การตัดกันหลายครั้งของสนามแม่เหล็กสุ่มกับคลื่นกระแทก และการเร่งอนุภาค ในกรณีสนามแม่เหล็กทำมุมเกือบ 90 องศากับเวกเตอร์ปกติของคลื่นกระแทก

นายจตุรงค์ สุคนธชาติ

หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาฟิสิกส์ ภาควิชาฟิสิกส์
บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2542

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ISBN 974-333-115-8

ลิขสิทธิ์ของบัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย

I 18749902

MULTIPLE MAGNETIC FIELD-SHOCK CROSSINGS AND PARTICLE ACCELERATION AT QUASI-PERPENDICULAR SHOCKS

Mr. Jaturong Sukonthachat

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Physics
Department of Physics
Graduate School
Chulalongkorn University
Academic Year 1999

ISBN 974-333-115-8

Thesis Tittle Multiple Magnetic Field-Shock Crossings and Particle Acceleration at Quasi-Perpendicular Shocks
By Mr. Jaturong Sukonthachat
Department Physics
Thesis Advisor Associate Professor David Ruffolo, Ph.D.

Accepted by the Graduate School, Chulalongkorn University in Partial Fulfillment of the Requirement for the Degree of Master of Science

Suchada Kiranadana .... Dean of Graduate School
(Associate Professor Suchada Kiranadana, Ph.D.)

Thesis Committee

Thaworn Souttipongse .... Chairman
(Associate Professor Thaworn Souttipongse, M.Sc.)

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

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

.... Member
(Pong Songpongs, Ph.D.)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

จตุรงค์ สุคนธชาติ: การตัดกันหลายครั้งของสนามแม่เหล็กสุ่มกับคลื่นกระแทก และการเร่งอนุภาคในกรณีสนามแม่เหล็กทำมุมเกือบ 90 องศากับเวกเตอร์ปกติของคลื่นกระแทก (MULTIPLE MAGNETIC FIELD-SHOCK CROSSINGS AND PARTICLE ACCELERATION AT QUASI-PERPENDICULAR SHOCKS) อ. ที่ปรึกษา : รศ. ดร. เดวิด รูฟโฟโล, 114 หน้า. ISBN 974-333-115-8

ความสำคัญของการฟุ้งของอนุภาคในทิศตั้งฉากจากสนามแม่เหล็กเฉลี่ย ในกรณีของการเร่งอนุภาคที่สนามแม่เหล็กเฉลี่ยทำมุมเกือบ 90 องศากับเวกเตอร์ปกติของคลื่นกระแทกนั้นเป็นที่รู้จักแล้ว การฟุ้งดังกล่าวสามารถอธิบายโดยการเดินสุ่มของเส้นสนามแม่เหล็ก โดยอนุภาคโคจรรอบเส้นสนามแม่เหล็กและฟุ้งไปกลับตามแนวของเส้นสนามแม่เหล็กนี้ด้วย ในการจำลองเราเริ่มโดยใช้แบบจำลองความแปรปรวนใน 1 มิติ คือ แบบจำลองสแลบ ซึ่งความแปรปรวนของสนามแม่เหล็กขึ้นอยู่กับทิศทางในแนวแกน z เท่านั้น จากนั้นใช้แบบจำลองความแปรปรวนของสนามแม่เหล็กใน 3 มิติ คือ แบบจำลอง 2 มิติ ซึ่งความแปรปรวนขึ้นอยู่กับทิศทางในแนวแกน x และแนวแกน y บวกกับผลของแบบจำลองสแลบ โดยใช้อัตราส่วนของแบบจำลอง 2 มิติ 80% และแแบบจำลองสแลบ 20% เนื่องจากว่าอัตราส่วนนี้สามารถอธิบายความแปรปรวนของสนามแม่เหล็กของตัวกลางระหว่างดาวเคราะห์ได้เป็นอย่างดี ถ้าเส้นสนามแม่เหล็กที่แปรปรวนนั้นข้ามคลื่นกระแทก N^2 ครั้ง โดยที่มีระยะห่าง L มากกว่าระยะอิสระเฉลี่ยของอนุภาค $\lambda_{/\!/}$ แล้วอนุภาคจะต้องข้ามคลื่นกระแทกโดยเฉลี่ย N^2 ครั้งจึงจะเป็นอิสระ การข้ามคลื่นกระแทกนี้จะทำให้อนุภาคถูกเพิ่มพลังงาน และระยะการเคลื่อนที่ลอยเลื่อนขึ้นโดยกระบวนการเร่งอนุภาคแบบลอยเลื่อน (shock drift acceleration) ในการจำลองเราได้พิสูจน์ว่ามีการข้ามคลื่นกระแทกหลายครั้งสำหรับค่าที่สมเหตุสมผลของ $(\delta B^2 / B_0)^2$ ใกล้บริเวณที่เกิดคลื่นกระแทก และได้วัดผลการกระจายตัวทางสถิติของ N, θ (มุมระหว่างเส้นสนามแม่เหล็กกับเวกเตอร์ปกติของคลื่นกระแทก) และ L สำหรับการจำลองเส้นสนามแม่เหล็กสุ่ม สำหรับกรณีพิเศษคือกรณีของคลื่นกระแทกเทอร์มิเนชั่นของระบบสุริยะ (solar wind termination shock) กลไกนี้อาจช่วยในการอธิบายการเคลื่อนที่แบบดริฟเป็นระยะทางมากจากแนวเส้นศูนย์สูตรของระบบสุริยะถึงขั้ว หรือในทางกลับกัน

จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา ......ฟิสิกส์...... ลายมือชื่อนิสิต ....
สาขาวิชา ......ฟิสิกส์...... ลายมือชื่ออาจารย์ที่ปรึกษา ......เดวิด รูฟโฟโล......
ปีการศึกษา ......2542...... ลายมือชื่ออาจารย์ที่ปรึกษาร่วม ....

#4072449123 MAJOR PHYSICS
KEY WORD: SHOCK / TERMINATION SHOCK / TURBULENT MAGNETIC FIELD

JATURONG SUKONTHACHAT: MULTIPLE MAGNETIC FIELD-SHOCK CROSSINGS AND PARTICLE ACCELERATION AT QUASI-PERPENDICULAR SHOCK. THESIS ADVISOR: ASSOC. PROF. DAVID RUFFOLO, Ph.D. 114 pp.
ISBN 974-333-115-8

The importance of accounting for diffusion perpendicular to the mean magnetic field during nearly perpendicular shock acceleration is well documented. Here we note that perpendicular diffusion is typically envisioned as due to the random walk of field lines, with particle guiding centers closely tied to and diffusing back and forth along the field. We first simulate one-dimensional magnetic field turbulence by using a slab model, in which the turbulence depends on the z direction only. Then we simulate three-dimensional magnetic field turbulence by superimposing two types of turbulence, a 2D model depending on x and y, and the slab model, in the admixture of 80% 2D turbulence and 20% slab turbulence, which provides a good fit to interplanetary turbulence. If these turbulent field lines can cross and recross a shock at N magnetic field-shock crossing that are separated by distances $L > \lambda_{\parallel}$, the scattering mean free path, a particle will cross the shock an average of N^2 times before escaping. This could increase the total shock-drift distance and energization of particles. We have verified that multiple field-shock crossing do occur for reasonable values of $(\delta B / B_0)^2$ near the shock, and have measured the distribution of N, θ (the angle at which the field crosses the shock), and L for 1000 simulated random magnetic fields. For the special case of the solar wind termination shock, this mechanism may help to explain the observationally inferred drift of anomalous cosmic rays (ACR) over much of the distance from the heliospheric equator to the poles or vice-versa.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาควิชา.....ฟิสิกส์..... ลายมือชื่อนิสิต.....................

สาขาวิชา.....ฟิสิกส์..... ลายมือชื่ออาจารย์ที่ปรึกษา.....David Ruffolo.....

ปีการศึกษา.....2542..... ลายมือชื่ออาจารย์ที่ปรึกษาร่วม.....................

Acknowledgements

I would like to express my sincere gratitude to my thesis adviser, Dr. David Ruffolo, for his encouragement not only in his useful comments but also in his researcher's sipirit. I also wishes to express my thanks to a Basic Research Grant from the Thailand Research Fund for supporting this research.

I have to thank to Police Captain Dr. Wiwat Sidhisoradej, Mr. Songklod Riyavong and Mr. Paisan Tooprakai for their helpful suggestions and assistance with computer operating systems.

I would like to thank to Miss Thiranee Khumlumlert, Miss Supaporn Kamklad and Mr. Kobchai Tayanasanti for thier kindness and help.

Finally, I would like to dedicate the thesis to my parents who are working hard for their children. Special thanks go to my sisters for their attention.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Table of Contents

Abstract in Thai iv

Abstract in English v

Acknowledgements vi

Table of Contents vii

List of Figures x

List of Tables xvii

Chapter I Introduction 1

1.1 Thesis Purpose and Scope 1

1.2 Overview of the Thesis 3

Chapter II Theoretical Background 4

2.1 The Discovery of Cosmic Rays and the Power Law Spectrum 4

2.2 Shock Waves, the Solar Wind Termination Shock, and Anomalous Cosmic Rays 6

2.3 The Dynamics of Charged Particles in Magnetic Fields 7

2.3.1 A uniform, static magnetic field 8

2.3.2 A non-uniform, static magnetic field 10

2.3.2.1 Gradient drift 11

2.3.2.2 Curvature drift 11

2.4 Shock Drift Acceleration 13

2.5 Order of Magnitude Calculations of Shock Drift Acceleration

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

at the Solar Wind Termination Shock 15
2.5.1 Conclusions of the order of magnitude calculations 22
2.6 Multiple Magnetic Field-Shock Crossings 22

Chapter III Model of Turbulent Magnetic Fields 27
3.1 Slab Model 27
3.2 2D+Slab Model 31

Chapter IV Simulation Technique and Their Testing . 35
4.1 Overview of Numerical Simulations 35
4.2 Inverse Fourier Transform 36
4.3 Random Phase Approximation 37
4.4 Euler's Method and Interpolation 38
4.4.1 Euler's method 38
4.4.2 Linear interpolation 38
4.4.3 Bilinear interpolation 39
4.5 Trajectories of Magnetic Field Lines and Field-Shock Crossings .. 40
4.6 Simulation Testing 43
4.6.1 Fourier transform testing 43
4.6.2 Total turbulence and correlation length testing 44
4.6.3 Crossings of a shock and statistical data testing 46

Chapter V Statistical Results 49
5.1 One-Dimensional Statistical Results 50
5.2 Three-Dimensional Statistical Results 60

Chapter VI Discussion and Conclusions 67

6.1 Discussion of One-Dimensional Results 67

6.2 Discussion of Three-Dimensional Results 68

6.3 Comparison of the Upstream Crossings Angle Distributions 69

6.4 Conclusions 70

References 72

Appendix A One-Dimensional Random Walk 74

Appendix B Fast Fourier Transform 78

Appendix C Random Number Generator 89

Appendix D Computer Program 91

Curriculum Vitae 114

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

List of Figures

Figure Page

Figure: 2.1.1 Energy spectra for protons, helium, carbon and iron nuclei as a function of their kinetic energy per nucleon (Longair, 1997, after Simpson, 1983). 6

Figure: 2.2.1 Diagram of the heliosphere with the particle populations indicated schematically which influence termination shock structure: galactic cosmic rays (large dots), anomalous cosmic rays (small dots), interstellar neutral gas (solid trajectories), and interstellar pickup ions (dashed trajectories). The inner circle indicates the transition at which pickup ions are injected at the termination shock to become low-energy anomalous cosmic rays (Lee 1996). 8

Figure: 2.2.2 Structure of the heliosphere when interstellar neutral hydrogen is neglected altogether. Three boundaries are clearly discernable: the outermost bow shock, which decelerates an assumed supersonic interstellar (galactic) wind; the heliopause, a contact discontinuity that separates the interstellar wind and solar wind; and the solar wind termination shock (L. Pauls, private communication). 9

Figure: 2.3.1 Illustration of the gradient drift (Jackson 1975). 11

Figure: 2.3.2 Illustration of curvature drift (Tuska 1990). 12

Figure: 2.4.1 Drift acceleration of charged particles at the shock front (Decker and Vlahos, 1985) 13.

Figure: 2.4.2 Momentum-changing (Δp) mechanism of shock drift acceleration for a quasi-perpendicular shock. The energy change is (ΔE) $\approx$ (Δp)c (Terasawa 1995). 14

Figure: 2.5.1 Schematic spiral magnetic field, B_φ, and the radial speed vector of the solar wind $u\hat{r}$ (Khumlumlert 1996). 16

Figure: 2.5.2 Illustration of the directions of $\hat{r}$, $\hat{z}$, and ψ (Ruffolo 1995). 18

Figure: 2.5.3 a) Drift distance due to a single shock encounter (D_{drift}) of anomalous cosmic rays of kinetic energy 1 MeV. b) The observationally inferred large drift distance of the anomalous cosmic rays. 19

Figure: 2.6.1 Top view of the solar system. The random magnetic field can cross the quasi-perpendicular solar wind termination shock multiple times. 24

Figure: 2.6.2 A magnetic sample field line that crosses a shock (diagonal line) multiple times.Note the greatly expanded verticle scale. b) Schematic of the above multiple magnetic field-shock crossings, and boundary conditions for the random walk of particles along $\vec{B}$. 25

Figure: 3.1.1 Inthe slab model of turbulence, the turbulent magnetic field depends on the k_z component only. 28

Figure: 3.2.1 Inthe 2D model of turbulence, the turbulent magnetic field depends on the k_{bot} component only. 32

Figure: 4.4.1 Linear interpolation (depends on i only) at point P between two points $F(x_i)$ and $F(x_{i+1})$. 39

Figure: 4.4.2 Bilinear interpolation at point S. 40

Figure: 4.5.1 Three dimensional trajectories from the slab model of turbulence. 41

Figure: 4.5.2 Three dimensional trajectories from the 2D + slab model of turbulence. 42

Figure: 4.6.1 (a) Distribution of $R_{xx}(z)/\delta B_x^2$ vs. z. (b) inverse Fourier transform of P_{xx} vs. z (Mathematica plot) 45
Both of them have similar distributions, serving as a successful test of the program. 42

Figure: 4.6.2 Crossings between random magnetic field trajectories and a shock. We can see the total number of crossings, angles of the crossings, and
the distance between adjacent
crossing points. 47

Figure: 4.6.3 Calculation table for the figure above, showing for each crossing the grid point (i), x and y coordinates (xcoor[] and ycoor[]), number of crossings (Cross), upstream crossing angle (alpha[]), and distance along the shock (L[]) We can use these data to recheck all of the calculation values reported by the program for confirmation. 47

Figure: 5.1.1 Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$. 51

Figure: 5.1.2 Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$. 51

Figure: 5.1.3 Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$. 52

Figure: 5.1.4 Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$. 52

Figure: 5.1.5 Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 5 \times 10^{-2}$. 53

Figure: 5.1.6 Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-1}$. 53

Figure: 5.1.7 Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$. 54

Figure: 5.1.8 Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$. 54

Figure: 5.1.9 Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$. 55

Figure: 5.1.10 Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$. 55

Figure: 5.1.11 Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 5 \times 10^{-2}$. 56

Figure: 5.1.12 Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-1}$. 56

Figure: 5.1.13 Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$. 57

Figure: 5.1.14 Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$. 57

Figure: 5.1.15 Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$. 58

Figure: 5.1.16 Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$. 58

Figure: 5.1.17 Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 5 \times 10^{-2}$. 59

Figure: 5.1.18 Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-1}$. 59

Figure: 5.2.1 Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$. 61

Figure: 5.2.2 Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$. 61

Figure: 5.2.3 Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$. 62

Figure: 5.2.4 Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$. 62

Figure: 5.2.5 Histogram of the upstream crossing angle between the field and the shock normal from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$ 63

Figure: 5.2.6 Histogram of the upstream crossing angle between the field and the shock normal from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$ 63

Figure: 5.2.7 Histogram of the upstream crossing angle between the field and the shock normal from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$ 64

Figure: 5.2.8 Histogram of the upstream crossing angle between the field and the shock normal from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$ 64

Figure: 5.2.9 Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$. 65

Figure: 5.2.10 Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$. 65

Figure: 5.2.11 Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$. 66

Figure: 5.2.12 Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$. 66

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

List of Tables

Table Page

Table: 2.1.1 Examples of types of cosmic rays, their sources and their energy ranges. 5

Table: 5.1.1 Parameters of the One-Dimensional Simulations. 50

Table: 5.2.1 Parameters of the Three-Dimensional Simulations. 60

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

Chapter I

Introduction

This chapter provides the overview of this thesis and its purpose and scope. After reading this chapter, readers will obtain a useful summary of each chapter for planning their reading and quickly searching for topics of interest. In the thesis purpose and scope, readers will learn about the proposed ideas in this work and their scope, which will improve understanding and may give new ideas for future work.

1.1 Thesis Purpose and Scope

The purpose of this thesis is to prove that the random magnetic field lines can cross a quasi-perpendicular shock wave multiple times; "quasi-perpendicular" means that the angle between the shock normal vector and the trajectory of a random magnetic field is nearly 90 degrees. After that, we prove that if the random magnetic fields cross a shock with a crossing distance more than the mean free path N times, charged particles will encounter the shock N^2 times before escaping from the shock (Section 2.6). Because we cannot find the random magnetic field crossings of a shock by an analytical method, we must use a numerical method. This means that the computer software and hardware are requirements (and limitations) of this work.

1.2 Overview of the Thesis

In Chapter II, we describe details of cosmic rays and fundamentals of shock waves, which are needed to understand this research. First, we talk about the discovery of cosmic rays, the most important experiment that proved that the cosmic rays come from space, and describe one of the most important characteristics of galactic cosmic rays, the power law spectrum. Then we introduce the concept of a shock wave and in particular the solar wind termination shock, which is an example of an interesting case. After that, we discuss the dynamics of charged particles in magnetic fields, both in a uniform magnetic field and a non-uniform magnetic field. These results give more understanding of the shock drift acceleration, which is explained in Section 2.4. Finally, we calculate the order-of-magnitude parameter values relevant to particle acceleration at the solar wind termination shock and relate them to the multiple magnetic field-shock crossings mechanism (Section 2.6). We discuss the probability of escape after the multiple crossings by using a classical random walk in Appendix A.

We introduce models of turbulent magnetic fields in Chapter III. First, in the slab model we introduce 1D magnetic field turbulence, depending on the mean magnetic field direction (z direction). Then, we discuss a 2D model of turbulence in which the magnetic field turbulence depends on the x and y directions. After that, we discuss a realistic model that explains solar wind turbulence well, the 2D$-$Slab model, by using the previous discussions.

In Chapter IV, we introduce simulation techniques for simulating random magnetic field trajectories and their crossings of a shock. First, we discuss 1D and 2D inverse Fourier transforms (Appendix B), which are used for transforming the slab model and 2D model from wave vector space to position space. Then, we discuss a random phase approximation by using a random number generator (Appendix C) in section 4.3 and describe Euler's method which is used to simulate a random magnetic field and interpolation methods for this simulation, linear and bilinear interpolation. After that, we introduce concepts of multiple magnetic field crossings, including setting up the origin of the simulation, the shock wave plane and crossing conditions. Finally, we provide a section on simulation testing, including inverse Fourier transform testing, total turbulence and correlation length testing, and statistical data testing. All of the simulation and testing programs are shown in Appendix D.

In Chapter V, we show 1D and 3D statistical data that are calculated by setting the correlation length equal to 16 and varying the ratio of the total turbulence over the mean magnetic field to be 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 5×10^{-2}, and 1×10^{-1}. For each of the ratios, we show three simulation histograms for 1000 simulated turbulent magnetic fields: the number of field-shock crossings, the upstream angle between the field and the shock normal angle, and the distance between crossings.

In the last chapter, Chapter VI, we give our discussion and conclusions for the statistical results and use these conclusions to explain the long drift of anomalous cosmic rays from the heliospheric equator to the poles or vice-versa and their energy gain.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Chapter II

Theoretical Background

This chapter provides the important concepts for understanding this research. We start with the discovery of galactic cosmic rays and their power law spectrum. After that, we will describe fundamental concepts of the Sun, solar wind, anomalous cosmic rays, shock waves and termination shock. Then we will explain the concept of shock drift acceleration. Finally, we will introduce an idea of multiple magnetic field-shock crossings to compare with the single crossing idea. All of these concepts will be used in our simulations, which are described in Chapter 4.

2.1 The Discovery of Cosmic Rays and the Power Law Spectrum

The cosmic ray story began in about 1900 when it was found that electroscopes discharged in ordinary air. It was later shown by Rutherford that the leaves of the electroscopes provided a measure of the amount of ionization. However, the big breakthrough came in 1912 when Hess rode in a balloon to 5 km above the ground. The third balloon carried the ionization experiment for measuring the ionization of the atmosphere with increasing altitude. He found the startling result that the average ionization increased with respect to the ionization at sea-level above about 1.5 km. This means that the source of the ionizing radiation must be located above the Earth's atmosphere. Later, people referred to the source of the

หอสมุดกลาง สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ionization of the atmosphere as "cosmic rays." Nowadays people define cosmic rays as energetic particles or γ-rays from space. Examples of cosmic rays and their sources are shown in Table 2.1.1.

Source	Species	Energy
Sun,Shocks	$p, e, n, \alpha, {}^{12}C, {}^{16}O, \ldots$	~100 keV to 50 GeV
Jupiter	e	1 - 25 MeV
Termination Shock	$p, \alpha, {}^{12}C, {}^{16}O, \ldots$	1 - 100 MeV

Table 2.1.1: Examples of types of cosmic rays, their sources and energy ranges.

A striking feature of galactic cosmic rays (GCR), cosmic rays coming from our galaxy, is that their energy spectrum can often be represented by power-law energy distributions, a decrease in the differential flux of particles proportional to their kinetic energy to some power, as illustrated in Fig. 2.1.1. This means that galactic cosmic rays are accelerated to high energies by using the same mechanism in space. Nowadays, we know that the most important mechanism is shock acceleration, which can describe the power-law spectrum of galactic cosmic rays. In 1954, Fermi showed that when cosmic rays cross a shock wave, they can gain energy and produce a power-law spectrum (Fermi, 1954). Shock acceleration is also called 1^{st}-order Fermi acceleration. The differential energy spectrum of cosmic rays is measured at the Earth from observations made from above the Earth's atmosphere. The flux of helium nuclei below about 60 MeV nucleon^{-1} is due to an additional flux of these particles which are known as the anomalous He component (Simpson, 1983).

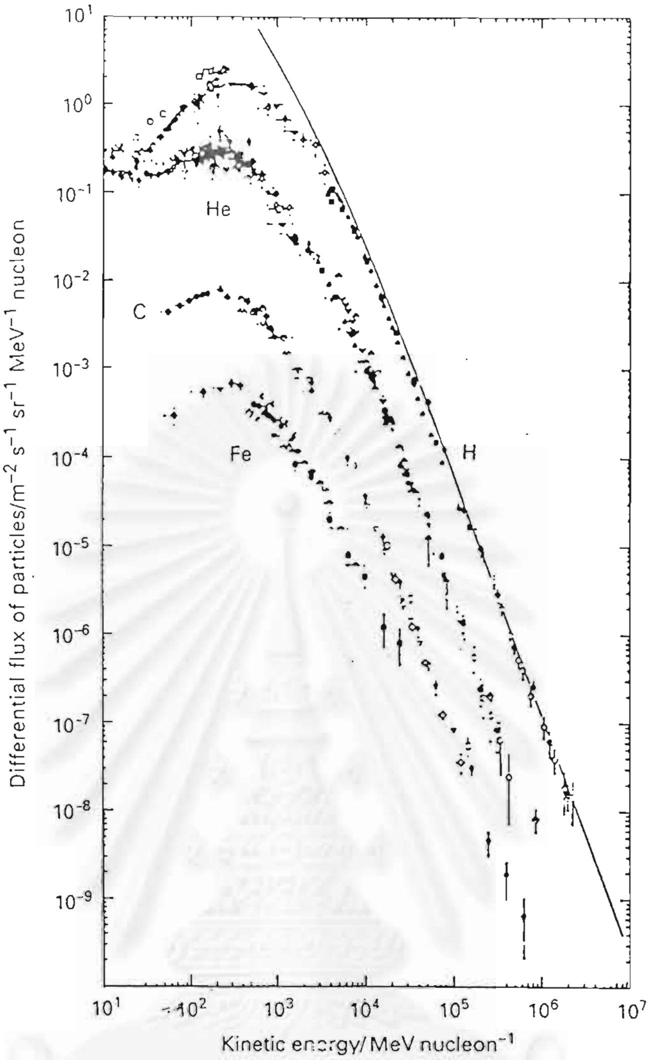

Figure 2.1.1: Energy spectra for protons, helium, carbon and iron nuclei as a function of their kinetic energy per nucleon (Longair, 1997, after Simpson, 1983).

2.2 Shock Waves, the Solar Wind Termination Shock, and Anomalous Cosmic Rays

In section 2.1, we learned that cosmic rays are most commonly accelerated by shock waves. In this section, we will discuss these shock waves. The definition of a shock wave is a discontinuity in fluid parameters (Longair, 1997). The discontinuity is caused by a collision between two fluids which have a supersonic relative velocity or between a supersonic fluid and an obstacle. At supersonic velocities, the forward region of a fluid cannot communicate with the region behind it, whereas for a subsonic flow, such communication would mitigate a collision into a continuous compression. For supersonic flow, when the forward region impacts the shock, the region behind will also collide with the same speed and make a compressional discontinuity in the fluid. Such discontinuities are called shock waves.

The solar termination shock is caused by the interaction between the solar wind, a supersonic plasma flow from the Sun, and the interstellar medium, a slow, denser plasma surrounding the solar system. The overpressure of the interaction region can drive two shocks: a stationary, reverse shock that attempts to propagate against the flow of the wind back toward the star, and a stationary forward shock, though the existence of the latter depends on whether the interstellar flow is supersonic, which is not certain. The structure of the termination shock is shown in Figure 2.2.2. We call the particles accelerated by the solar wind termination shock "anomalous cosmic rays." The identification of the anomalous cosmic rays is from an excess over the galactic spectrum, and because they have low charges, mostly +1 or +2 from charge exchange or ionization mechanisms at the termination shock.

2.3 The Dynamics of Charged Particles in Magnetic Fields

This analysis has applications regarding how the energy spectrum of the cosmic rays is influenced by the solar wind. The results are useful for understanding the effects of the magnetic field on the trajectories of particles.

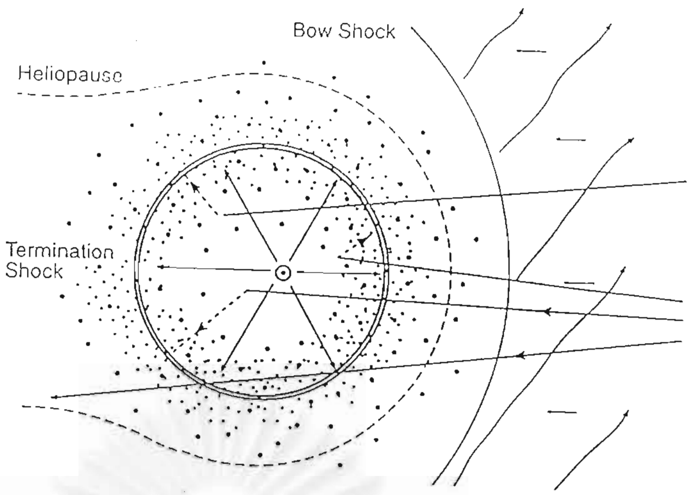

Figure 2.2.1: Diagram of the heliosphere with the particle populations indicated schematically which influence termination shock structure: galactic cosmic rays (large dots), anomalous cosmic rays (small dots), interstellar neutral gas (solid trajectories), and interstellar pickup ions (dashed trajectories). The inner circle indicates the transition at which pickup ions are injected at the termination shock to become low-energy anomalous cosmic rays (Lee 1996).

2.3.1 A uniform, static magnetic field

The equation of motion for a particle of rest mass m, charge Ze and Lorentz factor $\gamma = (1 - v^2/c^2)^{-1/2}$ in a uniform static magnetic field $\vec{B}$ is

$$\vec{F} = \frac{d\vec{p}}{dt} = \frac{d}{dt}(\gamma m\vec{v}) = Ze(\vec{v} \times \vec{B}). \tag{2.3.1}$$

The left hand side can be expanded as follows:

$$m\frac{d}{dt}(\gamma\vec{v}) = \gamma m\vec{a} + m\gamma^3\vec{v}\frac{\vec{v}\cdot\vec{a}}{c^2} \tag{2.3.2}$$

because the Lorentz factor γ can be written $\gamma = (1 - \vec{v}\cdot\vec{v}/c^2)^{-1/2}$. In a magnetic field, the acceleration $\vec{a} = d\vec{v}/dt$ is always perpendicular to $\vec{v}$ and consequently $\vec{v}\cdot\vec{a} = 0$. As a result

$$\gamma m\frac{d}{dt}\vec{v} = Ze(\vec{v} \times \vec{B}). \tag{2.3.3}$$

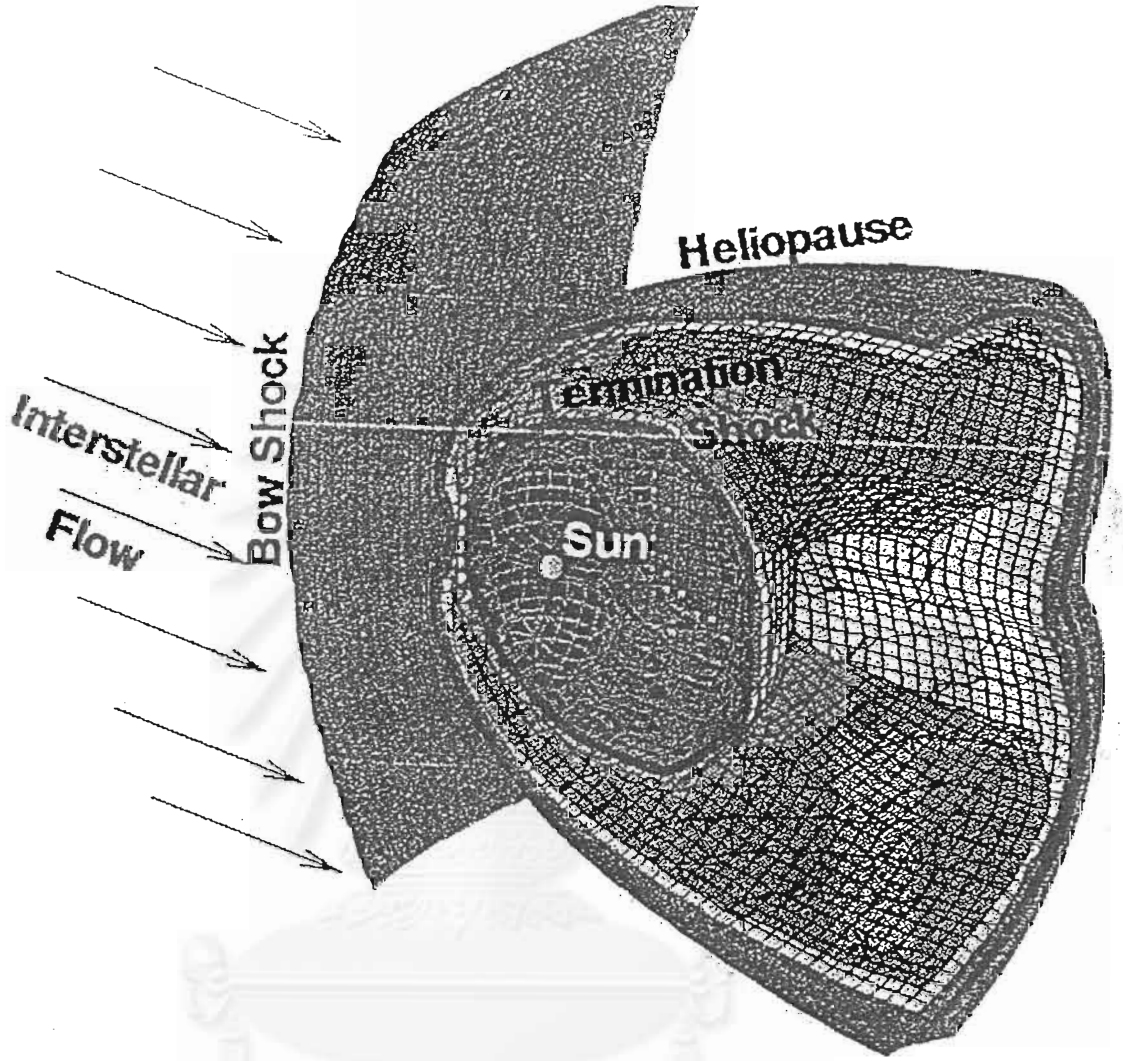

Figure 2.2.2: Structure of the heliosphere when interstellar neutral hydrogen is neglected altogether. Three boundaries are clearly discernable: the outermost bow shock, which decelerates an assumed supersonic interstellar (galactic) wind; the heliopause, a contact discontinuity that separates the interstellar wind and solar wind; and the solar wind termination shock (L. Pauls, private communication).

We split $\vec{v}$ into components parallel and perpendicular to the uniform magnetic field, $v_{||}$ and $v_\perp$, respectively. Thus

$$\frac{d}{dt}v_{||} = 0 \tag{2.3.4}$$

$$\frac{d}{dt}\vec{v}_\perp = \frac{-Ze}{\gamma m}(\vec{B} \times \vec{v}_\perp) = \vec{\omega} \times \vec{v}_\perp, \tag{2.3.5}$$

where $\vec{\omega} = -Ze\vec{B}/(\gamma m)$ can be interpreted as the angular frequency of gyration. Thus the parallel component $v_{||}$ is unchanged, while the perpendicular component $\vec{v}_\perp$ rotates with a frequency ω. Furthermore,

$$v_{||} = v\cos\theta \tag{2.3.6}$$

$$v_\perp = v\sin\theta, \tag{2.3.7}$$

where θ is the pitch angle, i.e., the angle between $\vec{v}$ and $\vec{B}$.

Because the magnetic field is uniform, this constant acceleration perpendicular to the velocity vector results in circular motion of $\vec{v}_\perp$ about the magnetic field as described above. Equating this acceleration to the centrifugal acceleration, we find

$$v_\perp^2/r = ZevB\sin\theta/(\gamma m) \tag{2.3.8}$$

$$r = \gamma mv\sin\theta/(ZeB). \tag{2.3.9}$$

Since the velocity v and the magnitude of the magnetic field B are constant, this equation shows that the particle moves in a spiral path with a constant pitch angle θ.

2.3.2 A non-uniform, static magnetic field

In the case of a non-uniform static magnetic field, there are interesting effects of the gradient and the curvature of the magnetic field. These effects make the guiding centers of particles drift from the mean magnetic field.

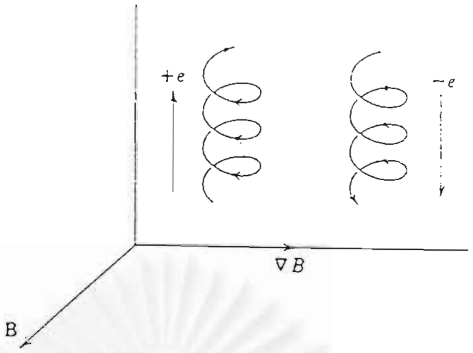

Figure 2.3.1: Illustration of the gradient drift (Jackson 1975).

2.3.2.1 Gradient drift

The gradient drift is the drift of the particle trajectory arising from the perpendicular gradient of the mean magnetic field. This gradient is small compared to the mean magnetic strength. The gradient velocity is shown in the equation below:

$$\vec{v}_g = \frac{a^2\omega_B}{2B^2}(\vec{B} \times \vec{\nabla}_\perp B) \tag{2.3.10}$$

where ω_B is the gyration frequency, $\omega_B = qB/\gamma mc$, and a is the gyroradius of the particle (Tuska 1990). The gradient drift is perpendicular to the magnetic field vector $\vec{B}$ and $\vec{\nabla}B$ as shown in Figure 2.3.1.

2.3.2.2 Curvature drifts

The curvature drift is a systematic motion perpendicular to the magnetic field, due to the curvature of the magnetic field. The curvature of the magnetic field lines induces a drift velocity of particles when the radius of curvature of magnetic lines of force, R, is longer than the gyroradius, a, of the particle. Here $\vec{R}$ points from the center of curvature to the point of interest on the field line. In the

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

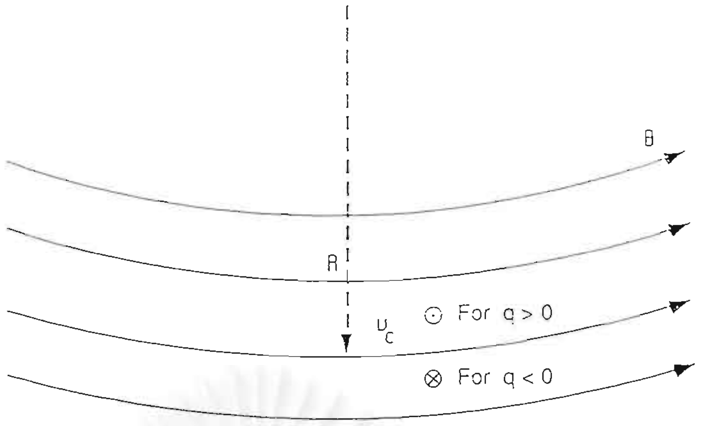

Figure 2.3.2: Illustration of quantities related to the curvature drift (Tuska 1990).

absence of a magnetic field, the guiding center of a particle would tend to follow a straight line tangent to the curving field line. This leads to an effective centrifugal acceleration $v_{\|}^2/R$, which can also be viewed as due to an effective electric field (Jackson 1975):

$$\vec{F_{\text{eff}}} = q\vec{E_{\text{eff}}} = \gamma m\vec{a_c}, \tag{2.3.11}$$

where $\vec{a_c}$ is the centrifugal acceleration. Then

$$\vec{E_{\text{eff}}} = \frac{rm}{q}\left(\frac{v_{\|}^2}{R}\right)\hat{R} = \frac{rm}{q}\left(\frac{\vec{R}}{R^2}\right)v_{\|}^2 \tag{2.3.12}$$

where $\vec{R}$ points from the center of curvature to the point of interest on the field line (Figure 2.3.2). The effective field is always perpendicular to the magnetic field. The drift velocity of a particle moving perpendicular to electric and magnetic fields is

$$\vec{v_d} = \frac{c(\vec{E}\times\vec{B})}{B^2}. \tag{2.3.13}$$

We find the velocity for the curvature drift velocity perpendicular to the magnetic field to be

$$\vec{v_d} = \frac{c(\vec{E_{\text{eff}}}\times\vec{B_0})}{B_o^2} \tag{2.3.14}$$

จุฬาลงกรณ์มหาวิทยาลัย

$$\vec{v_d} \approx \frac{crm}{q} v_{\|}^2 \frac{\vec{R} \times \vec{B_0}}{R^2 B_o^2}. \tag{2.3.15}$$

2.4 Shock Drift Acceleration

In section 2.3, we described some details of the trajectory of a charged particle in a non-uniform magnetic field. In this section, we will discuss the effect of the trajectories of charged particles on their acceleration.

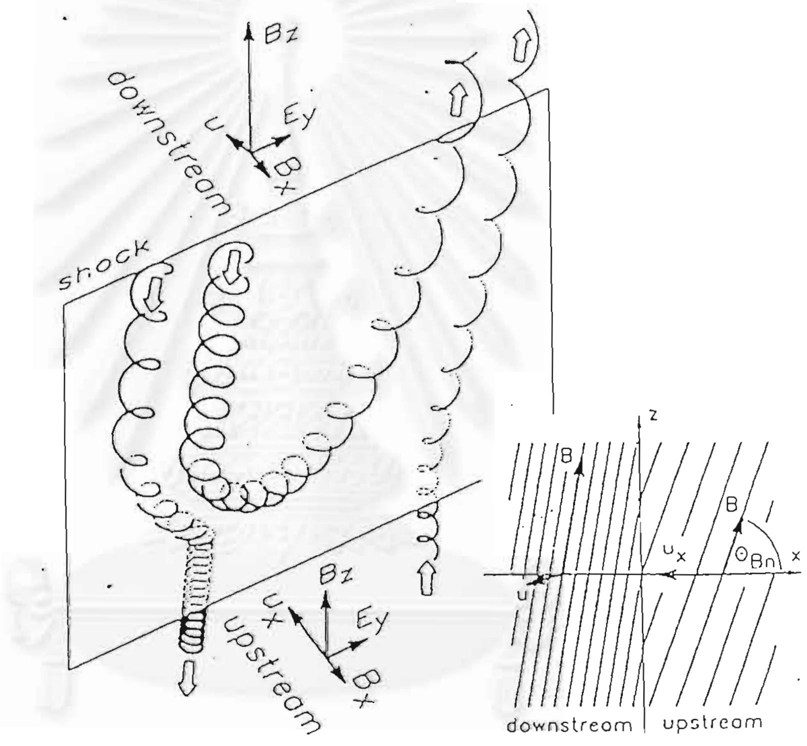

Figure 2.4.1: Drift acceleration of charged particles at the shock front (Decker and Vlahos, 1985).

จุฬาลงกรณ์มหาวิทยาลัย

Only in the special case of a parallel shock, in which $\vec{B}$ is parallel to the shock normal, is the magnetic field the same on either side of the shock. In this case, particles are accelerated by 1^{st}-order Fermi acceleration, compression at the shock front with scattering. Returning to the more general case of an oblique (non-parallel and non-perpendicular) shock as in Fig. 2.4.1, we find that the gradient drift due to the changing magnetic field is parallel to the electric field. This means that charged particles are accelerated by drifting in the same direction as the electric field, thereby gaining energy. This is called shock-drift acceleration.

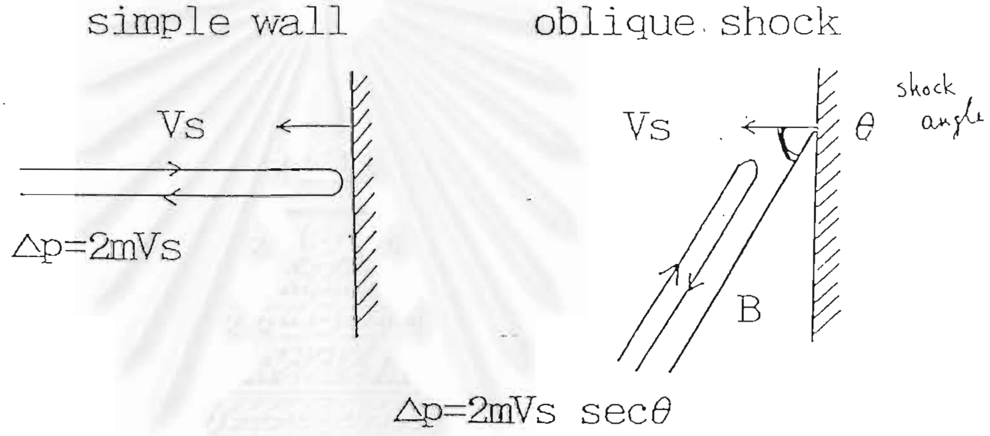

Figure 2.4.2: Momentum-changing (Δp) mechanism of shock drift acceleration for a quasi-perpendicular shock. The energy change is $(\Delta E) \approx (\Delta p)c$ (Terasawa 1995).

In the case of quasi-perpendicular shocks, shock-drift acceleration has been confirmed by observations, i.e., shock-spike events at the interplanetary shocks, and electron acceleration at the Earth's bow shock. The motion and energy and momentum changes due to shock drift acceleration are shown in Figs. 2.4.1 and 2.4.2.

2.5 Order of Magnitude Calculations of Shock-Drift Acceleration at the Solar Wind Termination Shock

This section shows details of rough calculations of shock-drift acceleration. We illustrate that in the case of the solar wind termination shock, charged particles can have difficulty drifting from the heliospheric equator to the poles or vice-versa. First, in the case of the solar wind termination shock, we can see that the momentum of the solar wind is approximately perpendicular to the shock normal vector $(\hat{n})$ (Fig. 2.2.1).

We want to calculate the gyroradius of a proton (r_g) at the solar wind termination shock in AU. Rewriting equation (12.42) from the classical electrodynamics text of Jackson (1977),

$$p_\perp/(\mathrm{MeV/c}) = 3.00 \times 10^{-4}(B_s/\mathrm{G})(r_g/\mathrm{cm}), \tag{2.5.1}$$

where $p_\perp$ is the perpendicular component of the momentum. At a quasi-perpendicular shock $p_\perp \approx p$. Also, r_g is the gyroradius of a proton at the shock, and B_s is the magnetic field at the solar wind termination shock.

We assume that the solar wind termination is at a distance of 110 AU from the Sun. Thus, we estimate this magnetic field as follows:

$$B_s = \frac{B(r=R)}{\sqrt{2}}\frac{R}{r}, \tag{2.5.2}$$

where B_s is the magnetic field at the shock in the φ direction at $\theta = 90°$ (see Fig. 2.5.1), $R = v_{sw}/(\omega \sin\theta)$ which is v_{sw}/ω at $\theta = 90°$, where v_{sw} is the solar wind speed, ≈400km/s, ω is the sidereal solar rotation angular frequency, $\approx 2\pi$ /(24 days) and r is the distance between the shock and the Sun in AU (110 AU). Now R is approximately equal to the distance between the Earth and the Sun (1 AU), and $B(r = R)$ is the magnetic field at $r = R$ (say, 5 nT). Thus the magnetic field in the φ direction (B_s) is 3.21×10^{-7} G or 3.21×10^{-11} T.

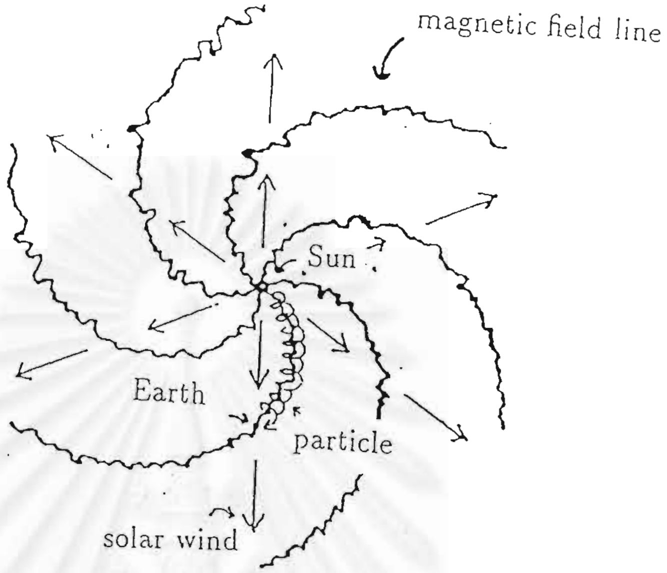

Figure 2.5.1: Schematic spiral magnetic field, B_φ, and the radial speed vector of the solar wind $u\hat{r}$ (Khumlumlert 1996).

From Pauls (1996),

$$\vec{B} = B_\varphi \hat{\varphi} \approx B_\varphi(\theta = 90°) \cdot \sin\theta \hat{\varphi} \tag{2.5.3}$$

where $B_\varphi(\theta = 90°) = B_s = 3.21 \times 10^{-11}$ T.

After that, we calculate the momentum of a proton at the solar termination shock (p) by using special relativity:

$$pc = \sqrt{T^2 + 2mc^2T}, \tag{2.5.4}$$

จุฬาลงกรณ์มหาวิทยาลัย

where mc^2 is the rest energy of a proton equal to 938.26 MeV, and T is the kinetic energy of the proton. Suppose that it is 1, 10, or 100 MeV, for a proton momentum of 43.3, 137.4, or 444.6 MeV/c, respectively. Using equation (2.5.1), we can calculate the gyroradius of a proton (for $p_\perp = p$) at the solar wind termination shock to be 0.0301, 0.0954, or 0.309 AU for a kinetic energy of 1, 10, or 100 MeV, respectively.

To find the electric field at the solar wind termination shock ($\vec{E}$), we use the equation below:

$$\vec{E} = -\vec{v_{sw}} \times \vec{B}, \tag{2.5.5}$$

where $\vec{v_{sw}}$ is the solar wind speed vector and $\vec{B}$ is the magnetic field at the solar termination shock. Using a magnetic field vector ($\vec{B}$) of $3.21 \times 10^{-11} \cdot \sin\theta\hat{\varphi}$ T and an average solar wind speed u of approximately 600 km/s, we get an electric field of

$$\vec{E} = 1.926 \times 10^{-5} \cdot \sin\theta(\hat{\theta}), \tag{2.5.6}$$

Physically, the maximum energy that can be gained by the shock-drift mechanism is the potential difference V along the termination shock from $\varphi = 0$ to $\varphi = \pi/2$ or vice-versa:

$$\vec{E} = -\vec{\nabla}V \tag{2.5.7}$$

$$V = -\int \vec{E} \cdot d\vec{x}. \tag{2.5.8}$$

In polar coordinates, V depends on θ only. Thus

$$V = -\int E r d\theta \tag{2.5.9}$$

$$V = -1.926 \times 10^{-5} \int_{\frac{\pi}{2}}^{0} \sin\theta \cdot r d\theta = -1.926 \times 10^{-5} \cdot r, \tag{2.5.10}$$

where the radius of the termination shock (r) is $110 \times 1.496 \times 10^{11}$m = 1.65×10^{13}m. Finally, the voltage V is -3.18×10^{8} Volts or -318 MV. To find the maximum drift energy of a proton at the termination shock (E_{drift}),

$$E_{\mathsf{drift}} = |qV|, \tag{2.5.11}$$

where q is the charge of the proton. Thus, we obtain a maximum drift energy of 318 MeV.

Let us estimate the maximum energy change (ΔE) of a proton at the termination shock due to a single particle-shock encounter. We can rewrite the relativistic energy as:

$$E^2 = p^2c^2 + m^2c^4, \tag{2.5.12}$$

where E is the total energy, mc^2 is the rest mass, and pc is the kinetic energy. Considering infinitesimal changes, we get

$$2EdE = 2pdpc^2. \tag{2.5.13}$$

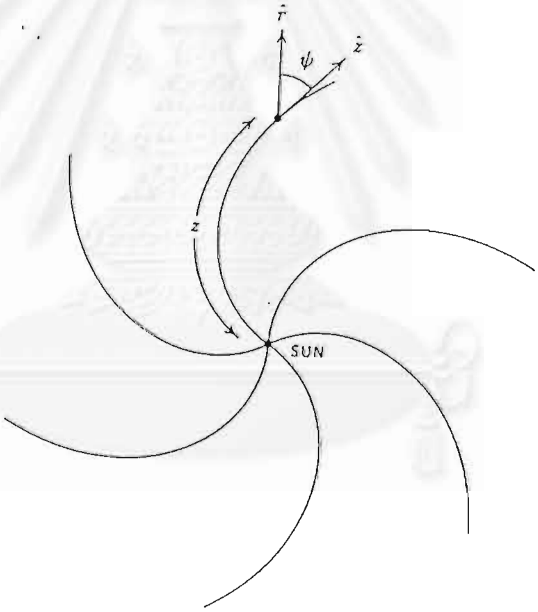

Figure 2.5.2: Illustration of the directions of $\hat{r}$, $\hat{z}$, and ψ (Ruffolo 1995).

Now since $E = \gamma mc^2$, and $p = \gamma mv$, where v is the particle velocity, we have

$$v = \frac{p}{E}c^2. \tag{2.5.14}$$

จุฬาลงกรณ์มหาวิทยาลัย

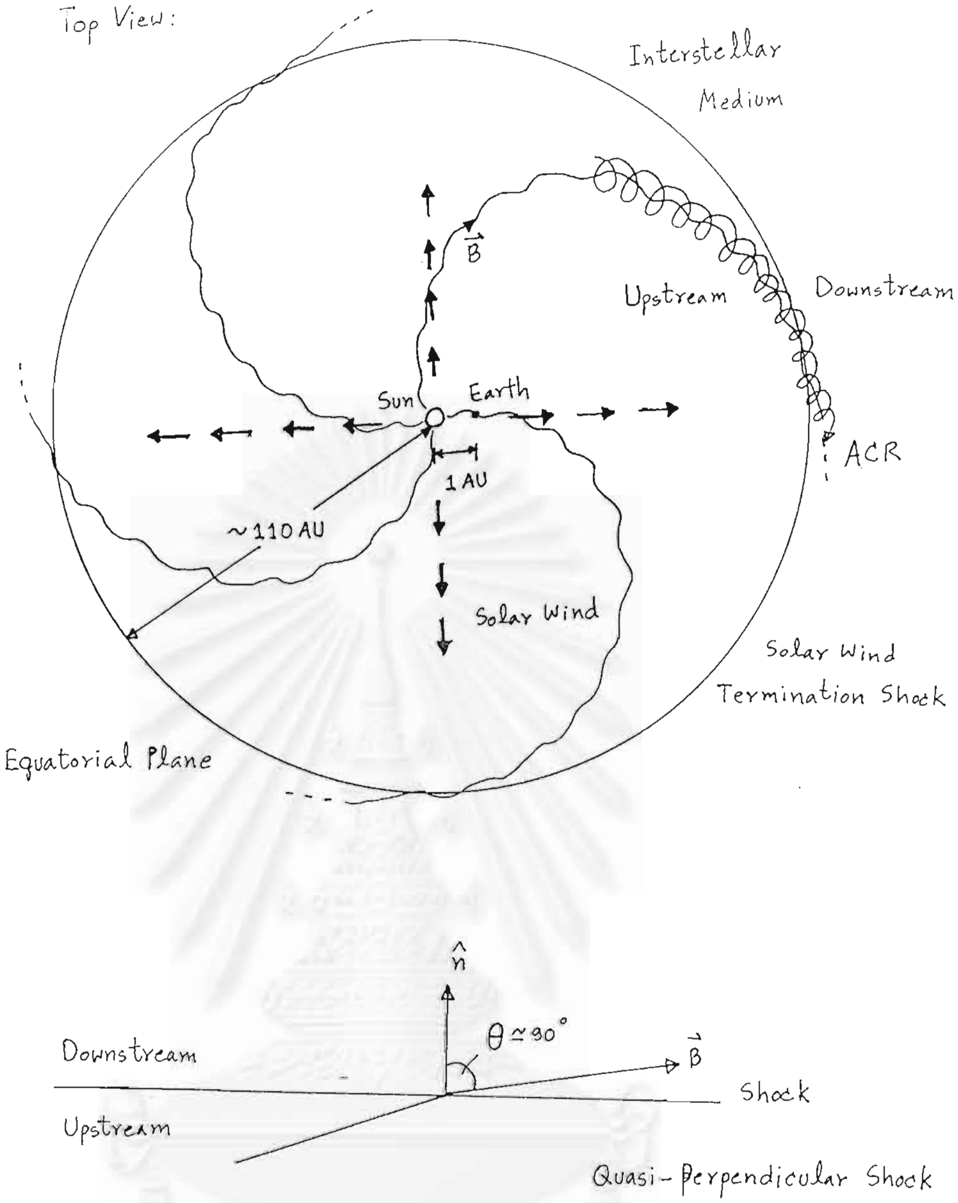

Figure 2.5.3: a) Drift distance due to a single shock encounter (D_{drift}) of anomalous cosmic rays of kinetic energy 1 MeV. b) The observationally inferred large drift distance of the anomalous cosmic rays.

จุฬาลงกรณ์มหาวิทยาลัย

Therefore equation (2.5.14) becomes

$$\Delta E = v\Delta p. \tag{2.5.15}$$

For an oblique shock, the momentum change of a particle (Δp), e.g., a pick-up ion, is approximately (Terasawa 1979; cited by Lagage and Cesarsky 1983)

$$\Delta p \approx 2mv_{sw} \sec\psi \tag{2.5.16}$$

(Figure 2.4.2) where ψ is the angle between $\hat{r}$ and $\hat{z}$, which is shown in Figure 2.5.2. For an ideal Archimedean spiral, $\sec\psi \approx r$ for $r \gg R$. Note that Eq. (2.5.15) only applies for small Δp ($\Delta pc << mc^2$). In fact, Δp will be small for realistic shock crossing angles, but can theoretically be large for a fortuitously large $\sec\psi$.

To calculate the drift distance of a particle, D_{drift} (see Figure 2.5.3), we use:

$$E = \frac{V}{D_{\text{drift}}}, \tag{2.5.17}$$

where E is the electric field at the termination shock as given by equation (2.5.5) or $|E| = v_{sw}B$, and V is the potential difference the particle drifts across or the energy change of particles (ΔE) divided by q. We can calculate the potential difference V as

$$V = \frac{v\Delta p}{q}. \tag{2.5.18}$$

Then D_{drift} is given by

$$D_{drift} = \frac{V}{E} = \frac{v\Delta p}{q} \cdot \frac{1}{v_{sw}B}, \tag{2.5.19}$$

and using equation (2.5.16), we get

$$D_{drift} = \frac{2m\sec\psi}{q} \cdot \frac{v}{B} \tag{2.5.20}$$

หอสมุดกลาง สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

where ψ is approximately 90 degrees at a nearly perpendicular shock. For the specific case of a proton of 1 MeV, v is the relativistic particle velocity of approximately 1.38×10^7 m/s, m is the mass of the proton (1.6725×10^{-27} kg), and B is the solar wind termination shock magnetic field that was calculated in equation (2.5.2), 3.21×10^{-11} T. We then obtain the drift distance of the proton at the solar termination shock as:

$$D_{\text{drift}} = \frac{8.98 \times 10^9}{\cos\psi}\text{m} = \frac{6.00 \times 10^{-2}}{\cos\psi}\text{AU} \tag{2.5.21}$$

For $\psi = 0$ degrees, the minimum of $D_{\text{drift}} = 6.02 \times 10^{-2}$AU. The drift distance increases with $\sec\psi$. [Note, however, that for $\sec\psi = 110, \Delta p$ is large and Eq. (2.5.15) is no longer valid.] For a perpendicular shock, $\psi \to 90°$, $D_{\text{drift}} \to \infty$.

Actually, a more detailed analysis has been performed by Jokipii (1987). The actual acceleration rate does not diverge as $\theta \to 90°$; rather it reaches a maximum of $1 + \eta^2$, where $\eta = \lambda_{\|}/r_g$, the ratio between the parallel scattering mean free path and the gyroradius. Also, considering the derivation of Jokipii (1982) for a nearly perpendicular shock, we find that

$$D_{\text{drift}} = -\frac{r-1}{3r}\frac{r_g}{|\mu|}\sec\psi \tag{2.5.22}$$

where r is the compression ration at the shock and μ is the cosine of the pitch angle. Therefore, while Jokipii (1982)'s analysis still contains a divergence, it shows that a typical drift distance is

$$D_{\text{drift}} \sim r_g \sec\psi. \tag{2.5.23}$$

จุฬาลงกรณ์มหาวิทยาลัย

For $\sec\psi = 110$, we have $D_{\text{drift}} \sim 3$ AU. This gives us a rough idea of how much drift is normally expected at the solar wind termination shock, in the idealized case without magnetic fluctuations. This is only a small fraction of the distance along the solar wind termination shock from the heliospheric equator to poles

(173 AU; see Figure 2.5.3). Later in this thesis, we will examine more realistic, turbulent fields, and we find that shock-crossing angles so close to 90° are in fact quite rare, so D_{drift} from a single shock encounter would be even smaller.

2.5.1 Conclusions of the order of magnitude calculations:

We assume that the radius of the solar wind termination shock is approximately 110 AU:

- The solar termination shock magnetic field is approximately 3.21×10^{-11} T.
- The electric potential at the solar wind termination shock from the heliospheric equator to poles is approximately 318 MV in magnitude.
- The gyroradius of a 1 MeV proton at the solar wind termination shock is approximately 0.03 AU.
- The maximum drift energy of such a proton from the equator to the poles is approximately 318 MeV.
- In the idealized case of a perfectly regular magnetic field nearly perpendicular to the shock normal, $\psi = 89^\circ$ ($\sec\psi = 110$), the drift distance is ~ 3 AU. This means that such a proton would typically drift only a small fraction of the distance along the solar wind termination shock from the heliospheric equator to poles.

2.6 Multiple Magnetic Field-Shock Crossings

In the last section, we showed that in the case of the solar wind termination shock, particles can typically drift only a small fraction of the distance (3 AU) along the solar wind termination shock from the heliospheric equator to the poles (see Figure (2.5.3c). Thus a single field-shock encounter, or even several of them, are insufficient to explain the observationally inferred drift of anomalous cosmic rays over much of that distance (Cummings, Stone, and Webber 1985).

We introduce the concept of multiple magnetic field-shock crossings as a more realistic model in which random trajectories of magnetic field lines can cross and recross a shock many times before completely passing the shock. If the plasma flow speed is much less than the velocity of particles, particle trajectories are helices around the magnetic field, and if we trace the magnetic fields, we should trace the particle trajectories. We trace random trajectories of the magnetic field by using a random phase in the magnetic field because the nature of the magnetic field itself is random (turbulent). After simulations, we want to characterize the multiple field-shock crossings as in Figure 2.6.1.

Let us first consider the random walk of particles along the random magnetic field by ignoring the possibility of reflection when approaching the shock from upstream. Considering Figure 2.6.2, in this framework we view the acceleration process in terms of discrete episodes of diffusive shock acceleration (which includes shock drift acceleration) when the particle encounters a field-shock crossing. In addition to the correlation lengths, other relevant length scales for a given particle species and energy include the gyroradius r_g and the scattering mean free path $\lambda_{||}$. Since the particle motion follows a sort of average of $\vec{B}$ over a gyroradius, field-shock crossings closer together than r_g should be grouped together so that the particle interaction in that region is considered to constitute a single particle-shock encounter.

Next, field-shock crossings spaced father than r_g but closer than $\lambda_{||}$ will generally be traversed in sequence; N such crossings can then yield an N-fold enhancement in shock acceleration. We refer to this as a linear enhancement.

Now consider only field-shock crossings or groups of field-shock crossings that are spaced father apart than $\lambda_{||}$. For this purpose, the magnetic field in Figure

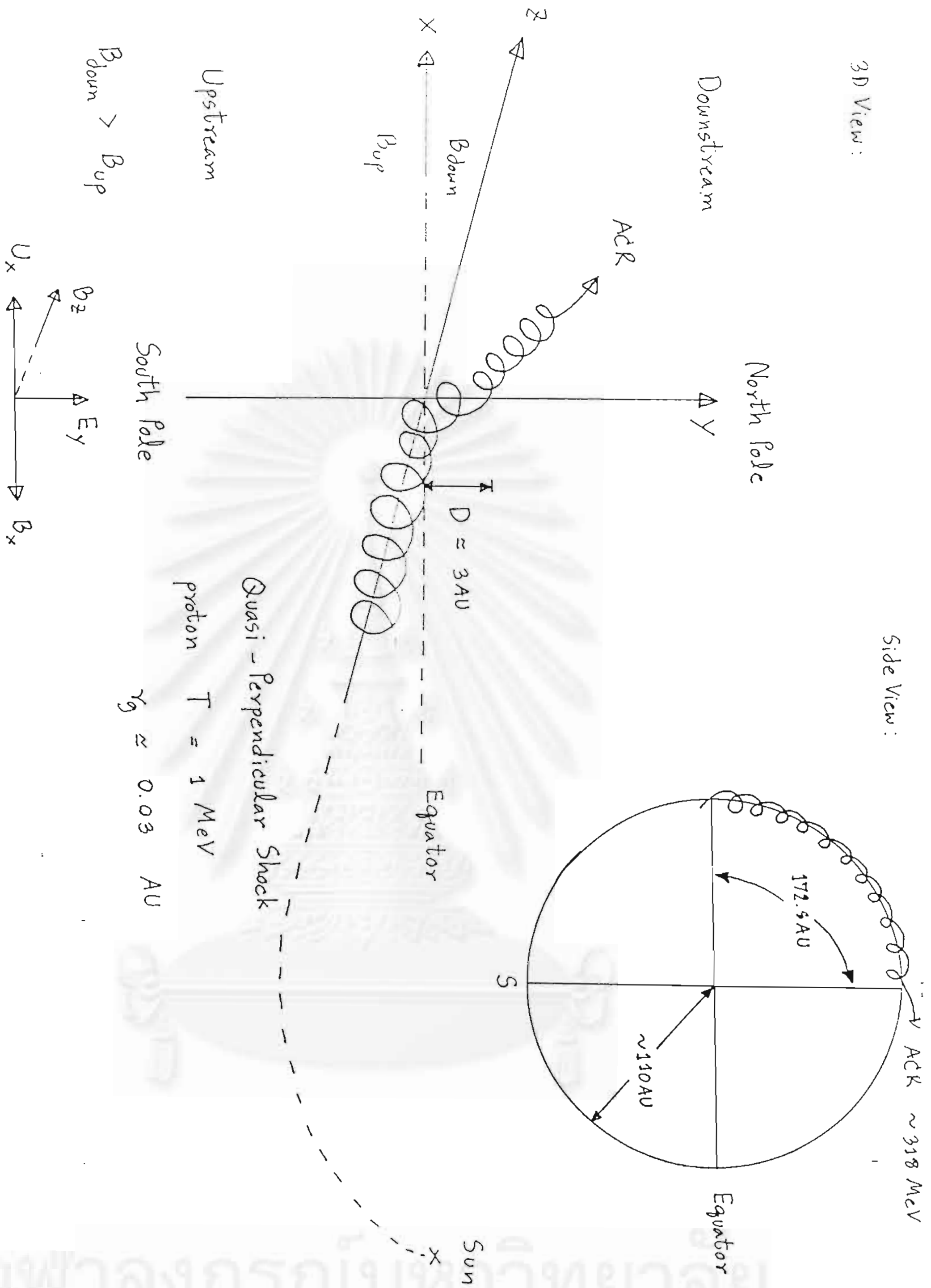

Figure 2.6.1: Top view of the solar system. The random magnetic field can cross the quasi-perpendicular solar wind termination shock multiple times.

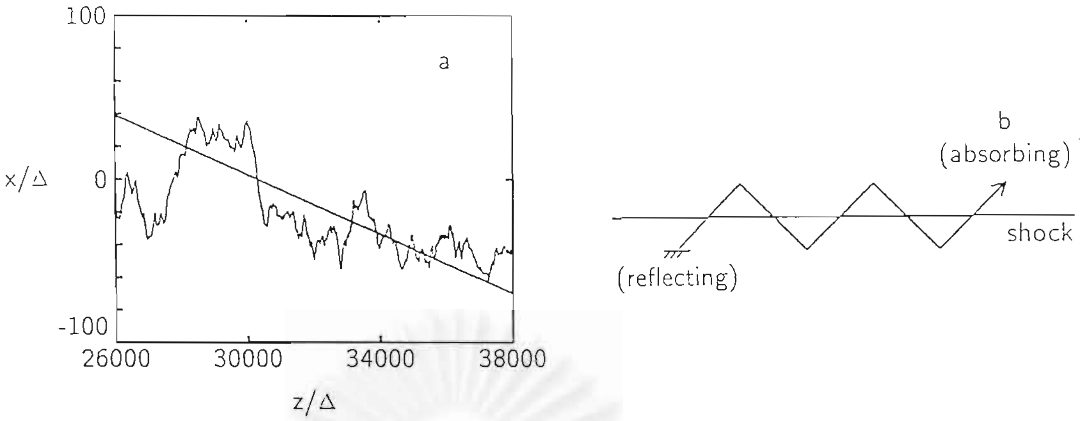

Figure 2.6.2: a) A sample magnetic field line that crosses a shock (diagonal line) multiple times. Note the greatly expanded vertical scale. b) Schematic of the above multiple magnetic field-shock crossings, and boundary conditions for the random walk of particles along $\vec{B}$.

2.6.2a can be conceptually simplified as in Figure 2.6.2b. (Again, for simplicity we have not indicated the refraction of magnetic field lines at the shock.) Between two crossings, a particle's motion is randomized, and there is an equal probability of either moving forward to the next crossing or returning to the previous crossing.

This then becomes a classic random walk problem (Chandrasekhar 1943), as described in detail in Appendix A, and is amenable to mathematical analysis. Starting from upstream of the shock on the left hand side, let n be the number of times a particle encounters the shock, and let m indicate the regions between field-shock crossings from left ($m = 0$) to right ($m = N$). To represent the ultimate return of particles upstream of the first field-shock crossing, due to convection, we place a reflecting barrier at $m = 0$, and to represent escape downstream, we use the conservative assumption of absorption at $m = N$. This is slightly different from the situation considered by Chandrasekhar (1943), who considered only one barrier at a time. The probability of escape after n shock encounters can be shown to be

$$P(n) = \sum_{j=0,m\leq n} \frac{(-1)^j}{2^{n-1}} \begin{pmatrix} n-1 \\ (n+m-2)/2 \end{pmatrix} - \sum_{j=0,m\leq n-2} \frac{(-1)^j}{2^{n-1}} \begin{pmatrix} n-1 \\ (n+m)/2 \end{pmatrix}, \tag{2.6.1}$$

where $n+N$ is even and $m = (2j+1)N$. This probability sums to 1 (when summing over all $n \geq N$ such that $n+N$ is even), and the mean value of n is N^2. The mean number of shock encounters before escape should actually be even larger if we consider that a large fraction of particles approaching a shock from upstream (87% for a strong shock with a compression ratio of 4) should be reflected backup upstream, which gives some probability of trapping between two adjacent field-shock crossings.

Therefore, even with conservative assumptions, the multiple magnetic field-shock crossing model predicts a quadratic enhancement by a factor of $\sim N^2$ in the shock drift and total energization of particles, where N is the number of field-shock crossings spaced farther than λ. Presumably this enhancement is occurring in Monte Carlo simulations of particle acceleration at nearly perpendicular shocks. In practice the total energy gain of particles will also be limited by the lateral extent of the shock, and the convection of field lines past the shock. This quadratic enhancement is our proposed solution to the problem we found in the order of magnitude calculations of the previous section. Taking into account turbulence in the magnetic field natually provides a large number of particle-shock encounters, potentially explaining the observations that a significant fraction of anomalous cosmic rays can achive energies on the order of the maximum drift energy, and can drift over a large fraction of the distance from the heliosphere equator to the poles or vice-versa.

Chapter III

Model of Turbulent Magnetic Fields

3.1 Slab Model

Magnetometer measurements of magnetic turbulence in a space plasma flowing past a spacecraft yield a one-dimensional spectrum of the turbulence. This inspired the one-dimensional model of turbulence called the "slab model" in which the turbulence depends on the z-direction only. A magnetic field can be expressed in terms of the mean magnetic field, $\vec{B}_0(z) = B_0\hat{z}$, and its turbulent components, $\delta B_x(z)$ and $\delta B_y(z)$, as follows:

$$\vec{B}(z) = B_o\hat{z} + \delta B_x(z)\hat{x} + \delta B_y(z)\hat{y}. \tag{3.1.1}$$

Because the magnetic field itself is random, thus we can describe it by statistical characteristics such as ensemble averages, correlations, etc. To describe the magnetic helicity in a spectrum of fluctuations with different wavelengths, it is assumed that the magnetic field $\vec{B}$ can be separated into a mean field $\vec{B}_0$ and a fluctuating component $\delta\vec{B}$, i.e., $\vec{B} = \vec{B}_0 + \delta\vec{B}$. Here, we introduce the two-point correlation $R_{ij}(\vec{r})$ defined as (Pauls 1993)

$$R_{ij}(\vec{r}) =< \delta B_i(\vec{x})\delta B_j(\vec{x}+\vec{r}) > . \tag{3.1.2}$$

In the case of homogeneous turbulence, $R_{ij}(\vec{r})$ is independent of the position $\vec{x}$ where the correlation is evaluated. The power spectrum matrix $P_{ij}(\vec{r})$ is given by its Fourier transform as:

$$P_{ij}(\vec{k}) = \frac{1}{(2\pi)^3}\int_{-\infty}^{\infty} \vec{R_{ij}}(\vec{r})e^{-i\vec{k}\vec{r}}d\vec{r}. \tag{3.1.3}$$

Bieber and Matthaeus (1997) suggest the form

$$P_{ij}(\vec{k}) = A(k_\perp, k_z)\{\delta_{ij} - k_i k_j/k^2\}, \tag{3.1.4}$$

where for the slab model

$$A(k_\perp, k_z) = C(1 + k_z^2\lambda^2)^{-\nu}\delta(k_\perp)/k_\perp. \tag{3.1.5}$$

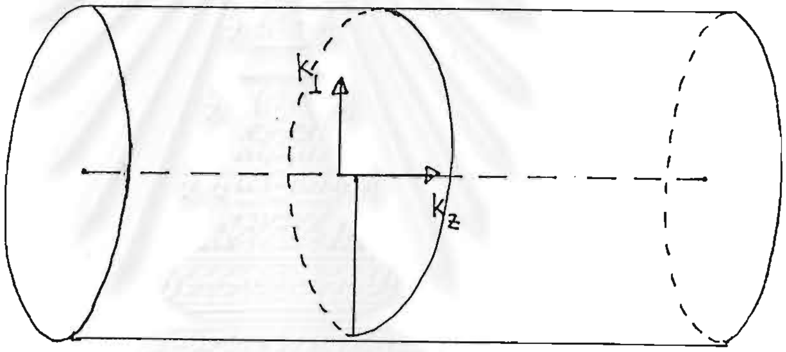

Figure 3.1.1: In the slab model of turbulence, the turbulent magnetic field depends on the k_z component only.

In the slab model, $k_\perp = 0$ because the turbulence corresponds to fluctuations along the solar wind flow (see Figure 3.1.1) and the only non-zero components of the power spectrum are P_{xx} and P_{yy}, which are equal. From equation (3.1.4), we got the power spectrum which was used to simulate a magnetic field line as follows:

$$P_{xx}(k_z) = C(1 + k_z^2\lambda^2)^{-\nu}. \tag{3.1.6}$$

The index ν is chosen as 5/6 based on the Kolmogorov model of turbulence. The parameter λ determines the scale at which the power spectrum starts to decrease. We want to relate the parameters λ and C to the physical parameters δB^2, the total turbulence of the magnetic field, and the correlation length, λ_c, which are defined as follows:

$$\delta B^2 = \delta B_x^2 + \delta B_y^2 \equiv < [\delta B_x(z)]^2 > + < [\delta B_y(z)]^2 > \tag{3.1.7}$$

and

$$\lambda_c \equiv \frac{\int\limits_0^\infty < \delta B_x(z)\delta B_x(z+z^{'}) > dz^{'}}{\delta B_x^2(z)}. \tag{3.1.8}$$

First, we introduce a version of the convolution theorem which is shown in Arfken and Weber (1995) as:

$$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\delta B_i(z_0)\delta B_j(z_0+z)e^{-ik_z z}dz\right]e^{ik_z^{'} z_0}dz_0 = \delta B_i(k_z^{'}-k_z)\delta B_j(k_z). \tag{3.1.9}$$

At $k_z^{'} = 0$, we get

$$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty}\delta B_i(z_0)\delta B_j(z_0+z)e^{-ik_z z}dz\right]dz_0 = \delta B_i^*(k_z)\delta B_j(k_z), \tag{3.1.10}$$

where $\delta B(-k_z) = \delta B_i^*(k_z)$ because $\delta B_i(z)$ is a real function. We can relate this to the one-dimensional power spectrum,

$$P_{ij}(k_z) \equiv \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} < \delta B_i(z_0)\delta B_j(z_0+z) > e^{-ik_z z}dz \tag{3.1.11}$$

as follows:

$$\frac{L}{\sqrt{2\pi}}\left[\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} < \delta B_i(z_0)\delta B_j(z_0+z) > e^{-ik_z z}dz\right] = \delta B_i^*(k_z)\delta B_j(k_z) \tag{3.1.12}$$

$$P_{ij}(k_z) = \frac{\sqrt{2\pi}}{L}\delta B_i^*(k_z)\delta B_j(k_z), \tag{3.1.13}$$

จุฬาลงกรณ์มหาวิทยาลัย

where L is a periodic length which is used in a simulation program, $L = N\Delta$, N is the total number of data points in the x-direction, and Δ is the grid size in the x-direction. In the case of $i = j = x$ or $i = j = y$, we get

$$|B_i(k_z)|^2 = \frac{L}{\sqrt{2\pi}} P_{ii}(k_z) \tag{3.1.14}$$

Here, we introduce Parseval's theorem as follows ($i = x$):

$$\int_{-\infty}^{\infty} |B_x(k_z)|^2 dk_z = \int_{-\infty}^{\infty} B_x^2(z) dz \tag{3.1.15}$$

or for a discrete Fourier transform over periodic ranges,

$$\sum |B_x^2(\bar{\nu})|^2 \Delta\bar{\nu} = \sum B_x^2(z) \cdot \Delta \tag{3.1.16}$$

$$\frac{1}{N\Delta} \sum |B_x^2(\bar{\nu})|^2 \Delta\bar{\nu} =< B_x^2(z) > \tag{3.1.17}$$

where $\bar{\nu} \equiv k/2\pi$ and $\Delta\bar{\nu} = 1/(N\Delta)$. The left side of equation (3.1.16) becomes [for $1/(2\Delta) >> 1/\lambda$, i.e., $2\Delta << \lambda$]

$$\begin{aligned} \text{L.S.} &\approx \frac{1}{N\Delta} \int_{-\infty}^{\infty} |B_x(\bar{\nu})|^2 d\bar{\nu} && (3.1.18) \\ &\approx \sqrt{2\pi} C \int_{-\infty}^{\infty} \frac{d\bar{\nu}}{(1 + k_z^2\lambda^2)^{\nu}} && (3.1.19) \end{aligned}$$

using equations (3.1.6) and (3.1.14), $L = N\Delta$, and the identity $B_x(\bar{\nu}) = \sqrt{2\pi} B_x(k_z)$. Finally, we get

$$\text{L.S.} \approx \frac{2C}{\sqrt{2\pi}} \int_0^{\infty} \frac{dk_z}{(1 + k_z^2\lambda^2)^{\nu}}, \tag{3.1.20}$$

where $\nu = 5/6$. We use the Mathematica program to calculate

$$\int_0^{\infty} \frac{du}{(1 + u^2)^{5/6}} = 2.104. \tag{3.1.21}$$

Thus we can estimate equation (3.1.20) as

$$\text{L.S.} = \frac{2C}{\sqrt{2\pi}} \frac{2.104}{\lambda}. \tag{3.1.22}$$

The right side of equation (3.1.16) becomes

$$< B_x^2 >= \delta B_x^2 = \frac{1}{2}\delta B^2 \tag{3.1.23}$$

Equation (3.1.22) is equal to equation (3.1.23), thus we get

$$\delta B^2 \lambda = \frac{4C}{\sqrt{2\pi}}(2.104) \tag{3.1.24}$$

From equation (3.1.8)

$$\delta B_x^2 \lambda_c = \frac{1}{2}\int_{-\infty}^{\infty} < B_x(z) B_x(z+z') > dz'. \tag{3.1.25}$$

Next, we note that from the definition, equation (3.1.11), with $k_z = 0$,

$$P_{xx}(0) \equiv \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} < B_x(z) B_x(z+z') > dz', \tag{3.1.26}$$

and $P_{xx}(0) = C$ from equation (3.1.6), so we get

$$\delta B^2 \lambda_c = \sqrt{2\pi} C. \tag{3.1.27}$$

From equations (3.1.24) and (3.1.27), we can relate the parameters λ and C to the physical parameters δB^2, the total turbulence of the magnetic field, and the correlation length λ_c as follows:

$$\lambda = \frac{2\lambda_c}{\pi}(2.104) \tag{3.1.28}$$

$$C = \frac{1}{\sqrt{2\pi}}\delta B^2 \lambda_c. \tag{3.1.29}$$

3.2 2D + Slab Model

We simulate three-dimensional magnetic field turbulence by superimposing two types of turbulence, the 2D model depending on x and y, and the slab model depending on z. An admixture of 80% 2D turbulence and 20% slab turbulence

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

provides a good fit to interplanetary turbulence (Matthaeus, Bieber and Zank 1995):

$$\delta B^2 = 0.8\delta B_{2D}^2 + 0.2\delta B_{slab}^2 \tag{3.2.1}$$

This three-dimensional model of turbulence can be calculated by using the equation below:

$$\vec{B}(x,y,z) = B_o\hat{z} + [\delta B_{x,slab}(z) + \delta B_{x,2D}(x,y)]\hat{x} + [\delta B_{y,slab}(z) + \delta B_{y,2D}(x,y)]\hat{y} \tag{3.2.2}$$

The slab components are found as before. For the power spectrum of $\delta\vec{B_{2D}}$,

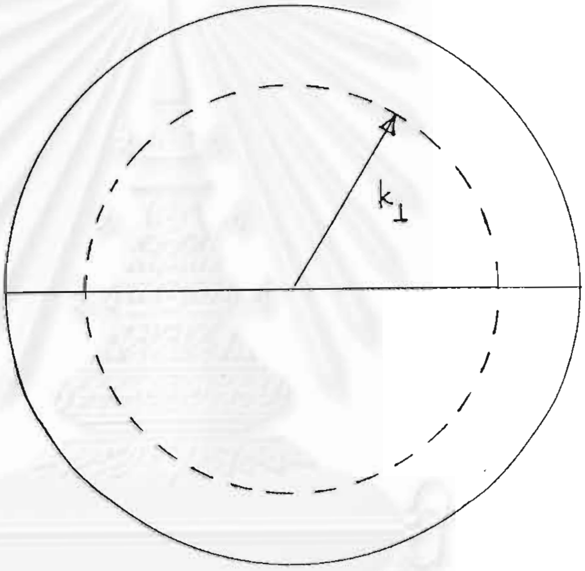

Figure 3.2.1: In the 2D model of turbulence, the turbulent magnetic field depends on the $k_\perp$ component only.

we note that the Fourier transform $\delta\vec{B_{2D}}(\vec{k_\perp})$ only depends on $\vec{k_\perp} = k_x\hat{x} + k_y\hat{y}$. Furthermore, we assume axisymmetry (see Figure 3.2.1) so that $\delta\vec{B_{2D}}(\vec{k_\perp})$ only depends on $k_\perp = |k_\perp| = \sqrt{k_x^2 + k_y^2}$. Then

$$\begin{aligned} \delta B_{2D}^2 &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |\delta B_{2D}(k_\perp)|^2 dk_x dk_y && (3.2.3) \\ &= 2\pi\int_0^{\infty} |\delta B_{2D}(k_\perp)|^2 k_\perp dk_\perp. && (3.2.4) \end{aligned}$$

According to the Kolmogorov theory of turbulence, the distribution of the energy density, $\delta B^2/(8\pi)$, vs. the scale of irregularities (irrespective of their orientation) should tend to $k^{-5/3}$ for large k. Thus we set

$$|\delta B_{2D}(k_\perp)|^2 k_\perp = \frac{C}{(1+k_\perp^2\lambda^2)^\nu} \tag{3.2.5}$$

$$|\delta B_{2D}(k_\perp)|^2 = \frac{C}{k_\perp(1+k_\perp^2\lambda^2)^\nu}, \tag{3.2.6}$$

where the index ν is chosen as 5.0/6.0 based on the Kolmogorov model of turbulence (ν in the 2D model is equal to ν in the slab model). Next, we note that the Fourier transform of equation (3.2.6) gives

$$\delta B_{2D}(\vec{k_\perp}) = \vec{k_\perp} \times [a(\vec{k_\perp})\hat{z}] \tag{3.2.7}$$

and

$$|\delta B_{2D}(\vec{k_\perp})|^2 = k_\perp^2 |a(\vec{k_\perp})|^2 \tag{3.2.8}$$

$$|a(\vec{k_\perp})|^2 = \frac{|\delta B_{2D}(\vec{k_\perp})|^2}{k_\perp^2} = \frac{C}{k_\perp^3(1+k_\perp^2\lambda^2)^\nu}. \tag{3.2.9}$$

After using the inverse Fourier transform (I.F.T.), we can get

$$a(\vec{k_\perp}) \rightarrow \text{ I.F.T. } \rightarrow a(x,y)\hat{z} \tag{3.2.10}$$

We can calculate the total turbulence of the magnetic field, $\delta \vec{B_{2D}}$, as:

$$\delta \vec{B_{2D}} = \vec{\nabla} \times [a(x,y)\hat{z}] \tag{3.2.11}$$

$$\text{or } \delta B_x = \frac{\partial a(x,y)}{\partial y} \text{ and } \delta B_y = -\frac{\partial a(x,y)}{\partial x} \tag{3.2.12}$$

As in the slab model, we want to relate the parameters λ and C to the physical parameter δB^2, the total turbulence of the magnetic field, and the correlation length λ_c. First, we write down Parseval's theorem in two dimensions.

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |B_x(k_x,k_y)|^2 dk_x dk_y = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} B_x^2(x,y)dxdy \tag{3.2.13}$$

The left side becomes

$$\text{L.S.} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |B_x(\bar{\nu_x}, \bar{\nu_y})|^2 d\bar{\nu_x} d\bar{\nu_y}, \tag{3.2.14}$$

where $\bar{\nu_x} \equiv k_x/2\pi$ and $\bar{\nu_y} \equiv k_y/2\pi$. Again, as for the slab model, for a discrete Fourier transform over periodic ranges,

$$\sum_{i,j} |B_x(\bar{\nu_{x_i}}, \bar{\nu_{y_j}})|^2 (\Delta\bar{\nu})^2 = \sum_{i,j} |B_x(x_i, y_j)|^2 \cdot (\Delta)^2, \tag{3.2.15}$$

where $\Delta\bar{\nu} = 1/(N\Delta)$. Then

$$\frac{1}{N^4\Delta^4}\sum_{i,j} |B_x(\bar{\nu_{x_i}}, \bar{\nu_{y_j}})|^2 = \frac{1}{N^2}\sum_{i,j} B_x^2(x_i, y_j) = < \delta B_x^2 > . \tag{3.2.16}$$

The left side of equation (3.2.15) becomes

$$\begin{aligned}
\text{L.S.} \ &\approx \ \frac{1}{N^2\Delta^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} |B_x(\bar{\nu_i}, \bar{\nu_j})|^2 d\bar{\nu_i} d\bar{\nu_j} && (3.2.17)\\
&\approx \ 2\pi C \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{d^2\bar{\nu}}{k_\perp(1+k_\perp^2\lambda^2)^\nu} && (3.2.18)\\
&\approx \ C\int_0^{\infty} \frac{dk_\perp}{(1+k_\perp^2\lambda^2)^\nu} && (3.2.19)\\
&\approx \ \frac{C}{\lambda}(2.104). && (3.2.20)
\end{aligned}$$

Finally, we get the relation

$$\delta B^2\lambda = 2C(2.104). \tag{3.2.21}$$

We estimate that

$$\lambda_{2D} = \lambda_{slab} = \frac{2(2.104)\lambda_c}{\pi}. \tag{3.2.22}$$

From equations (3.2.21) and (3.2.22), we can relate the parameters λ and C to the physical parameter δB^2, the total turbulence of the magnetic field, and the correlation length λ_c as follows:

$$\lambda = \frac{2\lambda_c}{\pi}(2.104) \tag{3.2.23}$$

$$C = \frac{1}{\pi}\delta B^2 \lambda_c. \tag{3.2.24}$$

จุฬาลงกรณ์มหาวิทยาลัย

Chapter IV

Simulation Techniques and Their Testing

4.1 Overview of Numerical Simulations

In order to determine the characteristics of magnetic field-shock crossings, we computationally generated random magnetic fields for a specified power spectrum matrix using inverse Fourier transforms and a random phase. In our work, we use an inverse fast Fourier transform program and a random number generator program from the standard reference, Numerical Recipes (Press et al. 1988). After an inverse Fourier transform, we get a random magnetic field in the x-direction and y-direction (B_x and B_y). A trajectory of the random magnetic field can be found by Euler's method by using B_x and B_y. We assume that the shock wave plane, e.g., representing the solar wind termination shock, is a flat plane parallel to the y-axis. In the quasi-perpendicular case, the angle between the mean magnetic field (z-axis) and the unit vector normal to a shock is nearly perpendicular. Our procedure for transforming the turbulent power in the wave vector space to the position space in $1D$ or $3D$ is shown below.

One-dimensional simulations (slab model):

$$(\sqrt{P_{xx}(k_z)}\cdot \mathrm{e}^{i\varphi})_{slab\ model} \rightarrow \text{1D Inverse Fourier Transforms} \rightarrow B_x(z),\ B_y(z) \rightarrow$$
$$\text{Euler's Method} \rightarrow \text{1D Random Trajectories} \rightarrow \text{Crossings of a Shock} \rightarrow$$

T18749902

Collect Statistical Data

Three-dimensional simulations (2D + slab model):

$(|a(k_\perp)|\, e^{i\varphi})_{2D\ model}$ →2D Inverse Fourier Transform→ $a_{xy}(x, y)$

$a_{xy}(x, y) + (B_x(z),\, B_y(z))_{slab}$ → Linear, Bilinear Interpolation + Euler's Method → $B_x(x, y, z)$, $B_y(x, y, z)$ → 3D Random Trajectories → Crossings of a Shock →

Collect Statistical Data

Here, $P_{ij}(k_z)$ is the power spectrum of the turbulent magnetic field which is calculated from the slab model (see Chapter 3), $|a(k_\perp)|$ is the magnitude of the scalar function which is calculated from the 2D model, and $e^{i\varphi}$ is a random phase.

4.2 Inverse Fourier Transform

In Chapter 2, the turbulence of the magnetic field is described in wave vector space, $\vec{k}$, but we want to know the turbulence of the magnetic field in position space by using a Fourier transfrom. The Fourier transform is well known as a technique for solving problems in linear systems. The Fourier transform (in terms of the wave number $\bar{\nu}$) of $H(z)$ is $h(\bar{\nu})$ and the inverse Fourier transform of $h(\bar{\nu})$ is $H(z)$:

$$h(\bar{\nu}) = \int_{-\infty}^{\infty} H(z) e^{-i2\pi\bar{\nu}z} dz \tag{4.2.1}$$

$$H(z) = \int_{-\infty}^{\infty} h(\bar{\nu}) e^{i2\pi\bar{\nu}z} d\bar{\nu}. \tag{4.2.2}$$

In our work, we use the one-dimensional inverse Fourier transform, which is shown in equation 4.2.3, to simulate B_x and B_y:

$$H(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} h(k) e^{ikz} dk, \tag{4.2.3}$$

where $k = 2\pi\bar{\nu}$ and $h(k) = h(\bar{\nu})/\sqrt{2\pi}$, k is the angular wave number, and $\bar{\nu}$ is the wave number. In our simulations, we use an inverse fast Fourier transform (IFFT) to transform $h(\bar{\nu})$ to $H(z)$.

In the 3D simulation, we use a 2D inverse Fourier transform $H(x, y)$ which is shown for simulating two dimensional magnetic turbulence:

$$H(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(\bar{\nu_x}, \bar{\nu_y}) e^{i2\pi(\bar{\nu_x}x+\bar{\nu_y}y)} d\bar{\nu_x} d\bar{\nu_y} \tag{4.2.4}$$

The two-dimensional inverse Fourier transform can also be written

$$H(x,y) = \left(\frac{1}{\sqrt{2\pi}}\right)^2 \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h(k_x, k_y) e^{i(k_x x+k_y y)} dk_x dk_y \tag{4.2.5}$$

where $k = 2\pi\bar{\nu}$, $h(k_x, k_y) = h(\bar{\nu_x}, \bar{\nu_y})/2\pi$, k is the angular wave number, and $\bar{\nu}$ is the wave number. In 2D simulations, we use a 2D inverse fast Fourier transform to transform $h(\bar{\nu_x}, \bar{\nu_y})$ to $H(x, y)$. Details of the inverse fast Fourier transform are shown in Appendix C.

4.3 Random Phase Approximation

We can specify the power $|h(\bar{\nu})|^2$ because it is related to the average properties which are described in Chapter II. This gives us the amplitude of $h(\bar{\nu})$, $|h(\bar{\nu})|$. Because the magnetic field itself is random, we describe it by using a random phase. Thus we set the phase of the magnetic field to a random number. We use a random number generator program from the book, Numerical Recipes in C (Press et al. 1988). We calculate this random phase φ, and multiply the amplitude $|B(\vec{k})|$ by $e^{i\varphi}$ in calculating the inverse Fourier transform. After that, we calculate an inverse Fourier transform, as described in the previous section. Finally, we will get a random magnetic field for further calculations.

4.4 Euler's Method and Interpolation

4.4.1 Euler's method

Euler's method is a simple method to solve a 1^{st}-order differential equation. The general form of this method is

$$\frac{dx}{dz} = f(x, z) \tag{4.4.1}$$

or

$$\frac{x_{i+1} - x_i}{z_{i+1} - z_i} = \frac{x_{i+1} - x_i}{\Delta} = f(x_i, z_i) \tag{4.4.2}$$

where Δ is the grid size spacing along the z-axis.

We can calculate the new value x_{i+1} from the previous value x_i and $f(x_i, z_i)$

$$x_{i+1} = x_i + f(x_i, y_i) \cdot \Delta \tag{4.4.3}$$

In our simulations, we use

$$x_{i+1} = x_i + \Delta \frac{\delta B_x(i)}{B_0} \tag{4.4.4}$$

where x_i is the trajectory of the magnetic field at $z = i\Delta$, $B_x(i)$ is the magnetic field at the same grid point, and B_0 is the mean magnetic field.

We use Euler's method to trace a random magnetic field because the nature of random numbers is unpredictable. We cannot accurately estimate the derivatives of a function that represents them. Euler's method is a simple and convenient choice because it only requires evaluating function values, not their derivatives. For the three-dimensional field model, bilinear interpolation is required.

4.4.2 Linear interpolation

We can describe the field line trajectory vs. z by using linear interpolation (e.g., in order to determine a shock-crossing location). The linear interpolation of two points is presented in Figure (4.4.1) below.

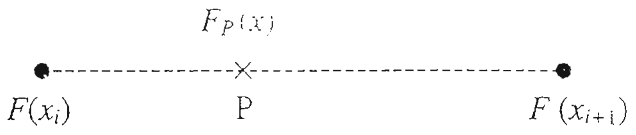

Figure 4.4.1: Linear interpolation (depends on i only) at point P between two points $F(x_i)$ and $F(x_{i+1})$.

We can estimate the function value F_p at point P as follows:

$$\begin{aligned} F_P &= \frac{x_{i+1} - x}{x_{i+1} - x_i} F(x_i) + \frac{x - x_i}{x_{i+1} - x_i} F(x_{i+1}) \\ &= (1 - f_x)F(x_i) + f_x F(x_{i+1}), \end{aligned} \tag{4.4.5}$$

where $f_x \equiv (x_{i+1} - x)/(x_{i+1} - x_i)$.

4.4.3 Bilinear interpolation

If we want to calculate the linear interpolation in 2D, we can use bilinear interpolation. In this work, we use bilinear interpolation for interpolating magnetic field values in 2D. The bilinear interpolation is presented in Figure (4.4.2) below.

We can calculate a function value F_S at point S by using bilinear interpolation as follows:

At point P:

$$F_P(x, y_j) = (1 - f_x)F(x_i, y_j) + f_x F(x_{i+1}, y_j) \tag{4.4.6}$$

At point Q:

$$F_Q(x, y_{j+1}) = (1 - f_x)F(x_i, y_{j+1}) + f_x F(x_{i+1}, y_{j+1}) \tag{4.4.7}$$

At point S:

$$F_S(x, y) = (1 - f_y)F_P(x, y_j) + f_y F_Q(x, y_{j+1}), \tag{4.4.8}$$

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

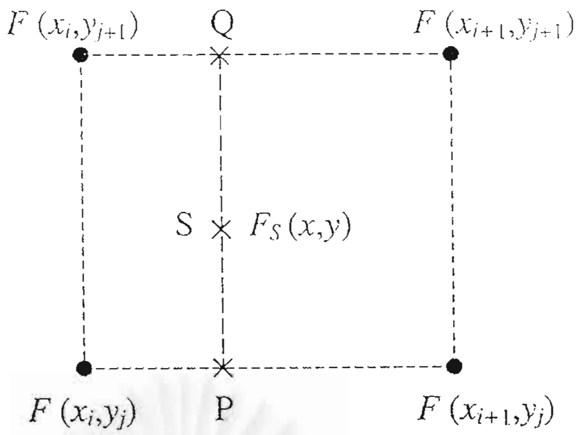

Figure 4.4.2: Bilinear interpolation at point S.

At point S:

$$F_S(x,y) = (1-f_y)F_P(x,y_j) + f_y F_Q(x,y_{j+1}), \tag{4.4.8}$$

where

$$f_y = \frac{y-y_j}{y_{j+1}-y_j}. \tag{4.4.9}$$

Now, we can calculate the function value $F_S(x,y)$ as follows:

$$\begin{aligned} F_S(x,y) &= (1-f_y)(1-f_x)F(x_i,y_j) + (1-f_y)f_x F(x_{i+1},y_j) \\ &\quad +(1-f_x)f_y F(x_i,y_{j+1}) + f_x f_y F(x_{i+1},y_{j+1}). \end{aligned} \tag{4.4.10}$$

4.5 Trajectories of Magnetic Field Lines and Field-Shock Crossings

จุฬาลงกรณ์มหาวิทยาลัย

After we find x- and y-components of the random magnetic field by using inverse Fourier transforms from a power spectrum in k-space with the random phase approximation, a trajectory of the random magnetic field can be found by Euler's method as follows:

$$x_i = x_{i-1} + \Delta \cdot \delta B_x(i)/B_0 \text{ and } y_i = y_{i-1} + \Delta \cdot \delta B_y(i)/B_0, \tag{4.5.1}$$

distance between adjacent grid points. In this simulation, we set B_0 equal to one for simplicity.

In the one-dimensional simulations, we only use Euler's method for simulating random trajectories (Figure 4.5.1) because the turbulence magnetic field in the xy-plane is a constant at each grid point. However, in the three-dimensional simulations, the turbulent magnetic field in the xy-plane varies with the x and y position, so we use bilinear interpolation to calculate these values (Figure 4.5.2).

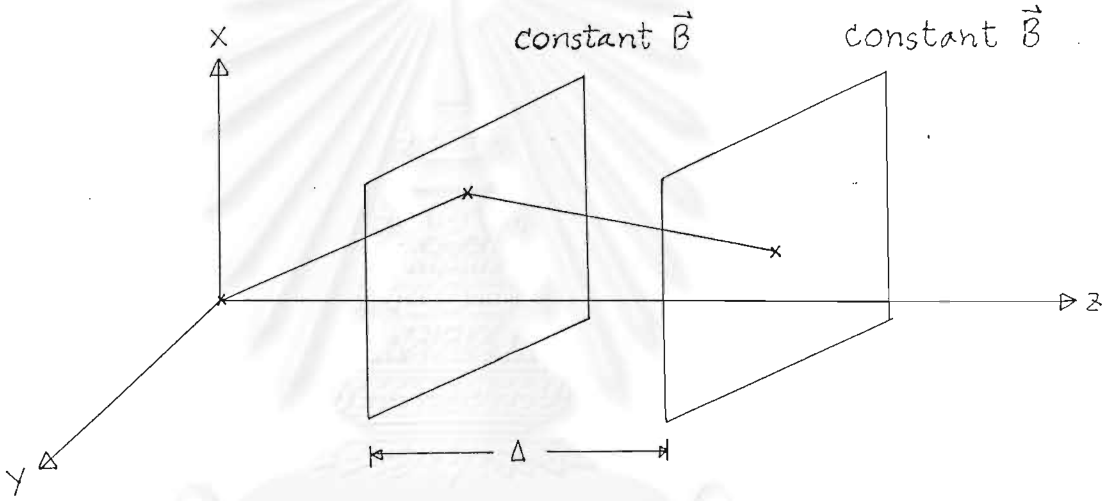

Figure 4.5.1: Three dimensional trajectories from the slab model of turbulence.

We assume that the shock wave plane, e.g., representing the solar wind termination shock, is a flat plane parallel to the y-axis and nearly parallel but slightly tilted to the z-axis. This means that the shock location in the x-direction is given by the equation

$$x_{shock}(z) = -mi + z_{cutoff}/\tan\psi, \tag{4.5.2}$$

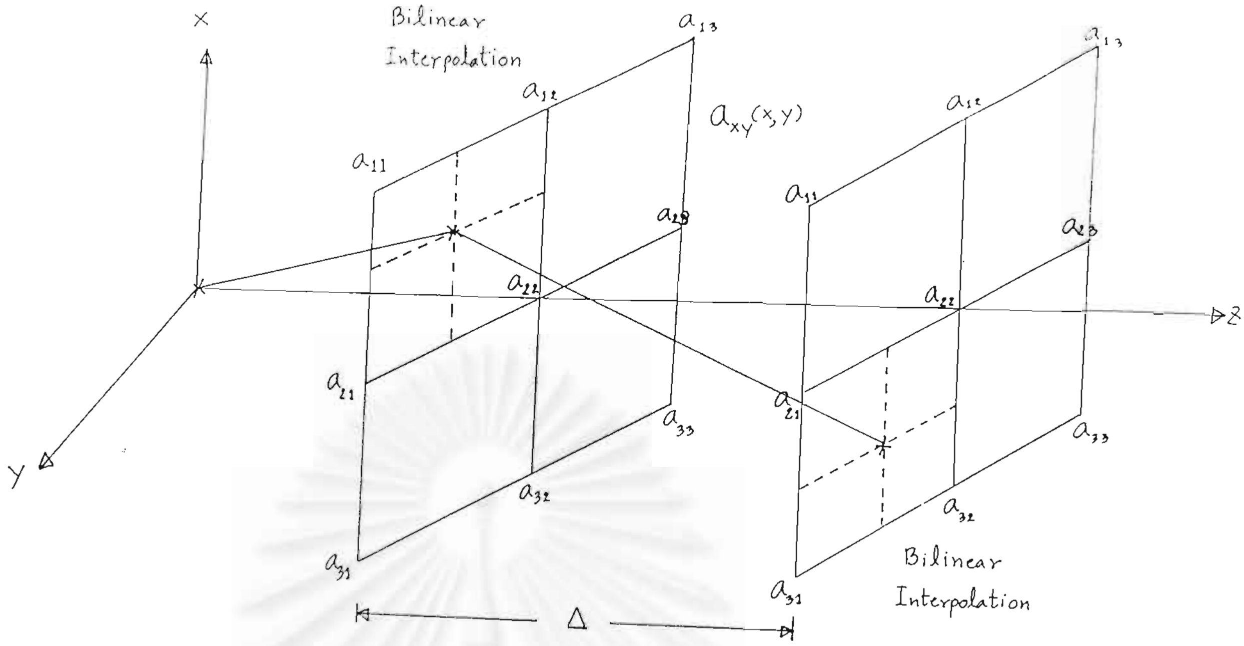

Figure 4.5.2: Three dimensional trajectories from the 2D + slab model of turbulence.

is given by the equation

$$x_{shock}(z) = -mi + z_{cutoff}/\tan\psi, \tag{4.5.2}$$

where $x_{shock}(z)$ is the x-coordinate at a given z, m is the slope of the shock wave, $m = \Delta/\tan\psi$, Δ is the grid size of our simulation in the z direction, z_{cutoff} is a crossing point between the shock and the z-axis, and ψ is the angle between the shock normal and z-direction. With a view toward the solar wind termination shock, with $r = 110$ and $R = 1$ AU (see Chapter II), we set $\tan\psi$ equal to 110.0 or ψ equal to 89.48 degrees.

An important thing that we need to calculate before simulating the trajectory of the magnetic field is the origin of the simulation. Our concept is that the origin should be sufficiently upstream of the shock that particles should not cross the before they get to the origin. In our simulation, we check this concept

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

for the first 100 iterations. If the random trajectory crosses the shock before getting to the origin for 1% of the 1D simulations or 5% of the 3D simulations of 100 iterations, we must increase this distance and check again.

When the random trajectories cross the shock from upstream to downstream, the value of x is greater than the value of x_{shock} at the same grid point, $z = i\Delta$. On the other hand, the value of x is less than the value of x_{shock} at the same grid point when crossing from downstream to upstream. We use this condition to check their crossings and are interested only in the x coordinate because the shock wave is a flat plane parallel to the y-axis. We have measured the distribution of the number of crossings (N), the angle between the magnetic field and the shock normal at each crossing (ψ), and the z-distance between two crossing points (L) for 1000 iterations for each physical situation.

4.6 Simulation Testing

4.6.1 Fourier transform testing

We use Parseval's theorem to check the normalization factor:

$$\int_{-\infty}^{\infty} F(k)G^*(k)dk = \int_{-\infty}^{\infty} f(z)g^*(z)dz \tag{4.6.1}$$

where $f(z)$ and $g(z)$ are functions which depend on the position parameter, z, and $F(k)$ and $G(k)$ are functions which depend on the wave vector parameter, k. For the special case of $F(k) = G(k)$, we get

$$\int_{-\infty}^{\infty} |G(k)|^2 dk = \int_{-\infty}^{\infty} |g(z)|^2 dz \tag{4.6.2}$$

We can change the equation 4.6.2 into a discrete form:

$$\sum_{n=0}^{N-1} |G(k_n)|^2 \Delta k = \sum_{n=0}^{N-1} |g(z)|^2 \Delta z, \tag{4.6.3}$$

where Δk and Δz are the grid spacings in wave number space and the position space, respectively. Equation 4.6.3 implies that if we calculate a summation of $|G(k)|^2 \Delta k$ before using the inverse Fourier transform, we should get the same value when we calculate the summation of $|g(z)|^2 \Delta z$ after using the inverse Fourier transform. This condition is used to test the fast Fourier transform program which is adapted from Numerical Recipes in C (Press et al., 1988) and also to check our normalization constants.

4.6.2 Total turbulence and correlation length testing

In our program, the most important thing is the random magnetic fields which are calculated by using a model of turbulence and a random phase approximation. The total turbulence, ΔB^2, and the correlation, λ_c, can be checked. The total turbulence can be recalculated from equation (3.1.7):

$$\delta B^2 = \delta B_x^2 + \delta B_y^2 \equiv < [\delta B_x(z)]^2 > + < [\delta B_y(z)]^2 > . \qquad (4.6.4)$$

We use the equation to directly calculate the total turbulence in position space (or after the inverse Fourier transform) and check the percentage error. We can also directly calculate the correlation length from equation (3.1.8) (Chapter III):

$$\lambda_c \equiv \frac{\int\limits_{-\infty}^{0} < \delta B_x(z) \delta B_x(z+z') > dz'}{\delta B_x^2(z)} \qquad (4.6.5)$$

$$\lambda_C = \frac{1}{2} \frac{\int\limits_{-\infty}^{\infty} < B_x(z) B_x(z+z') > dz'}{< B_x^2 >} \qquad (4.6.6)$$

$$\lambda_C = \frac{\frac{1}{2} \sum\limits_{j=1}^{N} < B_x(i) B_x(i+j) > \Delta z}{< B_x^2 >} = \frac{\frac{1}{2N} \sum\limits_{j=1}^{N} [\sum\limits_{i=1}^{N} B_x(i) B_x(i+j)] \Delta z}{\sum\limits_{i=1}^{N} B_x^2(i)} \qquad (4.6.7)$$

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

We use equation 4.6.8 to recalculate the correlation length and check the percent error. We can plot a relation between the correlation,

$$\frac{R_{xx}(i\Delta)}{\delta B_x^2} = \frac{1}{N\delta B_x^2}\sum_j B_x(j)B_x(i+j), \tag{4.6.9}$$

and $z = i\Delta$ as shown in Figure 4.6.1 below.

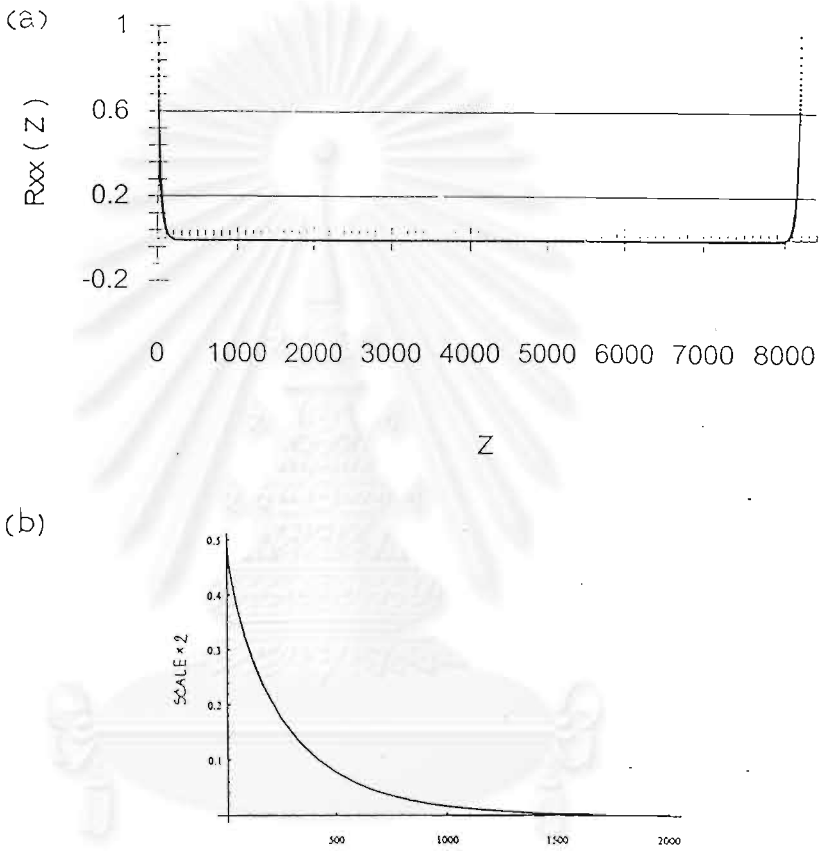

Figure 4.6.1: (a) Distribution of $R_{xx}(z)/\delta B_x^2$ vs. z. (b) Inverse Fourier transform of P_{xx} vs. z (Mathematica plot). Both of them have similar distributions, serving as a successful test of the program.

จุฬาลงกรณ์มหาวิทยาลัย

4.6.3 Crossings of a shock and statistical data testing

An important assumption of these simulations is that the random magnetic field trajectories do not cross the shock before the start of the simulation (i.e., for

$z < 0$). (Details are shown in section 4.5.) We check for crossings at z less than 0 by increasing the length of the simulation in the opposite direction and checking for field-shock crossings. If there are any crossings at $z < 0$ in 100 trials (i.e., $\geq 1\%$ trials for 1D simulations and $\geq 5\%$ trials for 3D simulations), we do not use the results because our assumption failed. In this case we increase z_{cutoff}, and also N if necessary.

Another way to check the crossings is a simple way that is the most accurate, that is, plotting the random trajectories and shock wave. We tried to use this idea to check every part of our program, such as the crossings at $z < 0$, and the random magnetic field before and after the 1D and 2D inverse Fourier transforms. Samples of these are shown below.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

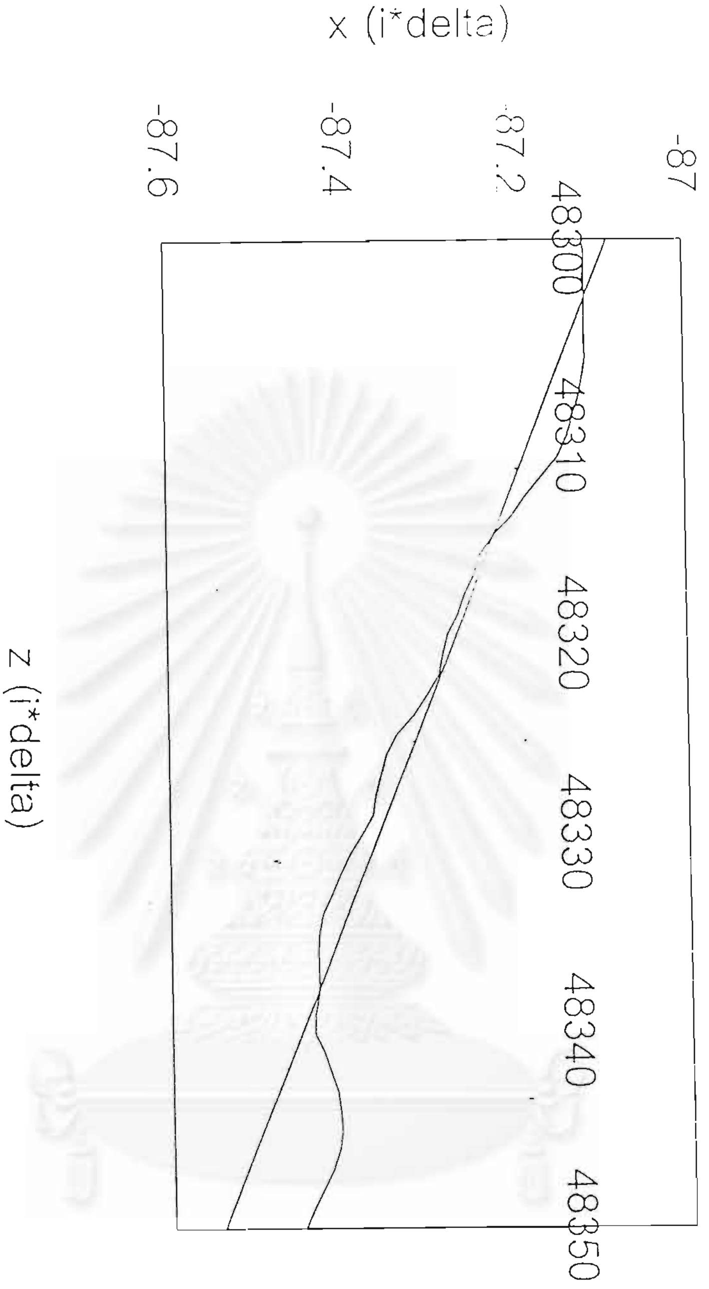

Figure 4.6.2: Crossings between random magnetic field trajectories and a shock. We can see the total number of crossings, angles of the crossings, and the distance between adjacent crossing points.

```
dbsq = 0.001000 dbsqSlab = 0.000200 dbsq2D = 0.000800

Nslab = 524288 N2D = 256


iterations = 1     Cross = 3

 no.      L[]           i          alpha[]     xcoor[]    zcoor[]

  1   48315.078  48303.000    136.985    -87.115   48303.000

  2   48338.078  48315.000    136.991    -87.224   48315.000

  3        0.000  48338.000    136.949    -87.433   48338.000

         ************************************
```

Figure 4.6.3: Calculation table for the figure above, showing for each crossing the grid point (i), x and y coordinates (xcoor[] and ycoor[]), number of crossings (Cross), upstream crossing angle (alpha[]), and distance along the shock (L[]). We can use these data to recheck all of the calculated values reported by the program for confirmation.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Chapter V

Statistical Results

In this chapter, we show all of the statistical results for one-dimensional and three-dimensional magnetic fields. We have three types of histograms, for the total number of field-shock crossings, the upstream crossing angle between the field and the shock normal, and the distance between crossings. Histograms were produced for different values of the physical parameter $\delta B^2/B_0^2$. From hybrid plasma simulations at a quasi-perpendicular shock, we found that $\delta B^2/B_0^2$ may be expect to be 1×10^{-1} due to turbulence generated by the shock. Thus we simulate random magnetic fields for $\delta B^2/B_0^2$ equal to 1×10^{-5}, 1×10^{-4}, 1×10^{-3}, 1×10^{-2}, 5×10^{-2}, and 1×10^{-1}. For each histogram, we collected statistical data for 1000 simulated turbulent magnetic fields and used a correlation length, λ_c, equal to 16. Histograms of the total number, N of field-shock crossings indicate the number of cases per value of N. Histograms of the shock crossing angle indicate the number of cases per one-degree bin. For histograms of the distance between crossings, the units are in each plot. The discussion of the results is reserved for the next chapter because we want to connect that to the conclusion. Details of simulations are shown in Appendix E. For each of our simulations, we set a different value of $\delta B^2/B_0^2$ and related parameters as shown in Tables (5.1.1) and (5.2.1).

5.1 One-Dimensional Statistical Results

Table 5.1.1 Parameters of the One-Dimensional Simulations.

Slab model		
dbsq	Nslab	ZcutTimes
1 e - 5	4096 x 64	30 x 64
1 e - 4	8192 x 32	10 x 32
1 e - 3	8192 x 32	62
1 e - 2	131072 x 8	8
5 e - 2	524288 x 8	8
1 e - 1	1048576 x 8	8

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

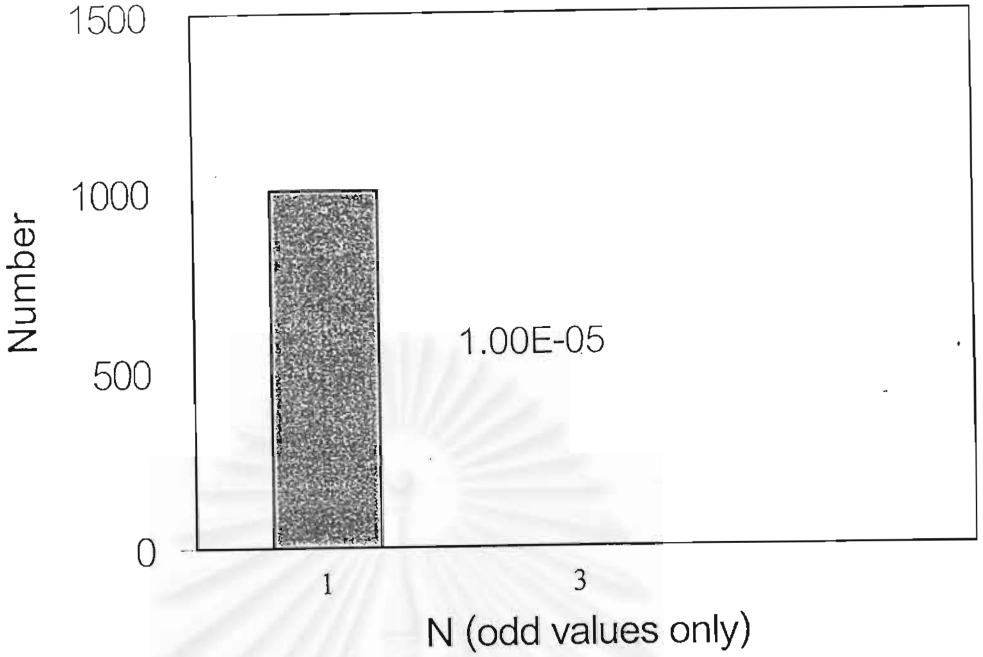

Figure 5.1.1: Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$ ($< N >= 1.0$).

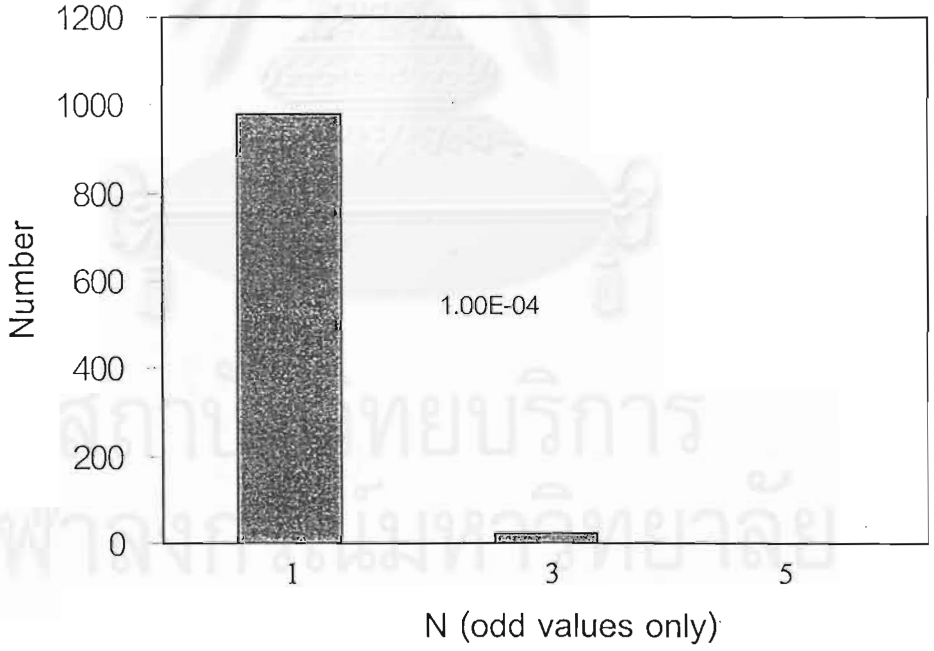

Figure 5.1.2: Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$ ($< N > \simeq 1.0$.)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

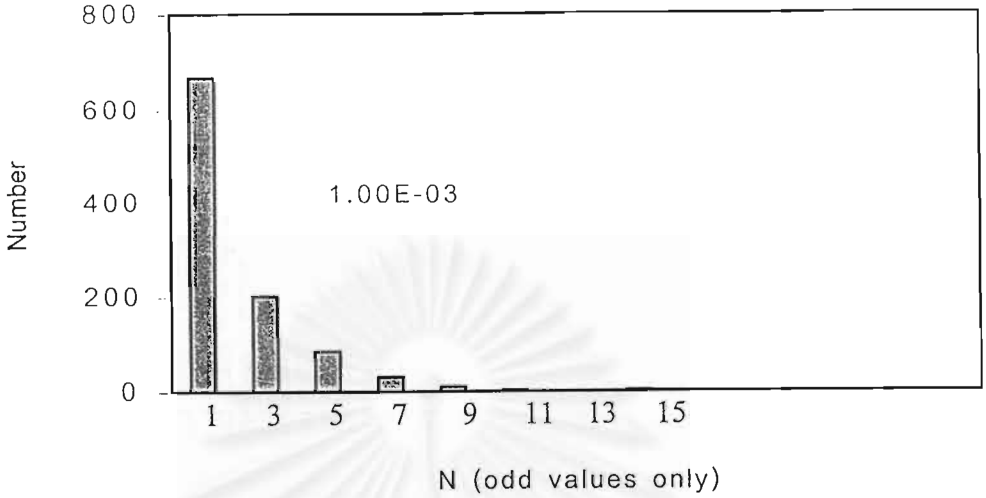

Figure 5.1.3: Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3} (<N>= 1.8)$.

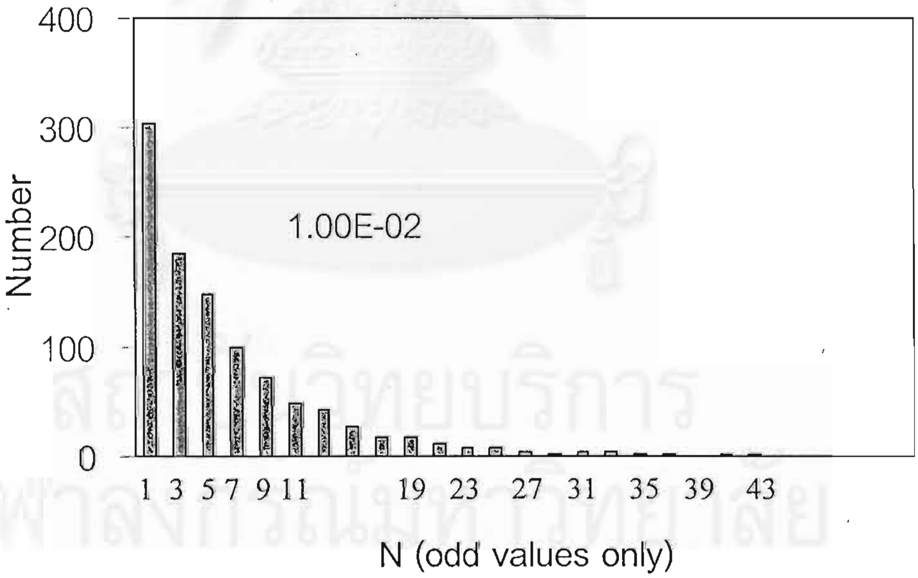

Figure 5.1.4: Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$ $(<N>= 11.2)$.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

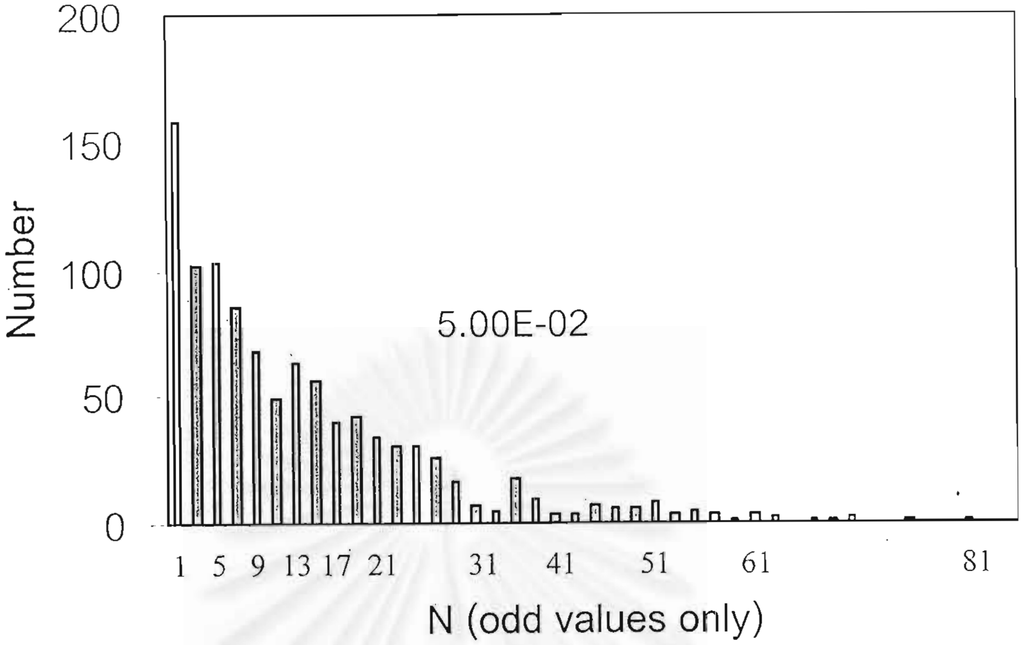

Figure 5.1.5: Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 5 \times 10^{-2}$ ($< N >= 13.2$).

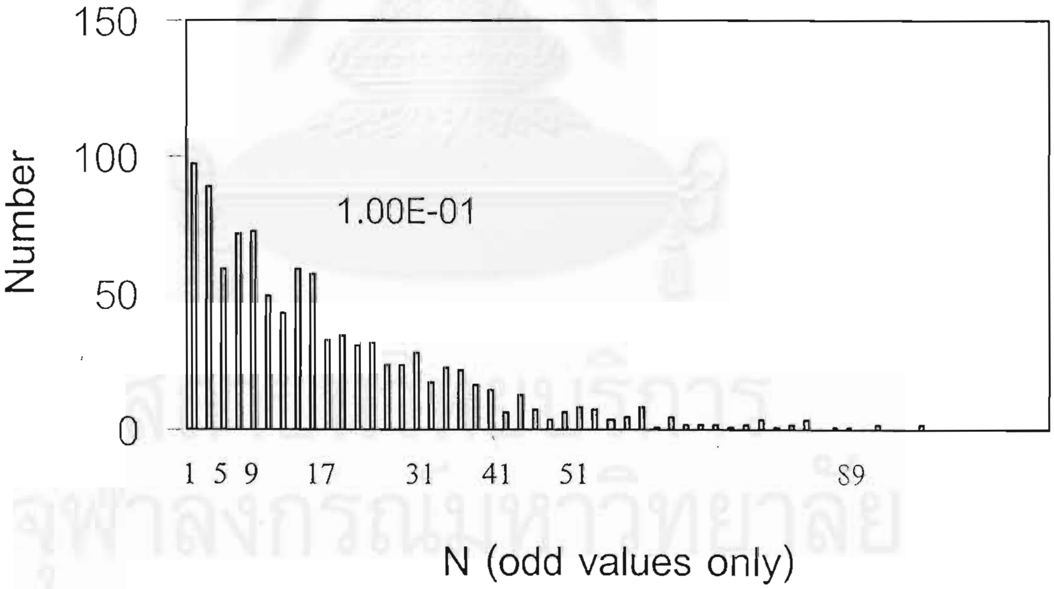

Figure 5.1.6: Histogram of the number of field-shock crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-1}$ ($< N >= 18.2$).

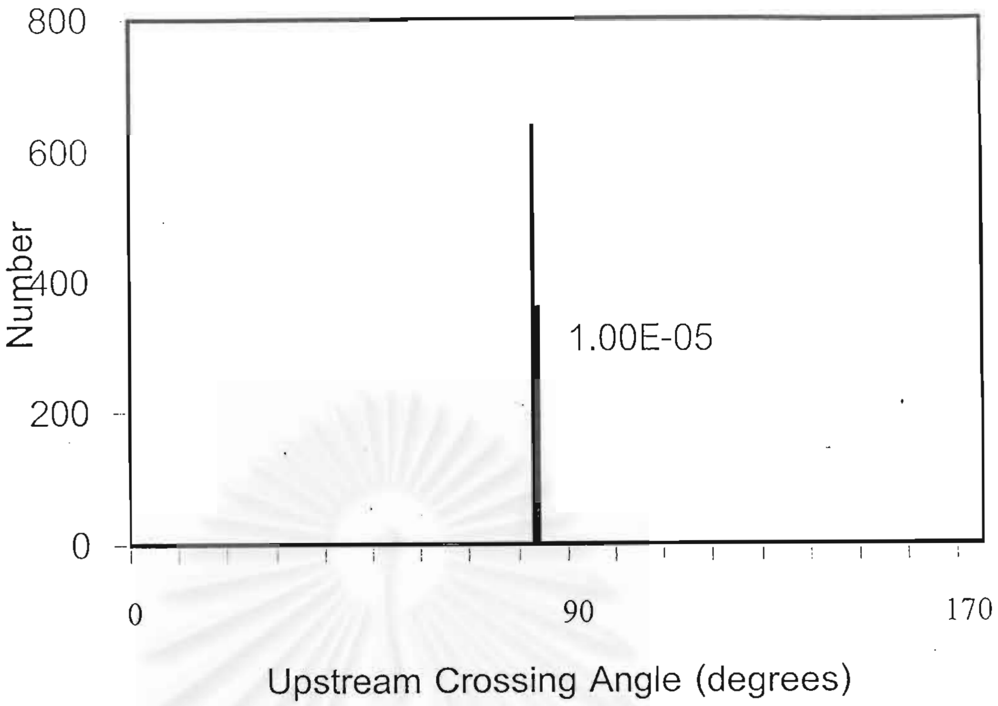

Figure 5.1.7: Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$.

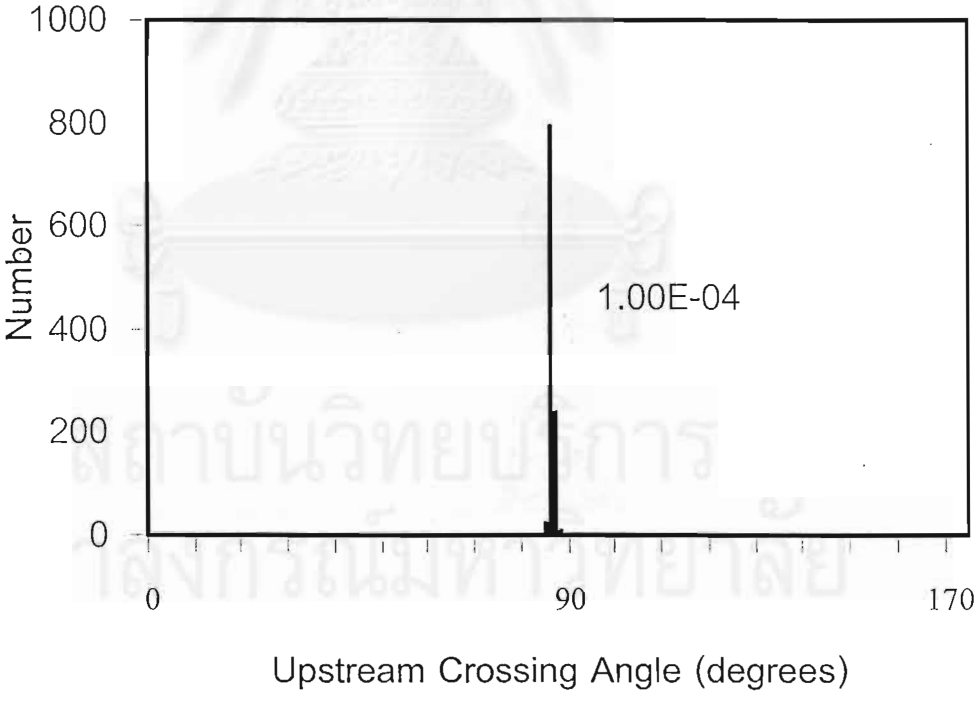

Figure 5.1.8: Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$.

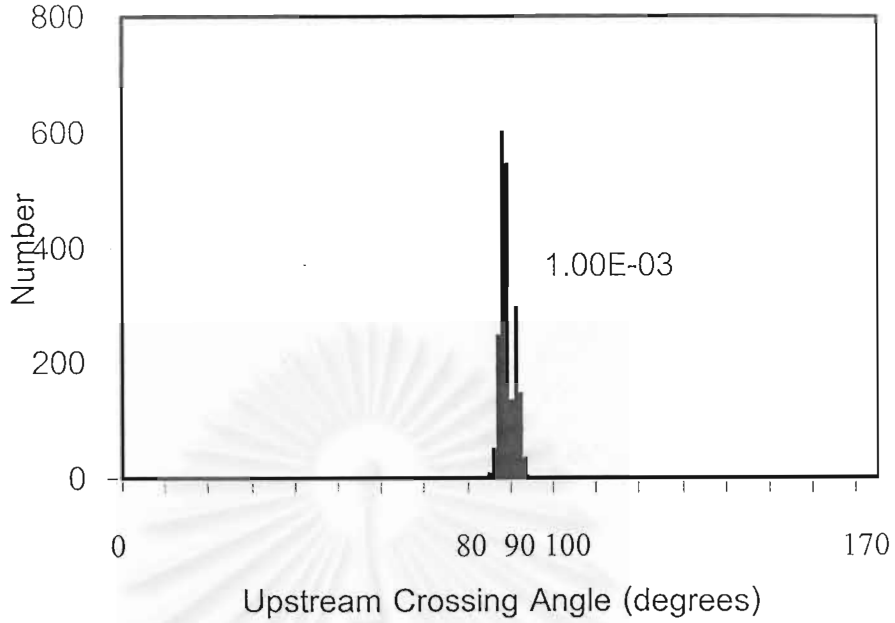

Figure 5.1.9: Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$.

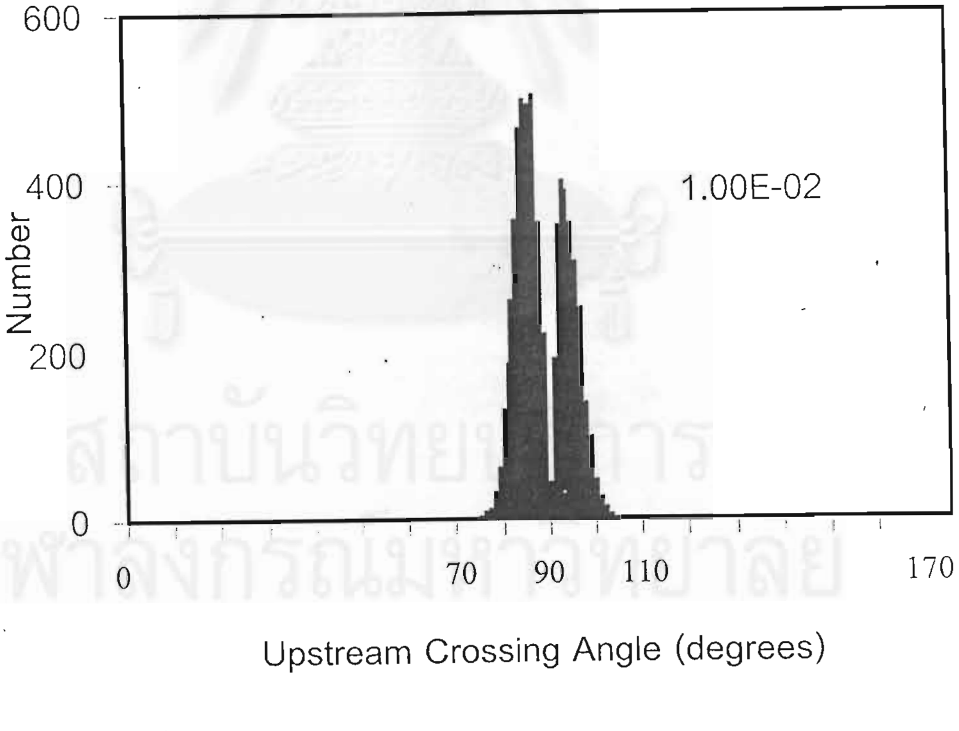

Figure 5.1.10: Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$.

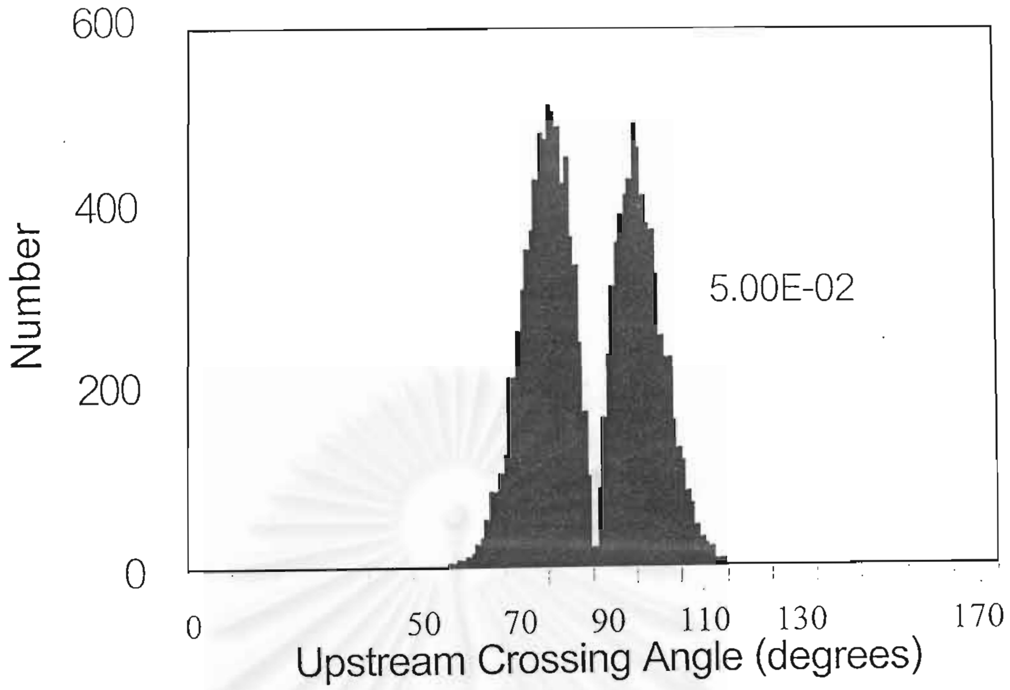

Figure 5.1.11: Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 5 \times 10^{-2}$.

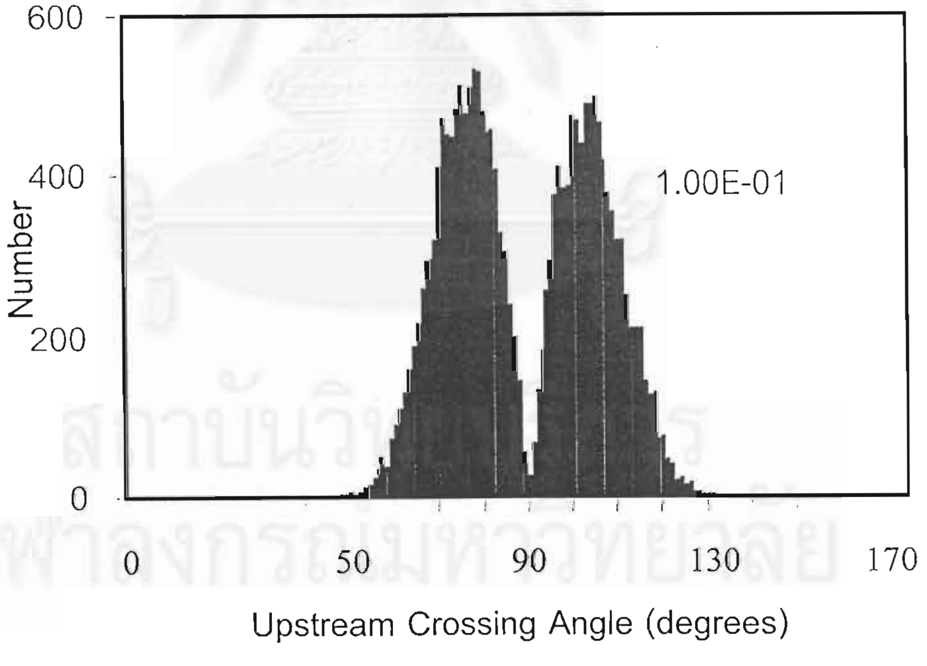

Figure 5.1.12: Histogram of the upstream crossing angle between the field and the shock normal from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-1}$.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

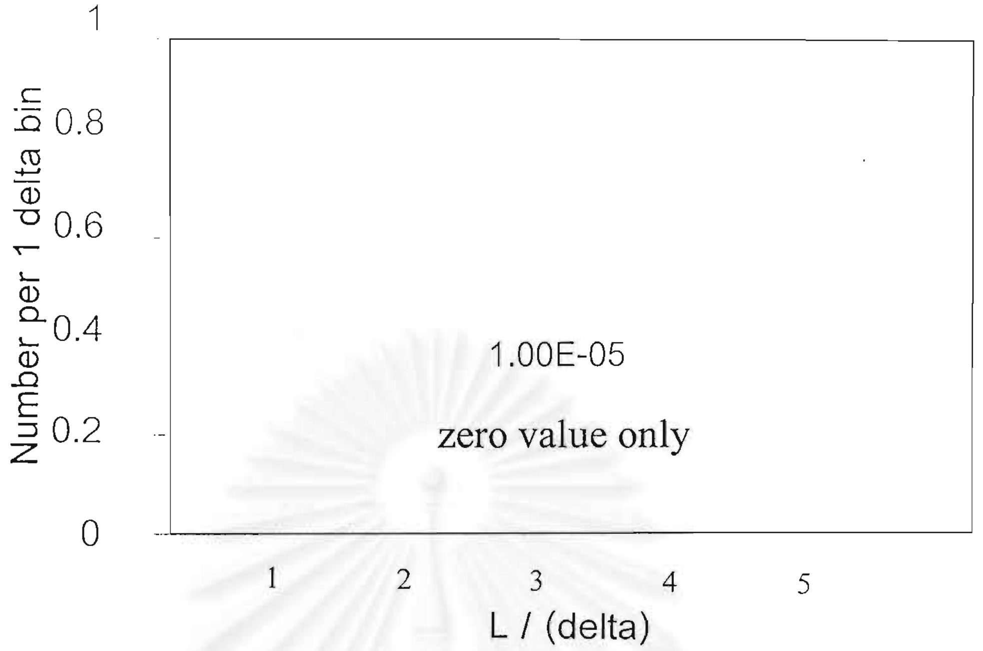

Figure 5.1.13: Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$.

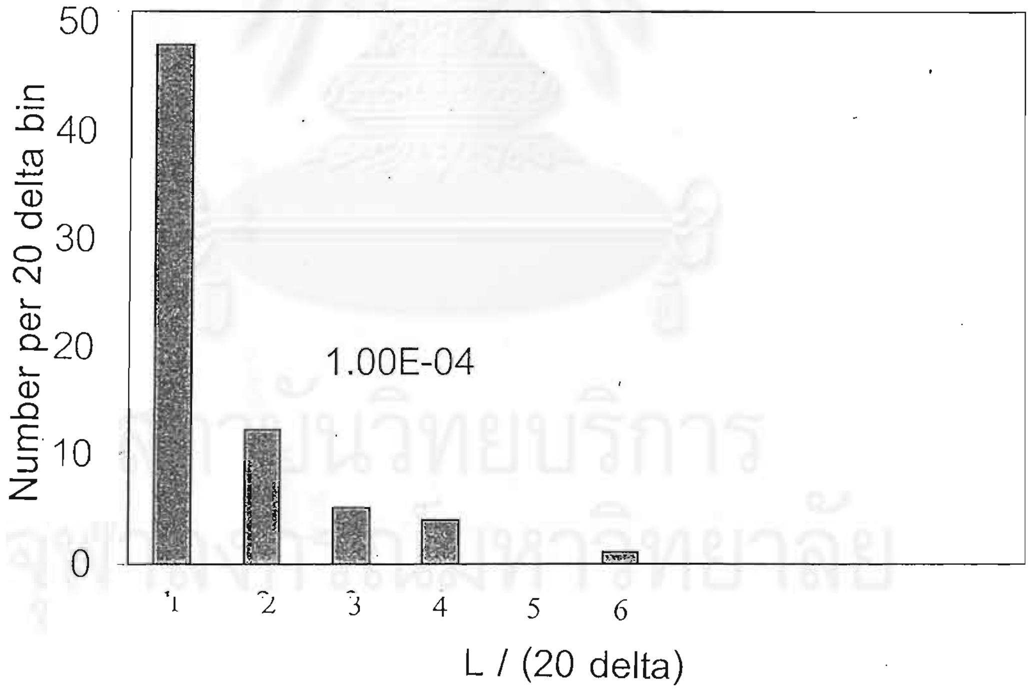

Figure 5.1.14: Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

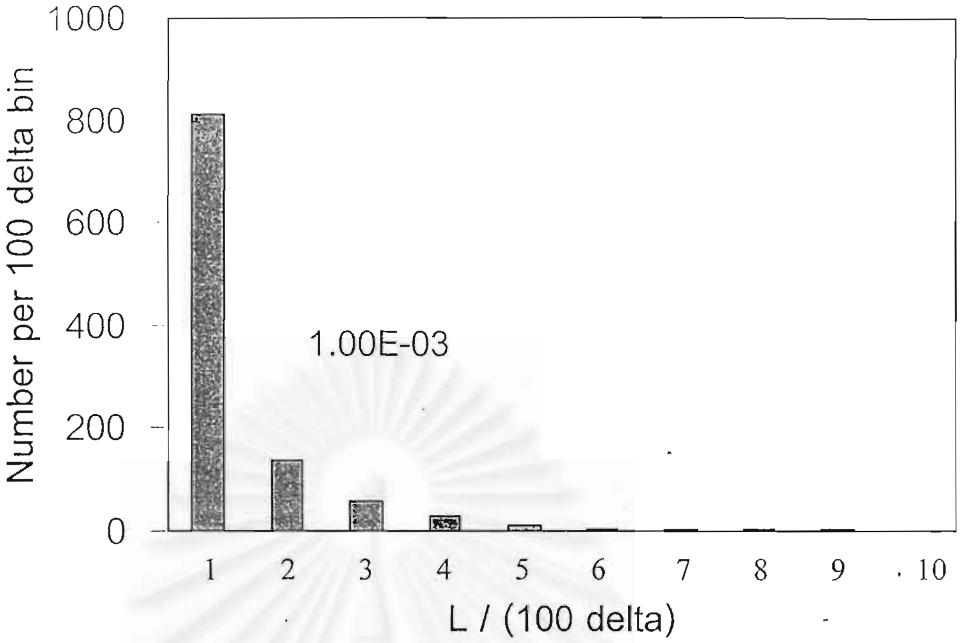

Figure 5.1.15: Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$.

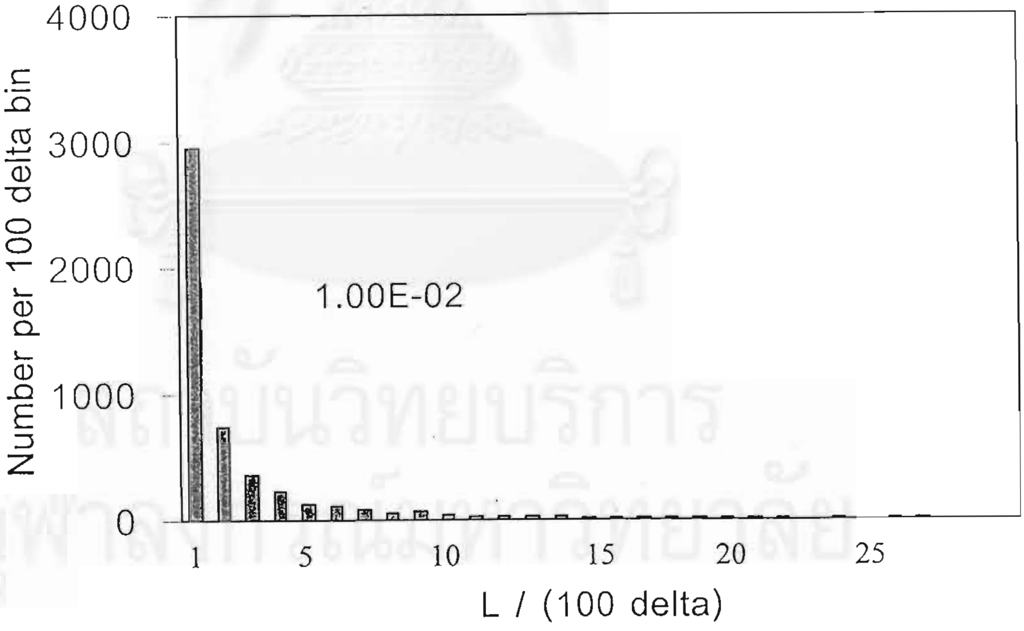

Figure 5.1.16: Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

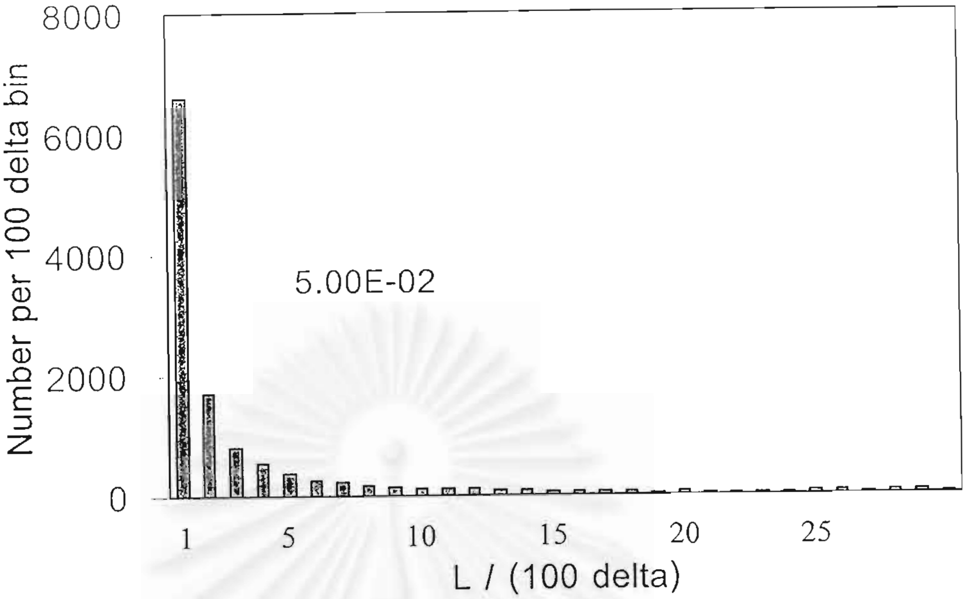

Figure 5.1.17: Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 5 \times 10^{-2}$.

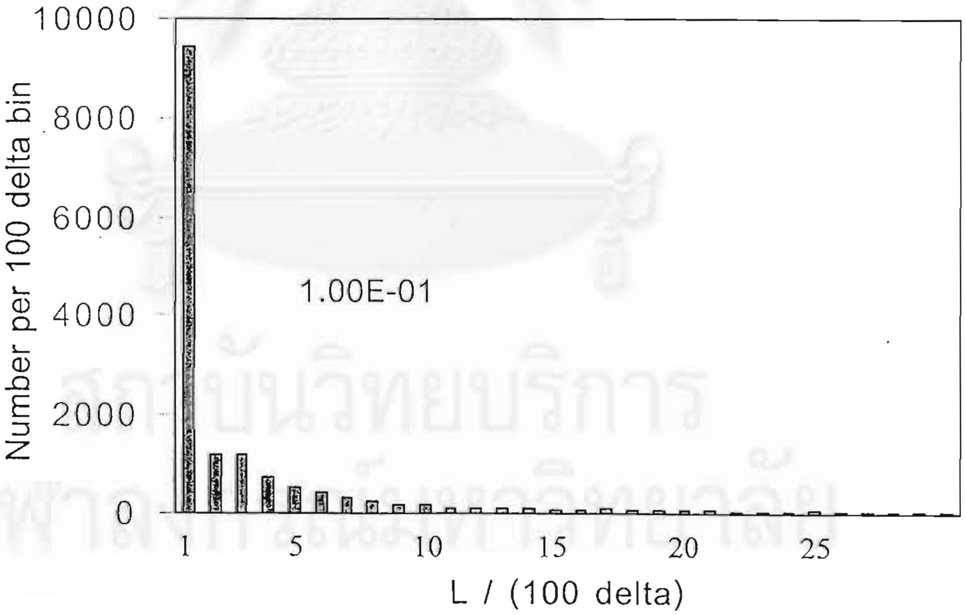

Figure 5.1.18: Histogram of the distance between crossings from the slab model using $\delta B^2/B_0^2 = 1 \times 10^{-1}$.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

5.2 Three-Dimensional Statistical Results

Table 5.2.1 Parameters of the Three-Dimensional Simulations.

2D + slab				
dbsq 2D	dbsq Slab	N2D	Nslab	ZcutTimes
8 e - 6	2 e - 6	64	4096 x 64	30 x 64
8 e - 5	2 e - 5	128	8192 x 64	10 x 64
8 e - 4	2 e - 4	256	8192 x 64	64
8 e - 3	2 e - 3	512	131072 x 8	8
4 e - 2	1 e - 2	2048	524288 x 8	8
8 e - 2	1 e - 2	2048	1048576 x 8	8

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

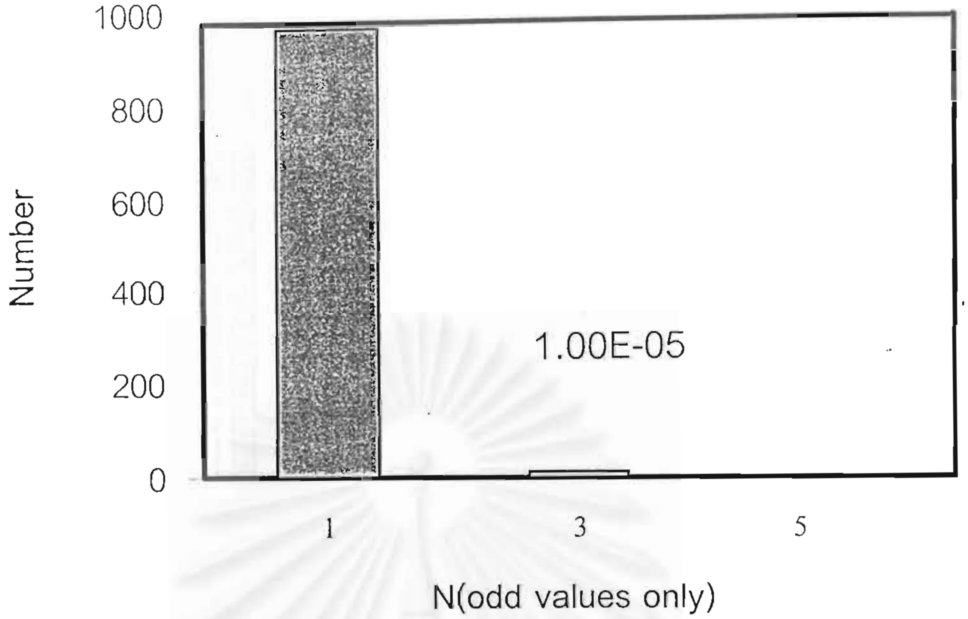

Figure 5.2.1: Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$ ($< N >= 1.0$).

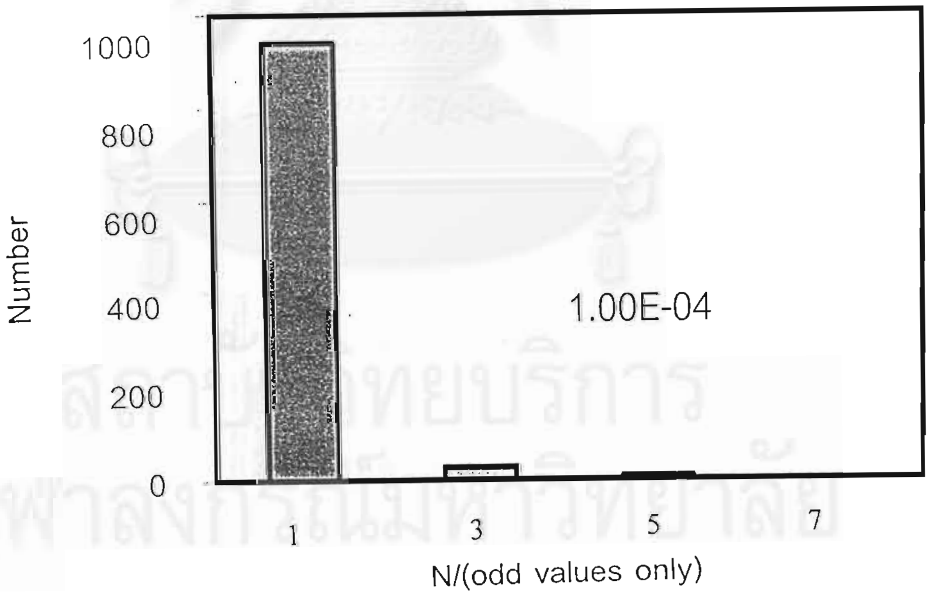

Figure 5.2.2: Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4} < N >= 1.1$)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

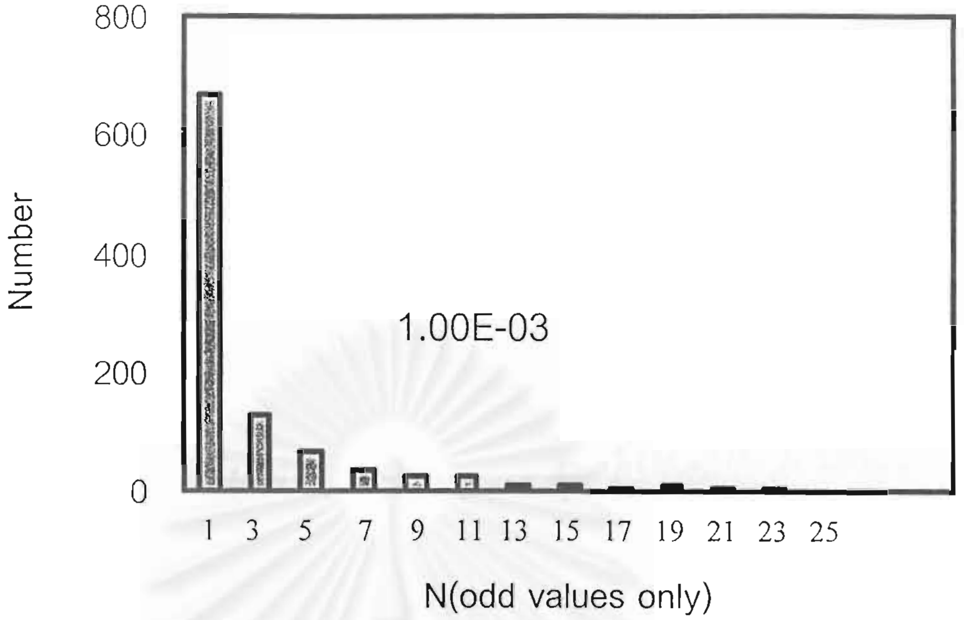

Figure 5.2.3: Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$ ($< N >= 1.6$).

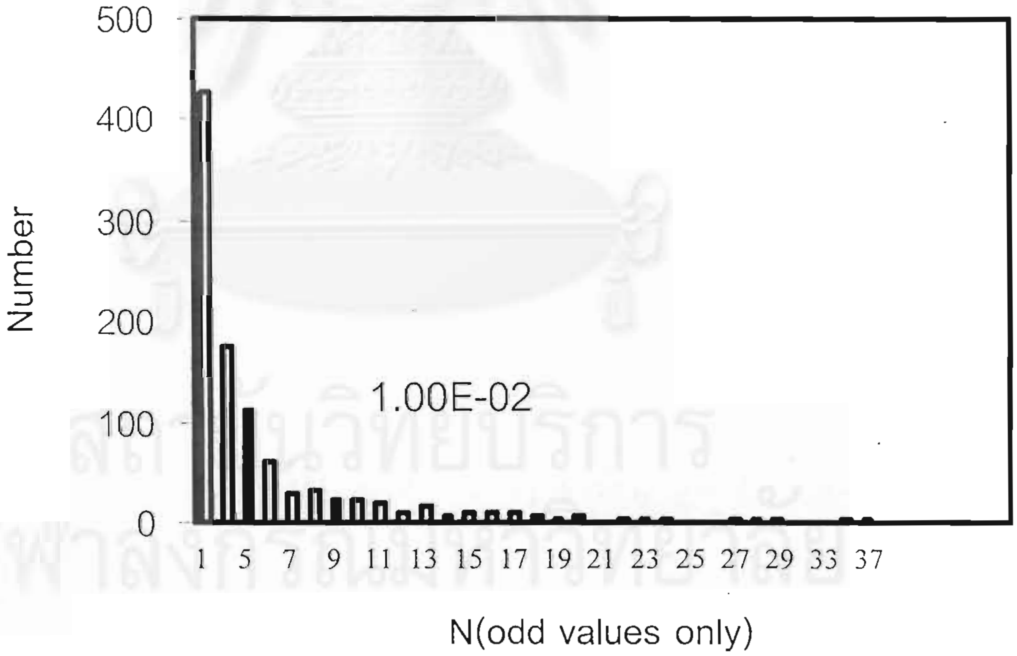

Figure 5.2.4: Histogram of the number of field-shock crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$ ($< N >= 3.5$).

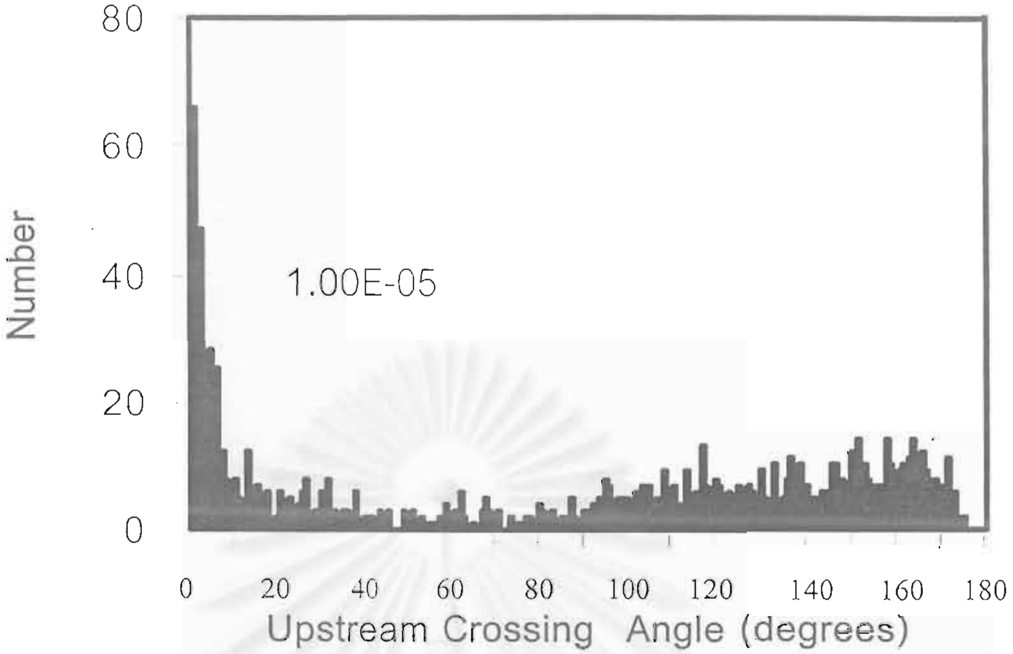

Figure 5.2.5: Histogram of the upstream crossing angle between the field and the shock normal from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$.

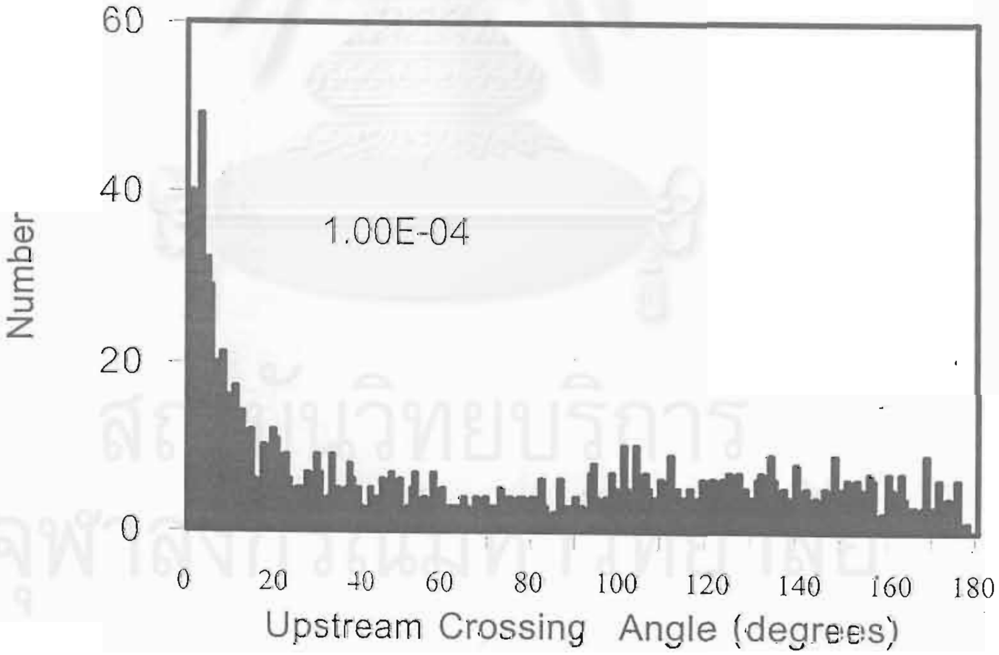

Figure 5.2.6: Histogram of the upstream crossing angle between the field and the shock normal from the 2D – slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

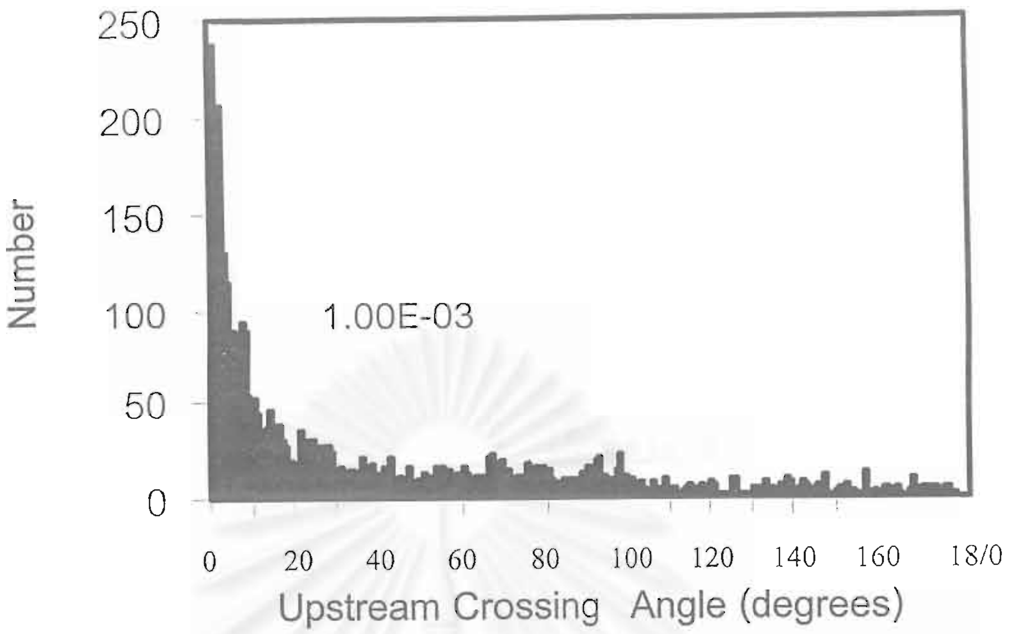

Figure 5.2.7: Histogram of the upstream crossing angle between the field and the shock normal from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$.

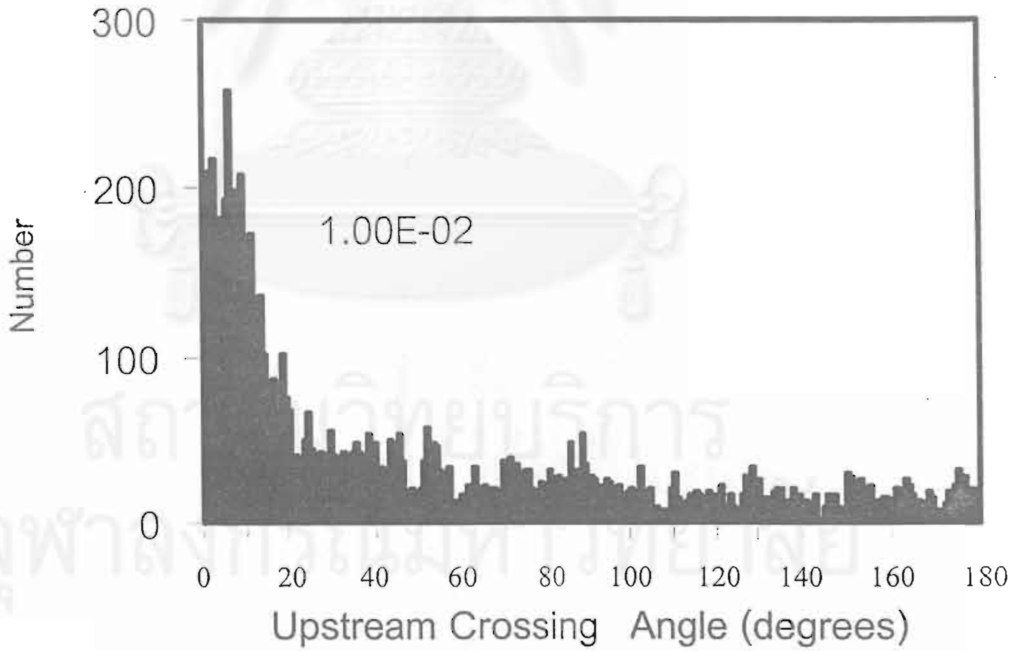

Figure 5.2.8: Histogram of the upstream crossing angle between the field and the shock normal from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$.

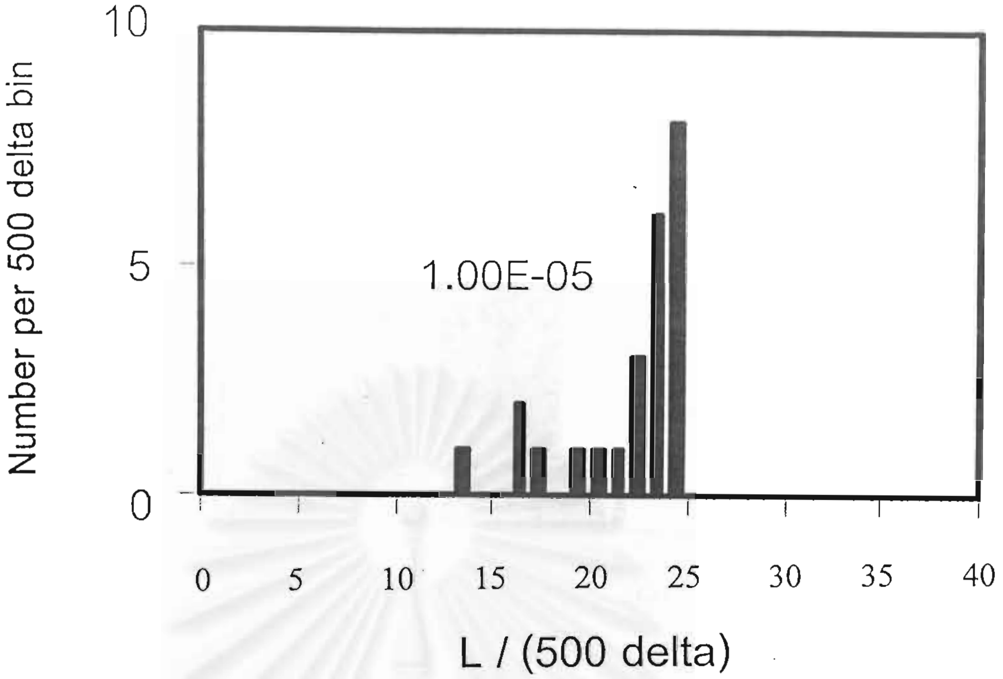

Figure 5.2.9: Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-5}$.

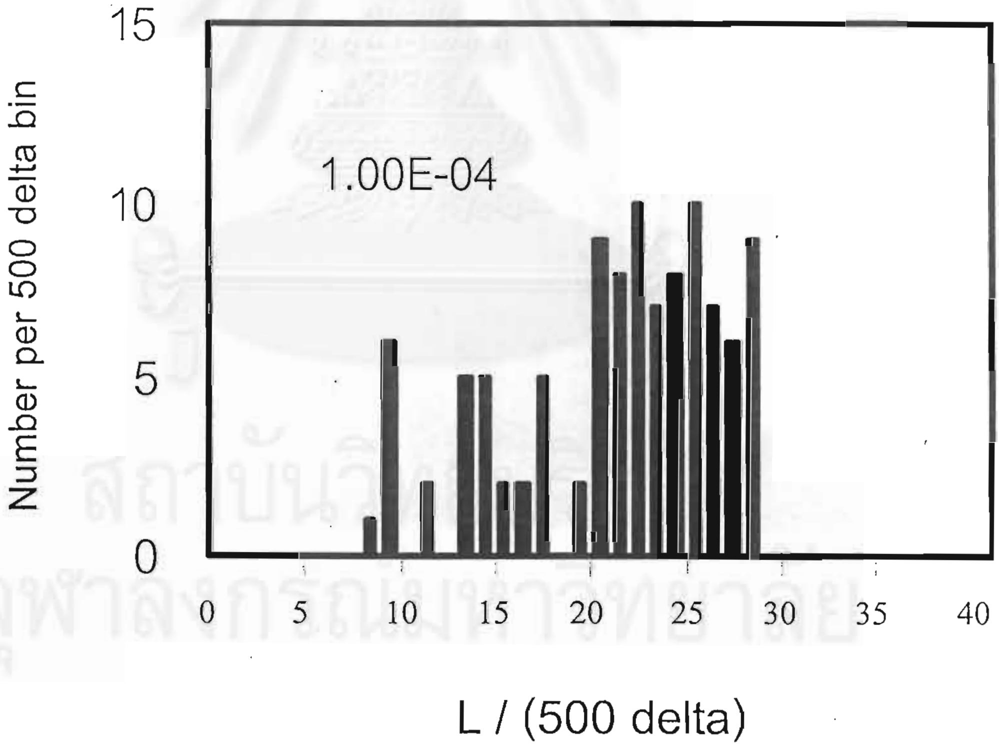

Figure 5.2.10: Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-4}$.

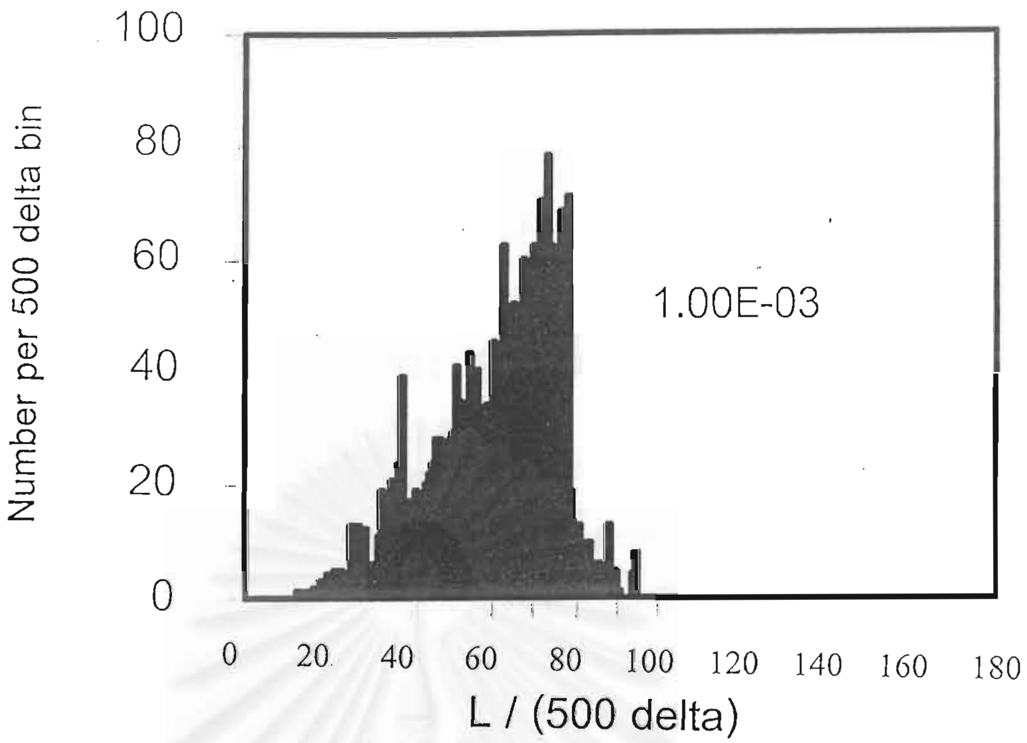

Figure 5.2.11: Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-3}$.

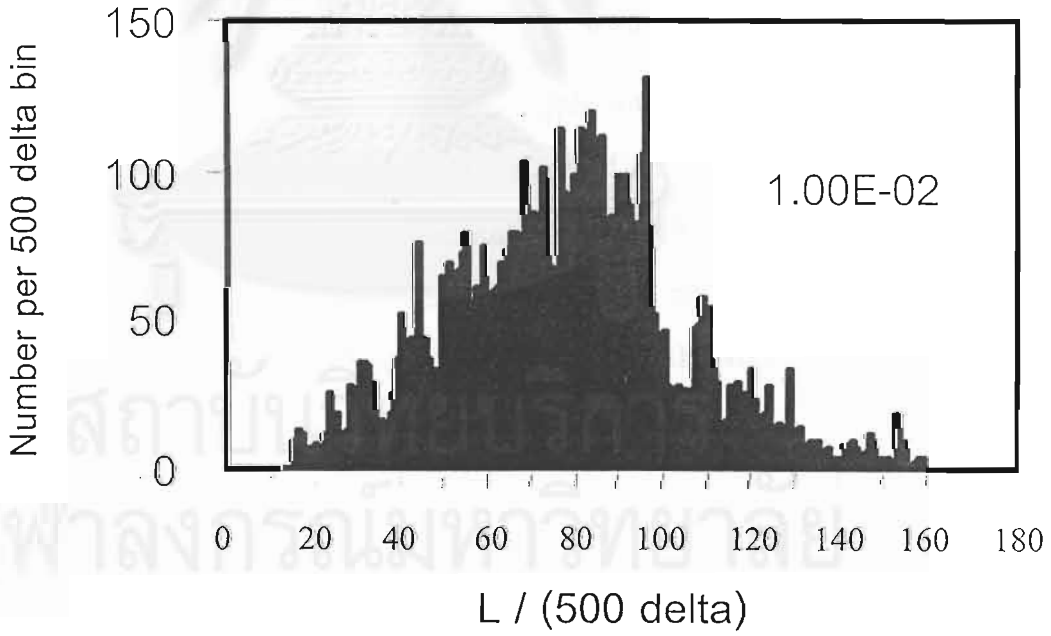

Figure 5.2.12: Histogram of the distance between crossings from the 2D + slab model using $\delta B^2/B_0^2 = 1 \times 10^{-2}$.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Chapter VI

Discussion and Conclusions

We simulated turbulent magnetic fields using the slab and 2D + slab models. The model power spectra, with random phases, were transformed to position space by using 1D and 2D inverse Fourier transforms for the slab model and the 2D model, respectively. We use a random number generator to produce random phases of the Fourier transforms of the magnetic fields. We use Euler's method to trace the random magnetic fields. We assume that the shock wave is a plane parallel to the y-axis and nearly parallel to the z-axis, the axis of the mean magnetic field. For such a nearly perpendicular shock (i.e., with $<\vec{B}>$ nearly perpendicular to the shock normal), we found that the random magnetic fields can cross and recross a shock multiple times. We believe that this has an important effect on the shock acceleration of cosmic rays, e.g., for anomalous cosmic rays at the solar wind termination shock. Therefore, for both turbulence models, we have measured the distribution of the number of field-shock crossings (N), the upstream angle between the field and the shock normal (θ), and the distance between crossings (L) for 1000 simulated turbulent magnetic fields at different values of $<\delta B^2>/B_0^2 = 10^{-5}$, 10^{-4}, 10^{-3}, 10^{-2}, 5×10^{-2}, and 10^{-1}.

6.1 Discussion of One-Dimensional Results

1) If $<\delta B^2>/B_0^2$ is small, we found that the random magnetic fields can cross a shock only one time. This is consistent with what we expect in

the limit $\delta B^2 \to 0$: then $\vec{B} = B_0\hat{z}$, fields lines are parallel to the z-axis, and therefore cross the shock only once. When this ratio increased, the number of crossings increased, too. This means that the number of crossings depends on the total turbulence and the multiple shock crossings occurred in cases with a high total turbulence. Note that hybrid plasma simulations indicate high levels of turbulence, $\delta B^2/B_0^2 \sim 0.1$.

2) For the upstream crossing angle distributions, we found that the upstream crossing angles are nearly 90 degrees. This result indicates that particle-shock encounters can potentially yield a large amount of shock-drift acceleration. However, in the case of low turbulence, i.e., $< \delta B^2 > /B_0^2 = 10^{-5}, 10^{-4}, \mathrm{or}\, 10^{-3}$, we found an asymmetry around 90 degrees because there are more crossings from upstream to downstream than vice-versa. At 90 degrees, there is a small number of crossings because when such a random flight (along the field direction) is nearly parallel to the shock and has a low probability of crossing. In cases with higher turbulence, we found symmetry around 90 degrees. This means that the random magnetic fields have the freedom to cross the shock back and forth.

3) For the distance between crossings, we found that the frequency decreased for large distances. This distribution can help us to determine the number of crossings which are sufficiently far apart for the multiple magnetic field-shock crossing mechanism (see Chapter II) to take effect, because if $L > \lambda_{||}$, particles have the freedom to go forward or backward along the random magnetic field line and gain energy each time.

6.2 Discussion of Three-Dimensional Results

1) For the number of field-shock crossings, we found essentially the same results as for one-dimensional simulations. Both of them supported the idea of

multiple crossings of a random magnetic field for high turbulence.

2) From the upstream crossing angle distributions, we found that the random magnetic field can cross a shock at any angle from 0 to 180 degrees but there are many more crossings for small values of angles. This result indicating that many particle-shock encounters will yield only a small amount of shock-drift acceleration. The very different distributions for the one-dimensional and the three-dimensional models are discussed in the next section.

3) For the distance between crossings, we found much longer distances between crossings than for the one-dimensional simulation, greatly increasing the number for which $L > \lambda_{\|}$. This means that particles have the freedom to go forward or backward along the random magnetic field line. Tanyong (1999) demonstrated that a charged particle can diffuse in a random magnetic field of the type generated in this work; thus in this case, a charged particle can diffuse along the random magnetic line and gain energy at every shock crossing.

6.3 Comparison of the Upstream Crossing Angle Distributions

There are very different results for the upstream crossing angles for one-dimensional and three-dimensional models. In one-dimensional simulations, the upstream crossing angle is nearly 90 degrees. This means that particle-shock encounters can potentially yield a large amount of shock-drift acceleration. On the other hand, we found many upstream crossings for small values of angles in the three-dimensional simulations, so that particle-shock encounters can potentially yield only a small amount of shock-drift acceleration. Why do they give the different results? The answer to this question should be related to the nature of turbulence. In the three-dimensional simulations, we used an admixture of 80% 2D turbulence and 20% slab turbulence, i.e., $< \delta B^2 >= 0.8 < \delta B_{2D}^2 > +0.2 < \delta B_{slab}^2 >$. This admixture provides a good fit to interplanetary turbulence near Earth but nobody knows the nature of the turbulence near the solar wind termination shock. Therefore, we report results for both types of turbulence, which can be viewed as two different extremes.

How do these results relate to the effects of multiple magnetic field crossings at a shock? It is an important result because if the particle-shock encounters can potentially yield only a small amount of shock-drift acceleration, they cannot explain the large drift of anomalous cosmic rays from the heliosphere equator to the poles or vice-versa. At the same time, we found a lot more crossings and larger crossing distances. This means that particles should often cross a shock and can diffuse along the random magnetic field line, which may compensate for the low crossing angles. In further work, one may be able to derive other observable consequences of the multiple magnetic crossings, e.g., the charge states of anomalous cosmic rays, so that a comparison with observations could constrain the nature of turbulence at the solar wind termination shock.

6.4 Conclusions

We found similar results for both one-dimensional and three-dimensional simulations for both the number of field-shock crossings and the distance between crossings. However, we found different results for the upstream crossing angle distribution. All of these results make us confident that the multiple magnetic field crossings occur at a nearly perpendicular shock. We found that first, a highly turbulent random magnetic field can cross a shock a large number of times. Then we expect that particles would drift every time they cross a shock. They should undergo a large total drift because they often cross a shock. Particles can diffuse along a random magnetic field line, and cross even more frequently when the crossing distance is larger than the parallel mean free path, $L > \lambda_{||}$. If random magnetic field lines cross a shock N times with a large separation, greater than the mean free path, particles will encounter the shock a total of N^2 times and gain energy every at every shock crossing before escaping downstream. The three distributions give us more understanding about the random magnetic field-shock crossings and their effects on particle acceleration.

For the special case of the solar wind termination shock, this mechanism may help explain the observationally inferred drift of anomalous cosmic rays (ACR) over much of the distance from the heliospheric equator to the poles or vice-versa. For the other quasi-perpendicular shock cases, if we know the admixture of the turbulence model, we can set this admixture in the program and run for an exact result. If we do not know the exact admixture, we can use that of interplanetary turbulence for approximate results.

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

References

Arfken, G. B. and Webber, M. J., Mathematical Methods for Physicists Acadamic Press, Inc (1995).

Chandrasekhar, S. "Stochastic Problem in Physics and Astronomy." Review of Modern Physics **15** (1943): 2-16.

Cummings, A. C., Stone, E, C., and Webber, W. R. "Changes in the Energy Spectrum of Anomalous Oxygen and Helium during 1977-1985." 19th Intern. Cosmic Ray Conf., **5**, 163.

Decker, R. B., and Vlahos, L. "Prompt Acceleration of Ions by Oblique Turbulent Shock in Solar Flares." Goddard Space Flight Center 19th Intern. Cosmic Ray Conf. **4**, (1985): 10-13.

Jackson, J. D. Classical Electrodynamics, John Wiley and Sons, New York (1975).

Jokipii, J. D. "Particle Drift, Diffusion, and Acceleration at Shocks." Astrophysical Journal Part 1, **255**, (1982): 716-720.

Khumlumlert, T. "Modeling Pulses of Solar Cosmic Rays." Master Degree Thesis, Chulalongkorn Univ, (1996): 109-116.

Lee, M. A. "The Termination Shock of the Solar Wind." Space Science Reviews **78** (1996): 109-116.

Longair, M. S. High Energy Astrophysics Cambridge Univ. Press (1997).

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Matthaeus, W. H., Bieber, J. W., and Zank, G. P. "Unquite on any front: Anisotropic Turbulence in the Solar Wind." Reviews of Geophysics 33 (1995): 609-614.

Pauls, H. L. "The Global Structure of our Heliosphere" Private communication (1997).

Pauls, H. L. The Propagation of Charged Particles in a Focusing Magnetic Field with Random Components Ph.D. thesis, Univ. of Potchefstroomse, (1993).

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. Numerical Recipes in C Cambridge Univ. Press, (1988).

Simpson, J. A. "Elemental and Isotropic Composition of the Galactic Cosmic Rays." Ann. Rev. Nucl. Part Sci **33** (1983).

Ruffolo, D. "Effect of Adiabatic Deceleration on the Focused Transport of Solar Cosmic Rays." Astrophysical Journal **442** (1995), 861-874.

Tanyong, N. Simulation of Diffusion of Cosmic Ray Particles. Senior Project, Chulalongkorn University (1998).

Terasawa, T. "Acceleration of Cosmic Rays. (I), (II)" JSPS-ICRR International Spring School'95 in Contemporary Astrophysics., eds. Yuda, T., and Hayashida, N. Tokyo: Instute for Cosmic Ray Research, University of Tokyo (1995), 172.

Tuska, E. B. Charge-sign dependent solar modulation of 1-10 GV cosmic rays. Ph.D. thesis, Univ. of Delaware, (1990).

Appendix A

One-Dimensional Random Walk

This appendix includes a brief selection from Chandrasekhar (1943), for the problem of random flights. In Chapter II, we use these concepts to show that if random magnetic field lines cross a shock N times with a large separation, greater than the particle mean free path, particles will encounter the shock a total of N^2 times before escaping downstream.

First, we begin the simplest one-dimensional problem of random flights: the problem of the random walk. The principle features of the solution of the problem of random flights in its most general form are disclosed and more clearly understood by considering first the following simplest version of the problem in one dimension (Chandrasekhar 1943).

A particle suffers displacements along a straight line in the form of a series of steps of equal length, each step being taken either in the forward or in the backward direction with equal probability 1/2. After taking N such steps the particle could be at any of the points

$$-N, -N+1, ..., -1, 0, +1, ...N-1 \text{ or } N. \qquad \text{(A.0.1)}$$

We ask: What is the probability $W(m, N)$ that the particle arrives at the point m after suffering N displacements?

We first remark that in accordance with the conditions of the problem each individual step is equally likely to be taken either in the backward or in the forward direction quite independently of the direction of all the preceding ones. Hence, all possible sequences of steps each taken in a definite direction have the same probability. In other words, the probability of any given sequence of N steps is $(1/2)^N$. The required probability $W(m, N)$ is therefore $(1/2)^N$ times the number of distinct sequences of steps which will lead to the point m after N steps. In order to arrive at m after the N steps, some $(N+m)/2$ steps should have been taken in the positive direction and the remaining $(N-m)/2$ steps when in the negative direction. (Notice that m can be even or odd only when N is even or odd, respectively.) The number of such distinct sequences is clearly

$$\frac{N!}{[N-m]!\frac{1}{2}(N-m)!} \tag{A.0.2}$$

Hence

$$W(m, N) = \frac{N!}{\left[\frac{1}{2}(N-m)\right]!\left[\frac{1}{2}(N-m)\right]!}\left(\frac{1}{2}\right)^N . \tag{A.0.3}$$

Next we shall continue the discussion of the problem of the random walk in one dimension, but with certain restrictions on the motion of the particle introduced by the presence of reflecting or absorbing walls. We shall first consider the influence of a reflecting barrier.

A.1 A Reflecting Barrier at $m = m_1$

Without loss of generality we can suppose that $m_1 > 0$. Then, the interposition of the reflecting barrier at m_1 has simply the effect that whenever the particle arrives at m_1 it has a probability of unity of retracting its step to $m_1 - 1$ when it takes the next step. We now seek the probability $W(m, N; m)$ that the particle will arrive at $m(\leq m_1)$ after N steps.

For the discussion of this problem it is convenient to trace the course of the particle in the (m, N)-plane as in Fig. A.1. In this diagram, the displacement

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

of a particle by a step means that the representative point moves upward by one unit while at the same time it suffers a lateral displacement also by one unit either in the positive or in the negative direction.

In the absence of a reflecting wall at $m = m_1$ the probability that the particle arrives at m after N steps is of course given by Eq. (A.0.3). However the presence of the reflecting wall requires $W(m, N)$ according to Eq. (A.0.3) to be modified to take account of the fact that a path reaching m after n reflections must be counted 2^n times since at each reflection it has a probability unity of retracing its step. It is now seen that we can take account of the relevant factors by adding to $W(m, N)$ the probability of $W(2m_1 - m, N)$ of arriving at the "image" point $(2m_1 - m)$ after N steps (also in the absence of the reflecting wall), i.e.,

$$W(m, N; m_1) = W(m, N) + W(2m_1 - m, N). \tag{A.1.1}$$

A.2 Absorbing Wall at $m = m_1$

We shall now consider the case when there is a perfectly absorbing barrier at $m = m_1$. The interposition of the perfect absorber at m_1 means that whenever the particle arrives at m_1 at once becomes incapable of suffering further displacements. There are two questions which we should like to answer under these circumstances. The first is the analog of the problems we have considered so far, namely the probability that the particle arrives at $m(\leq m_1)$ after taking N steps. The second question which is characteristic of the present problem concerns the average rate at which the particle will deposit itself on the absorbing screen.

Considering first the probability $W(m, N; m_1)$, it is clear that in counting the number of distinct sequences of steps which lead to m we should be careful to exclude all sequences which include even a single arrival to m_1. In other words , if we first count all possible sequences which lead to m in the absence of the absorbing screen we should then exclude a certain number of "forbidden" sequences. It is evident, on the other hand, that every such forbidden sequence uniquely defines another sequence leading to the image $(2m_1 - m)$ of m on the line $m = m_1$ in the (m, N)-plane (see Fig. A.1) and conversely. By reflecting about the line $m = m_1$ the part of a forbidden trajectory above its last point of contact with the $m = m_1$ before arriving at m we are led to a trajectory leading to the image point, and conversely for every trajectory leading to m (since any trajectory leading to $2m_1 - m$ must necessarily cross the line $m = m_1$). Hence,

$$W(m, N; m_1) = W(m, N) - W(2m_1 - m, N). \tag{A.2.1}$$

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Appendix B

Fast Fourier Transform[1]

B.1 Discrete Fourier Transform

Before we discuss a discrete Fourier transform, we represent a Fourier transform equations as follows:

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{2\pi ift}dt \tag{B.1.1}$$

$$h(t) = \int_{-\infty}^{\infty} H(f)e^{-2\pi ift}dt \tag{B.1.2}$$

which can also be written

$$H(f) = \int_{-\infty}^{\infty} h(t)e^{2\pi ift}dt \tag{B.1.3}$$

$$h(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} H(f)e^{-2\pi ift}dt \tag{B.1.4}$$

where $\omega \equiv 2\pi f$. We now estimate the Fourier transform of a function from a finite number of its sampled points. Suppose that we have N consecutive sampled values.

$$h_k \equiv h(t_k), t_k \equiv k\Delta, k = 0, 1, 2, \ldots, N-1 \tag{B.1.5}$$

so that the sampling interval is Δ. To make things simpler, let us also suppose that N is even. If the function $h(t)$ is nonzero only in a finite interval of time,

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

---

[1] from Press et al. (1988)

then that whole interval of time is supposed to be contained in the range of the N points given. Alternatively, if the function $h(t)$ goes on forever, that the sampled points are supposed to be at least "typical" of what $h(t)$ looks like at all other times.

With N numbers of input, we will evidently be able to produce no more than N independent numbers of output. So, instead of trying to estimate the Fourier transform $H(f)$ at all values of f in the range $-f_c$ to f_c, let us seek estimates only at the discrete values

$$f_n = \frac{n}{N\Delta}, n = -\frac{N}{2}, ..., \frac{N}{2}. \tag{B.1.6}$$

The extreme values of n in equation (B.1.6) correspond exactly to the lower and upper limits of the Nyquist critical frequency range. If you are really on the ball, you will have noticed that there are $N+1$, not N values of n in equation (B.1.6); it will turn out that the two extreme values of n are not independent (in fact they are equal), but all the others are. This reduces the count to N.

The remaining step is to approximate the integral in equation (B.1.1) by a discrete sum:

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f_n t} dt \approx \sum_{k=0}^{n-1} h_k e^{2\pi i f_n t_k} \Delta = \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N}. \tag{B.1.7}$$

Here equations (B.1.5) and (B.1.6) have been used in the final equality. The final summation in (B.1.7) is called the discrete Fourier transform of the N points h_k. Let us denote it by H_n ,

$$H_n = \sum_{k=0}^{N-1} h_k e^{2\pi i k n/N}. \tag{B.1.8}$$

The discrete Fourier transform maps N complex numbers (the h_k's) into N complex numbers (the H_n's). It does not depend on any dimensional parameter, such as the time scale Δ. The relation (B.1.7) between the discrete Fourier transform of a set of a set of numbers and their continuous Fourier transform when they are viewed as samples of a continuous function sampled at an interval Δ can be rewritten as

$$H(f_n) \approx \Delta H_n. \tag{B.1.9}$$

where f_n is given by equation (B.1.6). Up to now we have taken the view that the index n in equation (B.1.8) varies from $-N/2$ to $N/2$. You can easily see, however, that equation (B.1.8) is periodic in n, with period N. Therefore, $H_{-n} = H_{N-n}$ $n = 1, 2....$ With this conversion in mind, one generally lets the n in H vary from 0 to $N-1$ (one complete period). Then n and k (in h_k) vary exactly over the same range, so the mapping of N numbers into N numbers is manifest. When this convention is followed, you must remember that zero frequency corresponds to $n = 0$, positive frequencies $0 < f < f_c$ correspond to values $1 \leq n \leq N/2 - 1$, while negative frequencies $-f_c < f < 0$ correspond to $N/2 + 1 \leq n \leq N - 1$. The value $n = N/2$ corresponds to *both* $f = f_c$ and $f = -f_c$.

The discrete Fourier transform has symmetry properties almost exactly the same as the continuous Fourier transform. For example, all the symmetries in the table following equations (B.1.3) and (B.1.4) hold if we read h_k for $h(t)$, H_n for $H(f)$, and H_{N-n} for $H(-f)$. (Likewise, "even" and "odd" in time refer to whether the values h_k at k and $N-k$ are identical or the negative of each other.)

The formula for the discrete inverse Fourier transform, which recovers the set of h_k's exactly from the H_n's is:

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi ikn/N}. \tag{B.1.10}$$

Notice that the only differences between equation (B.1.10) and equation (B.1.8) are (i) changing the sign in the exponential, and (ii) dividing the answer by N. This means that a routine for calculating discrete Fourier transforms can also,

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

with slight modification, calculate the inverse transforms. The discrete form of Parseval's theorem is

$$\sum_{k=0}^{N-1} \mid h_k \mid^2 = \frac{\cdot}{1} N \sum_{n=0}^{N-1} \mid H_n \mid^2 \qquad \text{(B.1.11)}$$

B.2 Fast Fourier Transform (FFT)

How much computation is involved in computing the discrete Fourier transform equation (B.1.8) of N points? For many years, until the mid-1960s, the standard answer was this: Define W as the complex number

$$W \equiv e^{2\pi i/N} \qquad \text{(B.2.1)}$$

Then equation (B.1.8) can be written as

$$H_n = \sum_{k=0}^{N-1} W^{nk} h_k \qquad \text{(B.2.2)}$$

In other words, the vector of h_k's is multiplied by a matrix whose $(n,k)^{th}$ element is the constant W to the power $n \times k$. The matrix multiplication produces a vector result whose components are the H_n's. This matrix multiplication evidently requires N^2 complex multiplications, plus a smaller number of operations to generate the required powers of W. So, the discrete Fourier transform appears to be an $O(N^2)$ process. These appearances are deceiving! The discrete Fourier transform can, in fact be computed in $O(N \log_2 N)$ operations with an algorithm called the Fast Fourier Transform, or FFT. The difference between $N \log_2 N$ and N^2 is immense. With $N = 10^6$, for example, it is the difference between, roughly, 30 seconds of CPU time and 2 weeks of CPU time on a microsecond cycle time computer. The existence of an FFT algorithm became generally known only in the mid-1960s, which in turn had been proved by F. L. Garwin of IBM Yorktown Heights Research Center. Retrospectively, we now know that a few clever

จุฬาลงกรณ์มหาวิทยาลัย

individuals had independently discovered, and in some cases implemented, fast Fourier transforms as many as 20 years previously (see Brigham for references).

One of the earliest discoveries of the FFT, that of Danielson and Lanczos in 1942, still provides one of the clearest derivations of the algorithm. Danielson and Lanczos showed that a discrete Fourier transform of length N can be rewritten as the sum of two discrete Fourier transforms, each of length $N/2$. One of the two is formed from the even-numbered points of the original N, the other from the odd-numbered points. The proof is simply this:

$$F_k = \sum_{j=0}^{N-1} e^{2\pi ijk/N} f_j = \sum_{j=0}^{N/2-1} e^{2\pi ik(2j)/N} f_{2j} + \sum_{j=0}^{N/2-1} e^{2\pi ik(2j+1)/N} f_{2j+1} \qquad \text{(B.2.3)}$$

$$F_k = \sum_{j=0}^{N/2-1} e^{2\pi ik/(N/2)} f_{2j} + W^k \sum_{j=0}^{N/2-1} e^{2\pi ik/(N/2)} f_{2j+1} = F_k^e + W^k F_k^o. \qquad \text{(B.2.4)}$$

In the last line, W is the same complex constant as in equation (B.2.1), F_k^e denotes the k^{th} component of the Fourier transform of length $N/2$ formed from the even components of the original f_j's, while F_k^o is the corresponding transform of length $N/2$ formed from the odd components. Notice also that k in the last line of (B.2.4) varies from 0 to N, not just to $N/2$. Nevertheless, the transforms F_k^e and F_k^o are periodic in k with length $N/2$. So each is repeated thorough two cycles to obtain F_k.

The wonderful thing about the Danielson-Lanczos Lemma is that it can be used recursively. Having reduced the problem of computing F_k to that of computing F_k^e and F_k^o, we can do the same reduction of F_k^e to the problem of computing the transform of its $N/4$ even-numbered input data and $N/4$ odd-numbered data. In other words, we can define F_k^{ee} and F_k^{eo} to be the discrete Fourier transforms of the points which are respectively even-even and even-odd on the successive subdivisions of the data.

Although there are ways of treating other cases, by far the easiest case is the one in which the original N is an integer power of 2. In fact, we categorically recommend that you only use FFTs with N a power of two. If the length of your data set is not a power of two, pad it with zeros up to the next power of two. (We will give more sophisticated suggestions in subsequent sections below.) With this restriction on n, it is evident that we can continue applying the Daneilson-Lanczos Lemma until we have subdivided the data all the way down to transforms of length 1. What is the Fourier transform of length one? It is just the identity operation that copies its one input number into its one output slot. In other words, for every pattern of e's and o's (numbering $\log_2 N$ in all), there is a one-point transform that is just one of the input numbers f_n.

$$F_k^{eoeeoeo\cdots oee} = f_n, \text{for some} n. \tag{B.2.5}$$

(Of course this one-point transform actually does not depend on k, since it is periodic in k with period 1.)

The next trick is to figure out which value of n corresponds to which pattern of e's and o's in equation (B.2.5). The answer is: reverse the pattern of e's and o's, then let $e = 0$ and $o = 1$, and you will have, in binary, the value of n. Do you see why it works? It is because the successive subdivisions of the data into even and odd are tests of successive low-order (least significant) bits of n. This idea of bit reversal can be exploited in a very clever way which, along with the Danielson-Lanczos Lemma, makes FFTs practical: Suppose we take the original vector of data f_j and rearrange it into bit-reversed order (see Figure B.2.1), so that the individual numbers are in the order not of j, but of the number obtained by bit-reversing j. Then the bookkeeping on the recursive application of the Danielson-lanczos Lemma becomes extraordinarily simple. The points as given are the one-point transforms. We combine adjacent pairs to get two-point transforms, then combine adjacent pairs of pairs to get 4-point transforms, and so on, until the first and second halves of the whole data set are combined into the final transform. Each combination takes of order N operations, and there are evidently $\log_2 N$ combinations, so the whole algorithm is of order $N \log_2 N$ (assuming, as is the case, that the process of sorting into bit-reversed order is no greater than order $N \log_2 N$).

This, then, is the structure of an FFT algorithm: It has two sections. The first section sorts the data into bit-reversed order. Luckily this takes no additional storage, since it involves only swapping pairs of elements. (If k_1 is the bit reverse of k_2, then k_2 is the bit reverse of k_1.) The second section has an outer loop which is executed $\log_2 N$ times and calculates, in turn, transforms of length. For each stage of this process, two nested inner loops range over the sub transforms already computed and the elements of each transform, implementing the Danielson-Lanczos Lemma. The operation is made more efficient by restricting external calls for trigonometric sines and cosines to the outer loop, where they are made only $\log_2 N$ times. Computation of the sines and cosines of multiple angles is through a simple recurrence relation in the inner loops.

The FFT routine given below is based on one originally written by N. Brenner of Lincoln Laboratories. The input quantities are the number of complex data points (nn), the data array (data[1..2*nn]), and isign, which should be set to either ±1 and is the sign of i in the exponential of equation (B.1.8). When isign is set to -1, the routine thus calculates the inverse transform (B.1.10) - except that it does not multiply by the normalizing factor $1/N$ that appears in that equation. You can do that yourself.

Notice that the argument nn is the number of complex data points. The actual length of the real array (data[1..2*nn]) is 2 times nn, with each complex value occupying two consecutive locations. In other words, data[1] is the real

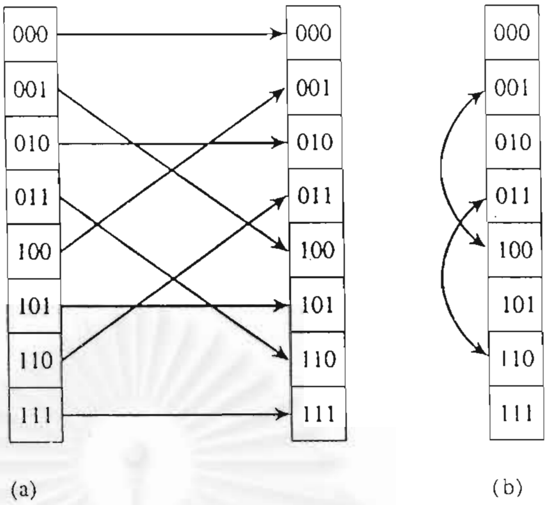

Figure B.2.1: Reordering an array (here of length 8) by bit reversal, a) between two arrays, versus b) in place. Bit reversal reordering is a necessary part of the Fast Fourier Transform (FFT) algorithm.

part of f_0, data[2] is the imaginary part of f_0, and so on up to data[2*nn-1], which is the real part of f_{N-1}, and data[2*nn], which is the imaginary part of f_{N-1}. The FFT routine gives back the F_n's packed in exactly the same fashion, as nn complex numbers. The real and imaginary parts of the zero frequency component F_0 are in data[1] and data[2]; the smallest nonzero negative frequency has real and imaginary parts in data[2*nn-1] and data [2*nn]. Positive frequencies increasing in magnitude are stored in the real-imaginary pairs data[5], data[6] up to data[nn-1], data[nn]. Negative frequencies of increasing magnitude are stored in data[2*nn-3], data[2*nn-2] down to data[nn+3], data[nn+4]. Finally, the pair data[nn+1], data[nn+2] contains the real and imaginary parts of the one aliased point which contains the most positive and the most negative frequency. You should try to develop a familiarity with this storage arrangement of complex spectra, also shown in the Figure 2, since it is the practical standard.

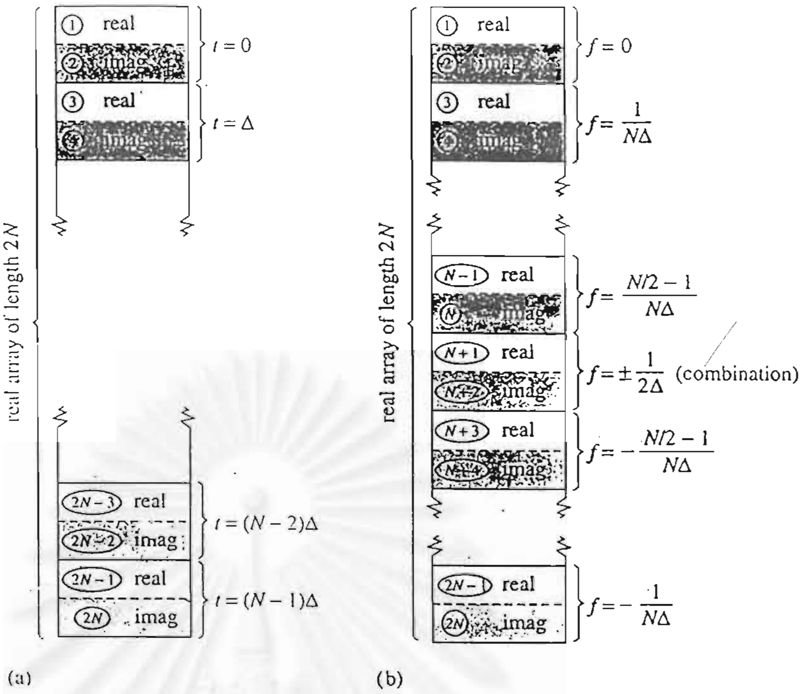

Figure B.2.2: Input and output arrays for FFT. a) The input array contains N (a power of 2) complex time samples in a real array of length $2N$, with real and imaginary parts alternating. b) The output array contains the complex Fourier spectrum at N values of frequency. Real and imaginary parts again alternate. The array starts with zero frequency, works up to the most positive frequency (which is ambiguous with the most negative frequency). Negative frequencies follow, from the second-most negative up to the frequency just below zero.

This is a good place to remind you that you can also use a routine like four1 without modification even if your input data array is zero-offset, that is has the range data[0..2*nn-1]. In this case, simply decrement the pointer to data by one when four1 is invoked, e.g. four1(data-1,1024,1). The real part of f_0 will now be returned in data[0], the imaginary part in data[1], and so on. We use the program below in our simulation program.

จุฬาลงกรณ์มหาวิทยาลัย

```
#include <math.h>
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

void fourn(float data[], unsigned long nn[], int ndim, int isign)
```

Replaces `data` by its `ndim`-dimensional discrete Fourier transform, if `isign` is input as 1. `nn[1..ndim]` is an integer array containing the lengths of each dimension (number of complex values), which MUST all be powers of 2. `data` is a real array of length twice the product of these lengths, in which the data are stored as in a multidimensional complex array: real and imaginary parts of each element are in consecutive locations, and the rightmost index of the array increases most rapidly as one proceeds along `data`. For a two-dimensional array, this is equivalent to storing the array by rows. If `isign` is input as −1, `data` is replaced by its inverse transform times the product of the lengths of all dimensions.

```
{
    int idim;
    unsigned long i1,i2,i3,i2rev,i3rev,ip1,ip2,ip3,ifp1,ifp2;
    unsigned long ibit,k1,k2,n,nprev,nrem,ntot;
    float tempi,tempr;
    double theta,wi,wpi,wpr,wr,wtemp;          Double precision for trigonometric recur-
                                                   rences.
    for (ntot=1,idim=1;idim<=ndim;idim++)      Compute total number of complex val-
        ntot *= nn[idim];                          ues.
    nprev=1;
    for (idim=ndim;idim>=1;idim--) {           Main loop over the dimensions.
        n=nn[idim];
        nrem=ntot/(n*nprev);
        ip1=nprev << 1;
        ip2=ip1*n;
        ip3=ip2*nrem;
        i2rev=1;
        for (i2=1;i2<=ip2;i2+=ip1) {           This is the bit-reversal section of the
            if (i2 < i2rev) {                      routine.
                for (i1=i2;i1<=i2+ip1-2;i1+=2) {
                    for (i3=i1;i3<=ip3;i3+=ip2) {
                        i3rev=i2rev+i3-i2;
                        SWAP(data[i3],data[i3rev]);
                        SWAP(data[i3+1],data[i3rev+1]);
                    }
                }
            }
            ibit=ip2 >> 1;
            while (ibit >= ip1 && i2rev > ibit) {
                i2rev -= ibit;
                ibit >>= 1;
            }
            i2rev += ibit;
        }
        ifp1=ip1;                               Here begins the Danielson-Lanczos sec-
        while (ifp1 < ip2) {                        tion of the routine.
            ifp2=ifp1 << 1;
            theta=isign*6.28318530717959/(ifp2/ip1);    Initialize for the trig. recur-
            wtemp=sin(0.5*theta);                           rence.
            wpr = -2.0*wtemp*wtemp;
            wpi=sin(theta);
            wr=1.0;
            wi=0.0;
            for (i3=1;i3<=ifp1;i3+=ip1) {
                for (i1=i3;i1<=i3+ip1-2;i1+=2) {
                    for (i2=i1;i2<=ip3;i2+=ifp2) {
                        k1=i2;                  Danielson-Lanczos formula:
                        k2=k1+ifp1;
                        tempr=(float)wr*data[k2]-(float)wi*data[k2+1];
                        tempi=(float)wr*data[k2+1]+(float)wi*data[k2];
                        data[k2]=data[k1]-tempr;
                        data[k2+1]=data[k1+1]-tempi;
                        data[k1] += tempr;
                        data[k1+1] += tempi;
                    }
                }
                wr=(wtemp=wr)*wpr-wi*wpi+wr;    Trigonometric recurrence.
                wi=wi*wpr+wtemp*wpi+wi;
            }
            ifp1=ifp2;
        }
        nprev *= n;
    }
}
```

B.3 FFT in Two Dimensions

Given a complex function $h(k_1, k_2)$ defined over the two-dimensional grid $0 \leq k_1 \leq N_1 - 1, 0 \leq k_2 \leq N_2 - 1$, we can define its two-dimensional discrete Fourier transform as a complex function $H(n_1, n_2)$ defined over the same grid,

$$H(n_1, n_2) \equiv \sum_{k_2=0}^{N_2-1} \sum_{k_1=0}^{N_1-1} \exp(2\pi i k_2 n_2/N_2) \exp(2\pi i k_1 n_1/N_1) h(k_1, k_2)$$

By pulling the "subscripts 2" exponential outside of the sum over k_1, or by reversing the order of summation and pulling the "subscript 1" outside of sum over k_2, we can see instantly that a two-dimensional FFT can be computed by taking one-dimensional FFTs sequentially on each index of the original function. Symbolically,

$$H(n_1, n_2) = \mathrm{FFT} - \mathrm{on} - \mathrm{index} - 1(\mathrm{FFT} - \mathrm{on} - \mathrm{index} - 2[h(k_1, k_2)])$$

$$= \mathrm{FFT} - \mathrm{on} - \mathrm{index} - 2(\mathrm{FFT} - \mathrm{on} - \mathrm{index} - 1[h(k_1, k_2)])$$

For this to be practical, of course, both N_1 and N_2 should be some efficient length for an FFT, usually a power of 2. Programming a two-dimensional FFT, using equation (B.2.4) with a one-dimensional FFT routine, is a bit clumsier than it seems at first, because the one-dimensional routine requires that its input be in consecutive order as a one-dimensional input array and then copying things out of the multidimensional technique. program given below.

Appendix C

Random Number Generator[1]

In this program, we want to have a portable random number generator which can be programmed in a high-level language, and which will generate the same random sequence on all machines. We can distinguish two classes of use for such portable routines: First, one might want a fully reliable generator. Second, one frequently one would like a quick generator to embed in a program.

For both purposes we do care that there be no sequential correlations, but we do not care that the discreteness of the random values returned be as fine as is allowed by all significant bits of the wordsize. In this case, one might desire the gain in speed that comes from using only one linear congruent generator. Then the following routine is perfectly adequate: edition, pages 211-212):

```
        #include <math.h>
        #define M 714025
        #define IA 1366
        #define IC 150889
        float ran2(idum)
        log *idum;
/* Returns a uniform random deviate between 0.0 and 1.0 Set idum to
any negative value value to initialize or reinitialize the sequence.
*/
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

---

[1] from Press et al. (1988)

```
{
        static long iy,ir[98];
        static int iff=0;
        int j;
        void nrerror( );
        if (*idum< 0|| iff== 0) { /* as above */
        iff=1;
        if ((*idum = (IC-(*idum) % M) < 0) *idum = -(*idum);
        for (j=i;j<=97; j++){ /* Initialize the shuffle table.  */
        *idum = (IA*(*idum)+IC) % M;
        ir[j]=(*idum);
        }
        *idum=(IA* (*idum)+IC) % M;
        iy=(*idum); /* Compare to ran(), above.  */
        }
        j=1 + 97.0*iy/M;
        if (j >97 || j< 1) nrerror (``RAN2:  This cannot happen.'');
        iy=ir[j];
        *idum=(IA*(*idum)+IC) % M;
        ir[j]=(*idum);
        return (float) iy/M;
}
```

The period of ran2 is again effectively infinite. Its principal limitation is that it returns one of only 714,025 possible values, equally spaced as a "comb" in the interval [0,1).

Appendix D

Computer Program

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

#define PI 3.141592654
#define iterations 1000
#define Rg 0.224    /* Rg is gyroradius of proton at 100 Mev Rg=0.224 A.U. */
#define delta 1.0   /* delta is delta of z axis                            */
#define lambdaC 16.0
#define Bo 1.0

/* define for slab calculation */
#define count1 3001
#define count 6001

/* define for fourn() function */
#define isign -1
#define SWAP(a,b) tempr=(a);(a)=(b);(b)=tempr

/* define for ran2() function */
#define M 714025
#define IA 1366
#define IC 150889

/* define for ran1() function */
#define Ia 16807
#define IM 2147483647
#define AM (1.0/IM)
#define IQ 127773
#define IR 2836
#define NTAB 32
#define NDIV (1+(IM-1)/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0-EPS)

/* slab simulations -> cal. only slab model                     */
/* 2d + slab simulations -> first cal. slab, afterthat cal 2d */

void main(void) {
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
int i,k,j,ndim,*NN,N2D,N2D1,N2D2,Nslab,Nslab1,Nslab2,P1=0;
int choice,type,Cross1,countCross1,zcutTimes,*ivector();

long idum,countalpha[count],countL[count],countCross[count];

float *data,**Axy,*Bx,*By,theta1,diffconts,zcutoff,dbsq,dbsqSlab,dbsq2D,Const;
float xcoor[count1],zcoor[count1],L[count1],alpha[count1];
float *dvector(),**dmatrix2();

void Cal2D(),fourn(),OutB2D(),OutAxy(),OutBxBy(),GetRealAxy(),OutBxBy(),Parseval2D();
void SlabCal(),Slab2DCross(),Setup(),CalBxBySlab(),output1(),output3(),LambdaC();
void Diffconts(),Zcutoff(),SlabCross(),AngleCoor(),Count(),Test2D(),Parseval();
void free_dvector(),free_dmatrix2(),OutBxnuba(),free_ivector();
void output31(),output32(),output33(),output34(),output35(),output36();
void output37(),output38(),output39();

theta1 = atan(110.0);
idum = 1;

/* type=1->run 2D+Slab, type=2->run Slab, type=3->test2D, type=4->testSlab */
/* choice (1-7) is choosen value of dbsq2D and dbsqSlab
type = 1;
choice = 6;

Setup(&ndim,&Nslab,&Nslab1,&Nslab2,&N2D,&N2D1,&N2D2,&dbsq,&dbsqSlab,&dbsq2D,
      &zcutTimes,choice,type,&Const);

NN = ivector(1,ndim);
if(Nslab1 >= N2D1) data = dvector(1,Nslab1);
else data = dvector(1,N2D1);
Bx = dvector(0,Nslab2);
By = dvector(0,Nslab2);
Axy = dmatrix2(0,2*N2D,0,2*N2D);

countCross1 = 0;
for(i=1;i<=count-1;i++)
    countL[i] = countalpha[i] = countCross[i] = 0;
for(k=1;k<=iterations;k++) {
    Cross1 = 0;
    for(i=0;i<=count1-1;i++)
        xcoor[i] = zcoor[i] = L[i] = alpha[i] = 0.0;
    if(type != 3){
        for(j=1;j<=2;j++) {
            SlabCal(Nslab,dbsqSlab,data,&idum,NN,&ndim);
            if(type == 4 && j == 1 && k == 1){
                Parseval(Nslab,Nslab1,P1,data);
                P1 = 1;
            }
            fourn(data,NN,ndim);
            CalBxBySlab(j,Nslab,Nslab1,data,Bx,By);
            if(type == 4 && j == 1 && k == 1)
                Parseval(Nslab,Nslab1,P1,data);
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
          }
          Diffconts(dbsq,&diffconts);
          Zcutoff(zcutTimes,theta1,diffconts,&zcutoff);
      }
      if(type == 3 && k == 1) {
          Cal2D(dbsq2D,N2D,data,&idum,NN,&ndim,Const);
          OutB2D(N2D,data);
          fourn(data,NN,ndim);
          OutB2D(N2D,data);
          GetRealAxy(N2D,NN,data,Axy,ndim);
          OutAxy(N2D,Axy);
          Test2D(k,Nslab2,N2D,Axy,dbsq2D);
          exit(0);
      }
      if(type == 4 && k == 1){
          LambdaC(Nslab,dbsqSlab,Bx);
          exit(0);
      }
      if(type == 1) {
          Cal2D(dbsq2D,N2D,data,&idum,NN,&ndim,Const);
          fourn(data,NN,ndim);
          GetRealAxy(N2D,NN,data,Axy,ndim);
          Slab2DCross(k,Nslab,Nslab2,N2D,&Cross1,&countCross1,Axy,Bx,By,
                      xcoor,zcoor,alpha,L,theta1,zcutoff);
      }
      if(type == 2){
          SlabCross(k,Nslab,Nslab2,&Cross1,Bx,By,xcoor,zcoor,
                    alpha,L,theta1,zcutoff,&countCross1);
      }
      if(type != 3){
          Count(L,alpha,Cross1,countalpha,countL,countCross);
          output1(k,xcoor,zcoor,alpha,L,Cross1);
          if(k == 1000)
              output3(k,countalpha,countL,countCross);
          }
   }
   free_dmatrix2(Axy,0,2*N2D,0);
   free_dvector(By,0);
   free_dvector(Bx,0);
   free_dvector(data,1);
   free_ivector(NN,1);
}

/* Setup(): Setup parameters of simulations */

/****************************************************************************/
  void Setup(ndim,Nslab,Nslab1,Nslab2,N2D,N2D1,N2D2,dbsq,dbsqSlab,dbsq2D,
             zcutTimes,choice,type,Const)
/****************************************************************************/

int *ndim,*Nslab,*Nslab1,*Nslab2,*N2D,*N2D1,*N2D2,*zcutTimes,choice,type;
float *dbsq,*dbsqSlab,*dbsq2D,*Const;
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
{
   FILE *out1,*out3;

   out1  = fopen("Out1.dat","w");
   out3  = fopen("Out3.dat","w");

   /* Computational Trick !!!! */
   /* *N2D = 64     *Const = 2*1.5*PI*PI*PI  ERROR = -0.23%   */
   /* *N2D = 128    *Const = 2*1.8*PI        ERROR = -1.09%   */
   /* *N2D = 256    *Const = 4*1.5*PI*PI     ERROR = 0.74%    */
   /* *N2D = 512    *Const = 2*1.5*PI*PI     ERROR = -0.14%   */
   /* *N2D = 1024   *Const = 4*1.35*PI       ERROR = 0.36%    */

    /* type = 1 for 2D+Slab model, use 80% of 2d + 20% of slab */
    if(type == 1 || type == 3) {
       *ndim = 2;
        if(choice == 1){   /* dbsq = 1e-5 */
            *N2D = 64;
            *Const = 2*1.5*PI*PI*PI;
            *Nslab = (4096)*64;
            *dbsq = 1e-5;
            *dbsqSlab = 2e-6;
            *dbsq2D = 8e-6;
            *zcutTimes = 30*64;
        }
        if(choice == 2){   /* dbsq = 1e-4 */
            *N2D = 128;
            *Const = 2*1.8*PI;
            *Nslab = (8192)*64;
            *dbsq = 1e-4;
            *dbsqSlab = 2e-5;
            *dbsq2D = 8e-5;
            *zcutTimes = 10*64;
        }
        if(choice == 3){   /* dbsq = 1e-3 */
            *N2D = 256;
            *Const = 4*1.5*PI*PI;
            *Nslab = (8192)*64;
            *dbsq = 1e-3;
            *dbsqSlab = 2e-4;
            *dbsq2D = 8e-4;
            *zcutTimes = 64;
        }
        if(choice == 4){   /* dbsq = 1e-2 */
            *N2D = 512;
            *Const = 2*1.5*PI*PI;
            *Nslab = 131072*8;
            *dbsq = 1e-2;
            *dbsqSlab = 2e-3;
            *dbsq2D = 8e-3;
            *zcutTimes = 8;
        }
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
    if(choice == 5){   /* dbsq = 2e-2 */
        *N2D = 512;
        *Const = 2*1.5*PI*PI;
        *Nslab = 262144*16;
        *dbsq = 2e-2;
        *dbsqSlab = 4e-3;
        *dbsq2D = 1.6e-2;
        *zcutTimes = 16;
    }
    if(choice == 6){   /* dbsq = 5e-2 */
        *N2D = 512;
        *Const = 2*1.5*PI*PI;
        *Nslab = 524288*8;
        *dbsq = 5e-2;
        *dbsqSlab = 1e-2;
        *dbsq2D = 4e-2;
        *zcutTimes = 8;
    }
    if(choice == 7){   /* dbsq = 1e-1 */
        *N2D = 1024;
        *Const = 2*1.35*PI;
        *Nslab = 1048576*8;
        *dbsq = 1e-1;
        *dbsqSlab = 2e-2;
        *dbsq2D = 8e-2;
        *zcutTimes = 8;
    }
}
/* type = 2 for Slab model */
if(type == 2 || type == 4) {
    *N2D = 0;
    *ndim = 1;
    if(choice == 1){   /* dbsq = 1e-5 */
        *Nslab = 4096;
        *dbsqSlab = 1e-5;
        *zcutTimes = 30;
    }
    if(choice == 2){   /* dbsq = 1e-4 */
        *Nslab = 8192;
        *dbsqSlab = 1e-4;
        *zcutTimes = 10;
    }
    if(choice == 3){   /* dbsq = 1e-3 */
        *Nslab = 8192*8;
        *dbsqSlab = 1e-3;
        *zcutTimes = 8;
    }
    if(choice == 4){   /* dbsq = 1e-2 */
        *Nslab = 131072*8;
        *dbsqSlab = 1e-2;
        *zcutTimes = 8;
    }
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
        if(choice == 5){   /* dbsq = 2e-2 */
            *Nslab = 262144*8;
            *dbsqSlab = 2e-2;
            *zcutTimes = 8;
        }
        if(choice == 6){   /* dbsq = 5e-2 */
            *Nslab = 524288*8;
            *dbsqSlab = 5e-2;
            *zcutTimes = 8;
        }
        if(choice == 7){   /* dbsq = 1e-1 */
            *Nslab = 1048576*2;
            *dbsqSlab = 1e-1;
            *zcutTimes = 2;
        }
        *dbsq = *dbsqSlab;
    }
    if(choice > 7 || choice < 1){
    printf("ERROR, select choice 1-7 only\n");
    exit(0);
    }
    *Nslab1 = 2*(*Nslab)+1;
    *Nslab2 = *Nslab+1;

    *N2D1 = 8*(*N2D)*(*N2D)+1;
    *N2D2 = 4*(*N2D)*(*N2D)+1;

    fprintf(out1,"dbsq = %lf dbsqSlab = %lf dbsq2D =% lf\n",*dbsq,*dbsqSlab,*dbsq2D);
    fprintf(out1,"Nslab = %d N2D = %d\n",*Nslab,*N2D);
    fprintf(out1,"\n");
    fprintf(out3,"dbsq = %lf dbsqSlab = %lf dbsq2D =% lf\n",*dbsq,*dbsqSlab,*dbsq2D);
    fprintf(out3,"Nslab = %d N2D = %d\n",*Nslab,*N2D);
    fprintf(out3,"\n");

    fclose(out1);
    fclose(out3);
}

/* Subroutine SlabCal is used for calculate Pk from a slab  model in Bieber 1990. */
/* P(k) = A(kp,kz) where A(kp,kz) = c/(1+kz^2lambda^2)^meu */
/* P(k) ==> data ==> sqrt(data)     */

/**************************************************/
   void SlabCal(Nslab,dbsqSlab,data,idum,NN,ndim)
/**************************************************/
int Nslab,NN[],*ndim;
long *idum;
float dbsqSlab, data[];

{
    int i;
    float  Meu, A, Kz, C, lambda, Ph, RePh, ImPh;
    float ran2();

    NN[1] = Nslab;
    *ndim = 1;

    Meu = 5.0/6.0;
    C = lambdaC*dbsqSlab/sqrt(2*PI);
    lambda = 4*2.104*lambdaC/(2*PI);

    Ph = RePh = ImPh = 0.0;

    /* special case for nn = 0 */
    data[1] = 0.0;
    data[2] = 0.0;

    /* special case for nn = Nslab/2 */
    Kz = PI*lambda/(delta);
    A = sqrt(2*PI)*Nslab*delta*C/(pow((1+Kz*Kz),Meu));

    /* sqrt(data) ==> FFT ==> Kzx */
    data[Nslab+1] = sqrt(A);
    data[Nslab+2] = 0.0;

    /* general case of nn */
    for(i=1;i<=Nslab/2-1;i++) {
        Kz = 2*PI*lambda*i/(Nslab*delta);
        A = sqrt(2*PI)*Nslab*delta*C/(pow((1+Kz*Kz),Meu));

        /* random phase ==> random magnetic field */
        Ph = 2.0*PI*ran2(idum);
        RePh = cos(Ph);
        ImPh = sin(Ph);

        data[2*i+1] = sqrt(A)*RePh; /* real part */
        data[2*(Nslab-i)+1] = sqrt(A)*RePh;

        data[2*i+2] = sqrt(A)*ImPh; /* Im part   */
        data[2*(Nslab-i)+2] = -sqrt(A)*ImPh;
    }
}

/* Subroutine Cal2D is used for calculate Pk from a 2D      */
/* P(k) = A(kp,kz) where A(kp,kz) = c/(1+kz^2lambda^2)^meu */

/*************************************************/
  void Cal2D(dbsq2D,N2D,data,idum,NN,ndim,Const)
/*************************************************/
float data[],dbsq2D,Const;
int N2D,*ndim;
long *idum;
unsigned long NN[];
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
{
    int i,j,g;
    float  Kx,Ky,Kperpen,Meu,A,C,lambda,Ph,RePh,ImPh;
    float ran2();

    *ndim = 2;

    Meu = 5.0/6.0;
    C = lambdaC*dbsq2D/(PI);
    lambda = 2*2.104*lambdaC/(PI);

    Ph = RePh = ImPh = 0.0;

    for(i=1;i<=*ndim;i++)
        NN[i] = 2*N2D;

    /* set all data[] = 0.0 */
    for(j=0;j<=2*N2D-1;j++){
        for(i=1;i<=4*N2D-1;i+=2){
            data[4*N2D*j+i] = 0.0;
            data[4*N2D*j+i+1] = 0.0;
        }
    }

    /* calculate q1 and q4 */
    for(j=0;j<=2*N2D-1;j++){
        for(i=0;i<=4*N2D-1;i+=2){
            Kx = 2*PI*lambda*(i/2.0)/(2*N2D*delta);
            Ky = 2*PI*lambda*j/(2*N2D*delta);
            Kperpen = sqrt(Kx*Kx+Ky*Ky);

            /* Const is come from Setup(), computational trick ! */
            A = lambda*lambda*lambda*Const*2*N2D*C/(Kperpen*Kperpen*Kperpen*
                pow((1+Kperpen*Kperpen),Meu));

            Ph = 2.0*PI*ran2(idum);
            RePh = cos(Ph);
            ImPh = sin(Ph);

            if(i <= 2*N2D && i >= 0){
                data[4*N2D*j+i+1] = RePh*sqrt(A);
                data[4*N2D*j+i+2] = ImPh*sqrt(A);
            }
            if((i == 2*N2D || i == 0) && j > N2D){
                 data[4*N2D*j+i+1] = data[4*N2D*(2*N2D-j)+i+1];
                 data[4*N2D*j+i+2] = -data[4*N2D*(2*N2D-j)+i+2];
            }

          /* calculate special value for H(-f) = conjugate of H(f) */
            data[4*N2D*N2D+2] = data[4*N2D*N2D+2*N2D+2] = data[2*N2D+2] = 0.0;
            data[1] = data[2] = 0.0;
        }
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
    }

    /* calculate q2 and q3 */
    for(j=0;j<=2*N2D-1;j++){
        for(i=0;i<=4*N2D-1;i+=2){
        g = 8*N2D*N2D+4*N2D+2;
            if(i > 2*N2D+1 && j > 0){
                data[4*N2D*j+i+1] = data[g-(4*N2D*j+i+1)];
                data[4*N2D*j+i+2] = -data[g-(4*N2D*j+i+1)+1];
            }
            if(i > 2*N2D && j == 0){
                data[i+1] = data[4*N2D+2-(i+1)];
                data[i+2] = -data[4*N2D-(i+1)+3];
            }
            /* set data which is shown below = 0.0 */
            data[4*N2D*N2D+2] = data[4*N2D*N2D+2*N2D+2] = data[2*N2D+2] = 0.0;
            data[1] = data[2] = 0.0;
        }
    }
}

/* GetRealAxy(): data[Real+Im] -> Axy{Real] */

/*****************************************/
   void GetRealAxy(N2D,NN,data,Axy,ndim)
/*****************************************/
float data[],**Axy;
int N2D,NN[],ndim;

{
    int i,j,p,idim,ntot;

    for(ntot=1,idim=1;idim<=ndim;idim++)
        ntot *= NN[idim];

    /* Calculate Axy in q1 */
    for(i=N2D-1;i<=2*N2D-1;i++) {
        p = 1;                           /* biginning point in q1 */
        for(j=N2D-1;j<=2*N2D-1;j++) {
            Axy[j][i] = data[p+(i-N2D+1)*4*N2D]/(float)ntot;
            p += 2;
        }
    }

    /* Calculate Axy in q2 */
    for(i=N2D-1;i<=2*N2D-1;i++) {
        p = 2*N2D+3;                     /* beginning point in q2 */
        for(j=0;j<=N2D-2;j++) {
            Axy[j][i] = data[p+(i-N2D+1)*4*N2D]/(float)ntot;
            p += 2;
        }
    }
```

จุฬาลงกรณ์มหาวิทยาลัย

```
    /* Calculate Axy in q3 */
    for(i=0;i<=N2D-2;i++) {
        p = 4*N2D*(N2D+2)-2*(N2D-1)+1;   /* beginning point in q3 */
        for(j=0;j<=N2D-2;j++) {
            Axy[j][i] = data[p+i*4*N2D]/(float)ntot;
            p += 2;
        }
    }

    /* Calculate Axy in q4 */
    for(i=0;i<=N2D-2;i++) {
        p = 4*N2D*(N2D+2)-4*N2D+1;       /* beginning point in q4 */
        for(j=N2D-1;j<=2*N2D-1;j++) {
            Axy[j][i] = data[p+i*2*N2D]/(float)ntot;
            p += 2;
        }
    }
}

/* SlabCross(): trace random trajectories, from slab model and cross a shock */

/*******************************************************************/
  void SlabCross(k,Nslab,Nslab2,Cross1,Bx,By,xcoor,zcoor,alpha,L,
                theta1,zcutoff,countCross1)
/*******************************************************************/

int k,Nslab,Nslab2,*Cross1,*countCross1;
float xcoor[],zcoor[],alpha[],L[],Bx[],By[],theta1,zcutoff;
{
    void AngleCoor(),CrossCheck();

    int i,a,b,g,h,s,Cross;
    float shock,xcutoff,m,tjX=0,tjY=0,tjXold=0,tjYold=0;

    i = a = b = g = h = s = 0;    /* Upstream a = 0, Downstream a = 1 */
    do {
         i += 1;
         xcutoff = zcutoff/tan(theta1);
         m = delta/tan(theta1);
         shock = -m*i+xcutoff;

         /* Euler's method for calculating tjtories of magnetic field */
         tjX = tjXold+delta*Bx[i]/Bo;
         tjY = tjYold+delta*By[i]/Bo;

         /* Check crossings at z<0 */
         if(k <= 100 && i >= 1){
             CrossCheck(i,&h,Nslab,Nslab2,k,tjX,&countCross1,theta1,zcutoff);
             if(h == 1)
                 break;
         }
```

```
            /* Check escaping from a shock */
            if (tjX > shock+xcutoff && s == 0) {
                printf("STOP when tjX > shock+xcutoff at i=%d =>
                Particle Escape from Shock\n",i);
                printf("tjX[%d] = %lf, shock+xcutoff = %lf\n",i,tjX,shock+xcutoff);
                s = 1;
            }

            /* Crossings a shock, Up -> Down */
            if (tjX >= shock && a == 0 && s == 0) {
                /* Check error of crossings */
                if(i >= Nslab/4) {
                    printf("ERROR, crossing at z >= N/4 Up -> Down at i = %d\n",i);
                    break;
                }
                *Cross1 += 1;
                Cross = *Cross1;
                AngleCoor(i,theta1,zcutoff,tjX,tjY,tjXold,tjYold,
                zcoor,xcoor,alpha,L,Cross);
                a = 1;
            }

            /* Crossings a shock, Down -> Up */
            if (tjX < shock && a == 1 && s == 0) {
                 /* Check error of crossings */
                if(i >= Nslab/4) {
                    printf("ERROR, crossing at z >= N/4 Down -> Up at i = %d\n",i);
                    break;
                }
                *Cross1 += 1;
                Cross = *Cross1;
                AngleCoor(i,theta1,zcutoff,tjX,tjY,tjXold,tjYold,
                zcoor,xcoor,alpha,L,Cross);
                a = 0;
            }
            tjXold = tjX;
            tjYold = tjY;
        } while (i <= Nslab2);
}

/* Slab2DCross(): trace random trajectories (2D + Slab model), */
/* bilinear interpolation and cross a shock                     */

/********************************************************************/
  void Slab2DCross(k,Nslab,Nslab2,N2D,Cross1,countCross1,Axy,Bx,By,
                   xcoor,zcoor,alpha,L,theta1,zcutoff)
/********************************************************************/

int k,Nslab,Nslab2,N2D,*Cross1,*countCross1;
float **Axy,xcoor[],zcoor[],alpha[],L[],Bx[],By[],theta1,zcutoff;
{
```

```
void AngleCoor(),CrossCheck();

int i,a,b,g,h,distanceX,distanceY,Cross,stop;
float A00,A01,A10,A11,fx,fy,Xold,Yold,X,Y;
float deltaX,deltaY,Bxtemp,Bytemp,shock,xcutoff,m,tjX,tjXold,tjY,tjYold;

FILE *out1;
out1  = fopen("Out1.dat","a");

i = a = b = g = h = stop = 0;   /* Upstream a = 0, Downstream a = 1 */
do {
/* set X = Y = 1e-4 (for the 1st time only) for calculate Bx(0,0) and By(0,0) */
     if(g == 0){
         i = 0;
         distanceX = distanceY = (N2D-1)*delta;
         deltaX = deltaY = 1e-6;
         Xold = Yold = distanceX;
         X = Y = distanceX+deltaX;
         g = 1;
     }
     A00 = Axy[distanceX][distanceY];
     A10 = Axy[distanceX+1][distanceY];
     A01 = Axy[distanceX][distanceY+1];
     A11 = Axy[distanceX+1][distanceY+1];

     fx = deltaX/delta;
     fy = deltaY/delta;

     /* bilinear interpolation of Axy */
     Bxtemp = (-(1-fx)*A00+(1-fx)*A01-fx*A10+fx*A11)/delta+Bx[i];
     Bytemp = ((1-fy)*A00+fy*A01-(1-fy)*A10-fy*A11)/delta+By[i];

     /* Euler's method (2) */
     X = Xold+Bxtemp*delta/Bo;
     Y = Yold+Bytemp*delta/Bo;

     /* tjX (tjY) is random trajectories in xz(yz)-plane */
     /* (N2D-1)*delta is set for a center of Axy          */
     tjX = X-(N2D-1)*delta;
     tjY = Y-(N2D-1)*delta;

     distanceX = (int)X;
     distanceY = (int)Y;
     deltaX = (X-distanceX)/delta;
     deltaY = (Y-distanceY)/delta;

     if((distanceX == 2*N2D-1)||(distanceY == 2*N2D-1)||
     distanceX < 0||distanceY < 0){
         fprintf(out1,"ERROR at iterations=%d, Out of Axy at distanceX=%d
         distanceY=%d => Stop this iteration\n",k,distanceX,distanceY);
         printf("ERROR at iterations=%d, Out of Axy at distanceX=%d distanceY=%d =>
         Stop this iteration\n",k,distanceX,distanceY);
     stop = 1;
 }

/* calculate shock */
 xcutoff = zcutoff/tan(theta1);
 m = delta/tan(theta1);
 shock = -m*i+xcutoff;

 /* Check crossings at z<0 (5%) */
 if(k <= 20 && i >= 1){
     CrossCheck(i,&h,Nslab,Nslab2,k,tjX,&*countCross1,theta1,zcutoff);
     if(h >= 1)
         stop = 1;
 }

 /* Crossing a shock from Up -> Down */
 if (tjX > shock+xcutoff && i >= 1) {
     printf("STOP when X > shock+xcutoff at iterations=%d =>
     Particle Escape from Shock\n",k);
     stop = 1;
 }
 if (tjX >= shock && a == 0 && i >= 1) {
     if(i >= Nslab/4) {
         printf("ERROR, crossing at z >= N/4 Up -> Down at i = %d\n",i);
         stop = 1;
     }
     *Cross1 += 1;
     Cross = *Cross1;
     AngleCoor(i,theta1,zcutoff,tjX,tjY,tjXold,tjYold,zcoor,xcoor,alpha,L,Cross)
     a = 1;
 }

 /* Crossing a shock from Up -> Down */
 if (tjX < shock && a == 1 && i >= 1) {
     if(i >= Nslab/4) {
     printf("ERROR, crossing at z >= N/4 Down -> Up at i = %d\n",i);
     stop = 1;
     }
     *Cross1 += 1;
     Cross = *Cross1;
     AngleCoor(i,theta1,zcutoff,tjX,tjY,tjXold,tjYold,zcoor,xcoor,alpha,L,Cross)
     a = 0;
 }

 /* Euler's method (2) */
 Xold = X;
 Yold = Y;
 tjX = X-(N2D-1)*delta;
 tjY = Y-(N2D-1)*delta;

 i += 1;
 if(stop >= 1)
```

จุฬาลงกรณ์มหาวิทยาลัย

```
              i = Nslab2+100;
    }
    while (i <= Nslab2);
    fclose(out1);
}

/* FAST FOURIER TRAN2DSFORM from "N2DUMERICAL RECIPES IN2D C" 2nd edition */
/* by Press, FlaN2Derry, Teukolsky, and Vetterling pages 523-524.          */
/* We define N2D and isign at the header.                                  */

/*******************************/
   void fourn(data,NN,ndim)
/******************************/
float data[];
unsigned long NN[];
int ndim;

{
   int idim;
   unsigned long i1,i2,i3,i2rev,i3rev,ip1,ip2,ip3,ifp1,ifp2;
   unsigned long ibit,k1,k2,n,nprev,nrem,ntot;
   float tempi,tempr;
   float theta,wi,wpi,wpr,wr,wtemp;

   for(ntot=1,idim=1;idim<=ndim;idim++)
       ntot *= NN[idim];
   nprev=1;
   for(idim=ndim;idim>=1;idim--){
       n=NN[idim];
       nrem=ntot/(n*nprev);
       ip1=nprev << 1;
       ip2=ip1*n;
       ip3=ip2*nrem;
       i2rev=1;
       for(i2=1;i2<=ip2;i2+=ip1){
           if(i2 < i2rev){
               for(i1=i2;i1<=i2+ip1-2;i1+=2){
                   for(i3=i1;i3<=ip3;i3+=ip2){
                       i3rev=i2rev+i3-i2;
                       SWAP(data[i3],data[i3rev]);
                       SWAP(data[i3+1],data[i3rev+1]);
                   }
               }
           }
           ibit=ip2 >> 1;
           while(ibit >= ip1 && i2rev > ibit){
               i2rev -= ibit;
               ibit >>= 1;
            }
            i2rev += ibit;
       }
       ifp1=ip1;
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
        while(ifp1 < ip2){
            ifp2=ifp1 << 1;
            theta=isign*6.28318530717959/(ifp2/ip1);
            wtemp=sin(0.5*theta);
            wpr = -2.0*wtemp*wtemp;
            wpi=sin(theta);
            wr=1.0;
            wi=0.0;
            for(i3=1;i3<=ifp1;i3+=ip1){
                for(i1=i3;i1<=i3+ip1-2;i1+=2){
                    for(i2=i1;i2<=ip3;i2+=ifp2){
                        k1=i2;
                        k2=k1+ifp1;
                        tempr=(float)wr*data[k2]-(float)wi*data[k2+1];
                        tempi=(float)wr*data[k2+1]+(float)wi*data[k2];
                        data[k2]=data[k1]-tempr;
                        data[k2+1]=data[k1+1]-tempi;
                        data[k1]+=tempr;
                        data[k1+1]+=tempi;
                    }
                }
                wr=(wtemp=wr)*wpr-wi*wpi+wr;
                wi=wi*wpr+wtemp*wpi+wi;
            }
            ifp1=ifp2;
        }
        nprev *= n;
    }
}

/* RANDOM NUMBER from "NUMERICAL RECIPES IN C" page 212 */
/* random number between 0.0 <= ran2 < 1.0             */
/*********************/
   float ran2(idum)
/*********************/

long *idum;

{
static long iy, ir[98];
static int iff=0;
int j;

if(*idum < 0 || iff == 0){
iff=1;
if((*idum=(IC-(*idum)) %M) <0)*idum = -(*idum);
for(j=1;j<=97;j++) {
*idum=(IA*(*idum)+IC) %M;
ir[j]=(*idum);
}
*idum=(IA*(*idum)+IC) %M;
iy=(*idum);
```

จุฬาลงกรณ์มหาวิทยาลัย

```
}
j=1+97.0*iy/M;
if(j>97 || j<1) {
printf("ERROR in RAN2\n");
printf("Press anykey to continue\n");
putchar('\n');
}
iy=ir[j];
*idum=(IA*(*idum)+IC) %M;
ir[j]=(*idum);
return (float) iy/M;
}

/* CalBxBySlab():  data[Real + Im] -> Bx[Real] */

/*******************************************************/
  void CalBxBySlab(j,Nslab,Nslab1,data,Bx,By)
/*******************************************************/

int j,Nslab,Nslab1;
float data[],Bx[],By[];
{
    int i;

    Bx[0] = By[0] = 0.0;
    if(j == 1) {
        for(i=1;i<=(Nslab1-1);i+=2)
            Bx[(i+1)/2] = data[i]/Nslab;
    }
    if(j == 2) {
        for(i=1;i<=(Nslab1-1);i+=2)
            By[(i+1)/2] = data[i]/Nslab;
    }
}

/* calculate a diffusion coefficient */

/*************************************/
   void Diffconts(dbsq,diffconts)
/*************************************/

float dbsq,*diffconts;
{
    *diffconts = dbsq*lambdaC/(2.0*Bo*Bo);
}

/* Calculate the cut-off distance in Z axis */

/*******************************************************/
   void Zcutoff(zcutTimes,theta1,diffconts,zcutoff)
/*******************************************************/
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
int zcutTimes;
float theta1,diffconts,*zcutoff;
{
    /* we use zcutTimes times of zcutoff, 6.25 = 5*5/2 */
    *zcutoff = zcutTimes*6.25*diffconts*tan(theta1)*tan(theta1);
}

/* CrossCheck(): Check crossings at z<0 */

/*******************************************************************/
  void CrossCheck(i,h,Nslab,Nslab2,k,tjX,countCross1,theta1,zcutoff)
/*******************************************************************/

int i,*h,Nslab,Nslab2,*countCross1,k;
float theta1,zcutoff,tjX;
{
    float m,c1,shock1;

    FILE *out1;
    out1  = fopen("Out1.dat","a");

    /* Calculate shock */
    m = delta/tan(theta1);
    c1 = (Nslab2*delta+zcutoff)/tan(theta1);
    shock1 = -m*i+c1;

    /* Check crossings at z <0 */
    if(tjX >= shock1) {
        printf("ERROR, There are crossings at z < 0 in itteration no. %d\n",k);
        fprintf(out1,"ERROR, There are crossings at z < 0 in itteration no. %d\n",k);
        *countCross1 += 1;
        *h = 1;
    }

    /* If crossings at z<0 higher than 1%, ERROR */
    if(*countCross1 > 1) {
        printf("ERROR, Crossings at z < 0 are higher than 5%% in
        itteration no = %d\n",k);
        fprintf(out1,"ERROR, Crossings at z < 0 are higher than 5%% in
        itteration no. = %d\n",k);
        *h = 10000;
    }
    if(*countCross1 <= 2 && i == Nslab2){
        fprintf(out1,"iterations %d is OK, crossings at z < 0 is less than 5%% of 100
        itterations\n",k);
    }
    fclose(out1);
}

/* AngleCoor(): Calculte Cross, Angle, and L at every iteration */

/************************************************************************************/
```

```
  void AngleCoor(i,theta1,zcutoff,tjX,tjY,tjXold,tjYold,zcoor,xcoor,alpha,L,Cross)
/**************************************************************************************/

int i,Cross;
float theta1,zcutoff,tjX,tjY,tjXold,tjYold,zcoor[],xcoor[],alpha[],L[];
{
    float c,d,e,x1=0,z1=0,xdiff=0,zdiff=0;

    /* Calculate Crossing-Position */
    c = tjX-(tjX-tjXold)*i;
    zcoor[2*Cross-1] = (float)(i);
    zcoor[2*Cross] = (zcutoff/tan(theta1)-c)/(1/tan(thetai)+(tjX-tjXold)/delta);
    xcoor[2*Cross-1] = (float)(i);
    xcoor[2*Cross] = -zcoor[2*Cross]/tan(theta1)+zcutoff/tan(theta1);

    /* Calculate distance between two crossing points */
    if(Cross >= 2) {
        xdiff = xcoor[2*Cross]-x1;
        zdiff = zcoor[2*Cross]-z1;
        L[Cross-1] = sqrt(xdiff*xdiff+zdiff*zdiff);
printf("!!!!!!!!!!!\n");
printf("L[%d]=%lf\n",Cross-1,L[Cross-1]);
    }
    x1 = xcoor[2*Cross];
    z1 = zcoor[2*Cross];

    /* Calculate Crossing angle */
    d = sin(theta1)*(tjX-tjXold)+cos(theta1)*delta;
    e = (tjX-tjXold)*(tjX-tjXold)+(tjY-tjYold)*(tjY-tjYold)+delta*delta;
    alpha[2*Cross-1] = (float)(i);
    alpha[2*Cross] = acos(d/sqrt(e))*180.0/PI;

}

/* Count() is used for collect statistical data for 1000 iterations */

/**********************************************************/
  void Count(L,alpha,Cross1,countalpha,countL,countCross)
/*********************************************************/

int Cross1;
float L[],alpha[];
long countalpha[],countL[],countCross[];
{
    int i,j;

    for(i=0;i<=200;i++) {     /* We estimate that No. of L[] is less than 200 */
        for(j=1;j<=count;j++) {
            if(((int)(L[i]+0.5) <= j*500) && ((int)(L[i]+0.5) > j*500-500)) {
                countL[j] += 1;
                break;
            }
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
        }
    }
    for(i=0;i<=200;i+=2) {    /* We estimate alpha[] and Cross1 > 200 too*/
        for(j=1;j<=180;j++) {
            if((int)(alpha[i]+0.5) == j) {
                countalpha[j] += 1;
                break;
            }
        }
   }
   for(j=1;j<=200;j++) {
       if(Cross1 == j) {
            countCross[j] += 1;
            break;
       }
   }
}

/* output program */
/* output1(): All data (Angle, L, Cross)  at every iteration */

/********************************************/
   void output1(k,xcoor,zcoor,alpha,L,Cross1)
/********************************************/

int Cross1,k;
float xcoor[],zcoor[],alpha[],L[];
{
    int i;

    FILE *out1;
    out1  = fopen("Out1.dat","a");

    fprintf(out1,"iterations = %d     ",k);
    fprintf(out1,"Cross = %d\n",Cross1);

    fprintf(out1,"  no.         L[]         i        alpha[]      xcoor[]    zcoor[]\n");
    for(i=1;i<=(count1-1)/2;i++) {
        if(alpha[2*i] >= 1e-7 || alpha[2*i] <= -1e-7) {
             fprintf(out1,"%4d   %9.3lf   %9.3lf   %9.3lf   %9.3lf   %9.3lf\n",i
                     ,L[i],alpha[2*i-1],alpha[2*i],xcoor[2*i],zcoor[2*i]);
        }
    }
    fprintf(out1,"             ************************************\n");
    fclose(out1);
}

/* output3(): Statistical data for plotting */

/***********************************************/
   void output3(k,countalpha,countL,countCross)
/***********************************************/
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
int k;
long countalpha[],countL[],countCross[];
{
    int i;

     FILE *out3;
     out3  = fopen("Out3.dat","a");

     fprintf(out3,"iterations = %4d\n",k);
     fprintf(out3,"   i     countalpha  countL(500) countCross \n");
     for(i=0;i<=(count-1);i++)
         fprintf(out3,"%5d       %3ld        %3ld        %3ld\n",i,countalpha[i],
                 countL[i],countCross[i]);
     fprintf(out3,"  *****************************\n");
     fclose(out3);
}

/* OutB2D(): Checking for 2D IFT by ourself */

/******************************/
  void OutB2D(N2D,data)
/*****************************/

float data[];
int N2D;

{
    int i,j;

    FILE *outB2D,*fopen();
    outB2D = fopen("OutB2D.dat","a");

    for(j=N2D;j>=0;j--){
        fprintf(outB2D,"\n");
        for(i=3;i<=2*N2D-1;i+=2)
          fprintf(outB2D,"%9.2e %9.2e ",data[2*N2D+4*N2D*j+i],data[2*N2D+4*N2D*j+i+1]);
    for(i=1;i<=2*N2D+1;i+=2)
         fprintf(outB2D,"%9.2e %9.2e ",data[4*N2D*j+i],data[4*N2D*j+i+1]);
    }
    for(j=2*N2D-1;j>=N2D+1;j--){
         fprintf(outB2D,"\n");
         for(i=3;i<=2*N2D-1;i+=2)
           fprintf(outB2D,"%9.2e %9.2e ",data[2*N2D+4*N2D*j+i],data[2*N2D+4*N2D*j+i+1]);
         for(i=1;i<=2*N2D+1;i+=2)
           fprintf(outB2D,"%9.2e %9.2e ",data[4*N2D*j+i],data[4*N2D*j+i+1]);
    }
    fprintf(outB2D,"\n");
    fclose(outB2D);
}

/* OutAxy(): Checking Axy */
```

```
/***************************/
  void OutAxy(N2D,Axy)
/**************************/

float **Axy;
int N2D;

{
    int i,j;

    FILE *outB2D;
    outB2D = fopen("OutB2D.dat","a");

    for(i=2*N2D-1;i>=0;i--) {
        fprintf(outB2D,"\n");
        for(j=0;j<=2*N2D-1;j++){
            fprintf(outB2D,"%9.2e ",Axy[j][i]);
        }
    }
    fprintf(outB2D,"\n");
    fclose(outB2D);
}

/* Test2D() is used for checking Cal2D() (calculate 2D turbulence) */

/************************************************/
  void Test2D(k,Nslab2,N2D,Axy,dbsq2D)
/***********************************************/

int k,Nslab2,N2D;
float **Axy,dbsq2D;
{
   int i,j;
   float A00,A01,A10,A11,fx,fy,Bx,By,Bxysqav=0.0,error;

   for(i=0;i<=2*N2D-2;i++) {
       for(j=0;j<=2*N2D-2;j++) {
           A00 = Axy[j][i];
           A10 = Axy[j+1][i];
           A01 = Axy[j][i+1];
           A11 = Axy[j+1][i+1];

           fx = 0.5/delta;
           fy = 0.5/delta;

           Bx = (-(1-fx)*A00+(1-fx)*A01-fx*A10+fx*A11)/delta;
           By = ((1-fy)*A00+fy*A01-(1-fy)*A10-fy*A11)/delta;
           Bxysqav += (Bx*Bx+By*By)/(float)(4*(N2D-1)*(N2D-1));
       }
   }
   error = (Bxysqav-dbsq2D)*100/dbsq2D;
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
   printf("Bxysqav = %.9lf dbsq2D = %lf ERROR = %lf%%\n",Bxysqav,dbsq2D,error);
   printf("N2D = %d  Nslab2 = %d\n",N2D,Nslab2);
}

/* Subroutine Perseval()is used for check the four1()  */
/* summation(|data^2|)/Nslab = summation(|Bx^2|)       */

/**********************************************/
  void Parseval(Nslab,Nslab1,P1,data)
/*********************************************/

int Nslab,Nslab1,P1;
float data[];
{
    int i;
    float sumdata=0;

    for(i=1;i<=Nslab1;i++) {
        sumdata+=data[i]*data[i];
    }
    if(P1 == 0){
        printf("Parseval checking\n");
        printf("sum(Bx)=%9.6lf\n",sumdata);
    }
    if(P1 == 1)
        printf("sum(data/Nslab)=%9.6lf\n",sumdata/Nslab);
}

/* subroutine Corelation length (lambdaC) is used for check  */
/* a SlabCal(). We use lambdaC = LC (LambdaC).               */
/* LC = (delta/2){sum j [sum i (Bx(i)*Bx(i+j)/Bx^2(i))]}     */
/* deltaBSQ = (sum i 2*(sum i Bx^2(i)/Nslab))                */
/* where (sum i Bx^2(i)/Nslab) = (sum i Bx^2(i)/Nslab)       */

/********************************************/
  void LambdaC(Nslab,dbsqSlab,Bx)
/********************************************/

int Nslab;
float Bx[],dbsqSlab;
{
   int i, j;
   float LC,Bxavsquare,deltaBSQ,error,error1;

   LC = Bxavsquare = deltaBSQ = 0.0;

   /* delta Bsquare Checking */
   for(i=1;i<=Nslab;i++) {
      deltaBSQ += Bx[i]*Bx[i];
   }
   deltaBSQ *= 2.0/Nslab;
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

```
    /* LambdaC Checking */
    for(i=1;i<=Nslab;i++) {
       Bxavsquare += Bx[i]*Bx[i];
    }
    for(j=-(Nslab/16-1);j<=(Nslab/16-1);j++) {
       for(i=1;i<=Nslab;i++) {
          if(i+j >= 1 && i+j <= Nslab) {
             LC += Bx[i]*Bx[i+j];
          } else
          if(i+j > Nslab) {
             LC += Bx[i]*Bx[i+j-Nslab];
          } else
          if (i+j < 1) {
             LC += Bx[i]*Bx[i+j+Nslab];
          }
       }
    }
    LC = LC*delta/(2.0*Bxavsquare);
    error = (LC-lambdaC)*100/lambdaC;
    error1 = (deltaBSQ-dbsqSlab)*100/dbsqSlab;
    printf("\n");
    printf("LambdaC and delta Bsquare Checking\n");
    printf("lambdaC(Cal) = %lf LambdaC = %lf ERROR = %lf%%\n",LC,lambdaC,error);
    printf("deltaBSQ(Cal) = %lf dbsqSlab = %lf ERROR = %lf%%\n",deltaBSQ,dbsqSlab,error1);
}
```

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

Curriculum Vitae

Jaturong Sukonthachat,

1972 Born : Sep, 17th 1972 in Bangkok, THAILAND.
Father : Buaphan Sukonthachat.
Mother : Boontha Sukonthachat.

1990-1994 Bachelor of Science (Physics),
Chiangmai University, Chiangmai, THAILAND.

หอสมุดกลาง สถาบันวิทยบริการ จุฬาลงกรณ์มหาวิทยาลัย

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย