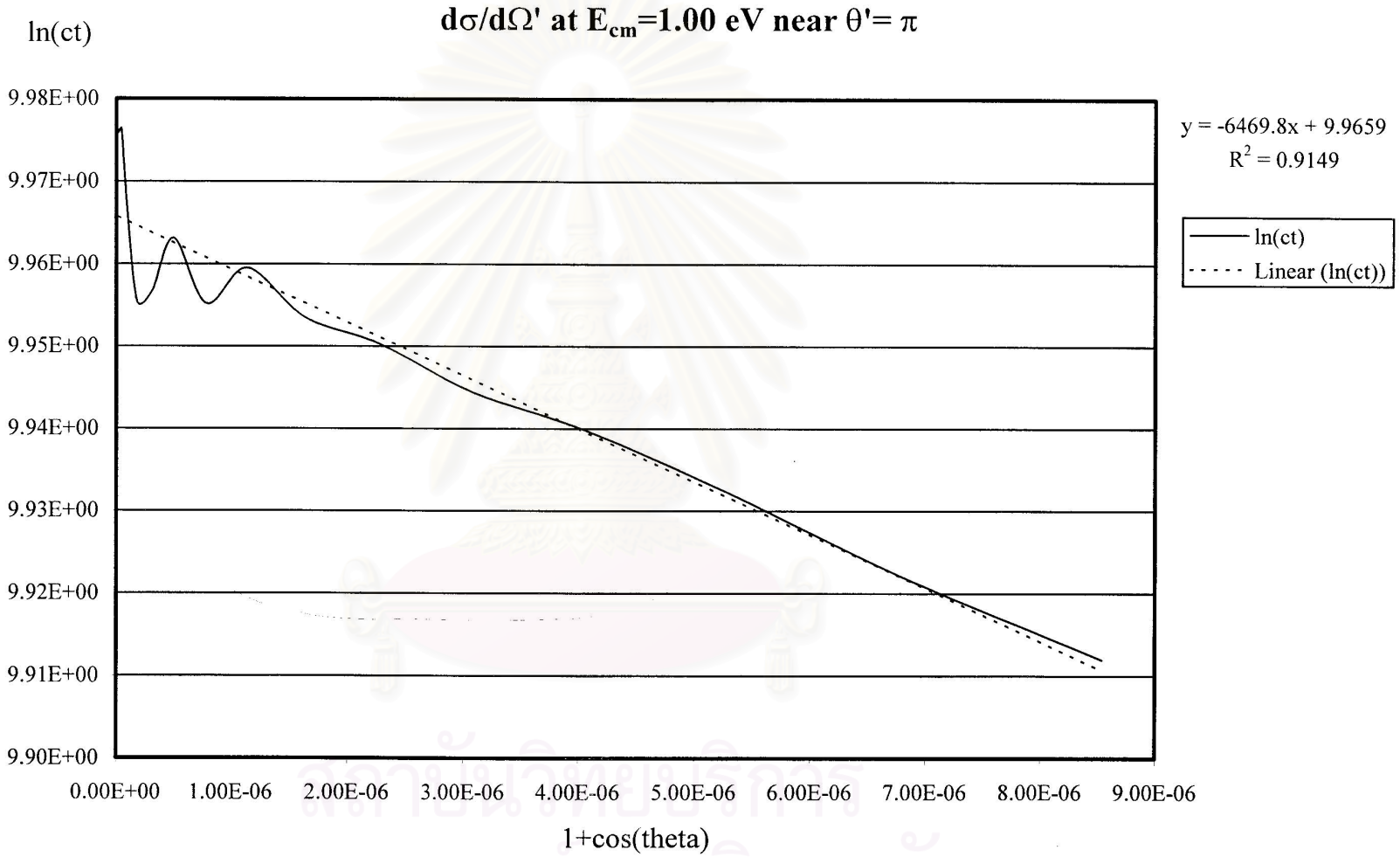

Figure 4.8: Charge transfer cross section (ct, in a.u.) vs. CM scattering angle (theta) at 1.00 eV.

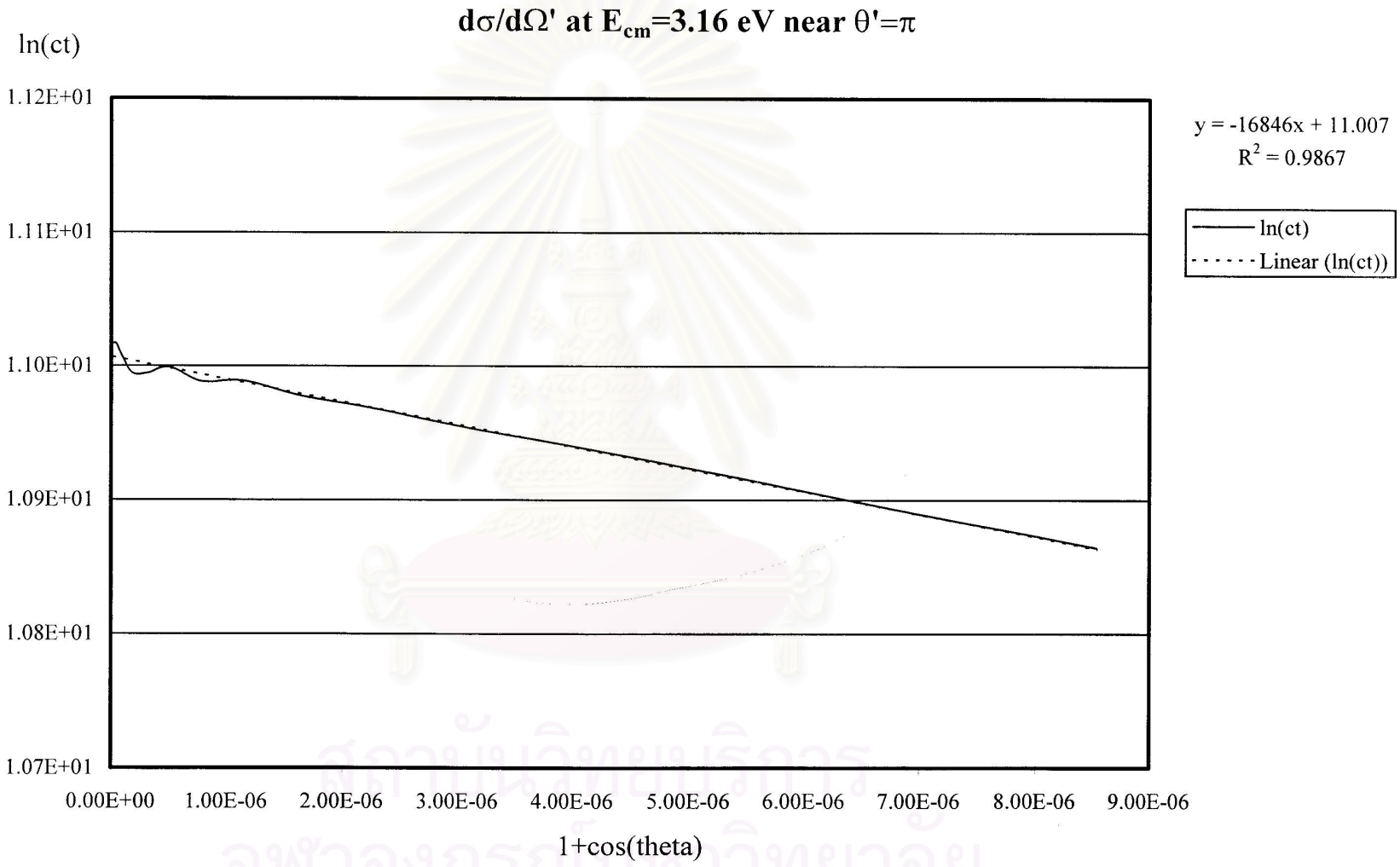

Figure 4.9: Charge transfer cross sections (ct, in a.u.) vs. CM scattering angle (theta) at 3.16 eV.

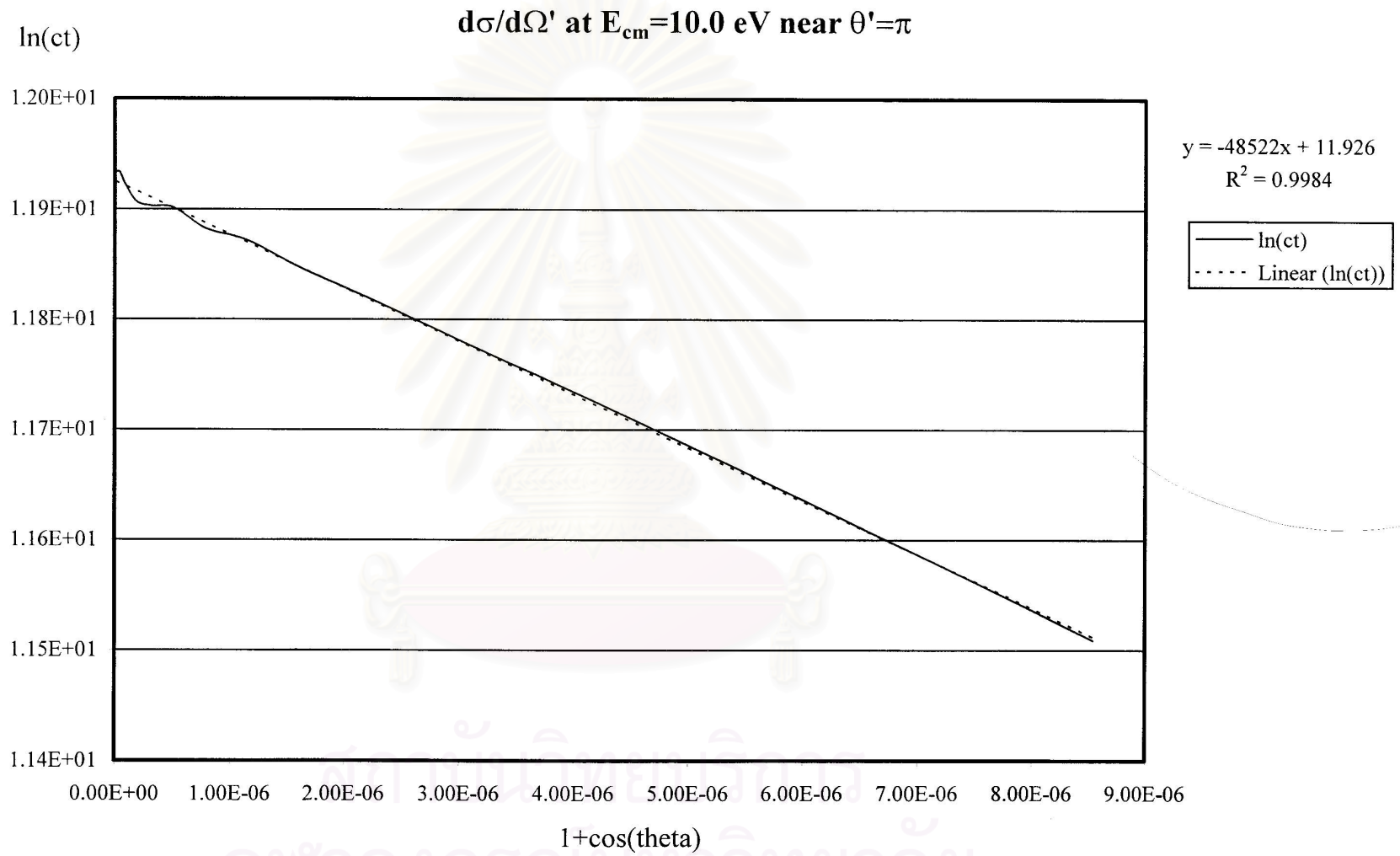

Figure 4.10: Charge transfer cross section (ct, in a.u.) vs. CM scattering angle (theta) at 10.0 eV.

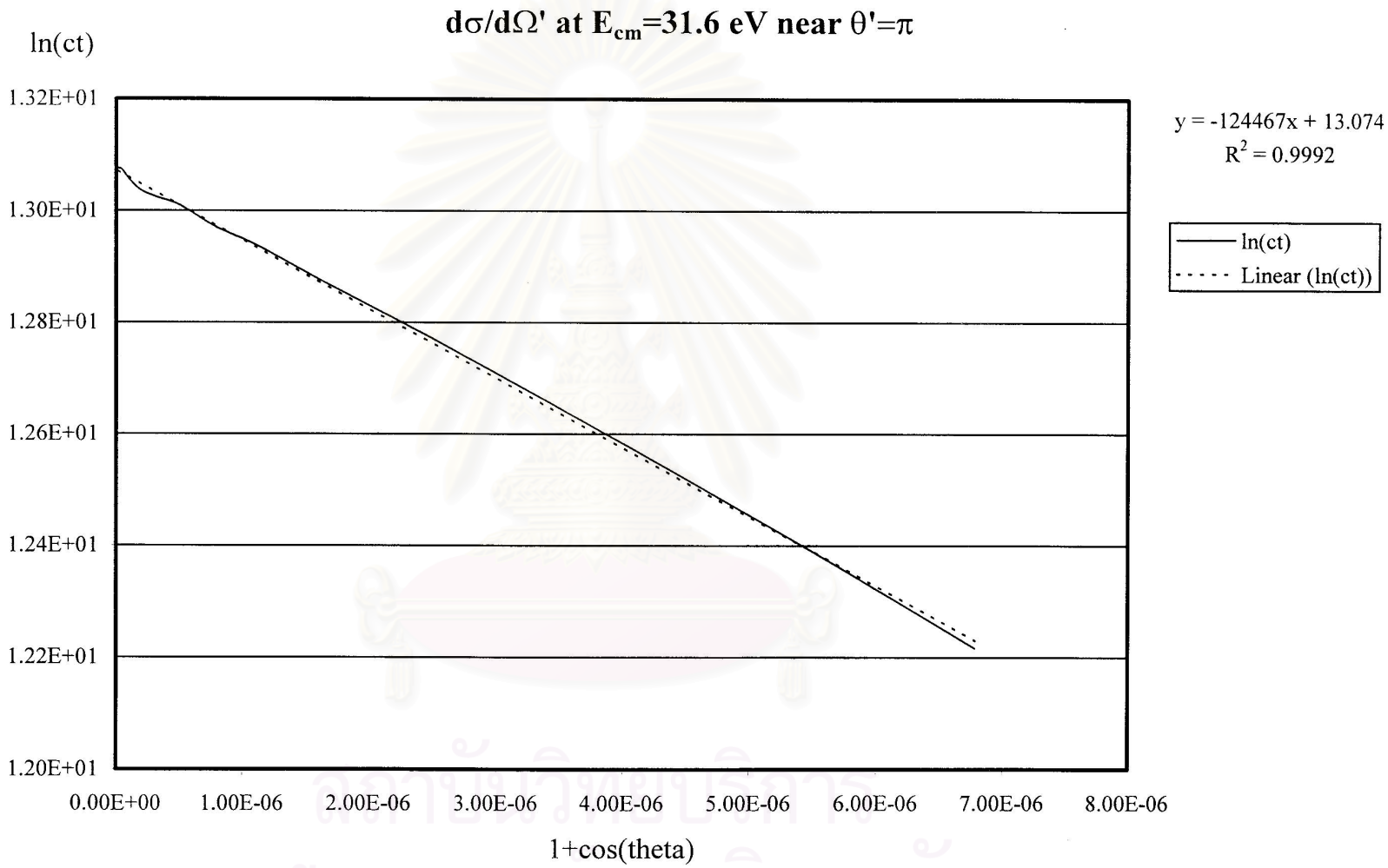

Figure 4.11: Charge transfer cross section (ct, in a.u.) vs. CM scattering angle (theta) at 31.6 eV.

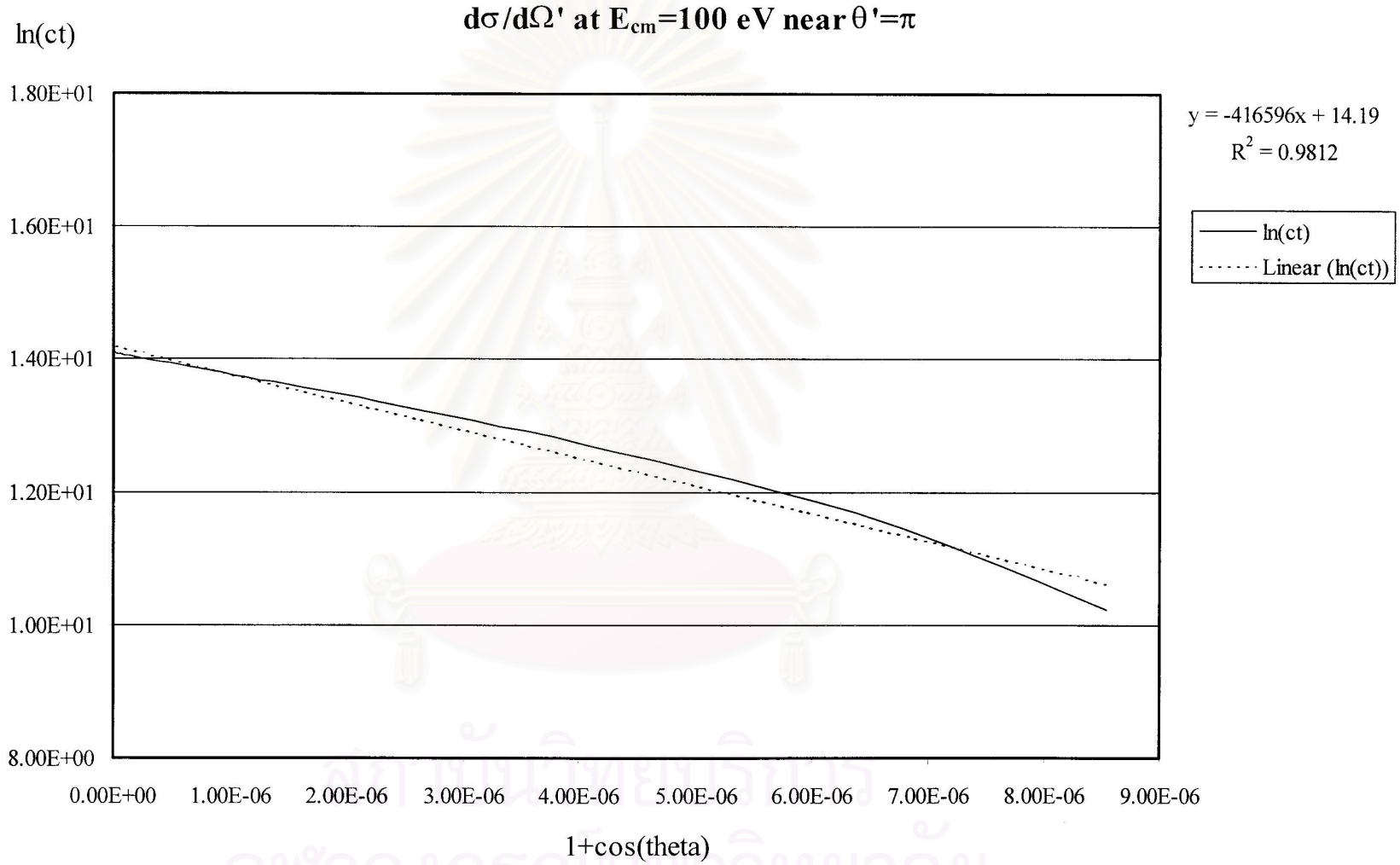

Figure 4.12: Charge transfer cross section (ct, in a.u.) vs. CM scattering angle (theta) at 100 eV.

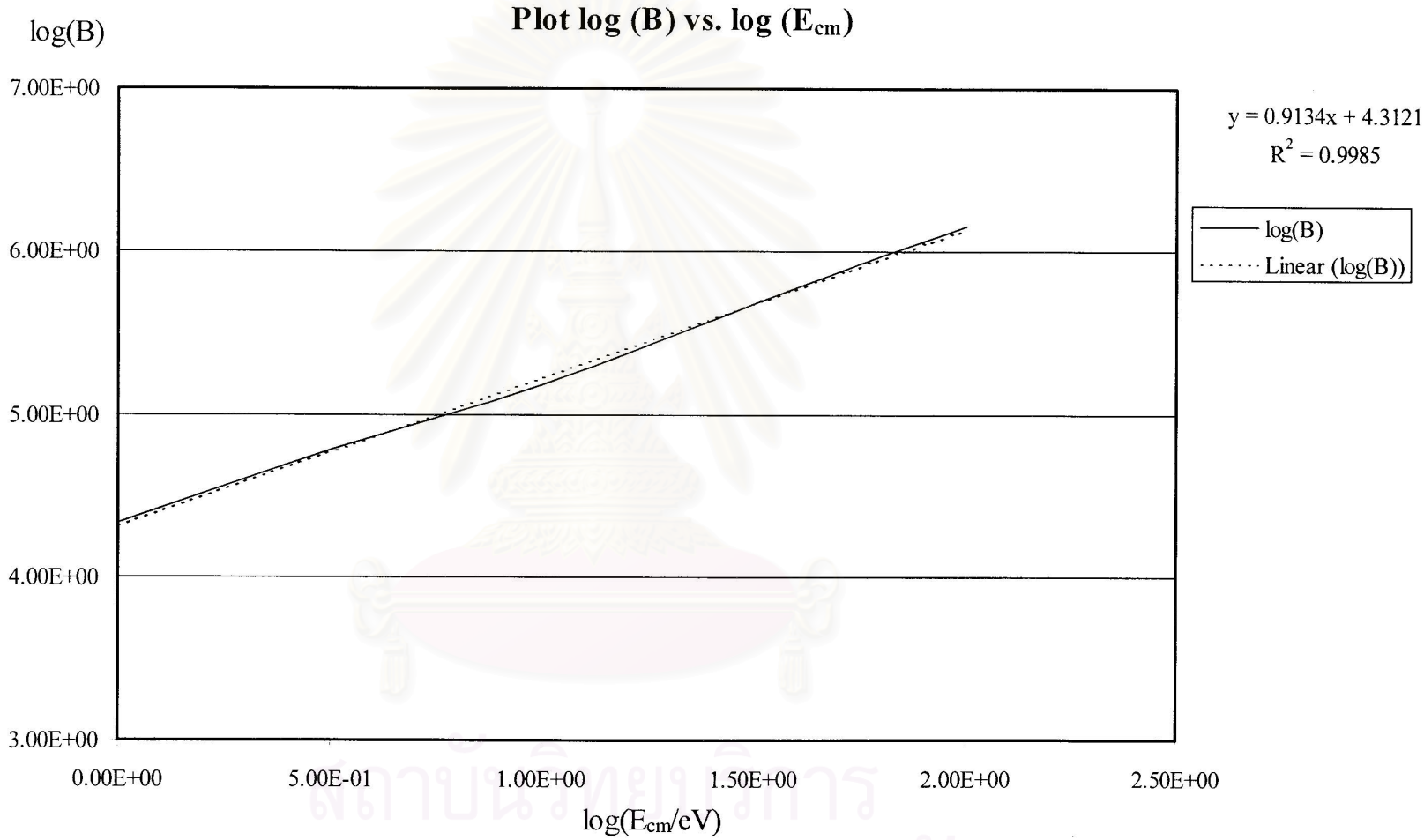

Figure 4.13: Plot of $\log(B)$ (in a.u.) vs. $\log(E_{cm})$ (in eV).

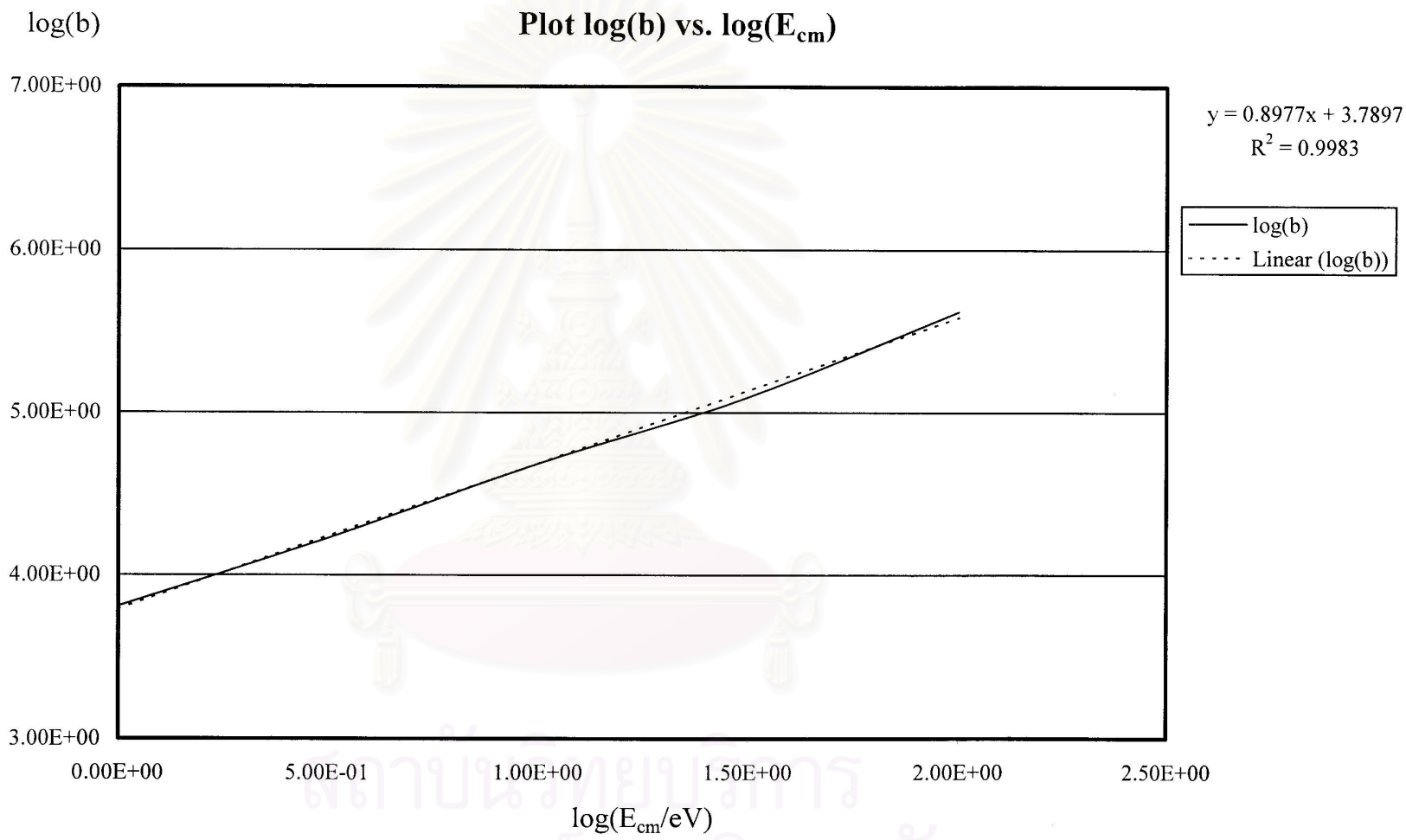

Figure 4.14: Plot of $\log(b)$ (in a.u.) vs. $\log(E_{cm})$ (in eV).

4.2.2 H-H elastic collisions

We consider

$$\int_{\vec{v}_1}\int_{\Omega'} |\vec{v}-\vec{v}_1|\frac{d\sigma}{d\Omega'}[f_H(\vec{p}\,')f_H(\vec{p}\,'_1)-f_H(\vec{p})f_H(\vec{p_1}\,')]d\Omega' d\vec{p}_1, \tag{4.22}$$

and as in H-p elastic collisions we get

$$\int_{\vec{P}_1} \underbrace{|\vec{v}-\vec{v}_1|}_{\text{term 1}} \underbrace{\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)\sum_{\varphi'\ grid}[f_H(\vec{p}\,')f_H(\vec{p}\,'_1)-f_H(\vec{p})f_H(\vec{p}_1)]\,\Delta\varphi'\sin\theta' d\theta'}_{\text{term 2}} \underbrace{d^3p_1}_{\text{term 3}}. \tag{4.23}$$

For the H-H elastic collision part, terms 1 and 3 are similar to those for H-p elastic collisions. Next, we consider term 2 of H-H elastic collisions,

$$\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)\sum_{\varphi'\ grid}[f_H(\vec{p}\,')f_H(\vec{p}\,'_1)-f_H(\vec{p})f_H(\vec{p}_1)]\,\Delta\varphi'\sin\theta' d\theta'. \tag{4.24}$$

We substitute

$$S_2(\theta') = \sum_{\varphi'\ grid}[f_H(\vec{p}\,')f_H(\vec{p}\,'_1)-f_H(\vec{p})f_H(\vec{p}_1)]\,\Delta\varphi' \tag{4.25}$$

so that (4.23) becomes

$$\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)S_2(\theta')\sin\theta' d\theta'. \tag{4.26}$$

• $d\sigma/d\Omega'$ **for H-H elastic collisions** ($\theta' \approx 0$)

We changed $d\theta'$ into dy where $y = 1-\cos\theta'$, so that $dy = \sin\theta' d\theta'$

$$\begin{aligned} \theta' = 0 \quad &\rightarrow \quad y = 0 \\ \theta' = \pi \quad &\rightarrow \quad y = 2. \end{aligned}$$

From (4.26) can be written

$$\int_0^{\pi} \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)S_2(\theta')\sin\theta' d\theta' = \int_0^{2} \frac{d\sigma}{d\Omega'}(y;|\vec{v}-\vec{v}_1|)S_2(y)dy. \tag{4.27}$$

We approximate

$$\int_0^2 \frac{d\sigma}{d\Omega'}(\theta';|\vec{v}-\vec{v}_1|)S_2(\theta')dy \approx \int_0^\infty Ce^{-cy}S_2(y)dy \tag{4.28}$$

Thus we approximate term 2 by

$$\int_0^\infty Ce^{-cy} \sum_{\varphi' \ grid} \left[f_H(\vec{p}')f_H(\vec{p}'_1) - f_H(\vec{p})f_H(\vec{p}_1)\right]\Delta\varphi' dy. \tag{4.29}$$

1. We plotted $d\sigma/d\Omega'$ for the elastic collisions (el) vs. y for the center of mass (CM) scattering angle (θ') for H+H at E_{cm} values of 1 eV, 3.16 eV, 10 eV, 31.6 eV, and 100 eV (Figures 4.15 to 4.19).

2. We plotted $\log(C)$ and $\log(c)$ vs. $\log(E_{cm})$ and added trend lines in the graphs (Figures 4.20 and 4.21).

Table 4.3: Trendline equations and values of C and c from Figures 4.15 to 4.21.

E_{cm} (in eV)	Trendline Equation	C	c
1.00	$y = -1.05 \times 10^3 x + 8.90$	$e^{8.90}$	1.04×10^3
3.16	$y = -2.86 \times 10^3 x + 9.81$	$e^{9.81}$	2.86×10^3
10.0	$y = -8.08 \times 10^3 x + 10.7$	$e^{10.7}$	8.08×10^3
31.6	$y = -2.19 \times 10^4 x + 11.6$	$e^{11.6}$	2.19×10^4
100	$y = -6.06 \times 10^4 x + 12.4$	$e^{12.4}$	6.06×10^4

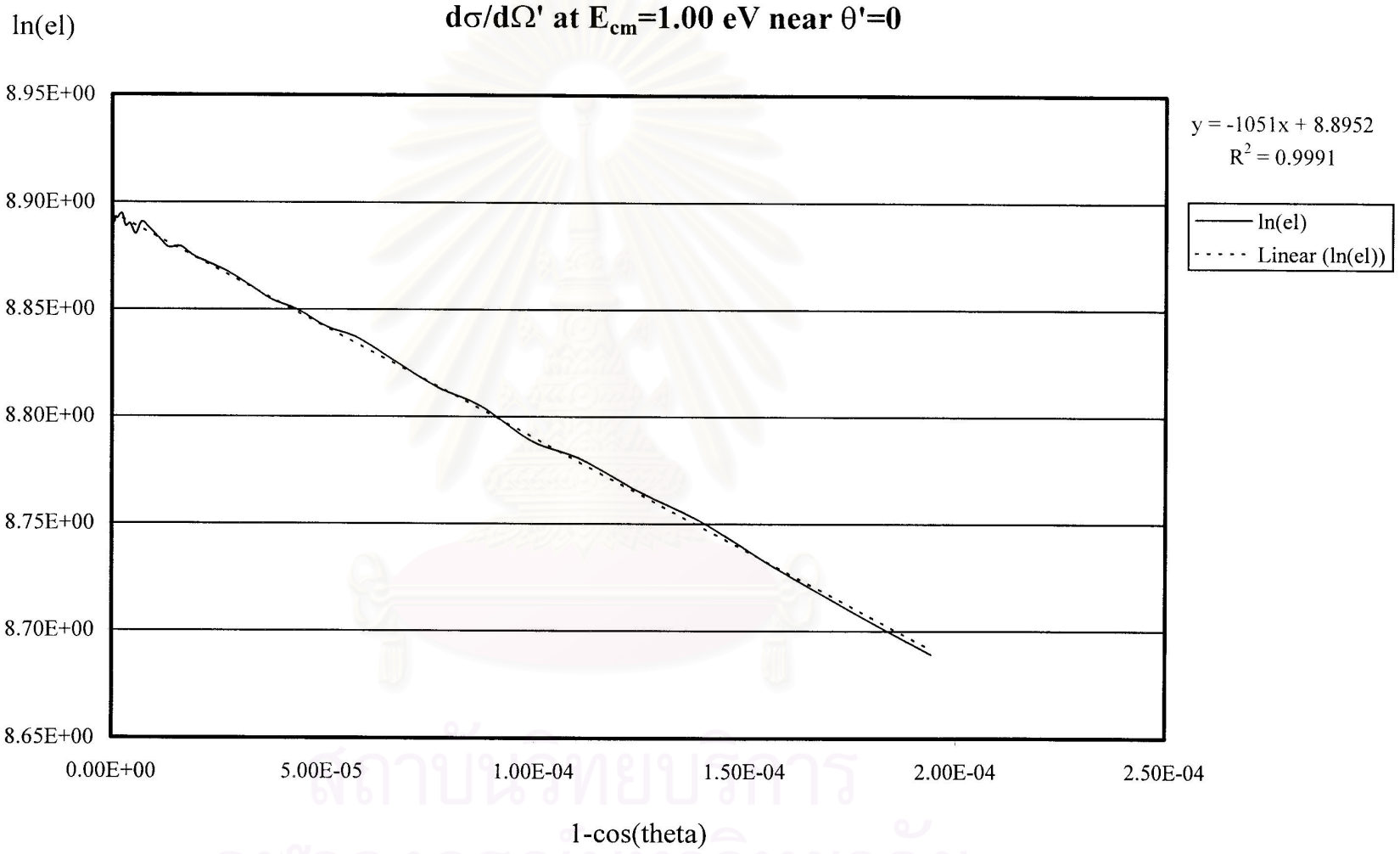

Figure 4.15: Elastic cross section (el, in a.u.) vs. CM scattering angle (theta) at 1.00 eV

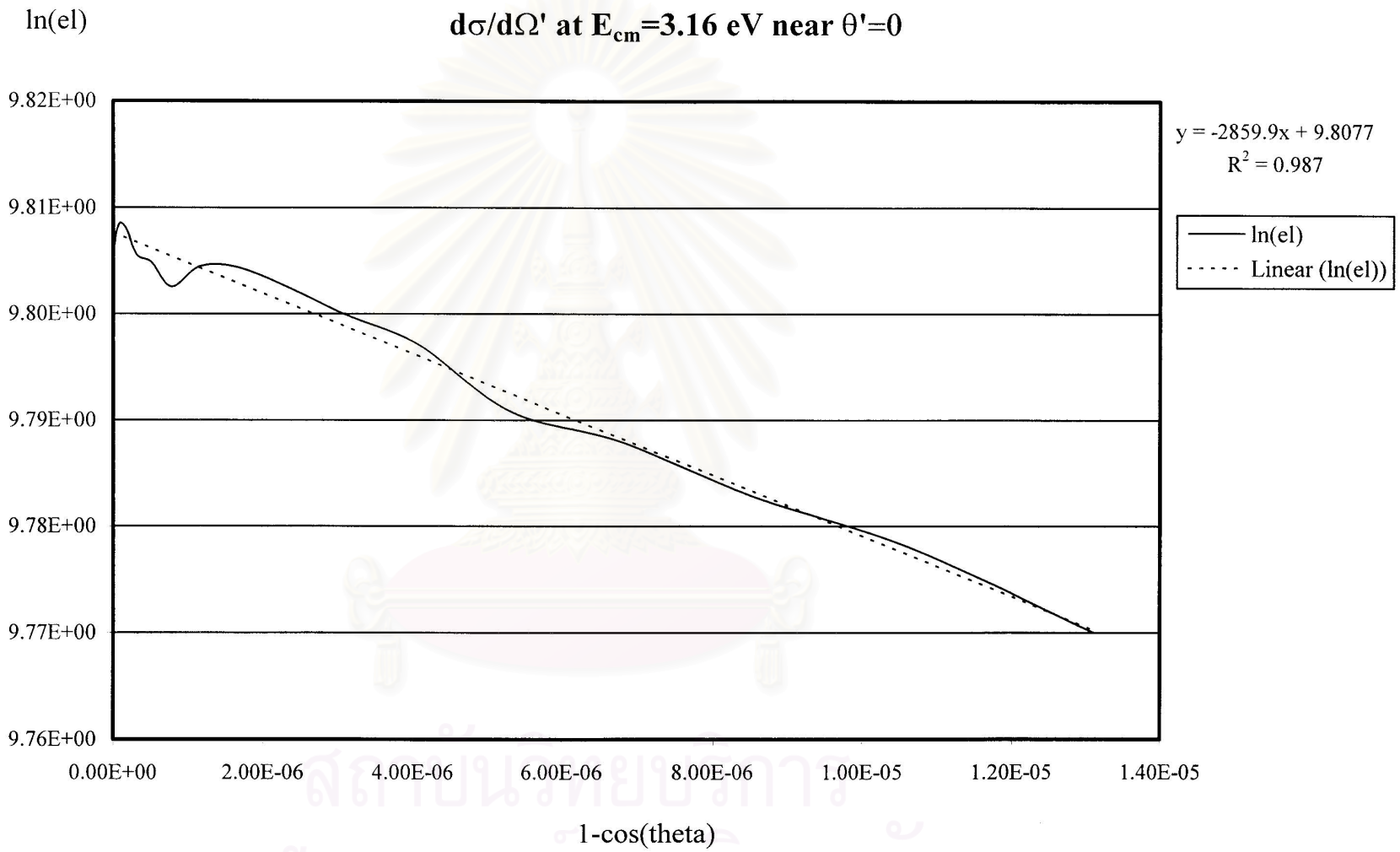

Figure 4.16: Elastic cross section (el, in a.u.) vs. CM scattering angle (theta) at 3.16 eV

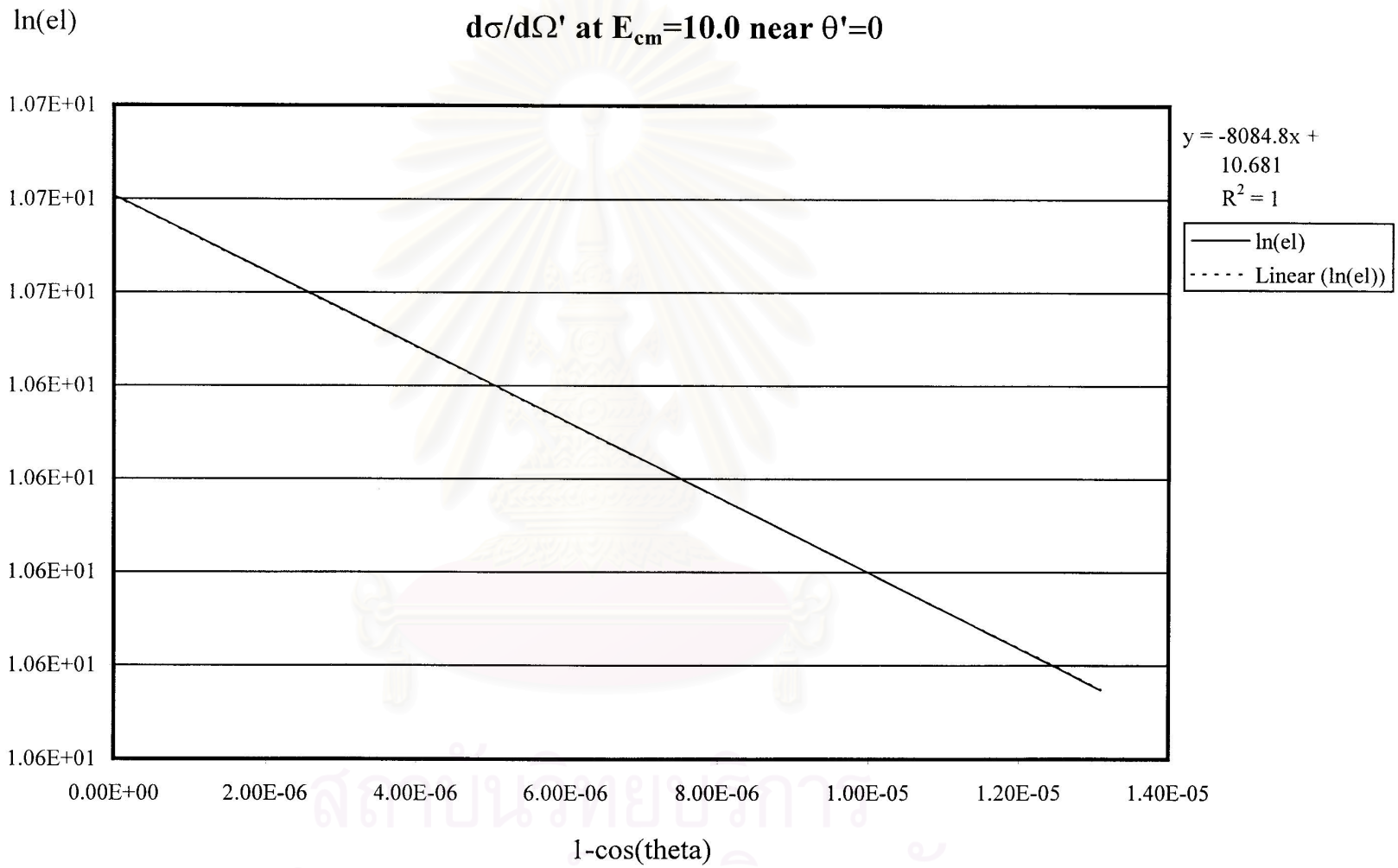

Figure 4.17: Elastic cross section (el, in a.u.) vs. CM scattering angle (theta) at 10.0 eV

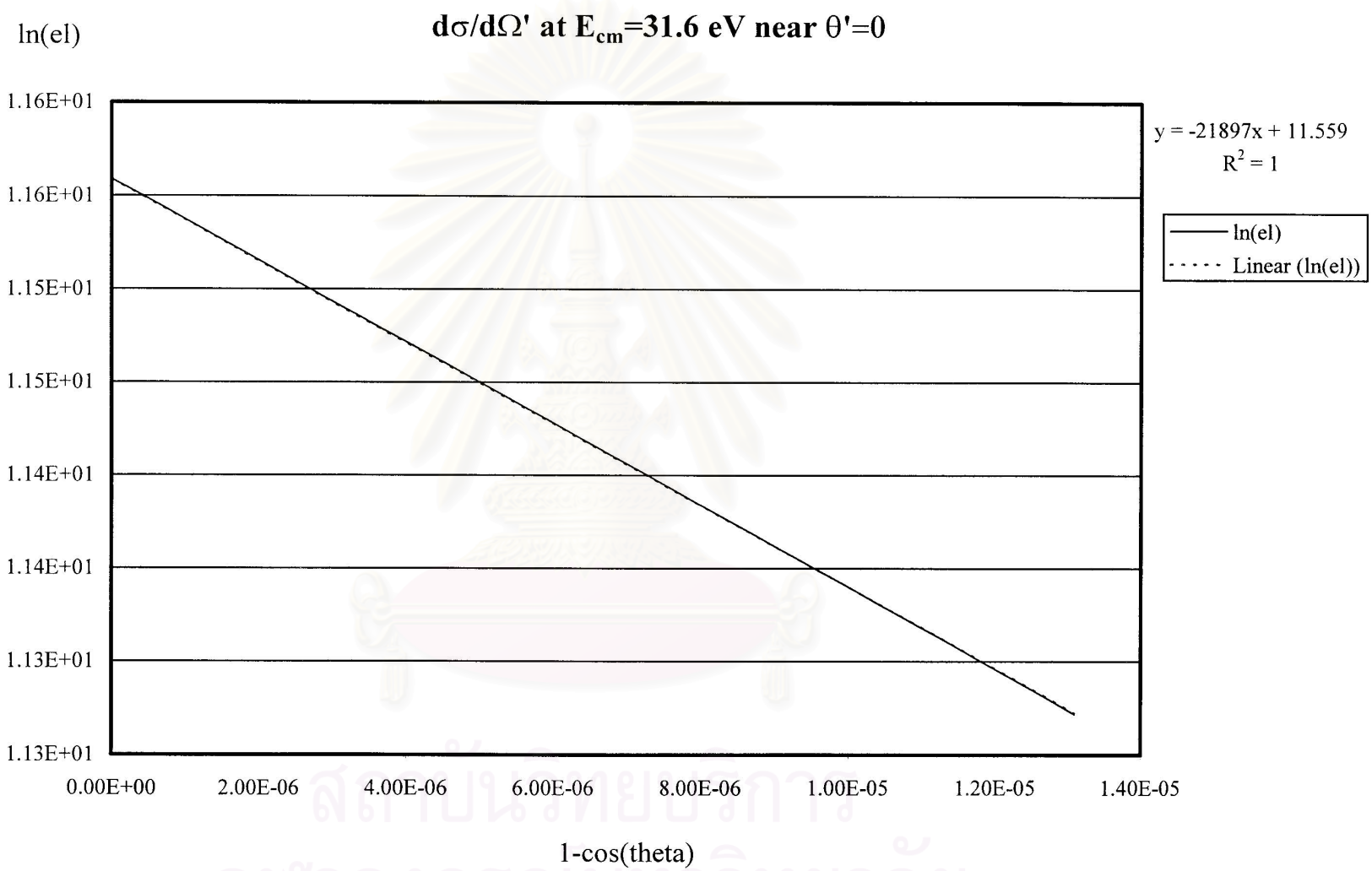

Figure 4.18: Elastic cross section (el, in a.u.) vs. CM scattering angle (theta) at 31.6 eV

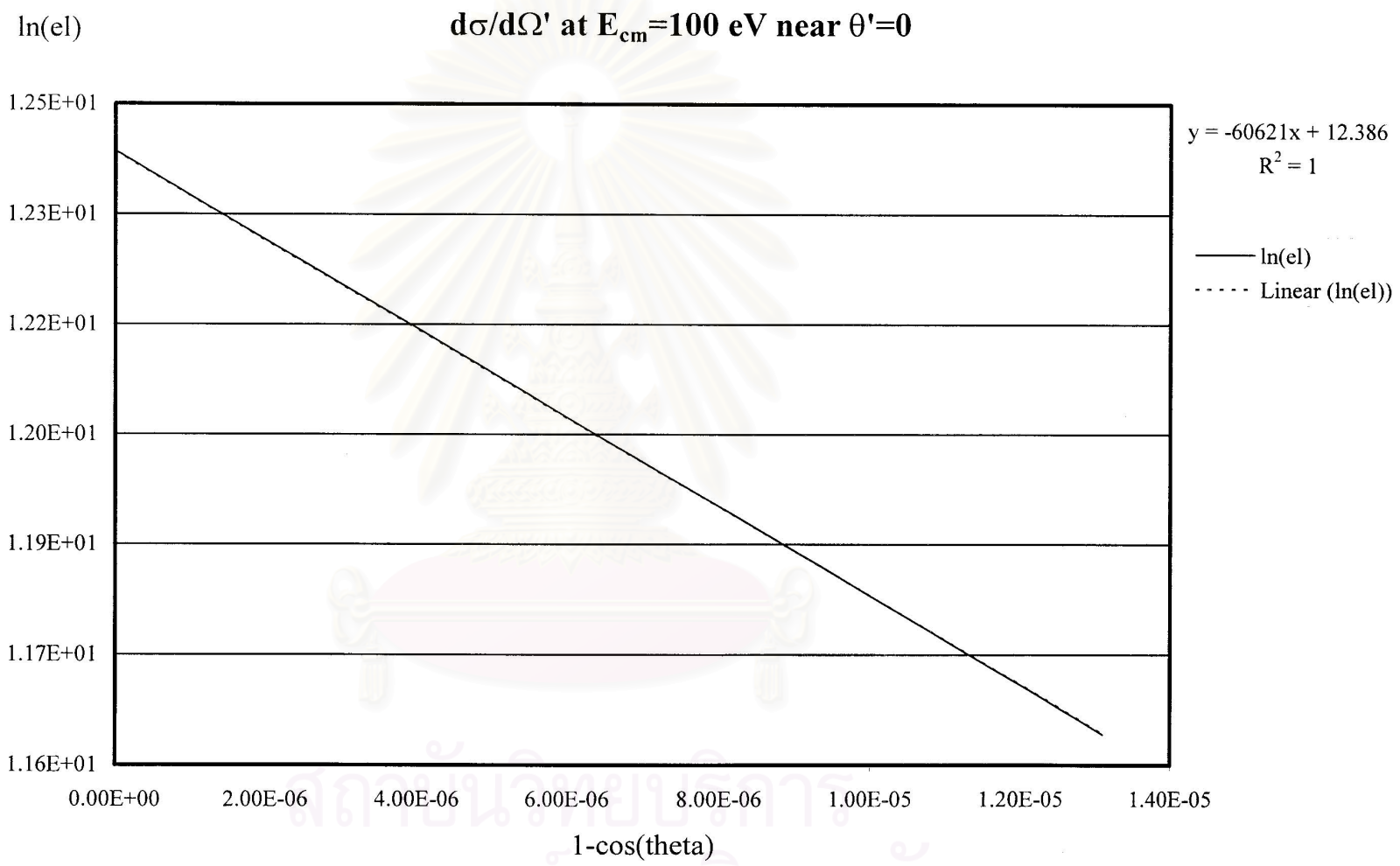

Figure 4.19: Elastic cross section (el, in a.u.) vs. CM scattering angle (theta) at 100 eV

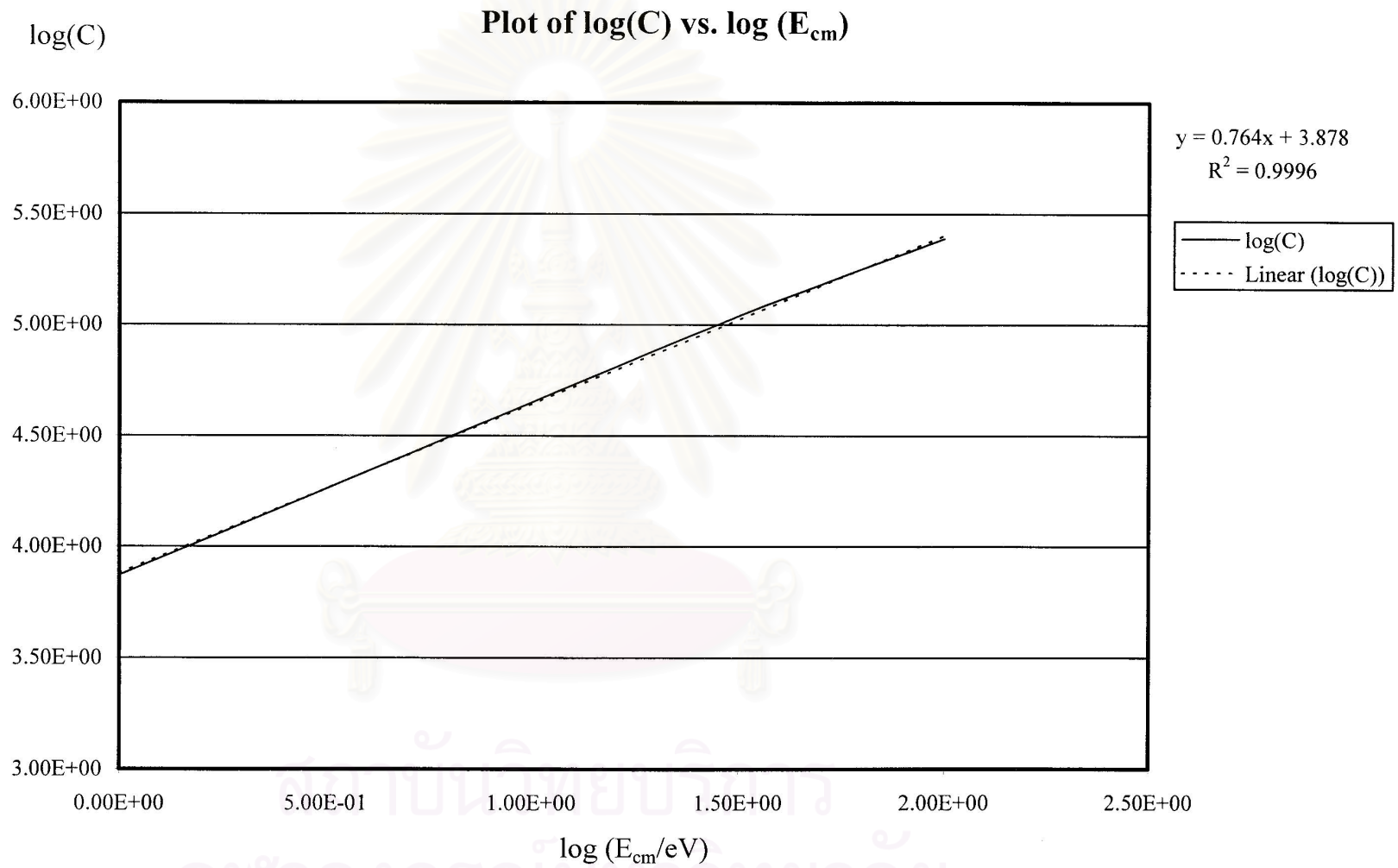

Figure 4.20: Plot of $\log(C)$ (in a.u.) vs. $\log(E_{cm})$ (in eV)

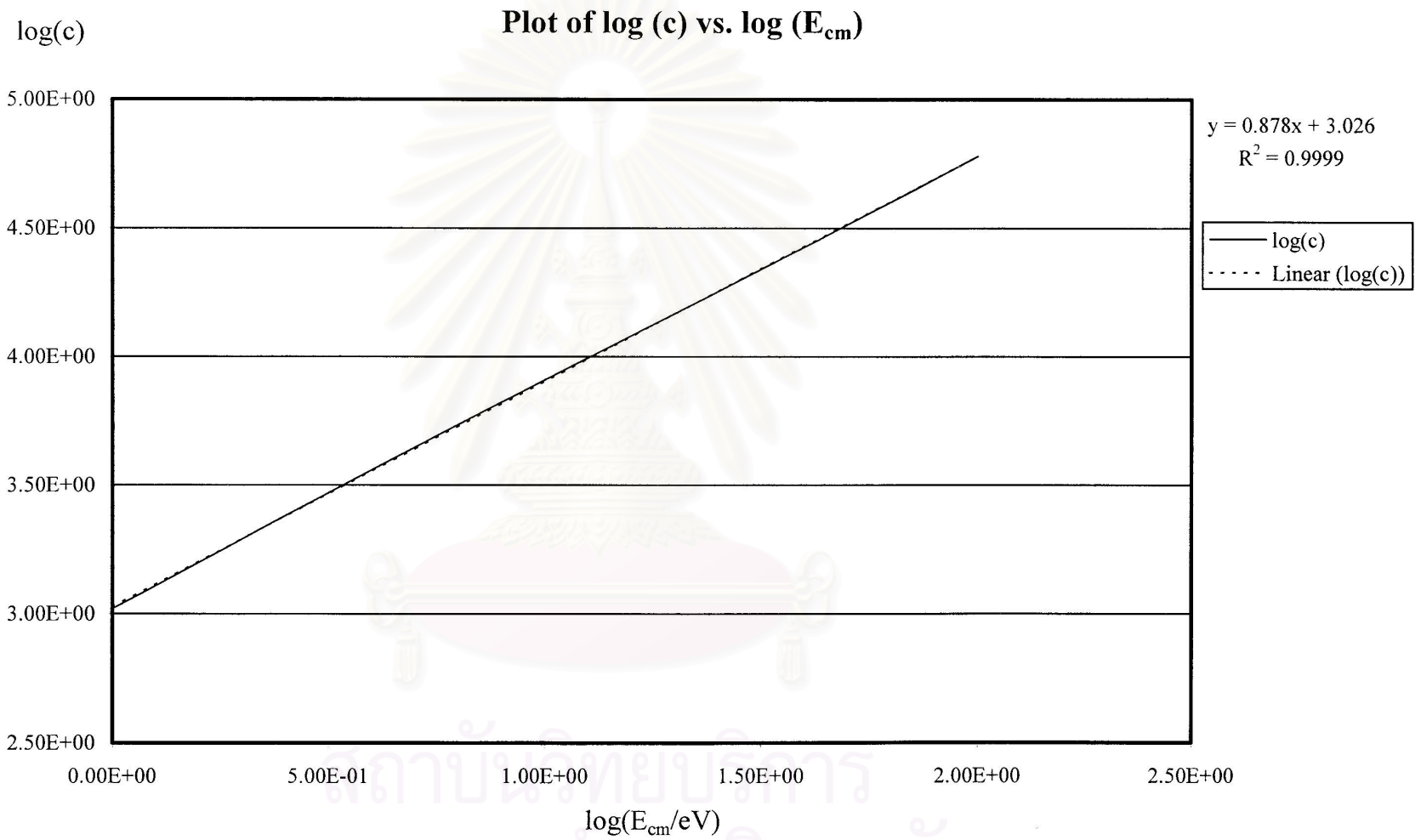

Figure 4.21: Plot of $\log(c)$ (in a.u. vs. $\log(E_{cm})$ (in eV.

4.3 Gauss-Laguerre Integration

The problem is to find the numerical value of a definite integral such as

$$I = \int_a^b f(z)w(z)dz. \tag{4.30}$$

We approximate integral by a finite sum

$$\int_a^b f(z)w(z)dz \approx \sum_{i=1}^{n} A_i f(z_i) \tag{4.31}$$

The parameters of Equation (4.31) are n values of z_i, for evaluating $f(z)$, n values of A_i coefficients, and n.

We replace $f(x)$ by an interpolating polynomial $P(z)$ of degree $n-1$ and a remainder term:

$$f(z) = P(z) + r(z), \tag{4.32}$$

where $P(z)$ is fitted to $f(z)$ at the n values of $z_i[P(z_i) - f(z_i)]$ by the choice

$$P(z) = \sum_{i=1}^{n} \frac{\alpha(z)}{(z - z_k)\alpha'(z_i)} \quad f(z_i) \ , \tag{4.33}$$

where $\alpha(z)$ is a completely factored nth-degree polynomial,

$$\alpha(z) = (z - z_1)(z - z_2)...(z - z_n). \tag{4.34}$$

Note that

$$\lim_{z \to z_i} \frac{\alpha(z)}{(z - z_i)\alpha'(z_k)} = 1. \tag{4.35}$$

For $f(x)$ a polynomial of degree $n-1$ the remainder term $r(z)$ is zero and Equation (4.33) becomes an identity. Using Equation (4.35), $P(z_i) = f(z_i)$, and the $(n-1)$ -degree polynomial is fitted to $f(z)$ at each z_i.

When the integral of the reminder term is small,

$$\begin{aligned} \int_a^b f(z)w(z)dz &\approx \int_a^b P(z)w(z)dz \\ &= \int_a^b \sum_{i=1}^{n} f(z_i)\frac{\alpha(z)}{(z - z_i)\alpha'(z_i)}w(z)dz. \end{aligned} \tag{4.36}$$

We obtain

$$\begin{aligned}\int_a^b f(z)w(z)dz &\approx \sum_{i=1}^{n} f(z_i)\int_a^b \frac{\alpha(z)}{(z-z_i)\alpha'(z_i)}w(z)dz \\ &= \sum_{i=1}^{n} A_i f(z_i). \end{aligned} \tag{4.37}$$

In this thesis we substituted $w(z)$ by e^{-z} and A_i by w_i, and we chose to use $n = 3$, so that

$$\int_0^\infty e^{-z}f(z)dz \approx \sum_{i=1}^{n} w_i f(z_i),$$

where w_i are weight values.

Table 4.4: Weights (w_i) of z_i for Gauss-Laguerre integration with $n = 3$ (H. E. Salzer and R. Zucker, 1949).

i	z_i	w_i
1	0.415774567830	0.711093009929
2	2.294280360279	0.278517733569
3	6.289945082937	1.038925650000

4.3.1 H-p elastic collisions including charge exchange

• **H-p elastic collisions** ($\theta' \approx 0$)

Consider (4.12):

$$\int_0^\infty Ae^{-ax} \sum_{\varphi' \ grid} [f_H(\vec{p}')f_p(\vec{p}'_1) - f_H(\vec{p})f_p(\vec{p}_1)]\Delta\varphi' dx. \tag{4.38}$$

Let $z = ax$ and $x = \dfrac{z}{a}$, so that (4.38) becomes

$$\begin{aligned}&\int_0^\infty Ae^{-z} \sum_{\varphi' \ grid} [f_H(\vec{p}')f_p(\vec{p}'_1) - f_H(\vec{p})f_p(\vec{p}_1)]\Delta\varphi' d\left(\frac{z}{a}\right) \\ &= \frac{A}{a}\int_0^\infty e^{-z} \sum_{\varphi' \ grid} [f_H(\vec{p}')f_p(\vec{p}'_1) - f_H(\vec{p})f_p(\vec{p}_1)]\Delta\varphi' dz. \end{aligned} \tag{4.39}$$

We substitute

$$\sum_{\varphi' \ grid} [f_H(\vec{p}')f_p(\vec{p}'_1) - f_H(\vec{p})f_p(\vec{p}_1)]\Delta\varphi' \quad \text{by} \quad S_1(z), \tag{4.40}$$

so that (4.39) becomes

$$\frac{A}{a}\int_0^\infty e^{-z}S_1(z)dz. \tag{4.41}$$

From $x = 1 - \cos\theta'$ and $z = ax$ we have

$$\begin{aligned} \cos\theta' &= 1 - \frac{z}{a} \\ \theta' &= \cos^{-1}\left(1 - \frac{z}{a}\right). \end{aligned} \tag{4.42}$$

We therefore approximate (4.41) by Gauss-Laguerre Integration with $n = 3$, so that

$$\begin{aligned} \frac{A}{a}\int_0^\infty e^{-z}S_1(z)dz &\approx \frac{A}{a}\sum_{i=1}^{3} w_i S_1(z_i) \\ &\approx \frac{A}{a}\sum_{i=1}^{3} w_i S_1\left[\theta'_i = \cos^{-1}\left(1 - \frac{z_i}{a}\right)\right], \end{aligned}$$

using w_i and z_i values from Table 4.4, and recalling that

$$S_1(\theta') = \sum_{\varphi' \ grid} [f_H(\vec{p}')f_p(\vec{p}'_1) - f_H(\vec{p})f_p(\vec{p}_1)]\,\Delta\varphi', \tag{4.43}$$

where

$f_H(\vec{p}')$ is the distribution function of hydrogen after the collision,

$f_p(\vec{p}'_1)$ is the distribution function of protons after the collision,

$f_H(\vec{p})$ is the distribution function of hydrogen before the collision, and

$f_p(\vec{p}_1)$ is the distribution function of protons before the collision.

• **H-p charge exchange** ($\theta' \approx \pi$)

Consider (4.21):

$$\int_0^\infty B e^{-bx} \sum_{\varphi' \ grid} [f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1)]\Delta\varphi' dx. \tag{4.44}$$

As for H-p elastic collisions ($\theta' \approx 0$), (4.44) becomes

$$\frac{B}{b}\int_0^\infty e^{-z} S_1(z) dz, \tag{4.45}$$

but in this case $x = 1 + \cos\theta'$ and $z = bx$, so we have

$$\begin{aligned} \cos\theta' &= \frac{z}{b} - 1 \\ \theta' &= \cos^{-1}\left(\frac{z}{b} - 1\right). \end{aligned} \tag{4.46}$$

By Gauss-Laguerre Integration with $n = 3$,

$$\begin{aligned} \frac{B}{b}\int_0^\infty e^{-z} S_1(z) dz &\approx \frac{B}{b}\sum_{i=1}^{3} w_i S_1(z_i) \\ &\approx \frac{B}{b}\sum_{i=1}^{3} w_i S_1\left[\theta'_i = \cos^{-1}\left(\frac{z_i}{b} - 1\right)\right]. \end{aligned}$$

Recall that

$$S_1(\theta') = \sum_{\varphi' \ grid} [f_H(\vec{p}\,')f_p(\vec{p}\,'_1) - f_H(\vec{p})f_p(\vec{p}_1)]\,\Delta\varphi'. \tag{4.47}$$

For f_p, we assume a Maxwellian distribution (chapter 2), with parameters from simulation results kindly provided by G. Zank (private communication, 1997). The f_H value is derived by interpolating (when necessary) from values at our grid points. (Recall that f_H is the dependent variable of this work.) Therefore, we need formulae for the magnitudes and polar angles (θ-coordinates) of $\vec{p}\,'$ and $\vec{p}\,'_1$. We want to derive formulae for

1. $\vec{p}\,' \cdot \vec{p}\,'$ for the magnitude of $\vec{p}\,'$
2. $\vec{p}\,' \cdot \hat{r}$ for the polar angle of $\vec{p}\,'$

3. $\vec{p}'_1 \cdot \vec{p}'_1$ for magnitude of $\vec{p}'_1$

4. $\vec{p}'_1 \cdot \hat{r}$ for the polar angle of $\vec{p}'_1$.

We define $\vec{P}$ as the vector from the center-of-mass momentum to $\vec{p}$ (see Figure 4.22),

$$\vec{P} \equiv \vec{p} - \frac{\vec{p} + \vec{p_1}}{2} = \frac{\vec{p} - \vec{p}_1}{2}, \tag{4.48}$$

and we define $\vec{P}_1$ as the analogous vector for $\vec{p_1}$,

$$\begin{aligned} \vec{P_1} &= \vec{p_1} - \frac{\vec{p} + \vec{p_1}}{2} = \frac{\vec{p_1} - \vec{p}}{2} \\ &= -\vec{P}. \end{aligned}$$

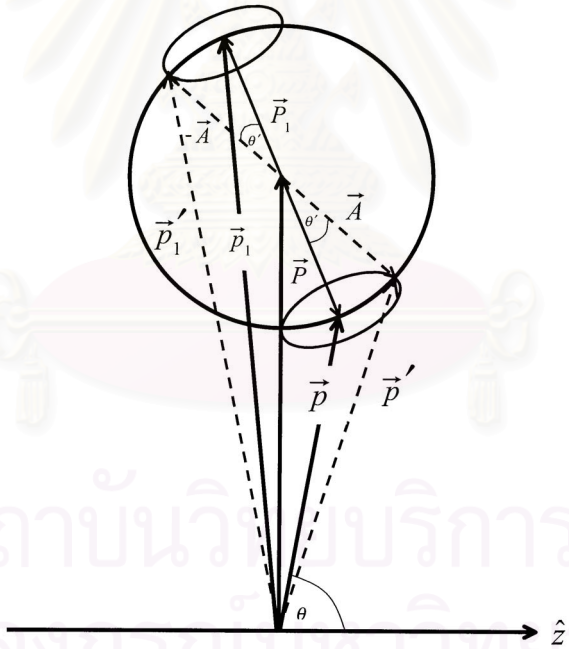

Figure 4.22: Momentum space variables before and after an elastic collision.

Formula for $\vec{p}' \cdot \vec{p}'$

We define $\vec{A} \equiv \vec{p}' - \dfrac{\vec{p}+\vec{p}_1}{2}$, so

$$\vec{p}' \cdot \vec{p}' = \left(\frac{\vec{p}+\vec{p}_1}{2}\right) \cdot \left(\frac{\vec{p}+\vec{p}_1}{2}\right) + 2\left(\frac{\vec{p}+\vec{p}_1}{2}\right) \cdot \vec{A} + \vec{A} \cdot \vec{A}. \tag{4.49}$$

From $\vec{P} = \vec{p} - (\vec{p}+\vec{p}_1)/2$, we substitute $(\vec{p}+\vec{p}_1)/2 = \vec{p} - \vec{P}$ to obtain

$$\begin{aligned} \vec{p}' \cdot \vec{p}' &= (\vec{p}-\vec{P}) \cdot (\vec{p}-\vec{P}) + 2(\vec{p}-\vec{P}) \cdot \vec{A} + \vec{A} \cdot \vec{A} \\ &= p^2 - 2\vec{p} \cdot \vec{P} + P^2 + 2(\vec{p}-\vec{P}) \cdot \vec{A} + \vec{P} \cdot \vec{P} \\ &= p^2 - 2\vec{p} \cdot \vec{P} + 2P^2 + 2\vec{A} \cdot \vec{p} - 2\cos\theta' P^2, \end{aligned} \tag{4.50}$$

where we note that θ' is the angle between the vectors $\vec{A}$ and $\vec{P}$, and $|\vec{A}| = |\vec{P}|$, so $\vec{A} \cdot \vec{P} = P^2 \cos\theta'$. We can derive $\vec{p} \cdot \vec{A}$ for Equation (4.50) as follows:

$$\begin{aligned} \vec{A} &= A_z \hat{z} + A_x \hat{x} \\ \vec{p} &= p_z \hat{z} + p_x \hat{x} \\ \vec{A} \cdot \vec{p} &= A_x p_x + A_y p_y + A_z p_z. \end{aligned}$$

In Cartesian coordinates with the $\hat{z}$ axis along $\vec{P}$, and the $\hat{x}$ axis along $\vec{p}_\perp$, we have

$$\begin{aligned} \vec{A} &= A(\sin\theta'\cos\varphi', \sin\theta'\sin\varphi', \cos\theta') \\ \vec{p} &= p_x \hat{x} + p_z \hat{z}, \end{aligned}$$

and we arbitrarily define φ' as 0 when $\vec{A}_\perp \parallel \vec{p}_\perp$, so that

$$\vec{A} \cdot \vec{p} = A p_x \sin\theta' \cos\varphi' + A p_z \cos\theta'. \tag{4.51}$$

Let us consider $A p_x \sin\theta' \cos\varphi'$:

$$p_x^2 = \vec{p}_\perp \cdot \vec{p}_\perp$$

$$
\begin{aligned}
&= \left[\vec{p} - \left(\frac{\vec{p}\cdot\vec{P}}{P^2}\right)\vec{P}\right]\cdot\left[\vec{p} - \left(\frac{\vec{p}\cdot\vec{P}}{P^2}\right)\vec{P}\right] \\
&= p^2 - 2\left(\frac{\vec{p}\cdot\vec{P}}{P^2}\right)\vec{p}\cdot\vec{P} + \left(\frac{\vec{p}\cdot\vec{P}}{P^2}\right)^2 P^2 \\
&= p^2 - 2\left(\frac{\vec{p}\cdot\vec{P}}{P}\right)^2 + \left(\frac{\vec{p}\cdot\vec{P}}{P}\right)^2 \\
&= p^2 - \frac{(\vec{p}\cdot\vec{P})^2}{P^2} \\
&= \frac{p^2P^2 - (\vec{p}\cdot\vec{P})^2}{P^2} \\
p_x &= \sqrt{\frac{p^2P^2 - (\vec{p}\cdot\vec{P})^2}{P^2}}. \qquad (4.52)
\end{aligned}
$$

Thus we substituted $p_x = \sqrt{p^2P^2 - (\vec{p}\cdot\vec{P})^2}/P$, $p_z = \vec{p}\cdot\vec{P}/P$, and $A = P$ in Equation (4.51) and Equation (4.50) to obtain

$$
\begin{aligned}
\vec{p}'\cdot\vec{p}' &= p^2 - 2\vec{p}\cdot\vec{P} + 2P^2 + 2\vec{p}\cdot\vec{P}\cos\theta' + 2\sqrt{p^2P^2 - (\vec{p}\cdot\vec{P})^2}\sin\theta'\cos\varphi' \\
&\quad -2\cos\theta' P^2 \\
&= p^2 - 2\vec{p}\cdot\vec{P}(1-\cos\theta') + 2P^2(1-\cos\theta') \\
&\quad +2\sqrt{p^2P^2 - (\vec{p}\cdot\vec{P})^2}\sin\theta'\cos\varphi'. \qquad (4.53)
\end{aligned}
$$

Note that

$$
\begin{aligned}
\vec{p}\cdot\vec{P} &= \vec{p}\cdot\frac{\vec{p} - \vec{p_1}}{2} \\
&= \frac{1}{2}(\vec{p}\cdot\vec{p} - \vec{p}\cdot\vec{p_1}),
\end{aligned}
$$

and in Cartesian coordinates, this time with the $\hat{z}$ axis along the spatial z-axis,

$$
\begin{aligned}
\vec{p} &= (p\sin\theta\cos\varphi, p\sin\theta\sin\varphi, p\cos\theta) \\
\vec{p_1} &= (p_1\sin\theta_1\cos\varphi_1, p_1\sin\theta_1\sin\varphi_1, p_1\cos\theta_1) \\
\vec{p}\cdot\vec{p_1} &= pp_1(\sin\theta\cos\varphi_1\sin\theta_1) + pp_1\cos\theta\cos\theta_1
\end{aligned}
$$

$$= \quad pp_1(\sin\theta\sin\theta_1\cos\varphi_1 + \cos\theta\cos\theta_1),$$

so

$$\vec{p}\cdot\vec{P} = \frac{1}{2}\left[p^2 - pp_1(\sin\theta\sin\theta_1\cos\varphi_1 + \cos\theta\cos\theta_1)\right]. \tag{4.54}$$

Similarly, P^2 can be written

$$\begin{aligned} P^2 &= \left(\frac{\vec{p}-\vec{p_1}}{2}\right)\cdot\left(\frac{\vec{p}-\vec{p_1}}{2}\right) \\ &= \frac{1}{4}(p^2 - 2p\cdot p_1 + p_1^2) \\ &= \frac{1}{4}\{p^2 - 2\left[pp_1(\sin\theta\sin\theta_1\cos\varphi + \cos\theta\cos\theta_1)\right] + p_1^2\}. \end{aligned}$$

Formula for $\vec{p}'\cdot\hat{r}$

Next, we need to find

$$\begin{aligned} \vec{p}'\cdot\hat{r} &= \left(\frac{\vec{p}+\vec{p_1}}{2}\right)\cdot\hat{r} + \vec{A}\cdot\hat{r} \\ &= \frac{1}{2}(p\cos\theta + p_1\cos\theta_1) + \vec{A}\cdot\hat{r}. \end{aligned} \tag{4.55}$$

We can derive a formula for $\vec{A}\cdot\hat{r}$ that is similar to Equation (4.51):

$$\begin{aligned} \vec{A}\cdot\hat{r} &= Ar_x\sin\theta'\cos\varphi'' + Ar_z\cos\theta' \\ &= Ar_x\sin\theta'\cos\varphi'' + (\hat{r}\cdot\vec{P})\cos\theta', \end{aligned} \tag{4.56}$$

where this time the azimuthal angle φ'' is defined to be 0 when $\vec{A}_\perp \parallel \vec{r}_\perp$. A formula for r_x, similar to that for p_x in Equation(4.52) and using $|\hat{r}|$=1, is

$$r_x = \frac{1}{P}\sqrt{P^2 - (\vec{r}\cdot\vec{P})^2}, \tag{4.57}$$

so that Equation (4.56) becomes

$$\vec{A}\cdot\vec{r} = (\vec{P}\cdot\hat{r})\cos\theta' + \sqrt{P^2 - (\vec{P}\cdot\hat{r})^2}\sin\theta'\cos\varphi'', \tag{4.58}$$

and Equation (4.55) becomes

$$\vec{p}' \cdot \hat{r} = \frac{1}{2}(p\cos\theta + p_1\cos\theta_1) + (\vec{P}\cdot\hat{r})\cos\theta' + \sqrt{P^2 - (\vec{P}\cdot\vec{r})^2}\sin\theta'\cos\varphi''. \quad (4.59)$$

Note that $\varphi'' = 0$ when $\vec{A}_\perp \parallel \hat{r}_\perp$, but $\varphi' = 0$ when $\vec{A}_\perp \parallel \vec{p}_\perp$, so φ'' is related to φ' by $\varphi'' = \varphi' + \varphi_{pr}$, where φ_{pr} is the azimuthal angle of $\vec{p}_\perp$ relative to $\hat{r}_\perp$. Thus

$$\begin{aligned}
\cos\varphi'' &= \cos(\varphi' + \varphi_{pr}) \\
&= \cos\varphi'\cos\varphi_{pr} - \sin\varphi'\sin\varphi_{pr} \\
&= \cos\varphi'\cos\varphi_{pr} - \sin\varphi'\sqrt{1-\cos^2\varphi_{pr}}. \qquad (4.60)
\end{aligned}$$

We defined

$$\begin{aligned}
p_\perp &= \sqrt{\frac{p^2P^2 - (\vec{p}\cdot\vec{P})^2}{P}} \\
r_\perp &= \sqrt{\frac{P^2 - (\vec{p}\cdot\vec{P})^2}{P}}
\end{aligned}$$

where $p_\perp$ is the magnitude of $|\vec{p}_\perp|$ and $\vec{p}_\perp$ is the projection of $\vec{p}$ onto a plane perpendicular to $\vec{P}$, and $r_\perp$ is defined similarly. Now consider

$$\begin{aligned}
\vec{p}\cdot\vec{r} &= p_z r_z + \vec{p}_\perp\cdot\vec{r}_\perp \\
&= p_z r_z + p_\perp r_\perp\cos\varphi_{pr} \\
&= \left(\frac{\vec{p}\cdot\vec{P}}{P}\right)\left(\frac{\hat{r}\cdot\vec{P}}{P}\right) \\
&\quad + \left(\frac{\sqrt{p^2P^2 - (\vec{p}\cdot\vec{P})^2}}{P}\right)\left(\frac{\sqrt{P^2 - (\vec{P}\cdot\hat{r})^2}}{P}\right)\cos\varphi_{pr} \\
\cos\varphi_{pr} &= \frac{\vec{p}\cdot\vec{r} - \dfrac{(\vec{p}\cdot\vec{P})(\hat{r}\cdot\vec{P})}{P^2}}{\sqrt{p^2P^2 - (\vec{p}\cdot\vec{P})^2}\sqrt{P^2 - (\vec{P}\cdot\hat{r})^2}/P^2} \\
&= \frac{P^2(\vec{p}\cdot\vec{r}) - (\vec{p}\cdot\vec{P})(\vec{P}\cdot\hat{r})}{\sqrt{p^2P^2 - (\vec{p}\cdot\vec{P})^2}\sqrt{P^2 - (\vec{P}\cdot\hat{r})^2}} \qquad (4.61)
\end{aligned}$$

Equation (4.61) cannot be evaluated if the denominator on the right hand side is zero, so we need to consider some special cases. If $\sqrt{p^2P^2-(\vec{p}\cdot\vec{P})^2}=0$, that is $p_\perp=0$, then $\vec{p}\parallel\vec{P}$. If $\sqrt{P^2-(\vec{P}\cdot\hat{r})^2}=0$, that is $r_\perp=0$, then $\hat{r}\parallel\vec{P}$. Thus we need to consider $\vec{p}'\cdot\hat{r}$ specially when $\vec{p}\parallel\vec{P}$ or $\hat{r}\parallel\vec{P}$:

- If $\sqrt{p^2P^2-(\vec{p}\cdot\vec{P})^2}=0$ and $\sqrt{P^2-(\vec{P}\cdot\hat{r})^2}=0$, the formula for this case is
$$\vec{p}'\cdot\hat{r}=\frac{1}{2}(p\cos\theta+p_1\cos\theta_1)+(\vec{P}\cdot\hat{r})\cos\theta'. \tag{4.62}$$
- If $\sqrt{p^2P^2-(\vec{p}\cdot\vec{P})^2}=0$ and $\sqrt{P^2-(\vec{P}\cdot\hat{r})^2}\neq 0$, i.e., $\vec{p}\parallel\vec{P}$, the definition of φ' is arbitrary, so in this case we define $\varphi'=0(=\varphi'')$ when $\vec{A}_\perp\parallel\hat{r}_\perp$ and use $\varphi''=\varphi'$, so the formula for $\vec{p}'\cdot\hat{r}$ is
$$\vec{p}'\cdot\hat{r}=\frac{1}{2}(p\cos\theta+p_1\cos\theta_1)+(\vec{P}\cdot\hat{r})\cos\theta'+\sqrt{P^2-(\vec{P}\cdot\vec{r})^2}\sin\theta'\cos\varphi'. \tag{4.63}$$
- If $\sqrt{P^2-(\vec{P}\cdot\hat{r})^2}=0$ ($\hat{r}\parallel\vec{P}$), then we do not need φ'' or φ_{pr}, and the formula for $\vec{p}'\cdot\hat{r}$ is
$$\vec{p}'\cdot\hat{r}=\frac{1}{2}(p\cos\theta+p_1\cos\theta_1)+(\vec{P}\cdot\hat{r})\cos\theta'. \tag{4.64}$$
- If $\sqrt{p^2P^2-(\vec{p}\cdot\vec{P})^2}$ and $\sqrt{P^2-(\vec{P}\cdot\hat{r})^2}\neq 0$, then we need the formulae for $\cos\varphi''$ and $\cos\varphi_{pr}$, that is
$$\vec{p}'\cdot\hat{r}=\frac{1}{2}(p\cos\theta+p_1\cos\theta_1)+(\vec{P}\cdot\hat{r})\cos\theta'+\sqrt{P^2-(\vec{P}\cdot\vec{r})^2}\sin\theta'\cos\varphi'', \tag{4.65}$$

where

$$\begin{aligned}\cos\varphi'' &= \cos\varphi'\cos\varphi_{pr}-\sin\varphi'\sqrt{1-\cos^2\varphi_{pr}}\\ \cos\varphi_{pr} &= \frac{P^2(\vec{p}\cdot\vec{r})-(\vec{p}\cdot\vec{P})(\vec{P}\cdot\hat{r})}{\sqrt{p^2P^2-(\vec{p}\cdot\vec{P})^2}\sqrt{P^2-(\vec{P}\cdot\vec{r})^2}}.\end{aligned}$$

Formula for $\vec{p}'_1 \cdot \vec{p}'_1$

We define $\vec{B}$ as a vector from the center-of-mass momentum $\vec{p}'_1$, and $\vec{p}'_1 = \left(\frac{\vec{p}+\vec{p_1}}{2} + \vec{B}\right)$, and then

$$\vec{p}'_1 \cdot \vec{p}'_1 = \left(\frac{\vec{p}+\vec{p}_1}{2}\right) \cdot \left(\frac{\vec{p}+\vec{p}_1}{2}\right) + 2\left(\frac{\vec{p}+\vec{p}_1}{2}\right) \cdot \vec{B} + \vec{B} \cdot \vec{B}. \tag{4.66}$$

Using $(\vec{p}+\vec{p}_1)/2 = \vec{p}_1 - \vec{P}_1$, we have

$$\begin{aligned} \vec{p}'_1 \cdot \vec{p}'_1 &= (\vec{p}_1 - \vec{P}_1) \cdot (\vec{p}_1 - \vec{P}_1) + 2(\vec{p}_1 - \vec{P}_1) \cdot \vec{B} + \vec{B} \cdot \vec{B} \quad (B^2 = {P_1}^2) \\ &= {p_1}^2 - 2\vec{p}_1 \cdot \vec{P}_1 + {P_1}^2 + 2\vec{p}_1 \cdot \vec{B} - 2\vec{P}_1 \cdot \vec{B} + P_1^2. \end{aligned}$$

We find a formula for $\vec{p}_1 \cdot \vec{B}$, analogous to that for $\vec{p} \cdot \vec{A}$, and we substitute $\vec{B} = -\vec{A}$ and $\vec{P}_1 = -\vec{P}$. Note that $\vec{B} = -\vec{A}$ implied that $\varphi' \to \varphi' + \pi$ in the formula analogous to Equation (4.51), effectively changing $\cos\varphi' \to -\cos\varphi'$. Thus the formula for $\vec{p}'_1 \cdot \vec{p}'_1$ is

$$\begin{aligned} \vec{p}'_1 \cdot \vec{p}'_1 &= {p_1}^2 - 2\vec{p}_1 \cdot \vec{P}_1 + 2{P_1}^2 + 2\vec{p}_1 \cdot \vec{B} - 2\cos\theta' {P_1}^2 \\ &= {p_1}^2 - 2\vec{p}_1 \cdot \vec{P}_1 + 2{P_1}^2 \\ &\quad +2\left(\vec{P}_1 \cdot \vec{p}_1 \cos\theta' - \sqrt{{p_1}^2 P^2 - (\vec{p}_1 \cdot \vec{P}_1)^2}\sin\theta'\cos\varphi'\right) \\ &\quad -2\cos\theta' {P_1}^2 \\ &= {p_1}^2 - 2\vec{p}_1 \cdot \vec{P}_1(1-\cos\theta') + 2{P_1}^2(1 - cos\theta') \\ &\quad -2\sqrt{{p_1}^2 P^2 - (\vec{p}_1 \cdot \vec{P}_1)^2}\sin\theta'\cos\varphi' \end{aligned} \tag{4.67}$$

Formula for $\vec{p}'_1 \cdot \hat{r}$

We derive a formula for $\vec{p}'_1 \cdot \hat{r}$ like that for $\vec{p}' \cdot \hat{r}$ (Equation 4.59):

$$\vec{p}'_1 \cdot \hat{r} = \frac{1}{2}(p\cos\theta + p_1\cos\theta_1) + (\vec{P}_1 \cdot \hat{r})\cos\theta' + \sqrt{{P_1}^2 - (\vec{P}_1 \cdot \hat{r})^2}\sin\theta'\cos\varphi''.$$

Note that φ'' is the angle that is 0 when $\vec{B}_\perp = -\vec{A}_\perp$ satisfies $\vec{B}_\perp \parallel \hat{r}_\perp$, so

$$\begin{aligned} \cos\varphi'' &= -\cos\varphi'\cos\varphi_{pr} + \sin\varphi'\sin\varphi_{pr} \\ &= -\cos\varphi'\cos\varphi_{pr} + \sin\varphi'\sqrt{1-\cos^2\varphi_{pr}} \end{aligned}$$

where

$$\cos\varphi_{pr} = \frac{P^2(\vec{p}\cdot\vec{r}) - (\vec{p}\cdot\vec{P})(\vec{P}\cdot\hat{r})}{\sqrt{p^2P^2 - (\vec{p}\cdot\vec{P})^2}\sqrt{P^2 - (\vec{P}\cdot\vec{r})^2}}. \tag{4.68}$$

We calculate $\cos\varphi_{pr}$ in $\vec{p}\,'_1\cdot\hat{r}$ as we did $\vec{p}\,'\cdot\hat{r}$, subject to analogous special cases.

Formula for d^3p_1

Next we consider term 3 or d^3p_1, which we call the "volume term." We approximate term 3 by summation over the $\vec{p}_1$ grid in 3D:

$$\begin{aligned} dV &= {p_1}^2 \sin\theta_1 dp_1 d\theta_1 d\varphi_1 \\ &= d\left(\frac{{p_1}^3}{3}\right) d\cos\theta_1 d\varphi_1 \\ \Delta V &= \Delta\left(\frac{{p_1}^3}{3}\right)\Delta\mu_1\Delta\varphi_1 \\ &= \left[\frac{p^3_{1,\ out}}{3} - \frac{p^3_{1,\ in}}{3}\right]\Delta\mu_1\Delta\varphi_1, \end{aligned} \tag{4.69}$$

where dV is the volume element in momentum space ($\vec{p}_1$), and ΔV is the volume corresponding to a given grid point.

4.3.2 H-H elastic collisions

Let $z = cy$ and $y = \dfrac{z}{c}$ so that Equation (4.29) becomes

$$\begin{aligned} &\int_0^\infty Ce^{-z} \sum_{\varphi'\ grid} [f_H(\vec{p}\,')f_H(\vec{p}\,'_1) - f_H(\vec{p})f_H(\vec{p}_1)]\Delta\varphi' d\left(\frac{z}{c}\right) \\ &= \frac{C}{c}\int_0^\infty e^{-z} \sum_{\varphi'\ grid} [f_H(\vec{p}\,')f_H(\vec{p}\,'_1) - f_H(\vec{p})f_H(\vec{p}_1)]\Delta\varphi' dz. \end{aligned} \tag{4.70}$$

We substitute

$$S_2(z) = \sum_{\varphi'\ grid} [f_H(\vec{p}\,')f_H(\vec{p}\,'_1) - f_H(\vec{p})f_H(\vec{p}_1)]\Delta\varphi', \tag{4.71}$$

so the right hand side of Equation (4.70) becomes

$$\frac{C}{c}\int_0^\infty e^{-z}S_2(z)dz. \tag{4.72}$$

From $y = 1-\cos\theta'$ and $z = cy$ we have

$$\begin{aligned}\cos\theta' &= 1-\frac{z}{c}\\ \theta' &= \cos^{-1}\left(1-\frac{z}{c}\right).\end{aligned} \tag{4.73}$$

We therefore approximate (4.72) by Gauss-Laguerre Integration with $n = 3$, so that

$$\begin{aligned}\frac{C}{c}\int_0^\infty e^{-z}S_2(z)dz &\approx \frac{C}{c}\sum_{i=1}^{3} w_i S_2(z_i)\\ &\approx \frac{C}{c}\sum_{i=1}^{3} w_i S_2\left[\theta_i' = \cos^{-1}\left(1-\frac{z_i}{c}\right)\right].\end{aligned}$$

We then use w_i and z_i values from Table 4.4. Recall that

$$S_2(\theta') = \sum_{\varphi'\ grid}\left[f_H(\vec{p}\,')f_H(\vec{p}\,'_1) - f_H(\vec{p})f_H(\vec{p}_1)\right]\Delta\varphi'. \tag{4.74}$$

Formula for d^3p_1

We use a volume term (term 3) similar to that for H-p elastic collisions including charge exchange. That is,

$$\Delta V = \left[\frac{p_{1,\ out}^3}{3} - \frac{p_{1,\ in}^3}{3}\right]\Delta\mu_1\Delta\varphi_1. \tag{4.75}$$

Finally, we approximate H-H elastic collisions by

$$\int_{\vec{p}_1}|\vec{v}-\vec{v}_1|\left[\frac{C}{c}\sum_{i=3}^{3} w_i S_2(\theta_i')\right]d^3p_1.$$

4.4 Interpolation

4.4.1 Linear interpolation

Linear interpolation aims to find a value of a function between two points with known function values in one dimension.

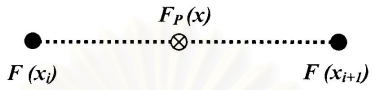

Figure 4.23: Linear interpolation.

We approximate $F_P(x)$ by

$$\begin{aligned} frac &= \frac{x_{i+1} - x}{x_{i+1} - x_i} \\ F_p(x) &= (1 - frac) \times F(x_i) + frac \times F(x_{i+1}). \end{aligned} \tag{4.76}$$

4.4.2 Bilinear interpolation

Bilinear interpolation aims to find a value of a function between four points with known function values in two dimensions.

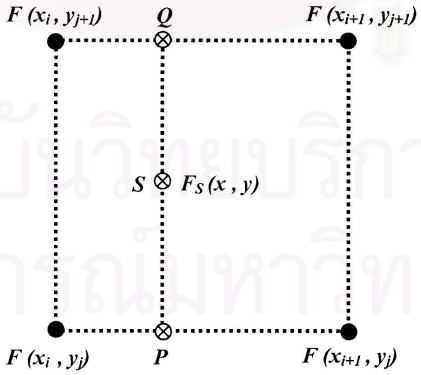

Figure 4.24: Bilinear interpolation.

We use four points, $F(x_i, y_j)$, $F(x_{i+1}, y_j)$, $F(x_i, y_{j+1})$, and $F(x_{i+1}, y_{j+1})$ in order to approximate at $F_s(x, y)$ by

- At point P:

$$\begin{aligned} frac &= \frac{x - x_i}{x_{i+1} - x_i} \\ F_P(x) &= (1 - frac) \times F(x_i, y_j) + frac \times F(x_{i+1}, y_j). \end{aligned} \tag{4.77}$$

- At point Q:

$$\begin{aligned} frac &= \frac{x - x_i}{x_{i+1} - x_i} \\ F_Q(x) &= (1 - frac) \times F(x_i, y_{j+1}) + frac \times F(x_{i+1}, y_{j+1}). \end{aligned} \tag{4.78}$$

- At point S:

$$\begin{aligned} frac &= \frac{y - y_j}{y_{j+1} - y_j} \\ F_S(x) &= (1 - frac) \times F_P(x, y_j) + frac \times F_Q(x, y_{j+1}). \end{aligned} \tag{4.79}$$

4.4.3 Geometric interpolation

In this thesis we have used geometric interpolation because a Maxwellian distribution has an exponential-based distribution, for which geometric interpolation is suitable. We approximate at $F_s(x, y)$ by

- At point P:

$$\begin{aligned} frac &= \frac{\log x - \log x_i}{\log x_{i+1} - \log x_i} \\ \log F_P(x) &= (1 - frac) \times \log F(x_i, y_j) \\ &\quad + frac \times \log F(x_{i+1}, y_j). \end{aligned} \tag{4.80}$$

- At point Q:

$$\begin{aligned} frac &= \frac{\log x - \log x_i}{\log x_{i+1} - \log x_i} \\ \log F_Q(x) &= (1 - frac) \times \log F(x_i, y_{j+1}) \\ &\quad + frac \times \log F(x_{i+1}, y_{j+1}). \end{aligned} \tag{4.81}$$

- At point S:

$$\begin{aligned} frac &= \frac{\log y - \log y_j}{\log y_{j+1} - \log y_j} \\ \log F_S(x) &= (1 - frac) \times \log F_P(x, y_j) \\ &\quad + frac \times \log F_Q(x, y_{j+1}). \end{aligned} \tag{4.82}$$

Chapter 5

Results

In this chapter we explain about how to use the program and describe functions that we use. After that we show results of the program by surface plots in Figures (5.1) - (5.3).

5.1 Testing the Program

We use the "boltz" program for these simulations. The boltz program consists of:

- boltz.c is a main program and it contains the main program and many functions such as the interpolation function and definitions of the fine grid and coarse grid.
- nrutil.c is written to allocate arrays for variables, and we developed this program from Press et al., 1988.
- printout.c is written to show the output of this work, and it displays the density of hydrogen (f_H) at each position in v_y-v_z space.
- initial.c is written to set the initial distribution function for hydrogen. We choose a Maxwellian distribution for this initial condition. In this formula

we set the density of particles at each z as one particle per cubic centimeter $[n(z)=1]$.

- elastic.c is written to calculate the change in the distribution function of hydrogen due to charge exchange and elastic collisions.

In this thesis we set input parameters by:

- Starting value of time (days): 0
- Final value of time (days): 1
- Time step (days): 1
- Time interval after which to print out data (days): 1
- Number of mu points (must be even): 10
- Length in the z-direction (AU): 100
- Step in the z-direction (AU): 1

5.2 Results of Computational Simulations

We plotted the distribution of hydrogen, f_H, in v_y-v_z space for 100 z-grid points and for z=10, 30 and 60 AU. The units of f_H are cm^{-3} (eV/c)$^{-3}$. Figures (5.1)-(5.3) show results from the program. We use surface plots to plot results from elastic collisions and charge exchange of hydrogen from the interstellar medium with protons from the solar wind. We found that the distribution of hydrogen (f_H) was substantial in two regions of velocity space, at low velocity (initial distribution) and at high velocity ($v_z \approx 400$ km/s), representing hydrogen atoms resulting from charge exchange with protons from the solar wind. We defined v_z as the velocity in the direction of flow from the Sun along the solar apex and v_y

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

Figure (5.1) is a surface plot of the distribution of hydrogen at z=10 AU. It shows that the low-velocity peak has a density of hydrogen higher than the high-velocity peak by $\sim 10^5$. Still, we found that the high-velocity density of hydrogen at z=10 AU is higher than that at the other z values, so that when protons from the solar wind collide with hydrogen, this results in a higher density of hydrogen at lower z.

Figure (5.2) is a similar plot for z=30 AU. This shows that the high-velocity distribution function of hydrogen is lower than at z=10 AU because the density of protons from the solar wind decreases (as r^{-2}). A similar trend is seen in Figure (5.3) for z=60 AU.

In addition to results above, we observed that the v_z value of the high-velocity peak decreases when z increases. The width of the high-velocity peak decreases with increasing z due to it depends on momenta. The momenta are different at each z, This results in the width of high-velocity peak different too. We found that the width of high-velocity at the higher z narrower than the lower z.

For the low-velocity peak, there is no z-dependence, as expected. In Figures (5.1)-(5.3), the low-velocity peak has $f_H = 10^{-14.41}$ cm^{-3}(eV/c)$^{-3}$.

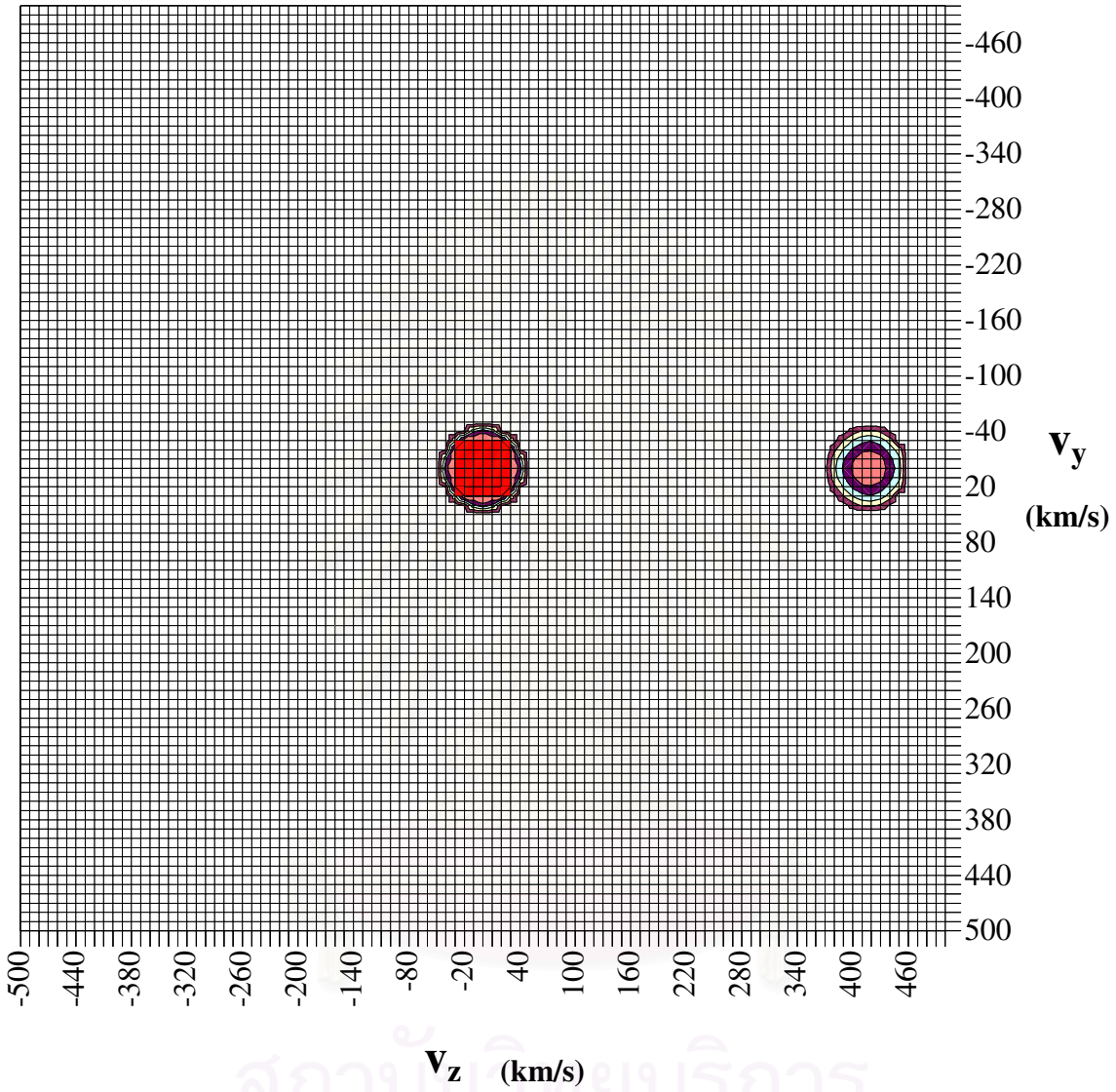

Figure 5.1: Surface plot of the distribution function of hydrogen in v_y-v_z space at z=10 AU. Contours are for $f_H = 10^{-21.0}$, $10^{-20.5}$, $10^{-20.0}$, $10^{-19.5}$ and $10^{-19.0}$ $\mathrm{cm}^{-3}(\mathrm{eV/c})^{-3}$.

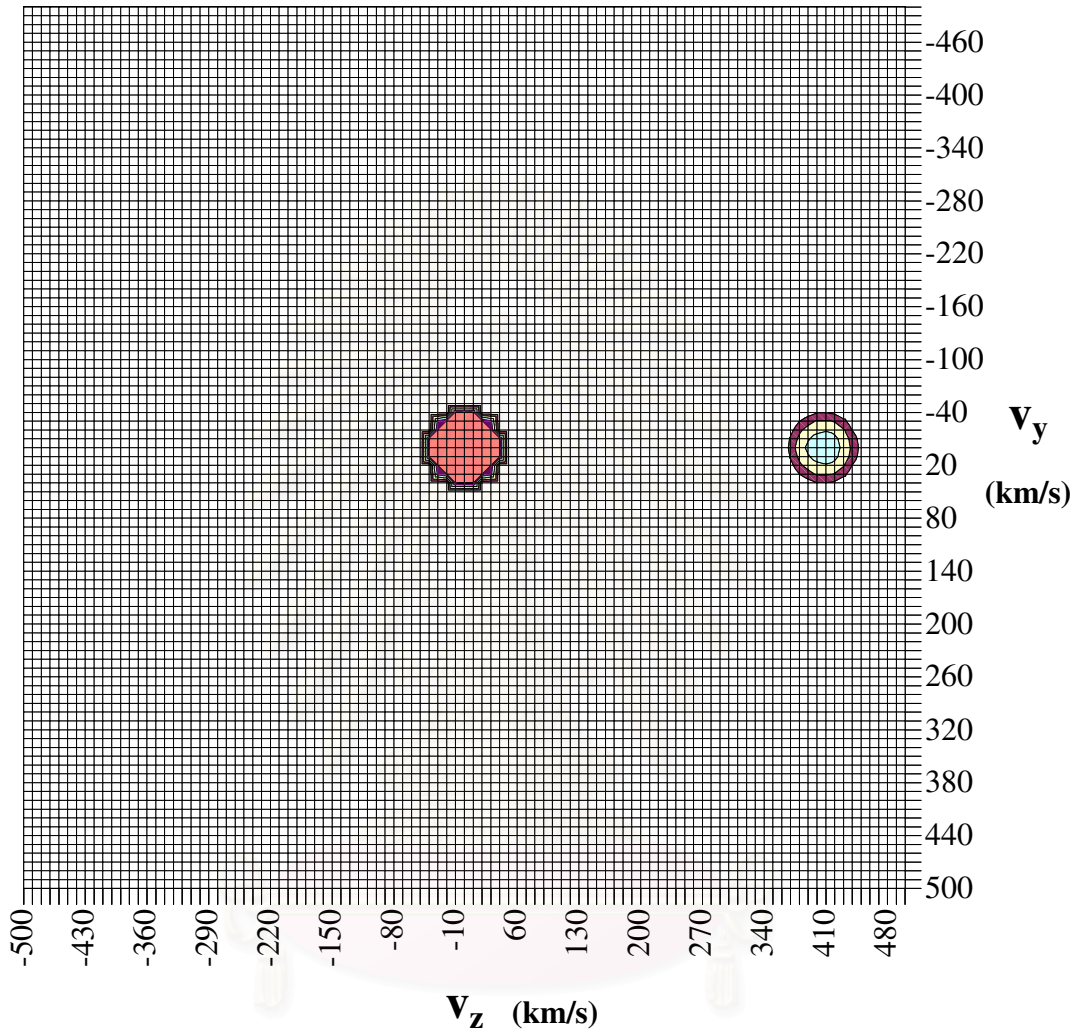

Figure 5.2: Surface plot of the distribution function of hydrogen in v_y-v_z space at z=30 AU. Contours are for $f_H = 10^{-21.3}$, $10^{-21.1}$, $10^{-20.9}$ and $10^{-20.5}$ $\mathrm{cm}^{-3}(\mathrm{eV/c})^{-3}$.

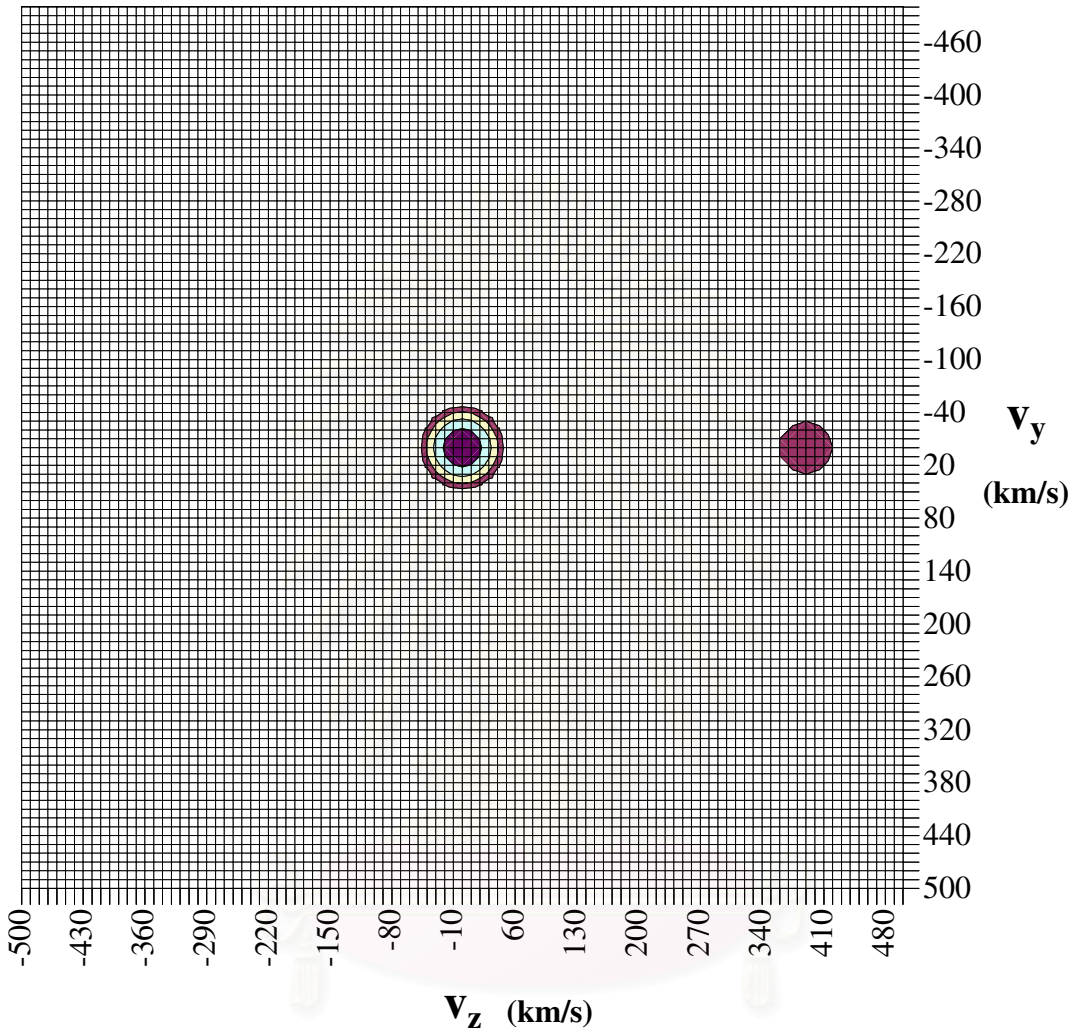

Figure 5.3: Surface plot of the distribution function of hydrogen in v_y-v_z space at z=60 AU. Contours are for $f_H = 10^{-22.0}$, $10^{-20.0}$, $10^{-18.0}$, $10^{-16.0}$ and $10^{-14.0}$ $\mathrm{cm}^{-3}(\mathrm{eV/c})^{-3}$.

Chapter 6

Conclusions

In this thesis, we study the distribution of interstellar neutral hydrogen due to atomic interactions in the outer heliosphere. We use the collisional Boltzmann equation to numerically simulate the interactions of protons from the solar wind with interstellar neutral hydrogen. We calculate the distribution function by a numerical method and we choose Gauss-Laguerre integration to solve the integral terms. The resulting distribution function shows the effects of charge exchange and elastic collisions.

The boltz program is written to solve the integral terms of Boltzmann equation and we set initial conditions and a specially designed system of grids. We have tested the code for obtaining the distribution of hydrogen atoms in v_y-v_z space. The test was successful in that results from these simulation indicated that the distribution of hydrogen remains mostly at low velocities $|v_z| < 10$ km/s and is also distributed at high velocity, $v_z \approx 400$ km/s, or the velocity of protons from the solar wind. Normally we would not have a distribution of hydrogen at high velocity, but that distribution of hydrogen at high velocity results due to hydrogen collisions with protons by charge exchange and the elastic collision process. This creates atoms of hydrogen at high velocity because protons from the solar wind exchange the charge and become atoms.

The resulting distribution function shows the effects of charge exchange and elastic collisions. Our results along with other information should be useful for further simulations to estimate the size of the solar system and understand some characteristics of the heliosphere.

For suggestions for future work, in this thesis we neglect the flow of particles, so in the future we can develop the program further by including streaming in order to examine the observed density of particles when time passes and including ionization processes to lead to more correct physical results.

References

Arfken, G. B., and Weber, H. J. Mathematical Methods for Physicists. 4th ed. San Diego: Academic Press, 1995.

Cravens, T. E. Physics of Solar System Plasmas. New York: Cambridge University Press, 1997.

Foukal, P. Solar Astrophysics. New York: John Wiley & Sons, 1990.

Jokipii, J. R., and Frank, B. M. Scientific American. April, 1995.

Pauls, L., and Zank, G. "Modeling the Solar Wind/Interstellar Wind Interaction," 25th International Cosmic Ray Conference **2** (1997): 241-244.

Pauls, L., and Zank, G. "Modelling the Heliosphere," Space Science Reviews **78** (1996): 95-106.

Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. Numerical Recipies in C. 2nd ed. New York: Cambridge University Press, 1992.

Appendices

Appendix A

Boltz Program

```
/*
   boltz.c -- June 4, 1998

      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:

         streaming

      Necessary files and subroutines:

         boltz.c     main

         initial.c   initial
         nrutil.c    nrerror, dvector, lvector, dmatrix, darray,
                     free_dvector, free_lvector, free_dmatrix,
                     free_darray
         printout.c  printout
         stream.c    stream

      Variables input from the user:

           starttime   Initial value of t (days)
           stoptime    Final value of t (days)
           timestep    Time step (days)
           printtime   Printing interval (AU)
           nmu         Number of mu points
           length      Length of simulation region (AU)
           startlogp   Starting value of log10(p) (units of eV/c)
           stoplogp    Final value of log10(p) (units of eV/c)
           logpstep    Step in log10(p) (units of eV/c)
           printextra  Print extra diagnostic information? (0/1)

      Worachate Boonplod, Panita Boonma, and David Ruffolo
      Department of Physics
      Faculty of Science
      Chulalongkorn University
      Bangkok 10330, Thailand
*/

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

#define C    173.1       /* in AU/day light velocity */
```

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

double ***f;
double *tosun;
double *dens, *px, *temp;

main()
{
   int    nmun,nmu,nmu0,nmu1,nzi,np,nn,k;
   int    printextra,u,w,fine[30],nz,l;
   FILE   *fp_dump;
   double nextprint;
   double starttime,stoptime,printtime,timestep,time;
   double startlogp,stoplogp,logpstep,logp;
   double *beta,*ke,length,*p ;
   double zstep,mu0[30],mu1[30],*mu,*theta;
   double ***darray2(),*dvector();
   double Velocity_term();
   void   free_darray2(),free_dvector();
   void   nrerror();
   void   elements(),gen_vel(),ini(),printout(),print_fp(),
          stream(),elastic(),test_f(),grid_t(),grid_p(),fn_theta();

   /* Input parameters from the user */
   printf("Hello! Welcome to boltz.\n");
   printf("Please input the following parameters:\n");
   printf("\nStarting value of time (days): ");
   scanf("%lf",&starttime);
   printf("\nFinal value of time (days): ");
   scanf("%lf",&stoptime);
   printf("\nTime step (days): ");
   scanf("%lf",&timestep);
   printf("\nTime interval after which to print out data (days): ");
   scanf("%lf",&printtime);
   printf("\nNumber of mu points (must be even): ");
   scanf("%d",&nmun);
   printf("\nLength in the z-direction (AU): ");
   scanf("%lf",&length);
   printf("\nStep in the z-direction (AU): ");
   scanf("%lf",&zstep);
   printf("\nStarting value of log(p) (units of eV/c): ");
   scanf("%lf",&startlogp);
   printf("\nFinal value of log(p) (units of eV/c): ");
   scanf("%lf",&stoplogp);
   printf("\nStep in log(p) (units of eV/c): ");
   scanf("%lf",&logpstep);

   nn = 100;
   np=22;
   if(np <= 0) nrerror("boltz: np <= 0");

   beta  = dvector(0,nn);
   ke    = dvector(0,nn);
   p     = dvector(0,nn);
   mu    = dvector(0,nn);
   tosun = dvector(0,nn);

   /* Following memory allocation moved from elastic.c
```

```
   find p[w] */
grid_p(p,&np,fine);
for(w=0;w<=np;w++) printf("\nfine[w=%2d]=%d",w,fine[w]);
for(w=0;w<=np;w++) tosun[w] = 0.0;

printf("\nDo you want me to print extra diagnostic information? ");
printf("\n(enter 1 for yes, 0 for no) ");
scanf("%d",&printextra);

/* Calculating nz, ke, and beta. */
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("\nboltz.c: ke[%2d] = %12lf, beta[%2d] = %12lf",
              w,ke[w],w,beta[w]);
}
ke[0] = beta[0] = 0;
printf("\n");

/*
   Calculating nz, mustep, zstep; length is now an integral
   multiple of zstep.  Reset sstep to ensure that
   zstep=mustep*sstep.
*/
nz=(length/zstep)+0.5;
if(nz==0) nrerror("boltz:nz==0");
printf("\ninput w=1:nz(double)=%lf,nz(int)=%d,length=%lf\n",
        length/zstep,nz,nz*zstep);
length = nz*zstep;
nmu = nmun+4;/* must be nmu=nmu0+nmu1*/
nzi  = nz;
temp = dvector(1,nzi);
dens = dvector(1,nzi);
px   = dvector(1,nzi);

grid_t(mu0,mu1,&nmu0,&nmu1);

/* Echoing */
printf("\nYou input the following parameters:\n");
printf("\nStarting time (days)                    : %12lf",
       starttime);
printf("\nFinal time (days)                       : %12lf",
       stoptime);
printf("\nTime step (days)                        : %12lf",
       timestep);
printf("\nPrint interval (days)                   : %12lf",
       printtime);
printf("\nNumber of mu points                     : %5d",nmun);
printf("\nLength (AU)                             : %12lf",length);
printf("\nStep in z-direction                     : %12lf",zstep);
printf("\nStarting value of log(p) (units of eV/c): %12lf",
       startlogp);
printf("\nFinal value of log(p) (units of eV/c)   : %12lf",
       stoplogp);
printf("\nStep in log(p) (units of eV/c)          : %12lf\n",
       logpstep);
printf("\nPrintextra                              : %5d\n",
       printextra);
```

```
theta = dvector(0,nmu0+nmu1-2);
fn_theta(theta);

for (w=0;w<=np;w++) printf("\np[%2d] = %12.4f",w,p[w]);

/* Idiot Proofing */

if(stoptime < starttime) nrerror("boltz: stoptime < starttime");
if(timestep <= 0) nrerror("boltz: timestep <= 0");
if(printtime < timestep) nrerror("boltz: printtime < timestep");
if(nmun%2 != 0 || nmun <= 0) nrerror("boltz: nmun is bad");
if(length <= 0) nrerror("boltz: length <= 0");
if(stoplogp < startlogp) nrerror("boltz: stoplogp < startlogp");
if(logpstep <= 0) nrerror("boltz: logpstep <= 0");
if(zstep <= 0) nrerror("boltz: zstep <= 0");

/* Defining array */

f = darray2(0,np,1,nz,0,nmu);
printf("\nboltz.c: f(0-%d,1-%d,0-%d)",np,nz,nmu);

if(printextra) printf("\n Now we are here step1 \n");

ini(theta,np,p,nz,zstep,nmu0,nmu1,fine);
printf("\n Now we are here step1 \n");

printout(starttime,printtime,np,p,theta,mu,mu0,mu1,nz,zstep,nmu,
        nmu0,nmun,nmu1,fine);
test_f(np,p,nz,zstep,nmu0,nmu1,fine);
nextprint = starttime + printtime;

   /*
      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).
   */

protondata(nz, zstep);

for(time=starttime;time+timestep<=stoptime+timestep/2.0;
    time+=timestep) {
   /*
      No need for streaming when w=0 because p=0 and therefore
      v_z=0 use initial gausian distribution without streaming.
   */
   test_f(np,p,nz,zstep,nmu0,nmu1,fine);
   elastic(timestep,theta,beta,np,p,mu,mu0,mu1,nz,zstep,nmu,nmun,
           nmu0,nmu1,fine);
   test_f(np,p,nz,zstep,nmu0,nmu1,fine);
   /*
   for(w=1;w<=np;w++)
      stream(timestep,beta,w,nz,zstep,nmu,printextra);
   elastic(timestep/2.0,theta,beta,np,p,mu,mu0,mu1,nz,zstep,nmu,
           nmun,nmu0,nmu1,fine);
   if(time+timestep >= nextprint-0.01*timestep) {
      printf("\n\n time = %lf\n",time+timestep);
   */
```

```
            printout(time+timestep,printtime,np,p,theta,mu,mu0,mu1,nz,
                     zstep,nmu,nmu0,nmun,nmu1,fine);
            printf("\n*************** test_f after printout2");
            test_f(np,p,nz,zstep,nmu0,nmu1,fine);
            nextprint += printtime;
      /* } */
      } /* End for time loop. */

      /* "DUMP" OUT f,SO THAT THE RUN MAY BE CONTINUED. */
      if(DUMP) {
         fp_dump = fopen("dump.dat","w");
         printf("\n\nNow dumping f(mu,z) for a later run...\n");
         /* w=0 */
         for (l=1;l<=nz;l++) fprintf(fp_dump,"%12le\n",f[0][l][0]);
         /* w>0 */
         for(w=1;w<=np;w++) {
            if(fine[w]==1) nmu=nmu1;
            else nmu=nmu0;
            for(l=1;l<=nz;l++) {
               for(u=0;u<nmu;u++) {
                  if(fine[w]==0) mu[u]=mu0[u];
                  else mu[u]=mu1[u];
                  fprintf(fp_dump,"\nf[w=%d][l=%d][u=%d]=%12le\n",w,l,u,
                          f[w][l][u]);
               }
            }
         }
         fclose(fp_dump);
      }
      free_dvector(theta,0);
      free_dvector(tosun,0);
      free_dvector(mu,0);
      free_dvector(p,0);
      free_dvector(ke,0);
      free_dvector(beta,0);
      free_darray2(f,0,np,1,nz,0);
      free_dvector(dens,1);
      free_dvector(px,1);
      free_dvector(temp,1);
      printf("\n ----- :o Wow! I have finished my work!  :) ----\n");
      printf("\n -------------- Finish Boltz.c -------------\n");
}

/*-------------------------arctan function----------------------*/
double arctan(vy,vz)
double vy,vz;
{
      double pi;

      pi = 4.0*atan(1.0);
      if((vy==0.0) && (vz==0.0)) {  /* at point 0 */
         return 0.0;
      } else if((vy>0.0) && (vz==0.0)) {  /* 90 degree */
         return 90.0;
      } else if((vy<0.0) && (vz==0.0)) {  /* 270 degree */
         return 90.0;
      } else if((vy>=0.0) && (vz>0.0)) {  /* Q1 */
         return 180.0*atan(vy/vz)/pi;
      } else if((vy>=0.0) && (vz<0.0)) {  /* Q2 */
         return 180.0+180.0*atan(vy/vz)/pi;
```

```
   } else if((vy<0.0) && (vz<0.0)) {  /* Q3 */
      return 180.0-180.0*atan(vy/vz)/pi;
   } else if((vy<0.0) && (vz>0.0)) {  /* Q4 */
      return -180.0*atan(vy/vz)/pi;
   }
}

/* This function is discritize theta(0 - 13) from 0 - 180 */
void fn_theta(theta)
double theta[];
{
   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] = 90.0;
   theta[9] = 120.0;
   theta[10] = 150.0;
   theta[11] = 165.0;
   theta[12] = 175.0;
   theta[13] = 180.0;
}

/* This function is discritize mu0(0 - 9) and mu1(0-4) */
void grid_t(mu0,mu1,nmu0,nmu1)
double mu0[],mu1[];
int *nmu0,*nmu1;
{
   int j;
   double pi;

   pi=4.0*atan(1.0);
   *nmu0  = 10;
   *nmu1  = 5;

   mu0[0]=cos(0.0*(pi/180.0));
   mu0[1]=cos(15.0*(pi/180.0));
   mu0[2]=cos(30.0*(pi/180.0));
   mu0[3]=cos(60.0*(pi/180.0));
   mu0[4]=cos(90.0*(pi/180.0));
   mu0[5]=cos(120.0*(pi/180.0));
   mu0[6]=cos(150.0*(pi/180.0));
   mu0[7]=cos(165.0*(pi/180.0));
   mu0[8]=cos(175.0*(pi/180.0));
   mu0[9]=cos(180.0*(pi/180.0));

   mu1[0]=cos(0.0*(pi/180.0));
   mu1[1]=cos(1.0*(pi/180.0));
   mu1[2]=cos(2.0*(pi/180.0));
   mu1[3]=cos(4.0*(pi/180.0));
   mu1[4]=cos(6.0*(pi/180.0));

   for(j=1;j<*nmu0;j++) printf("\n>>mu0[%d]=%lf",j,mu0[j]);
   for(j=1;j<*nmu1;j++) printf("\n>>mu1[%d]=%lf",j,mu1[j]);
}
```

```
/* This function is discritize p(11-12) from p(0-24) */
void grid_p(p,np,fine)
double p[];
int *np, fine[];
{
   double a;

   a=3136.125;
   *np = 24;

   /*----------fine[...] = 0  for course grid-----------*/
   /*----------fine[...] = 1  for fine grid-------------*/

   p[0]  = (0.0*a);      fine[0]  = 0;
   p[1]  = (10.0*a);     fine[1]  = 0;
   p[2]  = (15.0*a);     fine[2]  = 0;
   p[3]  = (20.0*a);     fine[3]  = 0;
   p[4]  = (25.0*a);     fine[4]  = 0;
   p[5]  = (30.0*a);     fine[5]  = 0;
   p[6]  = (40.0*a);     fine[6]  = 0;
   p[7]  = (60.0*a);     fine[7]  = 0;
   p[8]  = (100.0*a);    fine[8]  = 0;
   p[9]  = (140.0*a);    fine[9]  = 0;
   p[10] = (180.0*a);    fine[10] = 0;
   p[11] = (220.0*a);    fine[11] = 0;
   p[12] = (260.0*a);    fine[12] = 0;
   p[13] = (310.0*a);    fine[13] = 0;
   p[14] = (320.0*a);    fine[14] = 1;
   p[15] = (340.0*a);    fine[15] = 1;
   p[16] = (360.0*a);    fine[16] = 1;
   p[17] = (380.0*a);    fine[17] = 1;
   p[18] = (395.0*a);    fine[18] = 1;
   p[19] = (400.0*a);    fine[19] = 0;
   p[20] = (410.0*a);    fine[20] = 1;
   p[21] = (420.0*a);    fine[21] = 1;
   p[22] = (430.0*a);    fine[22] = 1;
   p[23] = (450.0*a);    fine[23] = 1;
   p[24] = (500.0*a);    fine[24] = 0;
}


/* Program interpo.c  13 June 2000
   use interpolate from test_interpo.c */

double interpo(p,theta,pp,mut,l)
int l;
double p[],theta[],pp,mut;
{
   int w,u,w_k,u_k,u0;
   double a;
   double lf00,lf01,lf10,lf11,lpp,lp0,lp1,frac,ans,tmp1,tmp2;

   a = 3136.125;
/*
   p[0]  = (0.0*a);
   p[1]  = (10.0*a);
   p[2]  = (15.0*a);
   p[3]  = (20.0*a);
   p[4]  = (25.0*a);
```

```
p[5]  = (30.0*a);
p[6]  = (40.0*a);
p[7]  = (60.0*a);
p[8]  = (100.0*a);
p[9]  = (140.0*a);
p[10] = (180.0*a);
p[11] = (220.0*a);
p[12] = (260.0*a);
p[13] = (310.0*a);
p[14] = (320.0*a);
p[15] = (340.0*a);
p[16] = (360.0*a);
p[17] = (380.0*a);
p[18] = (395.0*a);
p[19] = (400.0*a);
p[20] = (410.0*a);
p[21] = (420.0*a);
p[22] = (430.0*a);
p[23] = (450.0*a);
p[24] = (500.0*a);

for (u=0;u<=9;u++)  f[0][l][u]=0.0;
for (u=0;u<=9;u++)  f[1][l][u]=1.803798e-15;
for (u=0;u<=9;u++)  f[2][l][u]=6.977342e-16;
for (u=0;u<=9;u++)  f[3][l][u]=1.845835e-16;
for (u=0;u<=9;u++)  f[4][l][u]=3.339616e-17;
for (u=0;u<=9;u++)  f[5][l][u]=4.132388e-18;
for (u=0;u<=9;u++)  f[6][l][u]=2.024009e-20;
for (u=0;u<=9;u++)  f[7][l][u]=5.084466e-27;
for (u=0;u<=9;u++)  f[8][l][u]=3.857893e-48;
for (u=0;u<=9;u++)  f[9][l][u]=8.063202e-80;
for (u=0;u<=9;u++)  f[10][l][u]=4.642130e-122;
for (u=0;u<=9;u++)  f[11][l][u]=7.361724e-175;
for (u=0;u<=9;u++)  f[12][l][u]=3.215839e-238;
for (u=0;u<=9;u++)  f[13][l][u]=0.0;
for (u=0;u<=4;u++)  f[14][l][u]=0.0;
for (u=0;u<=4;u++)  f[15][l][u]=0.0;
for (u=0;u<=4;u++)  f[16][l][u]=0.0;
for (u=0;u<=4;u++)  f[17][l][u]=0.0;
for (u=0;u<=4;u++)  f[18][l][u]=0.0;
for (u=0;u<=9;u++)  f[19][l][u]=0.0;
for (u=0;u<=4;u++)  f[20][l][u]=0.0;
for (u=0;u<=4;u++)  f[21][l][u]=0.0;
for (u=0;u<=4;u++)  f[22][l][u]=0.0;
for (u=0;u<=4;u++)  f[23][l][u]=0.0;
for (u=0;u<=9;u++)  f[24][l][u]=0.0;

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] = 90.0;
theta[ 9] = 120.0;
theta[10] = 150.0;
theta[11] = 165.0;
theta[12] = 175.0;
```

```
   theta[13] = 180.0;

   for(w=0;w<=24;w++) {
      for(u=0;u<=9;u++) {
         printf("\nf[w=%d][u=%d]=%le",w,u,f[w][l][u]);
      }
   }
*/

   /*--------------------case 1-------------------------*/
   if((pp>=0.0)&&(pp<=p[1])) {
      if(mut>theta[5]) {
         u=5; u_k=1; u0=1;
      } else {
         u=0; u_k=5; u0=0;
      }
      while(mut>theta[u+u_k]) {
         u = u+u_k;
         u0= u0+1;
      }
      if(f[1][l][u0]*f[1][l][u0+u_k]*f[0][l][0]==0) {
         return -999;
      } else {
         frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
         tmp1 = ((1-frac)*log10(f[1][l][u0]))
                +(frac*log10(f[1][l][u0+u_k]));

         frac = pp/p[1];
         ans  = ((1-frac)*log10(f[0][l][0]))+(frac*tmp1);
         return ans;
      } /* end else */
   } /* end if case 1 */

   /*--------------------case 2-----------------------*/
   else if((pp>p[1])&&(pp<=p[13])) {
      if(pp==(259.8051736*3136.125)) printf("\ncase=> 2");
      w=1; u=0; w_k=1; u_k=5; u0=0;
      if(mut>theta[5]) {
         u=5; u_k=1; u0=1;
      }
      while(pp>p[w+w_k]) {
         w=w+w_k;
      }
      while(mut>theta[u+u_k]) {
         u=u+u_k; u0=u0+1;
      }
      if(f[w][l][u0]*f[w][l][u0+u_k]*f[w+w_k][l][u0]
         *f[w+w_k][l][u0+u_k]==0) {
         return -999;
      } else {
         lf00 = log10(f[w][l][u0]);
         lf01 = log10(f[w][l][u0+u_k]);
         lf10 = log10(f[w+w_k][l][u0]);
         lf11 = log10(f[w+w_k][l][u0+u_k]);

         lpp  = log10(pp);
         lp0  = log10(p[w]);
         lp1  = log10(p[w+w_k]);

         frac = (lpp-lp0)/(lp1-lp0);
         tmp1 = (1-frac)*lf00+frac*lf10;
```

```
      tmp2 = (1-frac)*lf01+frac*lf11;

      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 2 */

/*--------------------case 3-----------------------*/
else if((pp>=p[14])&&(pp<=p[18])&&(mut<=theta[4])) {
   w=14; u=0; w_k=1; u_k=1; u0=0;
   while(pp>p[w+w_k]) {
      w=w+w_k;
   }
   while(mut>theta[u+u_k]) {
      u = u+u_k;
      u0= u0+1;
   }
   if(f[w][l][u0]*f[w][l][u0+u_k]*f[w+w_k][l][u0]
      *f[w+w_k][l][u0+u_k]==0) {
      return -999;
   } else {
      lf00 = log10(f[w][l][u0]);
      lf01 = log10(f[w][l][u0+u_k]);
      lf10 = log10(f[w+w_k][l][u0]);
      lf11 = log10(f[w+w_k][l][u0+u_k]);

      lpp  = log10(pp);
      lp0  = log10(p[w]);
      lp1  = log10(p[w+w_k]);

      frac = (lpp-lp0)/(lp1-lp0);
      tmp1 = (1-frac)*lf00+frac*lf10;
      tmp2 = (1-frac)*lf01+frac*lf11;

      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 3 */

/*--------------------case 4-----------------------*/
else if((pp>p[18])&&(pp<=p[20])&&(mut<=theta[4])) {
   w=18; u=0; w_k=2; u_k=1; u0=0;
   while(pp>p[w+w_k]) {
      w=w+w_k;
   }
   while(mut>theta[u+u_k]) {
      u = u+u_k; u0= u0+1;
   }
   if(f[w][l][u0]*f[w][l][u0+u_k]*f[w+w_k][l][u0]
      *f[w+w_k][l][u0+u_k]==0) {
      return -999;
   } else {
      lf00 = log10(f[w][l][u0]);
      lf01 = log10(f[w][l][u0+u_k]);
      lf10 = log10(f[w+w_k][l][u0]);
      lf11 = log10(f[w+w_k][l][u0+u_k]);

      lpp = log10(pp);
      lp0 = log10(p[w]);
```

```
      lp1 = log10(p[w+w_k]);

      frac = (lpp-lp0)/(lp1-lp0);
      tmp1 = (1-frac)*lf00+frac*lf10;
      tmp2 = (1-frac)*lf01+frac*lf11;

      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 4 */

/*--------------------case 5-----------------------*/
else if((pp>p[20])&&(pp<=p[23])&&(mut<=theta[4])) {
   w=20; u=0; w_k=1; u_k=1; u0=0;
   while(pp>p[w+w_k]) {
      w=w+w_k;
   }
   while(mut>theta[u+u_k]) {
      u=u+u_k; u0=u0+1;
   }
   if(f[w][l][u0]*f[w][l][u0+u_k]*f[w+w_k][l][u0]
      *f[w+w_k][l][u0+u_k]==0) {
      return -999;
   } else {
      lf00 = log10(f[w][l][u0]);
      lf01 = log10(f[w][l][u0+u_k]);
      lf10 = log10(f[w+w_k][l][u0]);
      lf11 = log10(f[w+w_k][l][u0+u_k]);

      lpp = log10(pp);
      lp0 = log10(p[w]);
      lp1 = log10(p[w+w_k]);

      frac = (lpp-lp0)/(lp1-lp0);
      tmp1 = (1-frac)*lf00+frac*lf10;
      tmp2 = (1-frac)*lf01+frac*lf11;

      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 5 */

/*--------------------case 6-----------------------*/
else if((pp>p[13])&&(pp<p[14])&&(mut>=theta[0])&&(mut<=theta[4])) {
   w=13; u=0; w_k=1; u_k=4; u0=0;
   while(mut>theta[u+u_k]) {
      u=u+u_k; u0=u0+1;
   }
   if(f[13][l][0]*f[13][l][1]*f[14][l][u0]*f[14][l][u+u_k]==0) {
      return -999;
   } else {
      lf00 = log10(f[13][l][0]);
      lf01 = log10(f[13][l][1]);
      frac = mut/(theta[5]);
      tmp1 = (1-frac)*lf00+frac*lf01;

      lf10 = log10(f[14][l][u0]);
      lf11 = log10(f[14][l][u+u_k]);
      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
```

```
      tmp2 = (1-frac)*lf10+frac*lf11;

      lpp  = log10(pp);
      lp1  = log10(p[14]);
      lp0  = log10(p[13]);

      frac = (lpp-lp0)/(lp1-lp0);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 6 */

/*--------------------case 7-----------------------*/
else if((pp>p[23]) && (pp<=p[24]) && (mut>=theta[0]) &&
        (mut<=theta[4])) {
   w=23; u=0; w_k=1; u_k=4; u0=0;
   while(mut>theta[u+u_k]) {
      u=u+u_k; u0=u0+1;
   }
   if(f[23][l][0]*f[23][l][u0+u_k]*f[24][l][0]*f[24][l][1]==0) {
      return -999;
   } else {
      lf00 = log10(f[23][l][u0]);
      lf01 = log10(f[23][l][u0+u_k]);
      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
      tmp1 = (1-frac)*lf00+frac*lf01;

      lf10 = log10(f[24][l][0]);
      lf11 = log10(f[24][l][1]);
      frac = (mut/theta[5]);
      tmp2 = (1-frac)*lf10+frac*lf11;

      lpp  = log10(pp);
      lp1  = log10(p[24]);
      lp0  = log10(p[23]);

      frac = (lpp-lp0)/(lp1-lp0);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 7 */

/*--------------------case 8-----------------------*/
else if((pp>p[13])&&(pp<p[14])&&(mut>theta[4])&&(mut<=theta[5])) {
   w=13; u=4; w_k=1; u_k=1; u0=0;
   if(f[13][l][u0]*f[13][l][1]*f[14][l][u0]*f[14][l][4]*f[13][l][1]
      *f[19][l][1]==0) {
      return -999;
   } else {
      frac = log10(p[14]/p[13])/log10(p[19]/p[13]);
      lf11 = (1-frac)*log10(f[13][l][1])+frac*log10(f[19][l][1]);

      frac = mut/theta[5];
      tmp1 = (1-frac)*log10(f[13][l][0])+frac*log10(f[13][l][1]);
      tmp2 = (1-frac)*log10(f[14][l][4])+frac*lf11;

      frac = log10(pp/p[13])/log10(p[14]/p[13]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 8 */
```

```
/*--------------------case 9-----------------------*/
else if((pp>=p[14]) && (pp<=p[19]) && (mut>theta[4]) &&
        (mut<=theta[5])) {
   w=14; u=4; w_k=5; u_k=1; u0=0;
   if(f[w][l][u0]*f[13][l][u0+u_k]*f[13][l][1]*f[18][l][4]
      *f[19][l][1]*f[19][l][4]*f[20][l][4]==0) {
      return -999;
   } else {
      if(pp>=p[18]) {
         lf00 = log10(f[18][l][4]);
         frac = (log10(pp)-log10(p[18]))/(log10(p[20])-
                log10(p[18]));
         lf01 = log10(f[20][l][4]);

         tmp1 = (1-frac)*lf00+frac*lf01;
      } else {
         while(pp>p[w+w_k]) {
            w = w+w_k;
         }
         lf00 = log10(f[w][l][4]);
         lf01 = log10(f[w+w_k][l][4]);
         frac = (log10(pp)-log10(p[w]))/(log10(p[w+w_k])-
                log10(p[w]));

         tmp1 = (1-frac)*lf00+frac*lf01;
      }
      frac = (log10(pp)-log10(p[13]))/(log10(p[19])-log10(p[13]));
      tmp2 = (1-frac)*log10(f[13][l][1])+frac*log10(f[19][l][1]);

      frac = (mut-theta[4])/(theta[5]-theta[4]);
      ans  = (1-frac)*tmp1+frac*tmp2;

      printf("\ntmp1=%le ans9=%le",tmp1,ans);
      return ans;
   } /* end else */
} /* end if case 9 */

/*--------------------case 10----------------------*/
else if((pp>p[19])&&(pp<=p[23])&&(mut>theta[4])&&(mut<=theta[5])) {
   w=19; u=4; w_k=4; u_k=1; u0=0;
   if(f[19][l][1]*f[23][l][4]*f[24][l][1]*f[24][l][0]==0 ) {
      return -999;
   } else {
      frac = log10(p[23]/p[19])/log10(p[24]/p[19]);
      lf01 = (1-frac)*log10(f[19][l][1])+frac*log10(f[24][l][1]);

      frac = (mut-theta[4])/(theta[5]-theta[4]);
      tmp1 = (1-frac)*log10(f[24][l][0])+frac*log10(f[24][l][1]);
      tmp2 = (1-frac)*log10(f[23][l][4])+frac*lf01;

      frac = log10(pp/p[23])/(theta[5]-theta[4]);
      ans  = (1-frac)*tmp2+frac*tmp1;
      return ans;
   } /* end else */
} /* end if case 10 */

/*--------------------case 11----------------------*/
else if((pp>p[23])&&(pp<=p[24])&&(mut>theta[4])&&(mut<=theta[5])) {
   w=23; u=4; w_k=1; u_k=1; u0=0;
   if(f[19][l][1]*f[24][l][1]*f[24][l][0]*f[23][l][4]==0) {
```

```
      return -999;
   } else {
      frac = log10(p[23]/p[19])/log10(p[24]/p[19]);
      lf01 = (1-frac)*log10(f[19][l][1])+frac*log10(f[24][l][1]);

      frac = mut/theta[5];
      tmp1 = (1-frac)*log10(f[23][l][4])+frac*lf01;
      tmp2 = (1-frac)*log10(f[24][l][0])+frac*log10(f[24][l][1]);

      frac = log10(pp/p[23])/log10(p[24]/p[23]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 11 */

/*--------------------case 12----------------------*/
else if(((pp>p[13])&&(pp<=p[19]))&&(mut>theta[5])) {
   w=13; u=5; w_k=6; u_k=1; u0=1;
   while(mut>theta[u+u_k]) {
      u=u+u_k; u0=u0+1;
   }
   if(f[13][l][u0]*f[13][l][u0+u_k]*f[19][l][u0]*f[19][l][u0+u_k]
      ==0) {
      return -999;
   } else {
      frac = log10(pp/p[13])/log10(p[19]/p[13]);
      tmp1 = (1-frac)*log10(f[13][l][u0])+frac*log10(f[19][l][u0]);
      tmp2 = (1-frac)*log10(f[13][l][u0+u_k])
             +frac*log10(f[19][l][u0+u_k]);

      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 12 */

/*--------------------case 13----------------------*/
else if((pp>p[19])&&(pp<=p[24])&&(mut>theta[5])) {
   w=19; u=5; w_k=5; u_k=1; u0=1;
   while(mut>theta[u+u_k]) {
      u=u+u_k; u0=u0+1;
   }
   if(f[19][l][u0]*f[19][l][u0+u_k]*f[24][l][u0]
      *f[24][l][u0+u_k]==0) {
      return -999;
   } else {
      frac = log10(pp/p[19])/log10(p[24]/p[19]);
      tmp1 = (1-frac)*log10(f[19][l][u0])+frac*log10(f[24][l][u0]);
      tmp2 = (1-frac)*log10(f[19][l][u0+u_k])+frac
           *log10(f[24][l][u0+u_k]);

      frac = (mut-theta[u])/(theta[u+u_k]-theta[u]);
      ans  = (1-frac)*tmp1+frac*tmp2;
      return ans;
   } /* end else */
} /* end if case 13 */

/*--------------------case 14----------------------*/
else if((pp>p[24])&&(mut!=0)) {
   return -999;
} /* end else */
printf("\n");
```

```
} /* end interpo */

/* test_f December 23rd 1998 */
void test_f(np,p,nz,zstep,nmu0,nmu1,fine)
double p[],zstep;
int np,nmu0,nmu1,fine[],nz;
{
   int u,w,l,nmu;

   /* for w=0 */
   for(l=1;l<=nz;l++) {
      printf("\nf[0][l=%2d][0]=%le",l,f[0][l][0]);
   }

   /* for w>=1 */
   for(w=1;w<=np;w++) {
      if(fine[w]==0) nmu=nmu0;
      else nmu=nmu1;
      for(l=1;l<=nz;l++) {
         for(u=0;u<nmu;u++) {
            if(l==90) printf("\nf[w=%d][l=%d][u=%d]=%le",w,l,u,
                              f[w][l][u]);
         }
      }
   }
} /* end test_f */
```

Appendix B

Elastic Program

/* elastic.c – August 31, 2000 Elastic and include charge exchange */

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

#define   C    29979000000.0  /* in cm/s */
#define   M    938780000.0    /* Mass of a hydrogen atom (eV/c^2) */
#define   K    8.617384e-05   /* in eV/K -- Boltzmann constant */
#define   pi   3.141593

double **fp;
extern double ***f;
extern double *dens,*px,*temp;

void elastic(timestep,theta,beta,np,p,mu,mu0,mu1,nz,zstep,nmu
             ,nmun,nmu0,nmu1,fine)
int nmu,nmun,np,nmu0,nmu1,fine[],nz;
double beta[],p[],theta[],mu[],timestep,zstep,mu0[],mu1[];
{
   static char fn_n[]="l_plot.dat";
   FILE   *fp_n;
   int    u,w,w1,u1,anmu,anmu1;
   int    l,i,case_i;
   double **fnew,muu,muu1,v,v1;
   double arg,change,value1,value2,value3,*p1;
   double term1,term2_1,term2_2,term2_3,term3;
   double vel(),volume();
   double **dmatrix2(),approx(), *dvector();

   /* pi = 4.0*atan(1.0); */
   printf("\n----------------begin elastic.c-------------------\n");
   fnew = dmatrix2(0,np,0,nmu0+nmu1-2);
   fp   = dmatrix2(0,np,0,nmu0+nmu1-2);
   p1   = dvector(0,100);

   for(i=0;i<=100;i++) p1[i]=p[i];

   if((fp_n=fopen(fn_n,"w"))==NULL) {
      printf("\nError to openfile '%s' for writting.\n");
      exit(1);
   }
   for(l=1;l<=nz;l++) {
      printf("l=%d\n",l);
      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++) {
      /** m_mu[u]=mu[u] **/
      fnew[w][u]=f[w][l][u];
   }
}
arg = -px[l]*px[l]/(2.0*M*K*temp[l]);
if((arg>-30.0)&&(arg<30.0)) {
   fp[0][0] = (dens[l]/pow(2.0*pi*M*K*temp[l],1.5))*exp(arg);
} else {
   if(arg<=-30.0) {
      fp[0][0]=0.0;
   } else {
      nrerror("hello boy chex: bad fp");
   }
}
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) {
         muu=mu0[u];
      } else {
         muu=mu1[u];
      }
      arg = -(p[w]*p[w]-(2.0*muu*p[w]*px[l])+(px[l]*px[l]))
            /(2.0*M*K*temp[l]);
      if((arg>-30) && (arg < 30)) {
         fp[w][u] = ((dens[l]/(pow(2.0*pi*M*K*temp[l],1.5))))
                  *exp(arg);
      } else {
         if(arg <= -30) {
            fp[w][u] = 0.0;
         } else {
            nrerror("hello boy 2chex: bad fp");
         }
      }
      printf("\nl=%d fp[w=%d][u=%d]=%le",l,w,u,fp[w][u]);
   }
}
for(w=0;w<=np;w++) {
   if(fine[w]==0) anmu=nmu0;
   else anmu=nmu1;
   for(u=0;u<anmu;u++) {
      if(fine[w]==0) {
         muu=mu0[u];
      } else {
         muu=mu1[u];
      }
      if(w==0&&u==1) break;
      if(w==0) {
         w1=1;
         u1=0;
      } else {
         w1=w;
         u1=u+1;
      }
      /* muu=mu[u]; */
      for(;w1<=np;w1++) {
         if(fine[w1]==0) anmu1=nmu0;
```

```
else anmu1=nmu1;
for(;u1<anmu1;u1++) {
   if(fine[w1]==0){
      muu1=mu0[u1];
   } else {
      muu1=mu1[u1];
   }
   /** here only: v is in units of cm/s **/
   v  = beta[w]*C;
   v1 = beta[w1]*C;

   /** velocity term **/
   term1 = vel(muu,muu1,v,v1);

   /** volume term **/
   term3 = volume(p,theta,w1,u1);

   /** defined for h-p near o **/
   case_i  = 1;
   term2_1 = approx(case_i,timestep,p,p1,theta,mu,muu
                    ,muu1,v,v1,w,l,u,w1,u1,np,nz,
                    zstep,nmu0,nmu1,fine);
   value1 = timestep*86400.0*term1*term2_1*term3;

   /** defined for h-p near pi **/
   case_i  = 2;
   term2_2 = approx(case_i,timestep,p,p1,theta,mu,muu
                    ,muu1,v,v1,w,l,u,w1,u1,np,nz,
                    zstep,nmu0,nmu1,fine);
   value2 = timestep*86400.0*term1*term2_2*term3;

   /** defined for h-h near 0 **/
   case_i  = 3;
   term2_3 = approx(case_i,timestep,p,p1,theta,mu,muu
                    ,muu1,v,v1,w,l,u,w1,u1,np,nz,
                    zstep,nmu0,nmu1,fine);
   value3 = timestep*86400.0*term1*term2_3*term3;

   change=value1+value2+value3;
   fnew[w][u]+=change;
   if(l==10) {
      fprintf(fp_n,"\nfnew[w=%d][u=%d]=%le",w,u,
              fnew[w][u]);
   }
   if(!(change > -1000000.0 && change < 1000000.0)) {
   /* if(l==10) {
         printf("\nExit value1=%le value2=%le
                 value3=%le change=%le",value1,value2,
                 value3,change);
         exit(1);
      } */
   }
   term3 = volume(p,theta,w,u);

   value1 = timestep*86400.0*term1*term2_1*term3;
   value2  = timestep*86400.0*term1*term2_2*term3;
   value3  = timestep*86400.0*term1*term2_3*term3;

   change=value1+value2+value3;
   fnew[w1][u1]-=change;
   if(l==10) {
```

```
                    fprintf(fp_n,"fnew[w1=%d][u1=%d]=%le",w1,u1,
                                    fnew[w1][u1]);
                  }
               } /* END OF U1 */
               u1=0;
            }    /* END OF W1 */
         }       /* END OF U  */
      }          /* END OF W  */
      f[0][l][0] = fnew[0][0];
      for(w=1;w<=np;w++) {
         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) f[w][l][u]=0.0;
         }
      }
   } /* END OF L  */
   fclose(fp_n);
   free_dvector(p1,0);
   free_dmatrix2(fnew,0,np,0);
   free_dmatrix2(fp,0,np,0);
   printf("\n---------------end elastic.c-----------------\n");
}

/*--------------- 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 */

   int l;
   FILE *fp1;
   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;

   printf("\n----------------Begin protondata()----------------");
   fp1=fopen("pro97new.dat","r");
   fscanf(fp1,"%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; /* also changes km/s to cm/s */

   for(l=1;l<=nz;l++) {
      /* printf("\nl = %d",l); */
      z_value = l*zstep;
      if(l==10) {
         printf("\n\n ***z_value = %le  ***z_new = %le",z_value,
                z_new);
      }
      while(z_value>z_new) {
         z = z_new;
         d = d_new;
         vx = vx_new;
         t = t_new;
         fscanf(fp1,"%le %le %le %le %le %le %le %le",&z_new,&y,
                &d_new,&vx_new,&vy,&p,&t_new,&m);
```

```
      }
      px_old = M*vx*100000.0/C;
      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;
      printf("\n px[l=%d]= %le ",l,px[l]);
      temp[l] = (1.0-frac)*t+frac*t_new;
      printf("\n temp[l=%d] = %le ",l,temp[l]);
   }
   fclose(fp1);
   printf("\n----------------End protondata()----------------");
}


/*------------------- approximate function ----------------------*/
double approx(case_i,timestep,p,p1,theta,mu,muu,muu1,v,v1,w,l,u,w1,
              u1,np,nz,zstep,nmu0,nmu1,fine)
int case_i,w,l,u,w1,u1,np,nz,nmu0,nmu1,fine[];
double timestep,muu,muu1,v,v1,p[],p1[],theta[],mu[],zstep;
{
   int ub,phi_1;
   double value,Ecm,xx,y1,y2,A,a,term2;
   double vel(),E(),gauss(),volume(),sum_term2;

   /* printf("\n-----------------Begin approx()---------------"); */
   sum_term2 = 0.0;
   for(phi_1=0;phi_1<6;phi_1++) {
      Ecm = E(phi_1,p,p1,w,w1,muu,muu1);
      xx  = log10(Ecm);
      if(case_i==1) {
         y1 = 0.5744*xx+5.3335;
         y2 = (0.1706*xx*xx)+(0.1651*xx)+4.1985;
      } else {
         if(case_i==2) {
            y1 = 0.9134*xx+4.3121;
            y2 = 0.8977*xx+3.7897;
         } else {
            if(case_i==3) {
               y1 = 0.764*xx+3.87;
               y2 = 0.878*xx+3.026;
            }
         }
      }
      A = pow(10,y1)*(2.800285609e-18); /* in cm^2 */
      a = pow(10,y2);
      term2 = gauss(case_i,p,p1,mu,muu,muu1,theta,A,a,w,l,u,w1,u1,
                np,nz,zstep,nmu0,nmu1,fine,phi_1);
      sum_term2 = sum_term2+term2;
   }

   return sum_term2;
}

/*--------- Velocity Term ------------*/
double vel(muu, muu1, v, v1)
double muu, muu1, v, v1;
{
   double v_rel;
```

```
   v_rel = v*v+v1*v1-2.0*v*v1*(muu*muu1);
   v_rel = sqrt(v_rel);

   return v_rel;
}

/*---------- Formula for Center of Mass (Ecm)  -------------*/
double E(phi_1,p,p1,w,w1,muu,muu1)
int phi_1,w,w1;
double p[],p1[],muu,muu1;
{
   double sn_sn1,cs_1,e,e1,e2;

   sn_sn1 = sqrt(1.0-muu*muu)*sqrt(1.0-muu1*muu1);
   cs_1   = cos(phi_1*pi/3.0);
   e2=(p[w]*p[w])-((2*p[w]*p1[w1])*(sn_sn1*cs_1+muu*muu1))+
      (p1[w1]*p1[w1]);
   e =e2/(4*M);

   return e;
}

/*---------- gauss term -------------*/
double gauss(case_i,p,p1,mu,muu,muu1,theta,A,a,w,l,u,w1,u1,np,nz,
             zstep,nmu0,nmu1,fine,phi_1)
int case_i,w,l,u,w1,u1,np,nz,nmu0,nmu1,fine[],phi_1;
double p[],p1[],theta[],mu[],A,a,zstep,muu,muu1;
{
   int i,up,phi_p;
   double wg[4],z[4],after1,after2,before1,before2,fh_after,fp_after
          ,fh1_after,fh_before,fp_before,fh1_before,f_total;
   double gau,sum_gau,sum_total,cs[4],sn[4],p_1,p_2,p_3,p_4,t_ppr,
          t_pp1r;
   double interpo(),ini_fp(),p_pri(),p_r(),p1_pri(),p1_r();

   /*--- x1 = 0.415774556783 ---*/
   /*--- x2 = 2.294280360279 ---*/
   /*--- x3 = 6.289945082937 ---*/

   /* printf("\n--------------Begin gauss------------"); */
   sum_gau   = 0.0;
   sum_total = 0.0;

   wg[1] = 7.11093009929e-1;
   wg[2] = 2.78517733569e-1;
   wg[3] = 1.03892565016e-2;

   z[1] = 0.415774567830;
   z[2] = 2.294280360279;
   z[3] = 6.289945082937;

   for(up=1;up<=3;up++) {
      if((case_i==1)||(case_i==3)) {
         cs[up]=(1-(z[up]/a));
      } else {
         cs[up]=((z[up]/a)-1);
      }
      sn[up]=1-(cs[up]*cs[up]);
      sn[up]=sqrt(sn[up]);
      for(phi_p=0;phi_p<6;phi_p++) {
```

```
   /*----------------*/
   /*    formula 1   */
   /*----------------*/
   p_1=p_pri(p,w,mu,muu,muu1,u,p1,w1,u1,cs,up,sn,phi_1,phi_p);

   /*----------------*/
   /*    formula 2   */
   /*----------------*/
   p_2=p_r(p,w,mu,muu,muu1,u,p1,w1,u1,cs,up,sn,phi_1,phi_p);
   p_2=p_2/p_1;
   if(fabs(p_2)>1.0) {
      if(p_2>1.0) p_2=1.0;
      else if(p_2<-1.0) p_2=-1.0;
   }
   t_ppr=acos(p_2);
   t_ppr=(t_ppr*180.0)/pi;

   /*----------------*/
   /*    formula 3   */
   /*----------------*/
   p_3=p1_pri(p,w,mu,muu,muu1,u,u1,p1,w1,cs,up,sn,phi_1,phi_p);

   /*----------------*/
   /*    formula 4   */
   /*----------------*/
   p_4=p1_r(p,w,mu,muu,muu1,u,p1,w1,u1,cs,up,sn,phi_1,phi_p);
   p_4=p_4/p_3;
   if(fabs(p_4)>1.0) {
      if(p_4>1.0) p_4=1.0;
      else if(p_4<-1.0) p_4=-1.0;
   }
   t_pp1r = acos(p_4);
   t_pp1r = t_pp1r*180.0/pi;

   /* fh_after  = after1    fh_before  = before1 */
   /* fp_after  = after2    fp_before  = before2 */
   /* fh_after  = after1    fh_before  = before1 */
   /* fh1_after = after2    fh1_before = before2 */

   if((case_i==1)||(case_i==2)) {
      after1  = interpo(p,theta,p_1,t_ppr,l);
      after1  = pow(10.0,after1);
      after2  = ini_fp(p_3,l,p_4);
      before1 = f[w][l][u];
      before2 = fp[w1][u1];
   } else {
      after1  = interpo(p,theta,p_1,t_ppr,l);
      after1  = pow(10.0,after1);
      after2  = interpo(p,theta,p_3,t_pp1r,l);
      after2  = pow(10.0,after2);
      before1 = f[w][l][u];
      before2 = f[w1][l][u1];
   }
   f_total = after1*after2-before1*before2;
   f_total = f_total*(pi/3.0);

   sum_total = sum_total+f_total;
} /*end of phi_p*/
gau = wg[up]*sum_total;
```

```
      sum_gau += gau;
   } /*end of \cos(theta_\prime) ==> up*/
   /* printf("\n--------------End gauss1------------\n"); */

   return (A/a)*sum_gau;
}

/* formula 1 */
/*-------------------- p_prime_square ----------------------------*/
double p_pri(p,w,mu,muu,muu1,u,p1,w1,u1,cs,up,sn,phi_1,phi_p)
int w,u,w1,u1,up,phi_1,phi_p;
double p[],mu[],p1[],cs[],sn[],muu,muu1;
{
   double pp1,pp,tmp1,tmp2,tmp3,tmp_a,tmp_b,pP,sn_sn1,PP,cs_1;

   sn_sn1 = sqrt(1.0-muu*muu)*sqrt(1.0-muu1*muu1);
   cs_1 = cos(phi_1*pi/3.0);
   pP = 0.5*(p[w]*p[w]-p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1));
   PP = (p[w]*p[w]-2.0*p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1)+
         p1[w1]*p1[w1])/4.0;
   pp = p[w]*p[w];
   tmp_a = pp*PP;
   tmp_b = pP*pP;

   tmp1 = 2.0*pP*(1.0-cs[up]);
   tmp2 = 2.0*PP*(1.0-cs[up]);
   if ((tmp_a-tmp_b)<0.0) {
      tmp3=0.0;
   } else {
      tmp3 = 2.0*sqrt(tmp_a-tmp_b)*sn[up]*cos(phi_p*pi/3.0);
   }
   pp1 = pp-tmp1+tmp2+tmp3;
   pp1 = sqrt(pp1);

   return pp1;
}

/* formula 2 */
/*--------------------- p_prime_r --------------------------------*/
double p_r(p,w,mu,muu,muu1,u,p1,w1,u1,cs,up,sn,phi_1,phi_p)
int w,u,w1,u1,up,phi_1,phi_p;
double p[],mu[],p1[],cs[],sn[],muu,muu1;
{
   double sn_sn1,cs_1,PP,pp,pP,Pr,pr,a,b,c;
   double cpr,cpr1,cpr2,tmp1,tmp2,tmp3,tmp4,pp2;

   sn_sn1 = sqrt(1.0-muu*muu)*sqrt(1.0-muu1*muu1);
   cs_1 = cos(phi_1*pi/3.0);
   PP = (p[w]*p[w]-2.0*p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1)+
         p1[w1]*p1[w1])/4.0;
   pP = 0.5*(p[w]*p[w]-p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1));
   Pr = 0.5*(p[w]*muu-p1[w1]*muu1); /* P1r = -Pr */
   pr = p[w]*muu;
   pp = p[w]*p[w];

   a = PP*pr;
   b = pP*Pr;
   cpr1 = sqrt(pp*PP-pP*pP);
   cpr2 = sqrt(PP-Pr*Pr);
```

```
   /* c = (a-b)/(cpr1*cpr2); */
   tmp1 = 0.5*(p[w]*muu+p1[w1]*muu1);
   tmp2 = Pr*cs[up];
   if ((PP-Pr*Pr)<0.0) {
      tmp3=0;
   } else {
      tmp3 = sqrt(PP-Pr*Pr);
   }

   if(phi_1==0 || phi_1==3 || phi_1==6) {
      /* printf("\ncase1"); */
      cpr = 1.0;
      tmp4 = sn[up]*cos(phi_p*pi/3.0);
   } else if(cpr1==0.0 && cpr2!=0.0) {
      /* printf("\ncase2"); */
      tmp4 = sn[up]*cos(phi_p*pi/3.0);
   } else if(cpr2==0.0) {
      /* printf("\ncase3"); */
      tmp4 = 0.0;
   } else {
      /* printf("\ncase4"); */
      cpr = (PP*pr-pP*Pr)/(cpr1*cpr2);
      if((1.0-cpr*cpr)<0) {
         tmp4 = sn[up]*(cpr*cos(phi_p*pi/3.0));
      } else {
         tmp4 = sn[up]*(cpr*cos(phi_p*pi/3.0)-sin(phi_p*pi/3.0)
                *sqrt(1.0-cpr*cpr));
      }
   }
   pp2 = tmp1+tmp2+tmp3*tmp4;

   return pp2;
}

/* formula 3 */
/*-------------------- p1_prime_square ---------------------------*/
double p1_pri(p,w,mu,muu,muu1,u,u1,p1,w1,cs,up,sn,phi_1,phi_p)
int u,u1,w,w1,up,phi_1,phi_p;
double p[],mu[],p1[],cs[],sn[],muu,muu1;
{
   double pp3;
   double p1p1, tmp1, tmp2, tmp3, t_pp1;
   double tmp_a, tmp_b, p1P1, cs_1, sn_sn1, P1P1;

   sn_sn1 = sqrt(1-muu*muu)*sqrt(1-muu1*muu1);
   cs_1 = cos(phi_1*pi/3.0);
   P1P1 = (p[w]*p[w]-2.0*p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1)+
           p1[w1]*p1[w1])/4.0;
   p1P1 = 0.5*(p1[w1]*p1[w1]-p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1));
   tmp_a = p1[w1]*p1[w1]*P1P1;
   tmp_b = p1P1*p1P1;

   p1p1 = p1[w1]*p1[w1];
   tmp1 = 2.0*p1P1*(1.0-cs[up]);
   tmp2 = 2.0*P1P1*(1.0-cs[up]);
   if((tmp_a-tmp_b)<0.0) {
      tmp3=0.0;
   } else {
      tmp3 = 2.0*sqrt(tmp_a-tmp_b)*sn[up]*cos(phi_p*pi/3.0);
```

```
   }
   pp3 = p1p1-tmp1+tmp2-tmp3;
   pp3 = sqrt(pp3);

   return pp3;
}

/* formula 4 */
/*--------------------- p1_prime_r -------------------------------*/
double p1_r(p,w,mu,muu,muu1,u,p1,w1,u1,cs,up,sn,phi_1,phi_p)
int w,u,w1,u1,up,phi_1,phi_p;
double p[],mu[],p1[],cs[],sn[],muu,muu1;
{
   double sn_sn1,cs_1,PP,pp,pP,Pr,pr,p1P1,p1r,P1P1,P1r,p1p1,a,b;
   double cpr,cpr1,cpr2,tmp1,tmp2,tmp3,tmp4,pp4;

   sn_sn1 = sqrt(1.0-muu*muu)*sqrt(1.0-muu1*muu1);
   cs_1 = cos(phi_1*pi/3.0);
   P1P1 = (p[w]*p[w]-2.0*p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1)+
           p1[w1]*p1[w1])/4.0;
   p1P1 = 0.5*(p1[w1]*p1[w1]-p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1));
   P1r = 0.5*(p1[w1]*muu1-p[w]*muu); /* P1r = -Pr */
   p1r = p1[w1]*muu1;
   p1p1 = p1[w1]*p1[w1];
   PP = P1P1;
   pp = p[w]*p[w];
   pP = 0.5*(p[w]*p[w]-p[w]*p1[w1]*(sn_sn1*cs_1+muu*muu1));
   pr = p[w]*muu;
   Pr = 0.5*(p[w]*muu-p1[w1]*muu1);

   a = PP*pr;
   b = pP*Pr;
   cpr1 = sqrt(pp*PP-pP*pP);
   cpr2 = sqrt(PP-Pr*Pr);
   /* c = (a-b)/(cpr1*cpr2); */
   tmp1 = 0.5*(p[w]*muu+p1[w1]*muu1);
   tmp2 = P1r*cs[up];
   if((P1P1-P1r*P1r)<0.0) {
      tmp3=0;
   } else {
      tmp3 = sqrt(P1P1-P1r*P1r);
   }
   if(phi_1==0 || phi_1==3 || phi_1==6) {
      /* printf("\ncase1"); */
      cpr  = 1.0;
      tmp4 = (-1)*sn[up]*cos(phi_p*pi/3.0);
   } else if(cpr1==0.0 && cpr2!=0.0) {
      /* printf("\ncase2"); */
      tmp4 = sn[up]*cos(phi_p*pi/3.0);
   } else if(cpr2==0.0) {
     /*printf("\ncase3");*/
      tmp4 = 0.0;
   } else {
      /*printf("\ncase4");*/
      cpr = (PP*pr-pP*Pr)/(cpr1*cpr2);
      if((1.0-cpr*cpr)<0) {
         tmp4 = sn[up]*(cpr*cos(phi_p*pi/3.0));
      } else {
         tmp4 = sn[up]*(sin(phi_p*pi/3.0)*sqrt(1.0-cpr*cpr)-
```

```
                cpr*cos(phi_p*pi/3.0));
      }
   }
   pp4 = tmp1+tmp2+tmp3*tmp4;

   return pp4;
}

/*-------------------- volume term ---------------------------------*/
double volume(p,theta,w,u)
int    w,u;
double p[],theta[];
{
   double a,result;

   a  = 3136.125;
   /*---------------- w=0 ------------------------*/
   if(w==0) {
      result = (4.0/3.0)*pi*pow(p[1]/2.0,3);
   /*---------------- 1<=w<=12 -------------------*/
   } else if(w>=1 && w<=12) {
      if(u==0) {
         result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
                   pow(((p[w]+p[w-1])/2.0),3))*fabs(cos((theta[u+5]
                   *pi/180.0)/2.0)-1.0);
      } else if(u==1) {
         result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
                  pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(((theta[u+5]
                  +theta[u+4])*pi/180.0)/2.0)-cos(((theta[u+4]+
                  theta[0])*pi/180.0)/2.0));
      } else if((u>1)&&(u<=8)) {
         result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
                  pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(((theta[u+5]
                  +theta[u+4])*pi/180.0)/2.0)-cos(((theta[u+4]+
                  theta[u+3])*pi/180.0)/2.0));
      } else if(u==9) {
         result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
                  pow(((p[w]+p[w-1])/2.0),3))*fabs(-1.0-
                  cos(((theta[u+4]+theta[u+3])*pi/180.0)/2.0));
      }
   /*---------------- w=13 -----------------------*/
   } else if(w==13) {
      if(u==0) {
         result = (2.0/3.0)*pi*(pow(315*a,3)
                   -pow(((p[w]+p[w-1])/2.0),3))*fabs(cos((theta[u+5]
                   *pi/180.0)/2.0)-1.0);
      } else if(u==1) {
         result = (2.0/3.0)*pi*(pow(((p[19]+p[w])/2.0),3)-
                   pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(((theta[u+5]
                   +theta[u+4])*pi/180.0)/2.0)-cos(((theta[u+4]
                   +theta[0])*pi/180.0)/2.0) );
      } else if((u>1)&&(u<=8)) {
         result = (2.0/3.0)*pi*(pow(((p[19]+p[w])/2.0),3)-
                  pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(((theta[u+5]
                  +theta[u+4])*pi/180.0)/2.0)-cos(((theta[u+4]
                  +theta[u+3])*pi/180.0)/2.0));
      } else if(u==9) {
         result = (2.0/3.0)*pi*( pow(((p[19]+p[w])/2.0),3)-
                  pow(((p[w]+p[w-1])/2.0),3))*fabs(-1.0-
```

```
                cos(((theta[u+4]+theta[u+3])*pi/180.0)/2.0));
   }
/*---------------- 14<=w=17 ----------------------*/
} else if(w>=14 && w<=17) {
   if(u==0) {
      result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
               pow(((p[w]+p[w-1])/2.0),3))*fabs(cos((theta[u+1]
               *pi/180.0)/2.0)-1.0);
   } else if(u>=1 && u<=3)  {
      result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
               pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(((theta[u+1]
               +theta[u])*pi/180.0)/2.0)-cos(((theta[u]+
               theta[u-1])*pi/180.0)/2.0));
   } else if(u==4) {
      result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
               pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(7.5*pi/180.0)
               -cos(((theta[u]+theta[u-1])*pi/180.0)/2.0));
   }
/*---------------- w=18 ----------------------*/
} else if(w==18) {
   if(u==0) {
      result = (2.0/3.0)*pi*(pow(397.5*a,3)-pow(387.5*a,3))
               *fabs(cos((theta[u+1]*pi/180.0)/2.0)-1.0);
   } else if(u>=1 && u<=3) {
      result = (2.0/3.0)*pi*(pow(400*a,3)-pow(387.5*a,3))
               *fabs(cos(((theta[u+1]+theta[u])*pi/180.0)/2.0)
               -cos(((theta[u]+theta[u-1])*pi/180.0)/2.0));
   } else if(u==4) {
      result = (2.0/3.0)*pi*(pow(400*a,3)-pow(387.5*a,3))
               *fabs(cos(7.5*pi/180.0)-cos(((theta[u]+
               theta[u-1])*pi/180.0)/2.0));
   }
/*---------------- w=19 ----------------------*/
} else if(w==19) {
   if(u==0) {
      result = (2.0/3.0)*pi*(pow(405*a,3)-pow(397.5*a,3))
               *fabs(cos(0.5*pi/180.0)-1.0);
   } else if(u==1) {
      result = (2.0/3.0)*pi*(pow(((p[24]+p[w])/2.0),3)-
               pow(((p[w]+p[13])/2.0),3))*fabs(cos(((theta[u+5]
               +theta[u+4])*pi/180.0)/2.0)-cos(((theta[u+4]+
               theta[0])*pi/180.0)/2.0));
   } else if((u>1)&&(u<=8)) {
      result = (2.0/3.0)*pi*(pow(((p[24]+p[w])/2.0),3)-
               pow(((p[w]+p[13])/2.0),3))*fabs(cos(((theta[u+5]
               +theta[u+4])*pi/180.0)/2.0)-cos(((theta[u+4]+
               theta[u+3])*pi/180.0)/2.0));
   } else if(u==9) {
      result = (2.0/3.0)*pi*(pow(((p[24]+p[w])/2.0),3)-
                pow(((p[w]+p[13])/2.0),3))*fabs(-1.0-
               cos(((theta[u+4]+theta[u+3])*pi/180.0)/2.0));
   }
/*---------------- w=20 ----------------------*/
} else if(w==20) {
   if(u==0) {
      result = (2.0/3.0)*pi*(pow(415*a,3)-pow(405*a,3))
               *fabs(cos((theta[u+1]*pi/180.0)/2.0)-1.0 );
   } else if(u>=1 && u<=3) {
      result = (2.0/3.0)*pi*(pow(415*a,3)-pow(400*a,3))
```

```
                *fabs(cos(((theta[u+1]+theta[u])*pi/180.0)/2.0)
                -cos(((theta[u]+theta[u-1])*pi/180.0)/2.0));
      } else if(u==4) {
         result = (2.0/3.0)*pi*(pow(415*a,3)-pow(400*a,3))
                *fabs(cos(7.5*pi/180.0)-cos(((theta[u]
                +theta[u-1])*pi/180.0)/2.0));
      }
   /*---------------- 21<=w<=22 ----------------------*/
   } else if(w>=21 && w<=22) {
      if(u==0) {
         result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)
                -pow(((p[w]+p[w-1])/2.0),3))*fabs(cos((theta[u+1]
                *pi/180.0)/2.0)-1.0);
      } else if(u>=1 && u<=3)  {
         result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
                pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(((theta[u+1]
                +theta[u])*pi/180.0)/2.0)-cos(((theta[u]+
                theta[u-1])*pi/180.0)/2.0) );
      } else if(u==4) {
         result = (2.0/3.0)*pi*(pow(((p[w+1]+p[w])/2.0),3)-
                pow(((p[w]+p[w-1])/2.0),3))*fabs(cos(7.5*pi/180.0)
                -cos(((theta[u]+theta[u-1])*pi/180.0)/2.0));
      }
   /*---------------- w=23 ----------------------*/
   } else if ( w==23 ) {
      if(u==0) {
         result = (2.0/3.0)*pi*(pow(460*a,3)-pow(440*a,3))
                *fabs(cos((theta[u+1]*pi/180.0)/2.0)-1.0);
      } else if(u>=1 && u<=3) {
         result = (2.0/3.0)*pi*(pow(460*a,3)-pow(440*a,3))
                *fabs(cos(((theta[u+1]+theta[u])*pi/180.0)/2.0)
                -cos(((theta[u]+theta[u-1])*pi/180.0)/2.0));
      } else if(u==4) {
         result = (2.0/3.0)*pi*(pow(460*a,3)-pow(440*a,3))
                *fabs(cos(7.5*pi/180.0)-cos(((theta[u]+theta[u-1])
                *pi/180.0)/2.0));
      }
   /*---------------- w=24 ----------------------*/
   } else if(w==24) {
      if(u==0) {
         result = (2.0/3.0)*pi*(pow(550*a,3)-pow(460*a,3))
                *fabs(cos((theta[u+5]*pi/180.0)/2.0)-1.0);
      } else if(u==1) {
         result = (2.0/3.0)*pi*(pow(550*a,3) - pow(450*a,3) )
                *fabs( cos(((theta[u+5]+theta[u+4])*pi/180.0)/2.0)
                -cos(((theta[u+4]+theta[0])*pi/180.0)/2.0));
      } else if((u>1)&&(u<=8)) {
         result = (2.0/3.0)*pi*(pow(550*a,3)-pow(450*a,3))
                *fabs(cos(((theta[u+5]+theta[u+4])*pi/180.0)/2.0)
                -cos(((theta[u+4]+theta[u+3])*pi/180.0)/2.0));
      } else if(u==9) {
         result = (2.0/3.0)*pi*(pow(550*a,3)-pow(450*a,3))
                *fabs(-1.0-cos(((theta[u+4]+theta[u+3])*pi/180.0)
                /2.0));
      }
   }
   return result;
}
```

```
/*------------------- ini_fp ---------------------*/
double ini_fp(p_3,l,p_4)
int l;
double p_3,p_4;
{
   double arg,f_tmp;

   arg = -(p_3*p_3-(2.0*p_4*p_3*px[l])+(px[l]*px[l]))/(2.0*M*K*
         temp[l]);
   if((arg>-30.0) && (arg < 30.0)) {
      f_tmp = dens[l]/pow(2.0*pi*M*K*temp[l],1.5)*exp(arg);
   } else {
      if(arg <= -30.0) {
         f_tmp = 0.0;
      } else {
         nrerror("\nIn ini_fp(): bad fp\n");
      }
   }
   return f_tmp;
}
```

Curriculum Vitae

Panita Boonma

1975 Born: Aug, 5^{th} 1975 in Bangkok, THAILAND.

Father: Payon Boonma.

Mother: Somruk Boonma.

1993-1996 Bachelor of Science (Mathematics),

Chulalongkorn University, Bangkok, THAILAND.