Winter 2011

High Lundquist Number Simulations of Parker's Model of Coronal Heating: Scaling and Current Sheet Statistics Using Heterogeneous Computing Architectures

LiWei Lin
University of New Hampshire, Durham

High Lundquist Number Simulations of Parker's Model of Coronal Heating: Scaling and Current Sheet Statistics Using Heterogeneous Computing Architectures

BY

LiWei Lin

B.A., Physics & Astronomy, Wesleyan University, 2002

Submitted to the University of New Hampshire
in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

in

Physics

December 2011

This dissertation has been examined and approved.

Dissertation Director, Amitava Bhattacharjee

Peter Paul Professor of Physics

Silas R. Beane

Associate Professor of Physics

Terry Forbes

Research Professor of Physics

Kai Germaschewski

Assistant Professor of Physics

Eberhard Möbius

Professor of Physics

Chung-Sang Ng, University of Alaska Fairbanks

Associate Professor of Physics

09/2011

Date

DEDICATION

To my mother and father who have shown me the meaning of fortitude.

ACKNOWLEDGMENTS

I owe my deepest gratitude to my advisors Prof. Chung-Sang Ng and Prof. Amitava Bhattacharjee for their patience, encouragement, and support. I am truly grateful to them for sharing with me their enthusiasm, experience, and insights throughout the course of this dissertation. I would like to also thank my committee members Prof. Terry Forbes, Prof. Eberhard Möbius, Prof. Silas Beane, and Prof. Kai Germaschewski for their careful review of this work and for their many helpful suggestions.

I am also grateful to the many colleagues and friends who have helped enormously me over the years. To my cubicle mates: Carrie Black, Lei Ni, Liang Wang, and Qian Xin. To my colleagues: Peera Pongkitiwanichakul, Parry Junnarkar, Qian Xia, Matt Gilson, Steve Abbott, Chris Schubert, Matt Argall, Trevor Leonard, Allison Jaynes, Matt Gorby, Carol Weaver, Sarah Lakatos, Jishnu Bhattacharyya, Mansi Patil, Becky Barlow, Jun Wang, Josh Barry, Steve Zaffke, Kevin Godin, Kris Maynard, Lijia Guo, Narges Ahmadi, Scott Baalrud, Yi-Min Huang, Douglas Larson, Will Fox, Brian Sullivan, Amber Perkins, Rualdo Soto Chavez, Aveek Sarkar, Fatima Ebrahimi, Li-Jen Chen, Bertrand Lefebvre, and Naoki Bessho. To those with whom I am only separated by time and space: Denise Wee, Sun Mi Chung, Dilyan Donchev, Hyungsoo Kim, Yi-Cheng Lin, An-Pin Lin, Felix Aristo Aridan, Ethel Seno, Eugene Misitski, Chiapo Wang, Chris Webster, and Estelle Lin. To my former teachers and mentors: Patrick Harmon, Vinay Kashyap, Jeremy Drake, and William Herbst. I am indebted to you all.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

ABSTRACT

High Lundquist Number Simulations of Parker's Model of Coronal Heating: Scaling and Current Sheet Statistics Using Heterogeneous Computing Architectures

by
LiWei Lin
University of New Hampshire, December, 2011

Parker's model [Parker, Astrophys. J., 174, 499 (1972)] is one of the most discussed mechanisms for coronal heating and has generated much debate. We have recently obtained new scaling results for a 2D version of this problem suggesting that the heating rate becomes independent of resistivity in a statistical steady state [Ng and Bhattacharjee, Astrophys. J., 675, 899 (2008)]. Our numerical work has now been extended to 3D using high resolution MHD numerical simulations. Random photospheric footpoint motion is applied for a time much longer than the correlation time of the motion to obtain converged average coronal heating rates. Simulations are done for different values of the Lundquist number to determine scaling. In the high-Lundquist number limit ($S > 1000$), the coronal heating rate obtained is consistent with a trend that is independent of the Lundquist number, as predicted by previous analysis and 2D simulations. We will present scaling analysis showing that when the dissipation time is comparable or larger than the correlation time of the random footpoint motion, the heating rate tends to become independent of Lundquist number, and that the magnetic energy production is also reduced significantly. We also present a comprehensive reprogramming of our simulation code to run on NVidia graphics processing units using the Compute Unified Device Architecture (CUDA) and report code performance on several large scale heterogenous machines.

CHAPTER 1

INTRODUCTION

1.1 Coronal Heating

The enormous energy content of high-beta photospheric plasma flows has long been suggested as the source of energy that ultimately heats the million degree solar corona. Unambiguously identifying the exact mechanisms that transfer this kinetic energy into the overlying solar atmosphere and the exact nature of how the coronal magnetic field responds and converts this energy into heat remains one of the longest standing issues in astrophysics.

In this disseration we investigate an idealized model of the corona proposed by Parker (1972) which applies to closed magnetic field structures whose field lines are embedded at both ends in the solar surface. The corona is modeled in Cartesian geometry where an initially uniform magnetic field along the $\hat{e}_z$ direction is "line tied" at $z = 0$ and $z = L$ in perfectly conducting end-plates representing the photosphere. Parker suggests that slow and continuous random shuffling of the footpoints at these end-plates, representing the turbulent buffeting of the coronal field embedded in the convecting photosphere, can tangle the field into a braided structure of sufficient complexity such that it cannot settle into a continuous smooth equilibrium, but rather necessarily evolves to one with tangential discontinuities. Whether or not true current singularities (as opposed to current layers with finite thickness) can form in this scenario and whether or not continuous footpoint mappings necessarily imply a non-smooth topology has been the subject of intense debate in the decades that have passed since Parker's seminal proposal. Extended discussion of this matter is beyond the scope of the present analysis, but it is appropriate to reiterate here (c.f. Ng & Bhattacharjee 1998) that this question is not merely of academic interest. That the plasma gradients will tend towards singularities has important bearing on the physics of magnetic reconnection and turbulence dynamics in the corona. The interested reader is

referred to Ng & Bhattacharjee (1998), Low (2010), Huang et al. (2010), Janse et al. (2010) and references therein.

In a process Parker calls "topological dissipation", it is at these tangential discontinuities where the corona's small but ultimately finite resistivity induces the formation of current sheets where magnetic energy is dissipated to heat the coronal plasma, and where magnetic reconnection proceeds to reduce the topological complexity of the coronal magnetic field. This essential concept was further developed in a series of studies (Parker 1979, 1983a,b, 1988, 1994) and has become known colloquially in the field as the "nanoflare model" of coronal heating. The appellation derives from the isolation of 10^{23} erg flares as the constitutive energy release events which occur in "storms" of sufficient ferocity to heat the corona and adequately account for observed conductive and radiative losses. While the concept of topological dissipation can be seen as the prototypical "DC" mechanism for coronal heating (c.f. Klimchuk 2006; Aschwanden 2004), the solar atmosphere surely admits more complex magnetic topology than is treated by the Parker model. In fact, many investigators have pursued reconnection based heating mechanisms using geometries that include separators, separatrices and magnetic-nulls (see Priest et al. 2005, and references therein), and more recently by analyzing the magnetic topology of active regions observed by TRACE (Lee et al., 2010). It remains clear however, that coronal loops are the basic building block of the solar corona, and their examination first as isolated entities is crucial in laying the foundations for a broader understanding of the corona and its activity (see Reale 2010 for a recent review).

The work we present in this dissertation is motivated by a recent study (Ng & Bhattacharjee, 2008) which developed a simplified version of the Parker scenario. The random braiding at the line tied ends was restricted to depend on only one coordinate transverse to the initial magnetic field. This strong assumption enables us to describe the complete dynamics of the system by a simple set of differential equations which are easily amenable to analytical and numerical solutions for prescribed footpoint motion. The geometric constraints imposed by our assumption preclude the occurrence of non-linear effects, such as reconnection and secondary instabilities, but enables us to follow for long times the dissipa-

tion of energy due to the effects of resistivity and viscosity. Using this model, it was shown both numerically and by scaling analysis that as long as the correlation time of turbulent photospheric flow (τ_c) is much smaller than the characteristic resistive time-scales (τ_R), ohmic dissipation becomes independent of resistivity (η). The absence of non-linear effects in this model allows the perpendicular magnetic field ($B_\perp$) to grow to un-physically large vales and is found to scale as $\eta^{-1/2}$. It was further shown by a simple analytical argument that even in the presence of reconnection and secondary instabilities, the heating rate would remain insensitive to resistivity. It is this conjecture that we examine for this dissertation using three-dimensional hydromagnetic simulations.

1.2 Structure of this Thesis

In Chapter 2 we introduce the numerical scheme and simulation codes used for our coronal heating experiments (Section 2.2). We also motivate our adoption of GPU acceleration on heterogeneous architectures and give an account of our reprogramming experience (Section 2.3). We will report code performance using the latest generation Nvidia architectures on dedicated GPU workstations as well as several distributed memory GPU accelerated machines (Section 2.4).

In Chapter 3 we present a careful analysis of the three-dimensional hydromagnetic simulations and examine these, observing the conjecture put forth by Ng & Bhattacharjee (2008). We provide a detailed comparison of our results to those of Longcope & Sudan (1994) who conducted a scaling study of Parker's model with Lundquist numbers spanning one order of magnitude. In this range they found that both heating rate and perpendicular field production scale as $\eta^{-1/3}$. These numerical results agreed with analysis based on the Sweet-Parker reconnection theory and measurements of current sheet statistics. We will show that we have recovered the scalings for heating rates and $B_\perp$ of Longcope & Sudan (1994) in the range they examined, and extending to lower η, we will show results that support a slower growth of $B_\perp$ which roughly scales as $\eta^{-1/5}$ and a heating rate that becomes insensitive to η. We also demonstrate by simple scaling analysis that the transition between these scaling behaviors results from the diminishing effects of random photospheric motion

as the energy dissipation time-scale (τ_E) becomes much smaller than the correlation time (τ_c), in accordance with Ng & Bhattacharjee (2008). Section 3.2 describes the properties of the Parker model as it evolves in a statistical steady state. Section 3.3 gives the details of the simulation results and Lundquist number scaling. Section 3.4 presents a scaling analysis describing the transition in scaling behavior we observe. Chapter 4 describes preliminary results of an effort to detect and characterize dissipative structures in our hydromagnetic simulations of coronal heating. This analysis provides justification for assumptions made previously in the scaling analysis of Chapter 3. In Section 4.2, we describe an ad-hoc algorithm for current sheet detection which is robust to periodic boundaries and also introduce a procedure to measure sheet parameters in two dimensions. In Section 4.3, we present scaling analysis in support of the theory developed in Chapter 3. In sections 4.4 and 4.5, we briefly introduce the extension of the analysis to three dimensions and future prospects for investigating this aspect of the Parker model.

We conclude the document by briefly motivating two new lines of investigation that will benefit from the theory and tools this dissertation develops. In Section 5.1.1, we introduce a possible mechanism for self-consistent generation of turbulent reconnection in high Lundquist number simulations of coronal loops and report initial proof-of-concept results. In Section 5.1.2 we discuss a possible extension of our analysis to include comparisons to flare frequency distributions and models of self-organized criticality.

Before proceeding with the body of this thesis, the reader is encouraged to review the two papers mentioned above that provide the primary motivation and background for the current analysis. These are Ng & Bhattacharjee (2008) and Longcope & Sudan (1994). Both can be considered required reading for a full appreciation of the work presented here.

CHAPTER 2

REDUCED MAGNETOHYDRODYNAMICS ON HETEROGENEOUS COMPUTING ARCHITECTURES

2.1 Background

The past few years have seen the rapid emergence of graphics processing units as hardware accelerators for general purpose computation and high performance computing. Computational scientists have benefited from GPUs in fields as diverse as geology, molecular biology, weather prediction, high energy nuclear physics (lattice QCD), quantum chemistry, finance and oil exploration. To understand the sudden popularity of the graphics processors for scientific computing, one can begin by distinguishing their development as throughput (tasks per fixed time) rather than latency (time per fixed task) driven architectures Garland & Kirk (2010). Consider the problem of rendering a three dimensional object modeled as a polygon mesh (unstructured grid) in real time, where the smoothness of movement relies on a rendering speed that can match the human eye's refresh rate, and where the number of vertices defines the model's visual fidelity. The GPU is specialized for this, having been designed to render millions of pixels, performing several simple and identical operations for millions of vertices at a time. This is achieved by sacrificing traditionally latency oriented modules on the chips (out of order execution, sophisticated memory, caches, speculative execution), making space on the silicon die for a large number of relatively simple execution cores.

The NVidia Fermi GTX 480 GPU for example fits up to 480 "shader cores" on a chip grouped into 15 "streaming multiprocessors" and clocked at 700 MHz. With a single clock cycle accounting for 2 floating point operations, this amounts to 1344.96 Gflops (Giga Floating Point Operations per Second) in single precision. Contrast this to an Intel Xeon

X5550 Nehalem 2.66GHz Quad Core CPU, which has a theoretical peak in single precision of 170.6 Gflops.

Clearly there is enormous potential here from the raw number of floating point operations available for the GPU. Problems that most benefit from GPU acceleration are those that exhibit a large level of data parallelism as one finds when rendering graphics. Algorithms exhibit data parallelism when a large set of data can be operated on by one or more operations simultaneously and are abundant in the fields of science and engineering. (This is as opposed to task parallelism, where distinct operations or sets of operations are performed on a set of data, same or not, in parallel.)

Two important issues must be considered when deciding whether or not to pursue GPU acceleration. The first is that GPUs are connected to a CPU via the PCI Express (PCIe) BUS. PCIe is a widely supported expansion card standard that allows GPU cards (amongst many other types of cards e.g. high speed network cards, sound cards, TV cards, high speed solid-state hard disks etc.) to be connected to the computer's CPU via the PCIe BUS. (a BUS is a general term for circuits on a motherboard that connect computer components together). While PCIe BUS speeds are typically much faster than typical computer network or Hard Disk access speeds, they are typically much slower than the speeds at which a CPU can access RAM memory. For this reason, for a code to see a significant advantage from GPU acceleration, one must be able to perform a large number of operations for each data transfer performed through the PCIe Bus. The ratio of number of operations performed for each word of data transferred is called arithmetic intensity. Maximizing it is one of the principal challenges of GPU coding, and for many problems, data transfers will simply be too costly for any significant acceleration to be seen. What's more, the on-board GPU memory is restricted to typically less than 2 GB for consumer grade graphics cards and 3-6 GB for more costly dedicated research grade GPUs. For very large problems then, one may typically not be able to simply shuffle the entire problem onto the GPU card, but rather feed the GPU data parallel portions of a code piecemeal.

A second important consideration is the cost of reprogramming the code. In the early years of GPGPU, programmers had to re-cast their science codes in terms of operations

Architecture (Model)	GHz	Cores	Gflops SP(DP)	GB/s	Mem(Gb)
G80(C870)	600	128	518.4(0)	76.8	1.5
GT200(C1060)	600	240	933.12(77.76)	102.4	4
GF100(C2070)	575	480	1288(515.2)	144	6

Table 2.1 Specifications for representative models for three NVidia GPU architectures. Gflops and GB/s are theoretical peak values. Note GF100 is the Fermi architecture.

native to graphics APIs such as OpenGL, Cg, or DirectX9. While many researchers were successful in accelerating their applications (Owens et al., 2007), such specialized knowledge of graphics languages and the hardware they support proved a significant enough barrier to prevent more widespread adoption.

Seeing the potential of general purpose GPU computing, NVidia released the first version of the Compute Unified Device Architecture (CUDA) in May 2007. CUDA is a parallel computing engine developed specifically for general purpose applications using NVidia GPUs. The architecture allows users to leverage the high throughput power by programming in the familiar ANSI C language rather than reverse engineer graphics languages. The CUDA programming model allows programmers to delegate serial tasks required by the CPU by usual C code while extensions to C are provided for programming GPUs to exploit data parallelism. CUDA (now version 4.0 as of this writing) provides libraries for basic linear algebra (CUBLAS), sparse matrices (CUSPARSE), random number generation (CURAND), standard templates (THRUST), and fast Fourier transforms (CUFFT) as well as tools for profiling (Compute Visual Profiler) and debugging (CUDA-gdb).

CUDA has been supported on almost all NVidia GPUs shipped since its initial release. The three latest generations of NVidia GPU architecture have had variants that targeted the general purpose computing specifically, where video output modules (VGA out and DVI out ports) of graphics cards are replaced with additional on board memory. Together with the widespread availability of NVidia GPUs, the relative familiarity of the CUDA C programming interface, and the availability of these HPC targeted devices, the CUDA programming model has garnered a widespread following. It is currently taught at hundreds of universities around the world, and the HPC scholarly literature is now saturated with papers reporting GPU acceleration results.

Numerous widely used scientific codes have been ported to CUDA. These include for example S3D (combustion physics), CHROMA, MILC (lattice QCD), AMBER, NAMD, GROMACS (molecular biology). Currently, three of the five most powerful machines in the current Top 500 list (http://www.top500.org/, November 2011) are accelerated by GPUs and most HPC hardware vendors now carry NVidia GPU accelerated solutions.

In computational plasma astrophysics, several groups have reported progress using GPU acceleration for magnetohydrodynamics (MHD) (Wong et al., 2009; Wang et al., 2010; Zink, 2011), astrophysical gyro-kinetics (Madduri et al., 2011), and particle-in-cell simulations (Stantchev et al., 2008). The work we describe here is closest akin to that of Stantchev et al. (2009) and we were in fact initially motivated by their results. Using a G80 generation NVidia GPU, compared with a single 3.0 GHz Intel Xeon using 1024^2 perpendicular resolution, they report an up to 14× speedup for a Hasegawa-Mima equation solver and 25-30× speedup for a pseudo-spectral RMHD code in single precision and two dimensions. We describe in this chapter a full fledged three dimensional reduced MHD double-precision production code.

In Section 2.2 we describe the reduced MHD approximation and the numerical scheme used to apply it to the Parker model. In Section 2.3 we motivate and present the comprehensive reprogramming of this code for GPU acceleration on heterogenous architectures. In Section 2.4 we report code performance using the latest generation NVidia architectures on dedicated GPU workstations as well as several distributed memory GPU accelerated machines. The work we present in this chapter forms the basis of a refereed journal article (Lin et al. 2011).

2.2 Reduced Magnetohydrodynamics and the Parallel Numerical Scheme

The RMHD equations are a simplified version of MHD applicable to systems where the plasma is dominated by a strong guide field such that the timescales of interest are slow compared with the characteristic Alfvén timescale (τ_A). These restrictions also imply incompressibility ($\nabla \cdot V = 0$) and the exclusion of magnetosonic modes (leaving only the shear Alfvén modes propagating parallel to the guide field in $\hat{e}_z$). The RMHD equations

were first derived for the study of tokamak plasmas by Kadomtsev & Pogutse (1974) and Strauss (1976). They can be written in dimensionless form as

$$\frac{\partial \Omega}{\partial t} + [\phi, \Omega] = \frac{\partial J}{\partial z} + [A, J] + \nu \nabla_\perp^2 \Omega \tag{2.1}$$

$$\frac{\partial A}{\partial t} + [\phi, A] = \frac{\partial \phi}{\partial z} + \eta \nabla_\perp^2 A \tag{2.2}$$

where A is the flux function so that the magnetic field is expressed as $\mathbf{B} = \hat{\mathbf{e}}_\mathbf{z} + \mathbf{B}_\perp = \hat{\mathbf{e}}_\mathbf{z} + \nabla_\perp A \times \hat{\mathbf{e}}_\mathbf{z}$; ϕ is the stream function so that the fluid velocity field is expressed as $\mathbf{v} = \nabla_\perp \phi \times \hat{\mathbf{e}}_\mathbf{z}$; $\Omega = -\nabla_\perp^2 \phi$ is the z-component of the vorticity; $J = -\nabla_\perp^2 A$ is the z-component of the current density; and the bracketed terms are Poisson brackets such that, for example, $[\phi, A] \equiv \phi_y A_x - \phi_x A_y$ with subscripts here denoting partial derivatives. The normalized viscosity (ν) is the inverse of the Reynolds number ($\mathsf{R_v}$), and resistivity (η) is the inverse of the Lundquist number (S). The normalization adopted in equations (2.1) and (2.2) is such that the magnetic field is in the unit of B_z (assumed to be a constant in RMHD); velocity is in the unit of $v_A = B_z/(4\pi\rho)^{1/2}$ with a constant density (ρ); length is in the unit of the transverse length scale $L_\perp$; time t is in the unit of $L_\perp/v_A$; η is in the unit of $4\pi v_A L_\perp/c^2$; and ν is in the unit of $\rho v_A L_\perp$.

Reduced magnetohydrodynamics has continued to see widespread use in the numerical investigations of astrophysical MHD turbulence (e.g. Müller & Grappin 2005; Perez & Boldyrev 2010; Beresnyak 2011), and magnetic reconnection (e.g. Loureiro et al. 2009). They also form the basis of nearly all numerical simulations of the Parker scenario of coronal heating (see Rappazzo et al. 2010; Ng et al. 2011a, and references therein), and notably, recently extended for a density stratified treatment of coronal loops (van Ballegooijen et al., 2011).

The numerical scheme we employ in this work was adapted from Longcope & Sudan (1994) and Longcope (1993). The simulation domain is a rectangular Cartesian box of ($L_z \times L_\perp \times L_\perp$), permeated by guide-field B_z line-tied at both ends representing the photosphere. Time integration is performed with a second-order predictor-corrector method. Perpendicular dimensions are bi-periodic for a pseudo-spectral scheme using standard two-

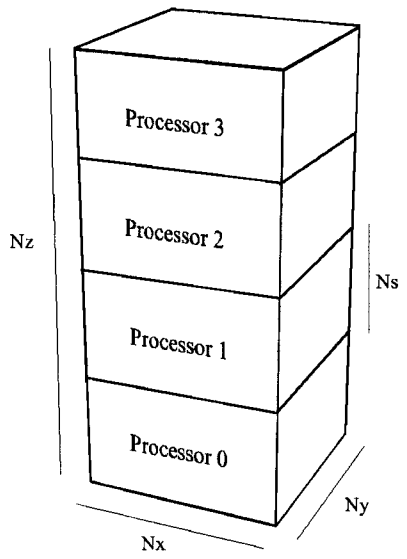

Figure 2-1 The simulation domain is divided in to sub-cubes of size $N_s = N_z/N_p$ where N_p is the number of MPI processors used.

thirds rule de-aliasing, and in the parallel dimension, a second-order finite difference method is used. Using a scheme developed in Longcope (1993), randomized boundary motions are applied at both line-tied ends to mimic turbulent photospheric flows. Convergence analysis and accuracy as compared to finite difference schemes of an equivalent 2D pseudo-spectral scheme to the one used here are reported in Ng et al. (2008) and Ng et al. (2011b).

The original code was written in Fortran, and parallelization is accomplished by domain decomposition in $\hat{\mathbf{e}}_\mathbf{z}$ using Message Passing Interface (MPI) as shown in Figure 2-1. A simulation domain of resolution $N_x \times N_y \times N_z$ is evenly divided into sub-cubes of size $N_x \times N_y \times N_s$ where $N_s = N_z/N_p$, N_p being the number of MPI processors. In this scheme, the maximum number or processors is N_p=N_z. The package is named the Reduced Magnetohydrodynamics Coronal Tectonics (RMCT) code following the language adopted by (Ng & Bhattacharjee, 2008).

Typical resolutions used for our coronal heating scaling study range from $64^2 \times 32$ up to $1024^2 \times 128$ and as with many pseudo-spectral schemes, the 2D FFTs dominate the computational burden typically consuming between 60% and 90% of computation time for the resolutions we target. Figure 2-2 shows profiling results of a run with perpendicular

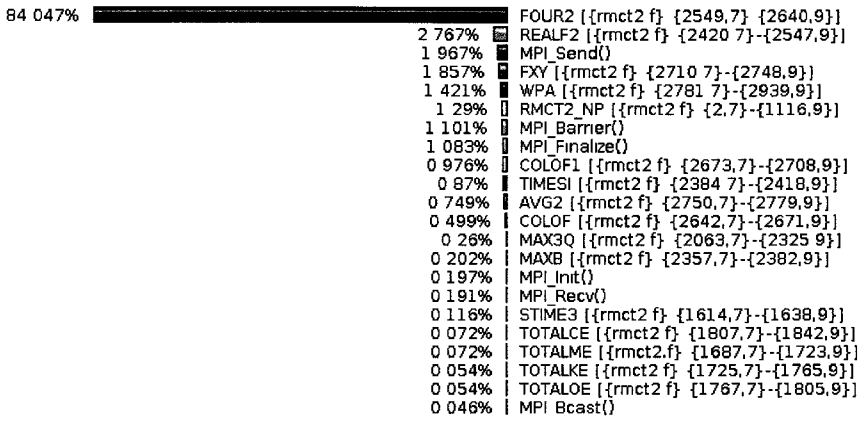

Figure 2-2 TAU Profiling of original RMCT Fortran code at 1024×1024.

resolution of 1024^2 using the TAU profiling suite (Shende & Malony, 2006), where the profiling is exclusive, meaning the timing of a particular routine does not include the time consumed by subroutines it calls. The FFT subroutine $FOUR2$ and its wrapper $REALF2$ together occupy 86.8% of total runtime.

For RMCT, we are most interested in the performance of the CUDA FFT library (CUFFT) compared to our original implementation. At the time of its writing, the original two dimensional FFT subroutine based on Numerical Recipes (Press et al., 1992) was comparable in performance to the industry standard FFTW (Frigo & Johnson, 2005) for the resolutions being considered. Subsequent releases of FFTW have featured the use of Streaming SIMD Extensions 2(SSE2), multi-threading (through Pthreads), and MPI for distributed memory parallelization.

In Figure 2-3 (a) we compare the end to end speed of several FFTW implementations for two dimensional out-of-place, real-to-complex transforms to CUFFT. The comparisons were performed using FFTW 3.3 (released July 2011) on an Intel Nehalem 2.67 GHz quad-core compiled with GCC 4.3.2 and CUFFT on a NVidia C2070 compiled with CUDA 4.0. Speeds were measured for resolutions ranging from 64×64 up to 4096×4096 by averaging over 512 individual FFT invocations for each resolution.

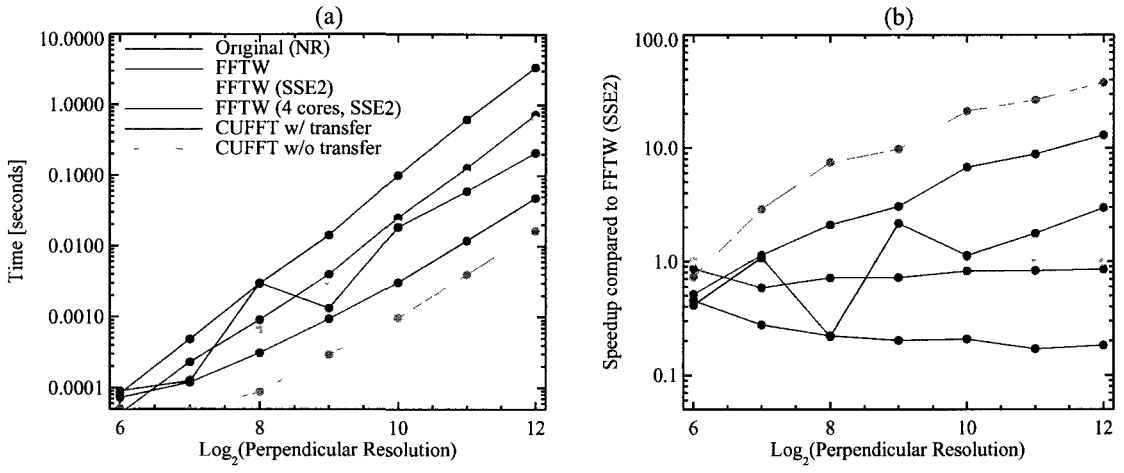

Figure 2-3 GPU (CUFFT) vs CPU (FFTW) comparison for real-to-complex 2-D FFTs.

2.3 GPU implementation

The choice of FFT program to adopt was mediated not only by considering overall speedup but also by considering ease of programming (the project was to be completed within a graduate student dissertation time-line). With relative ease, the RMCT code is adaptable to use the FFTW SSE2 implementation, and we use this as the basis for comparison in Figure 2-3 (b). Evidently this latest version will give a factor 5 improvement over our hand optimized Numerical Recipes subroutines, while going to 4 cores using a multi-threaded scheme (blue) yields an additional factor 2 to 3 for the two highest resolutions (while only marginal at lower resolutions). The CUFFT measurements are plotted both including PCIe memory transfer (purple) and without (green). The former yields roughly an order of magnitude improvement while the latter yields an approximately 20× improvement over the base FFTW implementation and nearly two orders of magnitude improvement over our original subroutines. Also included here for completeness is Figure 2-4, which shows similar results but for complex-to-complex transforms (except for our original Numerical Recipes based subroutines). At the beginning of our re-programming effort, real-to-complex transforms were not yet implemented in CUDA, and the performance observed in Figure 2-3 is what partially motivated us. We forgo any further discussion as the real-to-complex FFT performance is what is most relevant now.

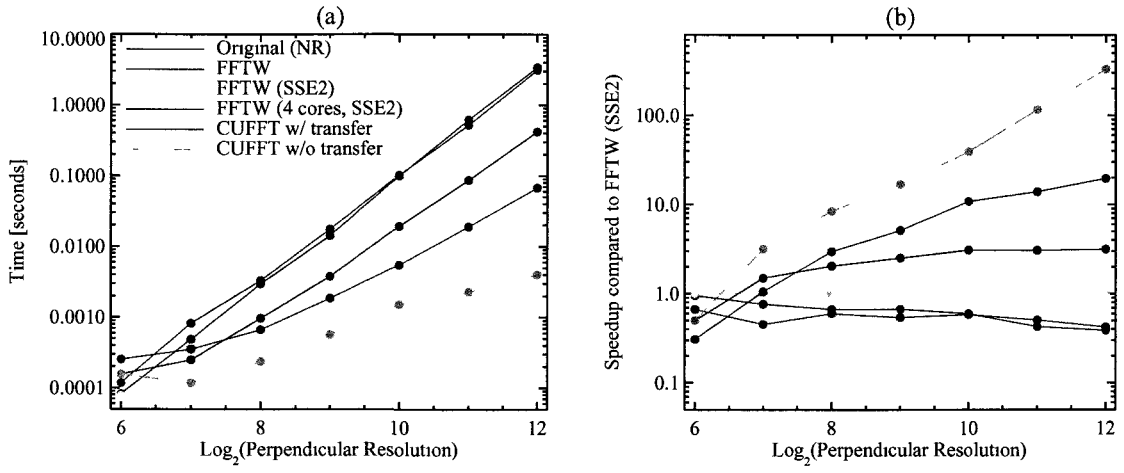

Figure 2-4 GPU (CUFFT) vs CPU (FFTW) comparison for complex-to-complex 2-D FFTs.

Given these remarkable speedups, we must also consider the limitations of Amdahl's law (Amdahl, 1967), which models the theoretical maximum code speedup given a projected improvement of a portion of a code (typically achieved through parallelization). In our code the FFTs typically consume $60 - 90\%$ of total computation time. For 90% and a factor 10 improvement in FFT performance, the resulting overall ideal speedup would be a around a factor 5. For 60%, only optimizing the FFT kernel would yield roughly a factor 2 improvement overall.

The advantage of the CUDA approach is then evident, not only does it admit a pathway for vast improvement with FFTs, the MHD algorithm is otherwise dominated by point-wise arithmetic perfectly suited to the high-throughput GPU architecture. While vectorization via SSE2 on x86 architectures can potentially yield around a factor two improvement for typical kernels (take for example the computation of the Poisson Bracket), a relatively trivial CUDA port yields around a factor 20 improvement.

Given these considerations, our strategy for a comprehensive reprogramming consists of the following:

(a) Minimize the number of memory transfers and maximize the ratio of the number of FFT invocations and point-wise arithmetic kernel invocations per transfer. As evident in Figure 2-3(b), simply replacing the FFTs in the original code with wrappers to

CUFFT invocations would yield a speedup of the order of what FFTW would give. Doing likewise for point-wise arithmetic would either not yield an improvement and might actually be detrimental in some cases.

(b) Recycle memory of intermediate quantities. There is limited memory available on the GPU board so one must adequately budget memory allocated on board in such a manner as to observe (a). Herein is where much of the difficulty in the reprogramming endevour lies. The task is to identify where in the code there exist substantial computing pathways dependent on only a small set of allocated arrays. In practice, what was done was an initial allocation of a set of arrays ($aux1, aux2, ..., aux10$) which would, during the course of the algorithm, each sequentially embody the memory of several quantities. $aux1$ would for example take on quantities a, b, and c which were never needed for the same kernel. Also, if a, for example, is not a derived quantity but rather one that is transferred from main memory, optimally, $aux1$'s re-incarnation as b would not be done until a has completed its full life cycle (thus not requiring a subsequent transfer). Ultimately, this is a constrained optimization problem where the constraint is the memory available on the GPU and one tries to minimize the absolute number of transfers, maximize the number of kernel invocations per transfer, and maximize the number of incarnations an allocated portion of memory inhabits during a full computation cycle. Abstracting the problem to such a level where one could actually plug this into an optimization algorithm perhaps merits further investigation. However, we have taken instead the "artistic route" and leave optimization to the coder's insight and skill (or lack thereof).

(c) Write simple kernels for point-wise arithmetic. Point-wise arithmetic is the bread and butter of stencil-operation based MHD codes, and CUDA implementations of such codes have been published (Wong et al. 2009; Wang et al. 2010; Zink 2011). It is tempting to say that such operations take a back seat in the current pseudo-spectral application to FFTs, but as we have seen Amdahl's law would require that these kernels also see full consideration.

(d) Preserve the underlying MPI decomposition. We are dealing here with two levels of parallelization, the first being the domain decomposition in z and the second through the many-core GPUs. The wide availability of multi-GPU workstations and GPU accelerated distributed memory machines warrants the pursuit of both parallelization methods in tandem. For this code, we pair one CPU core to one GPU, each core responsible for one sub-cube in the the z domain decomposition scheme. As mentioned in the preceding section, the extent of the decomposition is naturally limited in this scheme (as it stands) to $N_p = N_z$. Further parallelization is certainly possible, but we defer this to future investigation as discussed in the chapter conclusion.

Substantially more detailed discussion would not be helpful beyond examining the code itself. It suffices here to describe the result of porting a subroutine $colof()$ which calculates Fourier coefficients of the Poisson brackets by the collocation method and involves both point-wise arithmetic as well as an FFT invocation. The original Fortran subroutines is as follows:

```
c-----------------------------------------------------------------------2
      subroutine colof(t1,t2,t3,t4,bpa,rt)

      real bpa(n1n2),t1(n1n2),t2(n1n2),t3(n1n2),t4(n1n2),rt(n1n2)
      integer i

      bpa = t2*t3 - t1*t4
      call realf2(bpa,1)
c
c transform back to Fourier space
c
      bpa = bpa*rt
      return
      end
c-----------------------------------------------------------------------2
```

For the test case of 1024×1024 resolution, the $colof()$ subroutine typically consumes 1.095 seconds per call inclusively (including the FFT subroutines $realf2()$, and 0.079 seconds per call exclusively, when counting only the point-wise arithmetic operations. The first $t2 * t3 - t1 * t4$ calculates the Poisson bracket in real space while the second $bpa = bpa * rt$ applies a standards two-thirds de-aliasing mask once the array is taken to Fourier space.

Unsurprisingly the FFTs dominate, archetypal of the RMCT code profile as a whole. The equivalent code written in CUDA looks like the following:

```
void colof( Real* t1_d, Real* t2_d, Real* t3_d, Real* t4_d, \
  int NX, int NY, int NXNY, Real* rt, Real *aux, Complex *bpa,\
  cufftHandle plan)
{
  int blocksize = bksze;
  int grid = NXNY/blocksize + (NXNY%blocksize==0?0:1);
  realComm_g<<< grid, blocksize >>>(t1_d,t2_d,t3_d,t4_d,NXNY,aux);
  cfftf_w(bpa,plan);
  complexPtwsMlt_g<<< grid,blocksize >>>(bpa,NXNY,rt);
}
```

The subroutine is broken down into three CUDA kernel calls for *realComm_g*, *cfftf_w*, and *complexPtwsMlt_g*. *realComm_g* calculates the Poisson Bracket, *cfftf_w* is a wrapper that invokes CUFFT kernels. Wrapping is done here both to simplify the syntax and legibility of the code as well as to provide instrumentation targets for the TAU profiling tools. *complexPtwsMlt_g* performs point-wise multiplication operations and is called here to apply the de-aliasing mask. The point-wise arithmetic kernels are written simply as follows:

```
static __device__ __host__ inline Real realComm(Real a1, Real a2, \
  Real a3, Real a4)
{
  Real r;
  r=(a2*a3)-(a1*a4);
  return r;
}
static __device__ __host__ inline Complex complexScale(Complex a, Real s)
{
  Complex c;
  c.x = s * a.x;
  c.y = s * a.y;
  return c;
}
static __global__ void realComm_g(Real* a1, Real* a2, Real* a3, Real* a4, \
  int size, Real *aux)
{
  const int numThreads = blockDim.x * gridDim.x;
  const int threadID = blockIdx.x * blockDim.x + threadIdx.x;
  for (int i = threadID; i < size; i += numThreads) {
    aux[i] = realComm(a1[i],a2[i],a3[i],a4[i]);
  }
}
static __global__ void complexPtwsMlt_g(Complex* aa, int size, Real* scale)
{
  const int numThreads = blockDim.x * gridDim.x;
  const int threadID = blockIdx.x * blockDim.x + threadIdx.x;
  for (int i = threadID; i< size; i += numThreads) {
    aa[i] = complexScale(aa[i], scale[i] ) ;
  }
}
```

It is not the goal here to provide a tutorial on CUDA programming, but rather to just give a flavor of what the equivalent syntax looks like and how the general code re-programming task is conducted. A useful introduction to CUDA programming would unnecessarily add several dozen pages to this document which would largely re-iterate the myriad of excellent programming guides available online and in print. Given the widespread access to the world wide web, self-containment of this document is not relevant. The interested reader is referred to Kirk & Hwu (2010) for a good starting point. We focus on performance here as compared to the original Fortran subroutine.

The re-written *colof* function consumes 5109μs per call inclusively giving an over all factor 239 speedup over the original implementation. The real-to-complex CUFFT kernel is measured at 1748μs per call. (This is somewhat slower than what is reported for this resolution in Figure 2-3 because the measurement was made using CUDA 3.1, while substantial improvements have since been implemented as of CUDA 4.0.) There are two things to note here.

Firstly, for this example, the FFT portion of the routine is sped up by two orders of magnitude. If one were only to re-implement the FFTs in CUDA, the resulting speedup factor for the entire routine would be roughly a factor 12.4. It is the additional re-implementation of the point-wise arithmetic that adds another factor 20 improvement.

Secondly, if two orders of magnitude speedups are measured for this subroutine, archetypal of the program as a whole, then are we to expect such amazing results for the end to end production code? As mentioned earlier in this chapter, and as in fact this short example illustrates, Amdahl's law ultimately prevails. Even if one portion of the code is made infinitely faster, the remainder of the code restricts the overall speedup factor. Most importantly for our application, we have not considered here either CPU to GPU communication or inter-rank MPI communication overhead.

As this example shows, the reprogramming of individual subroutines is fairly straightforward. The true difficulty for this reprogramming task has been the budgeting of the limited on GPU memory in minute detail and the careful management of transfers between main memory and GPU memory, taking care to preserve a beneficial arithmetic intensity.

The fully re-programmed code is a mix of Fortran with MPI, C and CUDA. It is called from BASH shell scripts which coordinate runs and manage file storage and compression. Runs are broken up into sub-runs where a full data cube is written out at a prescribed cadence. Time-series of various quantities (total magnetic energy , kinetic energy, maximum current density, maximum vorticity etc.) are written out to file at a much higher cadence. Restarts are then possible from any of these sub-runs. Visualization and post-processing are carried out in IDL and VisIT.

The re-programming effort has undergone several phases roughly in correspondence to new releases of the Compute Unified Device Architecture as well as NVidia GPU models. The first preliminary re-programming was conducted using CUDA 2.3 which was the first release to feature double-precision accelerated kernels in the CUFFT library. This re-programming was undertaken during a summer term (roughly 4 months). The results of this first port were rather encouraging as the code running on 2 GT200 series GPUs was able to match the performance of 16 nodes (32 AMD Athlon cores) of the Zaphod Beowulf cluster at the University of New Hampshire Space Science Center. Two major revisions of the code have since been undertaken. The first major revision (hereafter called Revision 1) was done to reduce the on GPU memory footprint. At a resolution of $512^2 \times 32$, the entire problem fits on 2 GPUs each bearing 1.4 Gb. For each predictor step for example, half of the simulation domain ($512^2 \times 16$) would be uploaded at the start of the program and processed before the entire half domain is offloaded again for exchanging of ghost cells after which a similar process is done for the corrector step. Such a strategy was clearly not possible on two GPUs when going to a resolutions of $1024^2 \times 128$ or higher. The general strategy was to shuffle the N_z/N_s layers of each sub-domain individually to and from the GPUs rather than as an entire block for each predictor or corrector step. We reduced the GPU memory footpoint to be able to fit 2048^2 on a Tesla C1060 card (4 Gb). This required not just individual transfers at the start and end of a predictor or corrector step, but also careful coordination to refill memory buffers once quantities had reached their useful life cycle as described above. This resulted in various data shuffles and recycled memory buffers

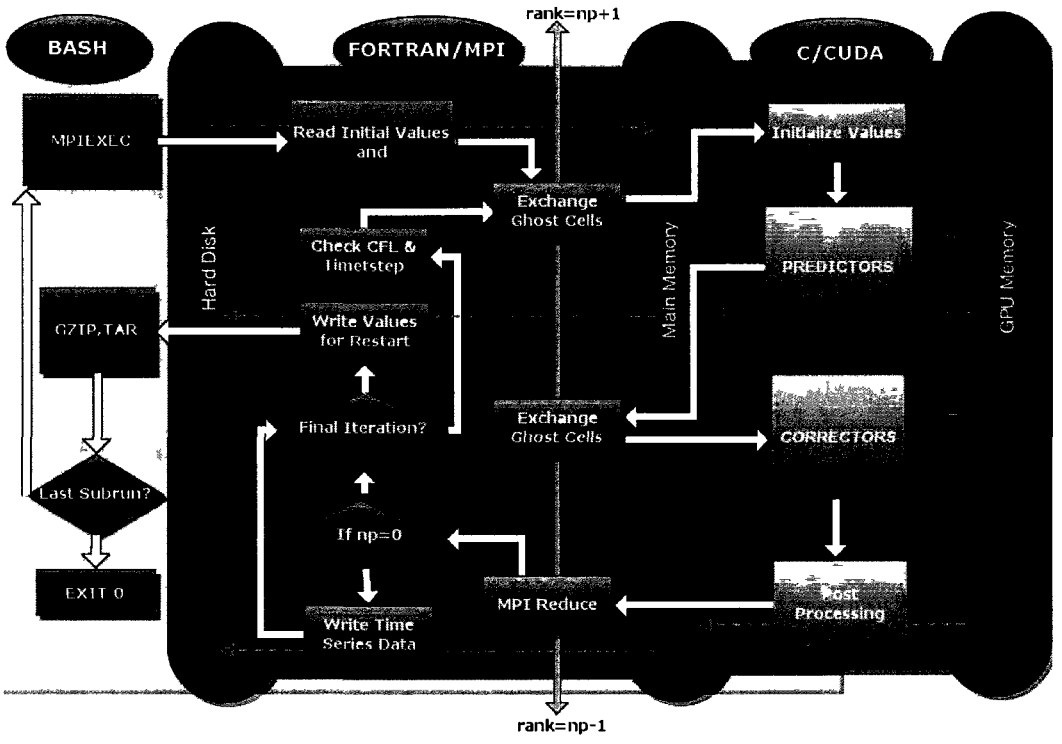

Figure 2-5 This flow chart shows the general structure of the RMCT CUDA package

within the main body of the code. Implementing this scheme while maximizing arithmetic intensity was the principle challenge of the effort.

A second major revision (Revision 2) was undertaken after the release of CUDA 3.2 which featured real-to-complex transforms in CUFFT. The initial CUDA 2.3 release featured only complex-to-complex transforms while two subsequent releases featured real-to-complex transforms as only wrappers to complex-to-complex transforms. CUDA 3.2 fully exploits the roughly factor 2 reduction in both computation and storage cost of considering hermitian symmetry of real transforms. The revision not only accommodated the use of real-to-complex transforms but also allowed further optimization of the data shuffling and memory recycling scheme. The additional memory savings were tuned so that perpendicular resolutions of 4096^2 could fit on the 5.25 Gb available on NVidia C2070 GPUs.

Figure 2-5 shows a flow-chart of the general scheme of the code package. The flow-chart is a hybrid schematic which shows both the general life-cycle of the simulation (white arrows) as well as memory transfers (green arrows). Dotted green arrows show intra-rank

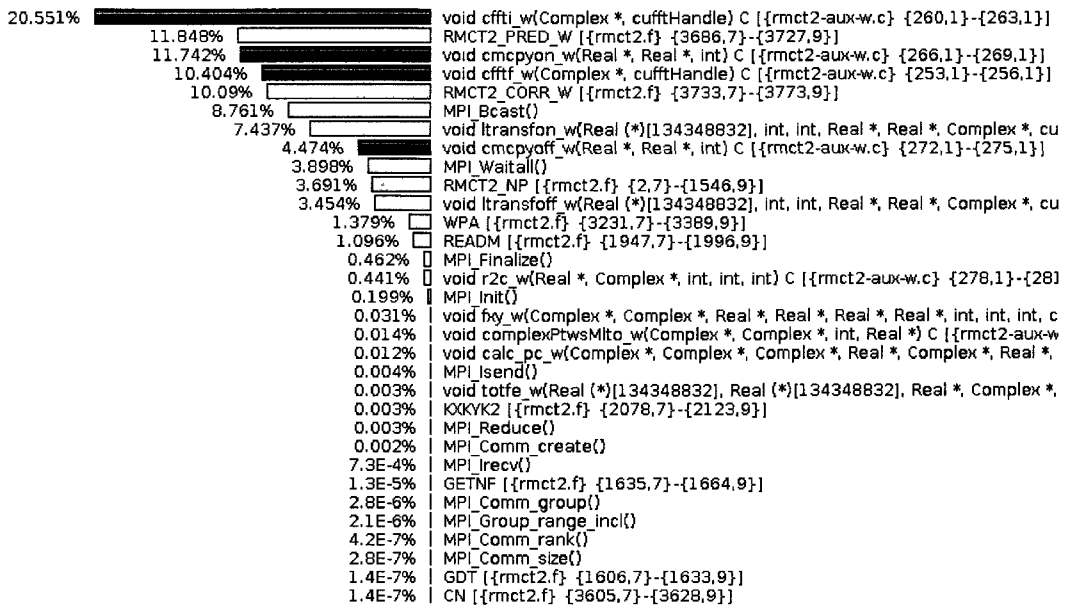

Figure 2-6 TAU Profiling of CUDA RMCT port for resolution 1024×1024.

memory transfers while thick green arrows show inter-rank transfers. Note, these are not individual transfer events, but rather ensembles of transfers just represented schematically. A full description would render this chart even more illegible. White arrows begin and end in code blocks which are color coded according to the language they are written in (Brown=Bash, Violet=Fortran, Green=MPI, and Orange=CUDA). The green arrows begin and end in blue columns each representing each type of memory (Hard Disk, Main Memory (RAM), and GPU Memory).

Writing out full data cubes to the Hard Disk is actually the most expensive single task in the program, however, this is only executed at a prescribed cadence typically of the order of 1 full output for every 1000-10000 time-steps. Given that the CFL condition limits the time-steps to much smaller than the dynamical time-scales of interest, the cadence is largely at the discretion of the user. Setting this cadence is important when considering for example: rendering visual animations, job-queuing on large scale parallel machines, and post processing tasks like current sheet characterization which we describe in detail in Chapter 4. Writing time series data to disk is relatively cheap and done at a much higher cadence, typically once every 50 to 100 time steps.

Figure 2-6 shows profiling results (exclusive) for the code after the first major revision. The profile is no longer completely dominated by the FFTs (Figure 2-2), but rather they occupy less than a third of total run time (summing $cfftf\_w$ and $cffti\_w$). CPU - GPU transfers (*cmcpy_on* and *cmcpy_off*) occupy roughly 25% of total time while MPI communication takes less than 15%. The remaining GPU tasks (mostly dominated by point-wise arithmetic kernels) are included in $RMCT\_PRED\_W$ and $RMCT\_CORR\_W$ totaling roughly 22%. The remainder here are I/O tasks and initialization overhead which can be rendered essentially negligible choosing an adequate sub-run cadence as just discussed.

This new rough equipartition in computing budget makes it possible to achieve a useful optimization to the code from a variety of perspectives now. It is no longer the case that only upgrading the FFT implementation would give a useful speedup. Rather, addressing any one of these code aspects merits consideration, be it through new hardware (faster CPU-GPU bus speed, faster interconnect, more powerful GPU) or further re-coding (further MPI decomposition, optimizing the arithmetic kernels, asynchronous GPU transfers, intra-GPU transfers, concurrent kernels etc.) Before further discussion of future code enhancements, we first present end-to-end code performance of current production versions of the RMCT CUDA code on a variety of hardware configurations.

2.4 Code Performance and Scaling

The effective equivalence of 2 GT200 generation NVidia GPUs with 32 AMD Athlon cores measured with the preliminary version of the CUDA RMCT port set an encouraging precedent for what we could expect from further development. Plotted in Figure 2-7 are wall clock timings for full production versions of RMCT at three resolutions: $512^2 \times 64$, $1024^2 \times 128$, and $2048^2 \times 256$. These measurements exclude I/O tasks and were measured by manually instrumenting the code with built-in GCC timing tools in Fortran.

Measurements were made for each of five machines whose relevant specifications are summarized in Table 2.2. For a base-line comparison we use Carver, an IBM iDataplex traditional CPU cluster at the National Energy Research Scientific Computing Center (NERSC) on which we have achieved the best performance thus far with the original version of the

code. Carver hosts a GPU testbed cluster named Dirac which accelerates each of 44 nodes with a single NVidia C2050 GPU. A larger dedicated GPU production cluster at the National Center for Supercomputing Applications (NCSA) was available through TeraGrid, and paired 96 S2070s (4 GT200 GPU module) with 192 nodes. A dedicated GPU accelerated desktop workstation was acquired for this project and is running at the University of Alaska Fairbanks with four C2050 GPUs. The most powerful machine on which we have tested is the Keeneland Initial Delivery System hosted at the National Institute for Computational Science (NICS) and Georgia Tech. It features 120 nodes each accelerated by 3 C2070s , and is to be expanded to a full production system for TeraGrid (now called XSEDE) using next generation NVidia GPUs in the coming year.

In Figure 2-7, Carver measurements are plotted with horizontal dashed lines labeled with the number of cores used for the run. The labels are color coded to indicate resolutions: $512^2 \times 64$ (Green), $1024^2 \times 128$ (Blue), $2048^2 \times 256$ (Red). A similar color scheme is employed for the GPU machine scaling results. Increasing the resolution here from $512^2 \times 64$ to $1024^2 \times 128$ we see that we get a factor $\sim$7.6 increase in time per simulation step and another factor $\sim$6.3 going up to $2048^2 \times 256$. As we have described above, the original code scales up to $N_p = N_z$.

Using Revision 1 of the CUDA code (memory optimized) measurements were made for NCSA/Lincoln. On this machine, the code scales reasonably well up to 128 GPUs (64 for the smallest resolution case). Using Revision 2 of the CUDA code (memory optimized + real-to-complex CUFFTs) we have measurements for the UAF workstation (open circles) with 4 GPUS, NERSC/Dirac up to 32 GPUs, and up to 264 GPUs for NICS/Keeneland. Note we have used slightly larger grids for Keeneland to accommodate for the 3 GPUs per node configuration. With this version we have the remarkable result of being able to match or exceed the performance of NCSA/Carver at full scale at the three resolutions considered (up to 256 cores using the highest resolution) using a 4 GPU desktop workstation.

It is important to consider here that this original Fortran implementation does not employ parallel FFTs and does not use the industry standard FFTW package. Therefore, at a resolution of $1024^2 \times 128$ for example, the code will scale at most to 128 CPU cores given

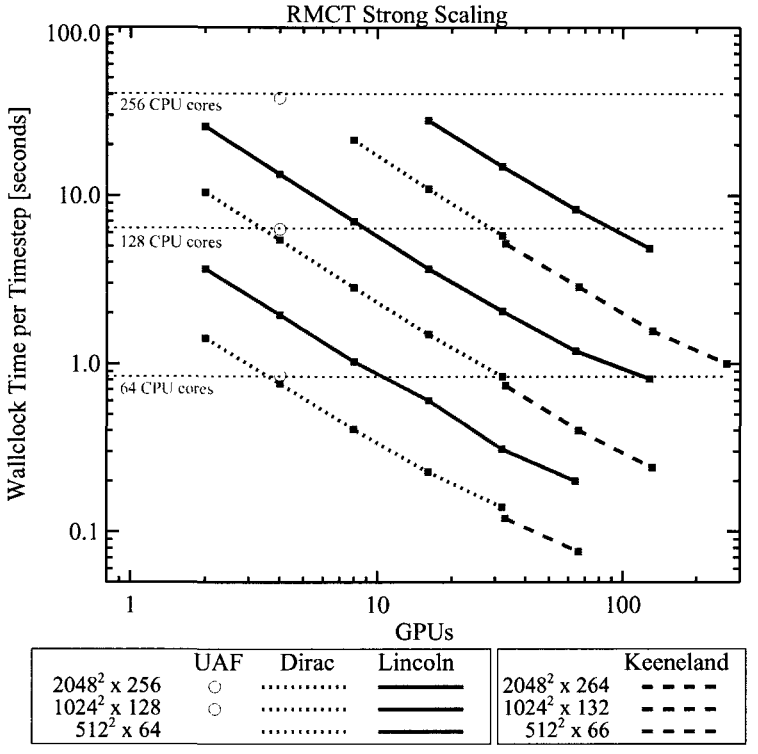

Figure 2-7 Strong scaling of RMCT CUDA on various large scale distrubuted memory machines. Horizontal lines correspond to the best performance of the original Fortran/MPI code on NERSC/Carver. Open circles show measurements on the 4 GPU workstation at the University of Alaska Fairbanks Geophysical Institute.

the simple domain decomposition scheme. We have admittedly not exhausted all avenues for further optimization. As evident in Section 2.3 significant speedups may be pursued by using SSE and multi-threading enabled FFTW with the latest release (FFTW 3.3 July 2011). Exploiting SSE2 for the remaining computational code also merits consideration.

The same however, can be said of the CUDA implementation. The Fermi architecture enables concurrent kernel execution admitting a further pathway to on GPU parallelization. Significant savings can also be made by combining simple point-wise arithmetic into more complex kernels minimizing thread launching overheads. Several third-party GPU FFT implementations exist which report improvements in speed for large transform sizes. With CUDA 3.2 (November 2010), the CURAND library allows for on GPU random number generation, precluding the need of the Fortran based serial CPU generator. In fact, a

Name	CPU	Nodes	GPUs	Network
Carver/NERSC	Nehalem(8 core)2.67 GHz	400	None	QDR
Lincoln/NCSA	Harpertown(4 core)2.33GHz	192	96 x S1070	SDR
Dirac/NERSC	Nehalem(8 core)2.67 GHz	44	44 x C2050	QDR
UAF Workstation	Gulftown(12 core)2.8 GHz	1	4 x C2050	-
Keeneland/NICS	Westmere(6 core)2.67 GHz	120	360 x C2070	QDR

Table 2.2 Specifications for Carver/NERSC and several GPU accelerated machines.

complete re-write of the Fortran/MPI portion of the code into C/MPI would itself prove beneficial. Although virtually no overhead is incurred by calling C from Fortran, the CUDA porting was done in such a way as to preserve the original MPI implementation as well as I/O interface. Because of this the predictor corrector steps were coded as separate C-functions which would each re-initialize several values and re-transfer certain quantities for each time-step. These extraneous re-initializations and transfers actually account for about 25% of all transfers. CUDA 4.0 (May 2011) also features GPU-to-GPU transfers, which would preclude then need for CPU-GPU transfers for exchanging ghost cells altogether.

With these caveats stated we can make general qualified statements that directly compare the CPU only and CPU-GPU codes. As the codes currently stand, the CUDA port on Lincoln/NCSA at full scale is able to achieve roughly an order of magnitude speedup compared with Carver. Considering the performance of the code on the UAF workstation, the chip-to-chip equivalence is roughly 4, 8, and 16. As expected, the possibility of exposing maximal fine-grained parallelism given finer grids improves the potential for increasing this equivalence ratio. Actually observing an increase in this ratio then attests to the quality of the re-coding effort. At full scale on Keeneland, using the Fermi class GPUs and Revision 2, the code is able to achieve up to a 30$\times$ speedup beyond what was previously achieved on Carver.

2.5 Current Status and Future Work

Reporting on code performance when going to new computing architecture as we just have is a thorny issue. Given the significant cost of acquiring and operating computing hardware, be it for a small academic research group or at large scale for a government

research lab, together with the significant time investment required to re-program large production codes, it is necessarily contentious. The sunk costs and opportunity costs for the programmer who has invested substantial effort in a recoding project can illicit a somewhat protective response to criticism of optimization results, and at worst a subjective reporting of code performance.

This has been an issue since the early days of high performance computing on large scale distributed memory architectures. A classic paper in the field (Bailey, 1991) is entitled "Twelve Ways to Fool the Masses When Giving Performance Results on Parallel Computers." Of the twelve ways stated, we are in the most danger of being guilty of two. "2. Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment." We have given results for the dedicated 4 GPU workstation at UAF and compared them to measurements on Carver which during the timing runs was in full production. It could be argued however, that this is in fact a deserved advantage of the GPU approach, where a large part of the motivation to purchase such a dedicated machine is to in fact avoid contention with other jobs be it during queuing or during runtime. "6. Compare your results against scalar, unoptimized code on Crays." In a modern interpretation, we are comparing our GPU results to a not fully optimized traditional parallel implementation. As just stated in the previous section, neither code is fully optimized for their target architecture. A recent paper by researchers at the Intel corporation "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing" on CPU and GPU" addresses this exact issue, claiming that in light of the PCIe memory transfer overhead, most applications would see a roughly 2.5× improvement in performance and at most a factor 14× improvement. NVidia researchers responded in kind with a short online public relations article entitled "GPUs are only up to 14 times faster than CPUs says Intel" (Keane, 2010) which then proceeded to list 9 research papers reporting greater than $100X$ speedups. We will take no sides here or beleaguer the points any further but rather just refer the reader to Bailey (2009) and Hager (2010) which both give updated versions of the classic Bailey paper which consider the current surge of interest in heterogenous high per-

formance computing. They delineate guide-lines to prevent abuse and misleading reporting of code performance in light of GPU hardware acceleration.

While we cannot give definitive numbers regarding chip-to-chip comparisons, what we do report here is a comparison of currently practically deployable codes. The coronal heating scaling experiment presented in the next chapter was largely undertaken using the original Fortran code on conventional hardware. The computational campaign was completed roughly over the course of a year and a half at two academic computing clusters, Midnight at the Arctic Region Supercomputing Center and Zaphod at the University of New Hampshire Space Science Center. The reprogramming of the code for CUDA was conducted while this computational campaign was well underway. It has since been deployed for production on two machines presented in this chapter UAF workstation and NCSA/Lincoln. running at the highest resolution $1024^2 \times 128$ to be included in the coronal heating study. NCSA/Lincoln was made available through a TeraGrid allocation of 500,000 SUs of which two-thirds were consumed before the machine was retired in April 2011. A renewal allocation of 600,000 SUs has been awarded on NCSA/Forge which features 38 nodes each equipped with 8 M2070 GPUs and replaced NCSA/Lincoln during Summer 2011. The renewal request focuses on a novel application of the RMCT CUDA code, self-consistent generation of MHD turbulence in high Lundquist number coalescing flux tubes. Because we target $2048^2 \times 256$ for this project the issue of data management and post processing requires attention given that a single sub-run for this project can require 30 GB storage. In the closing chapter of this document, we provide brief discussion of preliminary results of this project as well as a novel strategy of run-time post processing, where other-wise fallow CPU-cores are used to perform post processing tasks while co-processing GPUs are in production.

Currently, code development is being performed on the Keeneland Initial Delivery System with a program director discretionary allocation. This machine is set to be substantially expanded in early 2012 and integrated into TeraGrid (now called XSEDE) for production allocations. A projected 10-20 Petaflop class GPU accelerated machine named Titan will begin to come online in the same time-frame at Oak Ridge National Laboratory. The machine will be an upgrade of Jaguar (currently #3 on the Top 500 list) and likely be accelerated by

Nvidia's forthcoming Kepler series GPU, which will feature three times the double-precision performance per watt of the current NVidia Fermi architecture. Depending on budgeting considerations currently being deliberated, Titan might allow the United States to regain the top spot in the Top 500 list. The machine is to be made available through the Department of Energy Novel Computational Impact on Theory and Experiment (INCITE) program.

CHAPTER 3

HIGH LUNDQUIST NUMBER SCALING IN THE PARKER SCENARIO

3.1 Introduction

The work we present in this chapter is motivated by a recent study (Ng & Bhattacharjee, 2008) which developed a simplified version of the Parker scenario for coronal heating (Parker, 1972). As we discussed in the introductory chapter, Parker's model applies to closed magnetic field structures whose field lines are embedded at both ends in the solar surface. The corona is modeled in Cartesian geometry where an initially uniform magnetic field along the $\hat{e}_z$ direction is "line tied" at $z = 0$ and $z = L$ in perfectly conducting end-plates representing the photosphere. Parker suggests that slow and continuous random shuffling of the footpoints at these end-plates, representing the turbulent buffeting of the coronal field embedded in the convecting photosphere, can tangle the field into a braided structure of sufficient complexity such that it cannot settle into a continuous smooth equilibrium, but rather necessarily evolves to one with tangential discontinuities. It is at these discontinuities where current sheets form to heat the plasma ohmically, and where magnetic reconnection proceeds to reduce the topological complexity of the magnetic field.

Ng & Bhattacharjee (2008) simplify this model by restricting the random braiding at the line tied ends to depend on only one coordinate transverse to the initial magnetic field. This strong assumption enables a description of the complete dynamics of the system by a simple set of differential equations which is easily amenable to analytical and numerical solutions for prescribed footpoint motion. The geometric constraints imposed by the assumption preclude the occurrence of non-linear effects, such as reconnection and secondary instabilities, but enables us to follow for long times the dissipation of energy due to the effects of resistivity and viscosity. Using this model, Ng & Bhattacharjee (2008) show both numerically and by

scaling analysis that as long as the correlation time of turbulent photospheric flow (τ_c) is much smaller than the characteristic resistive time-scales (τ_R), ohmic dissipation becomes independent of resistivity (η). The absence of non-linear effects in this model allows the perpendicular magnetic field ($B_\perp$) to grow to un-physically large vales and is found to scale as $\eta^{-1/2}$. It was further shown by a simple analytical argument that even in the presence of reconnection and secondary instabilities, the heating rate would remain insensitive to resistivity. It is this conjecture that we examine here using three-dimensional hydromagnetic simulations.

The Parker model has been studied extensively using three-dimensional MHD numerical simulations (Mikic et al. 1989; Longcope & Sudan 1994; Einaudi et al. 1996; Hendrix et al. 1996; Galsgaard & Nordlund 1996; Dmitruk et al. 1998; Gomez et al. 2000; Rappazzo et al. 2008, 2010 amongst others). Here, we are interested in the precise scaling of dissipation with respect to plasma resistivity. Our study is most similar in design to that of Longcope & Sudan (1994) who used reduced MHD to simulate Parker's model with Lundquist numbers spanning one order of magnitude. In this range they found that both heating rate and perpendicular field production scale as $\eta^{-1/3}$. These numerical results agreed with analysis based on the Sweet-Parker reconnection theory and measurements of current sheet statistics.

In this chapter, we will show that we have recovered the scalings for heating rate and $B_\perp$ of Longcope & Sudan (1994) in the range they examined. Also, when extending to lower η, we will show results that support a slower growth of $B_\perp$, which roughly scales as $\eta^{-1/5}$, and a heating rate with a much weaker dependence on η.

We also demonstrate by simple scaling analysis that the transition between these scaling behaviors results from the diminishing effects of random photospheric motion as the energy dissipation time-scale τ_E becomes much smaller than the correlation time τ_c, in accordance with Ng & Bhattacharjee (2008).

The chapter is organized as follows. Section 3.2 describes the properties of the Parker model as it evolves in a statistical steady state. Section 3.3 gives the details of the simulation results and Lundquist number scaling. Section 3.4 presents a scaling analysis describing the transition in scaling behavior we observe.

3.2 Statistical Steady State

Our principal aim is a continuation of Ng & Bhattacharjee (2008) who conjecture that for sufficiently small τ_{coh} compared with τ_R , regardless of the specific saturation level of $B_\perp$ of the mechanism causing the saturation, the ohmic dissipation will become insensitive to resistivity. The numerical code described in Section 2.2 was run in a range of η spanning two orders of magnitude. The basic parameters and results in this scaling analysis are summarized in Table 3.1. It is crucial to the scaling study that we obtain good statistics when averaging over time evolution in statistical steady state. As with previous long time integration studies of the Parker model, the runs are started with a vacuum potential field (just a uniform guide field in $\hat{e}_z$). After a time of the order of the resistive diffusion time, the system will evolve to a statistical steady state.

Figure 3-1 shows the intermittent nature of various quantities in time for runs R5(Blue) and R12(Red) (see Table 3.1). Figure 3-1 shows total magnetic energy (a), maximum current density (b), ohmic dissipation (c), and $B_\perp$ (d). In each of these quantities, we see that the lower η case (R5) saturates to larger levels. These figures are discussed at length in the next section. For brevity, we only summarize some salient properties of this steady state already discussed extensively by previous investigators :

1. Energy conservation: The Poynting flux injected at the photosphere is eventually accounted for completely by either numerical or explicit dissipation (both resistive and viscous) (Longcope 1993; Longcope & Sudan 1994; Rappazzo et al. 2008; Hendrix et al. 1996; Galsgaard & Nordlund 1996; Rappazzo et al. 2010). Note that in these simulations, energy essentially disappears in the system once dissipated. No energy term or transport equations are included. This is perhaps the primary weakness of this model, as it prevents the model from predicting temperature and density profiles, which can be directly compared with observations. Energy is dissipated impulsively, as Poynting flux injection progressively braids the fields, energy is built up until an instability drives current sheet formation and reconnection, after which energy is released in a short time.

2. Partition of Energy: Most energy is not kinetic but rather contained in the magnetic field and most energy is lost via ohmic dissipation.

3. Not a Markovian process: Both Longcope (1993) and Hendrix et al. (1996) showed that singular current layers do not form directly at the sites of (and as a result of) footpoint displacements, but it is rather the ensuing dynamics of the tangled magnetic field that ultimately gives rise to current sheets.

4. Turbulence has been studied in various numerical experiments of Parker's model (Hendrix et al. 1996; Dmitruk et al. 1998; Rappazzo et al. 2008). As with Hendrix et al. (1996), energy spectra in our simulations are largely exponential during relatively quiescent periods with little or no impulsive energy release, but become progressively shallow power laws during particularly intense current sheet disruption events. Similar to their study however, computation grid resolutions available up to now only allow us to resolve a few decades of the developing energy cascade. The latter two studies used continuous boundary flows, which induced persistent turbulent states. While it seems that turbulence plays just a minor role in our present analysis, it is believed by many investigators to have a crucial role in determining the speed of magnetic reconnection. We discuss this further in Section 3.5.

3.3 Scaling Measurements

We have performed a series of simulations using our 3D RMHD code described in Section 2.2, using a range of η spanning two orders of magnitude to study scaling laws. Extending the range in η by an order of magnitude beyond what was studied by Longcope & Sudan (1994) poses a significant challenge. As the dissipation coefficients (η and ν) get smaller, higher resolutions have to be used to resolve smaller scales.

The range of η has been extended to lower values (with $\tau_c = 10 \ll \tau_r$) for about an order of magnitude as compared with the study in (Longcope & Sudan, 1994), which stopped at $\eta \sim 10^{-3}$. This extension, of course, requires significant increase in resolution, with our highest resolution case at $1024^2 \times 128$ so far, as compared with $48^2 \times 10$ in (Longcope &

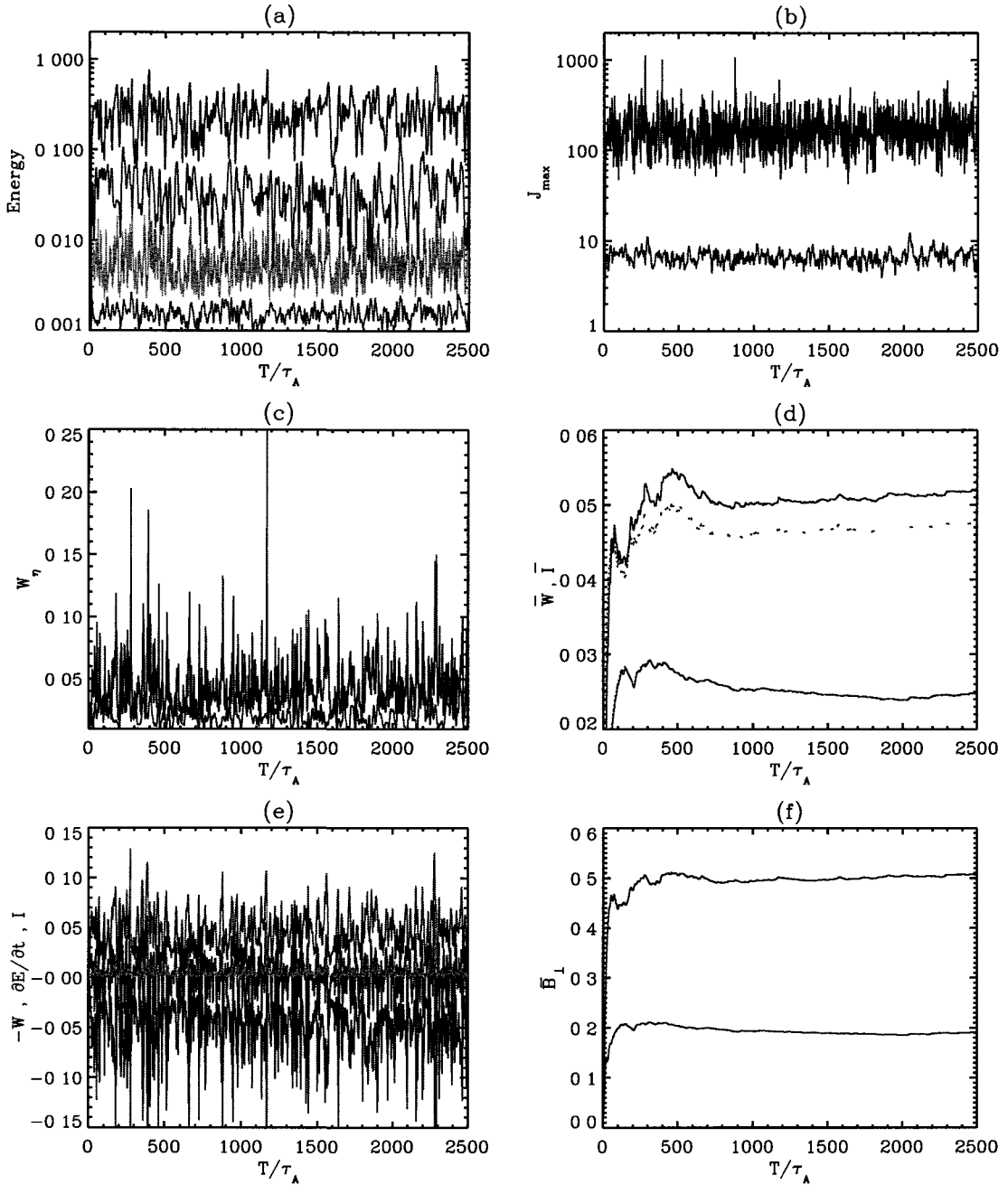

Figure 3-1 Plots (a),(b),(c) (d) and, (f) show time series of various quantities for runs R5 (Blue) and R12 (Red). In (a), green and orange corresponds to E_K for R5 and R12 respectively while blue and red show E_M. For plot (d), solid lines show $\bar{W}$ and dotted lines show $\bar{I}$. For run R3 (e), shows $-W = -(W_\eta + W_\nu)$ (Blue), I (Pink), $d(E_M + E_K)/dt$ (Purple), and the difference between the right and left hand sides of Eq. 3.1 (Green). Parameters used for these runs can be found in Table 3.1.

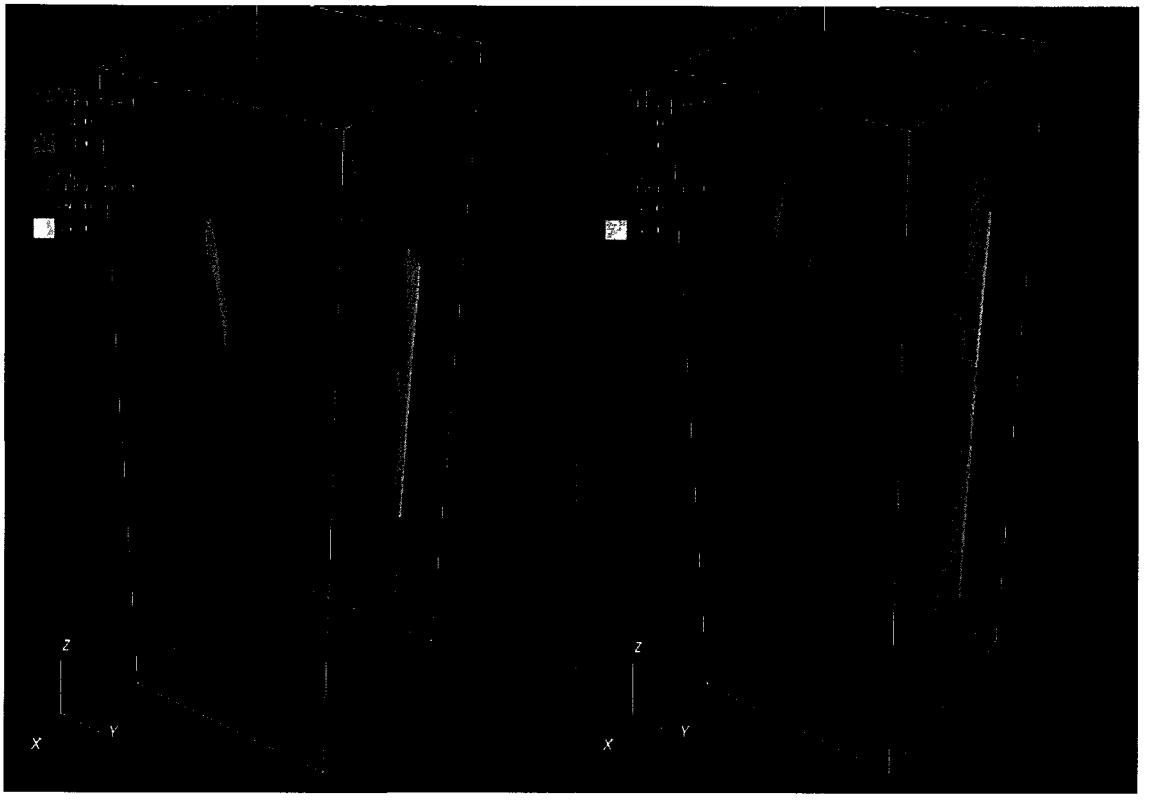

Figure 3-2 3D iso-surfaces of J at a time taken from the R7 run. Both figures are from the same time sample for the R7 run. The left panel shows iso-surfaces at $J = -60, -24, 24, 60$ while the right panel shows iso-surfaces at $J = -36, -12, 12, 36$. Blue and green iso-surfaces are made semi-transparent for greater visibility.

Sudan, 1994). The main difficulty in performing these simulations is the requirement to run up to hundreds or even thousands of Alfvén times in order to obtain good statistics of the average quantities under the driving of random boundary flow. The basic parameters and results in this scaling analysis are summarized in Table 3.1.

It is crucial to the scaling study that we obtain good statistics averaging over time evolution in statistical steady state. As with previous long time integration studies of the Parker model, the runs are started with a vacuum potential field in $\hat{e}_z$. After initial transients, the system will evolve to a statistical steady state. As mentioned above, thin current layers are formed and dissipated repeatedly during this statistical steady state. Figure 3-2 shows 3D iso-surfaces of J at a time taken from the R5 run, when there is a larger number of current sheets. This process is repeated indefinitely as the random

boundary flows keep twisting the magnetic fields. The energy of the system is dissipated impulsively. As Poynting flux injection progressively braids the fields, energy is built up until an instability drives current sheet formation and reconnection, after which energy is released in a short time. This is a major characteristic of the statistical steady state. Figure 3-1 shows the intermittent nature of various quantities in time in the unit of the Alfvén time τ_A, (the time it takes for Alfvén waves travel the distance of $L = 1$ between the two boundary plates along z) for runs R5(Blue) and R12(Red) (see Table 3.1).

Figure 3-1 (a) shows the total magnetic energy $E_M = \int \mathbf{B}_\perp^2 d^3x$, as well as total kinetic energy $E_K = \int \mathbf{v}_\perp^2 d^3x$, where the integration is over the 3D simulation box. Note that the magnetic energy does not include the contribution from the B_z component, which is constant in the RMHD model. Since the applied photospheric flow is chosen to be small (less than one tenth) compared with the Alfvén speed (with $\tau_c = 10\tau_A$), the magnetic field configuration maintains quasi-equilibrium for most of the time, excpet when strong current sheets episodically form and induce instabilities and strong dissipation. Therefore, E_M is usually much larger than E_K. Figure 3-1 (b) shows the maximum current density $J_{\max}$ over the whole 3D volume. $J_{\max}$ increases over an order of magnitude on average in R5 compared with R12, and also fluctuates in time over a much larger amplitude. Observing Figures 3-1(a) and 3-1(b), note that the ratio of the increase in $J_{\max}$ is much larger than the ratios of the increases of both E_M and E_K as η decreases.

Figure 3-1 (c) shows the Ohmic dissipation $W_\eta = \eta \int J^2 d^3x$. Similarly, there is also energy dissipated by the viscous effect, with a rate of $W_\nu = \nu \int \Omega^2 d^3x$ (not shown). For the same reason that E_K is much smaller than E_M, the viscous dissipation is much smaller than the Ohmic dissipation if we choose numerically $\nu = \eta$ (Prandtl number equal to unity), which holds for most of our simulations. The total energy dissipation rate (heating rate) in this case is dominated by Ohmic dissipation. If we use ν values much greater than η however, as we have in some of our trial runs, viscous dissipation can become a more significant fraction of the Ohmic dissipation. From the plot of W_η, we see that it fluctuates a lot in time. However, we can also see that it is fluctuating around a certain level, which is a little higher for R5 than R12 due to smaller resistivity.

Table 3.1. Summary of Numerical Runs

Run	η	ν	$B_\perp$	$S_\perp$	W_η	W_ν	Poynting	T/τ_A	Resolution	N_{tot}
R0	0.00015625	0.00015625	0.540	3450	0.0444	0.0102	0.0586	301.345	$1024^2\times128$	...
R1	0.00015625	0.00015625	0.537	3440	0.0468	0.0127	0.0546	487.252	$512^2\times64$	...
R2	0.00015625	0.00062500	0.610	3900	0.0513	0.0283	0.0519	245.546	$512^2\times32$	1330
R3	0.00015625	0.00062500	0.614	3930	0.0498	0.0275	0.0458	77.0269	$512^2\times32$	...
R4	0.00031250	0.00031250	0.492	1570	0.0433	0.00792	0.0491	857.407	$512^2\times64$	...
R5	0.00031250	0.00031250	0.503	1610	0.0452	0.00941	0.0478	9321.75	$256^2\times32$	...
R6	0.00031250	0.00062500	0.502	1610	0.0431	0.0111	0.0467	2032.02	$256^2\times32$	2047
R7	0.00062500	0.00062500	0.449	718	0.0416	0.00540	0.0427	19342.8	$128^2\times32$	...
R8	0.00062500	0.00062500	0.448	717	0.0399	0.00502	0.0401	820.339	$128^2\times32$	3509
R9	0.0012500	0.0012500	0.372	298	0.0370	0.00332	0.0385	11668.2	$128^2\times32$	...
R10	0.0012500	0.0012500	0.371	297	0.0373	0.00336	0.0411	706.141	$64^2\times16$	6947
R11	0.0025000	0.0025000	0.279	112	0.0299	0.00272	0.0311	1317.70	$64^2\times16$	2086
R12	0.0050000	0.0050000	0.183	36.7	0.0215	0.00317	0.0252	2566.96	$64^2\times16$	2129
R13	0.0100000	0.0100000	0.103	10.3	0.0132	0.00394	0.0168	5209.60	$64^2\times16$	...
R14	0.020000	0.020000	0.0547	2.73	0.00822	0.00511	0.0123	10245.4	$64^2\times16$	...
R15	0.040000	0.040000	0.0307	0.767	0.00623	0.00544	0.0113	10240.3	$32^2\times64$	...
R16	0.080000	0.080000	0.0197	0.246	0.00550	0.00612	0.0105	10240.5	$32^2\times64$	...
R17	0.000078135	0.000078135	0.537	6880	0.0467	0.0158	0.0472	128.496	$1024^2\times128$	...

To give a better measure of the level of energy dissipation, we can calculate the time averaged energy dissipation rates, e.g., $\bar{W}_\eta = [\int_0^t W_\eta dt']/t$, and similarly for $\bar{W}_\nu$. The total energy dissipation rate is then $\bar{W} = \bar{W}_\eta + \bar{W}_\nu$, which is plotted on Figure 3-1 (d). Our physical assumption here is that such averaged quantities will tend to saturated levels as t tends to infinity. In practice, since we can only simulate for a finite amount of time, such saturated levels are found at a time $t \gg \tau_c \gg \tau_A$, when these time averaged values are not fluctuating too much. We do see from this plot that $\bar{W}$ is saturating at a rather constant level after a time much larger than τ_A.

Also plotted on Figure 3-1 (d) is the time averaged Poynting flux $\bar{I}$, where $I = B_z \int \mathbf{v} \cdot \mathbf{B} d^2x$, integrated over the top and bottom boundary surfaces with $\mathbf{v} = \mathbf{u}_p$, the random photospheric flow. Note that I is not positive definite due to the fact that it involves the dot product between the velocity and magnetic field vectors and thus can be either positive or negative. However, the time averaged $\bar{I}$ is almost always positive due to two factors. First, due to Ohmic and viscous dissipation of energy into heat, if the total energy of the system is at a statistically steady level, there must be energy input from the boundary to provide this dissipation loss. Secondly, even when there is not much energy dissipation during a certain period, magnetic energy E_M generally increases, since the magnetic footpoints at the two

boundaries connected to the same magnetic field line will move apart from each other in a random walk fashion due to random photospheric motion. Therefore, a typical magnetic field line will generally be stretched by the separation of the footpoints, and the magnetic energy of the system increases. This increase in the magnetic energy must come from the Poynting flux. We see from Figure 3-1 (d) that $\bar{I}$ also saturates at a rather constant level in the long time limit, i.e. a level close to that of $\bar{W}$. In principle, these two rates should be the same, since the time averaged total energy also tends to a constant level. Numerically there is a slight difference between the two. Convergence studies show that this is mainly due to inaccuracy from finite resolution, and thus the difference decreases when higher resolutions, especially in the parallel direction, are used.

Another measure of the accuracy is to test the energy balance equation,

$$\frac{d(E_M + E_K)}{dt} = I - W_\eta - W_\nu \tag{3.1}$$

Figure 3-1 (e) shows I as a function of time in pink for the run R5, $d(E_M + E_K)/dt$ (calculated by taking finite difference in time) in purple, $-W_\eta - W_\nu$ in blue, and the difference between the right and left hand sides of Eq. 3.1 in green. We do see that the residual power due to numerical inaccuracy is generally small compared with other terms. While accuracy can be improved by running at higher resolutions, doing so would require much longer run times and ultimately limit the highest Lundquist numbers that can be simulated. In the context of energy balance in our simulations, we remark that the energy dissipated due to Ohmic or viscous terms is essentially converted into thermal energy. No energy term or transport equations are included. This is perhaps the primary weakness of this model, as it prevents us from predicting temperature and density profiles which can be directly compared with observations (See Dahlburg et al. 2009). However, the heating rate required to maintain observed coronal temperatures can indeed be estimated as has been done in, e.g., (Priest et al., 2002). Ng & Bhattacharjee (2008) followed this practice and found that the heating rate determined from 2D simulations is consistent with such estimation, if the energy dissipation does turn into heat as assumed. Readers should compare similarities

and differences between this treatment with those used in other studies (Longcope 1993; Longcope & Sudan 1994; Rappazzo et al. 2008; Hendrix & van Hoven 1996; Hendrix et al. 1996; Galsgaard & Nordlund 1996; Rappazzo et al. 2010).

Figure 3-1 (f) shows $\bar{B}_\perp$ as a function of time. $\bar{B}_\perp$ is defined as a root-mean-square value of the magnetic field strength, and so is effectively the square root of E_M per unit volume. Similar to other time-averaged quantities, $\bar{B}_\perp$ also saturates at a rather constant level in later time. We note here that these saturated levels in the 3D runs are already much more reasonable than those in the 2D runs that were found to have a scaling of $\bar{B}_\perp \propto \eta^{-1/2}$, which can be much larger than unity (the value of the constant B_z used in the simulations). Therefore, including 3D effects can reduce $\bar{B}_\perp$ back to reasonable values that are less than B_z. These effects include the formation of thin current layers, onset of instabilities, and subsequent reconnection and enhanced energy dissipation. All these effects are more prominent when $B_\perp$ is larger, effectively limiting the growth of $B_\perp$. Thus, these 3D effects can self-regulate the level of $B_\perp$ that can be built up when subjected to the driving of the random footpoint motion.

Because we are injecting energy into the system through random photospheric footpoint motion, a natural question to ask is whether this would induce other random processes, such as a turbulent cascade of energy that contributes to the heating of the corona. Indeed, turbulence has been studied in various numerical experiments of Parker's model (Hendrix et al. 1996; Dmitruk et al. 1998; Rappazzo et al. 2008). However, as mentioned in the above discussion, we are driving with slow boundary flows (less than 1/10 of the Alfvén speed) with $\tau_c \gg \tau_A$, and thus the magnetic field configuration maintains quasi equilibrium most of the time. Moreover, we apply random boundary flows, instead of constant motion as in, e.g., Rappazzo et al. (2008), so that energy injection is much slower due to the fact that magnetic field lines are stretched in a random walk fashion rather than at a constant rate. As a result, we have $E_K \ll E_M$, which is not consistent with quasi equipartition of energy in Alfvén wave turbulence. Similar to Hendrix et al. (1996), energy spectra in our simulations are largely exponential (not shown here) during relatively quiescent periods, with little or no impulsive energy release, but become progressively shallow power laws during particularly

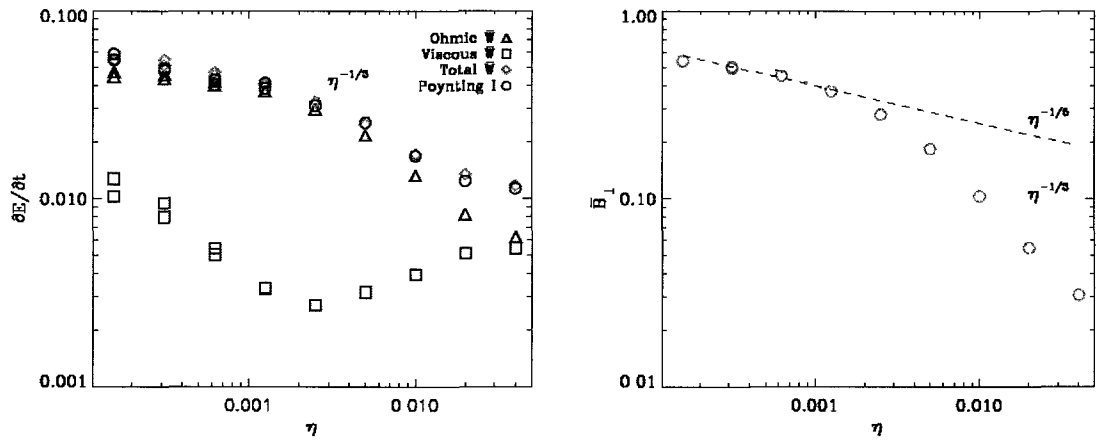

Figure 3-3 (a) Average energy dissipation rate for different values of η. $\triangle$ is Ohmic dissipation, $\square$ is viscous dissipation, $*$ is the total of the two, and $\diamond$ is the footpoint Poynting flux. (b) Average perpendicular magnetic field strength for different values of η.

intense current sheet disruption events, with possible excitation of more Alfvén waves for a short duration. As in their study however, computation grid resolutions available up to now only allow us to resolve less than a decade of the inertial range of energy cascade.

While it seems that turbulence plays just a minor role in our present analysis, whether it plays a crucial role in determining the speed of magnetic reconnection has attracted a number of recent investigations (e.g. Lazarian & Vishniac 1999; Smith et al. 2004; Fan et al. 2004; Loureiro et al. 2007; Bhattacharjee et al. 2009; Loureiro et al. 2009; Kowal et al. 2009; Kulpa-Dybel et al. 2009). It is evident that there is a surge of interest in numerical experiments concerning turbulent reconnection, and that much has yet to be settled. It would be interesting to see if any insights can be gleaned from our own data. As mentioned above, the presence of turbulence seems to be intermittent in our simulations, presenting mainly during intense impulsive current sheet disruption events. Of crucial importance is how well resolved we can be and how extensive an inertial range we can identify. This will depend on how low a value of η (and thus how high a Lundquist number S) we can simulate in 3D, as well as how important physical properties scale with η or S. To this we turn our attention to now.

Fig. 3-3 shows some of the scaling results we have obtained so far. In Fig. 3-3 (a), the time-averaged Ohmic dissipation rate $\bar{W}_\eta$ (at the saturated level), for different η for the runs

listed in Table 3.1 is plotted in triangles, while the viscous dissipation rate $\bar{W}_\nu$ is plotted in squares. As pointed out above, $\bar{W}_\nu \ll \bar{W}_\eta$ in general, and thus the total dissipation rate (heating rate) $\bar{W} = \bar{W}_\eta + \bar{W}_\nu$ (plotted in asterisks), is very close to $\bar{W}_\eta$, except in the large resistivity limit, which is not the focus of the current study but is included for completeness. The time-averaged Poynting flux $\bar{I}$ is also plotted in the same graph in circles. It is supposed to be of the same value as $\bar{W}$ theoretically, and we do see that the differences between these two quantities are generally small in our numerical results, indicating acceptable accuracy.

From this plot, we see that $\bar{W}$ actually only changes within an order of magnitude, and levels off at both the large and small η limit. This has important implications for the coronal heating problem, since the Lundquist number (on the order of the inverse of the normalized η in our simulations), can be as high as 10^{14} in the solar corona. Therefore, the leveling off of $\bar{W}$ at the small η limit is especially important, and is in fact predicted by Ng & Bhattacharjee (2008) based on 2D simulations and theoretical arguments. As mentioned above, this level of $\bar{W}$ was shown in Ng & Bhattacharjee (2008) to be independent of the dissipation mechanism provided that the correlation time τ_c is small compared with the time over which magnetic energy is accumulated. It was also estimated that this level of heating rate can give the same order of magnitude required for realistic coronal heating, following similar considerations in Priest et al. (2002). However, the amount of magnetic energy built up in this process does depend on the dissipation mechanism and becomes un-physically large in 2D simulations in the small η limit (with $\bar{B}_\perp$ scales as $\eta^{-1/2}$). We will now show that this scaling becomes much weaker in 3D.

Fig. 3-3 (a) shows the time-averaged $\bar{B}_\perp$ (at the saturated level) for different η. This is a measure of the magnetic field (or magnetic energy) production at the statistical steady state due to the applied random photospheric motion. Unlike $\bar{W}$, $\bar{B}_\perp$ production changes over an order of magnitude from large to small η. This is because in the high resistivity limit, magnetic field produced is quickly dissipated and can only reach a low magnitude, while the dissipation rate does not decrease that much. At the small resistivity limit, the increase of $\bar{B}_\perp$ slows down significantly.

Because we are doing high resolution 3D simulations and need to simulate for a long time to obtain good statistics, so far we have only been able to extend the value of η to about an order of magnitude lower, as compared with similar studies in (Longcope & Sudan, 1994). Nevertheless, we can see already that below $\eta \sim 10^{-3}$, there is a significant deviation from the scalings obtained in (Longcope & Sudan, 1994), who showed by numerical results and scaling analysis that both $\bar{W}$ and $\bar{B}_\perp$ should scale with $\eta^{-1/3}$ in the small η limit. We have added dotted lines in Fig. 3-3 (a) and (b) showing the $\eta^{-1/3}$ scaling. We see that the portion of the data in a range close to $\eta \sim 10^{-3}$ is indeed consistent with a $\eta^{-1/3}$ scaling. However, as described above, both $\bar{W}$ and $\bar{B}_\perp$ increase much slower with the decrease of η for even smaller η. This result has important implications on the solar coronal heating problem, since the Lundquist number in the solar corona is so high, and thus we most likely need to have a mechanism to provide coronal heating that is independent of the Lundquist number in order to get physically reasonable heating rate. At the same time, the magnetic field energy production should not increase to unreasonable levels compared with observations. In addition to this numerical evidence, we will provide our own scaling analysis to make sense of these results, as well as compared with results in (Longcope & Sudan, 1994).

3.4 Transition in Scaling Behavior

We have shown an initial confirmation of the hypothesis of Ng & Bhattacharjee (2008), but as an additional goal we would like to understand the exact mechanism giving rise to saturation. Their conjecture made clear that the insensitivity to η holds true no matter what the saturation mechanism is. In order to provide a more complete numerical confirmation of their conjecture, it becomes necessary to identify the possible physical mechanisms behind saturation.

A natural place to begin would be to examine the results of Longcope (1993) and Longcope & Sudan (1994) who derived scaling laws based on Sweet-Parker reconnection theory and analyzed a range in η which we have covered in our own study. By looking at where their scaling behavior or where their assumptions might be failing in our own numerical results, we might gain some insight into the physics occurring at even lower η.

The reader is referred to these papers for a detailed review of their scaling arguments. Here, we will only discuss their assumptions and results briefly.

They assumed Sweet-Parker theory is valid in the sense that when looking at only the current sheet region that forms between two coalescing islands (flux tubes), the reconnection can be treated as a steady process in resistive MHD, which results in the classic Sweet-Parker scaling relating the width δ and length Δ of a reconnecting current sheet:

$$\delta/\Delta \sim S_{\perp}^{-1/2} \tag{3.2}$$

Here, $S_{\perp} \equiv \bar{B}_{\perp} w/\eta$ is the perpendicular Lundquist number, with w being the perpendicular length scale of the reconnecting islands, and so $w \sim v_p \tau_c$ with v_p being the root-mean-square value of the random photospheric flow velocity. They also observed that both the number of current sheets N in the simulation box and the length of the current sheets Δ are relatively insensitive to resistivity. We follow these assumptions as a starting point of our discussion, although we recognize that some of them need to be re-examined more carefully. We revisit this issue in some detail in Chapter 4.

The Sweet-Parker reconnection theory should apply only to higher Lundquist number (smaller η) cases, in which the energy dissipation is dominated by the reconnection process. Therefore, the scaling analysis presented here should not work for larger η (i.e. $\eta > 0.01$ here), which is actually not within the focus of our studies here. When the energy dissipation is mainly from the Sweet-Parker current sheets, the dissipation rate can be estimated by

$$\bar{W} \sim \eta N \Delta L \frac{\bar{B}_{\perp}^2}{\delta} \sim \frac{\bar{B}_{\perp}^2 L L_{\perp}^2}{\tau_E} \tag{3.3}$$

where we have used the estimation that the current density of the current sheet is given by $J \sim \bar{B}_{\perp}/\delta$ and that the volume of the simulation box is $LL_{\perp}^2$. The energy dissipation time scale τ_E in Eq. (3.3) can then be solved as

$$\tau_E \sim L_{\perp}^2/N(\eta w \bar{B}_{\perp})^{1/2} \tag{3.4}$$

where we have used the Sweet-Parker scaling in Eq. (3.2). In a statistical steady state, the energy dissipation by Eq. (3.3) has to be replenished by the production of magnetic field energy due to the footpoint motion within the same amount of time τ_E.

In the studies of Longcope (1993) and Longcope & Sudan (1994), although random photospheric motion was used in the simulations, the effects due to such random flows were not taken into account in their scaling analysis. This can be justified if τ_c is much larger than the energy dissipation time τ_E. In this case, the magnetic field strength production is given by

$$\bar{B}_\perp \sim B_z \frac{v_p \tau_E}{L} \sim \left[\left(\frac{B_z v_p}{LN} \right)^2 \frac{L_\perp^4}{w\eta} \right]^{1/3} \tag{3.5}$$

where we have used Eq. (3.4) and solved for $\bar{B}_\perp$. Putting back Eq. (3.5) into Eq. (3.4) results in

$$\tau_E \sim \left(\frac{L_\perp^4 L}{N^2 w B_z v_p \eta} \right)^{1/3} \tag{3.6}$$

and so the energy dissipation rate becomes

$$\bar{W} \sim \left(\frac{L_\perp^{10} B_z^5 v_p^5}{L^2 N^2 w \eta} \right)^{1/3} \tag{3.7}$$

after putting Eqs. (3.5) and (3.6) into Eq. (3.3). Note that all three of these quantities, $\bar{B}_\perp$, τ_E, and $\bar{W}$ scale with $\eta^{-1/3}$, and thus we have recovered scaling laws derived in Longcope (1993) and Longcope & Sudan (1994), but using a slightly different approach.

We may now put reasonable numbers into Eqs. (3.5) to (3.7) and compare with our simulation results. Our simulations are set up to use $L = L_\perp = B_z = 1$. The root-mean-square photospheric flow velocity is measured numerically to be $v_p \sim 0.075$, and thus $w \sim v_p \tau_c = 0.75$. The average number of current sheets N is more difficult to determine and is subject to some uncertainties. A preliminary analysis of our simulations for different η gives $N \sim 7$ numerically in the small η limit. This seems to be somewhat higher than expected from the number of reconnecting islands (flux tubes). However, it is actually quite common to see multiple current sheets in a simulation output, as shown in Figure 3-2. We discuss this in more detail in Chapter 4

Based on these values, we have $\tau_E \sim 0.71/\eta^{1/3}$, and so $\tau_E \sim 7.1$ for $\eta = 10^{-3}$. At the same time we get $\bar{B}_\perp \sim 0.53$ from Eq. (3.5), and $\bar{W} \sim 0.04$ from Eq. (3.7) at the same η. Both of these are close enough to the values found in Fig. 3-3 (a) and (b), and so it is an indication that our parameters used in these estimates are consistent with simulations. Compared with the value of $\tau_c = 10$, we see that although τ_E is still smaller than τ_c, it is getting to about the same level and thus Eq. (3.5) is only marginally justified. For larger η, τ_E is smaller, e.g., $\tau_E \sim 3.3$ for $\eta = 0.01$ and thus is much smaller than τ_c so that the random effect is not as important. This qualitatively explains why we see from Fig. 3-3 (a) and (b) that there is a range roughly around $\eta \sim 0.01$ to 0.001 where both $\bar{W}$ and $\bar{B}_\perp$ scale approximately as $\eta^{-1/3}$, as indicated by the two dotted lines in the two plots. However, for smaller η, τ_E becomes larger, e.g., $\tau_E \sim 15$ for $\eta = 10^{-4}$ (neglecting that this estimate might be no longer valid) and thus it is larger than τ_c such that the random effect should be important. This explains the deviation from the $\eta^{-1/3}$ scaling for both $\bar{W}$ and $\bar{B}_\perp$ for η smaller than around 10^{-3}.

Now, taking into account the effect of random boundary flow, which makes the footpoints move in a random walk fashion as argued in Ng & Bhattacharjee (2008), the estimate for magnetic field production must be changed from Eq. (3.5) to

$$\bar{B}_\perp \sim B_z \frac{v_p(\tau_c \tau_E)^{1/2}}{L} \sim \left[\left(\frac{B_z v_p L_\perp}{L} \right)^4 \frac{\tau_c^2}{N^2 w \eta} \right]^{1/5} \tag{3.8}$$

where we have again used Eq. (3.4) and solved for $\bar{B}_\perp$. Substituting Eq. (3.8) back into Eq. (3.4) results in

$$\tau_E \sim \left[\frac{L_\perp^8}{N^4 \tau_c} \left(\frac{L}{w B_z v_p \eta} \right)^2 \right]^{1/5} \tag{3.9}$$

and so the energy dissipation rate becomes

$$\bar{W} \sim \frac{L_\perp^2}{L} B_z^2 v_p^2 \tau_c \tag{3.10}$$

after putting Eqs. (3.8) and (3.9) into Eq. (3.3), and thus it is independent of η. Note that Eq. (3.10) is exactly the same as found in Ng & Bhattacharjee (2008) for systems regardless

of dissipation mechanism, and is estimated to give the same order of heating consistent with observations.

Using the same values of L, $L_\perp$, B_z, τ_c, v_p, w, and N, Eq. (3.9) becomes $\tau_E \sim 0.42/\eta^{2/5}$, and thus $\tau_E \sim 6.7$ for $\eta = 10^{-3}$, if we could apply this equation. This turns out to be very close to $\tau_E \sim 7.1$ estimated above using Eq. (3.6), which indicates that the transition point between these two regimes of scalings is around $\eta = 10^{-3}$ in our simulations. For $\eta = 10^{-4}$, Eq. (3.9) gives $\tau_E \sim 17$, which is significantly larger than τ_c, and so these scalings based on random walk of footpoints are justified.

Based on this set of parameters, Eq. (3.10) predicts $\bar{W} \sim 0.056$ (independent of η), which is close to the asymptotic values found in Fig. 3-3 (a) in the small η limit. We do see from this plot that $\bar{W}$ indeed does not increase as fast when η is below 10^{-3}, and is consistent with a trend to a constant level in small η, although we still only have a limited range of η that we can simulate. At the same time, Eq. (3.8) gives a value of $\bar{B}_\perp \sim 0.97$, which is somewhat larger than expected from Fig. 3-3 (b), although we do need to recognize that there are uncertainties in these scaling estimates.

A better test of Eq. (3.8) would be the scaling with η in the small η limit. In Fig. 3-3 (b), we have also plotted a dashed line indicating the scaling of $\eta^{-1/5}$. We do see that this seems to be consistent with a portion of the data of $\bar{B}_\perp$ below $\eta \sim 10^{-3}$. However, we cannot rule out the possibility that $\bar{B}_\perp$ is actually increasing slower than $\eta^{-1/5}$, possibly due to a modification of the Sweet-Parker reconnection scalings, e.g., Eq. (3.2). We will further discuss this possibility in the next section.

3.5 Discussion and Conclusions

In this chapter we have presented an analysis of Lundquist number scaling of ohmic dissipation and perpendicular field production based on numerical simulations of a 3D RMHD model of solar coronal heating with random photospheric motion. These simulations were performed over a period of more than two years, and numerical results have been verified carefully to eliminate possible errors. So far, we have been able to simulate cases with η about one order of magnitude smaller than those presented in similar studies in

Longcope (1993) and Longcope & Sudan (1994). While this extension seems modest, it actually requires much more computational efforts due to the increase in resolution and running time required, as well as the decrease of time-step for numerical stability. To be able to achieve that, we have been running our simulations in parallel computers, as well as using GPU acceleration as described in Chapter 2.

Moreover, we have shown that the extension of this scaling study towards smaller η turns out to have very important physical consequences. Numerically, we have shown that the scaling laws (with $\bar{W}$ and $\bar{B}_\perp$ scale with $\eta^{-1/3}$) found in Longcope (1993) and Longcope & Sudan (1994) become invalid for η smaller than what was used in their studies (around $\eta \sim 10^{-3}$). Both $\bar{W}$ and $\bar{B}_\perp$ are now found to be increasing much slower for smaller η, with $\bar{W}$ possibly leveling off to an asymptotic value. We have presented our own scaling analysis to justify our numerical results. By following similar assumptions as in Longcope (1993) and Longcope & Sudan (1994), e.g., using Sweet-Parker scalings, we have been able to recover their $\eta^{-1/3}$ scaling laws for a range of η larger than 10^{-3}. We have demonstrated that the transition between scaling behaviors derives from the fact that the effects of random photospheric motion are not important in the larger η range where the energy dissipation time τ_E is smaller than the correlation time τ_c of the random flow. For η smaller than around 10^{-3}, τ_E becomes comparable or even larger than τ_c. In this range, an analysis based on the random walk of photospheric footpoint motion predicts the insensitivity to η we observe, further substantiating the results found in Ng & Bhattacharjee (2008), which were based on 2D simulations and more general theoretical considerations. This is important to the problem of coronal heating since this heating rate has been shown to be consistent with the requirements for coronal heating (Ng & Bhattacharjee, 2008). This result shows that random photospheric motion is indeed important in simulations of the coronal heating process. Therefore, our results are significantly different from those simulations using photospheric motion that is steady in time, (e.g., Rappazzo et al. (2008)).

We have also shown that now $\bar{B}_\perp$ has a much weaker scaling with η, i.e., $\eta^{-1/5}$ instead. This is much better than the $\eta^{-1/2}$ scaling in 2D simulations, as well as weaker than the $\eta^{-1/3}$ scaling found in Longcope (1993) and Longcope & Sudan (1994). This scaling predicts

a more physically realistic level of magnetic field as compared with observations. However, because the Lundquist number ($\sim$ inverse of η) in the solar corona can be very high (up to 10^{12} to 10^{14}), even a $\eta^{-1/5}$ scaling would result in an unrealistically large magnetic field, despite a much weaker dependence. The reason behind this is the fact that the Sweet-Parker reconnection rate, which scales with $\eta^{1/2}$ is too slow for high Lundquist numbers.

One solution for this problem is the possibility of a higher rate of magnetic reconnection even under resistive MHD. This possibility has attracted a number of recent investigations, (e.g. Lazarian & Vishniac 1999; Smith et al. 2004; Fan et al. 2004; Loureiro et al. 2007; Bhattacharjee et al. 2009; Loureiro et al. 2009; Kowal et al. 2009; Kulpa-Dybel et al. 2009). Many of these studies fall within the scope of turbulent reconnection. While there are some indications that $\bar{B}_\perp$ found in our simulations might actually scale weaker than even $\eta^{-1/5}$, we still have not been able to simulate even smaller η to confirm this definitively. Moreover, the effects due to turbulence are still too difficult to study using our currently achievable resolutions. However, this question is important enough that we are trying different ways to extend our range of η to even smaller values to study these effects.

In summary, by simulating with η about one order of magnitude smaller than previous studies, we have been able to find new physical effects due to the random photospheric flows and thus new scalings with Lundquist number. We are pushing our simulations with even smaller η, and are expecting that another order of magnitude decrease of η would get us to another regime with new physical effects, possibly due to turbulence and turbulent reconnection.

CHAPTER 4

CURRENT SHEET STATISTICS: LOCATION AND CHARACTERIZATION OF DISSIPATIVE STRUCTURES IN CORONAL LOOP SIMULATIONS

4.1 Introduction

In Chapter 3, we made passing reference to current sheet detection and characterization when discussing details of the Lundquist number scaling of coronal heating rate and perpendicular field production in three dimensional numerical simulations of coronal heating. More specifically, the average number of current sheets appearing as the model evolves in statistical steady state ($N \sim 7$) was invoked in support of scaling analysis describing the transition in scaling behavior of the coronal heating rate and perpendicular magnetic field production (see the discussion immediately following Eq. 3.7).

In this short chapter we describe in some detail the method used to arrive at such a value. Devoting an entire chapter in explication of one estimate, which seemingly may simply be done by eye (this approach is what was in fact employed by Longcope 1993), may seem unnecessarily laborious. As will become evident however, this estimate actually results from a broader effort to fully characterize the dissipative structures in Parker's model. In the analysis supporting the coronal heating simulations, Sweet-Parker reconnection was invoked without explicit justification. Longcope & Sudan (1994) devised a robust measure of current sheet widths that began with the observation that the root-mean square of the perpendicular Fourier (x-y plane) modes of current density varied exponentially with corresponding wave vectors. Exponential fits yielded coefficients to the exponent giving a correlation length for the current density, which was taken as a measure of current sheet thickness δ. Together with the observation that current sheet lengths did not vary substantially in the Lundquist number range they examined, these measurements were shown to be consistent with Sweet-Parker scaling (c.f. Eq. 3.2).

In the present analysis, because we extend our simulations of coronal loops one order of magnitude beyond the Lundquist numbers they examined, it would prove beneficial to apply a method to characterize current sheets for their lengths and widths. The task is formidable for the following reasons: (1) Given the stochastic nature of the imposed photospheric boundary driving, current sheet orientations are random. (2) We are dealing with tens of thousands of individual instances of current sheets forming during steady state evolution of the Parker model, for which we have data cubes saved at a prescribed cadence. (3) We use periodic boundary conditions in which current sheets often traverse the edges. (4) In three dimensions, current sheets appear to branch out, so a structure appearing as a single current layer in one specific cross-section of the loop might appear as several in a different cross-section at a location further along the loop, possibly with different characteristics. Figure 3-2 attests to each of these issues.

In Section 4.2, we describe an ad-hoc algorithm for current sheet detection robust to periodic boundary conditions, together with a procedure to measure sheet parameters in two dimensions. In Section 4.3, we present scaling analysis in support of the theory developed in Chapter 3. In sections 4.4 and 4.5, we briefly introduce the extension of the analysis to three dimensions and future prospects for investigating this aspect of the Parker model.

4.2 Current Sheet Detection and Fitting: Method and Caveats

For the present analysis, we take a particularly straightforward approach to current sheet characterization, which consists of two steps. First, an ad-hoc thresholding algorithm identifies current sheet candidates by simply taking all pixels in $|J|$ above a predefined fraction of $|J|_{max}$ and testing for contiguity of the selected regions. This is done in two-dimensional cross-sections of the loop simulations, and the algorithm is robust to the periodic boundary conditions required by the pseudo-spectral RMHD scheme. By this we mean that a current sheet that appears at a border of the simulation box will appear at the other border (or at up to 4 edges if it appears at a corner), but will be identified by the algorithm as only one occurrence. This feature is crucial, considering that we are automating this procedure to analyze tens of thousands of simulation sub-runs and the likelihood of current sheets ap-

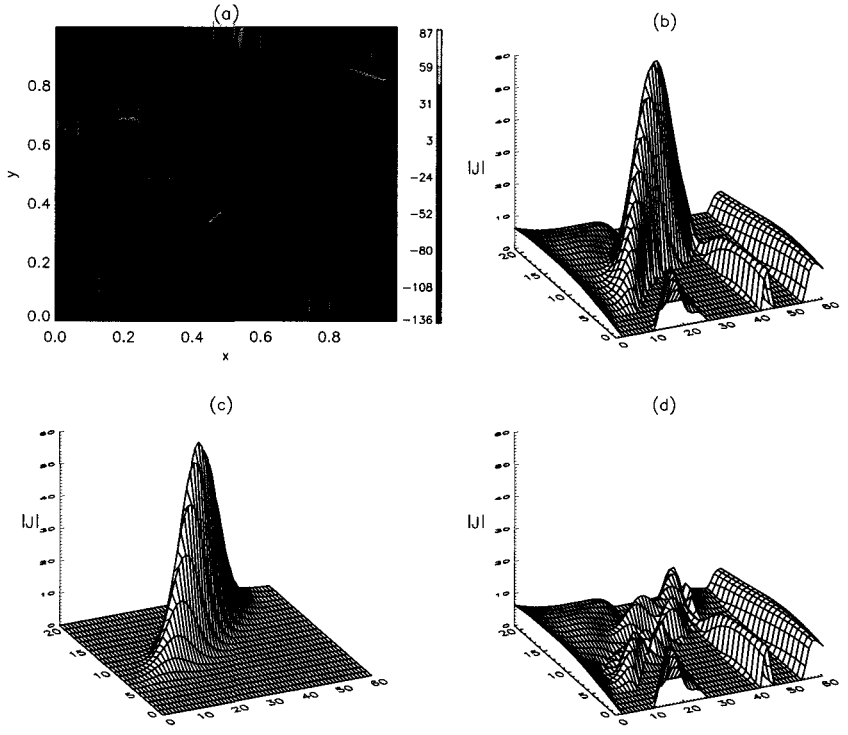

Figure 4-1 Plot (a) shows a contour plot of current density calculated at $T/\tau_A = 1932.82$ at $z = 5$ for R4 (see Table 3.1 for parameters of this run). Bracketed green numbers mark current sheets identified by the thresholding algorithm described in Appendix A. After identification, current sheets are fit with bi-variate Gaussian. The three other plots show one such fit for the current sheet labeled [9]. A surface plot shows $|J|$ in the region where current sheet [9] resides in plot (b). Plot(c) shows the bi-variate Gaussian fit and (d) shows the residual.

pearing at domain edges is quite high. Figure 4-1(a) shows contour plots of current density for one time-slice of run R4 (see Table 3.1). Current sheet candidates identified by the routine are labeled by green bracketed numbers. Sheets labeled [1] and [6], for example, appear at edges but are uniquely identified. The algorithm is described in detail in Appendix A where an implementation in Interactive Data Language (IDL) is included.

After current sheet candidates are identified they are morphologically examined by another automated algorithm, which performs least-square fitting with a bi-variate Gaussian. The automated algorithm is implemented using fitting and parameter constraining tools found in the Package for the Interactive Analysis of Line Emission (PINTofALE, Kashyap & Drake 2000). The resulting IDL software chain is similar in spirit to those used to generate rasterized images from coronal imaging spectrographs, such as the Coronal Diagnostic

Spectrometer (CDS) aboard the Solar and Heliospheric Observatory (SOHO). Together, these two algorithms yield current sheet orientations with respect to the axes (θ), number of current sheets present (N), local J_{max}, together with σ_{small} and σ_{large}, which serve as proxies for current sheet width (λ) and current sheet length (Δ), respectively. The reader is referred to Figure 4-1 for an appreciation of the quality of this process. The primary shortfalls of this approach can be summarized as follows:

(a) Many current sheets are not well approximated by bi-variate Gaussians. Profiles are often asymmetric, and the 2-D support of the current sheet structures is often bow-shaped rather than linear. In Figure 4-2, we report weighted average quantities, where we use goodness-of-fit as the weighting factor. By construction, these averages will be biased towards current sheets particularly well fit by Gaussians. The justification to use Gaussians can only be given as a subjective observation that most can be qualitatively viewed as such.

(b) Because we are taking only discrete samples in time (full data cubes are saved at a predetermined cadence during simulations runs), the measurements will be biased towards current sheet structure that is most long-lived during the lifetime of the sheets. It might be argued that this is actually a feature of the algorithm, but because we seek to ultimately characterize the dissipation scaling of the model, we should not just assume most dissipation occurs during any one phase of current sheet lifetime (See e.g., Wang et al. 1996).

(c) The identification algorithm itself, although robust for its purpose, admits a certain subjectiveness as the threshold for identification is set by the investigator. In the current analysis, the threshold is set at 10% of maximum $|J|$ (as measured in each time step). Additionally, false positives are a serious issue here. The only quality control measure we can impose is the subsequent fitting, whereby we are able to select structures whose shapes (aspect ratios) are not current sheet-like. Again, some measure of subjectiveness is imposed here.

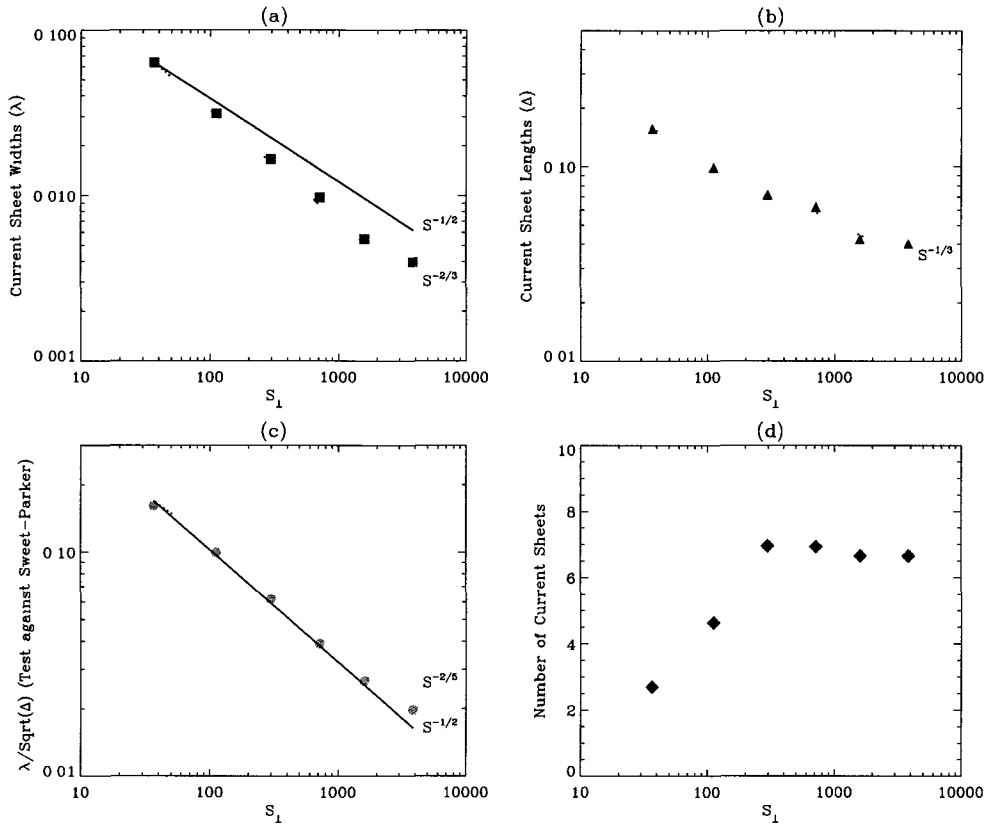

Figure 4-2 Plots (a) and (b) show weighted average values for current sheet widths and lengths from runs R2, R6, R8, R10, R11, and R12 against $S_\perp$. Plot (c) demonstrates good agreement with Sweet-Parker scaling (note that $S_\perp \equiv B_\perp/\eta$). Plot (d) shows the number of current sheets averaged over all post-processed time slices. Also note that all lines drawn are fiducial lines only, not least squares fits.

4.3 Current Sheet Statistics: Lundquist Number Scaling

With caveats of our current sheet detection and fitting method stated in the previous section, we can proceed to describe some preliminary results, keeping in mind that there is much room for improvement in our algorithms. The automated routines were applied to runs R2, R6, R8, R10, R11, and R12 for a subset of available time samples, and for the mid-z level cross-section in the simulation domain. The total sample size (N_{tot}) of current sheets measured for each of these runs is listed in the last column of Table 3.1.

We compare them first to the scaling analysis of Longcope & Sudan (1994). They are summarized in Figure 4-2. In plots (a) and (b) weighted average widths and lengths are shown as a function of $S_\perp$ where we recover scalings of $\lambda \propto S_\perp^{-2/3}$ and $\Delta \propto S_\perp^{-1/3}$. Comparing to the results of Longcope & Sudan (1994) we are well below the $\lambda \propto S_\perp^{-1/2}$

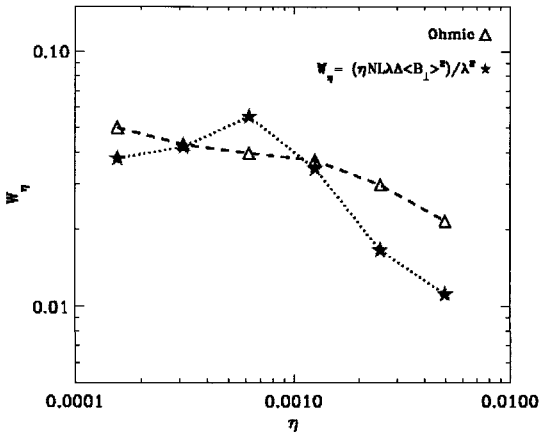

Figure 4-3 This plot compares the resistive dissipation computed using measured $< B_\eta >$ (shown in Figure 3-3(b)), λ (figure 4-2(a)), Δ (figure 4-2(b)), and N (figure 4-2(d)) to ohmic dissipation shown in Figure 3-3(a) for the cases where current sheet fitting was performed.

they find, and clearly, Δ is sensitive to η, contrary to the observations they made in the range they were able to resolve. Even with these disparities, we do still see a Sweet-Parker scaling as evident in plot (c). It is rather encouraging that in spite of the shortcomings of the analysis algorithms stated above, we do recover the well established classical Sweet-Parker scaling. However, when we impose a threshold on the current sheet aspect ratios of $\Delta/\lambda > 4$ (in the interest of quality control), we see $\lambda/\sqrt{\Delta} \propto S_\perp^{-2/5}$. Sample sizes after this threshold are included as the last column in Table 3.1. In Figure 4-2 we see the average number of sheets saturating at about $N = 7$ for at $S_\perp > 300$ showing a stronger dependence below that.

So aside from being able to recover the Sweet-Parker result averaging over all samples, the scalings of current sheet parameters we measure on the whole disagree with those observed by Longcope & Sudan (1994). As a final sanity check for these measurements, we compute $< W_\eta >= \eta \int J^2 d^3x$ by approximating $J_{max} \simeq B_\perp/\delta$ and computing $W_\eta \simeq \eta N L \Delta\lambda (B_\perp/\delta)^2$. Figure 4-3 compares this calculation to the time series average $< W_\eta >$ where we see broad agreement at the low η end and a poor estimate for the two largest η.

4.4 Current Sheet Parameters in the Parallel Dimension

The analysis has also been extended to include all z cross-sections to examine substructure along the parallel dimension. The same algorithm performed for the mid-z plane as in the previous section was applied to each cross-section along the entire domain. Figure 4-4 shows current sheet widths (left panel) and lengths (right panel) as a function of position along z for runs (c.f. Table 3.1) R1, R6, R8, R10, R11, R12, and R13 (bottom to top). From the left panel, it is clear that significant dependence on the parallel dimension does exist, and that the character of this dependence shifts according to Lundquist number. At the lower Lundquist number end (in green hues), the current sheet widths are generally thinner towards the line-tied ends. This behavior changes as one goes to larger Lundquist numbers (bluer hues), where the opposite becomes true. Current sheets become thinner towards the mid-z plane. Current sheet length measurements (right panel) show a similar behavior at low Lundquist numbers, however, z dependence is much less pronounced or absent altogether at the other end. Particularly striking is that the transition in the character of the current sheets in the parallel dimensions appear to change in the same regime where the transitions in heating rate and perpendicular magnetic field production were observed in Chapter 3 (c.f. Figure 3-3). This hints at the possibility that these qualitative observations might be understood with similar analytical arguments, where the random walk nature of boundary driving plays a pivotal role.

4.5 Current Status and Future Work

It is difficult to provide any conclusions with just the preliminary observations we have presented in this chapter. We will postpone any more discussion until more data can be incorporated into the statistics. These are results from only a subset of the available time series data for a subset of the runs shown in Table 3.1. We are lacking in good statistics particularly at the low η regime. It is rather worrisome that the lowest η (largest $S_\perp$) case seems to depart slightly from the otherwise rather clean $\lambda \propto S_\perp^{-2/3}$ scaling we showed in Figure 4-1(a) (see also Figure 4-1(c)). Post processing of the full data set is currently underway, and an interpretation of the substructure found in the parallel dimension is

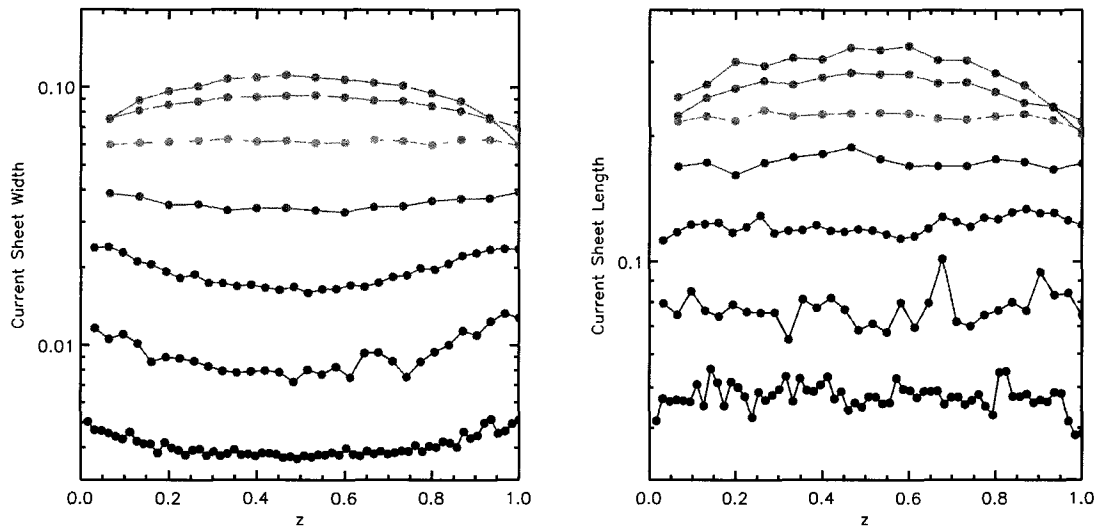

Figure 4-4 The left panel shows average current sheet widths as a function of position along z for runs R1, R6, R8, R10, R11, R12, and R13 (bottom to top). The right panel shows average current sheet lengths as a function of position along z for the same runs (bottom to top).

being developed. It is also worth noting here that the topic of systematic characterization of dissipative structures in MHD has received quite a lot of attention in recent years both in two dimensions (Servidio et al. 2009; Zhdankin et al. 2010) and three dimensions (Yoshimatsu et al. 2009; Uritsky et al. 2010) mostly in the context of MHD turbulence. In carrying out our current analysis, we are mindful of these recent results, from which we may attempt to draw further insights into the Parker model.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Future Work

5.1.1 Self-Consistent Turbulent Magnetic Reconnection in Corona Loops

In closing Chapter3, we mentioned several recent studies based on numerical simulations that suggest a fast magnetic reconnection rate is obtainable in the presence of MHD turbulence (e.g. Smith et al. 2004; Fan et al. 2004; Kowal et al. 2009; Loureiro et al. 2009). However, invariably in all these these studies, turbulence is imposed externally rather than from self-consistent evolution of a prescribed magnetic configuration. While such artificial generation of turbulence allows for the evaluation of its effects on magnetic reconnection with precise control, whether such situations are realizable in physical systems is still an open question. It is therefore important to consider the problem of turbulent reconnection together with the problem of MHD turbulence excitation within a common configuration. In an XSEDE (Extreme Science and Engineering Discovery Environment, successor to TeraGrid) allocation request for time on NCSA/Forge, we have proposed simulations which will use RMCT-CUDA to study a configuration of coronal loops within the framework of Parker's coronal heating model where turbulent reconnection can appear self-consistently.

The proposed configuration is essentially a three-dimensional generalization of the well studied 2D magnetic island coalescence instability problem (see e.g. Knoll & Chacón 2006 and references therein). Coalescing magnetic islands can be regarded as a cross section of coronal loops, such as those obtained in the Parker model from random photospheric motion. If the magnetic field energy injected from footpoint motions is larger than a certain limit, such configurations in 3D will become unstable Longcope & Sudan (1994). Magnetic islands (or flux-tubes in three dimensions) with the same sign of current will attract as shown in Figure 5-1. This is a 2D simulation using a resolution of 256^2 with $\eta = \nu = 1 \times 10^{-3}$

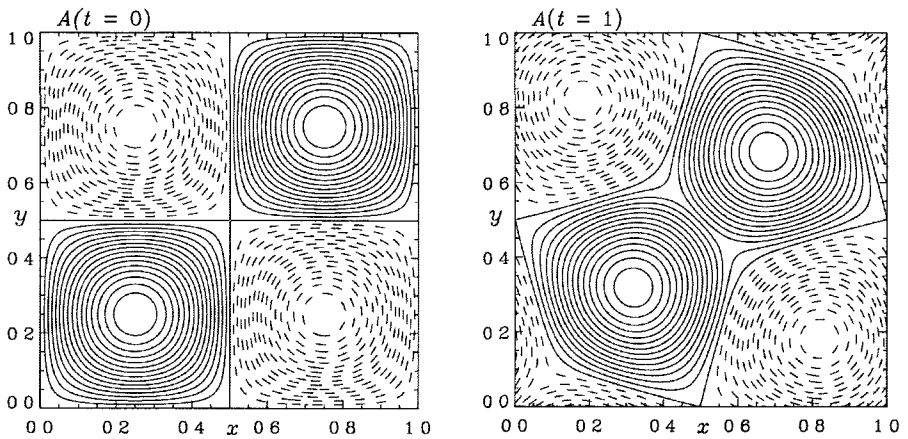

Figure 5-1 2D run at 256^2 with $\eta = \nu = 1 \times 10^{-3}$. The left panel shows contour plots of the initial flux function A. Non-negative contours are solid and negative contours broken. Contour levels range from -0.4 to 0.4 with increments of 0.025. The right panels shows contours at a subsequent time t=1 where the coalescence of the two islands has begun.

($S = 10^3$). Normalizations are chosen so that magnetic field strength, length scale, and Alfvén speed are all of the order unity. Due to the coalescence instability, the two island have begun to merge with each other at t=1. At this relatively small Lundquist number, the reconnection rate is large enough to reconnect the flux from the incoming coalescing islands at a current sheet which forms at the center of the simulation domain. At larger Lundquist numbers however, relatively slow reconnection rates are not able to reconnect the incoming flux fast enough, causing a flux-pileup, and subsequent back reaction. At Lundquist numbers around 10^4 or higher, the two islands will bounce back and forth several times as reconnection completes. Figure 5-2 shows such a case with $S = 5 \times 10^4$ ($\eta = \nu = 2 \times 10^{-5}$) at a resolution of 4096^2. The O-points are observed to slosh with a period of the order of the Alfvén time scale, and the corresponding O-point velocities are also a fraction (up to 0.3) of the Alfvén speed.

If such large scale motions with fast oscillations were to happen in the cross-sections of coronal loops, it is very possible that Alfvén waves are excited , considering the field lines near the photospheric boundaries do not move as fast due to line-tying. This launching of large amplitude Alfvén waves will result in partial reflection at the photospheric boundaries, thus leading to a situation where counter-propagating waves interact. This kind of

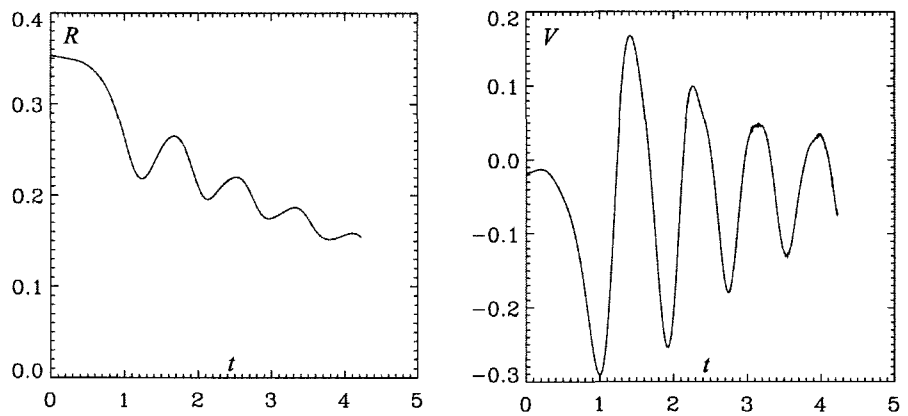

Figure 5-2 2D run at 4096^2 with $\eta = \nu = 2 \times 10^{-5}$. The left panel shows Distance R between O-point of a merging island to the center as a function of time for the case with S=5×10^4. The right panel shows the velocity of the merging island.

interaction is exactly the basic process responsible for cascading energy down to smaller spacial scales, especially perpendicular to the large-scale magnetic field, and the generation of MHD turbulence (e.g. Ng & Bhattacharjee 1996, 1997).

This possible mechanism for producing MHD turbulence in line-tied flux tubes can then provide a self-consistent way to investigate if the presence of turbulence can increase reconnection rates. If so, this possibility indicates a self-consistent, self-regulated process, in which the slow Sweet-Parker reconnection rate allows the build-up and storage of magnetic energy, so that large-scale motions between flux tubes proceed to generate waves and turbulence. This in turn induces fast turbulent reconnection, which impulsively releases the stored energy. This self-consistent mechanism, if indeed shown to work well, can potentially contribute to different energetic processes, such as solar flares, coronal mass ejections, and coronal heating due to nanoflares.

What is crucial here then, is the ability, to simulate a 3D line-tied coronal loop at sufficiently large Lundquist number, at sufficiently high resolution, and for long enough in order to correctly resolve such a self-regulated turbulent reconnection process. This is precisely the capability afforded by the GPU accelerated RMCT code. Figure 5-3 shows a preliminary 3D flux tube oscillation run performed on NICS/KIDS using $\eta = \nu = 1 \times 10^{-4}$ at $1024^2 \times 128$. O-point locations are plotted for several slices at different positions along the

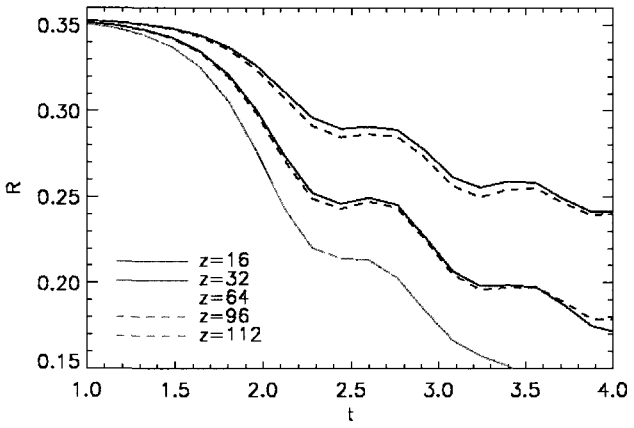

Figure 5-3 O-point locations for different positions along z in a line-tied flux tube coalescence simulation at high Lundquist number ($\eta = \nu = 1 \times 10^{-4}$ at $1024^2 \times 128$).

loop. The onset of sloshing is clearly evident. Isosurface plots of current density threading two dimensional cross-sections of flux function for two times in this simulation are shown in Figure 5-4.

For the XSEDE allocation, the proposed production runs consist of two types. We proposed 3 runs for the flux tube coalescence problem at $2048^2 \times 256$. These will be run on a range of Lundquist numbers spanning the transition from the quasi-laminar case to the expected violently sloshing regime. A set of 10 runs were proposed at $1024^2 \times 128$, each using as an initial condition a data-cube obtained from R0 (c.f. Table 3.1). It is not uncommon for Parker's model to be dominated by several large flux tubes resembling the 'artificial' setup in the standard coalescence problem. Restarting at sub-runs preceding large dissipation events and writing data cubes out at higher cadence, we will search for evidence of sloshing and a possible transition to turbulence.

5.1.2 Flare Frequency Distributions and Self-Organized Criticality

We have not discussed in this dissertation any attempt at comparisons of our simulations of coronal loops to solar observations. Parker's nanoflare heating concept was originally motivated by the balloon borne observations of Lin et al. (1984), in which hard X-ray (> 12 keV) bursts were observed at energies much smaller (10^{27} ergs) than the largest solar flares (10^{33} ergs). Lin et al. (1984) suggested that averaged over time these microflares could sig-

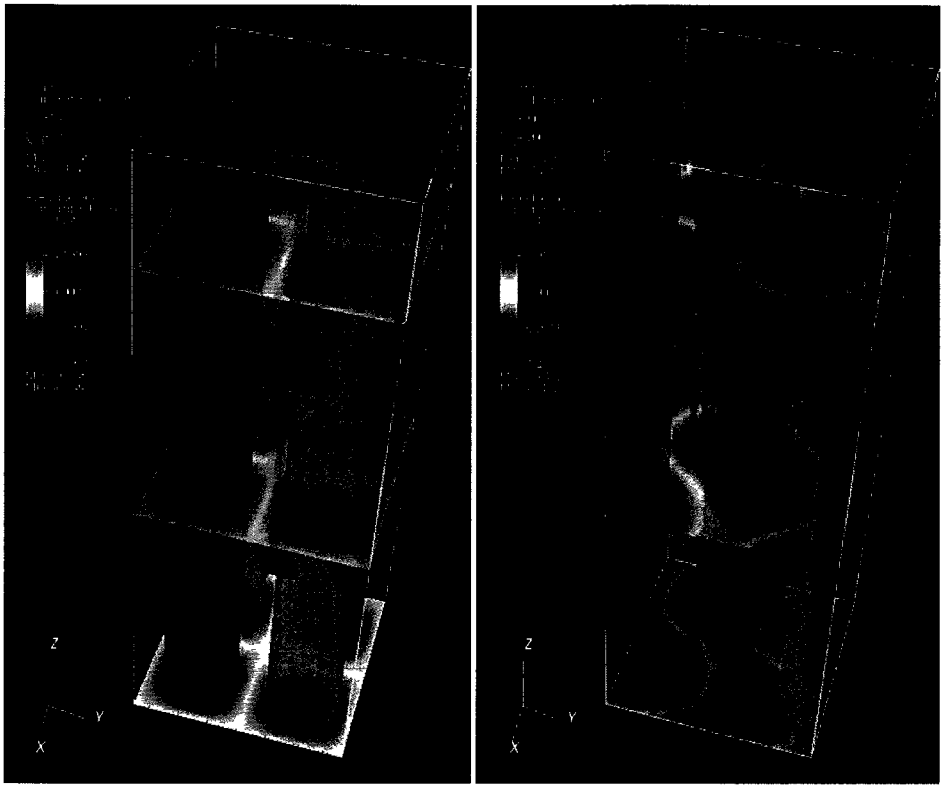

Figure 5-4 Iso-surface plots of current density threading 2D cross-sections of flux function for two times in a line-tied flux tube coalescence simulation ($\eta = \nu = 1 \times 10^{-4}$ at $1024^2 \times 128$). The left panel corresponds to $t = 0$ while the right panel corresponds to $t = 3.4$.

nificantly contribute to coronal heating. Parker (1988) envisioned that flares smaller still, would exist beyond their instrumental cutoff and that flares and microflares were made of ensembles of nanoflares. He also suggested that swarms of these small impulsive events occurred as a result of reconnection, driven by photospheric footpoint motions, complementing the idea of topological dissipation (Parker, 1972). The viability of this mechanism of coronal heating crucially relies on the occurrence rate of such small scale nanoflares. Flare frequency distributions (normally of peak flux or integrated flux) are generally found to behave as power laws, and have been used as a standard gauge by which to assess the nanoflare theory. The number of flares N is distributed as $dN/dE = aE^{-\alpha}$ where E is the flare energy with normalization a and power law index α. Hudson (1991) identified that $\alpha \leq 2$ would indicate flare distributions in which the largest flare events would dominate the total energy release. Larger indices ($\alpha > 2$) would indicate that small scale events

would dominate instead and that microflares and nanoflares play a significant role in coronal heating.

Establishing the power laws governing the low energy flare regime has been a long standing goal in observational solar physics. Flare frequency distributions of solar observations ranging from EUV to Hard X-rays have been widely reported in the literature for a variety of solar observatories and report a range of power law indices on both sides of the $\alpha = 2$ threshold (e.g. Lin et al. 1984 [balloon $\alpha = 2$)]; Dennis 1985 [OSO V $\alpha = 1.9$]; Biesecker 1994 [CGRO-BATSE $\alpha = 1.7$]; Shimizu 1995 [YOHKOH $\alpha = 1.5 - 1.6$]; Benz & Krucker 1998 [SOHO-EIT $\alpha = 2.3 - 2.6$]; Parnell & Jupp 2000 [TRACE $\alpha = 2.0 - 2.6$]; Veronig et al. 2002 [GOES-XRS $\alpha = 2.03 \pm 0.09$]; Christe et al. 2008 [RHESSI $\alpha = 1.5 - 1.58$]) and have also been measured for stellar corona (e.g. Kashyap et al. 2002 [EUVE $\alpha > 2$]). A host of issues make the unambiguous determination of a scaling law difficult. These include differences in indices for quiet sun and active region, temperature biases, selection effects, variations with solar cycle and variations of scaling with energy. The reader is referred to Hannah et al. (2011) for a recent comprehensive review (See also Biesecker 1994 for an extensive compilation). This review concludes that observational evidence as a whole currently does not support significant heating by small scale events and that higher fidelity observations and more sophisticated modeling efforts are required for a definitive verdict.

What can numerical simulations such as those we have undertaken in this dissertation contribute to our understanding of flare distributions? Identifying energy release events in time series of magnetic energy dissipation in our MHD simulations of coronal loops (e.g. Figure 3-1) is not unlike the analysis performed for real observational data in the studies just mentioned, and in fact suffer from similar selection effects and biases (Buchlin et al., 2005). Such an exercise has been carried out by Dmitruk & Gomez (1997) who find from a 2D externally driven RMHD model of coronal heating that energy release events take a power law form with $\alpha = 1.5$. Galsgaard & Nordlund (1996) also constructed power law distributions in Joule dissipation for fixed times in 3D MHD coronal simulations with footpoint driving finding $\alpha = 1.55 - 1.75$. Many investigators significantly simplify simulations by adopting models based on cellular automata. Rather than relying on computationally

expensive direct numerical MHD simulations, such models construct discretized systems governed by physically motivated binary occurrence criteria evaluated according to nearest-neighbor states to exploit the hypothesized self-similar behavior of constitutive events (see e.g. Krasnoselskikh et al. 2002; Podladchikova & Lefebvre 2006; Morales & Charbonneau 2010). These studies generally follow Lu & Hamilton (1991), who first cast the problem in terms of self-organized criticality (SOC) (Bak et al., 1987). SOC systems consist of scale-invariant ensembles of minimally stable states. Such systems self-organize into critical states in which small purtubations of the system will lead to relaxation cascades with power law character. The relaxation events can be as small as a single minimally stable state and up to cascades the size of the entire system. A classic example of SOC is a sand pile which is built by consecutive depositions of sand grains. After sufficient sand accumulates, the system settles into a pile with characteristic slope. Any further deviations from the slope will be relaxed by avalanches whose sizes are distributed as power laws. In the context of flaring plasma in the solar corona, magnetic flux progressively injected by photospheric motion plays the role of sand grains, and flare ensembles with power law character take the role of avalanches.

In Chapter 4 we described the development of algorithms for the detection and characterization of dissipative structures in our coronal loop simulations. It is then possible that a detailed analysis of current sheet statistics may complement an analysis of the energy dissipation event distribution like the one carried out by Dmitruk & Gomez (1997). The new tools we have developed can for example identify individual events that may constitute an apparently larger monolithic event when observed in time series of ohmic dissipation. We see for example in the left panel of Figure 3-2, three dominant dissipative structures (two yellow iso-surfaces and one red one), which would appear as one strong dissipation event in a time series plot. A study of this kind would of course be initially limited by the cadence at which we output full data cubes on which we can subsequently perform the feature recognition and modeling exercise. However, as we mention in the preceding section, we can restart our simulations with any one full cube of data, thus allowing for higher temporal resolution to follow the full lifetimes of dissipative structures.

Such an analysis could also be synergistic with a search for signatures of self-organized criticality in our coronal loop numerical experiments. Few studies of SOC in MHD direct numerical simulations exist. One of these is a recent paper by Uritsky et al. (2010) who find initial evidence for SOC in 3D direct numerical simulations of MHD turbulence. As they discuss, having a means to characterize both the spatial size and energy of dissipation structures, as well as individual event life-times is a crucial first step towards assessing the presence of SOC. This is exactly the kind of capability afforded by the current sheet analysis tools we have developed. Extending our algorithms to be aware of spatial extent and connectivity in the parallel dimension, and to track dissipative structures temporally might merit further development effort.

5.2 Conclusions

This dissertation has examined three distinct but related aspects of the Parker model of coronal heating. In Chapter 2, we described a port of a reduced MHD code tailored for the simulation of Parker's model for hardware acceleration using GPUs with NVidia CUDA. The reprogrammed code is now in production on a dedicated GPU workstation at the University of Alaska Fairbanks, on Forge at NCSA, and on the Keeneland Initial Delivery System at NICS. For the highest resolution considered,($2048^2 \times 256$) the GPU workstation is able to match the performance of 256 CPU cores on NERSC/Carver with 4 C2050 NVidia GPUs. When scaling up to 264 GPUs on Keeneland, we effectively have a 30 fold speedup beyond what was previously possible. These performance results do not represent a full assessment of what is possible on either CPU or GPU architectures, but rather compare two currently deployable codes, which are both in production in support of our computational study. Finer resolutions might be possible on larger machines, such as Tian-he 1A and the forthcoming ORNL/Titan. This will likely require further refinements of the code as we have described in some detail in Section 2.3. The GPU code is currently running the highest resolution cases for the three dimensional coronal heating scaling study.

In Chapter 3, we have reported our results to date for a computational campaign conducted over the course of nearly two years. With the results of this campaign, we showed

that we have recovered the scalings for heating rates and $B_\perp$ of Longcope & Sudan (1994) in the range they examined, and extending to lower η, we presented results that support a slower growth of $B_\perp$, which roughly scales as $\eta^{-1/5}$, and a heating rate that becomes insensitive to η. We also demonstrated by simple scaling analysis that the transition between these scaling behaviors results from the diminishing effects of random photospheric motion as the energy dissipation time-scale τ_E becomes much smaller than the correlation time τ_c, in accordance with Ng & Bhattacharjee (2008). As described in the preceding section, the data sets acquired from this scaling study may be used as a starting point from which to conduct targeted numerical simulations, which would search for evidence of self-consistent turbulent reconnection. Extending the scaling study to even higher Lundquist numbers is extremely challenging even using GPUs, but might be possible with larger hardware commissions and further code refinement as just discussed.

In Chapter 4 we developed a novel approach to the problem of identification and characterization of dissipative structures in loop simulations. We applied these methods to the mid-loop cross-sections, and unambiguously recovered the classical Sweet-Parker reconnection scaling, thereby justifying its use for the analysis performed in Chapter 3. We also extended the analysis to three dimensions and identified as of yet un-explained substructure whose character varies with Lundquist number. Further analysis of these observations are currently underway.

BIBLIOGRAPHY

Amdahl, G. M. 1967, in Proceedings of the April 18-20, 1967, spring joint computer conference, AFIPS '67 (Spring), 483–485

Aschwanden, M. J. 2004, Physics of the Solar Corona. An Introduction, ed. Aschwanden, M. J. (Praxis Publishing Ltd)

Bailey, D. H. 1991, Supercomputing Review, 4, 54

Bailey, D. H. 2009, in Proceedings of the 46th Annual Design Automation Conference, DAC '09 (New York, NY, USA: ACM), 528–533

Bak, P., Tang, C., & Wiesenfeld, K. 1987, Physical Review Letters, 59, 381

Benz, A. O., & Krucker, S. 1998, Sol. Phys., 182, 349

Beresnyak, A. 2011, Physical Review Letters, 106, 075001

Bhattacharjee, A., Huang, Y., Yang, H., & Rogers, B. 2009, Physics of Plasmas, 16, 112102

Biesecker, D. A. 1994, PhD thesis, University of New Hampshire

Buchlin, E., Galtier, S., & Velli, M. 2005, A&A, 436, 355

Christe, S., Hannah, I. G., Krucker, S., McTiernan, J., & Lin, R. P. 2008, ApJ, 677, 1385

Dahlburg, R. B., Rappazzo, A. F., & Velli, M. 2009, ArXiv e-prints

Dennis, B. R. 1985, Sol. Phys., 100, 465

Dmitruk, P., & Gomez, D. O. 1997, ApJ, 484, L83+

Dmitruk, P., Gómez, D. O., & Deluca, E. E. 1998, ApJ, 505, 974

Einaudi, G., Velli, M., Politano, H., & Pouquet, A. 1996, ApJ, 457, L113+

Fan, Q.-L., Feng, X.-S., & Xiang, C.-Q. 2004, Physics of Plasmas, 11, 5605

Frigo, M., & Johnson, S. G. 2005, Proceedings of the IEEE, 93, 216, special issue on "Program Generation, Optimization, and Platform Adaptation"

Galsgaard, K., & Nordlund, Å. 1996, J. Geophys. Res., 101, 13445

Garland, M., & Kirk, D. B. 2010, Commun. ACM, 53, 58

Gomez, D. O., Dmitruk, P. A., & Milano, L. J. 2000, Sol. Phys., 195, 299

Hager, G. 2010, in 6th Erlangen International High-End-Computing Symposium, http://www10.informatik.uni-erlangen.de/Misc/EIHECS6/Hager.pdf

Hannah, I. G., Hudson, H. S., Battaglia, M., Christe, S., Kasparova, J., Krucker, S., Kundu, M. R., & Veronig, A. 2011, Space Sci. Rev., 87

Hendrix, D. L., & van Hoven, G. 1996, ApJ, 467, 887

Hendrix, D. L., van Hoven, G., Mikic, Z., & Schnack, D. D. 1996, ApJ, 470, 1192

Huang, Y.-M., Bhattacharjee, A., & Zweibel, E. G. 2010, Physics of Plasmas, 17, 055707

Hudson, H. S. 1991, Sol. Phys., 133, 357

Janse, Å. M., Low, B. C., & Parker, E. N. 2010, Physics of Plasmas, 17, 092901

Kadomtsev, B. B., & Pogutse, O. P. 1974, Soviet Journal of Experimental and Theoretical Physics, 38, 283

Kashyap, V., & Drake, J. J. 2000, Bulletin of the Astronomical Society of India, 28, 475

Kashyap, V. L., Drake, J. J., Güdel, M., & Audard, M. 2002, ApJ, 580, 1118

Keane, A. 2010, in Nvidia Blog, http://blogs.nvidia.com/2010/06/gpus-are-only-up-to-14-times-faster-than-cpus-says-intel/

Kirk, D. B., & Hwu, W.-m. W. 2010, Programming Massively Parallel Processors: A Hands-on Approach (Applications of GPU Computing Series), 1st edn. (Morgan Kaufmann)

Klimchuk, J. A. 2006, Sol. Phys., 234, 41

Knoll, D. A., & Chacón, L. 2006, Physics of Plasmas, 13, 032307

Kowal, G., Lazarian, A., Vishniac, E. T., & Otmianowska-Mazur, K. 2009, ApJ, 700, 63

Krasnoselskikh, V., Podladchikova, O., Lefebvre, B., & Vilmer, N. 2002, A&A, 382, 699

Kulpa-Dybel, K., Kowal, G., Otmianowska-Mazur, K., Lazarian, A., & Vishniac, E. 2009, ArXiv e-prints

Lazarian, A., & Vishniac, E. T. 1999, ApJ, 517, 700

Lee, J.-Y., Barnes, G., Leka, K. D., Reeves, K. K., Korreck, K. E., Golub, L., & DeLuca, E. E. 2010, ApJ, 723, 1493

Lin, L., Ng, C. S., & Bhattacharjee, A. 2011, in Numerical Modeling of Space Plasma Flows: ASTRONUM-2011, ed. N. V. Pogorelov, E. Audit, P. Colella, & G. P. Zank, Astronomical Society of the Pacific Conference Series, 255–+

Lin, R. P., Schwartz, R. A., Kane, S. R., Pelling, R. M., & Hurley, K. C. 1984, ApJ, 283, 421

Longcope, D. W. 1993, PhD thesis, Cornell Univ., Ithaca, NY.

Longcope, D. W., & Sudan, R. N. 1994, ApJ, 437, 491

Loureiro, N. F., Schekochihin, A. A., & Cowley, S. C. 2007, Physics of Plasmas, 14, 100703

Loureiro, N. F., Uzdensky, D. A., Schekochihin, A. A., Cowley, S. C., & Yousef, T. A. 2009, MNRAS, 399, L146

Low, B. C. 2010, Sol. Phys., 266, 277

Lu, E. T., & Hamilton, R. J. 1991, ApJ, 380, L89

Madduri, K., Im, E.-J., Ibrahim, K. Z., Williams, S., Ethier, S., & Oliker, L. 2011, Parallel Computing, In Press, Corrected Proof,

Mikic, Z., Schnack, D. D., & van Hoven, G. 1989, ApJ, 338, 1148

Morales, L. F., & Charbonneau, P. 2010, Nonlinear Processes in Geophysics, 17, 339

Müller, W.-C., & Grappin, R. 2005, Physical Review Letters, 95, 114502

Ng, C. S., & Bhattacharjee, A. 1996, ApJ, 465, 845

—. 1997, Physics of Plasmas, 4, 605

—. 1998, Physics of Plasmas, 5, 4028

—. 2008, ApJ, 675, 899

Ng, C. S., Lin, L., & Bhattacharjee, A. 2011a, ArXiv e-prints

Ng, C. S., Rosenberg, D., Germaschewski, K., Pouquet, A., & Bhattacharjee, A. 2008, ApJS, 177, 613

Ng, C. S., Rosenberg, D., Pouquet, A., Germaschewski, K., & Bhattacharjee, A. 2011b, in Astronomical Society of the Pacific Conference Series, Vol. 406, Numerical Modeling of Space Plasma Flows: ASTRONUM-2011, ed. N. V. Pogorelov, E. Audit, P. Colella, & G. P. Zank, 255–+

Owens, J. D., Luebke, D., Govindaraju, N., Harris, M., Krüger, J., Lefohn, A. E., & Purcell, T. 2007, Computer Graphics Forum, 26, 80

Parker, E. N. 1972, ApJ, 174, 499

—. 1979

—. 1983a, ApJ, 264, 642

—. 1983b, ApJ, 264, 635

—. 1988, ApJ, 330, 474

—. 1994, Spontaneous current sheets in magnetic fields : with applications to stellar x-rays. International Series in Astronomy and Astrophysics, Vol. 1. New York : Oxford University Press, 1994., 1

Parnell, C. E., & Jupp, P. E. 2000, ApJ, 529, 554

Perez, J. C., & Boldyrev, S. 2010, ApJ, 710, L63

Podladchikova, O., & Lefebvre, B. 2006, in IAU Symposium, Vol. 233, Solar Activity and its Magnetic Origin, ed. V. Bothmer & A. A. Hady, 481–488

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterling, W. T. 1992, Numerical Recipes in Fortran 77: The Art of Scientific Computing, 2nd edn. (Cambridge University Press)

Priest, E. R., Heyvaerts, J. F., & Title, A. M. 2002, ApJ, 576, 533

Priest, E. R., Longcope, D. W., & Heyvaerts, J. 2005, ApJ, 624, 1057

Rappazzo, A. F., Velli, M., & Einaudi, G. 2010, ApJ, 722, 65

Rappazzo, A. F., Velli, M., Einaudi, G., & Dahlburg, R. B. 2008, ApJ, 677, 1348

Reale, F. 2010, Living Reviews in Solar Physics, 7, 5

Servidio, S., Matthaeus, W. H., Shay, M. A., Cassak, P. A., & Dmitruk, P. 2009, Physical Review Letters, 102, 115003

Shende, S. S., & Malony, A. D. 2006, Int. J. High Perform. Comput. Appl., 20, 287

Shimizu, T. 1995, PASJ, 47, 251

Smith, D., Ghosh, S., Dmitruk, P., & Matthaeus, W. H. 2004, Geophys. Res. Lett., 31, L02805

Stantchev, G., Dorland, W., & Gumerov, N. 2008, J. Parallel Distrib. Comput., 68, 1339

Stantchev, G., Juba, D., Dorland, W., & Varshney, A. 2009, Computing in Science and Engg., 11, 52

Strauss, H. R. 1976, Physics of Fluids, 19, 134

Uritsky, V. M., Pouquet, A., Rosenberg, D., Mininni, P. D., & Donovan, E. F. 2010, Phys. Rev. E, 82, 056326

van Ballegooijen, A. A., Asgari-Targhi, M., Cranmer, S. R., & DeLuca, E. E. 2011, ApJ, 736, 3

Veronig, A., Temmer, M., Hanslmeier, A., Otruba, W., & Messerotti, M. 2002, A&A, 382, 1070

Wang, P., Abel, T., & Kaehler, R. 2010, New Astronomy, 15, 581

Wang, X., Ma, Z. W., & Bhattacharjee, A. 1996, Physics of Plasmas, 3, 2129

Wong, H.-C., Wong, U.-H., Feng, X., & Tang, Z. 2009, ArXiv e-prints

Yoshimatsu, K., Kondo, Y., Schneider, K., Okamoto, N., Hagiwara, H., & Farge, M. 2009, Physics of Plasmas, 16, 082306

Zhdankin, V., Uzdensky, D., Perez, J., & Boldyrev, S. 2010, APS Meeting Abstracts, 6010

Zink, B. 2011, ArXiv e-prints

APPENDIX

CURRENT SHEET IDENTIFICATION

A.1 Ad-Hoc Thresholding Algorithm

The absolute efficiency of this algorithm is questionable as it loops through invididual pixels to ascertain sorting group memberships. An extra step is included which resamples to a coarser grid to drastically reduce the number of pixels the algorithm loops through. This step reduces the required run-time by orders of magnitude and does not significantly change the resulting identifications. The resampling IDL routine used is *rebinx*() included in Package for the INTeractive Analysis of Line Emission (PINTofALE) Kashyap & Drake (2000). The thresholding algorithm can be stated in five simple steps:

1. Construct auxiliary array BB which is essentially OO tiled 9-fold in larger square. The Center square is our original square.

2. Search for all pixels in OO which are above threshold and hold in array ngO. Search for all pixels in BB which are above threshold and hold in array ngB.

3. If first iteration, identify lower leftmost sheet pixel in ngO as centroid and mark this pixel. Leftmost takes precedence.

   By "mark" I mean change its value to an arbitrary value below threshold which identifies the sheet.

   Example: For first sheet use mrk=1 and OO[ngb(some-ndx)]=mrk*(1d-23) For second sheed use mrk=2 and OO[ngb(some-ndx)]=mrk*(1d-23)

   We need to check of course that no other pixels share this value, if so, set them to oblivion: OO[offending pixels]=1d-123

   Now map this centroid pixel in OO to a centroid pixel in BB and mark.

4. Identify which pixels in BB eq mrk. Loop through these and search for all surrouding pixels which also meet threshold and compile into array NHBD. Sort NHBD for uniqeness and mark them.

5. Repeat (4) until search fails. If it does go to (6)

6. Increment mrk. Repeat (2) $- - >$ (5). When search in (2) fails end program.

A.2 IDL Impelementation

```
function thrshid, OO, nx, ny, thrsh = thrsh, verbose = verbose, $
periodic = periodic, rbn = rbn, _extra = e

;+
; function thrshid
;
; Purpose: given 2-d array OO of size [nx,ny], msheet() will find
```

```
; and enumerate all contigous structures whose value in OO are greater
; than the threshold
;
;
;-------------------------------------------------------------------------------
; Ad-hoc threshold ID algorithm (assume periodic here, if non-periodic just
; ignore all mention of BB array) :
;
;(0) Construct auxiliary array BB which is essentially OO tiled 9-fold in
;     larger square. THE CENTER SQUARE IS OUR ORIGINAL SQUARE.
;
;(1) Search for all pixels in OO which are above threshold and hold in array ngO.
;    Search for all pixels in BB which are above threshold and hold in array ngB.
;
;(2) If first iteration, identify lower leftmost sheet pixel in ngO as
;    centroid and mark this pixel. Leftmost takes precedence.
;
;    By "mark" I mean change its value to an arbitrary value below threshold
;    which identifies the sheet.
;
;    Example: For first sheet use mrk=1 and OO[ngb(some-ndx)]=mrk*(1d-23)
;             For second sheed use mrk=2 and OO[ngb(some-ndx)]=mrk*(1d-23)
;
;    We need to check of course that no other pixels share this value, if so,
;    set them to oblivion: OO[offending pixels]=1d-123
;
;    Now map this centroid pixel in OO to a centroid pixel in BB and mark
;
;(3) Identify which pixels in BB eq mrk.
;    Loop through these and search for all surrouding pixels which also meet
;    threshold and compile into array NHBD. Sort NHBD for uniqeness and mark
;    them.
;
;(4) Repeat (3) until search  fails. If it does go to (5)
;(5) Increment mrk. Repeat (1)-->(4). When search in (1) fails end program.
;
;(6)  Optionally:  Loop through mrk sheets and begin:
;   (a) compute total flux contribution
;   (b) compute percent flux contribution
;   (c) do linear fit to estimate orientation, length, width
;       Try polynomial fit later to investigate shape parameters.
;   (d) summarize all in IDL sructure
;-------------------------------------------------------------------------------
; Some details:
; keep two sets of coordinates:
;   occ= original coordinates  in [O]
;   bcc= big box coordinates in [B]
;
; schmeatically: If we represent 2-d original box by : [O]
;                            [O][O][O]
; Then big box [B] will be:  [O][O[[O]
;                            [O][O][O]
;
;   occ for bookeeping to see if all relevant pixels counted/sorted
;   bcc for the actual counting and sorting and marking.
;  convert bcc to occ coordinates like this:
;occ= [bcc(0) mod nx, bcc(1) mod ny]
;  convert occ to bcc coordinates like this:
;bcc= [ occ(0)+nx, occ(1)+ny ]
;  mirror the marking of to other panes bcc corrdinates like this:
;
;  if one marks : OO[occ(0),occ(1)]=12345d-14
;  then one should mark:
;                  BB[occ(0),occ(1)]=12345d-14
;                  BB[occ(0)+0*nx,occ(1)+ny]=12345d-13
;                  BB[occ(0)+1*nx,occ(1)+ny]=12345d-13
```

```
;                   BB[occ(0)+2*nx,occ(1)+ny]=12345d-13
;                   BB[occ(0)+nx,occ(1)+0*ny]=12345d-13
;                   BB[occ(0)+nx,occ(1)+1*ny]=12345d-13
;                   BB[occ(0)+nx,occ(1)+2*ny]=12345d-13
;                   BB[occ(0)+2*nx,occ(1)+2*ny]=12345d-13
;                   BB[occ(0)+3*nx,occ(1)+3*ny]=12345d-13
;
; inputs
;
; outputs
;
;IDL subroutines: min(), uniq(), keyword_set(), where()
;
;
;10/22/09 LiWei Lin
;04/29/11 LL add keyword rbn
;-

if not keyword_set(thrsh) then thrsh = 0.90*max(abs(OO))
; else thrsh=thrsh*max(abs(OO))
if not keyword_set(verbose) then verbose = 10

OOO = OO
onx = nx
ony = ny

if keyword_set( rbn ) then begin
   OO = rebinx( OOO, findgen(nx), rbn*findgen(nx / rbn), Xindex=0 )
   OO = rebinx( OO, findgen(ny), rbn*findgen(ny / rbn), Xindex=1 )
   nx = nx / rbn
   ny = ny / rbn
endif


if verbose gt 10 then begin
   window, 1 ,xsize = 64 * 6, ysize = 64 * 6, xpos = 1200 + 50, ypos = 580 + 50
endif

;(0) Construct auxiliary array BB which is essentially OO tiled 9-fold in
;     larger square. THE CENTER SQUARE IS OUR ORIGINAL SQUARE.
BB = [[OO, OO, OO], [OO, OO, OO], [OO, OO, OO]]
nx = float(nx)
ny = float(ny)
nbx = nx * 3.
nby = ny * 3.

;(1) Search for all pixels in OO which are above threshold and hold in array ngO.
;    Search for all pixels in BB which are above threshold and hold in array ngB.
ngO = where(abs(OO) gt thrsh)
ngB = where(abs(BB) gt thrsh)
nngO = n_elements(ngO)
mrkb = 0  ; initialize id number for pixel marking

; initialize sanity count

sane = 1.0
;print, nx*ny

if ngO(0) ge 0 then begin

   ; ensure that our pixel marker values aren't already present
   mrkb = mrkb + 1.0
   mrk = mrkb * 1e-23
   jnk = where(OO eq mrkb)
   if jnk(0) ne -1 then OO(jnk)=1d-26
   jnk = where(BB eq mrkb)
```

```
   if jnk(0) ne -1 then BB(jnk)=1d-26
   npix = 0.0
   while ng0(0) ge 0 do begin     ; relaxing instilled aversion to while loops
      sane = sane + 1.0
      ; get full array of thresholded coordinates for both 00 and BB
      xndxB = ngB mod long(nx*3)
      yndxB = ngB / (long(nx*3))

      if npix eq 0 then begin
;         print, 'npixloop'
;(2) If first iteration, identify lower leftmost sheet pixel in ng0 as centroid and mark this
;    pixel. Leftmost takes precedence.
         lx0 = min(ng0) mod long(nx)
         ly0 = min(ng0) / long(nx)
         lxB = lx0 + nx
         lyB = ly0 + ny
         00(lx0, ly0) = mrk
         ; need to mark all corresponding pixels in BB
         ; This initialized the sheet marking loop
         BB(lx0, ly0) = mrk
         BB(lx0 + nx, ly0) = mrk
         BB(lx0 + 2. * nx, ly0) = mrk

         BB(lx0, ly0 + ny) = mrk
         BB(lx0 + nx, ly0 + ny) = mrk
         BB(lx0 + 2. * nx, ly0 + ny) = mrk

         BB(lx0, ly0 + ny * 2.0) = mrk
         BB(lx0 + nx, ly0 + ny * 2.0) = mrk
         BB(lx0 + 2. * nx, ly0 + ny * 2.0) = mrk

         npix = npix + 1.0
      endif else begin ; if not the first pixel in sheet

;(3) Identify which pixels in BB eq mrk.
;    Loop through these and search for all surrouding pixels which also meet
;    threshold and compile into array lBn. Sort lBn for uniqeness and mark them.


;         (a) search for all pixels whose value eqals mrk in BB
         lBm = where(BB eq mrk)
         lBmx = lBm mod long(nbx)
         lBmy = lBm / long(nbx)
         nlBm = n_elements(lBm)
;         stop
;         (b) loop through the lBm pixels and search for surrounding pixels
;          which meet threshold criteria. pixels which meet criteria are
;           already held in xndxB and yndxB which will be updated below.
         lBn = [-1]

         for j = 0L, nlBm - 1L do begin ; loop through lBm pixels and search for bordering positives
            lBn = [lBn, where((abs(xndxB - lBmx(j)) le 1) and (abs(yndxB - lBmy(j)) le 1))]
         endfor

         if sft(0) gt 0 then begin; if there are bordering positives mark them
;(4) Repeat (3) until search  fails. If it does go to (5)
;            lBn=lBn(sft)
;            nlBn=n_elements(lBn)

            tmp = ngb(lbn(sft))
            tmp = tmp[uniq(tmp, sort(tmp))]

            xtmp = tmp mod long(nx * 3)
            ytmp = tmp / long(nx * 3)
            ntmp = n_elements(tmp)
            for j = 0, ntmp-1 do begin
```

```
                txx = xtmp(j)
                tyy = ytmp(j)

                txx = txx mod long(nx)
                tyy = tyy mod long(nx)

                OO(txx,tyy) = mrk

                BB(txx,tyy) = mrk
                BB(txx + nx, tyy) = mrk
                BB(txx + 2. * nx, tyy) = mrk

                BB(txx ,tyy + ny) = mrk
                BB(txx + nx, tyy + ny) = mrk
                BB(txx + 2 * nx, tyy + ny) = mrk

                BB(txx, tyy + 2. * ny) = mrk
                BB(txx + nx, tyy + 2. * ny) = mrk
                BB(txx + 2 * nx, tyy + 2 * ny) = mrk
             endfor
             npix = npix + n_elements(tmp)
             ngO = where(abs(OO) gt thrsh)
             ngB = where(abs(BB) gt thrsh)
             xndxB = ngB mod long(nx*3)
             yndxB = ngB / (long(nx*3))
          endif else begin
;(5) Increment mrk. Repeat (1)-->(4). When search in (1) fails end program.
             ngO = where(abs(OO) gt thrsh)
             ngB = where(abs(BB) gt thrsh)
             nngo = n_elements(ngO)
             mrkb = mrkb + 1.0
             mrk = mrkb * 1e-23
             npix = 0.0
          endelse  ; if all pixel sheets seem to be marked
       endelse                   ; if not the first pixel in sheet

       if sane gt nx*ny then begin
          print, "Error, while loop fails...", sane
          return, 0
       endif
;       print, 'sane',sane, ngO(0), nngo, mrk,npix,mrkb
   endwhile
endif else begin
   print, "Error: No pixels above threshold"
   str = create_struct('N', -1, 'Index', -1)
   return, str
endelse

N = mrkb;(mrkb-1) > 1

Ns = [-1]
for j = 0, N-1 do begin
   mrk = (j + 1) * 1e-23
   cc = where(oo eq mrk)
   ncc = n_elements(cc)
   Ns = [Ns, ncc]
endfor
Ns = Ns[1: *]
mNs = max(Ns)
tmpa1 = fltarr(mNs) - 1

tmpa2 = tmpa1
cc = where(oo eq 1e-23)
if keyword_set(rbn) then begin
   fx = rbn*rbn*floor(cc/nx)
   fy = rbn*(cc mod ny)
```

```
   cc = fx*nx+fy
endif
mrk = j * 1e-23
tmpa2[0: Ns(0) -1 ] = cc
cc = where(oo eq mrk)
ncc = n_elements(cc)
str = create_struct('N', Ns(0),'Index', tmpa2)

for j = 1, N - 1 do begin
   tmpa2 = tmpa1
   mrk = (j + 1) * 1e-23
   cc = where(oo eq mrk)
   if keyword_set(rbn) then begin
      ;cc = cc * rbn
      fx = rbn*rbn*floor(cc/nx)
      fy = rbn*(cc mod ny)
      cc = fx*nx+fy
   endif
   ncc = n_elements(cc)
   tmpa2[0: Ns(j) - 1] = cc
   tmp = create_struct('N', Ns(j), 'Index', tmpa2)
   str = [str, tmp]
endfor

OO = OOO
nx = onx
ny = ony

return, str
end
```