Abstract

Title of Dissertation: Planetesimal Evolution and the Formation of Terrestrial Planets

Zoë Malka Leinhardt, Doctor of Philosophy, 2005

Dissertation directed by: Dr. Derek C. Richardson
Department of Astronomy

To create an accurate numerical model of solar system formation it is necessary to understand how planetesimals, the planetary building blocks, evolve and grow into larger bodies. Traditionally, numerical simulations of planet formation have used extrapolations of impact experiments in the strength regime to model the effects of fragmentation in planetesimal collisions (e.g. Greenberg et al. 1978; Beaugé & Aarseth 1990; Wetherill & Stewart 1993). However, planetesimals, which are large enough to decouple from the gaseous nebula, are dominated by self-gravity not material strength (Holsapple 1994). As a result, such extrapolations may give misleading results since much more energy is needed to disperse than to disrupt a planetesimal in the gravity regime. Moreover, the effects of impact angle, spin, and impactor mass ratio are often not taken into account. In order to determine the effects of various collision parameters, I have completed several parameter-space studies of collisions between kilometer-sized planetesimals. The planetesimals are modeled as "rubble piles"—gravitational aggregates of indestructible particles bound together purely by gravity. These rubble pile planetesimals have no tensile strength.

I find that as the ratio of projectile to target mass departs from unity the impact angle has less effect on the collision outcome. At the same time, the probability of planetesimal growth increases. Conversely, for a fixed impact energy, collisions between impactors with mass ratio near unity are more dispersive than those with

mass ratio far from unity. Net accretion dominates the outcome in slow head-on collisions while net erosion dominates for fast off-axis collisions. The dependence on impact parameter is almost as important as the dependence on impact speed. Off-axis collisions can result in fast-spinning elongated remnants or contact binaries while fast collisions result in smaller fragments overall. Clumping of debris escaping from the post-collision remnant can occur, leading to the formation of smaller rubble piles. In the cases tested, less than 2% of the system mass ends up orbiting the remnant. Initial spin can reduce or enhance collision outcomes, depending on the relative orientation of the spin and orbital angular momenta. For an average mass ratio of 1:5, the accretion probability is $\sim 60\%$ over all impact parameters.

Results are presented from a dozen direct N-body simulations of terrestrial planet formation with various initial conditions. In order to increase the realism of the simulations and investigate the effect of fragmentation on protoplanetary growth, a self-consistent planetesimal collision model was developed that includes fragmentation and accretion of debris. The collision model is based on the rubble-pile planetesimal model developed and investigated in the parameter space studies summarized above. The results are compared to the best numerical simulations of planet formation in the literature (Kokubo & Ida 2002) in which no fragmentation is allowed—perfect merging is the only collision outcome. After 400,000 years of integration our results are virtually indistinguishable from those of Kokubo & Ida (2002). We find that the number and masses of protoplanets, and time required to grow a protoplanet, depends strongly on the initial conditions of the disk and is consistent with oligarchic theory. The elasticity of the collisions, which is controlled by the normal component of the coefficient of restitution, does not significantly affect planetesimal growth over a long timescale. In contrast to the suggestion by Goldreich et al. (2004), it appears that there is negligible debris remaining at the

end of oligarchic growth, where "debris" is defined to be those particles smaller than our resolution that are modeled semi-analytically.

I have also looked to the small bodies currently in our solar system to help constrain its evolution. Asteroids and comets are the closest remnants in our solar system to the original building blocks of planets. Understanding the dynamics and evolution of these objects will also place constraints on the initial conditions of planet formation models. The most can be learned from binary and multiple systems since they provide mass and density information. High-resolution simulations of binary asteroid formation produce a tremendous amount of data, making it difficult to look for binary and multiple systems. I present a new code (`companion`) that identifies bound systems of particles in $\mathcal{O}(N \log N)$ time. In comparison, brute-force binary search methods scale as $\mathcal{O}(N^2)$ while full hierarchy searches can be as expensive as $\mathcal{O}(N^3)$, making analysis highly inefficient for multiple data sets with $N > 10^3$. A simple test case is provided to illustrate the method. Timing tests demonstrating $\mathcal{O}(N \log N)$ scaling with the new code on real data are presented. The method is applied to data from asteroid satellite simulations (Durda et al. 2004) and previously unknown multi-particle configurations are noted.

Planetesimal Evolution and the Formation of Terrestrial Planets

by

Zoë Malka Leinhardt

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland at College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2005

Advisory Committee:

Dr. Derek C. Richardson, advisor
Dr. Michael A'Hearn
Dr. Phil Armitage
Dr. Daniel Lathrop
Dr. M. Coleman Miller
Dr. Eve Ostriker

Acknowledgements

I want to thank the Chair of my dissertation committee, Derek Richardson. Ours has been a long and productive collaboration, one that I am enormously grateful for and one that I hope will continue. Cole Miller has been a helpful and caring colleague over the last four years. Eve Ostriker has stood and continues to stand as a significant role model. Mike AHearn, Phil Armitage, and Daniel Lathrop brought unique and important insights to the discussion of this work.

I also want to acknowledge the immense debt that I owe to the Carleton College Physics Department especially William Titus, Cynthia Blaha, and Joel Weisberg for the superb grounding in physics, and the rich introduction to the study of science. I doubt that I would have even begun such an arduous endeavor without the wonderful summer experience at the Pennsylvania Governors School for Science.

I would also like to thank Jim Greeno, whose timely advice encouraged me to go to the Institute for Theoretical Physics. Douglas N. C. Lin,

whom I met at ITP, has continued to press me to think outside the box. Makiko Nagasawa and Eiichiro Kokubo have been wonderful and generous colleagues, offering important and valuable criticism.

Finally, I would like to thank my family for their loving support and encouragement. My mother, Gaea Leinhardt, both challenged and inspired me. My father, Sam Leinhardt, rescued me more than once. My husband, Andrew James Young, kept me calm and collected even long distance.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

A fundamental question in planetary science is how solar systems form and more specifically what initial conditions lead to the formation of planets like our own. In this thesis I pursue answers to these questions using newly developed numerical models of the nonlinear evolution of planetesimals as they collide with one another and grow into planets.

1.1 Observational Constrains

Any planet formation model must be flexible enough to produce a range of solar systems consistent with all observational evidence. Therefore, I begin with a summary of observational characteristics of known solar systems.

1.1.1 Our Solar System

The Solar System contains nine planets: four terrestrial rocky planets (Mercury, Venus, Earth, and Mars) in the inner Solar System, with semi-major axes (a) between 0.4-1.5 Astronomical Unit (AU, defined to be the average distance between the Earth and the Sun) and masses (m) between 0.06-1 Earth masses ($M_\oplus$); four gas giants (Jupiter, Saturn, Uranus, and Neptune) in the outer Solar System, with a between 5-30 AU and m between 14 and 300 $M_\oplus$; and one large Kuiper Belt Object, Pluto, that is somewhere in between an asteroid and a terrestrial planet, with $a =$ 40 AU and $m = 2 \times 10^{-2}$ $M_\oplus$. The Solar System also contains three main reservoirs of smaller objects: the Asteroid Belt between Mars and Jupiter, the Kuiper Belt between 30 and 50 AU, and the Oort Cloud, a spherical distribution of comets with semi-major axes between thousands and tens of thousands of AU. All of the planets are effectively coplanar with the equator of the Sun—all but Pluto ($i \sim 17°$) have orbital inclination $i < 7°$ with respect to the ecliptic (the path that the Sun appears to follow in the sky over the course of a year due to the Earth's motion around the Sun). Most of the planets have low eccentricities, making their orbits close to circular—Earth and Venus have eccentricities $e \sim 0.01$, and the gas giants are all $e \leq 0.05$; only Mercury and Pluto have large eccentricities ($0.20 < e < 0.25$).

The coplanarity of the planets in our Solar System is consistent with our understanding of star formation: a molecular cloud with a small amount of spin collapses while conserving angular momentum. As the protostar collapses the angular momentum of the cloud prohibits mass from simply falling radially onto the star, so an accretion disk forms. The accretion disk contains most of the angular momentum from the initial molecular cloud. The planets, asteroids, and comets in our Solar System form out of this accretion disk and thus most share similar inclinations with

each other and the equator of the Sun.

The eccentricities in our Solar System are much harder to explain. Except for Mercury, Mars, and Pluto, the eccentricities in our Solar System are exceptionally low. Mercury and Pluto are locked in resonances with much more massive objects protecting their orbits—Mercury is in a spin-orbit resonance with the Sun, Pluto is in a 3:2 mean-motion resonance with Neptune. But it is not at all obvious why the rest of the planets are on such circular orbits. Numerical simulations of terrestrial planet formation find terrestrial planets end up with eccentricities that are an order of magnitude larger than that of the Earth ($e \sim 0.1$).

1.1.2 Extrasolar Planets and Detection Techniques

Over 136 planets have been detected outside our own Solar System in the past decade. These systems are diverse and unlike our own. Most of the detected extrasolar planets fall into one or more of the following catagories: 1) *hot Jupiters*—Jupiter-mass planets within 0.1 AU of their parent star on circular orbits; 2) *warm Jupiters*—Jupiter-mass planets outside of 0.25 AU with average eccentricities of ~ 0.3; 3) *multiple systems*—two or more Jupiter mass planets; 4) *pulsar planets*—terrestrial-mass planets orbiting pulsars. Selection effects significantly bias the planets that are detectable with current techniques. The planets that have been detected thus far may just be the tip of the iceberg. The large majority of planets known to date have been discovered using the *radial velocity* doppler shift technique which detects planets by the wobble they induce in the star.

In a star-planet system the center of mass is not quite aligned with the center of the star. Both the star and the planet orbit the center of mass. From our perspective as long as the system is not face-on the star moves toward and away from us as it orbits the center of mass and as a result the light from the star is blue- and red-

shifted. The more massive the planet, the farther the center of mass is from the center of the star and the larger the amplitude of the doppler shift. The closer the planet is to the star the shorter the period of the wobble. The radial velocity technique can detect wobbles in a star down to ~ 3 m s^{-1}.

Our Solar System would be outside detectibility limits using this method. Jupiter produces a stellar wobble of 12.5 m s^{-1} over a period of 12 years which is longer than the current detectable period (~ 10 yr). The smallest planet detected using the radial velocity technique is one of Neptune mass. The largest is at least 13 Jupiter masses, just on the border between what is considered a planet and a brown dwarf. Terrestrial planets induce wobbles ≤ 0.5 m s^{-1} which is currently undetectable. In addition, at fractions of a m s^{-1} the siesmology of the star becomes a significant factor. However, terrestrial planets can be detected using *pulsar timing* which works on the same principle as the radial velocity technique but instead of detecting doppler shifts in light it is sensitive to changes in pulse period.

The pulsar timing technique was used in detecting the first planets (planets B and C around millisecond PSR B1257+12; Wolszczan & Frail 1992; Wolszczan 1994). Pulsars emit a beam of radiation with a period that is predictable to high precision. As a result, they are accurate clocks, therefore, good places to look for perturbations in the expected period due to an orbiting planet. Just as described above the emission from a pulsar wobbles if another object is also in the system causing the center of mass to be offset from the pulsar center. Because pulsars are such good clocks, terrestrial- and lunar-mass objects are detectable using this technique. Timing of PSR B1257+12 reached a precision of 10^{-6} s in time delay or a velocity precision of $\sim$ mm s^{-1} (Wolszczan 1994). Even though the precision available with the pulsar timing technique is so high only one other pulsar has a planet candidate (PSR B1620-26; Thorsett et al. 1999). The low number of planet

candidates around pulsars may be a result of the pulsar planet formation process.

PSR B1257+12 is an isolated millisecond pulsar. Planets B and C have similar orbital inclinations of 47° and 53°, suggesting that the planets formed from a disk of material that surrounded the pulsar. Generally, it is thought that millisecond pulsars are created by spinning up an old neutron star that has spun down and has a relatively weak magnetic field. The old neutron star accretes mass and angular momentum from a companion star that has overflowed its Roche Lobe. Since the magnetic field is weak there is little magnetic braking and the pulsar period becomes short (ms). In the case of PSR B1257+12 it seems that the companion was destroyed—potentially tidally disrupted. The planets could have formed out of the disk of material that was left over (for an alternate theory see Miller & Hamilton 2001). If this is the mechanism, pulsar planets should be rare since the number of isolated millisecond pulsars is so small (~ 10; Konacki & Wolszczan 2003).

However, the planet detected around PSR B1620-26 did not form this way. This pulsar is in a low metallicity globular cluster, M4 (Sigurdsson et al. 2003). It has a white dwarf companion ($m = 0.34$ $M_\odot$) with an orbital period of ~ 0.5 year. The planet has a mass of ~ 2.5 M_J and an orbital period of ~ 100 years (Thorsett et al. 1999) and is seemingly orbiting the binary system. This planet was most likely captured from a passing system (Sigurdsson et al. 2003). Although these pulsar planets are certainly different in composition from the planets in our own Solar System, the planet formation process is robust and occurs in a variety of environments.

Although the radial velocity technique has been the most successful to date, there are many other techniques that show promise for detecting terrestrial planets around main-sequence stars. There are four main techniques: 1) *Astrometry*—measures the change in position of a star as it orbits around the center of mass of the

star-planet system. There are several ground-based (ALMA, Keck Interferometer, VLTI) and satellite-based (HST, GAIA, SIM) surveys that will use astrometry. SIM will have the sensitivity to detect terrestrial planets (micro arcsecond relative stellar positions). Currently, only HST is up and running. HST has confirmed one planet detection (Pravdo et al. 2004) but no new discoveries have been made. Astrometry is technically difficult because it requires determining stars' positions to high accuracy. HST fine guidance sensors can measure relative stellar positions to 0.001 arc seconds. Such accuracy is not possible on the ground where seeing can be $\sim$ 1 arc second. Adaptive optics and speckle interferometry can improve the seeing by at least an order of magnitude but still cannot touch what can be attained with satellites; 2) *Transits*—detects the small drop in the intensity of star light as the planet crosses in front of the star. All transit surveys are currently ground based but that will change with the launch of KEPLER, which should be sensitive enough to detect terrestrial planets (2×10^{-5} relative phototmetry). The transit technique has successfully discovered one planet (Alonso et al. 2004) that was confirmed using the radial velocity technique at Keck. The advantage of this technique is that many stars can be studied at one time but the star-planet orbit orientation must be close to edge-on. Also detections need to be confirmed using another technique since the mass of the transiting objects is not observed directly; 3) *Microlensing*—detect perturbations in Einstein rings from planets orbiting the lensing star. Several searches using this technique are in progress but have yet to find any planets (OGLE III, MPS, MOA, University of St. Andrews Planet Search). Microlensing is difficult—a terrestrial planet will create a 1-2% deviation in the ring, and the chance of catching an event is small, however, the number of potential events is high; 4) *Direct detection*—observe the planet directly using photometery of reflected light, infrared observations, and/or spectroscopy. The biggest problem

with direct detection is getting enough contrast between the star and planet to detect the planet—detecting a terrestrial planet requires a sensitivity of 1 part in 10^{10}. The planet is faint and close to the star. Often the star light is blocked by a coronagraph or a nulling interferometer (MOST, SIM, TPF) but it is a tricky balance to block out as much star light as possible without blocking out the planet. Direct detection has found one planet (2M1207), a wide companion to a brown dwarf (Chauvin et al. 2004).

In the next decade the number of known extrasolar planets will increase significantly as a result of the many surveys that are currently in the pipeline. The variety of planetary systems found will undoubtedly also increase and may eventually include systems like our own. The observed extrasolar planets do not provide many constraints on planet formation, instead the diversity of planets detected emphasizes how flexible a complete model of planet formation must be.

1.2 Planet Formation

Developing a complete theory of planet formation is a challenging problem because planet formation takes a long time ($10^7 - 10^8$ years) and most of the process is observationally undetectable using current techniques. The most observable stages of planet formation are the initial conditions—a young star encircled by a dusty gaseous disk, and the end result—planets. However, observations cannot tell us which stars will produce planets and which will not. There is some evidence of a correlation between metallicity and percentage of stars with detectable planets. However, it is not clear if the higher metallicity stars had a higher metallicity before planet formation began and, as a result, more metals condensed out of their nebula, creating a more massive protoplanetary disk, or if the metallicity was increased dur-

ing the planet formation process. In addition, there is currently little observational evidence to link young stars with stars that have planets because the intermediate phases of planet formation are hard to observe directly. To connect the snapshots from observations and determine what initial conditions lead to planet formation requires a detailed theoretical model and numerical simulations to show the evolution from dust grains to planets.

There are two main theories of planet formation: *gravitational instability*, in which planets form directly from gravitational collapse in the gas disk; and the *core accretion* model, in which dust grains condense out of the gas disk, grow into planetesimals (this stage is not particularly well understood), and then grow into planets via collisions. Both models have problems and neither has been shown in a numerical simulation to create a solar system with all the elements of our own: terrestrial planets close to the Sun, giant gas planets further from the Sun, all on roughly circular orbits.

1.2.1 Gravitational Instability

A gaseous disk of a given surface density has a temperature at which it becomes unstable to gravitational collapse (conversely, a gaseous disk of a given temperature has a surface density at which it becomes unstable). A stable disk is one in which rotation and thermal pressure dominate over the self-gravity of the disk. The critical balance between gravitational, rotational, and thermal forces is characterized by the Toomre Q stability criterion,

$$Q \equiv \frac{v_s \kappa}{\pi G \Sigma} > 1 \tag{1.1}$$

where v_s is the sound speed in the gas ($v_s = \sqrt{\frac{\gamma RT}{M}}$, where γ is the adaibatic constant, R is the gas constant, T is the temperature, and M is the molecular mass of the gas), κ is the epicyclic frequency equal to the mean angular speed in a Keplerian disk,

G is the gravitational constant, and Σ is the surface density of the disk (Binney & Tremaine 1987). In a protoplanetary disk there are constraints on the temperature and mass of the disk. The disk must be massive enough to create all the planets and other small bodies in our Solar System (assuming that the disk was all at solar metallicity, the minimum-mass solar nebula is 0.01-0.02 $M_{\odot}$; Weidenschilling 1977). In addition, the disk must have a temperature gradient to explain the lack of volatiles in the inner Solar System while the outer Solar System is rich in volatiles. Using a starting mass of ten times the minimum mass solar nebula and temperature profiles based on grain growth models numerical simulations suggest that gas giants may be able to form quickly (less than 1000 years) if the gaseous disk becomes gravitationally unstable (Boss 1998; Mayer et al. 2002).

The condensations or planetary embryos (protoplanets) formed in the outer regions of the disk where T $<$ 60 K. The gravitational instability model did produce massive eccentric planets, some of which migrated close to the star (similar to the extrasolar planets detected). However, these simulations used very simple equations of state for the gas disk (isothermal and adiabatic) neither of which is realistic. Other simulations using more realistic cooling models have also found that gravitationally unstable disks fragment (Johnson & Gammie 2003; Rice et al. 2003). Protoplanets formed this way have solar metallicity—Jupiter and Saturn are significantly more metal rich than the Sun (Jupiter is 90% H and He by mass, Saturn is 77%, the Sun is $\sim$ 98%; Lissauer 1993). Mayer et al. (2002) argue that a core formed after the initial coagulation by accretion or absorption of metal-rich planetesimals (that formed through some other mechanism).

The temperature in the inner solar system was never cool enough to allow the disk to become gravitationally unstable, so the formation of terrestrial planets cannot be explained by gravitational instability and direct collapse to protoplanets. It

is also unlikely that Uranus and Neptune formed via gravitational instability since their composition varies significantly from solar (5-25 % H and He by mass; Lissauer 1993; Lodders & Fegley 1998) unless they lost a significant amount of their hydrogen-rich atmosphere and the present-day Uranus and Neptune are the cores of the original planets. This would require that both Uranus and Neptune were significantly larger than Jupiter initially and an OB star photoevaporated their atmospheres with extreme ultraviolet radiation Boss et al. (2002).

1.2.2 Core Accretion Model

In contrast to the gravitational instability model, the core accretion model assumes that dust grains grow into planetesimals and planetesimals grow into planets by accretion-dominated collisions. This model is much slower than gravitational instability, requiring $\sim 10^6$ years to form protoplanets and up to 10^8 years to form a full-fledged solar system. However, the core accretion model can explain the formation of both terrestrial and gas giant planets. The process begins when refractory elements with high sublimation temperatures condense from the cooling protoplanetary nebula and form metal-dominated dust grains. The dust grows into *planetesimals*, objects that are large enough ($\sim$ 1 to 10 km for a planetesimal bulk density of 2 g cm^{-3} and a gas mass density of 10^{-9} g cm^{-3}; Lissauer 1993) that their dynamics are dominated by the tidal field of the Sun and gravitational interactions with each other, as opposed to turbulence and drag forces from the gas that dominate the dynamics of dust grains.

The mechanism for *planetesimal formation* is an open question—there are three main models: 1) *pair-wise accretion*—dust grains collide with one another and "stick". In theory the dust grains grow slowly from μm to km sizes while embedded in a gaseous primordial disk. However, experiments of colliding dust par-

ticles have failed to get them to "stick" to one another at the speeds predicted to occur in the solar nebula (> 1 m s^{-1}; Blum & Wurm 2000; Blum & Muench 1993; Weidenschilling & Cuzzi 1993). In addition, meter-sized objects should migrate quickly into the central star as the result of efficient gas drag; 2) *gravitational instability*—planetesimals form through gravitational instability of the dust layer that has condensed in the midplane of the nebula (Goldreich & Ward 1973; Youdin & Shu 2002; Youdin & Chiang 2004). This model avoids the problems of pair-wise accretion because planetesimals form almost instantaneously. But, as is the case with gravitational instability in the gas disk, the dust disk must be cool and dense, yet, vertical shear may prevent the dust disk from settling (Weidenschilling 1995); 3) *vortices*—dust particles gather within vortices that have formed in the gaseous disk (Tanga et al. 1996). In theory vortices form from random motions in the Keplerian shear flow or instabilities (baroclinic instability) that lead to the growth of vorticity perturbations into full-fledged vorticies. If vortices do form within the protoplanetary disk they could successfully gather dust in the protected "eye" allowing the planetesimals to grow.

Despite the theoretical uncertainties there is significant observational evidence that planetesimals do form. Our Solar System has remnants of them, namely, comets and asteroids in the asteroid belt, Kuiper Belt, and Oort Cloud which formed early in the history of the Solar System.

Once planetesimals have formed they enter into a new phase of evolution. It is generally assumed that planetesimals begin on effectively circular orbits having just decoupled from the gaseous disk. Numerical and semi-analytic simulations show that planetesimals go through two phases of growth: *runaway growth*—dynamical friction, the gravitational scattering of smaller planetesimals by larger planetesimals, causes equipartition of kinetic energy between the small and large planetesimals.

This means that the velocity dispersion of the smaller planetesimals increases while the velocity dispersion of the larger planetesimals decreases. The larger planetesimals have a larger geometric cross section, significant gravitational focusing, and low eccentricity, whereas the smaller planetesimals have a small geometric cross-section, little gravitational focusing, and high eccentricities. Therefore, most collisions consist of a large planetesimal accreting a smaller planetesimal. The larger planetesimals exponentially runaway in size and separate from the background population. Once large planetesimals can significantly alter the velocity dispersion of the background population of smaller planetesimals, the growth of the larger planetesimals, now considered *protoplanets*, slows and the growth of the planetesimals enters the next phase—*oligarchic growth*. All the protoplanets grow in an orderly fashion, approaching a similar mass (Kokubo & Ida 1998, 2002). All of the protoplanets are separated by $\geq 5r_H$, where $r_H \equiv ((m_1 + m_2)/3M_*)^{1/3}a$ is the mutual Hill radius (m_1 and m_2 are the masses of two neighboring protoplanets, M_* is the mass of the central star, and a is the semi-major axis of the protoplanets). This organization evolves through "gravitational repulsion" (Kokubo & Ida 1995) where protoplanets approaching crossing orbits strongly scatter each other, increasing their eccentricies. The large eccentricities of the protoplanets are quickly damped via dynamical friction provided by the small planetesimal background. Gravitational repulsion continues gradually until all of the protoplanets are outside the gravitational influence of one another ($\sim 5r_H$). Oligarchic growth ends when there are not enough planetesimals to continue the growth process. Over the following ten million years the protoplanets grow into planets by infrequent collisions with each other.

The core accretion model also has problems and is potentially incomplete. In the outer solar system this model may be too slow to form a core massive enough to accrete gas. In the terrestrial region simulations using this model produce eccentric-

ities for the protoplanets that are an order of magnitude larger than the eccentricity of the Earth. Since this is the only model that qualitatively explains the formation of terrestrial planets, we focus on solving its shortcomings with respect to terrestrial planet formation by making the numerical implementation of the core accretion model more realistic.

1.3 Thesis Approach and Organization

In order to create an accurate numerical model of solar system formation it is necessary to understand how the planetary building blocks, namely, kilometer-sized planetesimals, evolve and grow into larger bodies. Traditionally, numerical simulations of planet formation use extrapolations of impact experiments in the strength regime to model the effects of fragmentation in planetesimal collisions (e.g. Greenberg et al. 1978; Beaugé & Aarseth 1990; Wetherill & Stewart 1993). However, planetesimals that are large enough to decouple from the gaseous nebula are dominated by self-gravity not material strength. As a result, such extrapolations may give misleading results since generally much more energy is needed to disperse than to disrupt a planetesimal in the gravity regime. Moreover, effects of impact angle, spin, and impactor mass ratio are not usually taken into account. In this thesis these issues are addressed in chapters 2-4 through a series of increasingly specific and realistic simulations. Chapter 5 takes a different tack in an attempt to put further constraints on the initial conditions of planet formation scenarios by learning more about the small bodies currently in our Solar System.

Chapter 2 focuses on the effect of impact parameter, speed, spin, and coefficient of restitution on the collision outcome. This chapter was published as Leinhardt et al. (2000). In these simulations, planetesimals are modeled as rubble piles—

gravitationally bound objects with no tensile strength. The rubble pile model was chosen based on evidence that a significant fraction of small bodies in our Solar System, asteroids and comets, may be gravitational aggregates. In addition, planetesimals in the middle stage of planet formation are large and their self-gravity far dominates over their material strength. In all of the simulations, the positions and velocities of the rubble pile particles are evolved using a direct numerical method under the constraints of gravity and physical collisions. Speeds are kept low (< 10 m s^{-1}, appropriate for dynamically cool systems such as the primordial disk during early planet formation) so that the maximum strain on the component material does not exceed the crushing strength. The purpose of this study is to begin to understand what the necessary parameters are for planetesimals to grow in a protoplanetary disk.

Chapter 3 is an expansion of the parameter space study presented in chapter 2. This chapter was published as Leinhardt & Richardson (2002). Results are presented from direct N-body simulations of collisions between gravitational aggregates of varying size over a range of impact parameter and speed as part of a study to further parameterize planetesimal growth in the Solar System. The goal is to provide a recipe for planetesimal evolution that can be used in solar system formation models. In this chapter the study of planetesimal evolution is split into two experiments. The first experiment quantifies which collisions cause planetesimal *growth* or *erosion*. Growth occurs when the largest post-collision remnant exceeds the initial mass of the target (the more massive of the initial planetesimals). Similarly, erosion occurs when the largest post-collision remnant is less massive than the initial mass of the target. Accretion/erosion probabilities are derived based on the results of these simulations. The second experiment determines the *critical dispersal energy* ($Q_D^\star$, the energy per unit mass necessary to create a post-collision remnant of 50% the

mass of the system) as a function of the mass ratio of the larger rubble pile to the smaller rubble pile. This allows a comparison of the rubble pile collision results directly with those of other groups that use different collision models and numerical methods.

The results of chapters 2 and 3 are used in chapter 4 to create a more realistic planetesimal collision model for planet formation simulations. This chapter is currently in press as Leinhardt & Richardson (2005b). In contrast to extrapolations from laboratory experiments in the strength regime the model is appropriate for the gravity regime and includes both fragmentation of planetesimals and accretion of debris onto planetesimals. A dozen simulations of terrestrial planet formation were conducted in order to investigate how initial conditions and fragmentation of planetesimals affect planetesimal evolution and planet formation. The results are compared to the best numerical simulations of planet formation in the literature (Kokubo & Ida 2002) in which no fragmentation is allowed—perfect merging is the only collision outcome. In other words in the comparison work all planetesimal collisions are assumed to result in growth—there is no loss of mass as the result of a planetesimal collision.

The planetesimal collision model used in the planet formation simulations consists of two phases. When a collision is predicted, the first-order outcome is looked up in a database of collision outcomes based on the speed, impact parameter, and mass ratio of the two colliding planetesimals. The simulations used to produce the database are similar to the simulations presented in chapters 2 & 3: two rubble piles built up of identical, indestructible spheres are collided with one another over a range of collision parameters. If the largest post-collision remnant contains most of the mass of the initial system, the mass of the largest post-collision remnant from the lookup table is used and the planet formation simulation continues as before.

If the second-largest post-collision remnant is close in mass to the largest remnant, the collision model moves into the second phase: the collision is resolved in detail by substituting rubble-pile planetesimals for the single particle planetesimals and evolving them using the same technique that was used to create the look up table within the protoplanetary disk. The purpose of this collision model is to create a realistic scenario for planetesimal evolution that allows both accretion and erosion of planetesimals but assumes neither.

In addition to working from the beginning and creating a more detailed model of planet formation in an attempt to understand how our Solar System came to be, we can also learn about the conditions in the early Solar System by looking at the current Solar System. Asteroids and comets are present-day analogs to planetesimals. Understanding the dynamics of these objects will also help constrain planet formation models. We can learn the most from binary and multiple systems of small objects since these systems provide mass and density information. There are many models of binary asteroid formation, but in the Main Belt it is thought that they form via catastrophic impacts (so-called family-forming events, where an asteroid family is a group of asteroids that have similar colors and orbital elements). High-resolution simulations of these events produce a tremendous amount of data, making it computationally difficult to look for binary and multiple systems.

To help address this problem, in chapter 5 I present `companion`, a hierarchical tree code that detects binaries, multiple, and complex hierarchical systems in the output from numerical simulations in $\mathcal{O}(N \log N)$ time. This chapter is in press as Leinhardt & Richardson (2005a). In comparison, brute force binary search methods scale as $\mathcal{O}(N^2)$ while full hierarchy searches can be as expensive as $\mathcal{O}(N^3)$, making analysis highly inefficient for multiple data sets with $N > 10^3$. The code is also used to reanalyze published data from Durda et al. (2004), highlighting newly detected

hierarchical systems.

Chapter 6 summarizes the findings of chapters 2-5 and presents future work.

Chapter 2

Direct N-body Simulations of Rubble Pile Collisions

This chapter has been published: Leinhardt, Z. M., Richardson, D. C., & Quinn, T. 2000, Icarus, 146, 133

ABSTRACT

There is increasing evidence that many kilometer-sized bodies in the Solar System are piles of rubble bound together by gravity. We present results from a project to map the parameter space of collisions between kilometer-sized spherical rubble piles. The results will assist in parameterization of collision outcomes for Solar System formation models and give insight into disruption scaling laws. We use a direct numerical method to evolve the positions and velocities of the rubble pile particles under the constraints of gravity and physical collisions. We test the dependence of the collision outcomes on impact parameter and speed, impactor spin, mass ratio, and coefficient of restitution. Speeds are kept low (< 10 m s^{-1}, appropriate for dynamically cool systems such as the primordial disk during early planet formation)

so that the maximum strain on the component material does not exceed the crushing strength, assuming sufficient granularity. We compare our results with analytic estimates and hydrocode simulations. We find that net accretion dominates the outcome in slow head-on collisions while net erosion dominates for fast off-axis collisions. The dependence on impact parameter is almost equally as important as the dependence on impact speed. Off-axis collisions can result in fast-spinning elongated remnants or contact binaries while fast collisions result in smaller fragments overall. Clumping of debris escaping from the remnant can occur, leading to the formation of smaller rubble piles. In the cases we tested, less than 2% of the system mass ends up orbiting the remnant. Initial spin can reduce or enhance collision outcomes, depending on the relative orientation of the spin and orbital angular momenta. We derive a relationship between impact speed and angle for critical dispersal of mass in the system. We find that our rubble piles are relatively easy to disperse, even at low impact speed. This may provide a way of constraining the energy dissipation parameter and related properties of the initial planetesimal population.

2.1 Introduction

There is growing interest in understanding the dynamics of collisions between small bodies in the Solar System. Typically such collisions are divided into two regimes: those dominated by material strength and those dominated by self-gravity (Holsapple 1994). The transition from the strength to the gravity regime may occur at body sizes as small as a few kilometers for basalt (Ryan & Melosh 1998; Benz & Asphaug 1999) or as small as 250 m for silicates (Love & Ahrens 1996). In this paper we present numerical results from simulations of collisions in the gravity regime. Our experiments are primarily concerned with low-speed collisions between equal-mass,

kilometer-sized rubble piles, gravitationally bound aggregates of loose material. We believe that these experiments will shed light on the collisional dynamics of the protoplanetary disk when typical encounter speeds are comparable to the surface escape speed (about 1 m s^{-1} for kilometer-sized planetesimals of 2 g cm^{-3} bulk density).

2.1.1 Definitions

We begin with definitions of terms frequently encountered in the context of binary collision experiments. Typically in the literature one impactor (the larger one) is stationary and is considered to be the target, while the other (the smaller one) is moving and is called the projectile. In our experiments, the impactors are comparable in size and are both in motion, so we generally do not distinguish between a target and a projectile. Most laboratory experiments involve solid targets that possess tensile strength, so the outcome is measured in terms of the extent of disruption or shattering of the target. A critical or catastrophic shattering event is one in which the largest post-impact fragment (the remnant) has 50% of the target mass. Following a recently adopted convention in the literature (Durda et al. 1998), we use $Q_S^\star$ to denote the kinetic energy per unit target mass to achieve critical shattering. A rubble pile, by definition, has no tensile strength, so $Q_S^\star$ is effectively zero. However, a rubble pile can still be disrupted in the sense that one or more of the component particles becomes separated from the rest for at least an instant.

For collisions in free space, fragments or particles are said to be *dispersed* if they attain positive orbital energy with respect to the remnant. Hence, a critical or catastrophic dispersal is one in which the largest remnant is left with 50% of the original target mass after the remaining material has dispersed to infinity. The energy per unit target mass to achieve this is denoted by $Q_D^\star$. In our experiments, since we do not distinguish between a target and a projectile, $Q_D^\star$ refers to the

energy per unit total mass, in the center-of-mass frame, needed to critically disperse the entire system. Finally, we define *erosion* to mean permanent removal of mass from a body, and *accretion* to mean permanent retention of mass. In the context of our experiments, net erosion means that one body (the largest if the impactors are of unequal mass) had less mass at the end of the run than it started with. Net accretion means it had more mass at the end.

2.1.2 Motivation

Many asteroid characteristics are inconsistent with monolithic configurations. Recent observations by the Near Earth Asteroid Rendezvous spacecraft of Mathilde, a 53-km C-class asteroid, are particularly suggestive. First, Mathilde's largest crater is enormous: it has a diameter of 33.4 km, almost 7 km larger than the asteroid's mean radius (Veverka et al. 1997). Numerical hydrocode simulations and laboratory experiments strongly suggest that in order for Mathilde to have survived the impact that formed such a substantial crater, the asteroid must be made of some material that does not efficiently transmit energy throughout the body (Love et al. 1993; Asphaug et al. 1998; Housen et al. 1999).

Second, Mathilde has a remarkably low density of 1.3 g cm^{-3} (Yeomans et al. 1997), about one-third the average value for the chondritic meteorites that are thought to originate from C-class asteroids (Wasson 1985). Such a low density suggests that Mathilde is highly porous. If true, the voids in the material could impede the transmission of energy from a collisional shock wave and allow a rather weak body to survive an otherwise catastrophic impact event. We also note the recent discovery of the asteroid satellite S/1998 (45) 1, which implies a density of $\sim$ 1.2 g cm^{-3} for the main body Eugenia (Merline et al. 1999).

In addition to Mathilde, the surfaces of 243 Ida, 951 Gaspra, and Phobos show

several sizable craters that have diameters on the order of the mean radius of the body (for references, see Richardson et al. 1998, hereafter Paper I). As in the case of Mathilde, the energy necessary to create craters of this size would disperse or disrupt the original body if it were solid Asphaug & Melosh (1993).

Further evidence for the prevalence of rubble piles comes from asteroid spins. In a sample of 107 asteroids smaller than 10 km in diameter, Harris (1996) found that the spin period distribution truncates at fast spin rates, where rubble piles would start to fly apart[1].

One explanation for the observed characteristics of these asteroids and their craters is that they are rubble piles. Although rubble pile configurations are more susceptible to disruption by tidal forces than monolithic configurations (Paper I), there is increasing evidence that rubble piles have a higher impact strength (Ryan et al. 1991; Love & Ahrens 1996; Asphaug et al. 1998). There are two scenarios for creating a rubble-pile asteroid: (1) the asteroid is initially one solid body of material and is rubblized over time by multiple impacts; (2) the rubble-pile configuration of the asteroid is primordial. Regardless of how rubble-pile asteroids are formed it is interesting to investigate how they interact and evolve in the Solar System. In addition to asteroids there is a considerable amount of evidence that a large percentage of comet nuclei are rubble piles, for example, the tidal disruption of Comet D/Shoemaker Levy 9 (Richardson et al. 1995; Asphaug & Benz 1996).

[1]At least one asteroid spinning faster than this limit has since been discovered (Ostro et al. 1999), but its small size ($\sim$ 30 m) puts it comfortably in the strength regime.

2.1.3 Laboratory Experiments: Strength vs Gravity

Ryan et al. (1991) presented results from a laboratory study of impacts into weak inhomogeneous targets. Due to practical limitations they used $\sim$ 0.5-cm targets of gravel and glue. As a result, their specific experimental results are firmly rooted in the strength regime. However, the most general conclusion that the group arrived at from dropping, crashing, and shooting at the gravel aggregates was that the relatively weak targets have a surprisingly high impact strength. In other words, it took a large amount of energy (at least $Q_S^\star = 40$ J kg^{-1}) to critically disrupt or shatter the target such that the largest remnant was one-half the mass of the original object. The nonuniformity of the target causes a greater fraction of the impact energy to dissipate thermally; therefore, the collisional shock wave is more efficiently absorbed by the target.

Laboratory experiments on Earth to investigate directly the collisional dynamics of the gravity regime are difficult to conduct since the target size necessary to reach this regime is impractically large. Instead, overpressure and centrifuge techniques have been used to artificially simulate the gravity regime in the laboratory. In an overpressure experiment, Housen et al. (1991) used nitrogen gas at various pressures to mimic the lithostatic stress felt inside a large target. At these pressures they were unable to carry out true impact tests, so they used a buried charge instead of a projectile. As the pressure was increased, the size of the largest remnant after each explosion also increased, indicating a transition from the strength-dominated regime to the pressure dominated regime. Housen et al. (1991) argued that the pressure regime was analogous to the gravity regime and extrapolated a scaling law for the gravity regime from the overpressure data. This laboratory study has two important drawbacks: (1) by using a buried charge the experiment does not model

the actual surface dynamics of an asteroid during an impact; (2) the gas overpressure is not an r^{-2} force law. They were able to reach a regime in the laboratory that was not dominated by the strength of the material, but it is unclear whether the gravity-regime scaling law derived from the overpressure data is valid.

In a centrifuge experiment, Housen et al. (1999) were able to conduct true impact tests by firing a small projectile (a polyethylene cylinder 0.65 cm in diameter) from a gas gun strapped to the arm of a centrifuge. They positioned a porous target (composed of quartz sand, perlite, fly ash, and water) at the end of the arm. The centrifuge was used to mimic the gravitational force at the surface of a much larger body. The use of the centrifuge introduces second-order complexities due to the Coriolis force and the field orientation in general at the surface of the cylindrical target (though this is only really a problem in the event of high ejecta trajectories). In addition, the flat surface of the target may subtly affect crater morphology. Nonetheless, this experiment showed that porous targets in the gravity regime are efficient at absorbing impact energy at the surface by compacting the underlying material.

2.1.4 Numerical Simulations of Collisions and Tidal Disruptions

Extrapolations of laboratory experiments have resulted in rough strength and gravity scaling laws. In order to truly understand the collisional dynamics and evolution of large bodies, numerical simulations are a necessity. For example, Love & Ahrens (1996) used a three-dimensional smoothed particle hydrodynamics (SPH) code to simulate high-speed catastrophic collisions. They used various impact speeds (3-7 km s^{-1}), impact angles (5-75°), target diameters (10-1000 km), and projectile diameters (0.8-460 km) in order to explore a large region of parameter space. The

big targets placed the experiments securely in the gravity regime, allowing the researchers to treat gravity carefully and neglect the strength and fracturing of the target completely. Their extrapolated scaling law for the gravity regime placed the transition from the strength to the gravity regime at a target diameter of 250 ± 150 m, much smaller than that predicted by laboratory experiments (Holsapple 1994). Love & Ahrens (1996) argue that since smaller asteroids are more common than larger ones, a given asteroid is more likely to suffer a shattering impact before a dispersing impact. Thus, it seems plausible that many asteroids in our Solar System are at least partial rubble piles.

More recent simulations have had similar results. Asphaug et al. (1998) conducted three high-speed (5 km s^{-1}) impact experiments using a solid target, a partially rubblized contact binary, and a totally rubblized target. In each case the researchers used a small projectile six orders of magnitude less massive than the target. There are three major conclusions from this study: (1) it is much easier to disrupt a solid target than it is to disperse it—this conclusion is evidence that it is possible to change a solid body into a rubble pile with impacts; (2) rubble regions can insulate and block energy from traveling through a body—in a contact binary, for example, one end could be critically disrupted while the other remains undamaged; (3) the fully rubblized targets efficiently localize the energy transmitted during a collision which in turn minimizes the damage outside the collision region and allows weak bodies to survive high-energy impacts with much less damage than solid targets. This again implies that many small bodies in the Solar System may be rubble piles. Other similar numerical experiments include Ryan & Melosh (1998) and Benz & Asphaug (1999).

Watanabe & Miyama (1992) used 3D SPH code to investigate the effects of tidal distortion and shock compression from collisional impacts in the process of

planetary accumulation. They used two equal-sized spherical bodies and assumed a perfect Newtonian fluid. It is important to note that their code did not model an incompressible fluid (their adopted polytropic indices were always greater than zero). As a result of experimenting with impact angle, speed, and density gradients, they found that tidal forces can enlarge the coalescence rate of planetesimals by almost a factor of 2. In addition, when the initial speed of the impactor is significantly lower than the escape speed of the system, less than a few percent of the total mass is lost from the system in the collision. They did not attempt any simulations with initial speeds in excess of 50% of the escape speed.

In Paper I, Richardson et al. numerically simulated the effects of Earth's tidal force on rubble-pile asteroids. Unlike Watanabe & Miyama (1992), they simulated the Earth-crossing asteroids as incompressible fluids using a hard-sphere model. They varied the asteroids speed, spin, shape, and close-approach distance. Generally, slow-moving, close-approaching, prograde-rotating, elongated asteroids were the most susceptible to tidal disruption. They found several distinct classes of outcome: in the most violent disruption cases, the asteroid was stretched into a line and recollapsed into a string of pearls reminiscent of Comet D/Shoemaker Levy 9 at Jupiter; for moderate disruptions, large pieces of the asteroid were stripped off in many cases, forming satellites or contact binaries; the mildest disruptions resulted in little mass loss but significant shape changes. These various outcomes could lead to the formation of crater chains (Bottke et al. 1997) asteroid satellites and doublet craters (Bottke & Melosh 1996b,a), and unusually shaped asteroids (Bottke et al. 1999).

Durda (1996) carried out simulations to study how readily satellites form as a result of mutual gravitational attraction after the catastrophic disruption of the progenitor. Durda (1996) came to three major conclusions: (1) satellites do form

immediately after a catastrophic collision; (2) contact binaries form more easily than true binary systems; (3) the binary systems form in a wide range of size ratios. It is important to realize that Durda (1996) assumed a power-law mass distribution for the catastrophically fragmented asteroid. The slope index used (1.833) was taken from extrapolations of laboratory experiments.

2.1.5 Implications for Planet Formation

Traditionally, numerical simulations of planet formation use extrapolations of impact experiments in the strength regime to model the effects of fragmentation in planetesimal collisions (e.g. Greenberg et al. 1978; Beaugé & Aarseth 1990; Wetherill & Stewart 1993). From what we have already seen, such extrapolations may give misleading results since generally much more energy is needed to disperse than to disrupt a planetesimal in the gravity regime. Moreover, effects of impact angle, spin, and impactor mass ratio are often not taken into account. In the case of rubble piles, no empirical model actually exists. For example, we might expect reaccumulation like that seen in the tidal disruption models to also occur after the catastrophic impact of two rubble-pile planetesimals. In this paper we aim to explore these issues by simulating collisions between rubble-pile bodies over a wide range of parameter space and determining the implications of the results for planet formation. In Section 2.2 we describe our numerical method and analysis technique. Our results are presented in Section 2.3, followed by a general discussion in Section 2.4. We give our conclusions in Section 2.5.

2.2 Method

The simulation and analysis of the collisions presented here combine numerical methods introduced in Paper I and Richardson et al. (2000, hereafter Paper II). The rubble pile model is an extension of the model used for studying the tidal disruption of asteroids (Paper I). The integration engine is an extension of the parallel tree code used for planetesimal evolution simulations (Paper II).

2.2.1 Rubble Pile Model

Each rubble pile in our simulations consists, at least initially, of a fixed number of equal-size hard spheres arranged in hexagonal close-packed (HCP) form. The rubble piles are typically generated by specifying the bulk semi-axes, bulk density, and approximate number of particles (alternatively, the particle radius and/or density can be used as independent parameters). The generator attempts to match the requested properties on the basis of the estimated HCP efficiency of a sphere as a function of bulk radius or number of particles (derived from power-law fits to our own numerical experiments). Once the rubble pile is constructed, the constituent particles are reduced in size by a fixed factor (usually 1%) and given a small random velocity kick (no more than 10% of the particle surface escape speed in magnitude). This is to facilitate attaining the initial equilibrium (cf. Section 2.2.4). Finally, the rubble pile is tagged with a unique color so that mixing can be studied visually and statistically.

The collisional properties of the constituent particles are specified prior to each simulation. These include the normal and tangential coefficients of restitution, ϵ_n and ϵ_t (cf. Richardson 1994). Except for certain explicit test models, these values generally were fixed at $\epsilon_n = 0.8$ (mostly elastic collisions with some dissipation) and

$\epsilon_t = 1.0$ (no surface friction). Bouncing was the only possible collision outcome: no mergers or fragmentations of particles were allowed. The value of ϵ_n was chosen to be consistent with Paper I and is similar to experimentally determined values used in the literature (e.g. Beaugé & Aarseth 1990). Note that in the perfectly elastic case, particles cannot recollapse into condensed rubble piles after a disruption event but instead completely disperse or at best end up in centrally concentrated swarms. In the case of tidal disruption (Paper I) the outcome is relatively insensitive to the choice of ϵ_n, so long as $\epsilon_n < 1$. For the present study, however, varying ϵ_n has a stronger effect, an issue we explore in Section 2.3.2. We did not include surface friction in the present study, in order to keep the number of test cases manageable.

There are two circumstances under which ϵ_n is allowed to change. First, if the relative speed of two colliding *particles* is less than 10% of their mutual escape speed (i.e., typically $\sim$1 cm s^{-1}), ϵ_n is set to unity (no dissipation). This is to prevent computationally expensive "sliding motions" (Petit & Hénon 1987). Second, if the collision speed exceeds 10 m s^{-1}, ϵ_n is set to 0.2 (highly dissipative). This is to crudely model damping through internal fracture as the impact stress $\rho v c$ (ρ = internal density, c = sound speed $\sim 10^3$ m s^{-1}) exceeds the rock strength ($\sim 10^7$ N m^{-2}). This is not intended to be a physically rigorous model but rather a simple mechanism to prevent unrealistically high collision speeds. Initial encounter speeds between rubble piles were generally kept closer to 1 m s^{-1} in any case. Also, particle sizes were kept roughly comparable across rubble piles in order to minimize any strength-versus-size biases.

It is important to note that neither rolling nor true sliding motions are modeled in our code. Moreover, particles cannot remain mutually at rest in contact (i.e., there are no surface normal forces). Instead, the constituent particles of an otherwise quiescent rubble pile are in a constant state of low-energy collisional vibration

(dictated by the minimum sliding condition described above). Nevertheless, such small bounces can mimic transverse motions in an approximate sense in the presence of shear flow, giving realistic bulk properties to the material. To test this we have simulated the formation of sand piles using our collision code (with surface friction) that give reasonable values for the angle of repose when compared with laboratory experiments.

2.2.2 Numerical Code

Our simulations were performed using a modified version of a cosmological N-body code, `pkdgrav` (Richardson et al. 2000; Stadel 2001)[2]. This is a scalable, parallel tree code designed for ease of portability and extensibility. For the parameter space study, the parallel capability was not exploited owing to the modest number of particles in each run (a few thousand). However, even in serial mode, `pkdgrav` is arguably more efficient than any other existing code with similar capability. In particular, it is superior to `box_tree`, the code used in Paper I, which could handle only a few hundred particles in practical fashion.

A low-order leapfrog scheme is used as the `pkdgrav` integrator. The comparative simplicity of this scheme is a big advantage for collision prediction since particle position updates are linear in the velocity term. This means that every possible collision within the time step can be determined in advance and in the correct sequence. Time steps are smaller than in higher-order schemes for the same accuracy, but the cost of each gravity calculation is far outweighed by the collision search once the rubble piles are in contact and is comparable otherwise. Moreover, away from collision, particle trajectories are integrated symplectically, eliminating spurious numerical dissipation. For further detail and references, refer to Paper II.

[2]These references have been updated since publication of this chapter.

Although the collision search is relatively expensive, the scaling is modest: $\mathcal{O}(N \log N)$ with particle number and linear with the number of collisions per interval. A typical encounter between thousand-particle rubble piles can generate $\sim 10^8$ collisions over the course of a run! A balanced k-d tree (Bentley & Friedman 1979) is used to search for possible collisions at the beginning of each time step, giving the $\mathcal{O}(N \log N)$ dependence. Once a collision is performed, only particles that might be affected by the event in the remaining interval (numbering typically $\ll N$) are reconsidered via the neighbor search, giving the near linear dependence on the number of collisions. This latter enhancement is an improvement to the Paper II code, which did not require as much sophistication given the low collision frequency per step. Note that the collision search can also be performed in parallel, which proved necessary for the large-N models presented in Section 2.3.3 below.

2.2.3 Hardware

The parameter space models were run on a local cluster of 16 300-MHz Intel Pentium IIs using the High Throughput Computing (HTC) environment `condor` (cf. `http://www.cs.wisc.edu/condor/`) under RedHat Linux. The `condor` system supports automatic scheduling, submission, and restarting of jobs on shared resources, greatly simplifying management. A typical run required between 12 and 72 wallclock hours to complete and each generated $\sim 25 - 50$MB of data. Models requiring parallel resources were run either on a local cluster of four 433-MHz DEC Alpha PCs connected with a fast ethernet switch, or on a local SGI Origin 200 with four 180-MHz processors running IRIX. Both platforms typically achieved sustained performances of several hundred megaflops.

2.2.4 Initial Conditions

Generation of initial conditions and analysis of results were performed using code auxiliary to `pkdgrav`. The rubble pile generator has already been described (Section 2.2.1). Each new rubble pile was first run in isolation (with or without spin) using `pkdgrav` until the velocity dispersion of the constituent particles achieved a stable equilibrium. Next a new "world" was created by using a small program to position and orient any number of equilibrated rubble piles (always two in the present study) prior to simulation. Spherical bodies were usually given a random orientation in order to reduce the effect of HCP planes of symmetry. Bulk velocities were then applied to each rubble pile. Other rubble pile properties that could be changed at this point included the total mass, bulk radius, bulk density, and color. For the exploration of parameter space, usually only the positions (in the form of y offsets), velocities, spins, and colors were modified. Once all the rubble piles were in place, the world was adjusted so that the center of mass coincided with the origin and the velocity of the center of mass was zero. The output world was then read in by `pkdgrav` and the simulation would begin.

To facilitate the exploration of parameter space, a series of Unix scripts were written to generate and monitor each run. Starting with a given pair of rubble piles and a list of desired initial impact parameters, speeds, and spins, the world generator was run automatically to create the necessary initial conditions and support files in separate run directories. The scheduler `condor` was then invoked to farm the jobs out to all available machines. Analysis was performed on the fly using a machine outside the `condor` pool for maximum efficiency.

The choice of initial conditions was governed largely by prior test simulations. For the parameter space exploration, 10 values of impact parameter b and 10 values

of initial relative speed v were chosen for each set of runs, where a set consisted of a fixed choice of spin and/or offset direction (see Section 2.3 for a complete description of each model). From the test simulations it was clear that only about half of the possible 100 runs for each model were needed to find the representative cases and the $Q_D^\star$ boundary. In a plot of b vs v, the important region is the lower left triangle (see Fig. 2.2 for an example). The b and v values were therefore chosen to sample this region as finely as possible in a practical amount of time. Models with spin were chosen to sample representative combinations of spin and orbital angular momentum at a fixed rotation period.

2.2.5 Coordinate System and Units

We use an inertial Cartesian coordinate system in free space for our simulations, with the origin at the center of mass. In the parameter space studies, the initial motion of the colliding bodies is in the $\pm x$ direction. Any initial impact parameter is measured in the $\pm y$ direction. Most debris actually travels in directions perpendicular to the original axis of motion (cf. Section 2.4.3).

A natural unit for the impact parameter b is the sum of the radii $R_1 + R_2$ of the two (spherical) impactors. Hence $b = 0$ implies a head-on collision while $b = 1$ is a grazing encounter. Note, however, the true trajectories will generally be hyperbolae; no allowance is made for this in the definition. Since tidal effects may play an unpredictable role anyway, we adopt the simpler definition. In the absence of trajectory deflection, the impact angle is then $\phi = \sin^{-1} b$, for $b \leq 1$.

The unit for the initial relative speed v is more complicated. We chose a system in which $v = 0$ indicates no relative motion and $v = 1$ is the estimated critical speed for dispersal. The critical speed is found by equating the initial total kinetic energy with the gravitational binding energy of a rubble pile made up of a spherical and

homogeneous mixture of both colliders,

$$v_{crit} = M\sqrt{\frac{6G}{5\mu R}}, \tag{2.1}$$

where M is the combined mass, G is the gravitational constant, μ is the reduced mass $M_1 M_2/M$, and R is the radius of the sphere that contains the combined mass, assuming the same bulk density:

$$R = (R_1^3 + R_2^3)^{1/3}. \tag{2.2}$$

Note that the actual speed at impact will slightly exceed v due to gravitational acceleration.

In the parameter space models, the initial separation in x for all cases was $\sim 6R$, effectively 2.5 Roche radii for the combined mass, i.e., far enough apart that initial tidal effects were negligible. The total energy of the system was positive in all cases. For completeness, the speed at infinity is given by

$$v_\infty = \left(v^2 v_{crit}^2 - \frac{GM\cos\phi}{3R}\right)^{1/2}, \tag{2.3}$$

and the speed at impact is

$$v_{impact} = \left(v_\infty^2 + \frac{2GM}{R_1 + R_2}\right)^{1/2}. \tag{2.4}$$

2.2.6 Run Parameters

Most `pkdgrav` run parameters assumed default values for these simulations (cf. Paper II). However, in addition to the collision parameters described in Section 2.2.4, the run time, time step, and output frequency were specified explicitly for each model. The run time (t_r) was initially 10 times the characteristic time,

$$t_c \sim \sqrt{\frac{x^3}{GM}}, \tag{2.5}$$

where x is the initial separation. Typically t_r is $\sim$ 36 h. In most cases this is sufficient time for the post-collision system to reach a steady state. Some cases were run longer (typically a factor of 2) if necessary, on the basis of visual inspection of animations.

The time step for each run was set to a small value t_0 times a heuristic scale factor of $1/(2v + 1)$, arrived at by trial and error from our test runs (recall that v is the *initial* speed, so t_0 is a simple constant). The scaling ensures finer intervals for neighbor searches in higher-speed impacts (this is necessary to avoid missing any potential collisions). For our runs, $t_0 = 10^{-5}$ year$/2\pi$, or roughly 50 s. Note that for objects with bulk density a few g cm^{-3} the dynamical time $1/\sqrt{G\rho}$ $\sim$1 h, comfortably large compared to the maximum adopted time step. Generally our simulations are limited by the time needed to deal with particle collisions, so the gravity calculations can be of higher accuracy with little additional cost.

Finally, the output frequency was chosen so that there would be about 200 outputs per run, suitable for smooth animations and analysis.

2.2.7 Analysis Method

Much of our analysis method is similar to that presented in Paper I; the reader is referred to that work for additional details. The basic strategy is to identify the largest post-collision remnant, compute its various properties, and generate statistics for the relative distribution of the smaller fragments. We use a slightly different clump-finding algorithm (Section 2.2.7.1) and now employ a shape drawing technique (Section 2.2.7.2). We have made other refinements that should improve the accuracy of the analysis.

2.2.7.1 Clump finding.

The clump-finding algorithm iteratively refines guesses as to what constitutes a rubble pile by merging groups of particles together in bottom-up fashion. The first guess is that every particle in the system is its own rubble pile. On each pass basic properties are computed for each clump: mass, position, axis lengths, and orientation. Clumps are then compared in pairwise fashion. In order for two clumps to be merged (i.e., to be considered one clump), spheres of diameter equal to the major axes times a fixed linking scale (a dimensionless number > 1, typically 1.1) and centered on each clump must overlap. If the scaled minor axes also overlap, then the clumps are merged. Failing that, if either body has its center of mass in the other's scaled ellipsoid, the bodies are merged. Otherwise, no merge occurs. This process iterates until there are no more mergers during an iteration.

This method is purely geometrical: gravitational groupings are not considered. This was done mostly because there is no natural gravitational length scale in the present context, unlike in Paper I where the Hill radius could be used. However, osculating elements of groups measured with respect to the largest fragment are still calculated and give a good indication of the future evolution of the system. The present method also differs from Paper I in that it treats each clump as an ellipsoid rather than a sphere, allowing more refined boundaries to be drawn. The linking scale of 1.1 was adopted through trial and error (visual inspection).

2.2.7.2 Shape drawing.

During the course of the present investigation we came across some unusual, often asymmetrical shapes following collision events. In order to characterize these forms, a shape-drawing algorithm was devised. The algorithm attempts to trace the outer surface of a given rubble pile (either in cross section or by projection to the $x-y$ plane). The resulting shape is equivalent to what would be measured by laser beams aimed at the surface in the direction of the center of mass. Note that this means that any outcroppings can conceal underlying structure. Generally such complex surfaces are not seen in our models, however (as confirmed by 3D VRML viewing). The projection method is used in the parameter space plots of Section 2.3.

2.2.7.3 Mixing.

The unique color assigned to each rubble pile makes it easy to assess visually the degree of mixing following a collision. In order to make a more quantitative assessment, we have constructed the following statistic,

$$f_{mix} = 1 - \frac{1}{\sqrt{N_v}} \sum_c \left[\sum_v \left(\frac{m_{c,v}}{\sum_c' m_{c',v}} - \frac{m_{c,world}}{\sum_{c'} m_{c',world}} \right)^2 \right]^{1/2}, \tag{2.6}$$

where subscript c denotes a color, subscript v denotes a subvolume of the rubble pile, N_v is the number of subvolumes, and "world" refers to the entire population of particles in the system. Note that particle number is conserved so that $\sum_c m_{c,world} = M$, the total mass of the system. This formula is generalized for any number of components (colors); in the present study only two populations were considered. A mixing fraction of unity implies the rubble pile contains a perfectly homogeneous mixture of the world colors. A value of zero means no mixing has taken place at all.

Spherical subvolumes are used to sample different regions of the rubble pile (which itself need not be spherical). The size of the sample region is set so that it

contains $\sqrt{N}$ particles on average. The center of a subvolume is chosen randomly within a rectangular prism enclosing the rubble pile. A new subvolume is chosen if the region is found to contain fewer than $N^{1/4}$ particles. Otherwise, the argument of the $\sum_v$ in Eq. 2.6 is computed and added to the running sum. This is repeated until $\sqrt{N}$ subvolumes are successfully sampled.

2.3 Results

We now present the results of our simulations. First we describe the parameter space exploration which consisted of numerous runs of modest size. Highlights are shown in Fig. 2.1, where we have endeavored to illustrate the various classes of outcomes. Second we show the dependence on the coefficient of restitution ϵ_n for a particular high-energy run. Finally we present the results of two high-resolution cases and compare with the corresponding moderate-resolution runs.

2.3.1 Parameter Space

We divided our exploration of parameter space into three models: a generic case as a baseline, a case with spinning impactors, and a case with unequal-mass impactors. Graphical summaries of these models are given in Figs. 2.2 and 2.4, which are discussed in detail below.

2.3.1.1 Model A: Equal size, no spin.

Model A, our generic case, consisted of two equal-size rubble piles of 1 km radius and 2 g cm^{-3} bulk density. The rubble piles were generated and equilibrated using the process described in Section 2.2.4. Each rubble pile contained 955 identical spherical

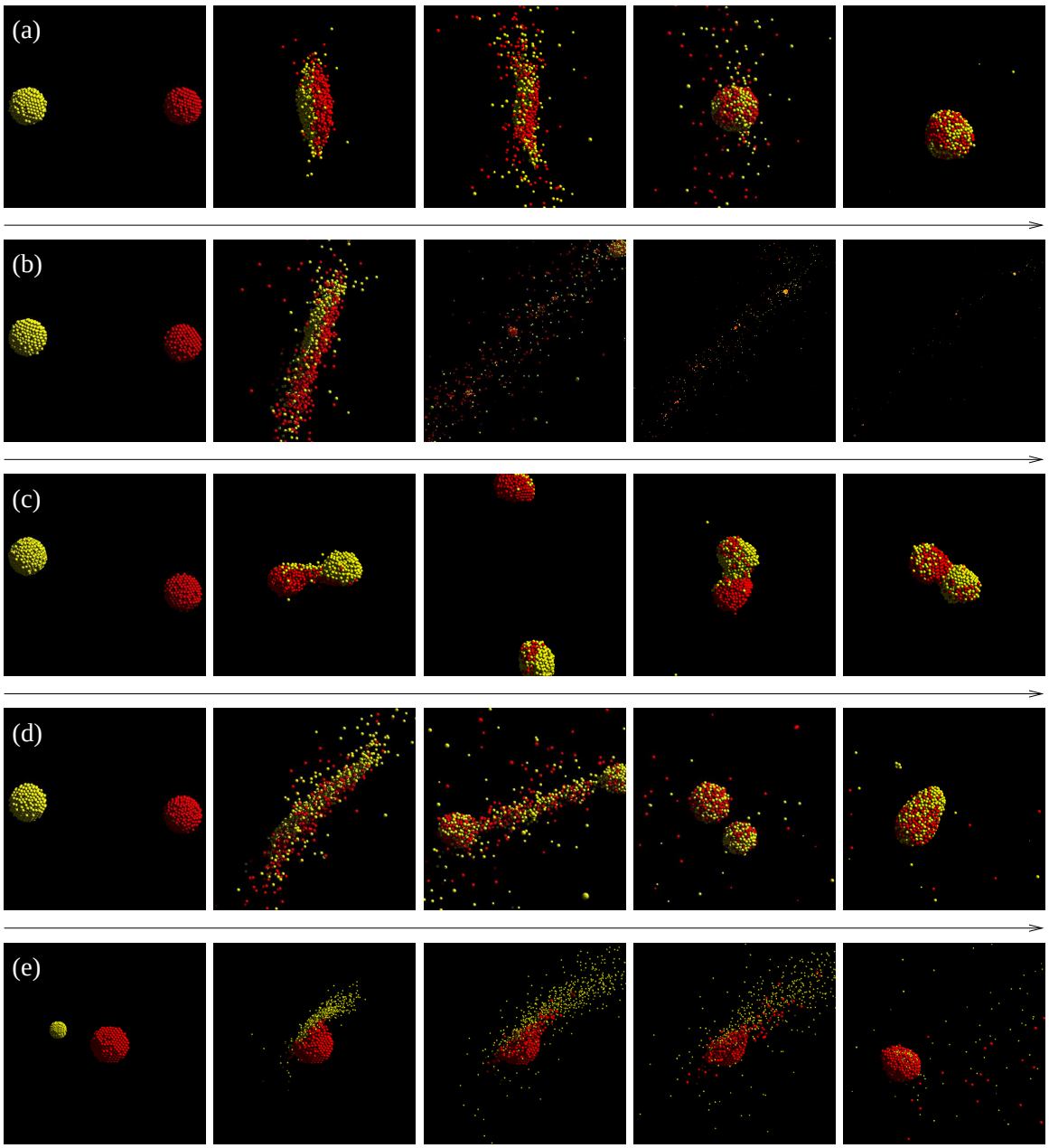

Figure 2.1: Snapshots of rubble pile collisions from representative runs as seen in the center-of-mass frame. The models and runs are: (a) Model A, $b = 0.00$, $v = 1.00$; (b) Model A, $b = 0.15$, $v = 2.00$; (c) Model A, $b = 0.904$, $v = 0.52$; (d) Model B1, $b = 0.30$, $v = 1.10$; and (e) Model C, $b = 0.50$, $v = 1.25$. The arrow of time is to the right. The interval between frames is not regular: the snapshots were chosen to highlight distinct stages in the evolution of each run. In run (b), the final two frames have been brightened for clarity.

particles of 83 m radius, so the packing efficiency was 55%.[3] The parameter space extends from 0.00-1.25 in b and 0.52-2.50 in v (the units of b and v are defined in Section 2.2.5; $v_{crit} = 2.06$ m s^{-1} for this model). The impact parameter values were chosen to encompass a range of dynamic interactions from head-on collisions to glancing distortions. The lowest value of v is twice the value corresponding to $v_\infty = 0$ (cf. Eq. 2.3; smaller v leads to strong trajectory deflections). The largest value of v was chosen to be a moderately high-speed impact to ensure that the catastrophic dispersal regime was entered.

Figure 2.2 summarizes the results of this model (Figs. 2.1(a)-2.1(c) give snapshots of three distinct outcomes). The shapes in Fig. 2.2 trace the projected silhouettes of the largest post-encounter fragment at the end of each run. We have used nested squares of different line styles to divide our results into three mass regimes. A solid-line inner square indicates that the largest fragment contains 90% or more of the total mass of the system, i.e., nearly perfect accretion. A dashed-line inner square indicates that the largest fragment contains at least 50% but less than 90% of the total mass. The remaining cases contain less than 50% of the total mass in the largest fragment, i.e., net erosion. Note if there is no mass loss or exchange during the encounter the largest fragment will contain 50% of the total mass of the system by definition. We see in this model that 18 of 55 runs (33%) result in net mass loss, although we caution that several cases are just on the border of 50%.

The general trends in Fig. 2.2 are twofold, namely, as the encounter speed increases, the size of the largest fragment decreases, and as the impact parameter increases, the axis ratio increases, up to a certain point. Higher encounter speeds imply larger kinetic energy so it is more likely for the system to become unbound.

[3]The effective packing efficiency is less than the maximum close-packed efficiency of 74% due to finite-size effects (Paper I).

Larger impact parameters imply larger angular momentum which results in an increase in the axis ratio until the critical spin value of the combined rubble pile is reached (cf. Eq. 2.7). In addition to the general trends, the middle-mass group has two distinct populations that reflect their formation history. The small b, large v group (top left in the figure) represents a net loss of mass of 10-50% from the system. The large b, small v group (lower right) represents grazing collisions with little mass loss or exchange.

We note that for the head-on case our definition of v_{crit} does not correspond to critical dispersal, rather, critical dispersal seems to occur at $\sim 1.9 v_{crit}$ (~ 4 times the binding energy of the rubble pile). This probably reflects the fact that the energy of the collision is not immediately transported to all of the particles and that the voids in between the particles decrease the efficiency of energy propagation. Moreover, we did not take into account ϵ_n in the definition of v_{crit}. Regardless, v_{crit} is intended as an approximate scaling only.

More detailed results for this model are given in Table 2.1. In the table, b and v have the usual definitions; M_{rem} is the mass fraction of the largest post-encounter remnant; P is its instantaneous spin period in hours; ε is the remnant's "ellipticity": $\varepsilon \equiv 1 - \frac{1}{2}(q_2 + q_3)$, where $q_2 \equiv a_2/a_1$, $q_3 = a_3/a_1$, and $a_1 \geq a_2 \geq a_3$ are the semi-axes ($\varepsilon = 0$ is a sphere); f_{mix} is given by Eq. 2.6; M_{acc}, M_{orb}, and M_{esc} are the mass fractions that are accreting, orbiting, and escaping from the largest remnant, respectively[4]; and n_1, n_2, and n are the number of single particles, two-particle groups, and discrete rubble piles (i.e., groups with three or more particles), respectively, at the end of the run. The M_{rem} column of Table 2.1

[4]To be considered accreting, a clump must have $q < r + R$, where q is the close-approach distance to the remnant, and r and R are the radii of minimal spheres enclosing the clump and remnant, respectively. This differs somewhat from Paper I.

compliments Fig. 2.2 by providing a finer gradation of the remnant mass. Note that $M_{rem} + M_{acc} + M_{orb} + M_{esc} \equiv 1$.

Table 2.1 shows how the remnant spin P is coupled to the angular momentum in each run. Since there are no external torques in the system, angular momentum is conserved. In the case of head-on collisions ($b = 0$), there is exactly zero total angular momentum, which accounts for the large remnant P values (i.e., low spin). P is never infinite in these cases because some particles escape and carry angular momentum away from the remnant, even in the slowest collision case ($v = 0.52$). At higher collision speeds, more mass is carried away from the system, generally resulting in smaller P values. As b increases, so does the net angular momentum, resulting in faster spins (smaller P). This trend continues until $b \sim 1$ which corresponds to a grazing collision. In this case, the encounter generally does not result in a merger so the remnant is effectively one of the initial bodies plus or minus some mass exchange. Mass exchange and/or tidal torquing following deformation converts orbital angular momentum into spin angular momentum. As b increases further, there is little spin-up, since torquing becomes less effective. All of these trends can be seen in the table.

Similarly, ε depends on the total angular momentum of the system. Larger angular momentum allows the remnant to support a more elongated shape as long as most of the system mass ends up in the remnant ($\sim 75\%$, from the table). Consequently there is also a relationship between ε and P: smaller P values correspond to larger ε values, in general. The smallest P in the table is 4.1 h with $\varepsilon = 0.26$; the largest ε is 0.45 with $P = 4.3$ h. These values are within the classical limit for mass retention at the surface:

$$P_{\rm crit} \simeq \frac{1}{1-\varepsilon}\sqrt{\frac{3\pi}{G\rho}}, \tag{2.7}$$

where ρ is the bulk density and we have assumed $a_2 = a_3$. In this expression

Table 2.1: **Summary of Model A Results (Section 2.3.1.1)**

b	v	$M_{\rm rem}$	P	ε	$f_{\rm mix}(\%)$	$M_{\rm acc}$	$M_{\rm orb}$	$M_{\rm esc}$	n_1	n_2	n
0.00	0.52	0.992	641.2	0.09	26 ± 4	0.001	0.000	0.007	15	0	1
0.00	0.61	0.989	281.7	0.08	29 ± 3	0.001	0.000	0.010	21	0	1
0.00	0.75	0.971	220.8	0.06	33 ± 4	0.004	0.000	0.025	55	0	1
0.00	0.90	0.936	181.6	0.05	38 ± 5	0.007	0.000	0.057	114	4	1
0.00	1.00	0.928	110.8	0.05	50 ± 4	0.007	0.000	0.065	134	2	1
0.00	1.10	0.903	114.3	0.04	52 ± 4	0.007	0.000	0.090	182	2	1
0.00	1.25	0.843	116.2	0.08	64 ± 4	0.013	0.000	0.145	285	4	3
0.00	1.50	0.699	20.7	0.07	73 ± 3	0.022	0.000	0.279	531	11	7
0.00	2.00	0.374	19.1	0.04	80 ± 4	0.046	0.016	0.564	938	39	27
0.00	2.50	0.098	5.1	0.31	71 ± 4	0.004	0.007	0.891	1328	34	38
0.15	0.52	0.992	11.9	0.07	25 ± 3	0.002	0.000	0.006	16	0	1
0.15	0.61	0.984	11.0	0.14	28 ± 3	0.002	0.000	0.014	30	0	1
0.15	0.75	0.965	9.1	0.12	30 ± 4	0.005	0.000	0.030	66	0	1
0.15	0.90	0.944	7.4	0.14	37 ± 4	0.006	0.000	0.050	105	1	1
0.15	1.00	0.924	6.9	0.12	40 ± 4	0.004	0.000	0.072	142	2	1
0.15	1.10	0.885	6.1	0.13	47 ± 4	0.016	0.000	0.099	197	4	4
0.15	1.25	0.818	5.7	0.07	59 ± 3	0.012	0.001	0.169	324	8	2
0.15	1.50	0.695	5.6	0.09	59 ± 3	0.017	0.008	0.280	529	12	6
0.15	2.00	0.275	8.4	0.04	52 ± 5	0.009	0.005	0.710	939	24	30
0.30	0.52	0.994	6.9	0.20	21 ± 3	0.001	0.000	0.005	11	0	1
0.30	0.61	0.988	6.0	0.20	28 ± 3	0.002	0.000	0.010	23	0	1
0.30	0.75	0.974	5.3	0.22	34 ± 3	0.002	0.000	0.025	50	0	1
0.30	0.90	0.946	4.6	0.20	37 ± 3	0.006	0.001	0.047	103	0	1
0.30	1.00	0.901	4.4	0.24	37 ± 4	0.009	0.010	0.081	184	3	1
0.30	1.10	0.887	4.5	0.41	40 ± 4	0.018	0.010	0.085	202	5	2
0.30	1.25	0.786	4.5	0.42	42 ± 4	0.013	0.020	0.182	366	10	7
0.30	1.50	0.408	7.6	0.09	32 ± 4	0.029	0.007	0.555	481	17	12
0.45	0.52	0.995	5.2	0.33	23 ± 3	0.000	0.000	0.005	10	0	1
0.45	0.61	0.986	4.8	0.39	24 ± 4	0.002	0.000	0.012	26	0	1
0.45	0.75	0.982	4.3	0.40	26 ± 4	0.002	0.001	0.015	35	0	1
0.45	0.90	0.490	9.2	0.13	25 ± 4	0.457	0.009	0.043	105	0	2
0.45	1.00	0.469	7.9	0.15	25 ± 4	0.003	0.006	0.523	140	2	2
0.45	1.10	0.442	9.6	0.11	23 ± 4	0.008	0.006	0.545	191	4	4
0.45	1.25	0.416	9.2	0.06	20 ± 3	0.008	0.004	0.572	295	6	7
0.60	0.52	0.997	4.5	0.40	21 ± 3	0.000	0.000	0.003	6	0	1
0.60	0.61	0.989	4.2	0.38	24 ± 3	0.002	0.003	0.007	19	1	1
0.60	0.75	0.512	8.1	0.09	20 ± 4	0.468	0.003	0.018	41	0	2
0.60	0.90	0.499	8.1	0.13	18 ± 3	0.005	0.005	0.491	93	3	2
0.60	1.00	0.460	12.1	0.11	14 ± 2	0.003	0.003	0.535	134	3	4
0.60	1.10	0.454	10.9	0.05	11 ± 2	0.013	0.003	0.530	150	5	6

Table 1.1: Summary of Model A Results (continued)

b	v	$M_{\rm rem}$	P	ε	$f_{\rm mix}(\%)$	$M_{\rm acc}$	$M_{\rm orb}$	$M_{\rm esc}$	n_1	n_2	n
0.75	0.52	0.998	4.1	0.26	23 ± 4	0.000	0.001	0.001	4	0	1
0.75	0.61	0.996	4.9	0.39	23 ± 3	0.002	0.001	0.001	8	0	1
0.75	0.75	0.493	10.1	0.12	9 ± 2	0.001	0.001	0.506	32	0	2
0.75	0.90	0.484	10.0	0.14	7 ± 2	0.003	0.004	0.510	68	2	2
0.75	1.00	0.470	20.0	0.13	6 ± 2	0.003	0.002	0.525	111	3	3
0.90	0.52	0.999	4.3	0.45	16 ± 3	0.000	0.001	0.000	2	0	1
0.90	0.61	0.504	10.0	0.14	8 ± 2	0.000	0.002	0.494	10	1	2
0.90	0.75	0.496	13.9	0.16	4 ± 1	0.002	0.000	0.502	23	0	3
0.90	0.90	0.492	14.5	0.12	3 ± 1	0.001	0.000	0.507	42	0	4
1.00	0.52	0.502	9.6	0.12	6 ± 2	0.000	0.001	0.498	5	0	2
1.00	0.61	0.502	13.8	0.11	5 ± 1	0.002	0.001	0.495	13	0	2
1.00	0.75	0.499	16.6	0.20	3 ± 1	0.001	0.000	0.499	16	1	2
1.10	0.52	0.501	12.6	0.09	4 ± 1	0.000	0.001	0.498	4	0	2
1.10	0.61	0.499	17.2	0.12	2 ± 1	0.001	0.000	0.500	5	0	2
1.25	0.52	0.501	48.4	0.07	1 ± 1	0.000	0.000	0.499	2	0	2

$P_{\rm crit} = 2.3$ h for a spherical rubble pile with $\rho = 2$ g cm^{-3}, and increases to infinity as $\varepsilon \to 1$.

The sixth column in Table 2.1, labeled $f_{\rm mix}$, gives the *mean* percent mixing fraction and standard deviation after 100 repeated measurements (recall the mixing calculation subvolumes are chosen randomly—cf. Section 2.2.7.3). The errors are a small fraction of the mean except when the mixing fraction itself is small. In the head-on case, $f_{\rm mix}$ shows a simple trend of generally increasing with impact energy with a dip at the highest energy probably due to increased statistical fluctuation (the remnants are smaller). For most b values the situation is more complicated depending on whether the impactors accrete into a single body or exchange mass while remaining two separate bodies. For the largest b, little mass is exchanged, so the bodies remain essentially unmixed.

The next three columns give dynamical information about the remaining mass of the system, i.e. the material not incorporated in the largest remnant. Generally most of this mass is escaping from the largest remnant ($M_{\rm esc}$). Typically only

small amounts ($< 10\%$) of mass are accreting ($M_{\rm acc}$) and/or orbiting ($M_{\rm orb}$). In two cases, however, $M_{\rm acc}$ is close to 50%; these are instances of near escape that were too computationally expensive to run until final accretion and represent the transition from a high- to medium-mass remnant.

The final three columns contain information about the particle groupings at the end of each run. The number of free particles (n_1) increases dramatically with v, but decreases with b. This trend is also seen in the number of two-particle groups (n_2) and discrete rubble piles (n). Groups can form either from accretion among the free particles due to gravitational instability or from being stripped off as a clump during the collision event. Note that n is always at least 1 because the remnant is included.

In summary, the outcomes of this model depend in a natural way on the total angular momentum and impact energy of the system (both related to b and v). Larger b results in more elongated remnants with higher spins and reduced mixing. Larger v results in greater mass loss and increased mixing. In the remaining sections we explore how these trends are modified for non-identical or spinning bodies.

2.3.1.2 Model B: Equal size, spin.

In Model B we added a spin component to the impactors. The spin vectors are oriented perpendicular to the orbital plane (i.e. along the $\pm z$ axis). The rotation period of the impactors is 6 h, the median rotation period of Near Earth Asteroids (Bottke et al. 1997). We investigated three cases: in Model B1 the spins of the impactors have opposite orientation; in Model B2 and B3 the spins have the same orientation but the impactors have opposite y offsets (Fig. 2.3). By symmetry, these cases test all the unique z angular momentum combinations (spin + orbital). The remaining parameters are identical to those in Model A.

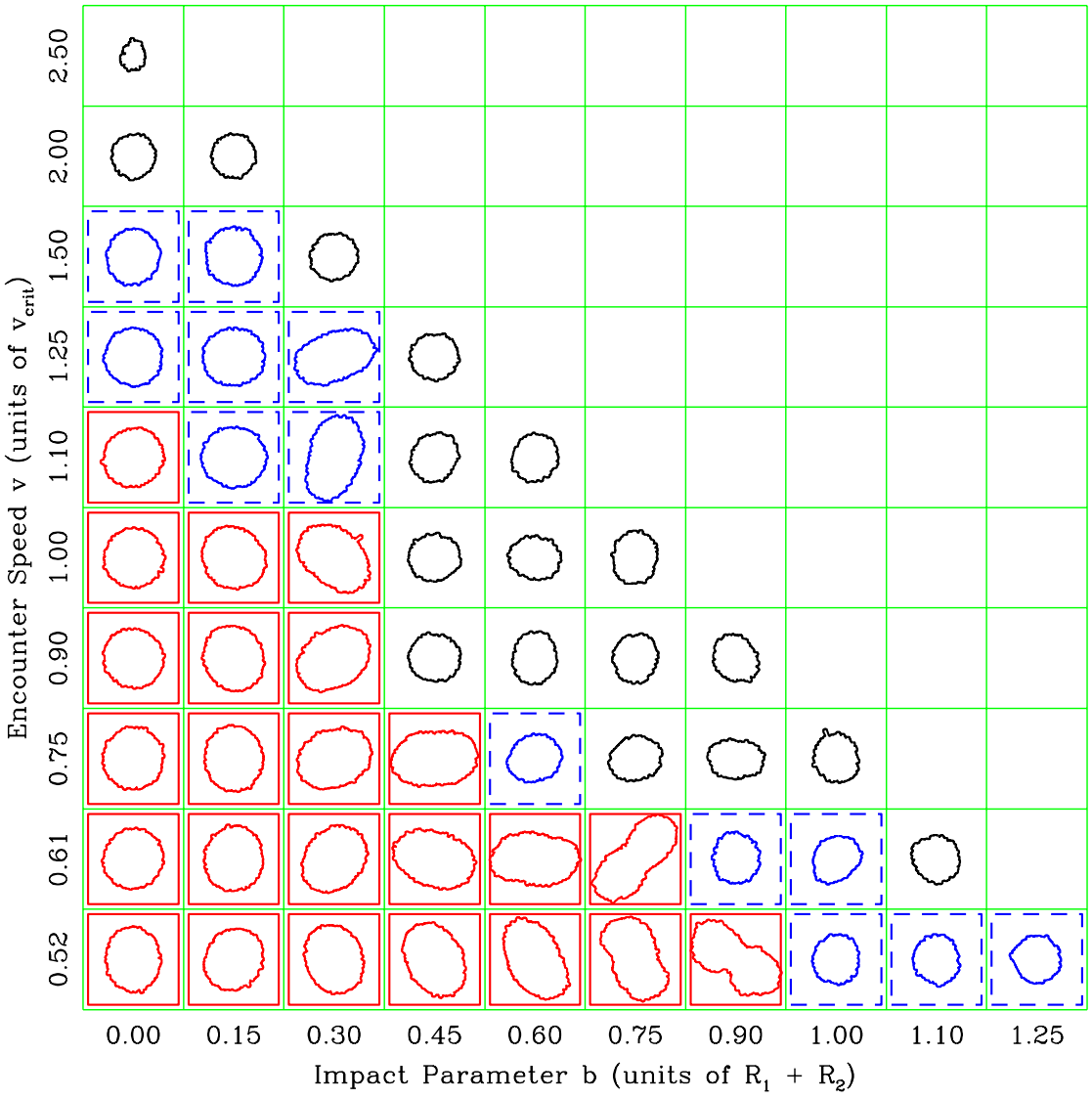

Figure 2.2: Projected shape of the largest remnant at the end of each Model A run as a function of b and v. At this scale each grid square measures 4 km on a side. Solid inner squares indicate remnants that retain at least 90% of the system mass; dashed squares indicate remnants with at least 50%. Critical dispersal generally corresponds to the transition from solid to dashed, although in some cases a sizeable fragment may be about to accrete with the remnant. Table 2.1 gives additional data for this model.

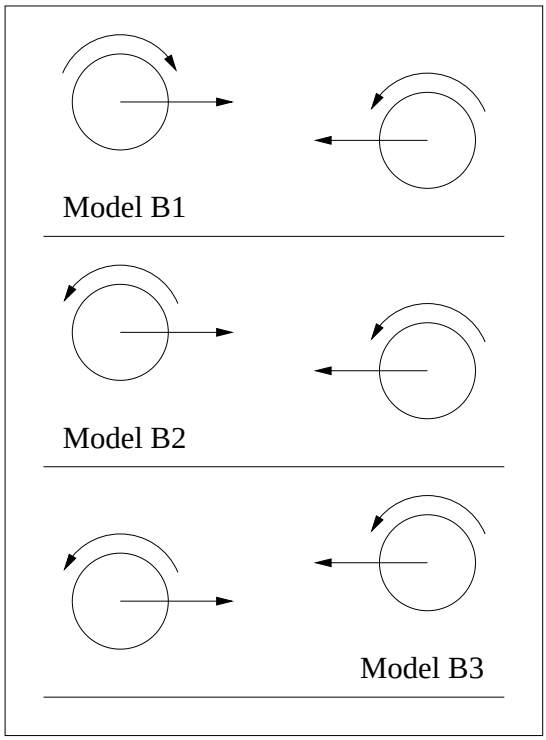

Figure 2.3: Illustration of the spin sense for the Model B impactors. In Model B1, the impactors have opposite spin; in B2 they have the same spin, oppositely aligned with the orbital angular momentum; in B3 the spins are aligned with the orbital angular momentum.

Figures 2.4(a)–(c) summarize the results for Models B1, B2, and B3, respectively (Fig. 2.1(d) is a snapshot sequence of a Model B1 run). The general trends seen in Fig. 2.4 are similar to those seen in Fig. 2.2. The head-on cases tend to result in spherical remnants of decreasing mass with increasing v. The elongation of the remnants tends to increase with an increase in b, up to a point. Of the three models note that Model B1 is the most similar to Model A. This is because Model B1 has the same amount of net angular momentum in the system since the spin components of the impactors cancel. Model B2 and Model B3, however, have smaller and larger net angular momentum in the system, respectively, than Model A or B1. This is reflected in the number of runs with fast-rotating and/or elongated remnants (Table 2.2). Models A and B1 have an intermediate number of runs with extreme P and/or

Table 2.2: **Comparison of runs with extreme P and e values (Section 2.3.1.2)**

Model	No. with $P \leq 5$ h	No. with $e \geq 0.35$
A	11	8
B1	10	8
B2	10	2
B3	15	10
C	0	0

ε values compared with Model B2 or B3 (Model C is a special case discussed in the next section).

Model B1 does differ from Model A in one important respect. As seen in Fig. 2.4(a), some of the remnants in this model (e.g. $b = 0.30$, $v = 1.10$; $b = 0.60$, $v = 0.61$) have unique asymmetries (i.e. broken eight-fold symmetry). This is because before the encounter one of the bodies is spinning prograde while the other body is spinning retrograde with respect to the orbit. The prograde rotator has larger angular momentum with respect to the center of mass of the system than its retrograde counterpart, consequently, it suffers more mass loss. This is analogous to the resistance of retrograde rotators to tidal disruption (Richardson et al. 1998).

To summarize other quantitative results, 24% of the Model B1 runs resulted in net erosion, while this value was 29% for B2, and 40% for B3. The mixing statistics are generally similar to those for Model A, namely that larger disruption resulted in more mixing. As for ejecta statistics, again no more than about 2% by mass remains in orbit around the remnant in all cases, while a somewhat larger percentage is destined to reaccrete (no more than $\sim 6\%$, except for a few cases where components of a future contact binary were on slow-return trajectories). The distribution of fragments (n_1, n_2, and n) followed similar trends to those of Model A.

2.3.1.3 Model C: Unequal size, no spin.

In Model C we used two different-sized impactors with no initial spin: one large sphere of 1357 particles and 1 km radius, and one small sphere of 717 particles and 0.46 km radius, keeping the bulk densities the same (2 g cm^{-3}) and the total number of particles similar to the previous models. Hence the larger sphere is ten times the mass of the smaller sphere and the particles in the two impactors are different sizes (the smaller body has smaller particles, to ensure adequate resolution). We caution that the difference in particle sizes implies different packing efficiencies (porosities) which may affect the outcome (cf. Section 2.3.3). Both impactors were equilibrated using the same process as before. Note that the parameter space investigated is different from the previous models, primarily for better sampling of the tidal regime (large b, small v). For this model, $v_{\rm crit} = 2.9$ m s^{-1}. As for the previous models, the minimum v is twice the value corresponding to $v_\infty = 0$.

In Fig. 2.4(d) it is evident that most collisions result in net growth of the larger body (only 9 cases, or 16%, result in net erosion e.g. remnants with less than 90% of the total mass of the system). None of the encounters resulted in critical *dispersal* and only the highest-speed cases resulted in violent disruption of the combined system (i.e. $b = 0$, $v = 2.5$). Also the remnants are all roughly spherical (the largest $\varepsilon = 0.26$). Fig. 2.1(e) shows a typical encounter: the small body is pulverized and in this case planes off a chunk of the larger body (so n_1 is typically a few hundred in all runs except the most grazing, while n_2 and n remain small, ≤ 10). Most of the smaller fragments escape the system, a tiny fraction ($< 1\%$ by mass) go into orbit around the remnant, while the remaining fragments return and blanket the remnant in the equatorial plane. The largest concentration of smaller particles is at the impact site. The rotation periods of the remnants in this model are typically long compared to those of the previous models due to the larger rotational inertia

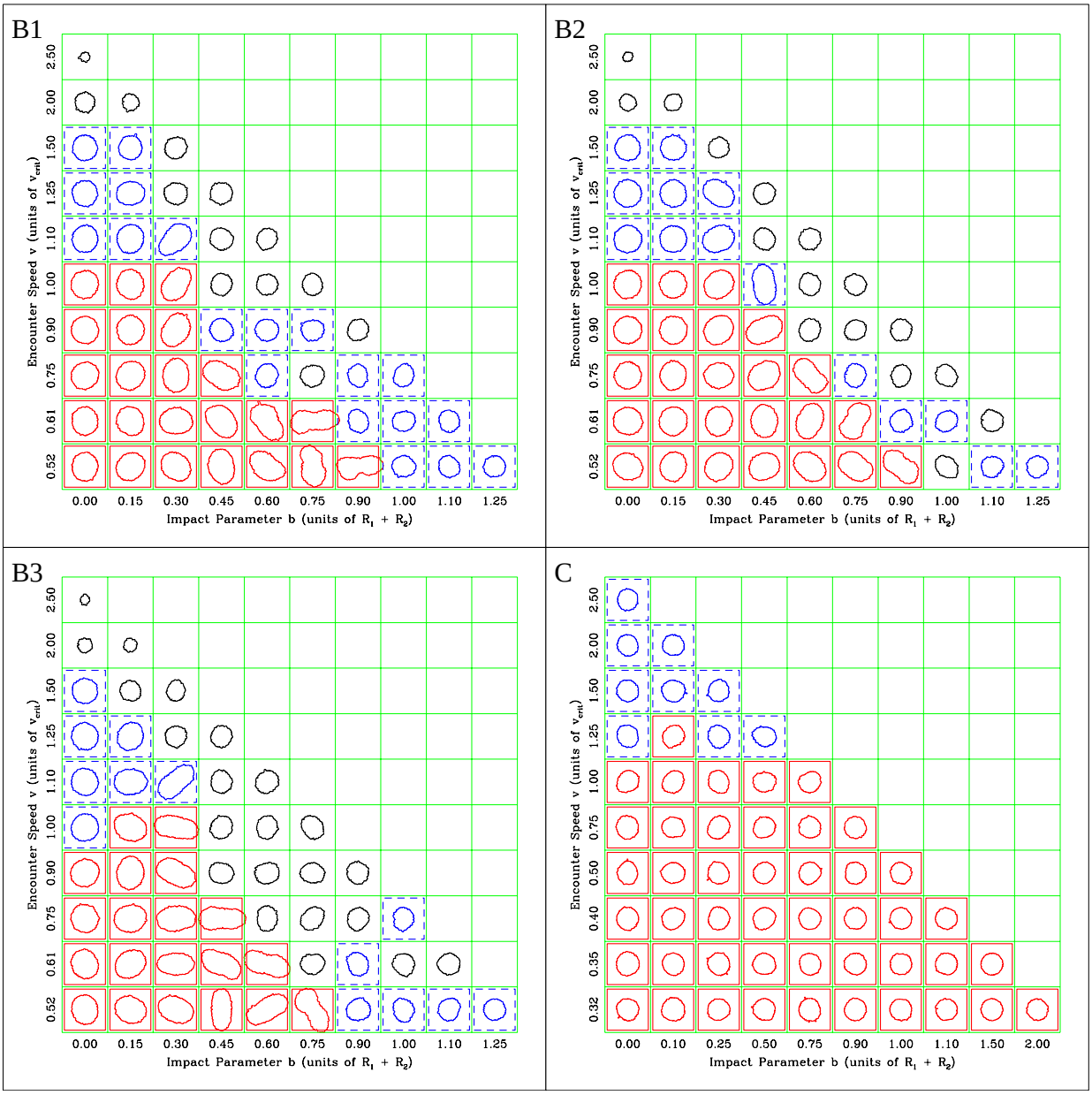

Figure 2.4: Remnant shapes for the remaining parameter space models. The model is indicated in the top left of each plot. Compare with Fig. 2.2.

of the bigger impactor. Finally, there was little tidal interaction seen in any of the cases, suggesting even the minimum v was too large. Unfortunately, smaller v would result in stronger path deflections, making interpretation more difficult.

Table 2.3: **Effect of varying dissipation in Model A run $b = 0.15$, $v = 2.00$ (Section 2.3.2)**

ϵ_n	$M_{\rm rem}$	$M_{\rm acc}$	$M_{\rm orb}$	$M_{\rm esc}$	n_1	n_2	n
0.2	0.196	0.177	0.028	0.598	420	29	44
0.5	0.364	0.006	0.004	0.626	483	31	33
0.6	0.301	0.007	0.007	0.685	597	31	41
0.7	0.284	0.029	0.006	0.682	746	32	39
0.8	0.275	0.009	0.005	0.710	939	24	30
0.9	0.040	0.006	0.005	0.949	1484	63	26
1.0	0.001	0.000	0.000	0.999	1904	3	0

2.3.2 Coefficient of Restitution Test

The energy change in the center-of-mass frame of a system of two smooth, colliding spheres is given by (e.g. Araki & Tremaine 1986):

$$\Delta E = -\frac{1}{2}\mu(1-\epsilon_n^2)v_n^2, \tag{2.8}$$

where v_n is the component of relative velocity normal to the mutual surfaces at the point of contact, and μ and ϵ_n have the usual definitions. Hence as $\epsilon_n \to 0$, all the impact energy—less a geometric factor that depends on b—is dissipated. Although a collision between two rubble piles consists of many individual particle collisions, the dependence of ΔE on ϵ_n suggests that $Q_D^\star$ will depend on ϵ_n in a similar way, namely that smaller ϵ_n implies larger $Q_D^\star$.

A simple test bears this out. Table 2.3 and Fig. 2.5 summarize the effect of varying ϵ_n for one of the Model A runs ($b = 0.15$, $v = 2.00$; cf. Fig. 2.1(b)). The general trend is clear: as ϵ_n decreases, the size of the largest remnant increases (note the large $M_{\rm acc}$ value for the $\epsilon_n = 0.2$ case, indicating that a big fragment is about to merge with the remnant, giving it the largest mass of all the runs). Runs with smaller ϵ_n form discrete rubble piles out of the collision debris faster and more efficiently than those with larger ϵ_n. For $\epsilon_n = 1$, no rubble piles actually form. The

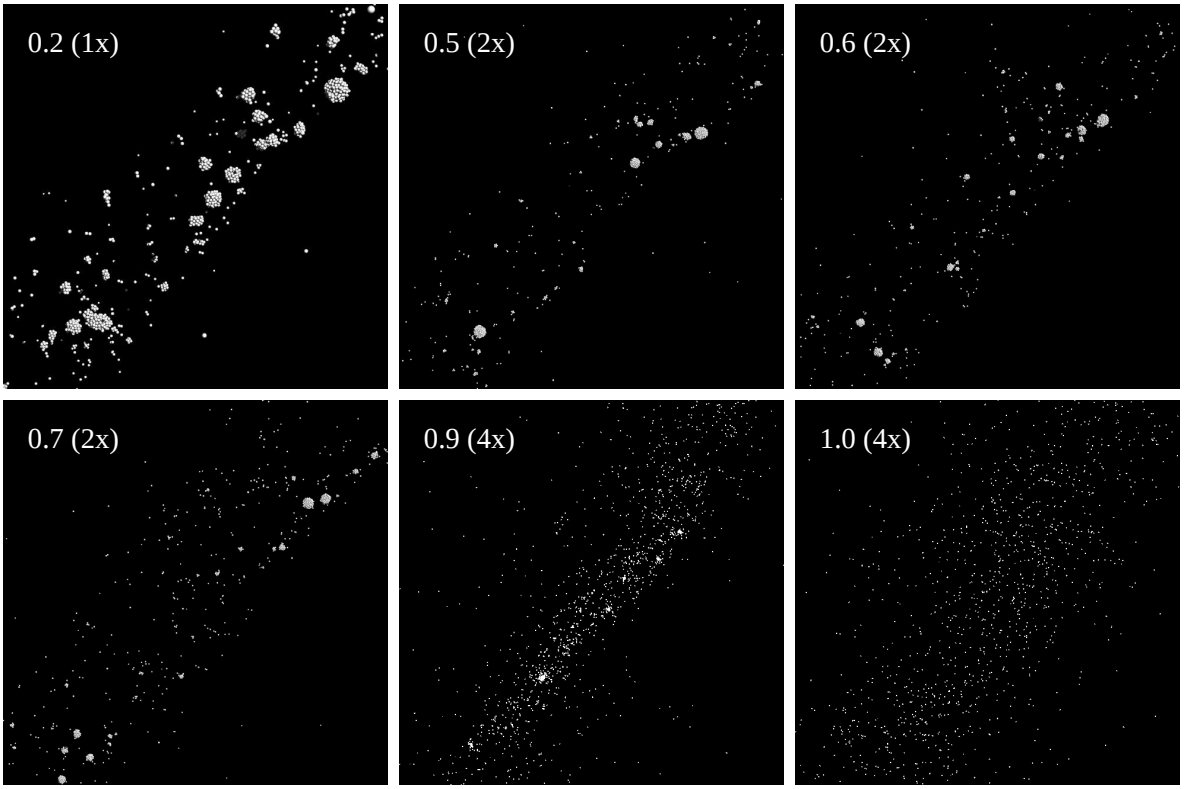

Figure 2.5: Snapshots showing the effect of varying ϵ_n for the Model A run with $b = 0.15$, $v = 2.00$ (cf. Fig. 2.1b). Each snapshot was taken about 6.5 h after impact. The chosen ϵ_n value and camera zoom-out factor are shown in the top left of each frame. For clarity, no color or shading distinction is made between the particles of the original impactors, and the $\epsilon_n = 0.9$ and $\epsilon_n = 1.0$ frames have been brightened. From these snapshots and the statistics in Table 2.3 it can be seen that rubble pile formation favors smaller ϵ_n values. The differences at the extremes are dramatic.

strong dependence on ϵ_n suggests that further study is needed to determine the value most representative of true rubble pile collisions.

2.3.3 High-Resolution Models

In order to test the degree to which particle resolution affects the collision outcome, we performed two high-resolution runs using parameters drawn from Model A. Each impactor for this test consisted of 4,995 identical particles, more than 5 times the number used in the parameter space runs (Section 2.3.1). The progenitor needed 1

CPU day to equilibrate using 2 processors on the SGI Origin 200. At equilibrium, the code was performing $\sim 4 \times 10^4$ collisions per step, with each step requiring ~ 140 s wallclock time. The enhanced packing efficiency of the high-resolution impactors plus their randomized orientations makes detailed comparison with the low-resolution runs difficult. However, we would expect the general trends to be similar (i.e. outcome class, etc.). Figure 2.6(a) shows snapshots shortly after the initial impact comparing the low- and high-resolution Model A runs with $b = 0.30$, $v = 1.25$. Figure 2.6(b) shows post-reaccretion snapshots for $b = 0.60$, $v = 0.61$. These runs were chosen because they are moderately well separated in b-v space while still being representative of the complex intermediate-energy regime (cf. Fig. 2.2). The expense of these calculations precluded a more thorough sampling.

Both high-resolution runs in this test show evolution similar to that of their low-resolution counterparts. In Fig. 2.6(a), the impactors mutually penetrate and lose most of their relative orbital energy (the bodies will eventually accrete into a single massive remnant). Note the presence of the "mass bridge" between the two bodies in both cases. The rotational phase and penetration distance differ somewhat, perhaps indicating that higher resolution (and hence lower porosity) gives rise to more efficient dissipation, by increasing the degrees of freedom. In Fig. 2.6(b), the final shape of the reaccreted body at low and high resolution is similar, but there is more structural detail in the high-resolution remnant, e.g. the depression on the upper surface. The remnant mass and rotation period are comparable at this instant: 0.985 and 4.2 h respectively at low resolution; 0.987 and 4.1 h at high resolution. We conclude that higher resolution may give insight into the more detailed aspects of reaccumulation, but low resolution is sufficient for a broad sampling of parameter space.

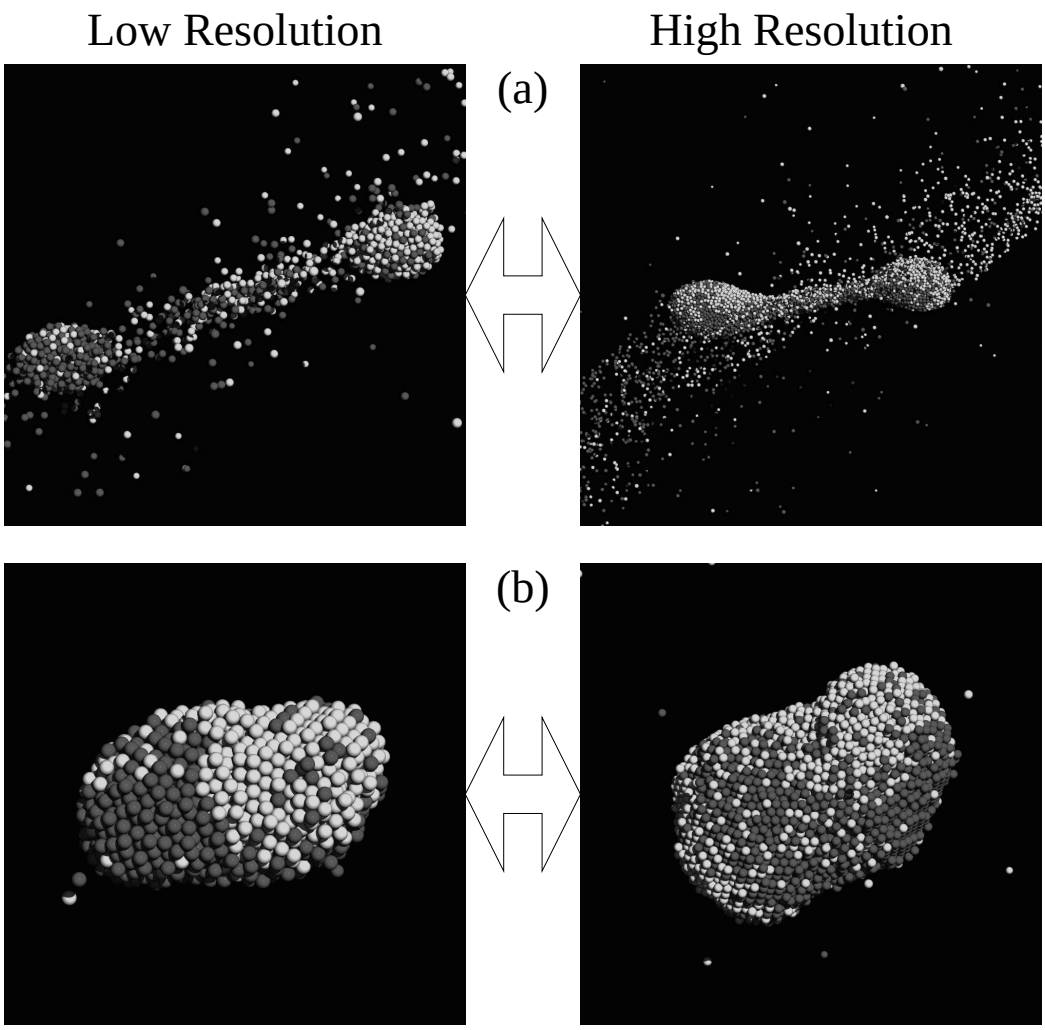

Figure 2.6: Comparison of two Model A runs performed at low resolution (955 particles per rubble pile; left column) and high resolution (4995 particles per rubble pile; right column). The run parameters are: (a) $b = 0.30$, $v = 1.25$; (b) $b = 0.60$, $v = 0.61$. The evolution is similar in both cases, with differences attributable to packing efficiency, initial orientation, and possibly enhanced dissipation at higher resolution.

2.4 Discussion

2.4.1 Critical Dispersal Threshold

Despite the large number of runs carried out for this investigation, the data are still too sparse in each model to reliably derive a generalized expression for the retained mass (remnant plus accreting and orbiting material) as a function of b and v, i.e. $1 - M_{\rm esc} = f(b, v)$. However, we can solve for the critical contour $f(b, v) = 0.5$, which is well sampled by our choice of parameter space (for Model C, we solve for $f(b, v) = 0.9$, the point of net erosion for the larger impactor). Our method is to perform bi-linear interpolation of our b-versus-v results onto a regular grid, root solve using Newton's method for the v value that gives a remnant mass of 0.5 at each grid line in b (we chose 20 lines for smooth sampling), and fit the resulting values to a functional form. After some experimentation, we found the contour is best represented by a Gaussian:

$$v|_{f=0.5} \equiv v_\star = \alpha \exp\left[-\frac{(b-\beta)^2}{\gamma}\right] + \delta, \tag{2.9}$$

where α, β, γ, and δ are parameters to be determined by non-linear least-squares fitting.

Figure 2.7 gives the best-fit values of the Gaussian parameters along with their 1-σ uncertainties for each of the parameter space models. Note that the fits are marginally consistent with $\beta = 0$, i.e. no b offset, except for Model C. The differences between the fits (except Model C) are slight, but they follow the trend mentioned in Section 2.3.1.2, namely that Model B2 has a higher disruption threshold than Model B3, with Models A and B1 having intermediate thresholds. Model C has a broader distribution that is somewhat offset in b, but inspection of the other models shows similar trends for the 0.9 contour.

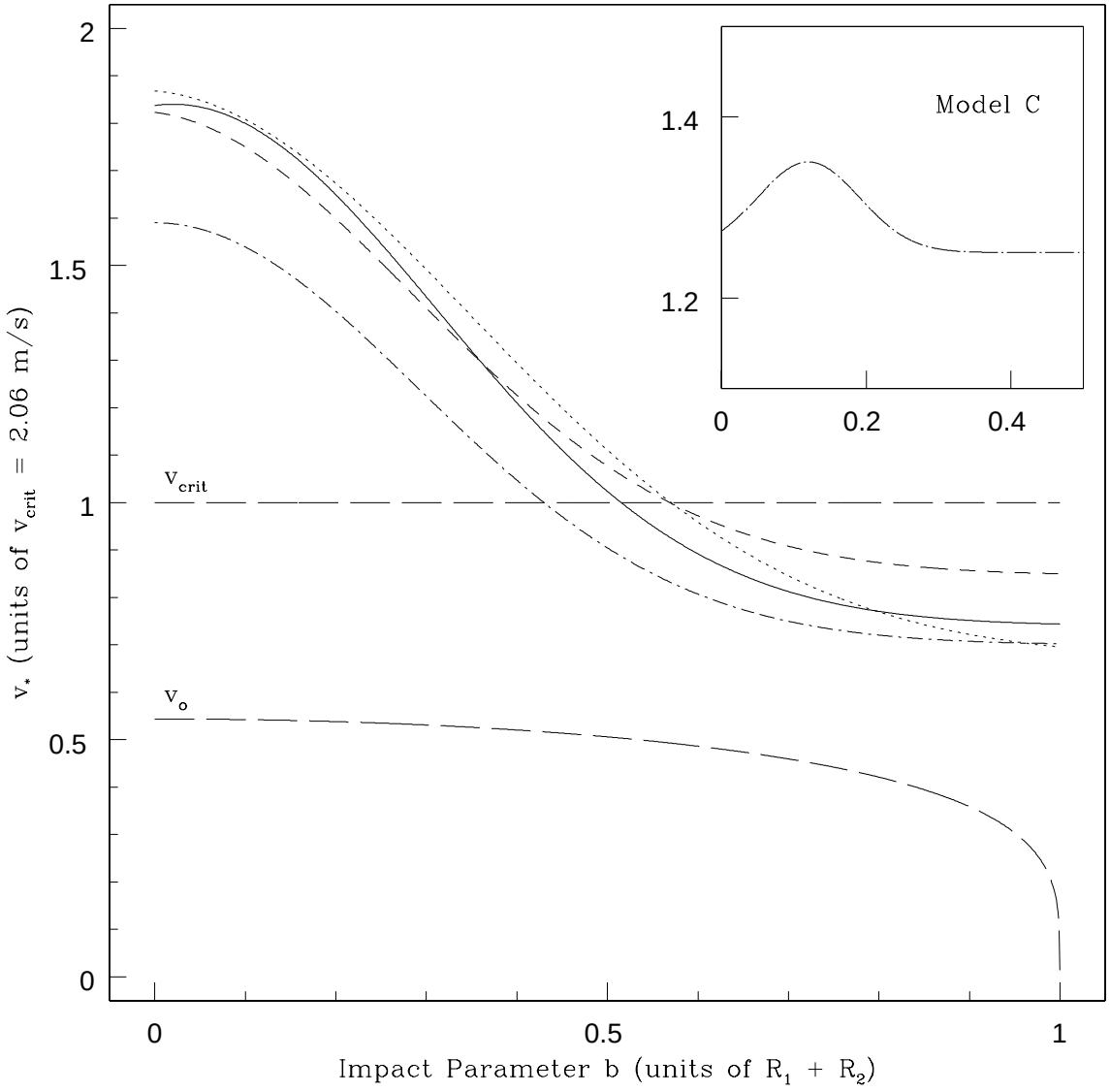

Figure 2.7: Best fits to Eq. 2.9 for $M_{\rm rem} = 0.5$ contours, $b \leq 1$: Model A (solid line; $\alpha = 1.10 \pm 0.03$, $\beta = 0.02 \pm 0.01$, $\gamma = 0.17 \pm 0.01$, $\delta = 0.74 \pm 0.02$), B1 (short dashed; $\alpha = 0.98 \pm 0.01$, $\beta = -0.024 \pm 0.005$, $\gamma = 0.190 \pm 0.005$, $\delta = 0.846 \pm 0.006$), B2 (dotted; $\alpha = 1.2 \pm 0.1$, $\beta = -0.02 \pm 0.04$, $\gamma = 0.27 \pm 0.05$, $\delta = 0.67 \pm 0.06$), B3 (dot-short dashed $\alpha = 0.89 \pm 0.03$, $\beta = 0.00 \pm 0.02$, $\gamma = 0.17 \pm 0.02$, $\delta = 0.70 \pm 0.02$), and C (inset; $\alpha = 0.10 \pm 0.03$, $\beta = 0.12 \pm 0.04$, $\gamma = 0.010 \pm 0.009$, $\delta = 1.25 \pm 0.02$, for the $M_{\rm rem} = 0.9$ contour, $b \leq 0.5$). The $v_\circ$ curve is the $v_\infty = 0$ contour. The value of $v_{\rm crit}$ is given by Eq. 2.1.

2.4.2 Debris Size Distributions

Combining all 275 runs of Models A, B, and C, we find the largest primary (remnant) mass is 0.999 (so there were no perfect mergers), the largest secondary mass is 0.498 (there was always some grazing mass exchange, at least in Models A and B), and the largest tertiary mass is 0.073. The smallest primary mass is 0.038. In Models A and B, the most common outcome was an even split in mass between the primary and secondary, since most runs at moderate to large b resulted in little to no mass exchange between the impactors. For $b \leq 0.30$ ($\phi \leq 17°$), the normalized primary mass function is well approximated by a curve of the form $n(m) \propto 1/(1 - m^2)$, $m < 1$ (Fig. 2.8).

2.4.3 Debris Spatial Distributions

In cases where debris escapes the central remnant, the ejected material is invariably concentrated in a plane normal to the orbital ($z = 0$) plane, although in some cases material can be spread out in the orbital plane as two returning fragments coalesce. For our head-on collisions ($b = 0$), the dispersal plane is normal to the x axis. For $b > 0$, the plane is initially normal to the impact angle ϕ, but rotational inertia from the orbital motion causes the dispersal plane to overshoot this value. As usual, initial spin may help or hinder this process (note for Model B3, $\phi \sim -\sin^{-1} b$).

Figure 2.9 illustrates the anisotropic distribution of ejecta as projected to the orbital plane for 4 of the 5 runs shown in Fig. 2.1. In the equal-mass cases the angular distribution is bi-modal, with the peaks roughly 180° apart. For the plot representing Fig. 2.1(b), the peaks are of unequal amplitude since the remnant is relatively small and displaced from the system center of mass. The one unequal-mass case is uni-modal, indicating that debris was scattered preferentially in one direction

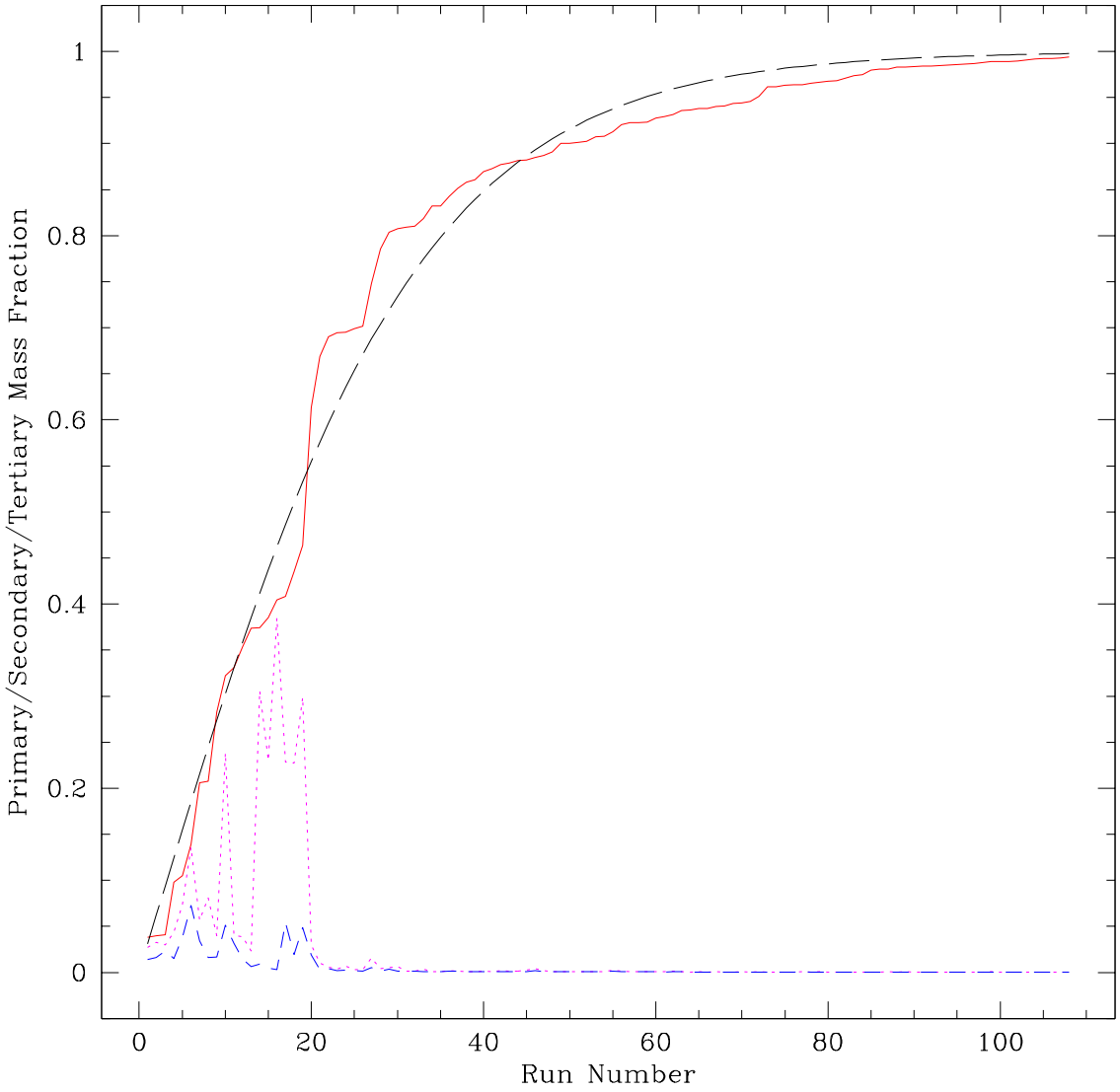

Figure 2.8: Primary (solid line), secondary (dotted), and tertiary (short dashed) mass fractions of every Model A and B run with $b \leq 0.30$, sorted by primary mass. The long-dashed line was obtained by integrating a rough fit to the primary mass function of the form $1/(1-m^2)$.

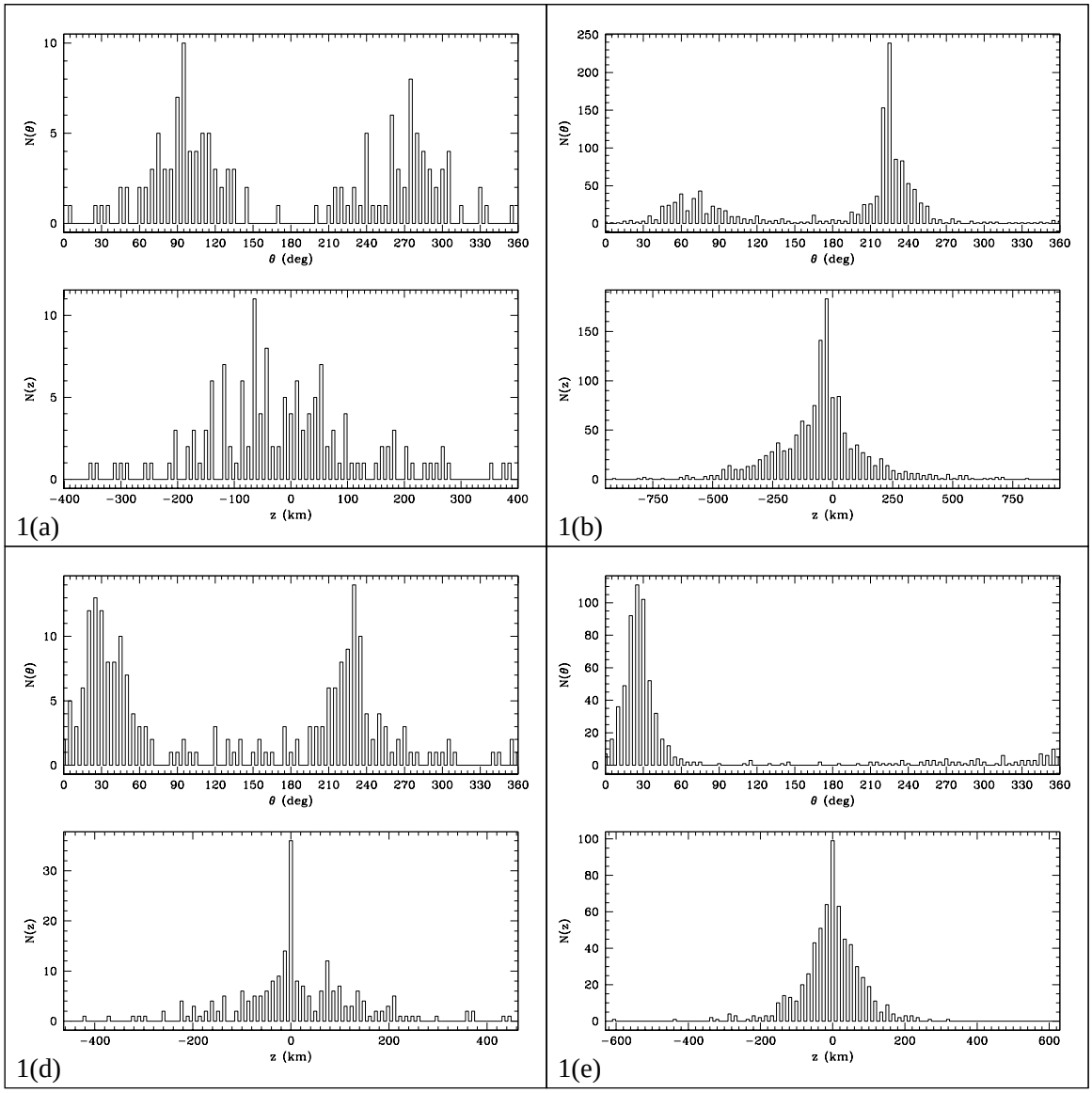

Figure 2.9: Debris dispersal patterns in the initial orbital plane relative to the largest remnant for the runs labeled (a), (b), (d), and (e) in Fig. 2.1. The θ histograms are binned in 5° increments. Only particles with projected distances in the z plane exceeding twice the remnant radius were included.

(roughly 30° measured counterclockwise from the x axis), as seen in Fig. 2.1(e).

Generally the z distributions are sharply peaked near the largest remnant but some particles end up many hundreds of km away. Recall that the tidal field of the Sun is not included in our simulations. If it were, these particles would be well

outside the remnant's Hill sphere at 1 AU (for example):

$$r_H = (210 \text{ km}) \left[\frac{a}{1 \text{ AU}}\right] \left[\frac{R}{1 \text{ km}}\right] \left[\frac{\rho}{2 \text{ g cm}^{-3}}\right]^{1/3}, \tag{2.10}$$

where a is the distance to the sun, R is the radius of the remnant, and ρ is its bulk density. Regardless, most of these particles would escape the remnant, even without the solar tides.

2.4.4 Outcome Probability

For a given impact parameter and speed distribution, the probability of a net accretional (as opposed to net erosional) outcome can be estimated from Eq. 2.9. Suppose we set $b = 0.7$, which corresponds to $\phi = 45°$, the most probable impact angle for randomly flying projectiles striking a spherical target (Love & Ahrens 1996). A monodispersive population of bodies with a Maxwellian distribution of speeds whose rms equals the escape speed v_e from a particle's surface has the following normalized distribution function in relative speed (e.g. Binney & Tremaine 1987, Problem 7-3):

$$g(v)dv = \frac{1}{2\sqrt{\pi}v_e^3} \exp\left(-\frac{v^2}{4v_e^2}\right) v^2 dv. \tag{2.11}$$

The probability of a net accretional impact for hyperbolic encounters with $b = 0.7$ is then:

$$P[f(b = 0.7, v) \geq 0.5] = \frac{\int_{v_\circ}^{v_\star} g(v)dv}{\int_{v_\circ}^{\infty} g(v)dv}, \tag{2.12}$$

where $v_\circ$ is the initial speed corresponding to $v_\infty = 0$ from Eq. 2.3 and $v_\star$ is obtained from Eq. 2.9. For Model A, we have $v_e = 0.51$, $v_\circ = 0.22$, and $v_\star = 0.81$. Solving Eq. 2.12 numerically we find the probability of an accretional impact in this case is 26%. The probability of erosion is $1 - P(f \geq 0.5) = 74\%$.

The full accretional cross section is obtained by integrating Eq. 2.12 over all impact parameters. The net accretion probability is then the ratio of this value to

the geometrical cross section:

$$P[f(b,v) \geq 0.5] = \frac{1}{\pi \int_0^1 bdb} \pi \int_0^1 bdb \frac{\int_{v_\circ(b)}^{v_\star(b)} g(v)dv}{\int_{v_\circ(b)}^{\infty} g(v)dv}, \tag{2.13}$$

where the dependence of $v_\circ$ and $v_\star$ on b has been made explicit. Solving this equation we find the accretion probability increases to 35% only, since head-on collisions are relatively rare. Such a low value implies that this population of rubble piles would not go on to form planets but would instead grind itself down to dust. If rubble piles were common during the early stages of planet formation then perhaps collisions were more dissipative than modeled here. Alternatively, the accretion probability may be enhanced when there is a distribution of masses, a possibility that can only be tested with more simulations using impactors of varying size.

2.4.5 Comparison with Previous Work (Gravity Regime)

Using the fits to Eq. 2.9 we can estimate the value of $Q_D^\star$ (recall this is for $\epsilon_n = 0.8$; further runs are needed to determine the dependence on dissipation). Restricting ourselves to Model A with $b = 0$, we find $Q_D^\star \sim 1.9$ J kg^{-1}. This lies very close to the Holsapple (1994); Durda et al. (1998), and gravitational binding energy curves described in Love & Ahrens (1996, see in particular their Eq. (2) and Fig. 7). It lies well off the extrapolation of their SPH results. In their paper they suggest that the discrepancy between their results and analytic or experimental results may arise from: 1) the local rather than global deposition of impact energy at the surface of the target; 2) the difference between the role of gravity in self-compression and ejecta retention; and 3) the finite size of the projectile. The present work however is similar to Love & Ahrens (1996) in all these respects, which suggests the difference may be attributable to the adopted equation of state (an incompressible fluid in our case, compared with the Tillotson equation of state for granite in theirs) or a

possible reolution problem in their simulations. Note that the SPH curve plotted in Fig. 7 of Love & Ahrens (1996) was for an impact angle of $\phi = 45°$, but this would amount to less than an order of magnitude difference in $Q^{\star}_{D}$.

If the outcome truly depends solely on the gravitational binding energy (ignoring the effect of dissipation for now as this requires further study), then we would expect $Q^{\star}_{D} \propto M/R \propto R^2 \propto M^{2/3}$. From our Model A point we can estimate the constant of proportionality: $Q^{\star}_{D} \sim 1.2 \times 10^{-6} R^2 \sim 2.9 \times 10^{-9} M^{2/3}$. Further models with different M are needed to confirm this result (our Model C case failed to sample the critical dispersal regime, so we cannot use it here).

Watanabe & Miyama (1992) found $M_{\rm esc} \propto v^3$ for their low-speed, head-on SPH models (see Eq. (3.5.1) in their paper). We find a similar trend. For the $b = 0$ outcomes of Model A, a least-squares fit to the form

$$M_{\rm esc} = \alpha v^{\beta} \tag{2.14}$$

yields $\alpha = 0.06 \pm 0.02$ and $\beta = 3.2 \pm 0.1$. Evidently this relation must break down at large v, otherwise $M_{\rm esc}$ would exceed unity. Indeed our only significant outlier is for our highest v value (2.50), with $M_{\rm esc}$ in this case $\sim 20\%$ below the curve.

2.5 Conclusions

In summary, we have conducted a series of numerical simulations to create a partial map of the parameter space of rubble pile collisions at low impact speeds. The general trends can be summarized as follows: 1) larger impact angles result in more elongated, faster-spinning remnants; 2) larger impact speeds result in greater mass loss and increased mixing of the remnant; and 3) initial impactor spin can increase or reduce the rotation period and elongation of the remnant. It is also possible to create asymmetric shapes if the impactors have oppositely oriented spins. These

general trends are directly related to the total energy and angular momentum of the system. In cases where one impactor is significantly larger than the other (Model C), the smaller body generally disrupts completely on impact, sometimes removing a modest fraction of the surface of the target body and sometimes redepositing material along the remnant's $z = 0$ equator.

We have been able to generate a wide variety of remnant shapes, including spheroids, ellipsoids, contact binaries (peanut shaped and S shaped), and shapes with broken eight-fold symmetry. It proved difficult to get a significant amount of material to orbit the remnant; most debris (98%) either accreted onto the remnant or escaped from the system. We found no detached binaries of significant size, but $\sim 10\%$ of the remnants in Model A and B are contact binaries. The coefficient of restitution appears to play a more important role in collisions than in tidal disruption and can strongly affect the number and size of post-impact rubble-pile fragments. Increased particle resolution (or reduced porosity) appears to augment dissipation and give rise to more complex shapes, but the effects are modest over a factor of 5 in particle number.

We found that the impact speed needed for critical dispersal is well represented by a Gaussian function of impact parameter. Given a velocity distribution it is possible to estimate the probability of either impactor gaining or losing mass as a result of the collision. At low impact angles with equal-size impactors the remnant mass function is roughly proportional to $1/(1-m^2)$. Secondary and tertiary masses are typically finite but small, except for near-grazing encounters. Most material is ejected in a plane perpendicular to the axis of the initial motion and in some cases the debris can coalesce into smaller rubble piles.

We found $Q_D^\star \sim 2$ J kg^{-1} for the head-on collisions in Model A and that $M_{\rm esc} \propto v^{3.2}$. The former result is in rough agreement with the theoretical gravity-regime

model of Holsapple (1994). The latter relation agrees with Watanabe and Miyama (1992). We find that km-sized rubble piles in general are much easier to disperse than previously thought. This may be due in part to our conservative choice for ϵ_n (0.8). Although more work needs to be done, we believe our simulations may provide a numerical basis for parameterizing collisions during the early stage of planet formation, when the planetesimals are dynamically cool and the dominant sizes are still $\leq$ 10 km.

2.5.1 Future Work

In this study we were restricted to investigating the dependence of collision outcome primarily on impact parameter and impact speed. We also examined a few spin combinations, a model with unequal masses, and a single run with various values of the restitution coefficient. But the parameter space is truly vast. Naturally we would like to test more values for the parameters we have already investigated, particularly the coefficient of restitution and the dependence on porosity. We also need a finer grid at small speed and near-grazing separation to fully investigate the tidal regime (this would also provide better data for comparison with stellar system collision models, e.g. Davies et al. 1991). However there are many other new parameters to explore. We would like to test the effect of changing the spin-axis orientations (beyond pure prograde or retrograde). We suspect this would result in even more unusual shapes. Non-spherical impactors with a variety of sizes would improve realism and provide better estimates of $Q_D^\star$. A spectrum of particle sizes could alter the effective dissipation as smaller particles fill the voids between larger ones. Adding surface friction could lead to steeper slopes and the possibility of simulating crater formation in large targets. We plan to add a simple model for compaction to allow higher impact speeds and compare with the porous models of

Housen et al. (1999). We would like to track the movement of particles near the cores of our rubble piles and compare with the surface particles to study "scrambling" in a single rubble pile. There are so many possibilities that likely the only practical approach would be to randomly sample points in this vast parameter space to get a feel for the overall trends and then concentrate on the most interesting aspects in detail. We will carry out such work in the future.

Acknowledgements

We are indebted to E. Asphaug and D. Durda for extensive comments. We also thank W. Bottke, C. Dominik, K. Holsapple, G. Lake, C. Reschke, J. Stadel, B. Titus, and F. van den Bosch. This work was supported in part by the NASA HPCC-ESS and Intel Technology 2000 Programs and a NASA Innovative Research grant. Ray-traced images were rendered using the Persistence of Vision Raytracer (POV-Ray version 3.02).

Chapter 3

N-body simulations of planetesimal evolution: Effect of varying impactor mass ratio

This chapter has been published: Leinhardt, Z. M., Richardson, D. C. 2002, Icarus, 159, 306

ABSTRACT

We present results from direct N-body simulations of collisions between gravitational aggregates of varying size as part of a study to parameterize planetesimal growth in the solar system. We find that as the ratio of projectile to target mass departs from unity the impact angle has less effect on the outcome. At the same time, the probability of planetesimal growth increases. Conversely, for a fixed impact energy, collisions between impactors with mass ratio near unity are more dispersive than those with impactor mass ratio far from unity. We derive an expression for the accretion probability as a function of mass ratio. For an average mass ratio of 1:5

we find an accretion probability of $\sim 60\%$ over all impact parameters. We also compute the critical specific dispersal energy $Q_D^\star$ as a function of projectile size. Extrapolating to a projectile size of 1 m with a 1 km target we find $Q_D^\star = 10^3$–10^4 J kg^{-1}, in agreement with several other collision models that use fundamentally different techniques. Our model assumes that the components of each gravitational aggregate are identical and indestructible over the range of sampled impact speeds. In future work we hope to incorporate a simple fracture model to extend the range of applicable speeds and plan to implement our results into a large-scale planetesimal evolution code.

3.1 Introduction

This paper is part of a larger project to investigate planetesimal evolution in the context of solar system formation (Leinhardt et al. 2000). In order to create an accurate numerical model of solar system formation it is necessary to understand how the planetary building blocks, namely, kilometer-sized planetesimals, evolve and grow into larger bodies. In the research presented here we find conditions necessary for planetesimal growth. Our goal is to provide a recipe for planetesimal evolution that can be used in solar system formation models.

Over the past decade evidence has been mounting that small bodies several hundreds of meters to tens of kilometers in size are gravitational aggregates (Leinhardt et al. 2000; Richardson et al. 2002, and references therein). Accordingly, we model our planetesimals as 0.25 to 1 km *rubble piles*—gravitational aggregates with no tensile strength (Richardson et al. 2002). We assume that planetesimal evolution in the early solar system is dominated by slow (a few m s^{-1}) orbit-crossing collisions between planetesimals. Thus, our simulations focus on slow collisions between

rubble-pile planetesimals.

In this paper the study of planetesimal evolution is split into two experiments. In the first experiment we quantify which collisions cause planetesimal growth or erosion (Sections 3.3.1 and 3.3.2). We consider collisions between planetesimals of different masses over a range of impact parameters and speeds. We derive accretion/erosion probabilities on the basis of these experiements (Section 3.3.3). In the second experiment we determine the critical dispersal energy ($Q_D^\star$)[1] as a function of the mass ratio of the larger rubble pile to the smaller rubble pile (Sections 3.4.1 and 3.4.2). This allows us to compare our results directly with those of other groups (Section 3.4.3).

It is useful to define accretion and erosion as they pertain to this paper. *Accretion* is the permanent retention of new mass, whereas *erosion* is the permanent loss of mass. In our simulations the largest initial rubble pile is said to have accreted material if it has gained mass at the end of the simulation and to have eroded if it has lost mass. A simulation is ended when the collision event has terminated. In our simulations termination of the collision event is reached when less than 10% of the system mass is accreting or orbiting the largest post-collision remnant (Section 3.2.2).

The remainder of this paper is divided into three sections. In Section 3.2 we summarize our numerical method. In Sections 3.3 and 3.4 we present the results of the accretion/erosion and critical dispersal simulations, respectively, and we compare these results to previous experiments. In Section 3.5 we discuss the limitations of our method and plans for future work.

[1] $Q_D^\star$ is the energy per unit mass necessary to create a post-collision remnant of 50% the mass of the system (see Section 3.4.1).

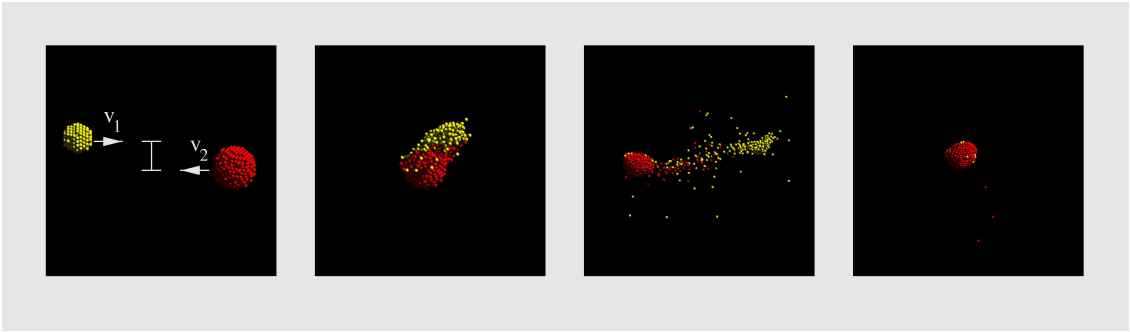

Figure 3.1: Visualization of a simulation. The yellow object is one third the mass of the red object. The impact parameter in this example is 0.75 and the initial relative speed is $1v_{\rm crit} = v_2 + v_1$.

3.2 Method

A detailed description of the numerical method used in these simulations is given in Leinhardt et al. (2000). In this section we present a summary of the numerical method and identify differences in the methodology.

3.2.1 Planetesimal Model

Typical initial conditions used in the collision simulations are illustrated in Figure 3.1. A Cartesian coordinate system is used with the origin at the center of mass. Initially, each simulation begins with two rubble piles set 2.5 Roche radii apart (in the $\pm x$ direction) to ensure that tidal forces are small. In most simulations presented here one rubble pile is significantly smaller than the other (less than half the mass of the larger rubble pile). In these cases we consider the smaller rubble pile the *projectile* (yellow rubble pile in Fig. 3.1) and the larger rubble pile the *target* (red rubble pile in Fig. 3.1). However, it should be noted that, unlike laboratory collision experiments, the projectile is a significant fraction of the target's mass. We define *mass ratio* as M_P/M_T, the mass of the projectile to the mass of the target. In all cases presented here $M_P/M_T \leq 1.0$.

Each rubble pile is built with identical spherical particles of 3.5 g cm^{-3} bulk density using hexagonal close-packed form (Leinhardt et al. 2000). The target has either $\sim$ 1000 particles (Sections 3.3.1 and 3.3.2) or $\sim$ 2000 particles (Sections 3.4.1 and 3.4.2). The projectiles have between 27 and 955 particles depending on mass ratio (1:64 to 1:1) and experiment type (accretion/erosion, Section 3.3.1, or critical dispersal, Section 3.4.1). Our rubble piles have a packing efficiency of $\sim$ 55% yielding a bulk density of $\sim$ 2 g cm^{-3}. The impact parameter b is defined at impact in units of the sum of the radii $R_P + R_T$, so $b = 0$ is a head-on collision and $b = 1$ is a glancing collision. Although the trajectories of the projectile and target will be affected by gravitational focusing, for simplicity we assume that trajectory deflection is zero, therefore,

$$b = \sin\phi, \tag{3.1}$$

where ϕ is the impact angle in the absence of deflection (between the line of centers and the x-component of the line of centers). In the simulations presented here b ranges from 0 to 0.75. For $b > 0.75$ there is little or no mass exchange between the projectile and the target (Leinhardt et al. 2000) thus, we do not investigate scenarios in this regime. Both the projectile and the target are given initial speeds between 1 and 20 m s^{-1} in the direction of the other body (Fig. 3.1) such that the center of mass is stationary. The speed of the encounter is limited on the low end by the assumption that both objects are initially on hyperbolic orbits. The largest initial speeds are limited in magnitude by requiring that they not greatly exceed the threshold for significant fracturing of rock (Leinhardt et al. 2000).

The collisional behavior of each particle is governed by normal and tangential coefficients of restitution, ϵ_n and ϵ_t, respectively. For most particle collisions ϵ_n is set to 0.8, which allows dissipation during a collision, and there is no surface friction, *ie.* $\epsilon_t = 1.0$. However, if the relative speed of two colliding particles is less than

10% of their mutual escape speed, ϵ_n is set to unity to prevent excessive bouncing (Richardson 1994).

In (Leinhardt et al. 2000) collision outcome as a function of impactor spin was explored. It was found that oppositely oriented spins reduced mass dispersal in general while aligned spins, depending on the orbital angular momentum, enhanced mass dispersal. In addition, asymmetries introduced by spin momenta often resulted in asymmetric remnant shapes. These effects are not explored in the present study which concentrates solely on the effect of varying impactor mass ratio. However, we expect that the results would be analogous to the original findings if spin were introduced, though presumably the smaller the projectile, the less effect its spin would have on the outcome. Also, an experiment varying ϵ_n was performed in Leinhardt et al. (2000), with the result that smaller values of ϵ_n (greater dissipation) gave rise to larger, more numerous, reaccreted remnants. Similarly, we would expect smaller values of ϵ_n to enhance remnant production in the present study, but do not explore this here. We would note that since energy dissipation in an inelastic collision goes as $\sim 1 - \epsilon_n^2$, the effective binding energy could be adjusted by a similar factor to take into account a different dissipation parameter. Testing this is deferred to future work.

3.2.2 Numerical Code

Our simulations were performed using a modified version of the cosmological N–body code `pkdgrav` which uses a low-order leap-frog integrator (see Richardson et al. 2000; Leinhardt et al. 2000, for details). In our implimentation of `pkdgrav` inelastic bouncing is the only allowed outcome of particle collisions; there is no merging or fracturing of particles.

The run time for our simulations was initially about 5 times the free-fall time,

$$t_f \sim \sqrt{\frac{x^3}{GM}}, \tag{3.2}$$

where x is the initial separation of the rubble piles along the x-axis, M is the combined mass $M_P + M_T$, and G is the gravitational constant. Typically, $t_f \sim 40$ h. In most cases this is sufficient time for the post-collision system to reach steady state. Simulations are run longer (by a factor of 2 to 4) if the mass accreting onto and/or orbiting the largest post-collision remnant is greater than 10% of the total mass of the system.

The time step for each run was set to $t_0 \sim 50$ s ($\simeq 10^{-5}$ year$/2\pi$) times a speed-dependent scaling factor $1/(2v+1)$, where v is in units of $v_{\rm crit}$, a convenient measure (Leinhardt et al. 2000) found by equating the initial total kinetic energy to the binding energy of a rubble pile made up of a homogeneous mixture of both the projectile and the target:

$$v_{\rm crit} = M\sqrt{\frac{6G}{5\mu R}}. \tag{3.3}$$

Here μ is the reduced mass $M_P M_T/M$ and R is the radius of a sphere of mass M, assuming the same bulk density:

$$R = (R_P^3 + R_T^3)^{1/3}. \tag{3.4}$$

The scaling term results in smaller time steps for simulations at higher speed which reduces the chance of missing a collision between particles that would otherwise result in an error condition. Since v is of order unity, t_0 is about two orders of magnitude smaller than the dynamical time $\sim 1/\sqrt{G\overline{\rho}} \sim 1$ h for an object with a bulk density $\overline{\rho} \sim 2$ g cm^{-3}. We have chosen an output frequency of 200 outputs per simulation in order to produce enough data for analysis without taking up an impractical amount of disk space.

3.2.3 Hardware

Most of the simulations were run on a local Beowulf cluster consisting of 24 machines with 1-GHz Athlon CPUs using the High Throughput Computing environment `condor` (Leinhardt et al. 2000`http://www.cs.wisc.edu/condor`) under RedHat Linux 7.1. One set of simulations was run on a Beowulf cluster of 32 machines with 1.2-GHz Athlon CPUs at the University of California Santa Cruz.

3.3 Accretion/Erosion Simulations

3.3.1 Accretion/Erosion: Method

In the first experiment we conducted four parameter-space studies, each with a different mass ratio [1:1 from Leinhardt et al. (2000), 1:3, 1:6, and 1:9]. In all of these studies the target had a mass of 8×10^{12} kg, a radius of $\sim$ 1 km, and contained 955 particles. For each mass ratio we explored the parameter space of b and v (impact parameter and speed, respectively) near the transition between accretion and erosion. The range of b for each study was from 0 to 0.75 in steps of 0.15. The range of velocity changed from study to study in order to follow the accretion/erosion transition (Fig. 3.2) which depends on the mass ratio. The initial speeds ranged from 2.1 to 3.4 m s^{-1} (1.00 to 1.60 $v_{\rm crit}$), 2.5 to 3.8 m s^{-1} (1.00 to 1.50 $v_{\rm crit}$), and 2.8 to 3.6 m s^{-1} (0.90 to 1.30 $v_{\rm crit}$) in steps of 0.10 for mass ratios 1:3, 1:6, and 1:9, respectively.

For each mass ratio we ran between 24 and 28 simulations to resolve the transition between accretion and erosion. The transition was deemed resolved at a given b if there was at least one simulation that resulted in erosion and one simulation that resulted in accretion. The collision speed at the transition, V_{trans}, was determined by

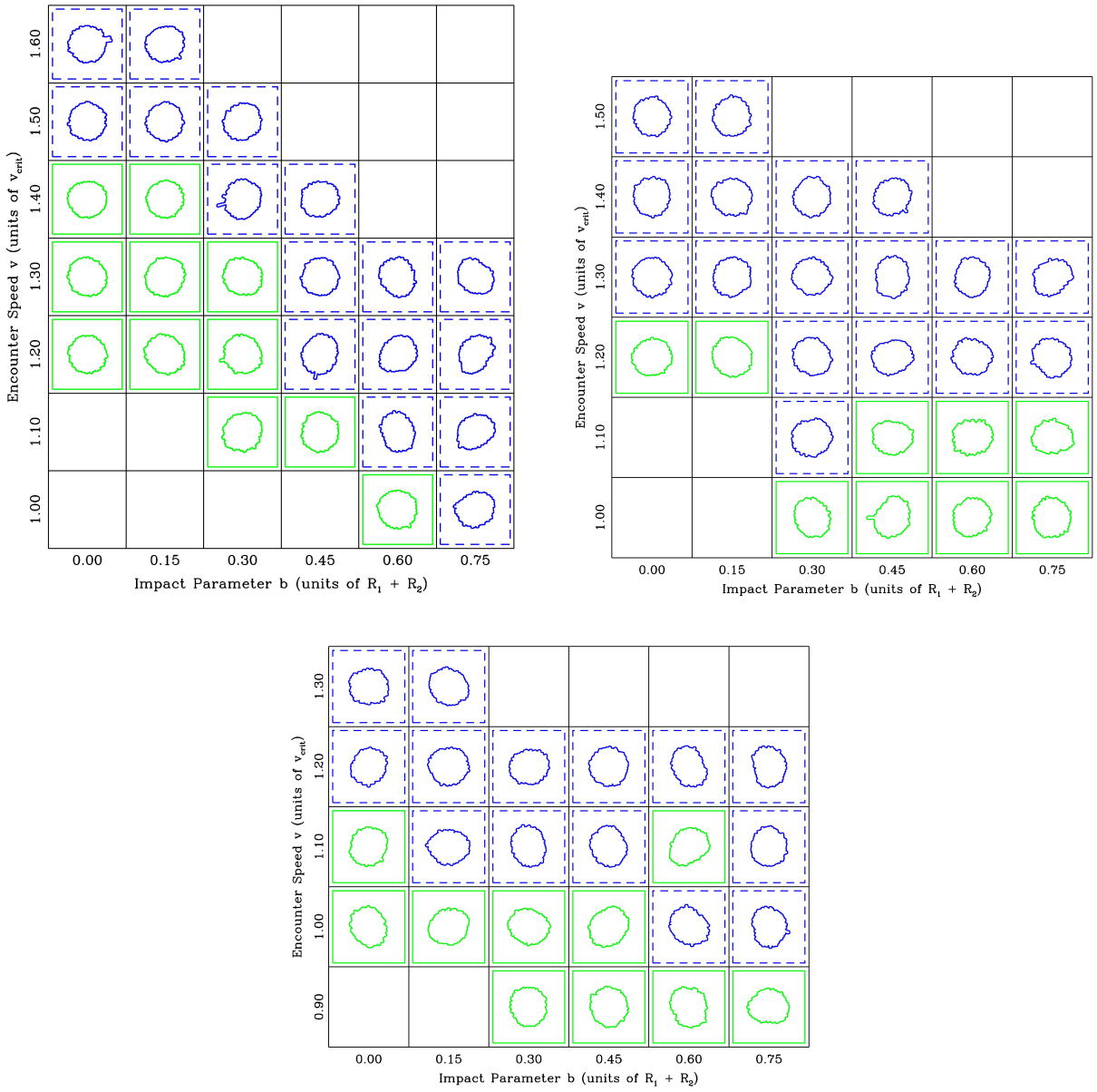

Figure 3.2: Parameter space of accretion/erosion simulations. The mass ratios are 1:3, 1:6, and 1:9 for the grids shown in the top left, top right, and bottom, respectively. The results of the 1:1 simulations are shown in Leinhardt et al. (2000). The x-axes are impact parameter b in units of the sum of the radii. The y-axes are speed in units of $v_{\rm crit}$. Each filled grid box with a cross section represents one simulation. The cross section is a slice through the largest post-collision remnant along its longest axis. The boxes with a dashed outline are erosion events. Those with solid lines are accretion events. The transition between them is the accretion/erosion curve.

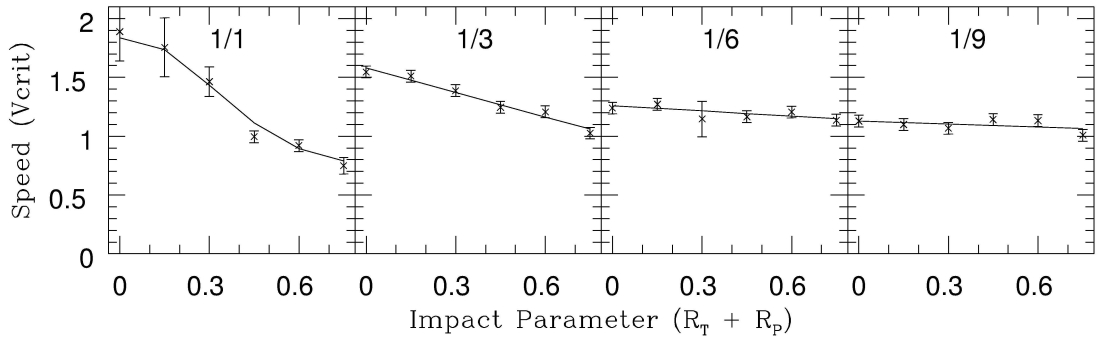

Figure 3.3: The accretion/erosion curve plotted for four different mass ratios. The y-axis is speed in units of $v_{\rm crit}$. The x-axis is impact parameter in units of the sum of the radii of the impactor and the projectile. The accretion/erosion curve on the far left (mass ratio of 1:1) was fit by a Gaussian (Leinhardt et al. 2000). All other mass ratios (1:3, 1:6, and 1:9) were fit with linear functions: the slopes are -0.79 ± 0.08, -0.14 ± 0.08, and -0.08 ± 0.08, respectively.

a linear interpolation between the minimum collision speed that resulted in erosion and the maximum collision speed that resulted in accretion.

3.3.2 Accretion/Erosion: Results

Fig. 3.2 summarizes the results of the accretion/erosion simulations (Fig. 3.1 gives snapshots of one simulation). Each grid shows the parameter space explored in b and v for a given mass ratio. The shape traced in each box is the cross section of the largest post-collision remnant along its longest axis. The objects in dashed bounded boxes have been eroded as a result of the collision. The objects in the dotted bounded boxes have accreted mass. The accretion/erosion curve $v_*(b)$ is the function that describes the transition between the erosion and accretion events. In order to resolve the accretion/erosion curve more clearly, V_{trans} was determined using the method described in Sec. 3.3.1 at each b for each mass ratio. V_{trans} is shown in Fig. 3.3 along with fits for the accretion/erosion curve. The error bars, which are

half of the difference between the speed of the simulation above the transition and the speed below the transition, approximate the error in the linear interpolation used to find V_{trans}. The fit for the accretion/erosion curve for mass ratio 1:1 (Fig. 3.3) is a Gaussian,

$$v_*(b;\xi) = \alpha \exp\left[-\frac{(b-\beta)^2}{\gamma}\right] + \delta, \tag{3.5}$$

where $\xi = M_P/M_T$ is the mass ratio (1:1 in this case), and α, β, γ, and δ are determined by a non-linear least-squares fit (Leinhardt et al. 2000). Mass ratios 1:3, 1:6, and 1:9 are well characterized by progressively shallower linear functions,

$$v_*(b;\xi_i) = m_i b + c_i, \tag{3.6}$$

where ξ_i = 1:3, 1:6, 1:9, and the slope and intercept, m_i and c_i respectively, are determined using a weighted linear least-squares fit. From Fig. 3.3 it is clear that as the mass ratio departs from unity, b becomes less and less important to the collision outcome.

3.3.3 Accretion/Erosion: Discussion

Given the data presented above we can calculate the probability that for a given mass ratio a collision will result in growth of the target. In order to do this we need to assume both an impact parameter and velocity distribution. If we assume that the velocity distribution is Maxwellian with v_{rms} equaling the escape velocity v_e from the target and the impact parameter distribution is uniform, the probability of planetesimal growth from a collision is,

$$P\left[f(b,v) \geq \frac{M_T}{M}\right] = \frac{1}{\pi \int_0^1 b\, db} \pi \int_0^1 b\, db \frac{\int_{v_0(b)}^{v_*(b)} g(v)\, dv}{\int_{v_0(b)}^{\infty} g(v)\, dv}, \tag{3.7}$$

where $f(b,v)$ is the mass fraction of the largest post-collision remnant, $v_*(b)$ is the critical dispersal fit described above (we have dropped the ξ_i to simplify the

equation), $g(v)$ is the normalized Maxwellian distribution of relative speed,

$$g(v)\,dv = \frac{1}{2\sqrt{\pi}v_e^3}\exp\left(-\frac{v^2}{4v_e^2}\right)v^2\,dv, \tag{3.8}$$

and $v_0(b)$ is the minimum initial speed, in units of $v_{\rm crit}$, for a hyperbolic encounter ($v_\infty > 0$). The expression for the speed at infinity is

$$v_\infty = \left(v^2 - \frac{2GM\cos\phi}{xv_{\rm crit}}\right)^{\frac{1}{2}}, \tag{3.9}$$

where the second term is due to gravitational focusing (x is the initial separation along the x-axis; cf. Eq. 3.2). If $v_\infty = 0$ then $v = v_0$ and

$$v_0 = \sqrt{\frac{2GM\cos\phi}{xv_{\rm crit}^2}}. \tag{3.10}$$

Substituting for ϕ from Eq. (3.1) we find v_0 as a function of b:

$$v_0(b) = \sqrt{\frac{2GM\sqrt{1-b^2}}{xv_{\rm crit}^2}}. \tag{3.11}$$

From our simulations, we find the probability that a collision between two rubble-pile planetesimals will result in the growth of one of the planetesimals is $37 \pm 3\%$, $46 \pm 1\%$, $73 \pm 1\%$, and $76 \pm 1\%$ for mass ratios of 1:1, 1:3, 1:6, and 1:9, respectively. Figure 3.4 shows the probability of an accretion event as a function of mass ratio fit with a power law of slope -0.47 ± 0.05.

Next we find the probability of an accretion event for the mean mass ratio,

$$\overline{\xi} = \frac{\int_{\xi_1}^{\xi_2}\xi\eta(\xi)\,d\xi}{\int_{\xi_1}^{\xi_2}\eta(\xi)\,d\xi} \tag{3.12}$$

where ξ_1 is the mass ratio with the largest difference in mass between projectile and target, ξ_2 is the mass ratio with the smallest difference in mass, and $\eta(\xi)$ is the distribution of ξ. We assume that the planetesimals have a power law distribution of size,

$$dN \propto R^{-\alpha}dR, \tag{3.13}$$

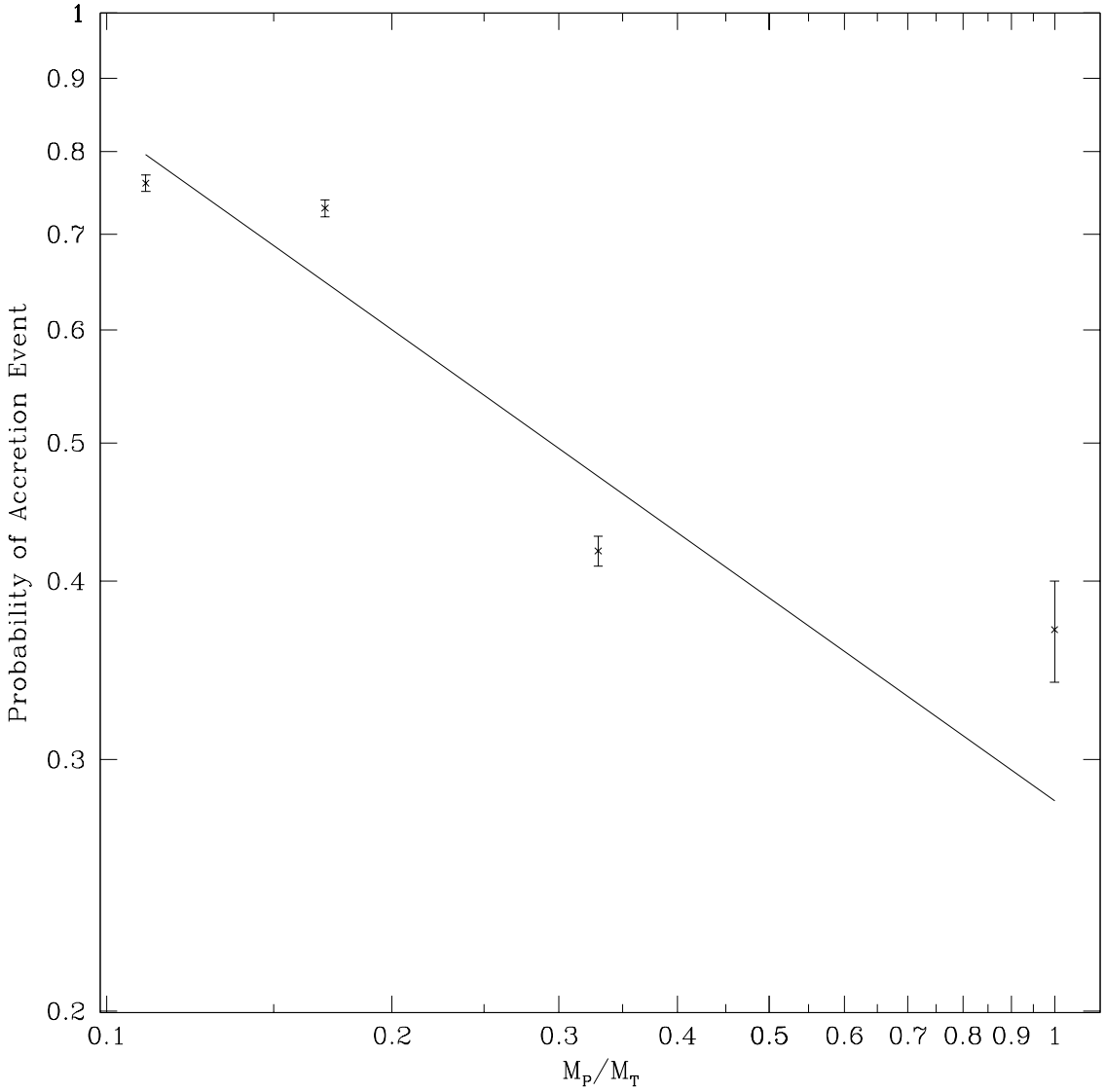

Figure 3.4: Probability of an accretion event as a function of mass fraction. The solid line is a power-law fit to the data with a slope of -0.47 ± 0.05 and intercept of -0.55 ± 0.04. The error bars represent the error in the $v_*(b)$ fits.

where N is the number of planetesimals, R is the radius of a planetesimal, and α is the power law index. Assuming constant bulk density we can express dN in terms of mass,

$$dN \propto M^{-(\alpha+2)/3} dM. \tag{3.14}$$

The mass ratio distribution function then has the same form,

$$\eta(\xi) = \xi^{-(\alpha+2)/3}. \tag{3.15}$$

Since $\eta(\xi)$ is a power law it will diverge as ξ approches zero (that is, as the mass ratio gets large) thus we define ξ_1 to be the mass ratio where the probability of an accretion event is unity. From the fit in Fig. 3.4, $\xi_1 \sim 0.06$. The upper limit $\xi = 1$ since $M_P \leq M_T$. If we take $\alpha = 3$ then $\overline{\xi} = 0.22$ or $\sim$ 1:5. From Fig. 3.4 the probability of an accretion event for $M_P/M_T = \overline{\xi}$ is then $\sim 57\%$ which means on average the target will grow. The more interesting question is how does the mean mass ratio and its corresponding probability for an accretion event change with time. This is complicated because the population changes after each collision which means that v_e and v_{rms} will also eventually change. This coupling suggests a numerical approach is needed to determine the evolution, a project we defer to future work.

It is also interesting to examine how the size of the projectile affects the efficiency of collision. Figure 3.5 shows the mass of the largest (primary) and second-largest (secondary) post-collision remnant as a function of impact energy,

$$E = \frac{\mu v^2}{2}, \tag{3.16}$$

where v is the relative speed in m s^{-1}. To identify the primary and secondary, we used the clump-finding algorithm described in Leinhardt et al. (2000). Notice that for the same impact energy the mass of the primary from the 1:3 simulations (crosses) is less than that for 1:6 (filled hexagons) which is less than that for 1:9

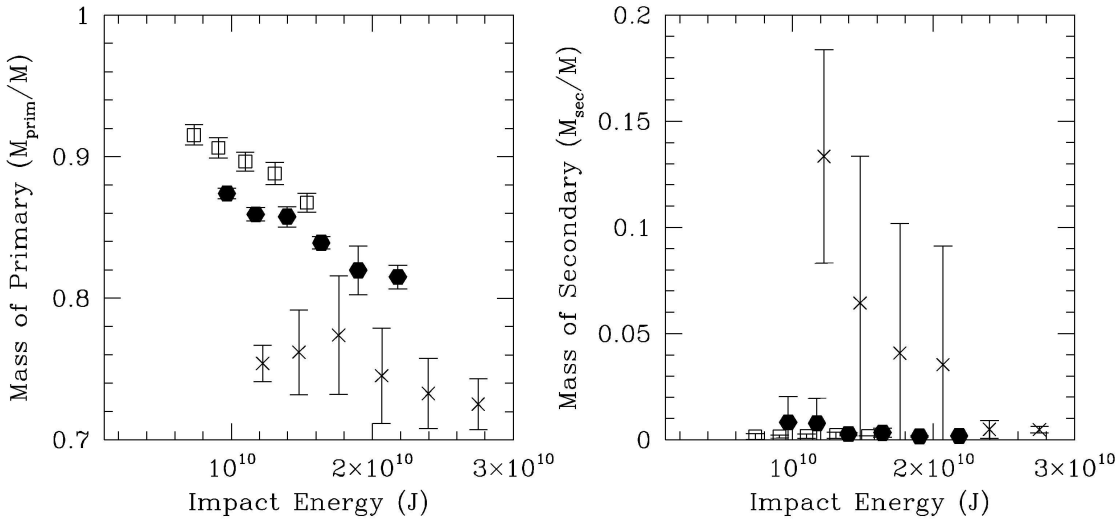

Figure 3.5: Plots of the largest and second-largest post-collision remnants as a function of impact energy. The crosses, filled hexagons, and open squares are from mass ratios 1:3, 1:6, and 1:9, respectively. All data points are averaged over b; the error bars are the rms.

(open squares). In addition, the secondary from the 1:3 simulations is significantly larger than the secondaries from the 1:6 and 1:9 simulations. These results show that for a given impact energy, a larger projectile will break a target into more pieces with a shallower distribution of mass than a small projectile. Basically the larger projectile hits more particles but imparts less energy to them than a smaller projectile. Similar results were found by Benz & Asphaug (1999) and Benz (2000).

3.4 Critical Dispersal Simulations

3.4.1 Critical Dispersal: Method

In this experiment we computed the *critical dispersal energy* ($Q_D^\star$) as a function of impactor mass ratio. $Q_D^\star$ is defined as the minimum kinetic energy per unit total mass necessary to create a post-collision remnant equal to 50% of the mass of the

total system while the rest of the mass is dispersed to infinity (Durda et al. 1998). In these simulations we kept the impact parameter fixed at $b = 0$ (head-on collision) and ran five mass-ratio models (1:8, 1:9, 1:16, 1:32, 1:64). Each model was run for at least 10 collision speeds in order to bracket $Q_D^\star$. Because the mass ratios are far from unity in these simulations we approximately doubled the resolution (number of particles) by using a target of ~ 2000 particles. However, the smallest projectiles (1:32 and 1:64 the mass of the target) still had relatively few particles and were therefore not very spherical. This meant that the orientation of the projectile had a significant effect on the collision outcome. In order to take this into account each mass-ratio system was run eight times at the same speeds with the projectile in different orientations. The critical dispersal speed V_{dis} (the speed necessary for critical dispersal) was found using linear interpolation with a similar method as that used to find V_{trans} (Section 3.3.1).

3.4.2 Critical Dispersal: Results

Figure 3.6 shows V_{dis} for each mass ratio averaged over all orientations. The error bars are the rms of the distribution of V_{dis} at any given mass ratio. The solid line is a least-squares power-law fit with a slope of -1.9 ± 0.1. Figure 3.7 shows the critical dispersal energy necessary to disperse 50% of a 1 km target. For this figure we converted the V_{dis} values to $Q_D^\star$ using

$$Q_D^\star = \frac{\mu V_{dis}^2}{2M}. \tag{3.17}$$

The error bars are propagated from the V_{dis} data. The fit is a power law of slope -1.1 ± 0.3 and intercept 3.5 ± 0.7. For a 1 m projectile this fit gives $Q_D^\star \sim 10^{2.8}$–$10^{4.2}$ J kg^{-1} which is consistent with Love & Ahrens (1996) and Benz & Asphaug (1999), for example, but disagrees with that found by Ryan & Melosh (1998).

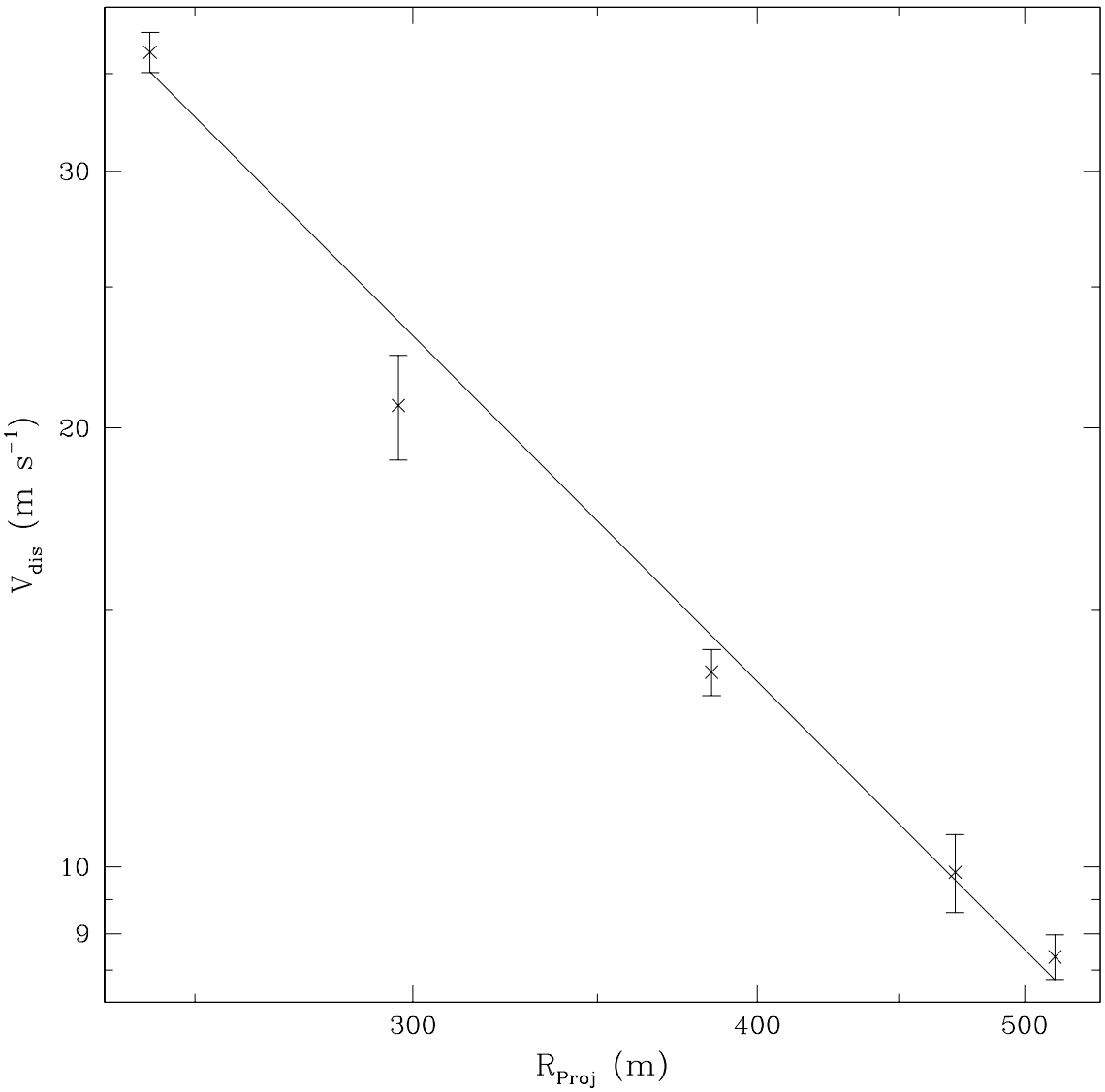

Figure 3.6: Critical dispersal speed V_{dis} as a function of projectile radius. The solid line is a power-law fit with a slope of -1.9 ± 0.1. The error bars represent spread in the critical speed as a result of orientation of the projectile.

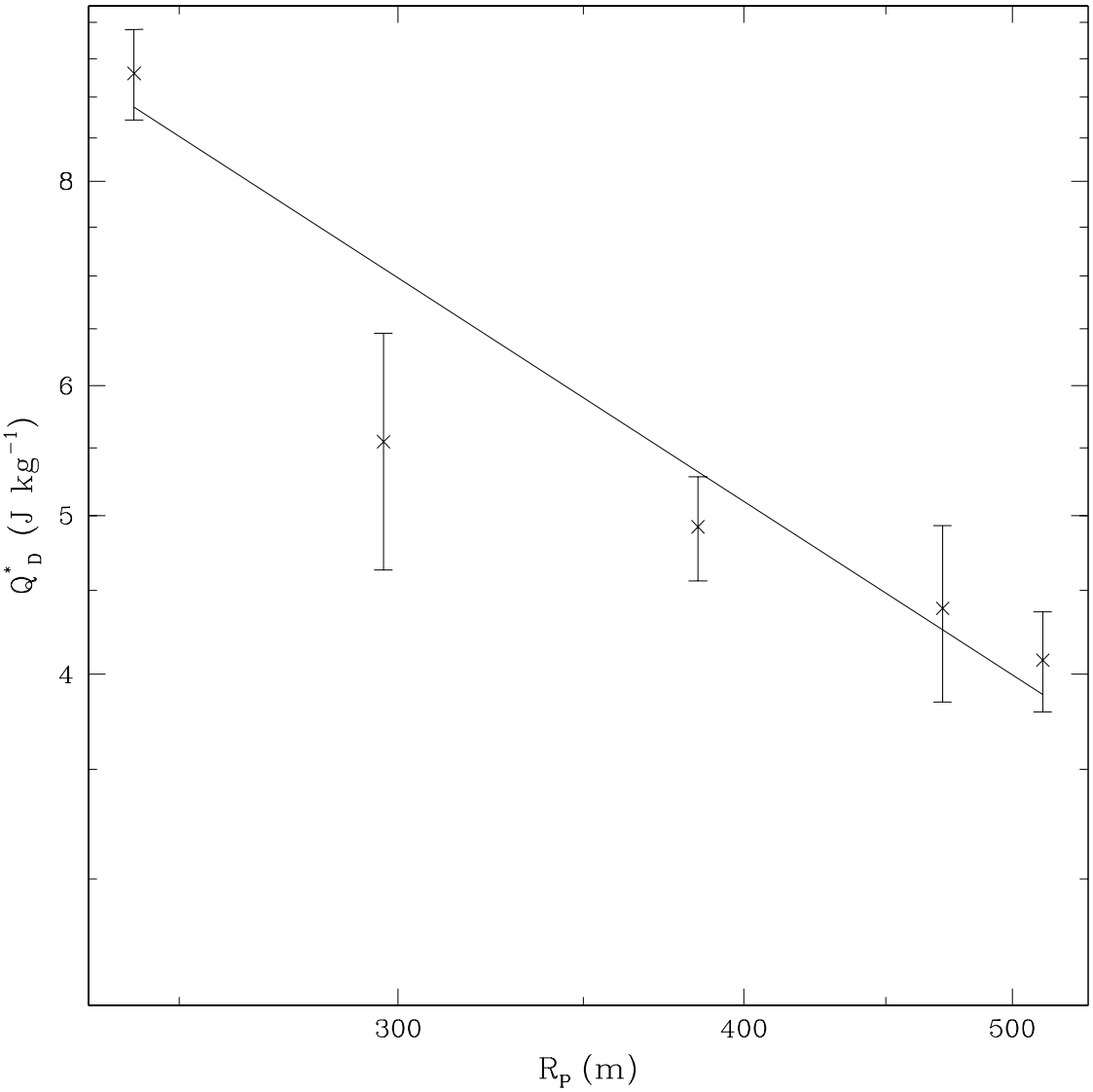

Figure 3.7: Critical disruption energy ($Q^\star_D$) as a function of projectile radius (R_P). The solid line is a power law fit with a slope of -1.1 ± 0.3 and an intercept of $\log(Q^\star_D) = 3.5 \pm 0.7$.

3.4.3 Critical Dispersal: Discussion

In many respects our simulations were conducted in a similar way to Love & Ahrens (1996). Using a 3D smoothed particle hydrocode with a Tillotson equation of state for granite without strength or fracturing they ran several simulations at various target diameters (10–1000 km) and speeds (3, 5, and 7 km s^{-1}) to find $Q^{\star}_{D}$ as a function of target diameter. For each target size they found $Q^{\star}_{D}$ by changing the projectile size and interpolating or extrapolating to find the energy necessary to produce a primary of 50% the mass of the system. They placed their data on the $Q^{\star}_{D}$ vs. D plot first constructed by Holsapple (1994) without correction for different projectile sizes. Our results from Section 3.3.2 suggest that projectile size is important in determining $Q^{\star}_{D}$. However, although the projectile size changed by two orders of magnitude, over half of their simulations used a projectile that was $< 1/100$ the mass of the target—small enough that changes in the projectile size may not be important. Benz & Asphaug (1999) found similar results to Love & Ahrens (1996) but with a more sophisticated code that included an explicit model of fracture.

Ryan & Melosh (1998) used a slightly different method. Using a 2D hydrocode with three different equations of state and including strength and fracturing, they ran a series of simulations to determine $Q^{\star}_{D}$ vs. D from the strength through the gravity regime by varying the target diameter from 10 cm–1000 km. They calibrated their code with impact experiments in the strength regime. However, they did not have a similar calibration for the gravity portion of their code. The impact speed was kept constant at 2 km s^{-1} and the projectile size was varied to find $Q^{\star}_{D}$. However, the mass ratio was consistently much more extreme than Love & Ahrens (1996), thus the change in the projectile size may not be as important.

3.5 Conclusions

In this paper we presented results from two sets of direct N-body experiments in order to investigate the collisional evolution of gravity-dominated planetesimals. In these simulations we focused on understanding the effect of impactor mass ratio on collision outcome. In our first set of simulations (Section 3.3.2) we presented four parameter space studies each with a different mass ratio. In these studies we found that as mass ratio increases the impact parameter becomes less important. There was almost no change in V_{trans} from $b = 0$ to $b = 0.75$ for mass ratio 1:6 and 1:9 (Fig. 3.3). As one might expect the probability of planetesimal growth increases steadily with decreasing $\xi = M_P/M_T$. For the mean mass ratio $\sim 1/5$ (assuming a size distribution $\propto R^{-3}$) the probability of an accretion event was $\sim 60\%$. In addition, we found that the size of the projectile is important to the collision outcome. A larger projectile is more efficient at disrupting a target than a smaller projectile for the same impact energy.

In the second series of experiments we conducted several head-on simulations at mass ratios far from unity (1:8 to 1:64) in order to find $Q^{\star}_D$ for a 1 km target. Based on a power-law fit to the above results we found $Q^{\star}_D = 10^{2.8}$–$10^{4.2}$ J kg^{-1} for a 1 m projectile.

There are two limitations to our numerical model that must be mentioned. First, because our model does not include a fracture model we are limited to relatively slow speeds. Although this does not affect our current results extensively we will need to model particle damage in order to extend the speed distribution. Second, all of the simulations presented here were done at relatively low resolution. In order to find out how the detailed mass distribution of the smaller post-collision remnants varies with speed, impact parameter, and mass ratio, higher resolution will be required.

3.5.1 Future Work

The present study had a fairly narrow focus so there are many avenues to explore in future work. We previously mentioned that collision outcome will depend on impactor spin and the choice of dissipation parameter ϵ_n—the latter effect in particular remains to be quantified. Ultimately our goal is to implement a planetesimal collision outcome "recipe" in a large-scale planetesimal evolution (planet formation) code, without having to resolve each collision in detail. To achieve this, it will be necessary to parameterize detailed collision simulations by the post-collision fragment/remnant mass and velocity distributions and derive representative distribution functions from these that can be sampled with random deviates. This would not require much more work than the present study and therefore this objective is definitely within reach.

ACKNOWLEDGEMENTS

The authors would like to thank Dr. Erik Asphaug (UCSC) for the use of his Beowulf cluster. We would also like to thank the numerical group at the University of Maryland for their helpful comments. ZML thanks John Ohlmacher and Chance Reschke for the Borg.

Chapter 4

Planetesimals to Protoplanets I: Effect of fragmentation on terrestrial planet formation

This chapter is in press: Leinhardt, Z. M. & Richardson, D. C. 2005, ApJ, in press

ABSTRACT

We present results from a dozen direct N-body simulations of terrestrial planet formation with various initial conditions. In order to increase the realism of our simulations and investigate the effect of fragmentation on protoplanetary growth, we have developed a self-consistent planetesimal collision model that includes fragmentation and accretion of debris. In our model we treat all planetesimals as gravitational aggregates so that gravity is the dominant mechanism determining the collision outcome. We compare our results to those of Kokubo & Ida (2002) in which no fragmentation is allowed—perfect merging is the only collision outcome. After 400,000 yr of integration our results are virtually indistinguishable from those

of Kokubo & Ida (2002). We find that the number and masses of protoplanets, and time required to grow a protoplanet, depends strongly on the initial conditions of the disk and is consistent with oligarchic theory. We have found that the elasticity of the collisions, which is controlled by the normal component of the coefficient of restitution, does not significantly affect planetesimal growth over a long timescale. In addition, it appears that there is a negligible amount of debris remaining at the end of oligarchic growth where "debris" is defined as particles too small to be resolved in our method, though we caution that these results are for an initial debris mass fraction of 1%. The debris component is not massive enough to alter the dynamics of the protoplanets.

4.1 Introduction

Over the past decade more than 130 Jupiter-sized extrasolar planets have been discovered. Innovations such as satellite interferometers and large ground-based surveys will allow observers to detect Earth-sized planets and increase the extra-solar planet inventory by orders of magnitude. At the same time the growing capabilities of computers make large direct simulations of solar system formation possible. Numerical simulations are essential to understanding how and under what conditions terrestrial planets form, because simulations, unlike observations, can show evolution of a single system over a large period of time. Observations, though indispensable, can provide only instantaneous information about terrestrial planets and their environment.

Due to computational limitations, previous numerical simulations have significantly simplified planetesimal collisions, the dominant growth mechanism in the protoplanetary disk. Past simulations of terrestrial planet formation have either

assumed that two colliding planetesimals merge completely (perfect merging), thus ignoring any erosion of the planetesimals, or have extrapolated the collision outcome over many orders of magnitude from a model based on laboratory impact experiments in which self-gravity is unimportant . In a real disk a range of collision circumstances are expected, from slow collisions in which most of the mass of the two colliding planetesimals ends up in the largest post-collision remnant, to fast collisions in which most of the mass ends up in small fragments. For planetesimals large enough not to be affected by nebular gas ($R > 10$ km), the most important force involved in collisions is gravity. At these sizes the material strength of the planetesimals is negligible compared to their gravitational binding energy (Holsapple 1994; Asphaug et al. 2002). The first simplification method, perfect merging, ignores the range of collision possibilities. The second simplification method, extrapolation of laboratory experiments, ignores the effect of gravity in the collision outcome. In both cases the numerical simulations produce terrestrial planet systems with eccentricities many times those of our own solar system suggesting that an important mechanism is missing (Agnor & Ward 2002; Kokubo & Ida 2002). More detailed modeling of the collisions between planetesimals is the next step toward making our numerical models of planet formation more realistic and complete.

We have developed the most realistic planetesimal collision model to date, in which gravity is the dominant mechanism in determining the collision outcome, and have incorporated it into a planet formation model. We have completed a series of high-resolution direct numerical simulations of terrestrial planet formation. We have found that fragmentation has little effect on the growth of protoplanets after several protoplanets have formed nor is there a sufficiently massive debris component remaining to affect the dynamics of the protoplanets. This suggests that either a different eccentricity damping mechanism is required, or more simulations are needed

to quantify the range of possible outcomes as a function of the initial conditions.

4.1.1 Previous Work on Planet Formation

Modern theories of terrestrial planet formation are divided into four stages (e.g. Lissauer 1993): 1) initial stage: dust condenses out of the hot gaseous disk surrounding the young star—significant growth of the grains is hindered by turbulence; 2) early stage: dust grains grow from centimeter-sized particles to kilometer-sized planetesimals by accretion—gas drag circularizes the orbits; 3) middle stage: planetesimals grow into protoplanets, again by accretion, but gravitational forces dominate—dynamical friction and the redistribution of energy via collisions causes large objects to maintain nearly circular orbits (low eccentricity and inclination) while the smaller bodies become excited (high eccentricity and inclination); 4) late stage: runaway accretion terminates due to lack of smaller material within the feeding zone of the protoplanets—the protoplanets grow into planets via long-term, long-distance, cumulative gravitational interactions. The initial and early stages of planet formation have proven the most difficult to model in a detailed way because of complex, uncertain physics. The early stage of planet formation ends when the masses of the largest planetesimals significantly exceed the mass in gas they intercept over one orbit; for planetesimal internal density $\rho \sim 2$ g cm^{-3} and gas density $\rho_g \sim 2 \times 10^{-9}$ g cm^{-3} at 1 AU this occurs at planetesimal sizes of 1-10 km in radius. The middle and late stages are much more straightforward to model directly since the planetesimals are large enough that gravity is the dominant force. Thus, most of the numerical work on planet formation has focused on these later phases of planet formation.

There are two complementary quantitative approaches that have been used to investigate the middle and late stages of planet formation: statistical methods (Green-

berg et al. 1978; Wetherill & Stewart 1989, 1993) and direct numerical methods (Lecar & Aarseth 1986; Beaugé & Aarseth 1990; Kokubo & Ida 1996, 1998, 2000, 2002; Richardson et al. 2000). The statistical method treats planetesimals as analogs to gas molecules and applies a method similar to the kinetic theory of gases to treat the evolution of planetesimals (Safronov 1969; Greenberg et al. 1978). Statistical methods are very powerful at the beginning of the middle stage of planet formation when the number of planetesimals is large and the planetesimal population can be accurately described as a thermal distribution. In addition, the statistical method can take into account any effect that can be described analytically such as gas drag, dynamical friction, and fragmentation. Using this method Wetherill & Stewart (1989) found that planetesimals go through a runaway growth phase in which the largest planetesimals grow faster than any other planetesimal due to the equipartition of energy from dynamical friction. This causes the larger planetesimals to separate from the background population of smaller planetesimals. At this point the gas dynamics treatment of the planetesimal population begins to break down because the spatial distribution is no longer homogeneous (Wetherill & Stewart 1993).

Direct numerical simulations can be integrated through the runaway growth phase and are limited only by computer capabilities, but they are much more computationally expensive. The largest direct simulation published of planet formation integrated through runaway growth uses 10^4 particles (Kokubo & Ida 2002). These direct simulations show two phases of planetesimal growth: first, runaway growth and second, oligarchic growth of protoplanets (planet embryos), in which large protoplanets grow more slowly than smaller protoplanets but all protoplanets continue growing faster than the background planetesimals. Kokubo & Ida (2002) simplify planetesimal collisions by neglecting erosion thereby assuming planetesimal colli-

sions always result in growth. This simplification may have a complex effect on the timescale of planet formation and the final outcome because the balance between growth and erosion of planetesimals is ignored. Other numerical simulations (Beaugé & Aarseth 1990) took into account fragmentation of planetesimals (in a very low resolution 2-D N-body simulation) using a semi-analytical prescription to that employed in statistical simulations (Wetherill & Stewart 1993). The effects of impact angle, spin, and the mass ratio of the colliding bodies are not taken into account in either prescription. In order to insure that our simulations correctly include as many effects of planetesimal collisions as possible we model them directly or interpolate from a table of our previous impact simulations.

The remainder of our paper is divided into four parts: §4.2 presents our numerical method in detail; §4.3 discusses our results in the context of previous numerical simulations; §4.4 summarizes our findings; §4.5 suggests future work.

4.2 Numerical Method

We use the highly efficient N-body gravity code `pkdgrav` for our simulations, which has been modified to resolve collisions realistically and account for the accretion of dust onto planetesimals. In this section we describe the numerical methods we use for the planetesimals, the planetesimal collisions, the unresolved debris, the planetesimal disk, and the integration.

4.2.1 Planetesimal Structure Model

There is significant observational evidence that small bodies—asteroids and comets—in our solar system are gravitational aggregates or "rubble piles" (objects with little or no tensile strength held together by gravity), not coherent objects (see Leinhardt et al. 2000; Richardson et al. 2002). For example, several asteroids have giant craters, low bulk densities, and almost all are rotating slower that the rubble pile break-up limit—of the 984 observed none with diameters larger than 150 m are spinning faster than this limit (Pravec et al. 2002). The evidence suggests many asteroids are likely made of loosely consolidated material and therefore contain a large fraction of void space. The voids impede the transmission of energy from collisional shocks and allow a rather weak body to survive what would otherwise be a catastrophic impact event (Ryan et al. 1991; Love & Ahrens 1996; Asphaug et al. 1998). However, it is unclear whether asteroids are a fair representation of planetesimals since asteroids have been collisionally processed during their lifetime. Nonetheless, even if planetesimals were originally coherent, the strength due to self-gravity of the planetesimal is many orders of magnitude larger than the material strength (Holsapple 1994).

Observations of comets suggest that they are also gravitational aggregates. The most impressive example of this was the tidal disruption of Comet D/Shoemaker-Levy 9 (SL9) by Jupiter in 1993. The disruption showed that SL9 was fragile, with little or no tensile strength (Asphaug & Benz 1996). Comets are much more prestine than asteroids and have not been as significantly altered by collisions as main belt asteroids.

Thus, in light of the observational evidence that a large percentage of small bodies in our solar system may be gravitational aggregates, and the understanding

that planetesimals are large enough that their gravitational strength is significantly larger than their material strength, we have chosen to model planetesimals involved in collisions in the nebular disk as "perfect" rubble piles (Richardson et al. 2005).

4.2.2 Planetesimal Collision Model

The growth of planetesimals into protoplanets is dominated by planetesimal-planetesimal collisions. The solar system formation simulations presented here use a two-phase process to determine the collision outcome. In the first phase the collision parameters—relative speed, impact parameter, and mass ratio of the projectile to the target (v, b, μ, respectively)—are used to interpolate/extrapolate the mass of the largest post-collision remnant from a collision outcome database. Spin of individual planetesimals is not a parameter in the database because the number of possible target-projectile spin vector orientations is large and thus hard to parameterize. In addition, the direction of the spin vectors of the planetesimals should be randomized. Therefore, on average the spin of the planetesimals should not affect the first-order approximation of the collision outcome (see Leinhardt et al. 2000; Leinhardt & Richardson 2002, for discussion of the effect of spin on collision outcome). The collision database consists of the results of several hundred rubble-pile planetesimal collisions over a wide range of parameter space (an extension of Leinhardt et al. 2000; Leinhardt & Richardson 2002).

Figure 4.1 shows the mass of the largest post-collision remnant, M_{lrem}, in units of system mass (M, the sum of the projectile and target mass, $M_{proj} + M_{targ}$) versus impact speed. Figure 4.2 shows the same results with M_{lrem} in units of M_{targ}. The columns represent different normal coefficients of restitution ($\mathbf{v}' = -\epsilon_n \mathbf{v}_n + \epsilon_t \mathbf{v}_t$, where the impact velocity $\mathbf{v} = \mathbf{v}_n + \mathbf{v}_t$, $\mathbf{v}_n$ is the component of the impact velocity normal to the plane of impact, $\mathbf{v}_t$ is the component tangent to the impact plane,

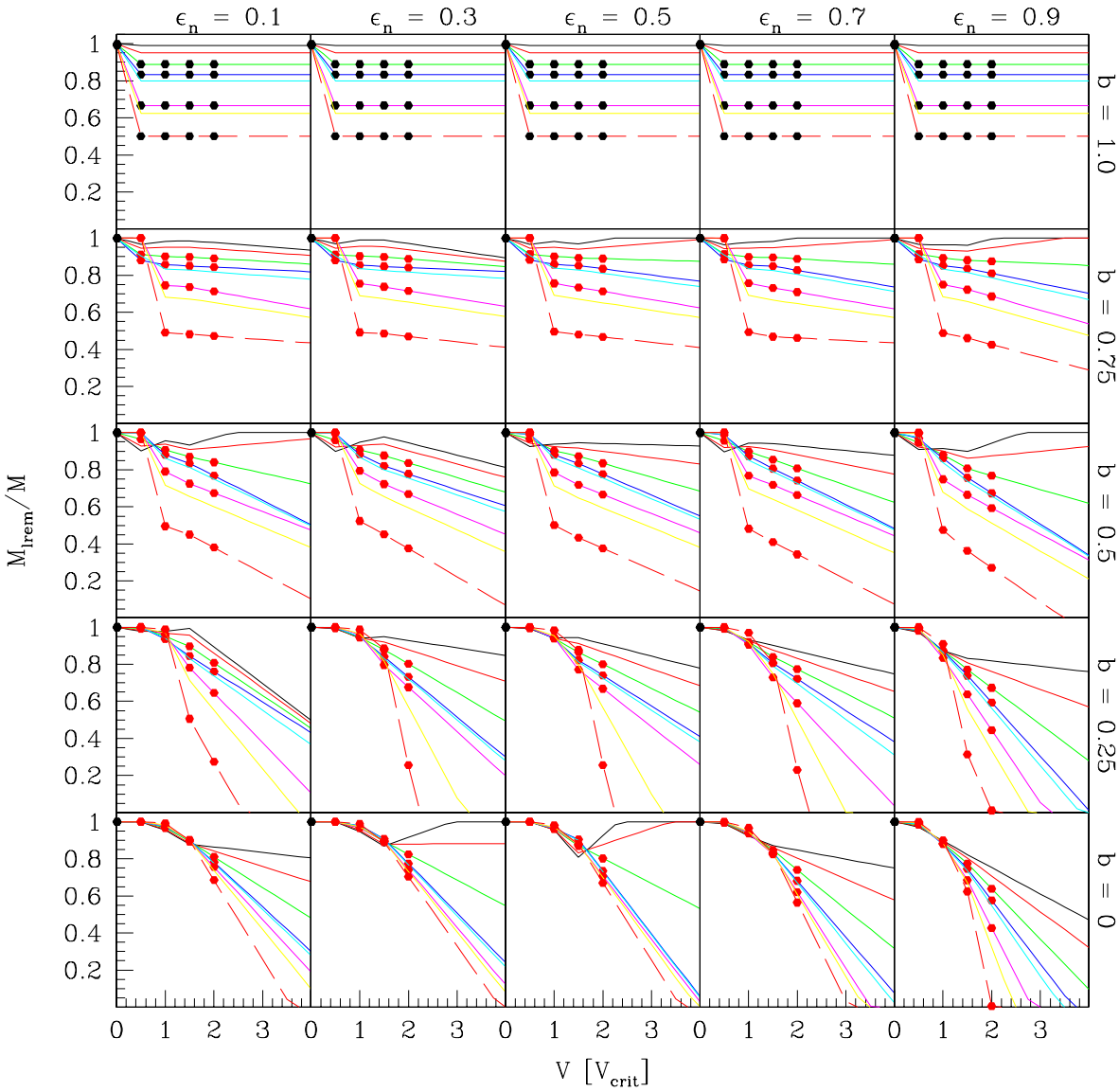

Figure 4.1: The interpolation/extrapolation table for the first phase of the collision model. Each plot in this table shows the mass of the largest post-collision remnant in units of the total system mass versus impact speed in units of v_{crit} (see text). The five columns correspond to different normal coefficients of restitution (ϵ_n). No surface friction was included in any of these simulations ($\epsilon_t \equiv 1$). The rows correspond to different impact parameters, b, in units of the sum of the radii of the impactors ($b = 0$ is a head-on collision, $b = 1$ is a glancing collision). The color lines represent various mass ratios (μ): black $\frac{1}{100}$, red $\frac{1}{20}$, green $\frac{1}{9}$, blue $\frac{1}{6}$, cyan $\frac{1}{5}$, magenta $\frac{1}{3}$, yellow $\frac{1}{2}$, red dashed $\frac{1}{1}$. The red dots are actual data from numerical simulations (similar to the one shown in Fig. 4.3). The black dots are points in the database that are fixed at theoretical limits.

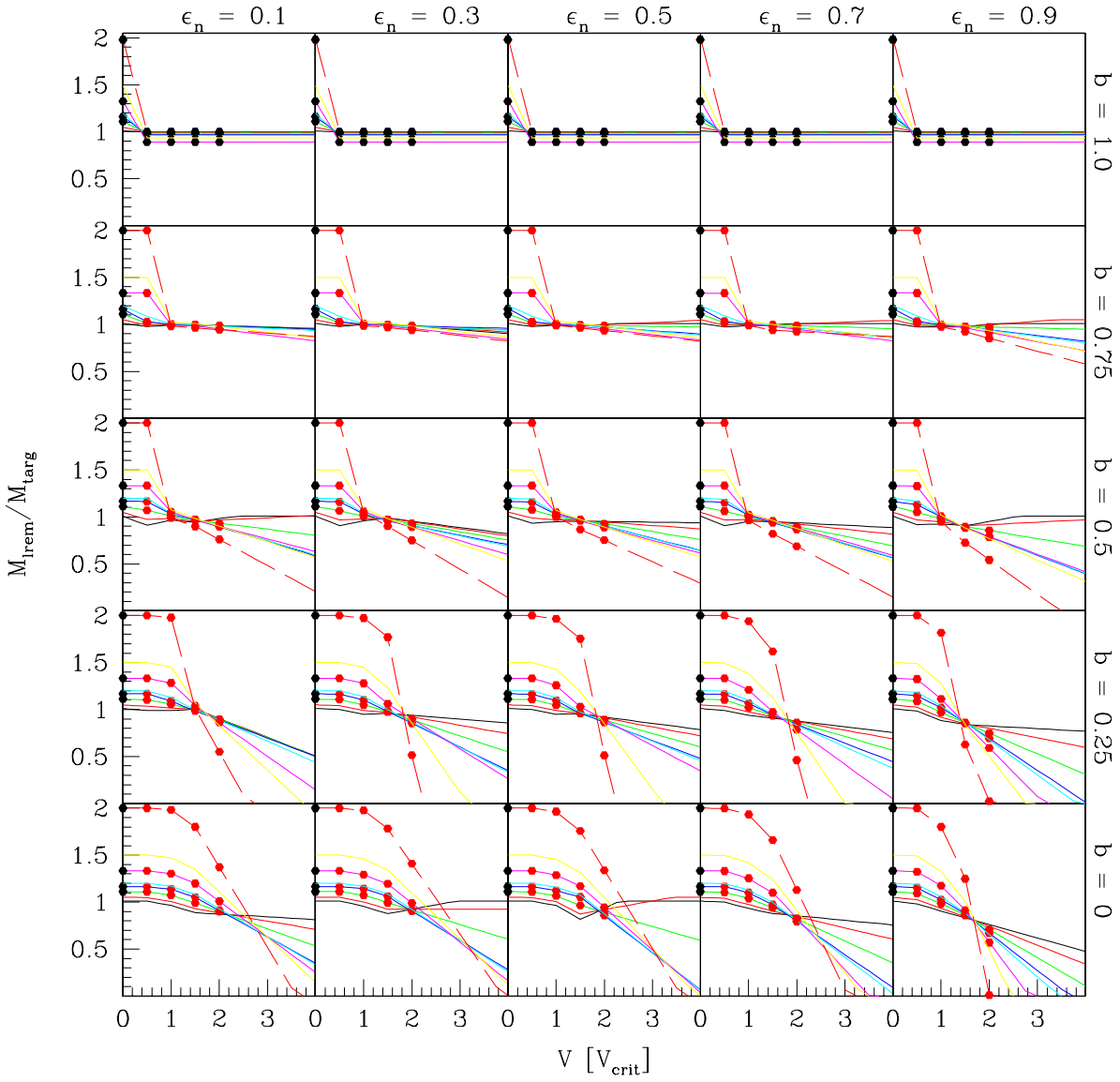

Figure 4.2: Same as Fig. 4.1 but the mass of the largest post-collision remnant is measured with respect to the initial mass of the target separating the lines of different mass ratio at low impact speed.

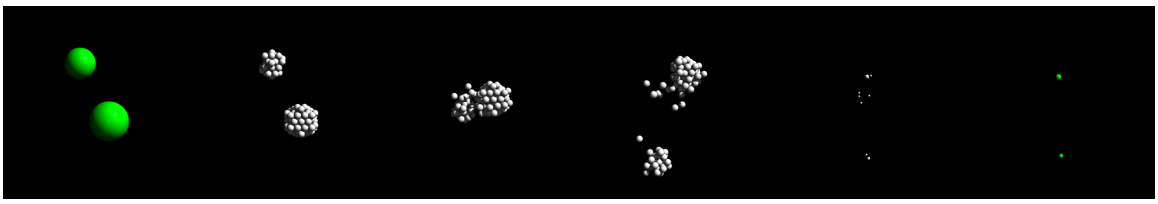

Figure 4.3: Snapshots of a collision, with time increasing to the right. The two planetesimals have a mass ratio of $\frac{1}{2}$. The impact parameter is $b = 0.89$ and the initial relative speed is 1.5 m s^{-1}. The initial rubble piles consist of a large number of hard spheres held together by their mutual gravity. Individual spheres are indestructible and bounce off one another inelastically.

and $\mathbf{v}'$ is the post-impact velocity). The rows represent various impact parameters in units of the sum of the projectile and target radii, $R_{proj} + R_{targ}$. The red points on these figures are results from actual simulations (see Figure 4.3 for an example). The black points are theoretical limits: M_{lrem} is fixed at 1 for $v = 0$ and to the mass of the target for $b = 1$. The colored lines are interpolation or extrapolation from these data points.

In order to increase the flexibility of the database the impact speed in the database is in units of

$$v_{crit} \equiv M\sqrt{\frac{6G}{5\mu_r R_V}}, \tag{4.1}$$

where $R_V \equiv (R_{proj}^3 + R_{targ}^3)^{1/3}$ is the radius of a spherical body with the combined volume of the projectile and target, assuming equal bulk density, and $\mu_r \equiv \frac{M_{proj}M_{targ}}{M}$ is the reduced mass. v_{crit} is found by equating the total kinetic energy to the gravitational binding energy ($v_{crit} = 1$ is the approximate speed necessary for catastrophic dispersal where the largest remnant is 50% of the original system mass; see Leinhardt et al. 2000). This means that when a collision is predicted the impact speed is converted into v_{crit} units which scale with binding energy allowing the same database to be used for planetesimals that have different bulk densities from those used to create the database. v_{crit} is proportional to the mutual escape speed, $v_{esc} \equiv \sqrt{\frac{2GM}{R}}$

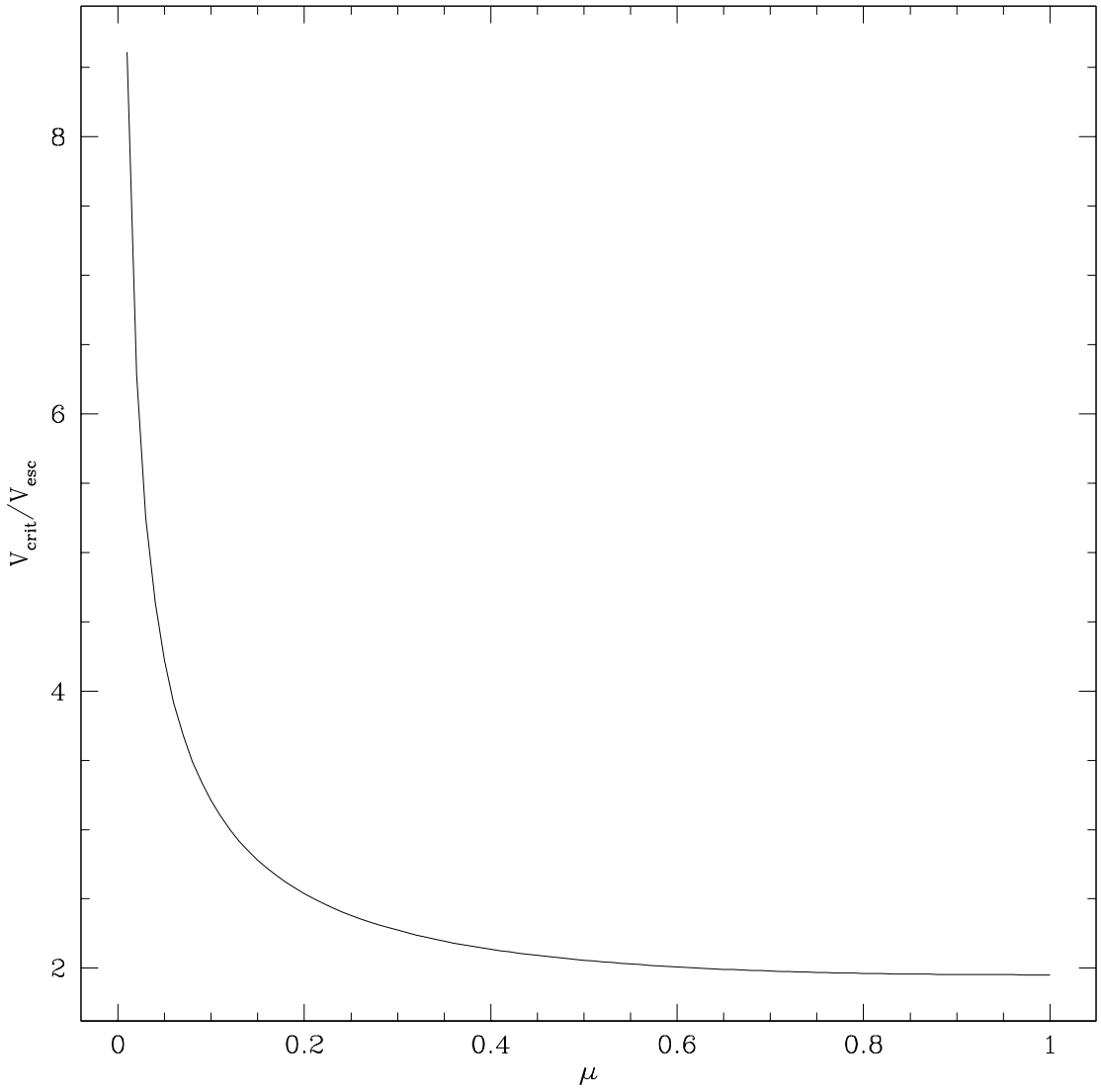

Figure 4.4: The ratio of v_{crit} to v_{esc} as a function of the mass ratio ($\mu = \frac{M_{proj}}{M_{targ}}$), assuming equal mass density.

except for cases of extreme mass ratio. Figure 4.4 shows how the ratio

$$\frac{v_{crit}}{v_{esc}} = (1+\mu)\sqrt{\frac{3}{5\mu}\frac{1+\mu^{1/3}}{(1+\mu)^{1/3}}} \tag{4.2}$$

varies with μ, the mass ratio of the projectile to the target and the mutual escape speed.

Each planetesimal used in the database was made up of a fixed number of identical self-gravitating hard spheres (Fig. 4.3). Inelastic bouncing was the only possible collision outcome between the spheres: no mergers or fragmentation of particles were allowed. All simulations used a direct numerical method (§4.2.5) to evolve the positions and velocities of the rubble pile particles under the constraints of gravity and physical collisions.

If the collision outcome from the database is one large body with a small amount of debris, this outcome is used in the simulations as the result of the planetesimal collision. In other words the colliders are replaced with the largest post-collision remnant from the database. The rest of the mass from the original planetesimals is considered unresolved debris and is tracked in a semi-analytic way by the numerical code (§4.2.3).

If the collision outcome predicted by the database consists of two or more massive remnants, the planetesimals involved in the collision, which were modeled as single particles up to this point, are substituted by actual rubble piles and the collision is then integrated directly. The total mass, bulk density, and angular momentum of the original planetesimals are preserved. The solar system formation simulation proceeds as before except for the inclusion of the rubble-pile planetesimals (see §4.2.5 for rubble-pile timestep). The number of particles in each rubble pile is between 100 and 2500 depending on the size of the target. Each particle in the rubble pile is constrained to be smaller than the initial size of the planetesimals at the beginning of the simulation (the resolution limit of the simulation; §4.2.3). Initially, a rubble

pile is created with 100 particles. If the particles in the rubble pile are larger than the resolution limit the number of particles is increased.

For ten dynamical times ($\tau_{dyn} \sim 1/\sqrt{G\rho}$, where G is the gravitational constant and ρ is the bulk density of the planetesimal), rubble-pile particles bounce when they collide with each other, allowing the collision remnants to reach equilibrium (many remnants will be gravitational aggregates; cf. Leinhardt et al. 2000; Michel et al. 2001). After ten dynamical times the rubble-pile particles merge with each other. This means that any gravitationally reaccreted remnants become single particles at this point in the simulation. After twenty dynamical times any remaining collisional debris that is smaller than the resolution limit is demoted to "unresolved debris" and is no longer followed directly; the mass is incorporated into the unresolved debris component.

4.2.3 Unresolved Debris

In order to handle debris either created by planetesimal collisions or existing initially as part of the starting conditions, we divide the planetesimal disk into a configurable number of cylindrical annuli. Any particles smaller than the resolution limit (usually taken as the radius of the starting planetesimals) are binned in the annulus at that radius. The debris particles are assumed to be on planar circular orbits. The larger planetesimals sweep up the debris as they pass through the annuli, thereby growing in mass, according to

$$M_p' = M_p + \delta m, \tag{4.3}$$

where M_p is the original mass and δm is the mass accreted given by

$$\delta m = e\pi R^2 2\pi a\rho \frac{\delta t}{P}, \tag{4.4}$$

where e is the planetesimal's eccentricity, R is its physical radius, a is the semi-major axis of its orbit, ρ is it's mass density, δt is the time since the last dust accretion update, and P is the Keplerian period corresponding to a. The accretion of the debris causes the orbits of the larger planetesimals to circularize; the accretion of the dust by the planetesimal is assumed to conserve linear momentum and thus the velocity components are updated according to

$$\begin{aligned} v'_x &= v_{kx} + \frac{M_p}{M'_p}(v_x - v_{kx}), \\ v'_y &= v_{ky} + \frac{M_p}{M'_p}(v_y - v_{ky}), \\ v'_z &= \frac{M_p}{M'_p} v_z, \end{aligned} \tag{4.5}$$

where $\mathbf{v} \equiv (v_x, v_y, v_z)$ is the initial velocity of the planetesimal, $\mathbf{v'} \equiv (v'_x, v'_y, v'_z)$ is the updated velocity, and $\mathbf{v_k}$ is the instantaneous Kepler velocity at the planetesimal's location.

The planetesimals' mass and velocity components are updated several times an orbit. The mass accreted by a planetesimal in each update is equal to the product of the mass density of debris in the annulus, the cross sectional area of the planetesimal, and the fraction of the orbit the planetesimal has traveled since the last update (Eq. 4.4).

4.2.4 Planetesimal Disk Model

In this paper we present two sets of simulations. The first set contains nine high-resolution ($N = 10^4$) simulations of various initial disk masses and surface density distributions to investigate the effect of fragmentation and environment on protoplanet formation (see §4.3). The standard model for a planetesimal disk assumes a "minimum-mass solar nebula" ($M_{solid} = 0.01\ M_{\odot}$), a surface density at 1 AU of $\Sigma_1 \sim 10$ g cm^{-2}, and a surface density distribution of solid material $\Sigma_{solid} = \Sigma_1(\frac{a}{1AU})^{-\alpha}$, with $\alpha = 1.5$. We also simulated disks that are more and less massive than the standard model ($\Sigma_1 = 100$, 1 g cm^{-2}) as well as disks where the mass is distributed more and less steeply ($\alpha = 2.5$, 0.5). Each of these simulations begins with a 1 AU-wide band of particles centered at 1 AU. The simulations are run for at least 5×10^5 yr—long enough to get through the runaway growth phase and show the formation of multiple protoplanets. The initial conditions chosen for these simulations are similar to those used by Kokubo & Ida (2002). This allows us to compare our results to theirs and thus understand how different collision outcomes affect the formation of planets in various environments.

The second set of simulations presented in this paper consist of three lower-resolution runs ($N = 4000$), each employing a different coefficient of restitution to investigate the effect of elasticity on planetesimal growth (§4.3.4). These simulations begin with a 0.085 AU band of equal-sized planetesimals at 1 AU, and a standard model surface density distribution with $\Sigma_1 = 10$ g cm^{-2} and $\alpha = 1.5$.

In all of these simulations the planetesimal collision model described in §4.2.2 is used. All planetesimals have an initial bulk density of 2 g cm^{-3}. Like Kokubo & Ida (2002) we are forced to employ a radial expansion parameter in order to complete our simulations in a reasonable amount of time. In order to stay consistent with previous

work we chose an expansion parameter of $f = 6$ for all simulations (see Kokubo & Ida 2002, for a discussion of the numerical effects of using $f > 1$)[1]. As a result of the expansion parameter all planetesimals actually have a bulk density of 0.00925 g cm^{-3}. Initially the planetesimals are given random velocities with respect to the Keplerian velocity in directions both in and out of the plane chosen from a Rayleigh distribution. The peak of the distribution is set by the escape speed from the largest starting planetesimal. The exact starting velocity distribution is not critical since the relaxation timescale of the planetesimal disk is short ($\sim 10^3$ yr) compared to the length of the simulation (Kokubo & Ida 1996). Each simulation presented here was run on our local computer cluster[2]. Each high-resolution simulation took about one month to complete while the lower-resolution simulations each took about one week on single processors.

4.2.5 Numerical Algorithm

Our numerical simulations use a modified version of `pkdgrav` (Stadel 2001; Richardson et al. 2000), a parallelized, hierarchical-tree N-body code that calculates gravity in $\mathcal{O}(N \log N)$ time. The code has been modified to include the planetesimal collision model (§4.2.2) by adding a module that uses the collision outcome database to determine whether a fully resolved collision is required. If a resolved collision is necessary this module is responsible for substituting single-particle planetesimals with rubble piles before the collision and substituting rubble piles with single particles

[1]Because of the expansion factor we do not test for excessive spin—the low density would force almost all interpolated collision outcomes to be resolved. As a result, we can say nothing about the spin of the protoplanets in the simulations presented here.

[2]The `borg` is owned and operated by the the Center for Theory and Computation (`http://www.astro.umd.edu/ctc/`) in the Department of Astronomy at the University of Maryland, College Park.

and unresolved debris after the planetesimal collision is complete.

The equations of motion in our simulations are integrated using a second-order leapfrog integrator with multi-stepping[3]. Collisions are predicted at the beginning of each position (drift) step by keeping the particle velocities fixed and extrapolating the particle positions. Once the collision outcome has been determined and new velocities (kicks) have been calculated, the post-collision particles are traced back to the start of the drift step so that they can be included in any remaining collision checks. This ensures that all collisions are detected and treated in the correct order, even if particles are involved in more than one collision during the drift step.

Since the dynamical time of a rubble pile (hours) and the orbital time of the planetesimal around the Sun ($\sim$ one year) differ by orders of magnitude, we use a two-phase timestep to increase the efficiency of our simulations. Initially all planetesimals are on the major timestep (0.01 yr). Once a collision is predicted, the timestep of the two planetesimals involved is reduced by a factor of 64. This means that gravity is calculated 64 times for the colliding particles while gravity is calculated once for the rest of the particles. All particles are drifted consistently through the major step but the colliding particles also have their kicks recalculated on the minor steps. In addition, the radius of the planetesimals is increased by a factor of 2.5 during the collision search to reduce the number of missed collisions and increase the accuracy of close approaches.

[3]For the two-body problem, without multi-stepping or collisions `pkdgrav` is symplectic—for a planetesimal at 1 AU, eccentricity of 0.01, and timestep of 0.01 years the energy error is bounded and never exceeding $2 \times 10^{-3}\%$ during an orbit and never increasing in time; for the same timestep at 0.5 AU the energy error $\leq 2 \times 10^{-2}\%$.

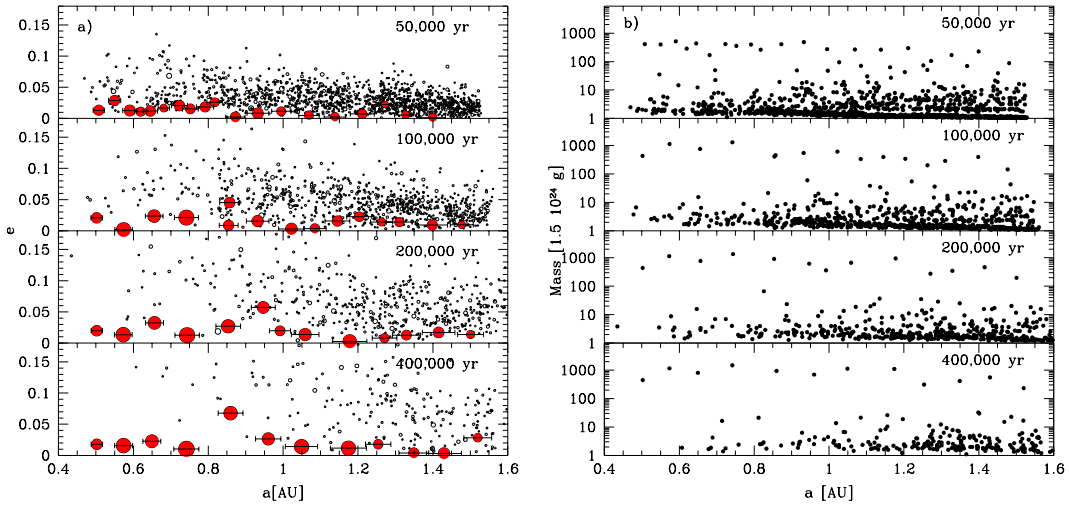

Figure 4.5: (a) Semi-major axis versus eccentricity for all particles in the standard model after 50,000, 100,000, 200,000, and 400,000 yr. The radius of each circle is proportional to the radius of the particles in the simulation. The filled circles are those protoplanets that have reached masses greater than 100 times the starting planetesimal mass (1.5×10^{24} g). The horizontal errorbars are 10 Hill radii in length. (b) Same as (a) but for semi-major axis versus mass in units of starting mass.

4.3 Results

4.3.1 Comparison with Kokubo & Ida (2002)

In this section we present a direct comparison of our global simulations of protoplanetary growth for different initial environments with that of Kokubo & Ida (2002). They used a simple perfect merging prescription to determine the collision outcome from planetesimal collisions. In order to determine the effect of our gravity dominated collision model we have completed a series of simulations similar to theirs. We begin the comparison of our results with the standard model.

4.3.1.1 The Standard Model

In our global standard model we integrated 10,000 equal-sized planetesimals for 500,000 yr. Recall, the planetesimals were placed between 0.5 and 1.5 AU with $\Sigma = \Sigma_1(\frac{a}{1AU})^{-\alpha}$, where $\Sigma_1 = 10$ g cm^{-2} and $\alpha = 3/2$. Figure 4.5 shows the location of the planetesimals and protoplanets on the semi-major axis–eccentricity and semi-major axis–mass planes at four times during the simulation. The filled circles in Figure 4.5a are those planetesimals that have grown larger than 100 times their initial mass (these are the protoplanets). The errorbars are 10 Hill radii (r_H) wide, the approximate separation expected due to orbital repulsion (Kokubo & Ida 1995), where

$$r_H \equiv \left(\frac{2M}{3M_*}\right)^{1/3} a, \tag{4.6}$$

M is the mass of the protoplanet, M_* is the mass of the central star (always $1M_\odot$), and a is the semi-major axis of the protoplanet.

In each stage of the simulation shown in Figure 4.5 we found roughly the same number of protoplanets as Kokubo & Ida (2002) but we had $\sim \frac{1}{2}$ to $\frac{2}{3}$ as many planetesimals. By 400,000 years (the last frame of Figure 4.5) we had 12 protoplanets and 236 planetesimals which is similar to Kokubo & Ida (2002) result of 12 protoplanets and 333 planetesimals. The protoplanets have relatively low eccentricity due to the dynamical friction from the planetesimals. The largest protoplanet is ~ 1500 times the initial planetesimal mass after 400,000 years. Figure 4.5b shows that the twelve protoplanets that have grown by this time are separated by at least two orders of magnitude in mass from the background planetesimal population. Note that all of the times that are used here apply to the "real" time growth of the artificially expanded planetesimals ($f = 6$). The growth timescale for uninflated planetesimals $\propto 1/f^2$ until gravitational focusing becomes effective at which point the growth

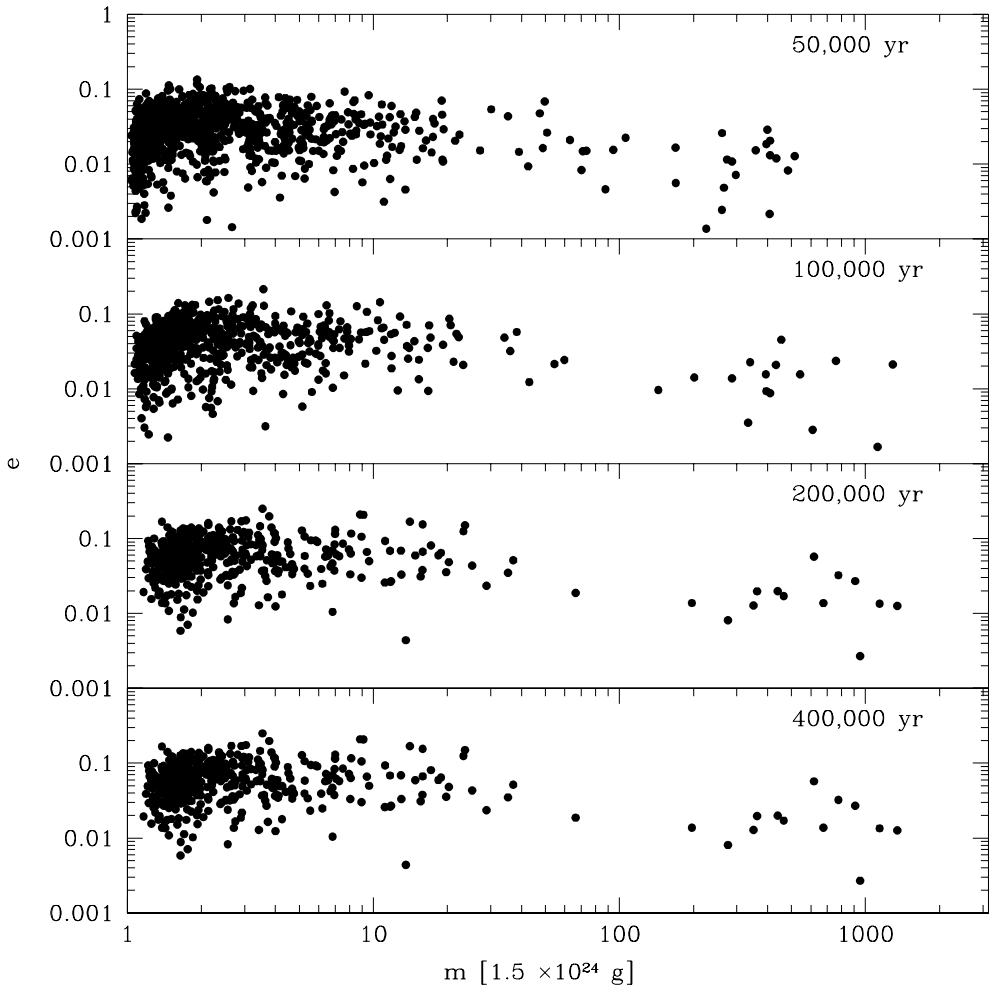

Figure 4.6: Shows the positions of all particles in the standard model simulation in mass-vs-eccentricity space at four different times during the simulation. The mass is in units of the initial mass.

timescale $\propto 1/f$ (Kokubo & Ida 1996).

Figure 4.6 shows the eccentricity of all particles in the simulation at four times during the simulation as a function of mass. By 400,000 yr the protoplanets have low eccentricity and have begun to stir up the eccentricities of the small planetesimals to $e > 0.1$ (bottom panel of Figure 4.5 and 4.6) via viscous stirring. The highest eccentricity of the planetesimals is ~ 0.27, about three times the "escape eccentricity" from the largest protoplanet (the escape speed divided by the Keplerian speed

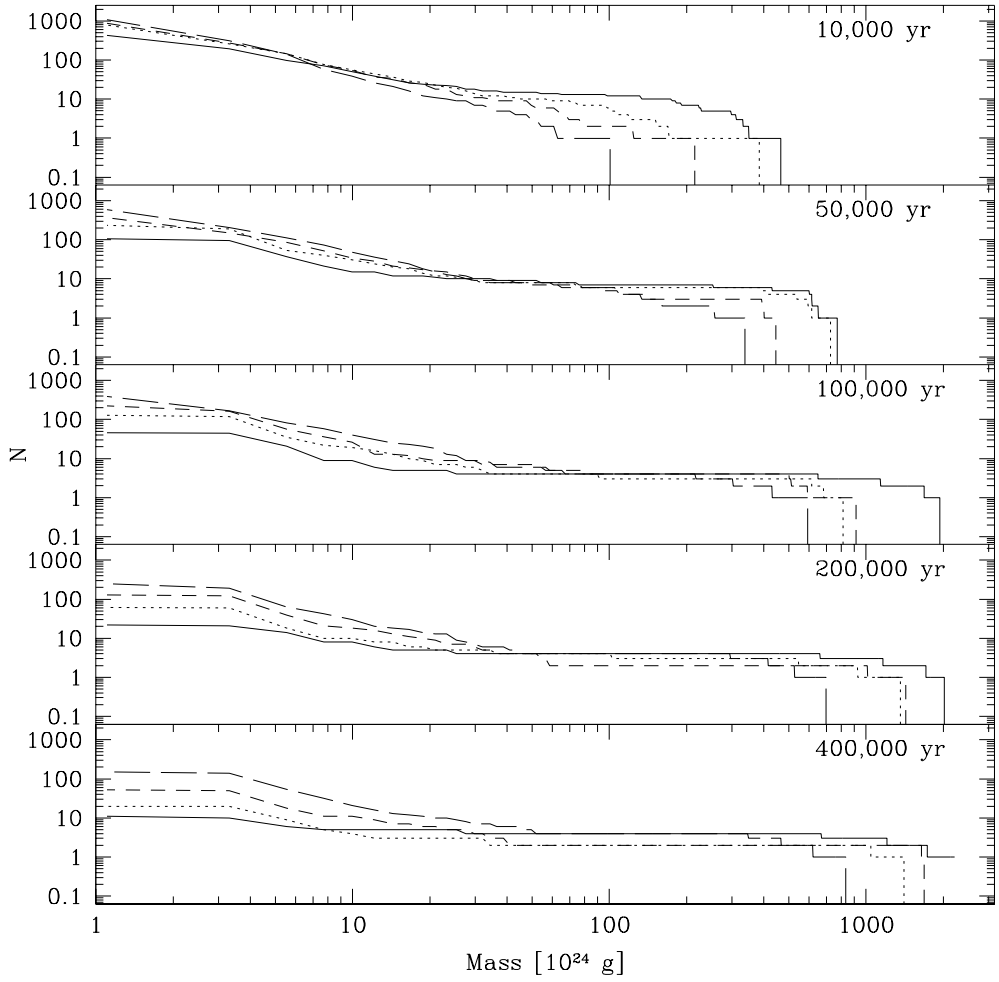

Figure 4.7: Cumulative number of particles by mass bin for five different stages in the simulation. Each line represents a different radial bin of the disk: the solid line is the innermost region of the disk ($a < 0.75$ AU), the dotted line represents particles between 0.75 AU and 1.00 AU, the short-dashed line represents particles between 1.00 AU and 1.25 AU, and the long-dashed line particles with $a > 1.25$ AU.

at the semi-major axis of the protoplanet; cf. §4.3 and Eq. 21 of Kokubo & Ida 2002). Both the escape eccentricity and the largest eccentricity of the planetesimals are consistent with the values found by Kokubo & Ida (2002).

Figure 4.7 shows the cumulative number of particles in a given mass bin at five stages of evolution in the simulation. The planetesimal disk is also divided into

four radial bins in this figure. Comparison of this plot with that of Kokubo & Ida (2002, their Fig. 4) reveals that our simulations initially evolve more quickly than theirs. By 50,000 years all regions of our disk are flattening in mass distribution. Kokubo & Ida (2002) still have quite steep distributions in the outer regions of the planetesimal disk at this point. In addition, the most massive protoplanet in the outermost radial bin is just under 200 times the initial mass at 50,000 yr; Kokubo & Ida (2002) most massive protoplanet at this time is just 50 times the initial mass. However, by 400,000 years the simulations appear virtually identical. Both show flattening of the mass distribution in all radial bins with the most massive in each radial bin clustering around 1000 times the initial mass.

Although the initial evolution is faster than that seen in Kokubo & Ida (2002) the nature of the evolution is similar. Namely, the slope of the mass distribution early in the simulations (shown in the top frame of Fig. 4.7) is characteristic of runaway growth (Kokubo & Ida 2000; Makino et al. 1998), $d \log n_c / d \log m \simeq -1.5$, where n_c is the cumulative number of planetesimals and m is the mass of the planetesimals in units of 10^{24} g. As time increases the slope becomes less steep as the number of small planetesimals drops. In Kokubo & Ida (2002) there was no source of small planetesimals to replenish the low mass end of the mass distribution. Our collision model allows for a resupply of small planetesimals via fragmentation events. However, the resupply of small planetesimals is not significant and we observe similar behavior as Kokubo & Ida (2002) in the reduction of the steep mass distribution slope as runaway growth transitions into oligarchic growth.

There are several reasons that could explain why our simulation initially evolved more quickly than that of Kokubo and Ida (2002): 1) the simulations are stochastic in nature: the initial conditions are randomized, resulting in a significant diversity of outcomes (see §4.3.4); 2) both our numerical integrator and our collision detec-

tion technique are quite different than those used by Kokubo & Ida (2002)—we use a second-order integrator and small timesteps to handle close approaches and collisions. Whereas Kokubo & Ida (2002) use a Hermite integrator with hierarchical timesteps; 3) our model includes fragmentation. We have tested the resolution of our timesteps by running the same initial condition with timesteps four times smaller. The initial evolution is consistent with the results presented here. In addition, we have investigated the effect of coefficient of restitution on the growth and evolution of protoplanets (§4.3.4). We see no obvious trend with coefficient of restitution and the mass of the most massive object. However, it is possible that the collision model does affect the early stages of planetesimal growth.

4.3.1.2 Surface Density Simulations

We have investigated the effect of varying surface density by integrating three different surface density distributions ($\Sigma_1 = 1$, 10, 100) for 500,000 y. For the simulations presented in this section, $\alpha = 3/2$. All three simulations started with 10,000 planetesimals distributed between 0.5 and 1.5 AU. The initial mass of the planetsimals was 1.5×10^{23}, 1.5×10^{24}, and 1.5×10^{25} g respectively. Figure 4.8 shows the results of the simulations in semi-major axis vs eccentricity space. The filled circles are the protoplanets that have grown larger than 100 times the initial mass of the planetesimals. The horizontal lines are ten times the Hill radius. The times have been chosen to roughly correspond to the growth timescale for the isolation mass. The isolation mass is the mass that the protoplanet reaches at the end of oligarchic growth when there are very few planetesimals left and the evolution enters the late stage.

Kokubo & Ida (2002) derived the isolation mass of a power-law mass distribution

$$M_{iso} = 0.16 \left(\frac{\tilde{b}}{10}\right)^{3/2} \left(\frac{f_{ice}\Sigma_1}{10}\right)^{3/2} \left(\frac{a}{1AU}\right)^{(3/2)(2-\alpha)} \left(\frac{M_*}{M_\odot}\right)^{-1/2} M_\oplus, \qquad (4.7)$$

[4] where $\tilde{b}$ is the separation between protoplanets in units of r_H and f_{ice} is the factor that the solid mass is increased due to the condensation of ice. In all simulations presented here, $f_{ice} = 1$. The isolation mass between 0.5 and 1 AU (assuming $\alpha = 3/2$) ranges from 3×10^{-3} – 6×10^{-3}, 9.5×10^{-2} – 2.17×10^{-1}, and 3.0 – 6.9 $M_\oplus$ for $\Sigma_1 = 1$, 10, 100 g cm^{-2}, respectively for the three simulations.

The time required to grow a protoplanet of a given mass (Kokubo & Ida 2002) is

$$t_{grow} = 1.7 \times 10^5 f^{-1} \left(\frac{\langle \tilde{e}^2 \rangle^{1/2}}{6}\right)^2 \left(\frac{M}{10^{26}\mathrm{g}}\right)^{1/3} \left(\frac{f_{ice}\Sigma_1}{10}\right)^{-1} \left(\frac{a}{1\mathrm{AU}}\right)^{\alpha+1/2} \left(\frac{M_*}{M_\odot}\right)^{-1/6} \mathrm{yr}, \tag{4.8}$$

where $f = 6$, $f_{ice} = 1$ at 1 AU is the enhancement in mass due to condensation of volatiles, $\langle \tilde{e}^2 \rangle^{1/2} \equiv \langle e^2 \rangle^{1/2}/h$ is the rms eccentricity in units of the reduced Hill radius of the protoplanet. Therefore, assuming $\langle e^2 \rangle^{1/2} = e_{esc}$ at 1 AU, it takes $\sim 2 \times 10^4$, $\sim 6 \times 10^4$, and $\sim 2 \times 10^5$ yr to grow a protoplanet with mass M_{iso} for $\Sigma_1 = 100$, 10, and 1, respectively. The protoplanets in Figure 4.8 are consistent with the Eq. 4.8 with masses 3×10^{-3} – 1.5×10^{-2}, 6×10^{-2} – 4×10^{-1}, and 1.6 – 7.8 $M_\oplus$ for $\Sigma_1 = 1$, 10, 100 g cm^{-2}, respectively.

Figure 4.9 shows protoplanet mass as a function of semimajor axis for three simulations with $\alpha = 3/2$ and $\Sigma_1 = 100$, 10, and 1 g cm^{-2}. The circles, squares, and triangles represent the protoplanets in the $\Sigma_1 = 100$, 10, and 1 simulations, respectively. The lines represent the isolation masses (Eq. 4.7) for each of the simulations. The solid line assumes a protoplanet separation of $10r_H$; the dashed line assumes $15r_H$. The simulations are consistent with the analytic predictions. The number of protoplanets decreases with increasing surface density while the protoplanet masses increase with surface density.

We have also run simulations of various mass distributions. Figure 4.10 shows the

[4]For derivation see Appendix A.

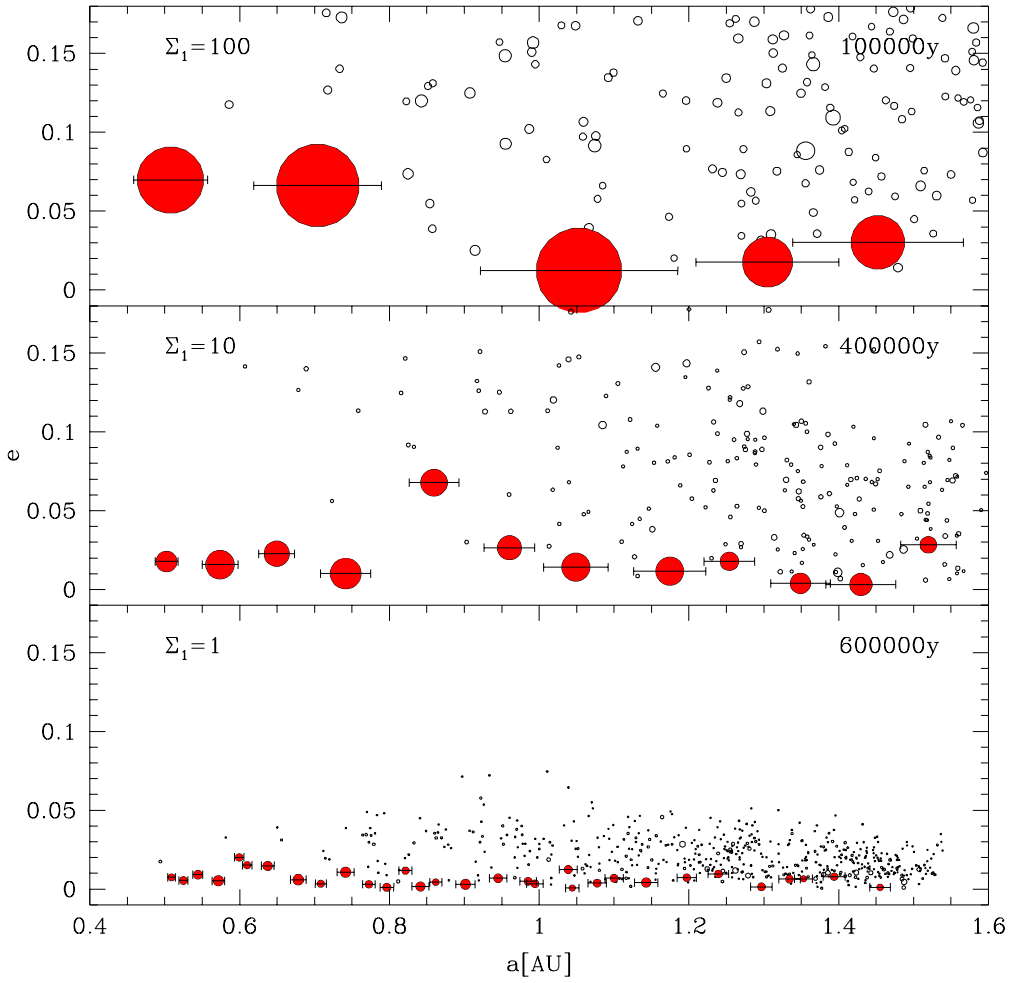

Figure 4.8: Eccentricity vs semi-major axis for three different surface density distributions: (top to bottom) $\Sigma_1 = 100$, 10, 1 g cm^{-2}. The runs shown here all have $\alpha = 3/2$. The simulations are shown at 100,000, 400,000, and 600,000 yr, respectively (a few times the time required to grow isolation masses for the respective initial surface density). The filled circles represent those protoplanets that have grown 100 times the initial planetesimal mass (1.5×10^{25}, 1.5×10^{24}, 1.5×10^{23} g, respectively).

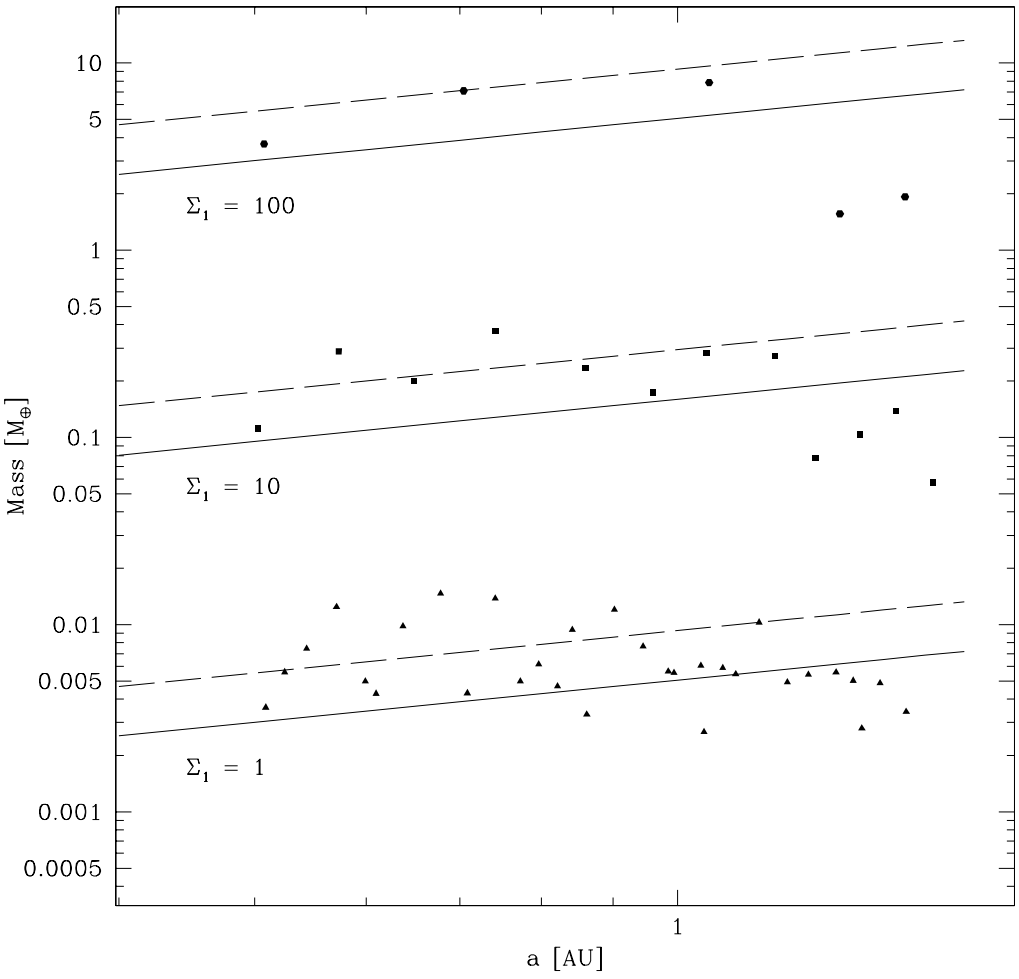

Figure 4.9: Protoplanet mass vs semi-major axis for simulations of three different surface densities (as in Figure 4.8). The circles represent protoplanets after 100,000 yr with initial $\Sigma_1 = 100$, the squares protoplanets after 400,000 yr with initial $\Sigma_1 = 10$, and the triangles protoplanets after 600,000 yr with initial $\Sigma_1 = 1$. The lines are the theoretical isolation masses for these cases. The solid lines are the isolation masses assuming protoplanet separation of $10r_H$; the dashed lines are the isolation masses assuming separations of $15r_H$.

results of three simulations with $\alpha = 1/2$, $3/2$, and $5/2$, after 400,000 yr, keeping $\Sigma_1 = 10$ g cm^{-2}. The data points represent the protoplanets. The lines again represent the isolation masses as a function of semi-major axis for each distribution. Again the data are consistent with the theoretical predictions and with the results of Kokubo and Ida (2002). Namely, the isolation mass increases with semi-major axis for $\alpha < 2$ and decreases with semi-major axis for $\alpha > 2$.

In summary, we have found that including fragmentation does affect the early evolution of protoplanets by altering the growth timescale. Our findings suggest that the collision model is important until large planetesimals/protoplanets emerge, at which point most collisions result in accretion events and the increase in velocity dispersion, eccentricity, and inclination of the background planetesimal population is dominated by the large bodies. The end results, however, are remarkably similar to those found using perfect merging.

4.3.2 Collision Rates and Statistics

Figure 4.11 shows the number of planetesimal collisions, the number of collisions that were interpolated, and the number of interpolated collisions that resulted in accretion or growth for all nine high-resolution simulations. Only $\sim 10\%$ of collisions needed to be resolved using rubble piles. Almost all of the collisions that did not require full resolution resulted in growth. These general characteristics are independent of the initial conditions. The evolution of planetesimal growth, indicated by the shape of the collision curve, is slightly dependent on the initial surface mass density and the power law of the surface density distribution. The more massive the initial disk, the earlier growth starts and the earlier runaway growth plateaus. Each disk initially has the same number of particles, so the more massive disks have larger particles with larger effective cross sections and thus the collisional evolution

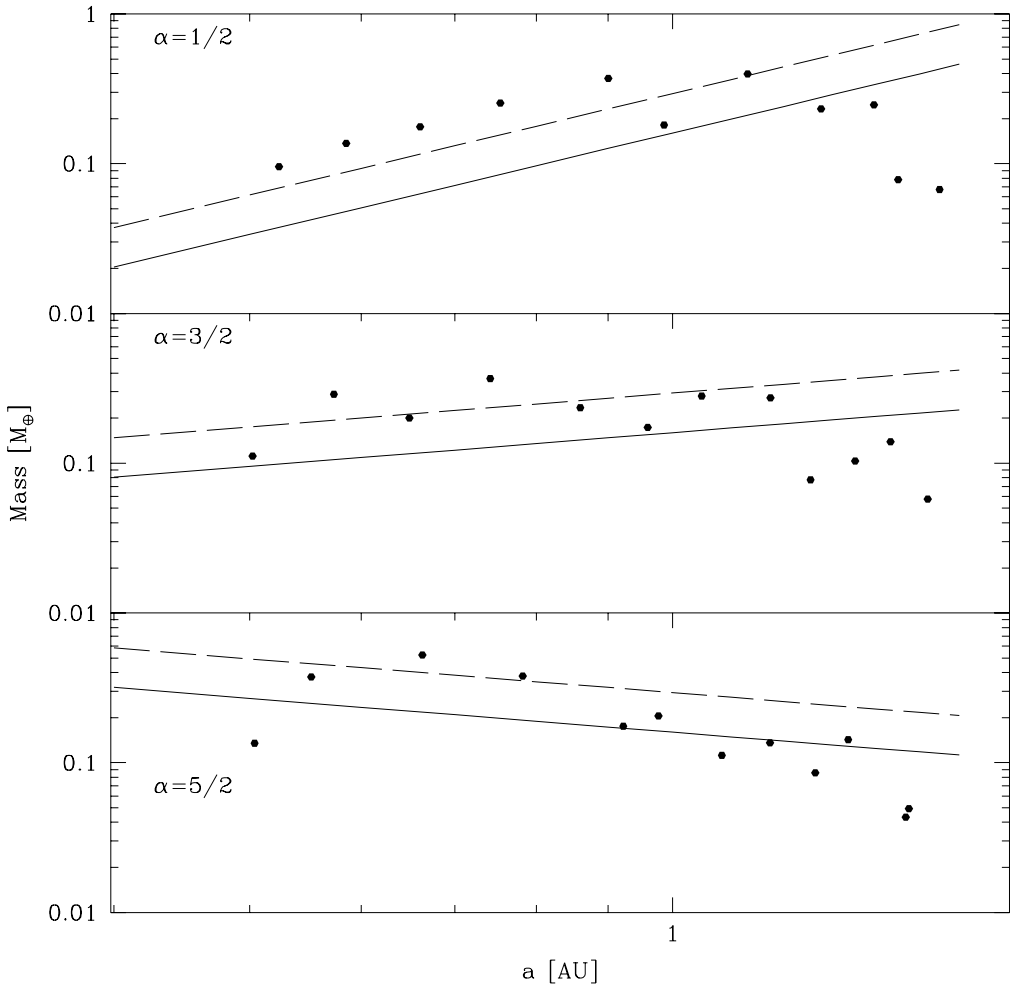

Figure 4.10: Protoplanet mass versus semi-major axis for three different initial surface density distributions with power-law exponents $\alpha = 1/2$, 3/2, and 5/2, respectively. $\Sigma_1 = 10$ g cm^{-2} for the runs shown here. The lines represent the isolation masses for protoplanet separations of $10r_H$ (solid lines) and $15r_H$ (dashed lines). The protoplanets masses are in units of Earth's mass.

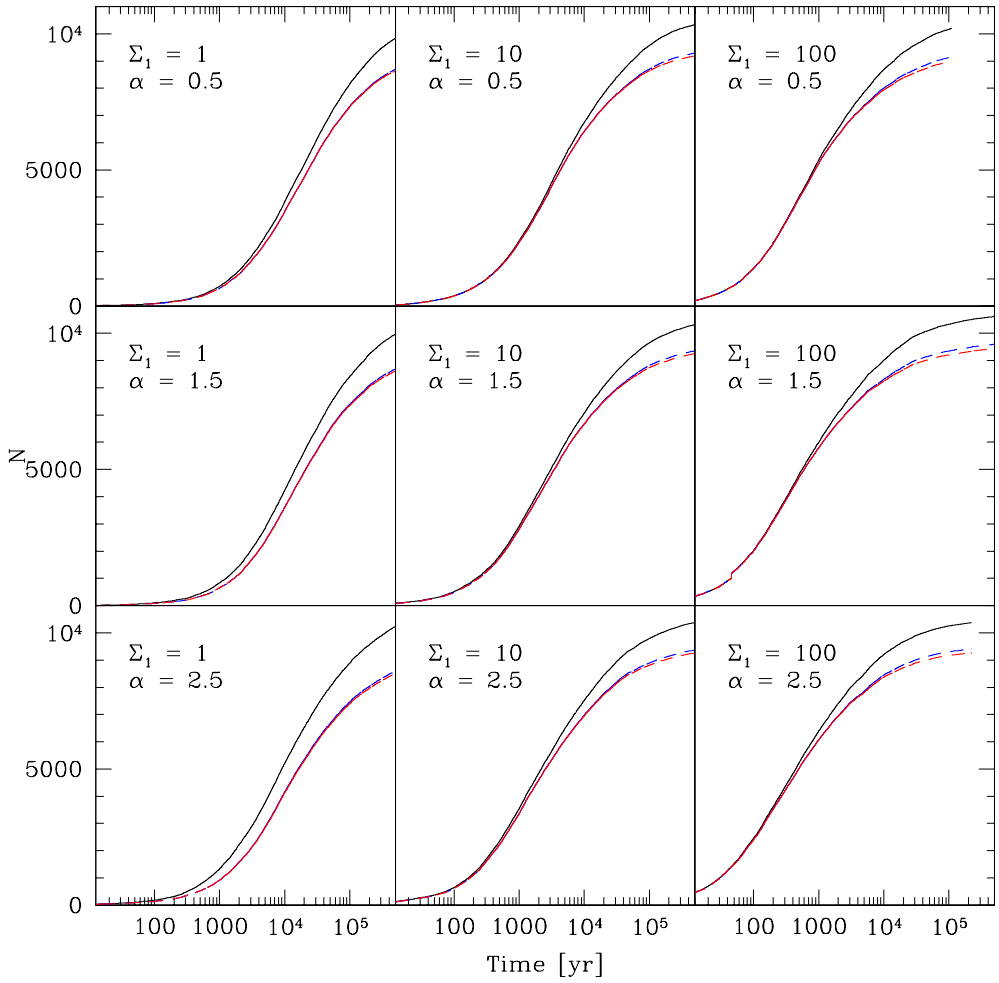

Figure 4.11: Cumulative plots of the number of collisions (solid black), the number of interpolated collisions (dashed-blue line), and the number of interpolated collisions that resulted in accretion (dotted-red line). An accretion event is a collision in which the mass of the largest post-collision remnant is larger than the mass of either colliding body.

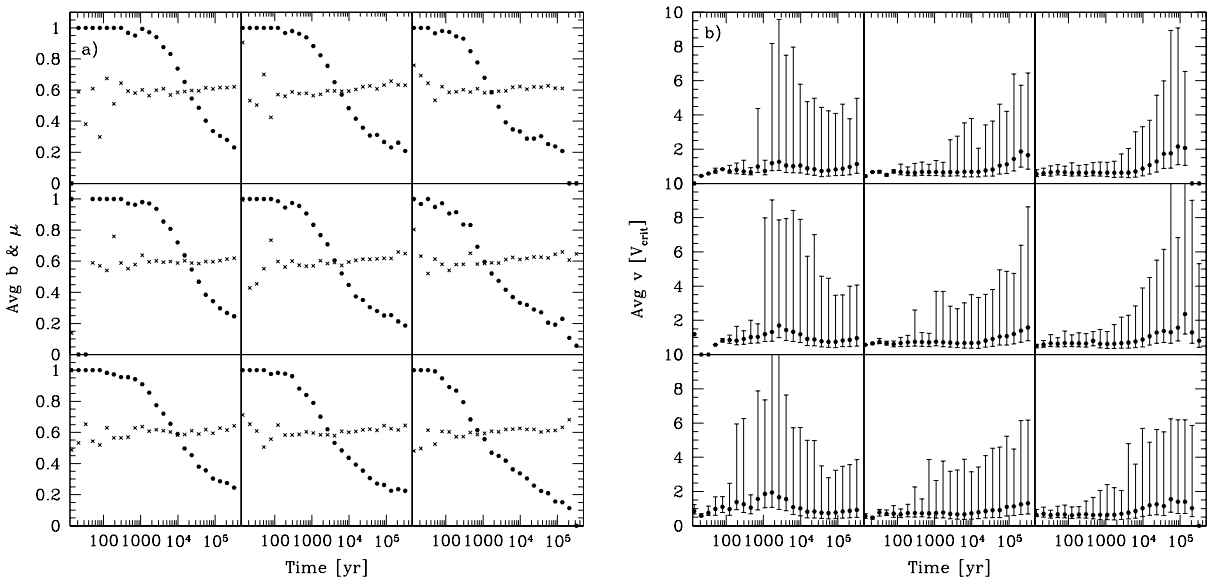

Figure 4.12: a) Average impact parameter (crosses) and mass ratio (solid dots) in logarithmic time bins. b) Average impact speed for these collisions with the same binning. The errorbars represent 50% of the most extreme values in that bin. Σ_1 and α same as Fig. 4.11.

is faster in these simulations.

Figure 4.12 shows the evolution of the collision parameters for the nine high-resolution cases. Figure 4.12a shows the time evolution of the average impact parameter and mass ratio. As a size distribution develops with the onset of runaway growth, the average impact mass ratio drops and the impact parameter remains roughly constant at ~ 0.6. The overall shape of the curves is similar for each run. Figure 4.12b shows the evolution of impact speed. The average impact speed stays low throughout the simulation, which is consistent with the $\geq 90\%$ accretion rate (Fig. 4.11). In most of the simulations the average impact speed grows as the planetesimals in the disk grow. This is due to gravitational scattering of planetesimals by the emerging protoplanets, increasing the eccentricities and the inclinations of the background planetesimals. The low-mass disk ($\Sigma_1 = 1$, first column in Figure 4.12b) shows a spike in impact speed starting at about $\sim 10^3$ yr. This is due to the

initial excitement of background planetesimals when runaway growth begins in the inner most region of the disk ~ 0.5 AU. The increase in eccentricities of background planetesimals also occurs in the more massive disks but the time resolution of the simulations is not fine enough to detect in these faster-evolving cases. Since planetesimal evolution takes the longest in $\Sigma_1 = 1$ the initial increase in impact speed at the beginning of runaway growth is detectable.

4.3.3 Unresolved Debris

As a result of our collision model, debris is created during most collisions. The debris is not followed directly (§4.2). Instead we keep track of only global properties. Figure 4.13[5] shows the evolution of debris along with the first, fifth, and tenth most massive protoplanets, and the average mass planetesimals, for comparison. All simulations were started with 1% of the total mass in planetesimals in unresolved debris. When the largest planetesimal (green line) reaches 50 to 100 times the initial mass of the planetesimals, the debris mass (black line) drops quickly. The spikes are due to individual collision events. By the end of the simulation the debris mass is at most an order of magnitude less than the initial condition and in most cases the debris mass has dropped to zero. In most simulations there is a negligible amount of debris outside the initial protoplanetary disk (dotted line).

In almost all simulations the growth of the largest object went through two phases. In the first phase—runaway growth—the slope (growth rate) for the largest object in Figure 4.13 is close to one. In all of the simulations except $\Sigma_1 = 1, \alpha = 0.5$ this slope turns over and then drops below one (but remains positive). This turnover

[5]The dust mass and mass of the planetesimals are output at a slightly different frequency in our simulations which results in noise of order of a few in the debris located outside of the original disk bounds. This offset is responsible for the small dotted line spike in $\sigma_1 = 100$, $\alpha = 5/2$.

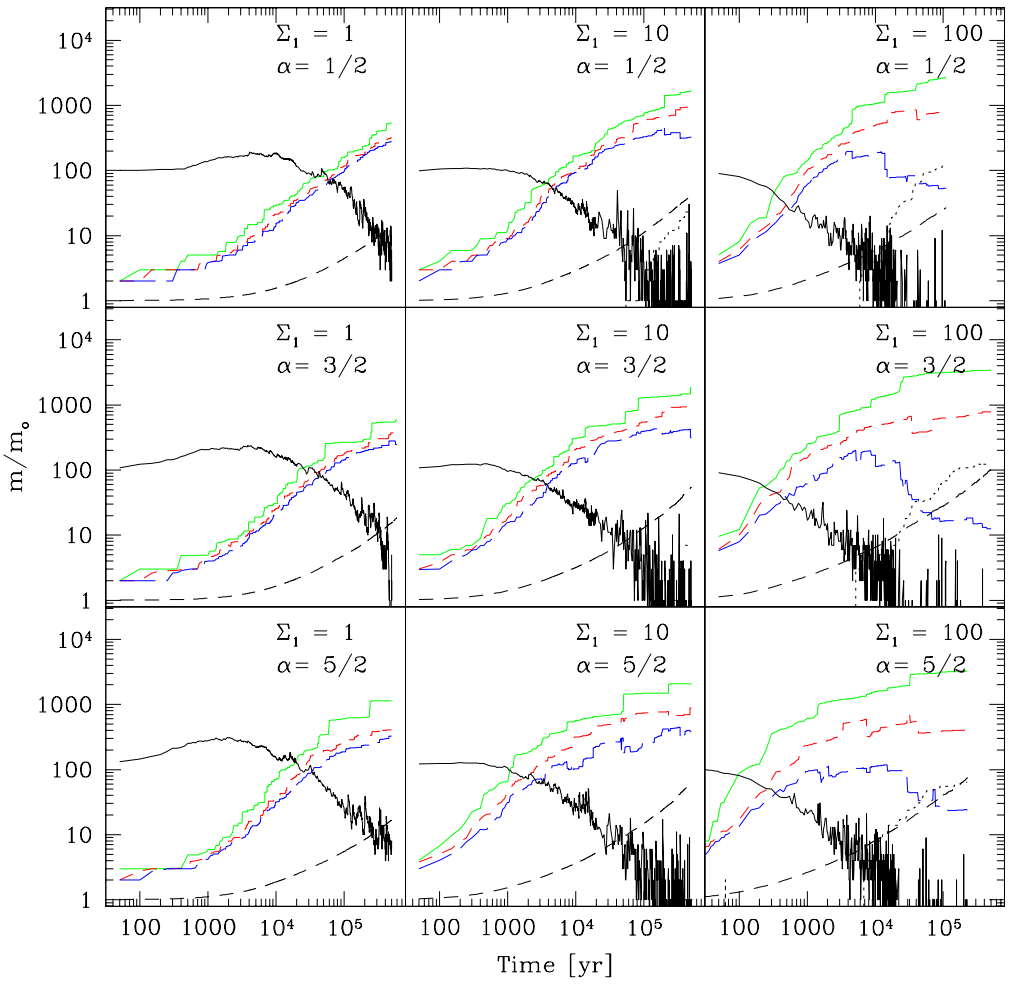

Figure 4.13: Evolution of the most massive planetesimals (solid green, dashed blue, dashed red lines), the average planetesimals (dashed black line), the debris (solid black line), and the debris located outside the original disk bounds (dotted black line). The mass of the first, fifth, and tenth instantaneous largest planetesimals are shown in green, blue, and red, respectively. All are in units of the initial planetesimal mass m_o for each simulation

is an indication of oligarchic growth. The $\Sigma_1 = 1, \alpha = 0.5$ simulation did not reach oligarchic growth. This conclusion is supported by Figures 4.14 and 4.15, which show snapshots of all high-resolution simulations at 500,000 yr (except for $\Sigma_1 = 100, \alpha = 0.5$ shown at 110,000 yr and $\Sigma_1 = 100, \alpha = 2.5$ shown at 225,000 yr) in $a-e$ and $a-m$ planes. Figure 4.14 shows that the isolation mass has been reached for all $\Sigma_1 = 10$ and 100 gm cm^{-2} simulations because the protoplanets shown in red are at least $10r_H$ from each other. Figure 4.15 shows that Σ_1 simulations have just begun forming a small distinct population of massive objects of which $\Sigma_1 = 1$, $\alpha = 0.5$ is the most undeveloped.

For the simulation with $\Sigma_1 = 100$ there is a noticeable amount of mass outside the initial protoplanetary disk by 10,000 yr. This is because the protoplanets in these simulations are more massive and viscous stirring is more effective (ie. e and i are higher for the planetesimals). As a result, some collisions between planetesimals occur outside the original protoplanetary disk. These collisions produce debris but the debris in these outer regions is not swept up. Once the amount of mass in debris outside the initial protoplanetary disk increases in cannot decrease. In these simulations it is considered "trash" and we keep track of it only to check mass conservation as a function of time. Regardless, as shown in Figure 4.16, by 500,000 yr the mass is always concentrated in a small number of massive protoplanets with a small amount of mass in planetesimals and a negligible amount of mass in debris for all runs.

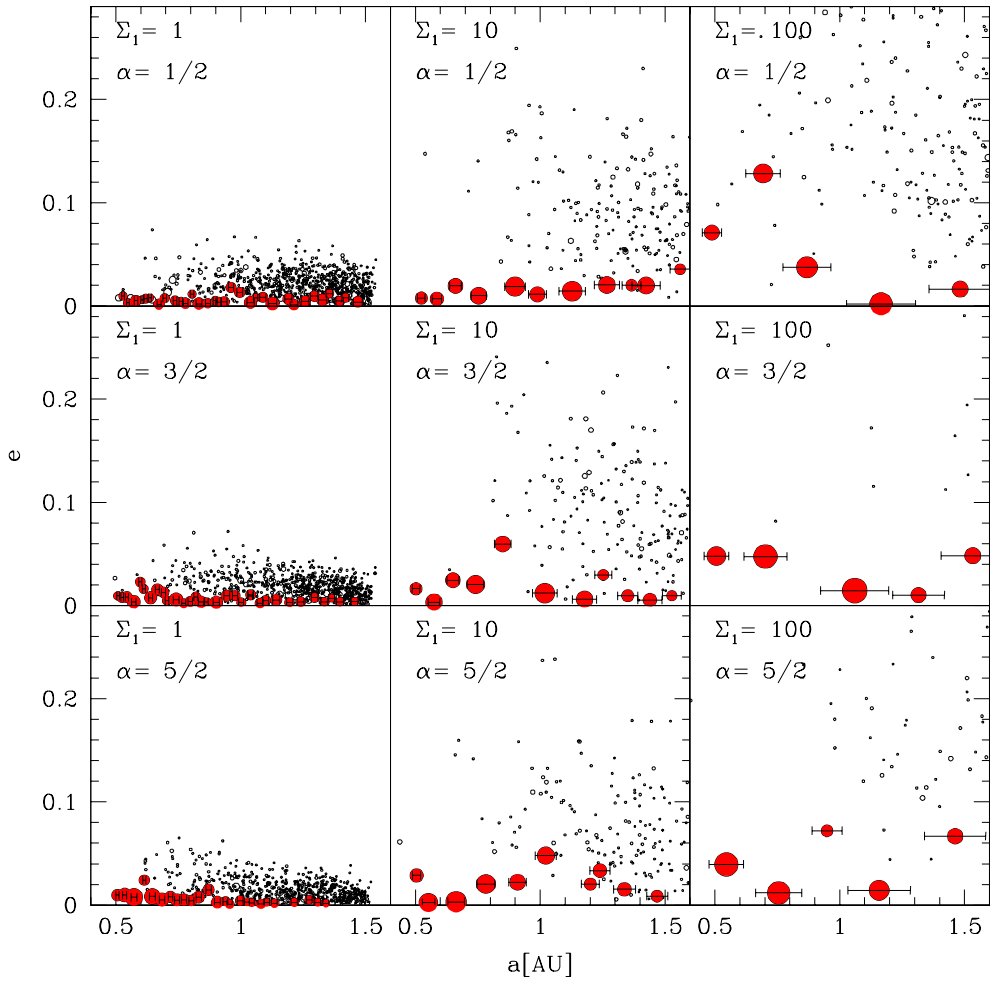

Figure 4.14: Particle locations in semi-major axis-eccentricity space for all high-resolution simulations. All simulations are shown at 500,000 yr except $\Sigma_1 = 100$, $\alpha = 1/2$ which is shown at 110,000 yr and $\Sigma_1 = 100$, $\alpha = 5/2$ which is shown at 225,000 yr. As in Fig. 4.5 and 4.8, the filled dots are protoplanets and the horizontal error bars represent $10r_H$.

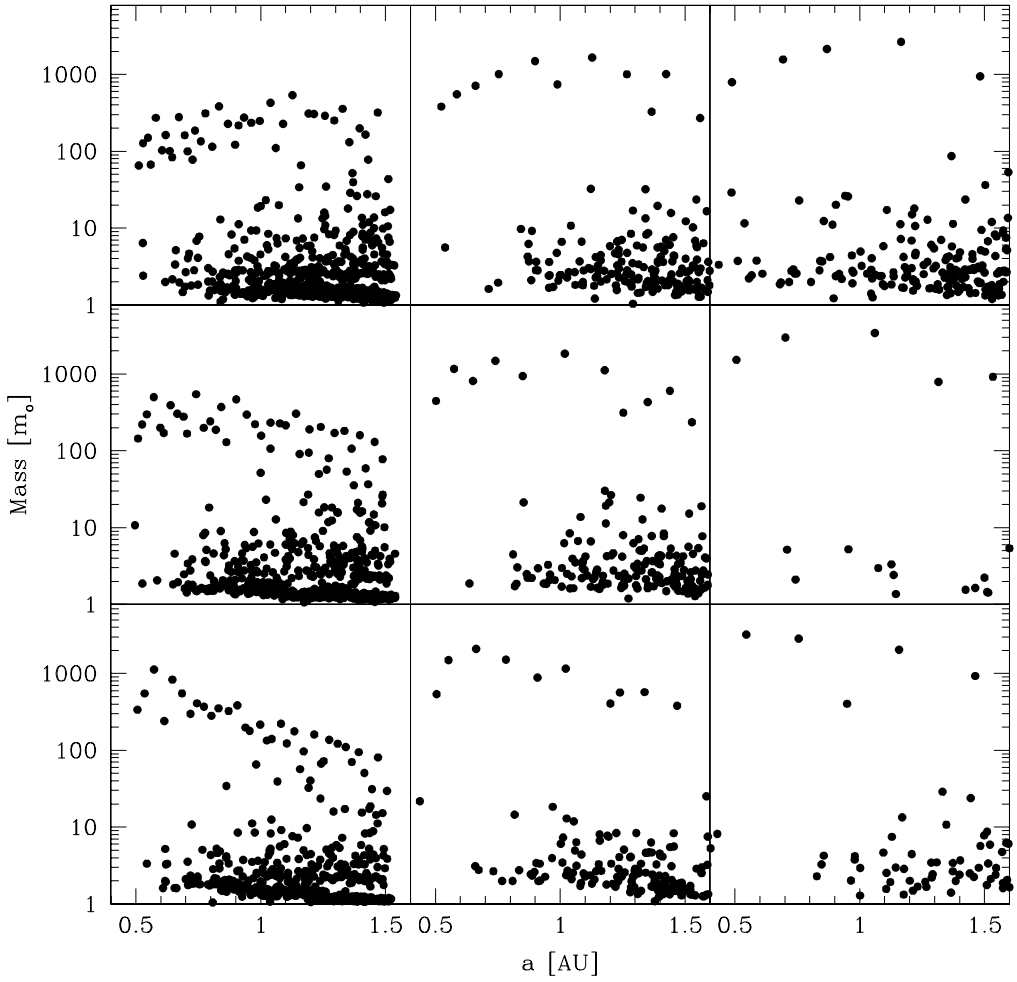

Figure 4.15: Mass of the planetesimals in units of m_o shown in Fig. 4.14.

4.3.4 Coefficient of Restitution

It is unknown what material best describes planetesimals. In order to investigate the effect of planetesimal composition in a simple way we conducted three sets of simulations using different normal coefficients of restitution ($\epsilon_n = 0.1$, 0.5, 0.8). As a control we also ran one perfect merging simulation ($\epsilon_n = 0$) with the same initial conditions and no fragmentation. These simulations are lower resolution ($N = 4000$, $m_o = 3 \times 10^{23}$), and the initial disk is significantly narrower $\Delta a/a = 0.085$ AU at

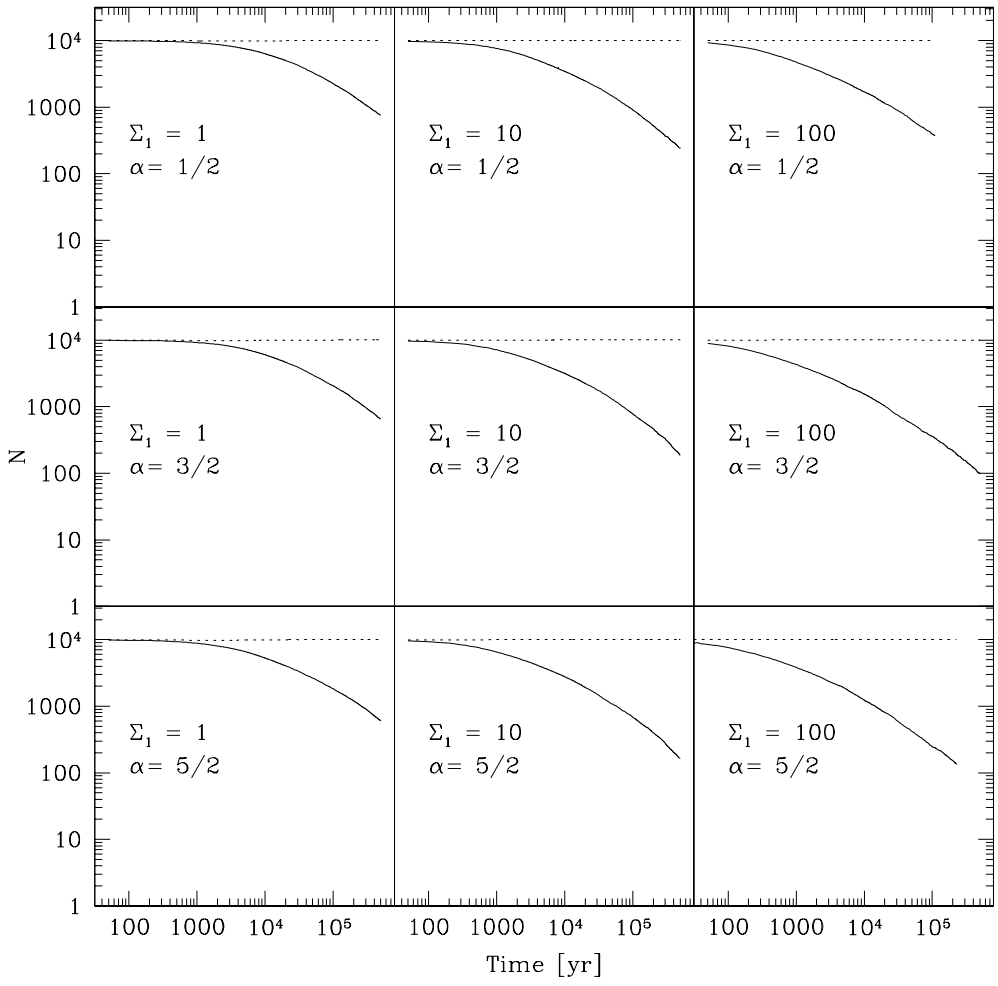

Figure 4.16: Evolution of the number of particles (ie. both planetesimals and protoplanets; solid line) and total mass in planetesimals and protoplanets in units of the initial mass (dotted line).

1 AU. As a result, planetesimals diffuse out of the initial annulus more quickly so these simulations are run for a shorter period of time, 2×10^4 yr.

Figure 4.17 shows the mass versus time in the top panels and velocity dispersion versus time in the bottom panels for these cases. The solid line in the top panels is the maximum instantaneous mass and the dashed line is the average mass. The coefficient of restitution appears to have less of an effect on the growth of the planetesimals than including a fragmentation model. The simulations with $\epsilon_n > 0$

have average planetesimal masses that are indistinguishable from each other and the range in maximum mass is also similar between cases. The average mass, maximum mass, and velocity dispersion of the $\epsilon_n = 0$ case are slightly lower than for the other simulations. However, the $\epsilon_n > 0$ simulations do show significant spread in outcome based on random changes in the initial conditions.

The second row in Figure 4.17 shows the velocity dispersion both weighted by mass (dashed line) and unweighted (solid line). The unweighted velocity dispersion is given by

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} \left|\mathbf{v_i} - \mathbf{v_{ki}}\right|^2}{N-1}}, \tag{4.9}$$

where $\mathbf{v_i}$ is the instantaneous velocity of particle i, and $\mathbf{v_{ki}}$ is the Keplerian velocity at the instantaneous position of particle i, and N is the instantaneous number of particles. The unweighted velocity dispersion follows the velocity dispersion of the most numerous particles, which in this case are the background planetesimals. The mass weighted velocity dispersion is given by,

$$\sigma_{vm} = \sqrt{\frac{\sum_{i=1}^{N} m_i \left|\mathbf{v}_i - \mathbf{v}_{ki}\right|^2}{\sum_{i=1}^{N} m_i}}, \tag{4.10}$$

where m_i is the mass of particle i. This quantity is dominated by the velocity dispersion of the more massive planetesimals. As a result σ_{vm} is less than σ and the difference between them grows as the largest planetesimals grow. The velocity dispersions also show little dependence on ϵ_n.

The energy change in the center-of-mass frame of a system of two smooth, colliding spheres is given by (Araki and Tremaine 1986)

$$\Delta E = -\frac{1}{2}\mu_r(1-\epsilon_n^2)v_n^2, \tag{4.11}$$

where μ_r is the reduced mass and v_n is the normal component of relative impact velocity. We have shown in past work that this relationship holds for rubble pile

collisions (Leinhardt et al. 2000). Thus, the lack of dependence of largest mass and velocity dispersion with coefficient of restitution suggests that collisions, though the primary growth mechanism, do not dominate the velocity field during most of protoplanetary growth. The effects of planetesimal collisions could be important in the early stage of terrestrial planet formation before the emergence of large planetesimals and protoplanets. Viscous stirring by the protoplanets dominants any change in the velocity field due to a collision by 10^4 yrs. As a result, we conclude that fragmentation is also not particularly important during most of the runaway growth and beyond.

4.4 Conclusions

We have completed a series of high-resolution direct N-body simulations of terrestrial planet formation. We have included a self-consistent planetesimal collision model in which gravity is the dominant mechanism for determining the collision outcome. We have determined that fragmentation is unimportant in determining the final outcome of protoplanet formation in a gas-free environment. The fragmentation model that we employed did affect the rate of planetesimal evolution, suggesting that fragmentation could be important in the early phase of runaway growth, but the end result, after oligarchic growth, was consistent with perfect merging simulations. We have also found that the coefficient of restitution does not affect the growth of planetesimals over a timescale of 10^4 years. The largest planetesimals dominate the growth through viscous stirring; the material properties are unimportant.

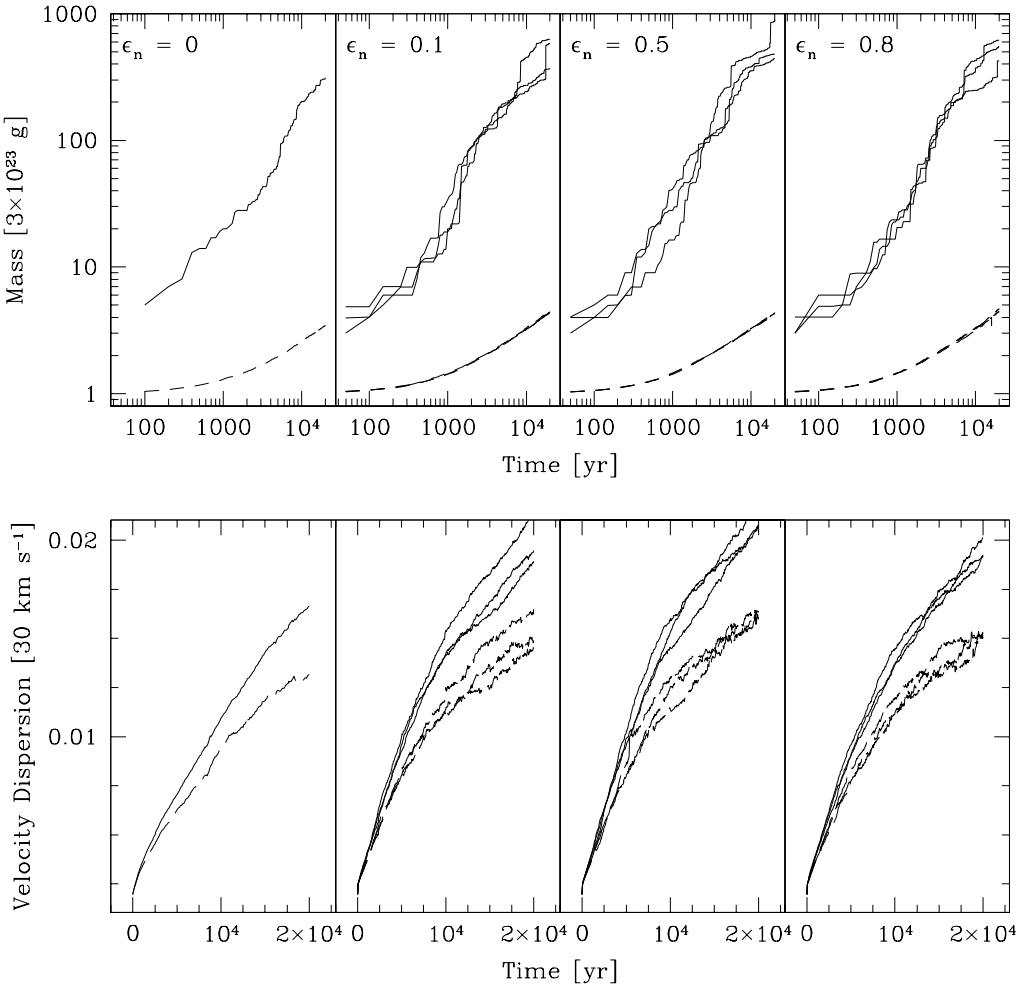

Figure 4.17: Mass as a function of time (top row) and velocity dispersion as a function of time (bottom row) for $\epsilon_n = 0$ (perfect merging), 0.1, 0.5, and 0.8. For each $\epsilon_n > 0$, three simulations were conducted. Each is represented by a separate line in the plots. The solid lines in the top row are for the largest instantaneous mass. The dashed line is the average mass. In the bottom row the solid line is the velocity dispersion and the dashed line is the velocity dispersion weighted by mass.

4.5 Future Work

It is possible that fragmentation could change the surface density distribution in a gaseous disk. Smaller fragments could migrate radially within the disk and may flatten or steepen the mass distribution and thus change the number, location, and mass of the protoplanets. We will investigate this in future work. We also did not fully investigate the debris initial condition. If the mass of debris is of the same order as the larger planetesimals they could have a significant dynamical affect on the larger planetesimals. Although this situation did not develop in any of the environments that we investigated, we always started with a debris population that was 1% the mass of the larger planetesimals. There may be some critical initial mass that is required to cause a noticeable dynamical affect. In order to study this in detail we would also need to include the effect of dynamical friction of the debris component on the planetesimals and gravitational focusing of the debris by the large planetesimals. In the simulations presented here we neglected gravitational focusing of the debris component because we made simplifying assumptions about the mass distribution and orbits of the debris—the debris was distributed smoothly though out the annulus and all debris was assumed to be on circular orbits. Adding gravitational focusing would not make the result more accurate in light of the above assumptions. In addition, there was never enough debris to significantly change the growth evolution of the protoplanets. In the next set of simulations in which the mass of the initial debris component will be increased by orders of magnitude gravitational focusing may become an important growth mechanism and must be investigated. We would also like to complete a simulation without the expansion parameter (ie. $f = 1$) to determine a true terrestrial planet formation timescale and to determine the distribution and evolution of spin states.

Acknowledgments

This material is based on work supported by NASA under Grant Nos. NGT550454 and NAG511722 issued through OSS. The authors would like to thank Dr. Kokubo for a careful and thoughtful review of this paper. ZML would also like to thank KITP at UCSB where a significant amount of work for this paper was completed.

Chapter 5

A fast method for finding bound systems in numerical simulations: results from the formation of asteroid binaries

This chapter is in press: Leinhardt, Z. M. & Richardson, D. C. 2005, Icarus, in press

ABSTRACT

We present a new code (`companion`) that identifies bound systems of particles in $\mathcal{O}(N \log N)$ time. Simple binaries consisting of pairs of mutually bound particles and complex hierarchies consisting of collections of mutually bound particles are identifiable with this code. In comparison, brute force binary search methods scale as $\mathcal{O}(N^2)$ while full hierarchy searches can be as expensive as $\mathcal{O}(N^3)$, making analysis highly inefficient for multiple data sets with $N > 10^3$. A simple test case is provided to illustrate the method. Timing tests demonstrating $\mathcal{O}(N \log N)$ scaling with the

new code on real data are presented. We apply our method to data from asteroid satellite simulations (Durda et al. 2004) and note interesting multi-particle configurations. The code is available at `http://www.astro.umd.edu/~zoe/companion/` and is distributed under the terms and conditions of the GNU Public License. The code listing is given in Appendix B.

5.1 Introduction

5.1.1 Binaries in the Solar System

Recent technical advances in observational techniques, specifically radar and adaptive optics (Merline 2001), have resulted in the detection of dozens of binaries among the Near-Earth Asteroid (NEA), Main Belt Asteroid (MBA), and Jupiter Trojan populations. Detailed lightcurve analysis (Pravec et al. 2002, 2000) and even a spacecraft flyby (Belton & Carlson 1994; Belton et al. 1995) have also revealed binaries among asteroids. Binaries also exist in the trans-Neptunian region (Margot et al. 2002, Pluto and Charon represent the most extreme example). Binary asteroids appear to represent a significant fraction of the asteroid population (10-20%) (Merline 2001). Given the relatively short lifetimes of MBAs and NEAs binaries (Bottke personal communication; Chauvineau & Farinella 1995), the solar system is evidently still dynamically active, continuously forming new binaries.

5.1.2 Numerical Simulations of Binary Formation

The diverse physical and dynamical properties of binary asteroids suggest at least three distinct formation mechanisms: 1) NEA binaries may have been formed by tidal disruption during close planetary encounters (Richardson 2001, Walsh et al. in prep.) or by fission following thermal spin-up (Margot et al. 2002); 2) MBA binaries may result from highly energetic collisions between asteroids, including family forming events (e. g. Michel et al. 2001; Durda et al. 2004); and 3) Kuiper Belt binaries, given their large separations, may have formed through three-body encounters or capture following energy loss via dynamical friction from small bodies (Goldreich et al. 2002; Weidenschilling 2002).

Simulations of MBA binary formation (e. g. Michel et al. 2001; Durda et al. 2004) are suitable for modest computer clusters, employing $N \sim 10^5$ particles with a two-phase numerical method. In the first phase, the physical collision and resulting fracture propagation is modeled with smoothed particle hydrodynamics (SPH) code (Benz & Asphaug 1999). In the second phase, after the collisional shock has propagated through the bodies, the simulation is switched to an N-body code (Richardson et al. 2000) which follows the debris for timescales of days under the mutual effects of gravity. Typical projectile and target asteroids are between 1 and 100 km in size, with impact speeds of kilometers per second. All simulations of this type require $N \sim 10^5$ in order to accurately model the collisional shock wave. Both groups found that binary asteroids formed as the result of catastrophic collision. In addition, (Richardson 2001) showed that NEA binaries could be formed via tidal disruption of a "rubble pile" and (Walsh et al. in prep.) have begun a systematic study of binary asteroid formation via tidal disruption.

These simulations raise an interesting problem for data analysis. In order to

understand the formation of binary asteroids fully, a fast, complete search method is needed that can identify both simple binary and hierarchical systems for $N \gg 10^3$. Once binaries and/or systems have been identified, their properties can be measured and compared to observed populations (with some assumptions on long-term stability). A brute-force search would require $\mathcal{O}(N^2)$ comparisons if each particle is compared with every other particle. A more complete and complex search would naively require $\mathcal{O}(N^3)$ comparisons if in addition every particle is compared with every system. Both searches are prohibitive for large N ($> 10^4$; less if multiple data sets or time series are considered).

5.1.3 Previous Work on Binary Detection in Numerical Simulations

The problem of developing an efficient method for finding bound groups of asteroids is related to searching for groups of galaxies in cosmological N-body simulations that contain large numbers of particles. In this case a nearest neighbor algorithm called "friend-of-friends" (FoF) (Davis et al. 1985) is often employed. FoF relies on a linking length test of a particle's nearest neighbors in order to determine what particles should be considered members of the group. For example, if particle B is within one linking length of particle A, particles A and B are in the same group. If particle C is within one linking length of particle B, particles A, B and C are in the same group, and so on. SKID (Governato et al. 1997) and the hierarchical clustering method (Zappalà et al. 1990) are more complex algorithms, but the group search is done in the same way.

We have developed our method along the lines of cosmological search methods, which are quite efficient. Because we are not specifically interested in spatial groups, we have replaced the linking length test with an escape speed test. The relative speed

of a particle and its possible companion is compared to their mutual escape speed to see if the pair is bound (in the absence of all other perturbations). To improve efficiency, we employ a Barnes & Hut (1986) hierarchical tree to limit the search for possible companions to those that are nearby (in the sense of being contained in a tree cell with a sufficiently large opening angle; cf. Section 5.2.1) or to those contained in a small or distant tree cell whose center-of-mass speed fails the escape speed test. It is possible that a small fraction of binaries may be missed with this method (see Section 5.3.2 for a discussion; in particular note that our tests show $\geq 99\%$ completeness in most cases, and it is always possible to set the tree criterion so low that all pairs are considered, at the expense of computation time).

In this paper we present `companion`, a hierarchical tree code that detects binaries, multiple, and complex hierarchical systems in the output from numerical simulations. Section 5.2 describes the numerical method in detail. Section 5.3 presents diagnostic and performance tests. In section 5.4 we present analysis of published data from (Durda et al. 2004), highlighting newly detected hierarchical systems. A summary and conclusions are given in section 5.5.

5.2 Numerical Method

In general, the most stable binaries are those that are the most tightly bound. This means that for a particle of a given mass the likelihood of having stable satellites decreases with increasing distance and relative speed (between the particle and potential satellite). The maximum distance at which a satellite can be bound to a particle depends on the combined mass of the system. As a result, `companion` uses a 3-D spatial tree code (Barnes & Hut 1986), augmented by a center of mass relative speed test to insure that widely separated systems are found (cf. section 5.2.2).

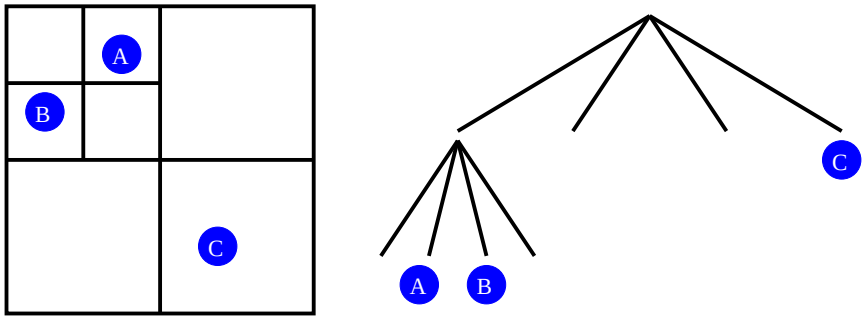

Figure 5.1: An example of a simple two dimensional spatial tree. On the right is a dipiction of division of the root cell of the tree after three particles (A, B, and C) have been placed into the tree. On the left is a "tree" diagram that describes the level of each particle in the tree. Starting at the top is the root cell. The first level below the root cell contains one cell with a particle C, two empty cells and a cell that has four daughter cells. At the second level below the root cell there are two cells each with one particle each (A and B) and two empty cells.

5.2.1 Hierarchical Spatial Tree

Our tree construction method closely follows the algorithm of (Barnes & Hut 1986). Particles are placed one at a time, according to their spatial coordinates, inside the "root cell," a cubical volume large enough to contain the entire system. Any time two particles end up in the same cell, the cell is divided in half along each coordinate axis, resulting in 8 daughter cells. The two particles in question are then placed into the respective daughter cell appropriate to their spatial coordinates. If they still share the same cell, the daughter cell is itself subdivided, and the process is repeated. The entire procedure continues recursively for all particles, until every particle resides in its own unique cell. At this point the entire tree has been built from the bottom up. Accessing any given particle requires "walking" the tree, beginning at the root cell, opening every cell that contains the particle of interest, and ending when the cell containing the particle has been reached. Figure 5.1 shows an example of a simple two dimensional spatial tree with three particles.

The premise of a spatial tree code is that particles far from a given particle of

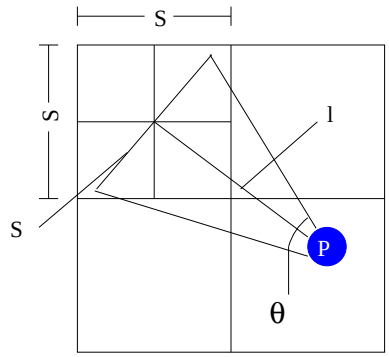

Figure 5.2: A graphical depiction of the opening angle test for a particle P, where $\theta = s/l$, s is the length of one side of the cell being tested, l is the distance between particle P and the center of the cell. The cell in question will be opened if $\theta > \theta_{crit}$.

interest (called particle P from now on) are generally not as important as those that are nearby. As a result, only particles that exert the most influence on P are considered in detail. In this case, such particles are those residing in cells that open an angle $\theta > \theta_{\rm crit}$ with respect to P, where $\theta = s/l$, s is the length of one side of the cell being tested, l is the distance between P and the center of the cell,[1] and $\theta_{\rm crit}$ is the "critical opening angle" (in radians), specified by the user. Figure 5.2 shows a diagram of the opening angle test. Tests show that $\theta_{\rm crit} = 0.5$ rad is a good compromise between speed and completeness (cf. section 5.3.2).

[1]Barnes and Hut (1986) used the center of mass instead of the geometric center of the cell for the opening angle test in order to have the dipole term in the multipole expansion of the gravitational potential vanish. Since `companion` does not use a multipole expansion, the geometric center is sufficient.

5.2.2 Binary Detection

After the tree is built the search for binaries begins. Every particle P is considered as a potential primary in turn and the opening angle test is used to determine whether a cell needs to be opened to search for satellites of P within that cell. If the open cell contains daughter cells, the same test is applied to them recursively. This continues until a cell passes the opening angle test ($\theta \leq \theta_{crit}$) or has no more daughters (i. e. the cell contains a single particle). In either case the speed v of the center of mass of the cell relative to P is then compared to the mutual escape speed $v_{\rm esc} = \sqrt{2GM/r}$, where G is the gravitational constant, M is the combined mass, and r is the separation. If the cell still has daughters and $v < v_{\rm esc}$, the daughter cells are forced open and the recursive procedure above resumes. This additional test insures `companion` identifies widely separated systems with low relative speed. Otherwise, if the cell contains a single particle and $v < v_{\rm esc}$, the particle is tagged as a companion to P.

5.2.3 System Detection

At this point `companion` contains a list of particle-particle binaries. The user has the option to use this list or to have `companion` go further and identify systems of particles (hierarchies). In that case, starting from the initial binary list, `companion` chooses the shortest-period system and replaces its two components with a single particle located at the center of mass of the binary, with the same total mass and linear momentum (angular momentum is ignored). The "radius" of the new particle is set equal to the semimajor axis of the binary orbit, in order to take advantage of filtering options described below (section 5.2.4; the collision cross section of the binary depends on the size of the orbit). Any binary in the original list that con-

tained either of the two components of the binary that was replaced is removed from the binary list. `Companion` then performs a binary test for the new center-of-mass particle using the method outlined above (section 5.2.2). Any new binaries that are detected are added to the binary list. This process is repeated until all bound systems of particles have been reduced to single center-of-mass particles.

Once the hierarchy option of `companion` has run to completion only two types of particles remain in the spatial tree—those particles that were never part of a binary and thus are original, unbound, single particles, and composite, center-of-mass particles. Each center-of-mass particle represents a separate system and each contains information about the primary and satellite of the system that it replaced. Thus the entire system represented by each composite particle can be reconstructed in the output (see section 5.3.1).

5.2.4 Usage Options

`Companion` provides several options to refine and filter searches. The user can choose to search for simple systems (section 5.2.2) or complex hierarchical systems (section 5.2.3). `Companion` accepts a variety of input and output units (cgs, mks, and "system units" in which $G \equiv 1$). Allowable input formats include plain text and binary, with one particle to a line and columns representing mass, radius, 3-D position vector, and 3-D velocity vector, respectively.

`Companion` also contains several filter options so that only binaries and hierarchical systems that meet certain criteria are reported. The user can specify a maximum eccentricity, minimum binding energy, maximum semimajor axis, and minimum periapse (including a criterion to reject binaries on re-impact trajectories). If a system is particularly interesting, it can be extracted from the original data, with or without the filtering options applied, and studied further in isolation. The user may also

change the critical opening angle $\theta_{\rm crit}$ used in the opening angle test—reducing $\theta_{\rm crit}$ will improve completeness but increase the computation time, and vice versa.

5.3 Tests

5.3.1 Illustrative Test

To test `companion` and demonstrate its capabilities, we created a hierarchical system based on our solar system that includes the Sun, the Earth and Moon, Jupiter, Io, and Europa, all in the $z = 0$ plane. We chose this system because it contains two subsystems (Earth-Moon and Jupiter-satellites) that `companion` should detect. Below is the normal (non-hierarchy) output from `companion` for this system. Each line of data output corresponds to a binary. In order, the columns are: mass ratio of the primary to the total system mass; index number of the primary (an integer assigned to each line of input data, starting at 0); radius of the primary; mass ratio of the satellite to the primary; index of the satellite; radius of the satellite; binary binding energy; semimajor axis; eccentricity; inclination; and orbital period. In this example output units are mks (inclination is always in radians). In this human-readable format, satellites sharing the same primary only show data from the fourth column on to emphasize associations. `Companion` also outputs a text machine-readable format for ease of interfacing with analysis routines.

```
 M_p/M_t     p_ind    p_rad  M_s/M_p     s_ind    s_rad  bind_eng        a    e    i      per
-------- --------- -------- -------- --------- -------- --------- -------- ---- ---- --------
9.99e-01         0 6.82e+08 9.45e-04         3 7.08e+07 -1.42e+35 8.82e+11 0.12 0.00 4.51e+08
                           2.94e-06         1 6.24e+06 -2.73e+33 1.43e+11 0.05 0.00 2.93e+07
                           3.68e-08         2 1.72e+06 -3.42e+31 1.43e+11 0.06 0.00 2.94e+07
9.44e-04         3 7.08e+07 4.71e-05         4 1.80e+06 -1.31e+31 4.25e+08 0.01 0.00 1.55e+05
                           2.54e-05         5 1.54e+06 -4.41e+30 6.84e+08 0.02 0.00 3.17e+05
2.93e-06         1 6.24e+06 1.25e-02         2 1.72e+06 -5.86e+28 2.45e+08 0.55 0.00 1.21e+06
```

```
Summary: 3 systems, 6 binaries, total mass considered = 1.995755e+30
```

In this example `companion` has identified three systems: 1) the Sun (particle 0) with Jupiter (particle 3), the Earth (particle 1), and the Moon (particle 2) as satellites; 2) Jupiter with Io (particle 4) and Europa (particle 5) as satellites; and 3) the Earth with the Moon as its satellite. The summary line at the end gives the number of systems (i. e. number of primaries), the number of binaries (primary-satellite pairs), and the total mass considered in the search. Note that since the relative speed between Jupiter's satellites and the Sun is greater than the escape speed from the Sun at their distance, `companion` does not identify them as members of the Sun system, even though they are members of the Jupiter system and Jupiter is a member of the Sun system.

Below is `companion` output for the same system with the hierarchy option turned on. The first column is the index number of the center-of-mass particle that has replaced the primary (third column) and satellite (sixth column). The other columns have the same meaning as in the normal `companion` output. Note that index numbers in the third and sixth column that are above 5 are also center-of-mass particles (recall numbering starts at 0 and there are 6 original particles in this test). Each separate system is identified by a new header line; in this case there is only one system identified (everything, including the Jovian satellites, is determined to belong to one system). The summary line for each system shows the total mass of the system with respect to the total mass of all particles considered, the maximum semimajor axis (a rough indication of the physical "size" of the system), and the total binding energy. After all systems have been listed, a global summary reports the total number of systems found (broken down into two-particle and multiple-particle systems), the total number of original particles, and the total mass considered in

the search.

```
    c_ind  M_p/M_t     p_ind    p_rad  M_s/M_p     s_ind    s_rad  bind_eng        a    e    i      per
--------- -------- --------- -------- -------- --------- -------- --------- -------- ---- ---- --------
       10 9.99e-01         9 1.43e+11 9.45e-04         7 6.84e+08 -1.42e+35 8.82e+11 0.12 0.00 4.51e+08
        9 9.99e-01         0 6.82e+08 2.97e-06         8 2.45e+08 -2.77e+33 1.43e+11 0.05 0.00 2.93e+07
        8 2.93e-06         1 6.24e+06 1.25e-02         2 1.72e+06 -5.86e+28 2.45e+08 0.55 0.00 1.21e+06
        7 9.45e-04         6 4.25e+08 2.54e-05         5 1.54e+06 -4.41e+30 6.84e+08 0.02 0.00 3.17e+05
        6 9.44e-04         3 7.08e+07 4.71e-05         4 1.80e+06 -1.31e+31 4.25e+08 0.01 0.00 1.55e+05
System summary: mass = 2.00e+30, max semimajor axis = 8.82e+11, total binding energy = -1.45e+35

1 system found: 0 2-particle systems and 1 multi-particle system
Total number of original particles: 6
Total mass in original particles: 2.00e+30
```

Figure 5.3 shows a visual representation of the hierarchical output for this test[2]. Jupiter (particle 3) and Io (particle 4) have the shortest period so they become the first center-of-mass particle (particle 6, shown in Fig. 5.3 as the black dot one level above Jupiter and Io). The next shortest period is the Jupiter-Io system with Europa (particle 5 in Fig. 5.3). The Jupiter-Io system is combined with Europa to form a new center-of-mass particle (7) that represents the entire Jupiter system. The next shortest period is the Earth-Moon system, particles 1 and 2 at the bottom of Fig. 5.3. They are combined to form another center-of-mass particle (8). The period of the Earth-Moon system around the Sun is shorter than the period of the Jupiter system around the Sun, thus the Earth-Moon system is combined with the Sun (particle 0) to form center-of-mass particle 9. Finally, the Jupiter system is combined with the Sun-Earth-Moon system to form particle 10. Ultimately the system is reduced to one center-of-mass particle.

[2]The software used to create the diagram figure 5.3 is also publically available at `http://www.astro.umd.edu/~zoe/companion/`. After `companion` has been run on the user's data with the hierarchy option run the plotting script with the index of the center of mass particle at the top of the desired system. The plotting script will produce a super mongo script.

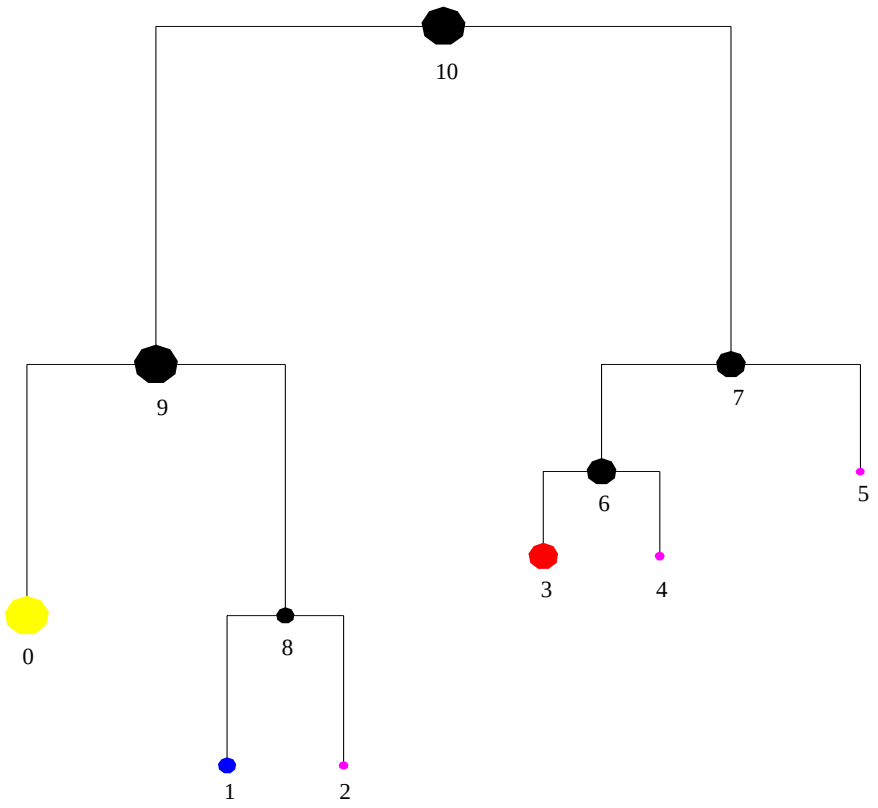

Figure 5.3: A visual representation of the output from the hierarchical option in `companion` for a pseudo solar system that included six particles: the Sun, the Earth, the Moon, Jupiter, Io, and Europa, all in a coplanar configuration. Particle 0 represents the Sun, 1 the Earth, 2 the Moon, 3 Jupiter, 4 Io, 5 Europa. All particles with particle indicies above 5 are center-of-mass particles. The radius of the dots corresponds to the mass of the particle with the most massive five times the radius of the smallest. Similarly, the lengths of the vertical branches in the tree correspond to the orbital period of each binary with the longest period four times that of the shortest.

5.3.2 Performance Tests

The development goal for `companion` was to find binaries, including hierarchical systems, in better than $\mathcal{O}(N^2)$ time. Figure 5.4 indicates this goal has been achieved: shown is the time needed to run `companion` on six numerical simulations of catastrophic asteroid collisions with various N and initial conditions. The default value $\theta_{\rm crit} = 0.5$ rad was used and no filtering was performed. Fig. 5.4 shows that the time it takes `companion` to complete the search for binary systems scales linearly with $N \log N$ for both normal and hierarchical search options. The scatter in both plots is due to the fact that several different simulations with different initial conditions were used in these tests. The hierarchy version of `companion` takes longer because each time a center-of-mass particle is replaced, a search for companions to that new particle is performed. In general, the number of binaries in a simulation is significantly less than the number of particles in the simulation.

To test the completeness of `companion` (the ability for it to identify all binaries in the data set being tested), we used $\theta_{\rm crit} = 0$, effectively forcing `companion` to behave as an inefficient N^2 code, without any chance of missing a binary. From this test we found that for $\theta_{\rm crit} = 0.5$ rad, `companion` is at least 99% complete for all data sets tested (N-body simulations of catastrophic asteroid collision events which have been run a few days past the collision) and two orders of magnitude faster than a traditional N^2 search method. For MBA collision simulations, $\theta_{\rm crit} = 0.5$ rad optimizes completeness and speed. In other scenarios it is possible that a more conservative opening angle is required.

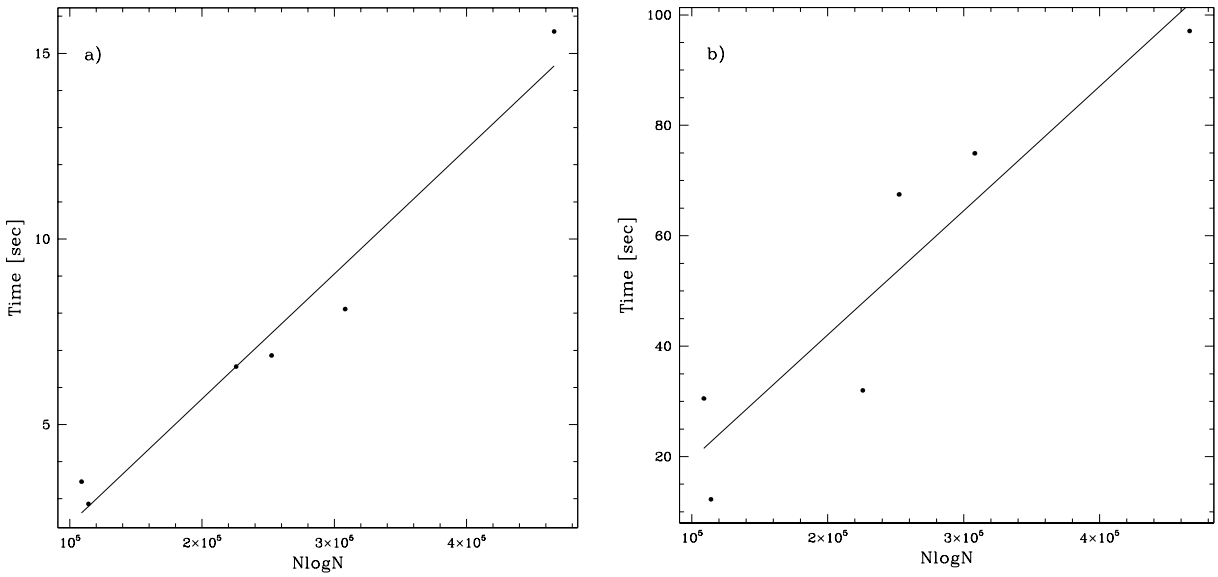

Figure 5.4: (a) CPU time versus $N \log N$ in seconds for default `companion` analysis of the results of catastrophic asteroid collision simulations (Durda et al. 2004, ; Durda, personal communication). (b) Timing results for the same data using the hierarchical search. The data sets contained between 2.6×10^4 and 9.4×10^4 particles. The solid lines are least-squares fits to each data set.

5.4 Results

An older version of `companion` without the hierarchy option was used for the analysis of satellite formation simulations in Durda et al. (2004). The updated version produces similar results for the three data files from Durda et al. (2004) that we used as test cases. For both versions, `companion` was used with two filters applied: 1) a maximum semimajor axis of one Hill radius (at 3 AU from the Sun); and 2) a minimum periapse distance of twice the primary radius. Due to some improvements in how the filters are applied in `companion`, we found a slight difference in the number of satellites reported by the new version ($< 0.5\%$ difference). Thus, the overall statistics reported in Durda et al. (2004) are consistent with our tests.

Since Durda et al. (2004) did not have the hierarchy option available, we have

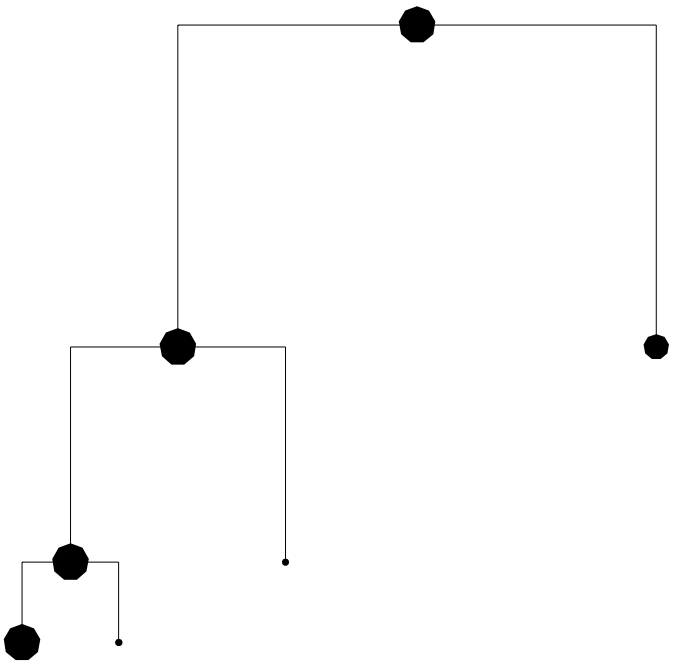

Figure 5.5: An example of an interesting hierarchy found by `companion` in a simulation from Durda et al. (2004), with no filtering applied.

done a preliminary analysis with it on the simulation that produced the most binaries. The impact parameters of this simulation are as follows: impact speed $\sim$ 3 km s^{-1}, impact angle at collision of 30 degrees, diameter of projectile of 34 km, diameter of target of 100 km. We have found a number of interesting hierarchical configurations in their data (Fig. 5.5 gives an example, using `companion` without any filtering). Most of the more interesting hierarchies occur between smaller particles (what Durda et al. 2004, call "EEBs", or escaping ejecta binaries). These are systems escaping the largest post-collision remnant and that consist of smaller fragments with low relative speeds. We have also run `companion` on the same data with

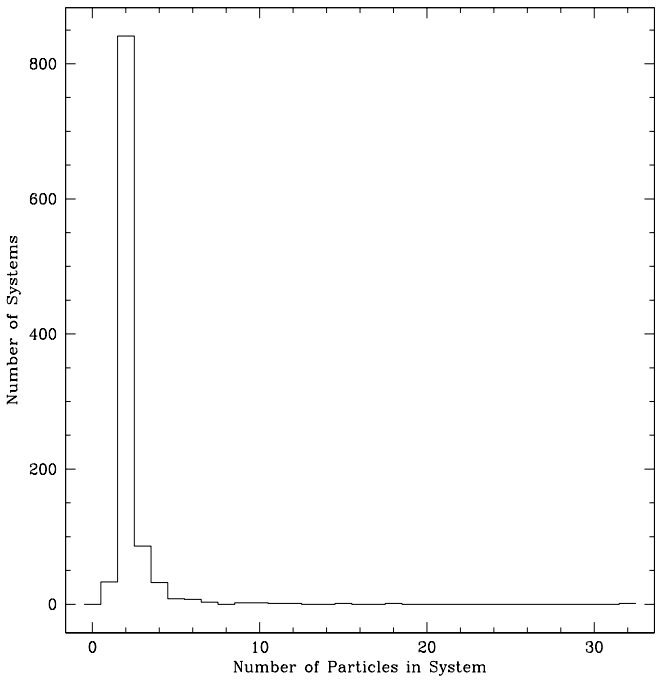

Figure 5.6: A histogram of the number of systems found with N particles using the hierarchy option in `companion`. Only original particles are counted, center of mass particles are not included in the calculation of the number of particles in a system.

the Hill sphere and pariapse cuts mentioned above. `Companion` found 1101 systems with 129 multiple systems and 972 2-particles systems applying the above mentioned cuts without the hierarchy option. With the hierarchy option turned on `companion` found 1020 systems with 862 2-particle systems and 158 multiple systems. This means that about 80 2-particle systems detected without the hierarchy option have their center of mass bound to another system. Figure 5.6 shows a histogram of the number of N-particle systems. As expected the majority of systems are binaries but there are a significant number of trinary systems ($\sim 10\%$ the number of binaries) and quarternary systems ($\sim 3\%$) that passed the orbital restrictions.

We also found 30 multiparticle systems (mostly triples) that seemed to be relatively stable in the sense that they survived for several days. These systems all

passed the periapse and semimajor axis filter options described above. In addition, these systems did not contain any particles or binaries that pass within one semimajor axis of any other binary in the system. As a test, some of these systems were extracted from the data file and intergrated in isolation for several orbits. Three configurations of particles were found to be most stable: 1) a large primary orbited by two-to-three small particles; 2) a tight binary orbited by a smaller particle; 3) a larger particle orbited by a tight binary. For the inner binary in configuration 2, both equal and unequal-size components worked well. The orbital parameters of the configurations varied but the tight binaries in configurations 2 and 3 often had relatively moderate-to-low eccentricity (≤ 0.4).

5.5 Conclusions

In this paper we presented `companion`, a publicly accessible, efficient code for finding binaries and bound systems in output from numerical simulations. We found that both simple and complex searches scale as $\mathcal{O}(N \log N)$ with the new code. We discussed the capabilities of this code in the context of binary asteroid formation, showing that data from Durda et al. (2004) contains previously unreported hierarchical systems. However, it should be noted that `companion` can in principle be applied to any data set that includes particle mass, radius, position, and velocity.

The completeness of `companion` is dependent on the critical opening angle $\theta_{\rm crit}$. For the evolved asteroid collision simulations tested here, the default value of 0.5 rad provided better than 99% completeness. Other configurations may exist for which a more stringent value of $\theta_{\rm crit}$ is required, at the cost of computation time, such as those with large numbers of barely bound, spatially far removed particles. It also must be emphasized that all binaries and multiple systems reported by `companion`

are instantaneous, could very well be transient, and may only exist in the context of surrounding particles (i. e. such systems may fly apart when extracted from their broader context). Thus, it may be most useful to apply `companion` to dynamically evolved data sets, as we have done, or to use `companion` to study the statistics and evolution of transient systems.

Acknowledgments

This material is based upon work supported by the National Aeronautics and Space Administration under Grant Nos. NAG511722 and NGT550454 issued through the Office of Space Science and by the National Science Foundation under Grant No. AST0307549. The analysis presented in this paper was carried out using the Beowulf cluster administered by the Center for Theory and Computation of the Department of Astronomy at the University of Maryland. ZML would like to thank the Kavli Institute of Theoretical Physics at the University of California, Santa Barbara, as well as D. Hamilton and A. J. Young.

Chapter 6

Conclusions

In this thesis I have investigated planetesimal evolution, the role of planetesimal collisions in determining the outcome of terrestrial planet formation, and the prevalence of hierarchical bound systems in numerical simulations of family-forming events. In order to parameterize planetesimal collision outcomes and determine what conditions are required for planetesimal growth, I investigated how impact parameter, impact speed, spin, and mass ratio affect collision outcomes. Since the self-gravity of a km-sized planetesimal is larger than its material strength I modeled the planetesimals as pure rubble piles—gravitational aggregates with no tensile strength. The parameter space studies produced a variety of shapes and spin states including elongated, pear-shaped, contact-binaries, and spherical remnants. A number of collision remnants were similar in physical characteristics to main belt asteroids such as Eros, Geographos, Kleopatra, and Hektor. No stable large satellites were produced in these studies. However, previous numerical simulations have found that binary asteroids form following catastrophic impact events within the asteroid belt. Upon analyzing this data I found that more complex hierarchical bound systems also formed as a result of these impacts.

The results of the parameter space collision studies suggested that rubble-pile

planetesimals in an primordial disk should typically grow. However, the parameter space studies were restricted to single isolated collisions. In order to determine what would happen if a realistic self-consistent planetesimal model were included in a global planet formation simulation I modified and expanded the planetesimal model developed in the parameter space studies and incorporated it into planet formation scenarios. I found that the initial conditions of the primordial disk were far more important in determining the mass and number of the protoplanets than the planetesimal collision model. In fact the work suggests that planetesimal composition is effectively unimportant in planet formation.

The first simulations of planetesimal collisions (chapter 2) concentrated on determining how collision parameters affected the collision outcome between equal-sized planetesimals. The total amount of energy and angular momentum of the collisions produced a variety of shapes and spin states for the largest post-collision remnant. Large impact parameter (large orbital angular momentum) resulted in elongated, fast spinning remnants. Large impact speed (large energy) resulted, as one might expect, in more mixing and a smaller largest remnant. The initial spin of the impactor could increase or decrease the spin period and elongation of the largest remnant depending on whether the spin angular momentum was parallel or anti-parallel to the orbital angular momentum. Aligned spin and orbital angular momentum vectors produced symmetric shapes, whereas anti-aligned angular momentum vectors produced triaxial shapes. Most of the debris from the collisions was distributed in a plane perpendicular to the impact. When the collision was catastrophic debris often coalesced into smaller rubble piles. In one set of simulations that were completed with a mass ratio of 1 to 10, the projectile was always effectively destroyed. In impacts with large impact parameter, a fraction of the target was sheared off. Collisions with large orbital angular momentum could also result in debris from the

disrupted projectile being deposited onto the target along the equator.

Contact binaries formed easily about 10% of the time but true binaries of significant size did not form from slow collisions—most of the post-collision debris was either accreted onto the largest remnant or escaped from it. The coefficient of restitution had a significant effect on the number and size of remnants. A lower value (more dissipation) resulted in fewer larger remnants, a larger value (less dissipation) produced a larger number of small remnants. Particle resolution had a moderate effect—a larger number of particles gave rise to more complexities in a given shape, such as cusps on S-shaped remnants. More particles, higher resolution, also increased dissipation slightly. I found that for equal-sized rubble piles, the critical dispersal energy $Q_D^\star \sim 2$ J kg^{-1} for a head-on collision which is orders of magnitude below what other numerical simulations of asteroid impacts have found (Love & Ahrens 1996; Ryan & Melosh 1998). This would suggest that rubble piles are relatively easy to disrupt, however, these results are for equal-sized impactors and, in addition, our coefficient of restitution was high ($\epsilon = 0.8$).

In the second set of parameter space simulations the focus was to quantify the effect of impactor mass ratio on collision outcome. The size of the projectile was important in determining the degree of disruption. More massive projectiles were more efficient at disrupting a target than a less massive projectile with the same impact energy because a larger projectile has a bigger surface area. The larger surface area means that more particles are directly involved in a collision with a larger projectile than with a smaller projectile. This means that larger projectiles are more efficient at distributing the impact energy evenly throughout the target, and hence a large fraction of the impact energy is available to disrupt the target. Smaller projectiles on the other hand tend to channel the impact energy into a small number of particles that may escape the system at high speed carrying much

of the impact energy with them leaving only a fraction of the impact energy to disrupt the rest of the target. Extrapolating from the results, the amount of energy necessary to critically disrupt a target of radius 1 km using a projectile of radius 1 m is between 1000 and 10,000 J kg^{-1}—three to four orders of magnitude greater than what is necessary to critically disrupt the same target with an equal sized projectile. Further, as mass ratio departs from unity, the impact parameter becomes less important and the probability of planetesimal growth increases. For an average encounter (assuming a power law size distribution with an index of -3—collisionally relaxed) the target is likely to gain mass.

In order to determine how planetesimals evolve over time after suffering repeated collisions I designed and integrated a rubble pile collision model into a planet formation code. The collision model was based on the method used in chapters 2 and 3. For maximum efficiency the collision model was multi-phase and the planet formation code used multi-stepping. Since the orbital time in these simulations ~ 1 year and the dynamical time of a planetesimal ~ 1 hour ($\tau \sim 1/\sqrt{G\rho}$) the planet formation code used two timesteps: one to resolve the orbits (0.01 year) and one to resolve collisions and close approaches between planetesimals (~ 0.0001 year). When a collision was predicted between two planetesimals in the the protoplantary disk the timesteps were reduced by a factor of 64 for the planetesimals involved. When a collision is confirmed between two planetesimals the collision parameters were used to look up the predicted mass of the largest post-collision remnant in a data table of previously integrated collisions. If the predicted largest post-collision remnant contains most of system mass ($> 80\%$) the predicted outcome was used as the collision outcome. If the second largest remnant is large ($> 20\%$ the system mass) the collision is fully resolved (that is, modeled as a rubble pile collision directly) within the planet formation simulation. A resolution limit was employed

to keep the number of particles in the simulation from growing too large. After 20 planetesimal dynamical times any material remaining below the resolution limit was followed in a semi-analytic manner as unresolved debris.

The results of the planet formation simulations, with a detailed, self-consistent collision model, were compared to simulations with the same initial conditions but with all collisions resulting in perfect merging. Planetesimals evolved more quickly during runaway growth in my model but by the end of oligarchic growth the number, mass, and separation of the protoplanets was very similar to the results that used perfect merging. Different initial conditions (total mass in the protoplanetary disk and distribution of the mass with semi-major axis) affected the number and mass of the protoplanets more significantly than the collision model. Thus, according to the collision model employed here, fragmentation is not important in determining the final outcome of planet formation, nor, as I also found, is the coefficient of restitution. As a result, the material composition of planetesimals seems unimportant in the gravity dominated phase of planet formation. At the end of our simulations the eccentricities of the protoplanets are about an order of magnitude above the eccentricity of the Earth (consistent with Kokubo & Ida 2002). It is possible that the gas-free initial condition is not entirely appropriate. Interactions between the protoplanets and the gas disk could potentially damp the remnant eccentricities (Agnor & Ward 2002).

Unlike the collisions of the planetesimal evolution presented in chapters 2 and 3 several numerical simulations have found that binary asteroids form easily as the result of catastrophic, family-forming impacts in the main asteroid belt. Asteroids are interesting because many are much less altered than planets from the original planetesimals. Binary and multiple systems are particularly useful because masses can be directly measured from binaries using Kepler's law. Numerical simulations of

family-forming events require high resolution to model all of the necessary physics. Large numbers of particles with large numbers of outputs present a problem for data mining since it becomes prohibitive to search through all of the data for pairwise correlations. I developed an efficient search code that finds simple bound and hierarchical systems in $\mathcal{O}(N \log N)$ time, compared to traditional brute force techniques that scale as $\mathcal{O}(N^2)$ and hierarchical searches that scale as $\mathcal{O}(N^3)$.

I applied our search code to previously published numerical simulations of catastrophic impacts between asteroids. I found several hierarchical systems that were relatively stable in the sense that they persisted for at least several hundred orbits. These results suggest that higher order systems ($N > 2$) could form in the asteroid belt. As observational techniques become more sensitive, observers should be on the look out for triple systems.

6.1 Future Work

My work will continue along three paths. First, I plan to test the planetesimal collision model further by running several high-resolution simulations with various initial debris values. It is possible that debris, if sufficiently massive, can become dynamically important if an equilibrium can be reached between the production and the accretion of the debris onto the planetesimals. It may also help to reduce the eccentricities of the protoplanets. In addition, I plan to confirm the results presented here by running a standard model simulation with no expansion parameter. This simulation will determine the timescale for planet formation and would also allow for the evolution of spins of the protoplanets to be investigated.

Second, planetesimal formation is an open question—no model has been shown to grow planetesimals—and yet planetesimals provide the initial conditions for most

planet formation simulations. At the moment these initial conditions are assumed because it has not been possible to grow planetesimals from dust in numerical simulations. I intend to begin developing a systematic program of study testing various methods of planetesimal formation. Although turbulence in the gas disk surrounding a young star constrains dust particle motion and drives dust collisions at speeds of $\sim$ m s^{-1}, these dust particles are also in a tidal field and as a result their orbits are also roughly circular and may collide many times an orbit. Each collision is inelastic resulting in a loss of energy. As long as the loss in energy is larger than the energy gain due to gas turbulence, the relative impact speeds of the dust particles decrease with time. This may result in clumping of dust particles. In addition, the fractal and porous nature of these clumps may help prevent future destruction. The N-body code that I have used here can support the investigation of this model of planetesimal formation. I will study the role of gravity, material properties of the dust, and the effect of gas drag and turbulence.

Third, comets are the oldest and most pristine objects in the solar system. A detailed understanding of their evolution will result in a very good constraint on the material requirements of the early solar system. One of the most significant modifications that comets have experienced are impacts. Modeling these impacts is difficult because impacts are fast ($\sim$ km s^{-1}) and comets have a large percentage of volatiles which can change phase as a result of the collision. I plan to hybridize a hydrodynamic code—to model the initial impact and shock propagation—and an N-body gravity code (`pkdgrav`)—to track the trajectories of the remnants under the influence of gravity. I will build simplified numerical models of comets composed of ice and basalt. Initially, I will ignore the effects of organics in order to keep the number of model parameters practical. I will investigate the effect of micro- and macro-porosity and the effect of varying internal structure (e.g. models in which

basalt and ice are uniformly interspersed, and in which ice surrounds a basalt core). The results of these simulations will help determine the percentage of volatiles lost as the result of each impact.

Together these simulations will further advance our understanding of our own solar system by connecting present-day objects within the solar system with the building blocks of our planet.

Appendix A

Derivation of Isolation Mass

Eq. 4.7 is derived by equating the isolation mass to the mass available for consumption in the feeding zone.

$$M_{iso} \equiv 2\pi a b \Sigma_{solid}, \tag{A.1}$$

where a is the semi-major axis of the planetesimal/protoplanet, b is the characteristic separation between protoplanets due to gravitational repulsion (Kokubo & Ida 1995), and Σ_{solid} is the surface density of solid material in the protoplanetary disk. The characteristic separation distance can be expressed in units of Hill radii (r_H),

$$\tilde{b} \equiv \frac{b}{r_H} \tag{A.2}$$

$$r_H \equiv \left(\frac{2M}{3M_*}\right)^{1/3} a \tag{A.3}$$

hence

$$b = \left(\frac{2M}{3M_*}\right)^{1/3} a\tilde{b}, \tag{A.4}$$

where M is the mass of the protoplanet and M_* is the mass of the central star. Substituting b and M_{iso} for M,

$$M_{iso} = a^2\tilde{b}\left(\frac{2M_{iso}}{3M_*}\right)^{1/3} \Sigma_{solid} \tag{A.5}$$

or

$$M_{iso} = Aa^3 \tilde{b}^{3/2} \Sigma_{solid}^{3/2} M_*^{-1/2}, \tag{A.6}$$

where all of the numerical constants have been consolidated into A. In the simulations presented here the mass was initially distributed as a power-law with index α,

$$\Sigma_{solid} = f_{ice} \Sigma_1 \left(\frac{a}{1\mathrm{AU}} \right)^{-\alpha} \tag{A.7}$$

therefore,

$$M_{iso} = Aa^{3/2(2-\alpha)} \tilde{b}^{3/2} \left(f_{ice} \Sigma_1 \right)^{3/2} M_*^{-1/2}, \tag{A.8}$$

where f_{ice} is the fraction of solids in ice and Σ_1 is the surface density of solids at 1 AU. Eq. 4.7 from Eq. A.8 by introducing standard unit normalizations.

Appendix B

companion.c

```
/*
 ** companion.c --- Version 1.0, 07.27.04 (ZML and DCR)
 ** searches for binaries
 */

/*
 ** This code is Copyright &copy; 2004 by Z. M. Leinhardt and D. C.
 ** Richardson.
 ** Under the terms of the GNU Public License, you are free to
 ** redistribute, modify, or even sell this code, but we ask that all
 ** headers identifying the original authors of this code be left
 ** intact.
 **
 ** This software comes with NO WARRANTY.  The authors cannot be held
 ** responsible for any undesirable consequences of using this code.
 */

/*
 ** Memory storage for particles will be inflated by the following
 ** factor.  This is needed because we cannot realloc() the particle
 ** storage (in order to add com particles) after the final data read;
 ** otherwise pointers in the tree and binary list to the particle data
 ** won't work.  A better method is to store particle array indices in
 ** the tree and binary list data fields and pass a pointer to the
 ** particle array to whatever routines need the particle data.  This
 ** is annoying because the functions affected include recursive tree
 ** functions and the sorting comparison functions, so for the moment
 ** we stick with this crude buffer inflation and hope it's enough
```

```
 ** storage (an assert() is used to make sure).
 */
#define EXTRA_STORE 2

/*
 ** Fraction of particles allowed to remain since last tree build
 ** before rebuilding the tree.  Used for hierarchical system search.
 ** A large value forces more frequent tree rebuilds.  A smaller value
 ** relies on older tree data for longer, with larger inefficiencies.
 ** More tests are needed to optimize this value, though it may vary
 ** depending on the specific problem.
 */
#define TREE_REBUILD_FRAC 0.9

/*
 ** If mass ratio between two components of a com particle is extreme
 ** force a tree rebuild.
 */
#define REBUILD_MASS_RATIO 1.0e6

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h> /* for getopt() */
#include <string.h>
#include <math.h>
#include <limits.h> /* may or may not contain DBL_MAX (values.h */
                    /* obsolete?) */
#include <assert.h>

#ifndef MAXPATHLEN
#define MAXPATHLEN 256
#endif

#ifndef DBL_MAX
#define DBL_MAX 1.7976931348623157E+308
#endif

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

#define ANA_EXT ".ana"
#define PR_EXT ".pr"
#define HIER_EXT ".hier"
```

```
#define EXT_EXT ".ext"
#define HIER_EXT_EXT ".hext"
#define VEC_EXT ".vec"

#define BUF_SIZE_INIT 256
#define BUF_SIZE_MULT 2
#define BUF_NPART_INIT 10000

#define CHILD_PER_NODE 8 /* don't change this! (oct-tree) */

#define FILE_TYPE_STR_MAX_LEN 4 /* 3 chars plus null char */

#ifdef SS_CORE

#include <rpu.h>
#include <ss.h>
#include <vector.h>

enum {FileTypeTxt,FileTypeBin,FileTypeSS};
const char FileTypeStr[][FILE_TYPE_STR_MAX_LEN] = {"txt","bin","ss"};

#define DFLT_FILE_TYPE FileTypeSS
#define DFLT_IN_SYS TRUE
#define DFLT_IN_CGS FALSE
#define DFLT_OUT_SYS TRUE
#define DFLT_OUT_CGS FALSE
#define DFLT_LENGTH_CONV    (1.0)       /* default in sys units if */
#define DFLT_MASS_CONV      (1.0)       /* ss_core defined */
#define DFLT_TIME_CONV      (1.0)
#else /* define macros and types from ss_core */

#define N_DIM 3
#define TWO_PI (2*M_PI)

#define BOOLEAN int
#define FALSE 0
#define TRUE 1

/* Fundamental constants from 1998 Astronomical Almanac, in mks */

/* One A.U. in metres [1.4959787066e11] */
#define AU      1.49597870e11
#define M_SUN   1.9891e30       /* Solar mass in kilograms */
/* One sidereal year in seconds (1998.0) */
```

```
#define SID_YR  3.15581497632e7

typedef double VECTOR[N_DIM];

#define X 0
#define Y 1
#define Z 2

/* Vector function prototypes */

void SET_VEC(VECTOR,double,double,double);
void ZERO_VEC(VECTOR);
void COPY_VEC(VECTOR,VECTOR);
void ADD_VEC(VECTOR,VECTOR,VECTOR);
void SUB_VEC(VECTOR,VECTOR,VECTOR);
void SCALE_VEC(VECTOR,double);
void NORM_VEC(VECTOR,double);
double DOT(VECTOR,VECTOR);
void CROSS(VECTOR,VECTOR,VECTOR);
double MAG_SQ(VECTOR);
double MAG(VECTOR);

/* Assigns a value (x,y,z) to vector v */

#define SET_VEC(v,x,y,z) {\
  (v)[X] = (x);\
  (v)[Y] = (y);\
  (v)[Z] = (z);\
  }

/* Assigns zero to vector v */

#define ZERO_VEC(v) SET_VEC((v),0,0,0)

/* Copies vector v1 to vector v2 */

#define COPY_VEC(v1,v2) {\
  (v2)[X] = (v1)[X];\
  (v2)[Y] = (v1)[Y];\
  (v2)[Z] = (v1)[Z];\
  }

/* Adds vectors v1 & v2 and puts the result in vector v */
```

```
#define ADD_VEC(v1,v2,v) {\
  (v)[X] = (v1)[X] + (v2)[X];\
  (v)[Y] = (v1)[Y] + (v2)[Y];\
  (v)[Z] = (v1)[Z] + (v2)[Z];\
  }

/* Subtracts vector v2 from vector v1 and puts the result in vector v */

#define SUB_VEC(v1,v2,v) {\
  (v)[X] = (v1)[X] - (v2)[X];\
  (v)[Y] = (v1)[Y] - (v2)[Y];\
  (v)[Z] = (v1)[Z] - (v2)[Z];\
  }

/* Multiplies vector v by scalar a */

#define SCALE_VEC(v,a) {\
  double _scalar = (a);\
  (v)[X] *= _scalar;\
  (v)[Y] *= _scalar;\
  (v)[Z] *= _scalar;\
  }

/* Divides vector v by scalar a */

#define NORM_VEC(v,a) SCALE_VEC((v),1.0/(a))

/* Returns dot product of vectors v1 & v2 */

#define DOT(v1,v2) ((v1)[X]*(v2)[X] + (v1)[Y]*(v2)[Y] + (v1)[Z]*(v2)[Z])

/* Returns cross product of vectors v1 & v2 in vector v */

#define CROSS(v1,v2,v) {\
  (v)[X] = (v1)[Y]*(v2)[Z] - (v1)[Z]*(v2)[Y];\
  (v)[Y] = (v1)[Z]*(v2)[X] - (v1)[X]*(v2)[Z];\
  (v)[Z] = (v1)[X]*(v2)[Y] - (v1)[Y]*(v2)[X];\
  }

/* Returns square magnitude of vector v */

#define MAG_SQ(v) (DOT((v),(v)))

/* Returns magnitude of vector v */
```

```
#define MAG(v) (sqrt(MAG_SQ(v)))

/* Struct for storing particle data */

typedef struct {
    double mass;
    double radius;
    double pos[N_DIM];
    double vel[N_DIM];
    double spin[N_DIM];
    int color;
    int org_idx;
    } SSDATA;

enum {FileTypeTxt,FileTypeBin};
const char FileTypeStr[][FILE_TYPE_STR_MAX_LEN] = {"txt","bin"};

#define DFLT_FILE_TYPE FileTypeTxt
#define DFLT_IN_SYS FALSE
#define DFLT_IN_CGS TRUE
#define DFLT_OUT_SYS FALSE
#define DFLT_OUT_CGS TRUE
#define DFLT_LENGTH_CONV    (AU*1.0e2)      /* default in cgs units */
#define DFLT_MASS_CONV      (M_SUN*1.0e3)
#define DFLT_TIME_CONV      (SID_YR/TWO_PI)
#define SQ(x) ((x)*(x))
#define CUBE(x) ((x)*(x)*(x))

#endif /* !SS_CORE */

#define SUPPORT_OLD_FORMAT

/* defaults */

#define DFLT_HIER FALSE
#define DFLT_TIPSY_FILE FALSE
#define DFLT_IN_MKS FALSE
#define DFLT_OUT_MKS FALSE
#define DFLT_DO_PERI_CUT FALSE
#define DFLT_DO_ORBIT_CUT FALSE
#define DFLT_APPLY_HIER_CUT FALSE
#define DFLT_ECC_CUT 1.0
#define DFLT_ENG_CUT 0.0
```

```
#define DFLT_HILL_CUT 0.0
#define DFLT_PERI_CUT 0.0
#define DFLT_ORBIT_CUT 0.0
#define DFLT_OPEN_ANGLE 0.5
#define DFLT_HIER_EXTRACT_INDEX (-1)
#define DFLT_EXTRACT_INDEX (-1)

typedef struct {
    /* args */
    BOOLEAN Hier,TipsyFile,InCgsUnits,InMksUnits,InSysUnits,OutCgsUnits,
            OutMksUnits,OutSysUnits;
    int FileType;
    long ExtIdx,HierExtIdx;
    double EccCut,EngCut,HillCut,PeriCut,OrbitCut,OpenAng;
    /* prompted */
    double avga,starmass;
    /* derived */
    BOOLEAN DoPeriCut,DoOrbitCut,ApplyHierCut;
    double InLengthConv,InMassConv,InTimeConv,InEnergyConv/*NOT USED*/;
    double OutLengthConv,OutMassConv,OutTimeConv,OutEnergyConv;
    } PARAMS;

struct compdata { /* contains particle info and a pointer to parent */
                  /* and children */
    struct compdata *prim;
    struct compdata *sat;
    struct compdata *com;
    SSDATA data;
    long index;
    };

typedef struct compdata COMPDATA;

typedef struct { /* contains the primary and satellite of a bound pair */
    COMPDATA *prim;
    COMPDATA *sat;
    double period; /* only used for hierarchical grouping */
    } BINARY;

typedef struct { /* contains summary data for each hierarchical system */
    const COMPDATA *com;
    double sys_mass;
    double max_a;
    double bind_E;
```

```
    } HIER_OUTPUT; /* only used in write_hier_output */

struct node {
    VECTOR pos;
    VECTOR vel;
    double size,mass;
    double half_size, eff_half_size,eff_size_sq;
    struct node *child[CHILD_PER_NODE];
    COMPDATA *leaf[CHILD_PER_NODE];
    int n_part;
    };

typedef struct node NODE;

#define BIN_ENERGY      (1 << 0) /* do not change these! */
#define BIN_ANGMOM      (1 << 1)
#define BIN_SEMI        (1 << 2 | BIN_ENERGY)
#define BIN_PERIOD      (1 << 3 | BIN_SEMI)
#define BIN_ECC         (1 << 4 | BIN_ANGMOM | BIN_SEMI)
#define BIN_PERIAPSE    (1 << 5 | BIN_SEMI | BIN_ECC)
#define BIN_INCL        (1 << 6 | BIN_ANGMOM)

/* (BIN_SEMI for Hill cut) */
#define BIN_CUT (BIN_ENERGY|BIN_SEMI|BIN_ECC|BIN_PERIAPSE)
#define BIN_ALL (BIN_ENERGY|BIN_ANGMOM|BIN_SEMI|BIN_PERIOD|BIN_ECC|
                 BIN_PERIAPSE|BIN_INCL)

int BIT_ON(int,int);
#define BIT_ON(flag,mask) (((flag) & (mask)) == (mask))

typedef struct {
    double E; /* total energy */
    double a; /* semimajor axis */
    double P; /* period */
    double e; /* eccentricity */
    double q; /* periapse */
    double i; /* inclination */
    } BIN_STATS;

/* handy macros for hierarchical system search */

BOOLEAN IS_COM_MBR(const COMPDATA *);
#define IS_COM_MBR(p) ((p)->com != NULL)
```

```
BOOLEAN HAS_COM_MBR(const COMPDATA *);
#define HAS_COM_MBR(p) ((p)->prim != NULL || (p)->sat != NULL)

BOOLEAN IS_SAME_BINARY(const BINARY *,const BINARY *);
#define IS_SAME_BINARY(b1,b2) ((b1)->prim == (b2)->prim &&
                               (b1)->sat == (b2)->sat)

void calc_bin_stats(const SSDATA *p1,const SSDATA *p2,int flag,
                    BIN_STATS *bs)
{
    VECTOR r,v;
    double M_inv;

    assert(p1 && p2 && bs);
    assert(flag > 0);

    SUB_VEC(p1->pos,p2->pos,r);
    SUB_VEC(p1->vel,p2->vel,v);

    M_inv = p1->mass + p2->mass; /* not needed if just BIN_INCL */
    assert(M_inv > 0.0);
    M_inv = 1.0/M_inv;

    if (BIT_ON(flag,BIN_ENERGY)) {
        double mag_r_inv,v2overM;

        mag_r_inv = MAG(r);
        assert(mag_r_inv > 0.0);
        mag_r_inv = 1.0/mag_r_inv;
        v2overM = MAG_SQ(v)*M_inv;
        bs->E = p1->mass*p2->mass*(0.5*v2overM - mag_r_inv);
        assert(bs->E < 0.0); /* must be bound to be a binary! */
        if (BIT_ON(flag,BIN_SEMI)) {
            bs->a = 1.0/(2.0*mag_r_inv - v2overM);
            assert(bs->a > 0.0);
            }
        if (BIT_ON(flag,BIN_PERIOD)) {
            bs->P = TWO_PI*sqrt(CUBE(bs->a)*M_inv);
            assert(bs->P > 0.0);
            }
        }

    if (BIT_ON(flag,BIN_ANGMOM)) {
        VECTOR h;
```

```
        double h2;

        CROSS(r,v,h); /* ang mom per unit reduced mass */
        h2 = MAG_SQ(h);
        if (BIT_ON(flag,BIN_ECC)) {
            double x = h2*M_inv/bs->a;
            assert(x <= 1.0);
            bs->e = sqrt(1 - x);
            /* allow e=1 for now, will be cut */
            assert(bs->e >= 0.0 && bs->e <= 1.0);
            }
        if (BIT_ON(flag,BIN_INCL)) {
            double mag_h = sqrt(h2);
            assert(mag_h > 0.0);
            bs->i = acos(h[Z]/mag_h);
            }
        }

    if (BIT_ON(flag,BIN_PERIAPSE)) {
        bs->q = bs->a*(1 - bs->e);
        assert(bs->q >= 0.0); /* allow q=0 for now, will be cut */
        }
    }

int sort_bin(const void *a,const void *b)
{
    /* sort function for "normal" output (not hierarchical) */

    const BINARY *b1,*b2;
    const SSDATA *p1,*p2;

    b1 = (const BINARY *) a; /* pointers to binaries */
    b2 = (const BINARY *) b;

    p1 = &(b1->prim->data); /* pointers to primaries */
    p2 = &(b2->prim->data);

    /* largest primary masses first */
    if (p1->mass < p2->mass) return 1;
    if (p1->mass > p2->mass) return -1;

    /* smallest indices first */
    if (b1->prim->index < b2->prim->index) return -1;
    if (b1->prim->index > b2->prim->index) return 1;
```

```
    /* within each system, sort by binding energy */

    {
    /* satellite pointers */
    const SSDATA *s1 = &(b1->sat->data),*s2 = &(b2->sat->data);
    BIN_STATS bs1,bs2;

    calc_bin_stats(p1,s1,BIN_ENERGY,&bs1);
    calc_bin_stats(p2,s2,BIN_ENERGY,&bs2);

    if (bs1.E < bs2.E) return -1; /* more bound first */
    if (bs1.E > bs2.E) return 1;

    /* smallest indices first */
    if (b1->sat->index < b2->sat->index) return -1;
    if (b1->sat->index > b2->sat->index) return 1;
    }

    assert(0); /* shouldn't be here (duplicates not allowed) */

    return 0;
    }

double hill(const PARAMS *p,double mass_i,double mass_c)
{
    /* Calculate and return value of Hill sphere */

    return pow((mass_i + mass_c)/(3.0*(p->starmass + mass_i + mass_c)),
               (1.0/3.0))*p->avga;
    }

const char *myBasename(const char *path)
{
    char *p;

    assert(path);
    p = strrchr(path,'/');
    if (p) return p + 1;
    else return path;
    }

int myNewExt(const char *infile,const char *inext,
             char *outfile,const char *outext)
```

```
{
    /* adds (or replaces) extension to filename */

    const char *basename;
    char *c;
    size_t n;

    assert(infile && inext && outfile && outext);
    basename = myBasename(infile);
    if ((c = strrchr(basename,'.')) && strstr(c,inext))
        n = c - basename;
    else
        n = strlen(basename);
    if (n + strlen(outext) >= (size_t) MAXPATHLEN)
        return 1;
    (void) strncpy(outfile,basename,n); /* not null terminated */
    (void) strcpy(outfile + n,outext);
    return 0;
    }

void add_to_list(COMPDATA *prim,COMPDATA *sat,BINARY **list,
                 long *list_size,long *list_posn)
{
    /* Add bound system to list */
    COMPDATA *temp;

    if (*list_posn >= *list_size) {
        (void) printf("Growing list space\n");
        if (*list_size == 0) {
            assert(*list == NULL);
            *list_size = BUF_SIZE_INIT;
            }
        else {
            assert(*list != NULL);
            *list_size *= BUF_SIZE_MULT;
            }
        *list = (BINARY *) realloc((void *) (*list),
        (*list_size)*sizeof(BINARY));
        assert(*list != NULL);
        (void) printf("New list size = %li\n",*list_size);
        }
    if ((HAS_COM_MBR(prim) || HAS_COM_MBR(sat)) &&
        prim->data.mass < sat->data.mass) {
        temp = prim;
```

```
        prim = sat;
        sat = temp;
        } /*DEBUG*/
    /* make sure that normal particle */
    else if (!HAS_COM_MBR(prim) && !HAS_COM_MBR(sat)) {
        /* binaries have prim > sat mass */
        assert(prim->data.mass >= sat->data.mass);
        }
    (*list)[*list_posn].prim = prim;
    (*list)[*list_posn].sat = sat;
    ++(*list_posn);
    }

void find_companion(const PARAMS *p,const NODE *node,COMPDATA *part,
                    BINARY **list,long *list_size,long *list_posn)
{
    /* Identifies particles with speeds less than the escape speed */

    const SSDATA *pd,*ld;
    VECTOR r,v;
    double r2,v2;
    int i;

    pd = &part->data;

    for (i=0;i<CHILD_PER_NODE;i++)
        if (node->child[i] != NULL) {
            SUB_VEC(node->child[i]->pos,pd->pos,r);
            r2 = MAG_SQ(r);
            assert(r2 > 0.0);
            if (node->child[i]->eff_size_sq/r2 > p->OpenAng)
                find_companion(p,node->child[i],part,list,list_size,
                               list_posn);
            else {
                SUB_VEC(node->child[i]->vel,pd->vel,v); /*v=rel vel*/
                v2 = MAG_SQ(v);
                /* do it this way to avoid sqrt()s... */
                if (v2*v2 < 4.0*SQ(pd->mass + node->child[i]->mass)/r2)
                    find_companion(p,node->child[i],part,list,
                                   list_size,list_posn);
                }
            }
        else if (node->leaf[i] != NULL && node->leaf[i] != part &&
                 !IS_COM_MBR(node->leaf[i])) {
```

```
            /* (note: member check in previous line only necessary */
            /* for hierarchical search) */
            ld = &node->leaf[i]->data;
            if (!HAS_COM_MBR(part) && (ld->mass > pd->mass ||
                (ld->mass == pd->mass && node->leaf[i]->index <
                part->index))) {
                 continue; /* to prevent double counting, but only */
                           /* for non-hierarchical searching case  */
                }
            SUB_VEC(pd->pos,ld->pos,r);
            SUB_VEC(pd->vel,ld->vel,v);
            r2 = MAG_SQ(r);
            assert(r2 > 0.0);
            v2 = MAG_SQ(v);
            if (v2*v2 < 4.0*SQ(pd->mass + ld->mass)/r2)
                add_to_list(part,node->leaf[i],list,list_size,
                            list_posn);
            }
    }

void get_com_vel(NODE *node)
{
    /*gets vel moments, com vel of node, and total mass in node*/

    VECTOR v;
    double m;
    int i;

    node->mass = 0.0;
    ZERO_VEC(node->vel);
    node->n_part = 0;
    for (i=0;i<CHILD_PER_NODE;i++){
        if (node->child[i]) {
            get_com_vel(node->child[i]);
            m = node->child[i]->mass;
            node->mass += m;
            COPY_VEC(node->child[i]->vel,v);
            SCALE_VEC(v,m);
            ADD_VEC(node->vel,v,node->vel);
            node->n_part += node->child[i]->n_part;
            }
        else if (node->leaf[i]) {
            m = node->leaf[i]->data.mass;
            node->mass += m;
```

```
            COPY_VEC(node->leaf[i]->data.vel,v);
            SCALE_VEC(v,m);
            ADD_VEC(node->vel,v,node->vel);
            ++node->n_part;
            }
        }
    assert(node->mass > 0.0);
    NORM_VEC(node->vel,node->mass);
    assert(node->n_part > 0);
    }

void make_node(const VECTOR pos,double size,NODE **node)
{
    /* creates new nodes for the tree */

    int i;

    assert(size > 0.0);

    *node = (NODE *) malloc(sizeof(NODE)); /*make space for a node*/
    assert(*node != NULL);

    COPY_VEC(pos,(*node)->pos);
    ZERO_VEC((*node)->vel);
    (*node)->mass = 0.0;
    (*node)->size = size;
    (*node)->half_size = (*node)->eff_half_size = 0.5*size;
    assert((*node)->size > 0.0); /* check for underflow */
    (*node)->eff_size_sq = SQ(size);
    assert((*node)->eff_size_sq > 0.0); /* ditto */

    for (i=0;i<CHILD_PER_NODE;i++) {
        /* set children and leaf pointers to null*/
        (*node)->child[i] = NULL;
        (*node)->leaf[i] = NULL;
        }
    }

void add_to_tree(NODE *node,COMPDATA *p)
{
    /* adds particles to tree */

    int i,idx,idy,idz;
    /* locates the particle in one */
```

```
    idx = (p->data.pos[X] < node->pos[X] ? -1 : 1);
    /* of eight quadrants */
    idy = (p->data.pos[Y] < node->pos[Y] ? -1 : 1);
    idz = (p->data.pos[Z] < node->pos[Z] ? -1 : 1);

    /* sets i=0-7 depending on quadrant*/
    i = (idx + 1)/2 + (idy + 1 + 2*(idz + 1));
    if (node->child[i])  /*if node contains children open the node*/
        add_to_tree(node->child[i],p);
        /* if node already contains a particle*/
        else if (node->leaf[i]) {
        VECTOR v;                  /*create children */
        SET_VEC(v,idx,idy,idz);
        SCALE_VEC(v,0.5*node->half_size);
        ADD_VEC(v,node->pos,v);
        make_node(v,node->half_size,&(node->child[i]));
        add_to_tree(node->child[i],node->leaf[i]);
        add_to_tree(node->child[i],p);
        node->leaf[i] = NULL;
        }
    else {
        /* if particle is in an empty node make it a leaf */
        node->leaf[i] = p;
        if (p->data.radius > node->eff_half_size) {
            /*if particle is large make cell*/
            /* large - scales with mass*/
            node->eff_half_size = p->data.radius;
            node->eff_size_sq = 4.0*SQ(p->data.radius);
            }
        }
    }

void kill_node(NODE *node)
{
    /* nodes are no longer needed: release memory used for nodes */

    int i;

    assert(node != NULL);

    for (i=0;i<CHILD_PER_NODE;i++)
        if (node->child[i])
            kill_node(node->child[i]);
```

```
    free((void *) node);
    }

int read_data(const PARAMS *p,const char *file_in,COMPDATA **part,
              long *n,double *m_tot,VECTOR root_center,
              double *root_size)
{
    /* read data from ss file */

    SSDATA *d;
    FILE *fp; /* for txt & bin file types only */
    double xmin,ymin,zmin,xmax,ymax,zmax;
    long i;

#ifdef SS_CORE
    SSIO ssio; /* for ss file type only */
    SSHEAD h;
#endif

    switch (p->FileType) {
    case FileTypeTxt:
    case FileTypeBin:
        if ((fp = fopen(file_in,"r"))  == NULL) {
            (void) fprintf(stderr,"Problem opening file %s\n",file_in);
            return 1;
            }
        *n = BUF_NPART_INIT;
        break;
#ifdef SS_CORE
    case FileTypeSS:
        if (ssioOpen(file_in,&ssio,SSIO_READ)) {
            (void) fprintf(stderr,"Unable to open %s for reading\n",
                           file_in);
            return 1;
            }
        if (ssioHead(&ssio,&h)){
            (void) fprintf(stderr,"Corrupt header\n");
            (void) ssioClose(&ssio);
            return 1;
            }
        if (h.n_data <= 0) {
            (void) fprintf(stderr,"Invalid file size\n");
            (void) ssioClose(&ssio);
            return 1;
```

```
            }
        *n = h.n_data;
        break;
#endif
    default:
        (void) fprintf(stderr,"file type %i is invalid\n",p->FileType);
        return 1;
        }

    /* allocate space for part */
    *part = (COMPDATA *) malloc((*n)*sizeof(COMPDATA));
    assert(*part != NULL); /*make sure space allocation worked*/

    *m_tot = 0.0;

    switch (p->FileType) {
    case FileTypeTxt:
    case FileTypeBin:
        i=0;
        while (feof(fp) == 0) {
            if (i>=(*n)) {
                (void) printf("Growing particle space\n");
                *n *= BUF_SIZE_MULT;
                *part = (COMPDATA *) realloc((void *) (*part),
                        (*n)*sizeof(COMPDATA));
                assert(*part != NULL);
                (void) printf("New data space = %li\n",*n);
                }
            d = &((*part)[i].data);
            switch (p->FileType) {
            case FileTypeTxt:
                if (fscanf(fp,"%lf%lf%lf%lf%lf%lf%lf\n",
                           &d->mass,&d->radius,
                           &d->pos[X],&d->pos[Y],&d->pos[Z],
                           &d->vel[X],&d->vel[Y],&d->vel[Z]) != 8)
                           goto error;
                break;
            case FileTypeBin:
                if (fread(&d->mass,sizeof(double),1,fp) != 1)
                    goto error;
                if (fread(&d->radius,sizeof(double),1,fp) != 1)
                    goto error;
                if (fread(d->pos,sizeof(double),3,fp) != 3) goto error;
                if (fread(d->vel,sizeof(double),3,fp) != 3) goto error;
```

```
                break;
            default:
                assert(0); /* should never get here */
                }
            d->mass = d->mass/p->InMassConv;
            d->radius = d->radius/p->InLengthConv;
            NORM_VEC(d->pos,p->InLengthConv);
            NORM_VEC(d->vel,p->InLengthConv);
            d->org_idx = i;
            *m_tot += d->mass;
            ++i;
            }
        *n=i;
        /* release unused buffer space */
        *part = (COMPDATA *) realloc((void *) (*part),
                (*n)*sizeof(COMPDATA));
        assert(*part != NULL);
        (void) printf("Number of particles = %li\n",*n);
        break;
#ifdef SS_CORE
    case FileTypeSS:
        for (i=0;i<*n;i++) {
            d = &((*part)[i].data);
            if (ssioData(&ssio,d) != 0) {
                (void) fprintf(stderr,"Corrupt data\n");
                (void) ssioClose(&ssio);
                return 1;
                }
#ifdef SUPPORT_OLD_FORMAT
            if (d->org_idx == -1) /* assign index for each particle */
                d->org_idx = i;
#endif
            *m_tot += d->mass;
            }
        break;
#endif /* SS_CORE */
    default:
        assert(0); /* invalid file type */
        }

    /* pad storage -- see comment at top of file */
    *part = (COMPDATA *) realloc((void *) (*part),
            (*n)*EXTRA_STORE*sizeof(COMPDATA));
```

```
    xmin = ymin = zmin = DBL_MAX;
    xmax = ymax = zmax = - DBL_MAX;
    for (i=0;i<*n;i++) {
        (*part)[i].prim = (*part)[i].sat = NULL;
        (*part)[i].com = NULL;
        (*part)[i].index = i;
        d = &((*part)[i].data);
        /* find max extent of particles */
        if (d->pos[X] < xmin) xmin = d->pos[X];
        /* for size of root cell */
        if (d->pos[Y] < ymin) ymin = d->pos[Y];
        if (d->pos[Z] < zmin) zmin = d->pos[Z];
        if (d->pos[X] > xmax) xmax = d->pos[X];
        if (d->pos[Y] > ymax) ymax = d->pos[Y];
        if (d->pos[Z] > zmax) zmax = d->pos[Z];
        }

    *root_size = xmax - xmin;
    if (ymax - ymin > *root_size) *root_size = ymax - ymin;
    if (zmax - zmin > *root_size) *root_size = zmax - zmin;
    SET_VEC(root_center,(xmin + xmax)/2,(ymin + ymax)/2,
           (zmin + zmax)/2);
    (void) printf("root center = (%g,%g,%g), size = %g\n",
                  root_center[X],root_center[Y],root_center[Z],
                  *root_size);

    switch (p->FileType) {
    case FileTypeTxt:
    case FileTypeBin:
        fclose(fp);
        break;
#ifdef SS_CORE
    case FileTypeSS:
        (void) ssioClose(&ssio);
        break;
#endif
    default:
        assert(0);
        }

    (void) printf("%li particle%s read\n",*n,*n==1 ? "" : "s");

    return 0;
```

```
 error:
    (void) fprintf(stderr,"Error during read.\n");
    (void) fclose(fp);
    return 1;
    }

void make_com_part(BINARY *tightest,long *n_part,long *part_buf_size,
                   COMPDATA **part)
{
    /*
     ** Creates com particle, adds it to the particle list, modifies
     ** primary and satellite structures so they will no longer be
     ** used in the search tree.
     */

    const SSDATA *ptr_prim,*ptr_sat;
    SSDATA *ptr_com;
    COMPDATA *comp_prim,*comp_sat,*comp_com;
    BIN_STATS bs;
    VECTOR v1,v2;

    /* check if buffer needs to grow */

    if (*n_part == *part_buf_size) {
        (void) printf("Growing particle list space\n");
        /* for now, particle list not allowed to grow -- see comment */
        /* at top */
        assert(0); /* particle realloc() forbidden */
        *part_buf_size *= BUF_SIZE_MULT;
        *part = (COMPDATA *) realloc((void *) (*part),
                (*part_buf_size)*sizeof(COMPDATA));
        assert(*part != NULL);
        (void) printf("New part list size = %li\n",*part_buf_size);
        }

    /* abbreviation for COMPDATA */

    comp_prim = tightest->prim;
    comp_sat = tightest->sat;
    comp_com = &((*part)[*n_part]); /* new com particle */

    /* abbreviation for SSDATA */

    ptr_prim = &comp_prim->data;
```

```
ptr_sat = &comp_sat->data;
ptr_com = &comp_com->data;

/* store pointers */

comp_com->com = NULL;
comp_com->prim = comp_prim;
comp_com->sat = comp_sat;
comp_com->index = *n_part;

comp_prim->com = comp_com;
comp_sat->com = comp_com;

++(*n_part); /* particle list grows by one for com particle */

ptr_com->mass = ptr_prim->mass + ptr_sat->mass;
calc_bin_stats(ptr_prim,ptr_sat,BIN_SEMI,&bs);
/* com "radius" is semimajor axis of binary */
ptr_com->radius = bs.a;

/* com position */

COPY_VEC(ptr_prim->pos,v1);
SCALE_VEC(v1,ptr_prim->mass);
COPY_VEC(ptr_sat->pos,v2);
SCALE_VEC(v2,ptr_sat->mass);
ADD_VEC(v1,v2,ptr_com->pos);
NORM_VEC(ptr_com->pos,ptr_com->mass);

/* com velocity */

COPY_VEC(ptr_prim->vel,v1);
SCALE_VEC(v1,ptr_prim->mass);
COPY_VEC(ptr_sat->vel,v2);
SCALE_VEC(v2,ptr_sat->mass);
ADD_VEC(v1,v2,ptr_com->vel);
NORM_VEC(ptr_com->vel,ptr_com->mass);

/* com spin zeroed (could store orbital ang vel instead) */

ZERO_VEC(ptr_com->spin);

/* cycle colors */
```

```
    ptr_com->color = 2 + (ptr_prim->color + ptr_sat->color - 2)%14;

    /* assign index of primary */

    ptr_com->org_idx = ptr_prim->org_idx;
    }

BOOLEAN ok_to_cut(const PARAMS *p,const COMPDATA *prim,
                  const COMPDATA *sat,const BIN_STATS *bs)
{
    /* returns 1 if any list cut criteria met, 0 otherwise */

    if ((p->DoPeriCut && !HAS_COM_MBR(prim) &&
         bs->q < (p->PeriCut < 0.0 ? - p->PeriCut*prim->data.radius :
                  p->PeriCut == 0.0 ? prim->data.radius +
                  sat->data.radius : p->PeriCut)) ||
                  (p->DoOrbitCut && HAS_COM_MBR(prim) &&
         bs->q < (p->OrbitCut < 0.0 ? - p->OrbitCut*prim->data.radius :
                  p->OrbitCut == 0.0 ? prim->data.radius +
                  (HAS_COM_MBR(sat) ? sat->data.radius : 0.0) :
                  p->OrbitCut*prim->data.radius)) ||
         bs->e >= p->EccCut || bs->E >= p->EngCut ||
         (p->HillCut > 0.0 && bs->a >= p->HillCut*
         hill(p,prim->data.mass,sat->data.mass)))
        return 1;
    else
        return 0;
    }

void cut_list(const PARAMS *p,BINARY *list,long *list_length)
{
    /* cuts out all systems with q<, e>, a>, eng> params recursively */

    SSDATA *pd,*sd;
    BIN_STATS bs;
    long i,n;
    BOOLEAN *flag;

    flag = (BOOLEAN *) malloc((*list_length)*sizeof(BOOLEAN));

    /* removes binaries from list */
    for (i=0;i<(*list_length);i++) {
        pd = &list[i].prim->data;
        sd = &list[i].sat->data;
```

```
        calc_bin_stats(pd,sd,BIN_CUT,&bs);
        flag[i] = ok_to_cut(p,list[i].prim,list[i].sat,&bs);
        }

    /* shrinks list and counts up total number of surviving binaries */
    n = 0;
    for (i=0;i<(*list_length);i++)
        if (flag[i] == 0) {
            if (n < i) list[n] = list[i];
            n++;
            }

    (*list_length) = n;

    free((void *) flag);
    }

int extract(const PARAMS *p,const char *filename,const BINARY *list,
            long nbin)
{
    /* extracts system with index = primary index and creates new */
    /* data file */

    SSDATA *part;
    FILE *fp; /* for txt & bin file types only */
#ifdef SS_CORE
    SSHEAD head; /* for ss file type only */
    SSIO ssio_out;
#endif
    long n,i;
    int primary_done;

    (void) printf("Extracting system...\n");

    switch (p->FileType) {
    case FileTypeTxt:
    case FileTypeBin:
        if ((fp = fopen(filename,"w")) == NULL) {
            (void) fprintf(stderr,"Can't open %s\n",filename);
            return 1;
            }
        break;
#ifdef SS_CORE
    case FileTypeSS:
```

```
        if (ssioOpen(filename,&ssio_out,SSIO_WRITE)) {
            (void) fprintf(stderr,"Unable to open %s for writing\n",
            filename);
            return 1;
            }
        break;
#endif
    default:
        (void) fprintf(stderr,"file type %i is invalid\n",p->FileType);
        return 1;
        }

    for (n=i=0;i<nbin;i++)
        if (list[i].prim->index == p->ExtIdx)
            ++n;

    if (n == 0) {
        (void) fprintf(stderr,"Index number %li not found\n",
                       p->ExtIdx);
        return 1;
        }

#ifdef SS_CORE
    if (p->FileType == FileTypeSS) {
        head.time = 0.0;
        head.n_data = n + 1; /* one primary plus n satellites */
        head.pad = -1;

        if (ssioHead(&ssio_out,&head)) {
            (void) fprintf(stderr,"Unable to write header.\n");
            return 1;
            }
        }
#endif

    primary_done = 0;
    for (i=0;i<nbin;i++)
        if (list[i].prim->index == p->ExtIdx) {
            if (!primary_done) {
                part = &list[i].prim->data;
                primary_done = 1;
                --i; /* go back and do satellite */
                }
            else
```

```
                part = &list[i].sat->data;
            switch (p->FileType) {
            case FileTypeTxt:
                if (fprintf(fp,"%e %e %e %e %e %e %e %e\n",
                            part->mass,part->radius,
                            part->pos[0],part->pos[1],part->pos[2],
                            part->vel[0],part->vel[1],part->vel[2])
                            < 1)
                    goto error;
                break;
            case FileTypeBin:
                if (fwrite(&part->mass,sizeof(double),1,fp) != 1)
                    goto error;
                if (fwrite(&part->radius,sizeof(double),1,fp) != 1)
                    goto error;
                if (fwrite(part->pos,sizeof(double),3,fp) != 3)
                    goto error;
                if (fwrite(part->vel,sizeof(double),3,fp) != 3)
                    goto error;
                break;
#ifdef SS_CORE
            case FileTypeSS:
                if (ssioData(&ssio_out,part) != 0) {
                    (void) fprintf(stderr,"Error writing particle %li.\n",
                                   list[i].prim->index);
                    return 1;
                    }
                break;
#endif
            default:
                (void) fprintf(stderr,"file type %i is invalid\n",
                               p->FileType);
                return 1;
                }
            }

    switch (p->FileType) {
    case FileTypeTxt:
    case FileTypeBin:
        fclose(fp);
        break;
#ifdef SS_CORE
    case FileTypeSS:
        (void) ssioClose(&ssio_out);
```

```
        break;
#endif
    default:
        assert(0);
        }

    return 0;

 error:
    (void) fprintf(stderr,"Error during write.\n");
    (void) fclose(fp);
    return 1;
    }


#ifdef SS_CORE
int find_real_part(const PARAMS *p,FILE *fp,SSIO *ssio_out,
                   COMPDATA *part,long *n)
#else
int find_real_part(const PARAMS *p,FILE *fp,COMPDATA *part,long *n)
#endif
{
    if (part->prim == NULL) { /*reached bottom of tree*/
        ++*n;
        switch (p->FileType) {
        case FileTypeTxt:
            if (fprintf(fp,"%e %e %e %e %e %e %e\n",
                        part->data.mass,part->data.radius,
                        part->data.pos[0],part->data.pos[1],
                        part->data.pos[2],part->data.vel[0],
                        part->data.vel[1],part->data.vel[2]) < 1)
                goto error;
            break;
        case FileTypeBin:
            if (fwrite(&part->data.mass,sizeof(double),1,fp) != 1)
                goto error;
            if (fwrite(&part->data.radius,sizeof(double),1,fp) != 1)
                goto error;
            if (fwrite(part->data.pos,sizeof(double),3,fp) != 3)
                goto error;
            if (fwrite(part->data.vel,sizeof(double),3,fp) != 3)
                goto error;
            break;
#ifdef SS_CORE
```

```
        case FileTypeSS:
            if (ssioData(ssio_out,&part->data) != 0) {
                (void) fprintf(stderr,"Error writing particle %li.\n",part->index)
                return 1;
                }
            break;
#endif
        default:
            (void) fprintf(stderr,"file type %i is invalid\n",
                           p->FileType);
            return 1;
            }

        return 0;
        }

    if (p->ApplyHierCut) {
        BIN_STATS bs;
        const SSDATA *pd,*sd;

        pd = &part->prim->data;
        sd = &part->sat->data;

        calc_bin_stats(pd,sd,BIN_CUT|BIN_INCL|BIN_PERIOD,&bs);
        if (ok_to_cut(p,part->prim,part->sat,&bs))
            return 0;
        }
#ifdef SS_CORE
    if (find_real_part(p,fp,ssio_out,part->prim,n) != 0) return 1;
    if (find_real_part(p,fp,ssio_out,part->sat,n) != 0) return 1;
#else
    if (find_real_part(p,fp,part->prim,n) != 0) return 1;
    if (find_real_part(p,fp,part->sat,n) != 0) return 1;
#endif
    return 0;

 error:
    (void) fprintf(stderr,"Error during write.\n");
    (void) fclose(fp);
    return 1;
    }

int hier_extract(const PARAMS *p,const char *filename,COMPDATA *part)
{
```

```
    /* extracts hierarchy system with top com particle index */
    /* specified by user */
    long n;
    FILE *fp; /* for txt & bin file types only */
#ifdef SS_CORE
    SSHEAD head; /* for ss file type only */
    SSIO ssio_out;
#endif

    /* check that particle past is top com part */
    if (!HAS_COM_MBR(part) || IS_COM_MBR(part)) {
        (void) fprintf(stderr,"Particle %ld is not a top center of mass
                               particle\n", part->index);
        return 1;
        }

    (void) printf("Extracting hierarchy system...\n");

    switch (p->FileType) {
    case FileTypeTxt:
    case FileTypeBin:
        if ((fp = fopen(filename,"w")) == NULL) {
            (void) fprintf(stderr,"Can't open %s\n",filename);
            return 1;
            }
        break;
#ifdef SS_CORE
    case FileTypeSS:
        if (ssioOpen(filename,&ssio_out,SSIO_WRITE)) {
            (void) fprintf(stderr,"Unable to open %s for writing\n",
            filename);
            return 1;
            }
        break;
#endif
    default:
        (void) fprintf(stderr,"file type %i is invalid\n",p->FileType);
        return 1;
        }

#ifdef SS_CORE /* write a dummy header */
    if (p->FileType == FileTypeSS) {
        head.time = 0.0;
        head.n_data = 1; /* one primary plus n satellites */
```

```
        head.pad = -1;

        if (ssioHead(&ssio_out,&head)) {
            (void) fprintf(stderr,"Unable to write dummy header.\n");
            return 1;
            }
        }
#endif
    n = 0; /* number of real particles */

#ifndef SS_CORE
    find_real_part(p,fp,part,&n);
#else
    find_real_part(p,fp,&ssio_out,part,&n);
    if (p->FileType == FileTypeSS) {
        ssioRewind(&ssio_out);
        head.time = 0.0;
        head.n_data = n;
        head.pad = -1;

        if (ssioHead(&ssio_out,&head)) {
            (void) fprintf(stderr,"Unable to write true header.\n");
            return 1;
            }
        }
#endif
    (void) printf("Found %ld real particles in extracted system\n",n);

    switch (p->FileType) {
    case FileTypeTxt:
    case FileTypeBin:
        fclose(fp);
        break;
#ifdef SS_CORE
    case FileTypeSS:
        (void) ssioClose(&ssio_out);
        break;
#endif
    default:
        assert(0);
        }

    return 0;
    }
```

```
int write_tipsy(const char *filename,long npart,const BINARY *list,
                long nbin)
{
    /* creates a tipsy vector file of binary energy to use with tipsy */
    /* software, NOTE: this function assumes list has been sorted */
    /* properly! */

    BIN_STATS bs;
    FILE *fp;
    long i,j;

    if ((fp = fopen(filename,"w")) == NULL) {
        (void) fprintf(stderr,"Can't open %s\n",filename);
        return 1;
        }

    (void) printf("Creating tipsy file\n");
    if (fprintf(fp,"%li\n",npart) < 1) goto error;
    for (i=0;i<npart;i++) {
        bs.E = 0.0;
        for (j=0;j<nbin;j++)
            /* should be most bound if list sorted */
            if (list[j].prim->index == i) {
                calc_bin_stats(&list[j].prim->data,&list[j].sat->data,
                               BIN_ENERGY,&bs);
                break;
                }
        if (fprintf(fp,"%e\n",bs.E) < 1) goto error;
        }

    (void) fclose(fp);

    return 0;

 error:
    (void) fprintf(stderr,"Error during write.\n");
    (void) fclose(fp);
    return 1;
    }

void usage(const char *progname)
{
    /* explains usage of companion and flag options */
```

```
    (void) printf("Usage: %s [-H [-z cutoff] [-g index[-a]]|-t]
                   [-c|-m|-s] [-C|-M|-S] [ -f filetype ]
                   [ -e cutoff ] [ -E cutoff ]\n",
                   progname);
    (void) printf("       [ -h cutoff ] [ -q cutoff ] [ -o angle ]
                   [ -x index ] file [ file ... ]\n");
    (void) printf("\n");
    (void) printf("Options: -H = search for hierarchies\n");
    (void) printf("         -z = close approach cutoff for center of
                   mass particles (0 to eliminate orbit crossers,\n");
    (void) printf("              < 0 for semimajor axis\n");
    (void) printf("         -g = extract hierarchy system\n");
    (void) printf("         -a = apply cuts to extraction of hierachy
                   system\n");
    (void) printf("         -t = create Tipsy vector file of binding
                   energy\n");
    (void) printf("         -c | -m | -s = input in cgs, mks or system
                   units (default \"%s\")\n",
#ifdef SS_CORE
                   "system");
#else
                   "cgs");
#endif
    (void) printf("         -C | -M | -S = output in cgs, mks or system
                   units (default \"%s\")\n",
#ifdef SS_CORE
                   "system");
    (void) printf("         -f = file type: plain text (\"%s\"), binary
                   (\"%s\"), or \"%s\" (default\"%s\")\n",
                   FileTypeStr[FileTypeTxt],FileTypeStr[FileTypeBin],
                  FileTypeStr[FileTypeSS],FileTypeStr[DFLT_FILE_TYPE]);
#else
                   "cgs");
    (void) printf("         -f = file type: plain text (\"%s\") or binary
                   (\"%s\") (default \"%s\")\n",
                   FileTypeStr[FileTypeTxt],FileTypeStr[FileTypeBin],
                   FileTypeStr[DFLT_FILE_TYPE]);
#endif
    (void) printf("         -e = eccentricity cutoff\n");
    (void) printf("         -E = binding energy cutoff\n");
    (void) printf("         -h = Hill sphere cutoff, in Hill radii
                   (prompts for semimajor axis and star mass)\n");
    (void) printf("         -q = close approach cutoff for normal
```

```
                particles (0 to eliminate colliders, < 0 for\n");
    (void) printf("              primary radii)\n");
    (void) printf("         -o = opening angle (default %g rad)\n",
                DFLT_OPEN_ANGLE);
    (void) printf("         -x = extracts system of given primary
                index\n");
    (void) printf("\n");
    (void) printf("NOTE: cutoff limits taken to be in output units
                where applicable.\n");

    exit(1);
    }

void set_defaults(PARAMS *params)
{
    /* parameters defaults */

    params->Hier = DFLT_HIER;
    params->TipsyFile = DFLT_TIPSY_FILE;
    params->InCgsUnits = DFLT_IN_CGS;
    params->InMksUnits = DFLT_IN_MKS;
    params->InSysUnits = DFLT_IN_SYS;
    params->OutCgsUnits = DFLT_OUT_CGS;
    params->OutMksUnits = DFLT_OUT_MKS;
    params->OutSysUnits = DFLT_OUT_SYS;
    params->FileType = DFLT_FILE_TYPE;
    params->EccCut = DFLT_ECC_CUT;
    params->EngCut = DFLT_ENG_CUT;
    params->HillCut = DFLT_HILL_CUT;
    params->PeriCut = DFLT_PERI_CUT;
    params->OrbitCut = DFLT_ORBIT_CUT;
    params->OpenAng = DFLT_OPEN_ANGLE;
    params->HierExtIdx = DFLT_HIER_EXTRACT_INDEX;
    params->ExtIdx = DFLT_EXTRACT_INDEX;

    /* derived parameters that need to be preset */

    params->DoPeriCut = DFLT_DO_PERI_CUT;
    params->DoOrbitCut = DFLT_DO_ORBIT_CUT;

    /* default units -- conversions between sys units and default units */

    params->InLengthConv = params->OutLengthConv = DFLT_LENGTH_CONV;
    params->InMassConv = params->OutMassConv = DFLT_MASS_CONV;
```

```
    params->InTimeConv = params->OutTimeConv = DFLT_TIME_CONV;
    }

void parse_in(int argc,char *argv[],PARAMS *p)
{
    /* in case unistd.h unavailable */
    extern int getopt(int,char *const *,const char *);
    extern int optind;
    extern char *optarg;

    char file_ext[FILE_TYPE_STR_MAX_LEN];
    int c;

    (void) strncpy(file_ext,FileTypeStr[DFLT_FILE_TYPE],
                   FILE_TYPE_STR_MAX_LEN);

    while ((c = getopt(argc,argv,"HatcmsCMSz:g:f:e:E:h:q:o:x:")) !=
            EOF)
        switch (c) {
        case 'H':
            p->Hier = TRUE;
            break;
        case 'a':
            p->ApplyHierCut = TRUE;
            break;
        case 't':
            p->TipsyFile = TRUE;
            break;
        case 'c':
            p->InCgsUnits = TRUE;
#ifdef SS_CORE
            if (p->InSysUnits)
                p->InSysUnits = FALSE;
#endif
            break;
        case 'm':
            p->InMksUnits = TRUE;
#ifndef SS_CORE
            if (p->InCgsUnits)
                p->InCgsUnits = FALSE;
#else
            if (p->InSysUnits)
                p->InSysUnits = FALSE;
#endif
```

```
            break;
        case 's':
            p->InSysUnits = TRUE;
#ifndef SS_CORE
            if (p->InCgsUnits)
                p->InCgsUnits = FALSE;
#endif
            break;
        case 'C':
            p->OutCgsUnits = TRUE;
#ifdef SS_CORE
            if (p->OutSysUnits)
                p->OutSysUnits = FALSE;
#endif
            break;
        case 'M':
            p->OutMksUnits = TRUE;
#ifndef SS_CORE
            if (p->OutCgsUnits)
                p->OutCgsUnits = FALSE;
#else
            if (p->OutSysUnits)
                p->OutSysUnits = FALSE;
#endif
            break;
        case 'S':
            p->OutSysUnits = TRUE;
#ifndef SS_CORE
            if (p->OutCgsUnits)
                p->OutCgsUnits = FALSE;
#endif
            break;
        case 'z':
            p->OrbitCut = atof(optarg);
            p->DoOrbitCut = TRUE;
            break;
        case 'g':
            p->HierExtIdx = atoi(optarg);
            break;
        case 'f':
            strcpy(file_ext,optarg);
            break;
        case 'e':
            p->EccCut = atof(optarg);
```

```
            break;
        case 'E':
            p->EngCut = atof(optarg);
            break;
        case 'h':
            p->HillCut = atof(optarg);
            break;
        case 'q':
            p->PeriCut = atof(optarg);
            p->DoPeriCut = TRUE;
            break;
        case 'o':
            p->OpenAng = atof(optarg);
            break;
        case 'x':
            p->ExtIdx = atoi(optarg);
            break;
        case '?':
        default:
            usage(argv[0]);
            }

    if (optind >= argc)
        usage(argv[0]);

    if (strcmp(file_ext,FileTypeStr[FileTypeTxt]) == 0)
        p->FileType = FileTypeTxt;
    else if (strcmp(file_ext,FileTypeStr[FileTypeBin]) == 0)
        p->FileType = FileTypeBin;
#ifdef SS_CORE
    else if (strcmp(file_ext,FileTypeStr[FileTypeSS]) == 0)
        p->FileType = FileTypeSS;
#endif
    else
        usage(argv[0]);

    /* sanity checks */

    if (p->Hier && p->TipsyFile)
        usage(argv[0]);

    if ((p->InCgsUnits == TRUE && p->InMksUnits == TRUE) ||
        (p->InCgsUnits == TRUE && p->InSysUnits == TRUE) ||
        (p->InMksUnits == TRUE && p->InSysUnits == TRUE) ||
```

```
    (p->OutCgsUnits == TRUE && p->OutMksUnits == TRUE) ||
    (p->OutCgsUnits == TRUE && p->OutSysUnits == TRUE) ||
    (p->OutMksUnits == TRUE && p->OutSysUnits == TRUE))
    usage(argv[0]);

if (p->EccCut < 0.0 || p->EccCut > 1.0) {
    (void) fprintf(stderr,"Eccentricity cut must be between 0
                           and 1.\n");
    exit(1);
    }

if (p->HillCut < 0.0) {
    (void) fprintf(stderr,"Hill sphere cut must be positive.\n");
    exit(1);
    }

if (p->EngCut > 0.0) {
    (void) fprintf(stderr,"Energy cut must be negative.\n");
    exit(1);
    }

if (p->DoOrbitCut == TRUE && p->Hier == FALSE) {
    (void) fprintf(stderr,"Close approach cut for center of mass
                           particles must be used with
                           hierarchy\n");
    exit(1);
    }

if (p->OpenAng < 0.0) {
    (void) fprintf(stderr,"Opening angle must be positive or
                           zero.\n");
    exit(1);
    }

if (p->HierExtIdx >= 0 && p->Hier == FALSE) {
    (void) fprintf(stderr,"Hierarchy extraction must be used with
                           hierarchy option\n");
    exit(1);
    }

if (p->ApplyHierCut == TRUE && p->HierExtIdx >= 0) {
    (void) fprintf(stderr,"ApplyHierCut option must be used with
                           hierarchy extraction\n");
    exit(1);
```

```
    }

if (p->HierExtIdx < -1) { /* -1 is default, i.e., no extraction */
    (void) fprintf(stderr,"Hierarchy extraction index must be
                           non-negative.\n");
    exit(1);
    }

if (p->ExtIdx < -1) { /* -1 is default, i.e., no extraction */
    (void) fprintf(stderr,"Extraction index must be non-
                           negative.\n");
    exit(1);
    }

/* Default I/O in cgs units (sys if SS_CORE defined); data stored */
/* internally in system units */

if (p->InCgsUnits == TRUE) {
    p->InLengthConv = AU*1.0e2;
    p->InMassConv = M_SUN*1.0e3;
    p->InTimeConv = SID_YR/TWO_PI;
    (void) printf("Input in cgs units.\n");
} else if (p->InMksUnits == TRUE) {
    p->InLengthConv = AU;
    p->InMassConv = M_SUN;
    p->InTimeConv = SID_YR/TWO_PI;
    (void) printf("Input in mks units.\n");
} else if (p->InSysUnits == TRUE){
    p->InLengthConv = 1.0;
    p->InMassConv = 1.0;
    p->InTimeConv = 1.0;
    (void) printf("Input in system units.\n");
    }
/* NOT USED */
p->InEnergyConv = p->InMassConv*SQ(p->InLengthConv/p->InTimeConv);

if (p->OutCgsUnits == TRUE) {
    p->OutLengthConv = AU*1.0e2;
    p->OutMassConv = M_SUN*1.0e3;
    p->OutTimeConv = SID_YR/TWO_PI;
    (void) printf("Output in cgs units.\n");
} else if (p->OutMksUnits == TRUE) {
    p->OutLengthConv = AU;
    p->OutMassConv = M_SUN;
```

```
        p->OutTimeConv = SID_YR/TWO_PI;
        (void) printf("Output in mks units.\n");
    } else if (p->OutSysUnits == TRUE) {
        p->OutLengthConv = 1.0;
        p->OutMassConv = 1.0;
        p->OutTimeConv = 1.0;
        (void) printf("Output in sys units.\n");
        }

    p->OutEnergyConv = p->OutMassConv*SQ(p->OutLengthConv/
                       p->OutTimeConv);

    p->OpenAng = SQ(p->OpenAng); /* store square of opening angle */

    /* convert cuts to system units as needed */

    p->EngCut /= p->OutEnergyConv;

    if (p->HillCut > 0.0){
        (void) printf("What is the average semimajor axis (in AU)?\n");
        (void) scanf("%lf",&(p->avga));
        if (p->avga <= 0.0) {
            (void) fprintf(stderr,"Semimajor axis must be
                                   positive.\n");
            exit(1);
            }
        (void) printf("What is the mass of the star (in M_Sun)?\n");
        (void) scanf("%lf",&(p->starmass));
        if (p->starmass <= 0.0) {
            (void) fprintf(stderr,"Star mass must be positive.\n");
            exit(1);
            }
        }

    if (p->PeriCut > 0.0)
        p->PeriCut /= p->OutLengthConv;

    if (p->OrbitCut > 0.0)
        p->OrbitCut /= p->OutLengthConv;
    }

int write_output(const PARAMS *p,const char *filename_in,
                 const BINARY *list,long nbin,double m_tot)
{
```

```
BIN_STATS bs;
const SSDATA *pd,*sd;
FILE *fp_pr,*fp_ana;
char pr_outfile[MAXPATHLEN],ana_outfile[MAXPATHLEN];
long nsys,i,last_index;

if (myNewExt(filename_in,FileTypeStr[p->FileType],pr_outfile,
    PR_EXT)) {
    (void) fprintf(stderr,"Unable to generate output filename for
                          %s.\n",filename_in);
    return 1;
    }

if (myNewExt(filename_in,FileTypeStr[p->FileType],ana_outfile,
    ANA_EXT)) {
    (void) fprintf(stderr,"Unable to generate output filename for
                          %s.\n",filename_in);
    return 1;
    }

if ((fp_pr = fopen(pr_outfile,"w")) == NULL) {
    (void) fprintf(stderr,"Can't open %s\n",pr_outfile);
    return 1;
    }

if ((fp_ana = fopen(ana_outfile,"w")) == NULL) {
    (void) fprintf(stderr,"Can't open %s\n",ana_outfile);
    return 1;
    }

(void) fprintf(fp_pr," M_p/M_t     p_ind    p_rad  M_s/M_p
                      s_ind    s_rad  bind_eng        a    e    i
                      per\n");
(void) fprintf(fp_pr,"-------- --------- -------- --------
                     --------- -------- --------- -------- ----
                     ---- --------\n");
last_index = -1;
for (nsys=i=0;i<nbin;i++) {
    pd = &list[i].prim->data;
    sd = &list[i].sat->data;
    calc_bin_stats(pd,sd,BIN_ENERGY|BIN_SEMI|BIN_ECC|BIN_INCL|
                   BIN_PERIOD,&bs);
    /* "pretty" output */
    if (list[i].prim->index != last_index) {
```

```
            if (fprintf(fp_pr,"%8.2e %9li %8.2e ",pd->mass/m_tot,
                                list[i].prim->index,
                                pd->radius*p->OutLengthConv) < 1)
               goto error;
            last_index = list[i].prim->index;
            ++nsys;
            }
        else
            (void) fprintf(fp_pr,"%28s",""); /* pad 28 spaces */
        if (fprintf(fp_pr,"%8.2e %9li %8.2e %9.2e %8.2e %4.2f %4.2f
                        %8.2e\n",
                        sd->mass/pd->mass,list[i].sat->index,
                        sd->radius*p->OutLengthConv,
                        bs.E*p->OutEnergyConv,bs.a*p->OutLengthConv,
                        bs.e,bs.i,bs.P*p->OutTimeConv) < 1)
                        goto error;
        /* machine output */
        if (fprintf(fp_ana,"%e %li %e %e %li %e %e %e %e %e %e\n",
                    pd->mass/m_tot,list[i].prim->index,
                    pd->radius*p->OutLengthConv,sd->mass/pd->mass,
                    list[i].sat->index,sd->radius*p->OutLengthConv,
                    bs.E*p->OutEnergyConv,bs.a*p->OutLengthConv,
                    bs.e,bs.i,bs.P*p->OutTimeConv) < 1) goto error;
        }

    (void) fprintf(fp_pr,"Summary: %li system%s, %li binar%s, total
                        mass considered = %e\n",nsys,nsys==1?"":"s",
                        nbin,nbin==1?"y":"ies",m_tot*p->OutMassConv);

    (void) fclose(fp_ana);
    (void) fclose(fp_pr);

    return 0;

 error:
    (void) fprintf(stderr,"Error during write.\n");
    (void) fclose(fp_ana);
    (void) fclose(fp_pr);
    return 1;
    }

int walk_system(const PARAMS *p,FILE *fp,const COMPDATA *part,
                double m_tot,BOOLEAN write_header,int *n_layer)
{
```

```
    BIN_STATS bs;
    const SSDATA *pd,*sd;
    const COMPDATA *primary,*satellite;

    if (part->prim == NULL)
        return 0;

    primary = part->prim;
    satellite = part->sat;
    pd = &primary->data;
    sd = &satellite->data;
    calc_bin_stats(pd,sd,BIN_CUT|BIN_INCL|BIN_PERIOD,&bs);
    if (ok_to_cut(p,primary,satellite,&bs))
        return 0;

    if (write_header) {
        *n_layer = 0;
        (void) fprintf(fp,"    c_ind  M_p/M_t     p_ind    p_rad
                          M_s/M_p     s_ind    s_rad  bind_eng
                          a    e    i     per\n");
        (void) fprintf(fp,"--------- -------- --------- --------
                           -------- --------- -------- ---------
                           -------- ---- ---- --------\n");
        }

    if (fprintf(fp,"%9li %8.2e %9li %8.2e %8.2e %9li %8.2e %9.2e %8.2e
                    %4.2f %4.2f %8.2e\n",
                part->index,pd->mass/m_tot,part->prim->index,
                pd->radius*p->OutLengthConv,sd->mass/pd->mass,
                part->sat->index,sd->radius*p->OutLengthConv,
                bs.E*p->OutEnergyConv,bs.a*p->OutLengthConv,
                bs.e,bs.i,bs.P*p->OutTimeConv) < 1) return 1;
    ++(*n_layer);
    if (walk_system(p,fp,part->prim,m_tot,FALSE,n_layer) != 0)
        return 1;
    if (walk_system(p,fp,part->sat,m_tot,FALSE,n_layer) != 0) return 1;

    return 0;
    }

void create_summary(const PARAMS *p,const COMPDATA *part,
                    HIER_OUTPUT *sum,double *max_a)
{
    BIN_STATS bs;
```

```
    const SSDATA *pd,*sd;

    if (part->prim == NULL)
        return;

    pd = &part->prim->data;
    sd = &part->sat->data;
    calc_bin_stats(pd,sd,BIN_CUT,&bs);
    if (ok_to_cut(p,part->prim,part->sat,&bs))
        return;

    if (bs.a > *max_a) {
        sum->max_a = bs.a;
        *max_a = bs.a;
        }

    sum->bind_E += bs.E;

    create_summary(p,part->prim,sum,max_a);
    create_summary(p,part->sat,sum,max_a);
    }

int sort_mass(const void *a, const void *b)
{

    const HIER_OUTPUT *s1,*s2;

    /* pointers to systems that made the cut */
    s1 = (const HIER_OUTPUT *) a;
    s2 = (const HIER_OUTPUT *) b;

    /* largest system first */
    if (s1->sys_mass < s2->sys_mass) return 1;
    if (s1->sys_mass > s2->sys_mass) return -1;

    /* if total system masses are equal */
    if (s1->com->index < s2->com->index) return -1;
    /* smallest com indices first */
    if (s1->com->index > s2->com->index) return 1;
    /* can't get here, top com particles can not have same indices */
    assert(0);

    return 0;
    }
```

```
int write_hier_output(const PARAMS *p,const char *filename,
                      const COMPDATA *part,long n_part,
                      double m_tot,long n_orig)
{
    BIN_STATS bs;
    const SSDATA *pd,*sd;
    HIER_OUTPUT *sum;
    FILE *fp;
    char outfile[MAXPATHLEN];
    int n_layer;
    long i,n_sys,n_true_hier,sum_buf_size;

    if (myNewExt(filename,FileTypeStr[p->FileType],outfile,HIER_EXT)) {
        (void) fprintf(stderr,"Unable to generate output filename for
         %s.\n",filename);
        return 1;
        }

    if ((fp = fopen(outfile,"w")) == NULL) {
        (void) fprintf(stderr,"Can't open %s\n",outfile);
        return 1;
        }

    n_sys = 0;
    sum_buf_size = BUF_SIZE_INIT;
    sum = (HIER_OUTPUT *) malloc(sum_buf_size*sizeof(HIER_OUTPUT));
    assert(sum != NULL);

    for (i=0;i<n_part;i++) {
        pd = &(part[i].prim->data);
        sd = &(part[i].sat->data);
        /* top of system tree */
        if (part[i].com == NULL && part[i].prim != NULL) {
            calc_bin_stats(pd,sd,BIN_CUT,&bs);
            if (ok_to_cut(p,part[i].prim,part[i].sat,&bs)) {
            /* if first layer of system does not make cut ignore system */
            /* altogether */
                continue;
                }
            sum[n_sys].sys_mass = pd->mass + sd->mass;
            sum[n_sys].max_a = bs.a;
            sum[n_sys].bind_E = bs.E;
            sum[n_sys].com = &part[i];
```

```
        create_summary(p,part[i].prim,&sum[n_sys],&(bs.a));
        create_summary(p,part[i].sat,&sum[n_sys],&(bs.a));

        if (++n_sys == sum_buf_size) {
            sum_buf_size *= BUF_SIZE_MULT;
            sum = (HIER_OUTPUT *) realloc((void *) sum,
                  sum_buf_size*sizeof(HIER_OUTPUT));
            assert(sum != NULL);
            }
        }
    }

/* sort systems with most massive first */
qsort((void *) sum,n_sys,sizeof(HIER_OUTPUT),sort_mass);
n_true_hier = 0;
for (i=0;i<n_sys;i++) {
    if (walk_system(p,fp,sum[i].com,m_tot,TRUE,&n_layer) != 0) {
        (void) fprintf(stderr,"Error during write.\n");
        (void) fclose(fp);
        return 1;
        }
    if (fprintf(fp,"System summary: mass = %8.2e, max semimajor
                    axis = %8.2e,
                    total binding energy = %9.2e\n",
                    sum[i].sys_mass*p->OutMassConv,
                    sum[i].max_a*p->OutLengthConv,
                    sum[i].bind_E*p->OutEnergyConv) < 1) return 1;
    fprintf(fp,"\n");
    if (n_layer > 1)
        ++n_true_hier;
    }
if (fprintf(fp,"%ld system%s found: %ld 2-particle system%s and %ld
                multi-particle system%s\n", n_sys,
                n_sys==1?"":"s",n_sys-n_true_hier,
                n_sys-n_true_hier==1?"":"s", n_true_hier,
                n_true_hier==1?"":"s") < 1) return 1;
if (fprintf(fp,"Total number of original particle%s: %ld\n",
                n_orig==1?"":"s",n_orig) < 1) return 1;
if (fprintf(fp,"Total mass in original particle%s: %8.2e\n",
                n_orig==1?"":"s",m_tot*p->OutMassConv) < 1)
    return 1;

(void) fclose(fp);
free((void *) sum);
```

```
    return 0;
    }

int sort_per(const void *a,const void *b)
{
    const BINARY *b1,*b2;
    const SSDATA *p1,*p2;

    b1 = (const BINARY *) a; /* pointers to binaries */
    b2 = (const BINARY *) b;

    p1 = &(b1->prim->data); /* pointers to primaries */
    p2 = &(b2->prim->data);

    if (b1->period < b2->period) return 1; /* largest periods first */
    if (b1->period > b2->period) return -1;

    if (p1->mass < p2->mass) return 1; /* largest primary masses first */
    if (p1->mass > p2->mass) return -1;

    /* smallest indices first */
    if (b1->prim->index < b2->prim->index) return -1;
    if (b1->prim->index > b2->prim->index) return 1;

    /* within each system, sort by binding energy, but check for */
    /* duplicate first */

    {
       /* satellite pointers */
        const SSDATA *s1 = &(b1->sat->data),*s2 = &(b2->sat->data);
        if (b1->sat->index == b2->sat->index) return 0; /* oops! */

        {
            BIN_STATS bs1,bs2;

            calc_bin_stats(p1,s1,BIN_ENERGY,&bs1);
            calc_bin_stats(p2,s2,BIN_ENERGY,&bs2);

            if (bs1.E < bs2.E) return -1; /* more bound first */
            if (bs1.E > bs2.E) return 1;

            /* smallest indices first */
            if (b1->sat->index < b2->sat->index) return -1;
            if (b1->sat->index > b2->sat->index) return 1;
```

```
            }
        }

    assert(0); /* can't get here */

    return 0;
    }

void calc_per(BINARY *list,long n)
{
    BIN_STATS bs;
    int i;

    for (i=0;i<n;i++) {
        calc_bin_stats(&list[i].prim->data,&list[i].sat->data,
                       BIN_PERIOD,&bs);
        list[i].period = bs.P;
        }
    }

void find_systems(const PARAMS *p,COMPDATA **part,long *npart,
                  VECTOR root_center,double root_size,NODE **root,
                  BINARY **bin,long *bin_buf_size,long *nbin)
{
    BINARY *binary,*add,*new,*ptr;
    COMPDATA *com;
    long *del;
    long part_buf_size,del_buf_size,add_buf_size,new_buf_size,buf;
    long nmax,nloops,ntree,ncmp,ncom,ndel,nadd,nnew,idel,ibin,iadd,n,i;

    (void) printf("Starting hierarchical search for systems...\n");

    /*
     ** Do the following just once: compute periods for each existing
     ** binary and sort the binaries in decreasing order of period.
     ** Subsequently binaries will be deleted and possibly added to
     ** the list while preserving the sort order.
     */

    calc_per(*bin,*nbin);
    qsort((void *) *bin,*nbin,sizeof(BINARY),sort_per);

    /* initialize */
    ntree = ncmp = *npart; /* used to monitor tree state */
```

```
ncom = 0; /* ditto */
/* see comment at top of file */
part_buf_size = (*npart)*EXTRA_STORE;

/* allocate space for maintenance lists */
del_buf_size = BUF_SIZE_INIT;
del = (long *) malloc(del_buf_size*sizeof(long));
assert(del != NULL);
add_buf_size = BUF_SIZE_INIT;
add = (BINARY *) malloc(add_buf_size*sizeof(BINARY));
assert(add != NULL);
new_buf_size = *bin_buf_size;
new = (BINARY *) malloc(new_buf_size*sizeof(BINARY));
assert(new != NULL);

/*
 ** Now loop, finding "tightest" (shortest period) binary each
 ** time, and updating the binary list as required, until no
 ** binaries remain.  Periodically rebuild the tree to improve
 ** efficiency.
 */

/* worst-case scenario */
nmax = (*npart > 0xffff ? INT_MAX : (*npart - 1)*(*npart)/2);
nloops = 0;

while (*nbin > 0) {
    --(*nbin); /* truncate list */
    /* last binary in list had shortest period */
    binary = &((*bin)[*nbin]);
    make_com_part(binary,npart,&part_buf_size,part);
    --ncmp; /* 2 particles replaced by 1 com particle */
    assert(ncmp > 0); /* can't run out of particles! */
    /* last particle in list is new com particle */
    com = &((*part)[*npart - 1]);

    /*
     ** Check to see if tree should be rebuilt:
     ** 1) if the ratio of the number of original particles left
     **    to the number of particles in the tree since the last
     **    tree build is less than TREE_REBUILD_FRAC.
     ** 2) if the mass ratio of the two components of the new com
     **    particle exceed REBUILD_MASS_RATIO.
     */
```

```
if ((double) ncmp/ntree < TREE_REBUILD_FRAC ||
    binary->prim->data.mass/binary->sat->data.mass >
    REBUILD_MASS_RATIO) {
    (void) printf("Rebuilding tree... (N = %li)\n",ncmp);
    kill_node(*root);
    /* note: recomputing center & size would improve efficiency */
    make_node(root_center,root_size,root);
    for (i=0;i<*npart;i++)
        if (!IS_COM_MBR(&((*part)[i]))) /* no child particles */
            add_to_tree(*root,&((*part)[i]));
    get_com_vel(*root);
    /* particle conservation check */
    assert((*root)->n_part == ncmp);
    /* number of particles in tree for this rebuild */
    ntree = ncmp;
    ncom = 0;
    }
else {
    add_to_tree(*root,com); /* add com particle to tree */
    /* number of com particles added since last tree rebuild */
    ++ncom;
    get_com_vel(*root);
    assert((*root)->n_part == ntree + ncom);
    }

/* create list of binaries to remove */

for (ndel=ibin=0;ibin<*nbin;ibin++) {
    binary = &((*bin)[ibin]);
    /*
     ** Record any binary whose primary or satellite was either
     ** of the children of the new com particle, and mark the
     ** members of that binary to be resent to
     ** find_companion().
     */
    if (IS_COM_MBR(binary->prim) || IS_COM_MBR(binary->sat)) {
        del[ndel] = ibin;
        if (++ndel == del_buf_size) {
            del_buf_size *= BUF_SIZE_MULT;
            del = (long *) realloc((void *) del,
                  del_buf_size*sizeof(long));
            assert(del != NULL);
            }
```

```
        }
    }

/*
 ** Now call find_companion() for the new com particle, storing
 ** results in new list.
 */

nadd = 0; /* (reuse existing storage) */
find_companion(p,*root,com,&add,&add_buf_size,&nadd);

/* compute periods of binaries to add, then sort */

calc_per(add,nadd);
qsort((void *) add,nadd,sizeof(BINARY),sort_per);

/*
 ** Update binary list by deleting old binaries and adding
 ** new binaries all in a single pass, being careful to
 ** reject any duplicated entries in the new list.
 */

idel = ibin = iadd = nnew = 0;
while (ibin < *nbin || iadd < nadd) {
    /* omit current binary from new list? */
    if (ibin < *nbin) {
        binary = &((*bin)[ibin]);
        if (idel < ndel && del[idel] == ibin) {
            ++idel; /* increment and don't copy */
            ++ibin;
            continue;
            }
        }
    /*
     ** Following "while" cascade does not consider binding
     ** energy, unlike sort_per() -- we'd rather avoid two
     ** calc_bin_stats() calls here.  Normally only an
     ** artificial test should lead to this being a problem.
     */
    while (iadd < nadd &&
           (ibin == *nbin ||
            (add[iadd].period > binary->period ||
             (add[iadd].period == binary->period &&
              (add[iadd].prim->data.mass >
```

```
                binary->prim->data.mass ||
                (add[iadd].prim->data.mass ==
                 binary->prim->data.mass &&
                 (add[iadd].prim->index < binary->prim->index ||
                  (add[iadd].prim->index ==
                   binary->prim->index &&
                   add[iadd].sat->index <
                   binary->sat->index)))))))) {
            new[nnew] = add[iadd];
            /* check for duplicate add -- they will always be
            /* paired together */
            if (++iadd < nadd && IS_SAME_BINARY(&add[iadd],
                &new[nnew]))
                ++iadd; /* skip it */
            /* increment and check for possible buffer overflow */
            if (++nnew == new_buf_size) {
                new_buf_size *= BUF_SIZE_MULT;
                new = (BINARY *) realloc((void *) new,
                      new_buf_size*sizeof(BINARY));
                assert(new != NULL);
                }
            }
        /* copy current binary to new list? */
        if (ibin < *nbin) {
            new[nnew] = *binary;
            ++ibin;
            /* increment and check for possible buffer overflow */
            if (++nnew == new_buf_size) {
                new_buf_size *= BUF_SIZE_MULT;
                new = (BINARY *) realloc((void *) new,
                      new_buf_size*sizeof(BINARY));
                assert(new != NULL);
                }
            }
        }

    /* swap new binary list with original list via pointers to */
    /* save time */

    ptr = *bin;
    buf = *bin_buf_size;
    n = *nbin;
    *bin = new;
    *bin_buf_size = new_buf_size;
```

```
        *nbin = nnew;
        new = ptr;
        new_buf_size = buf;
        nnew = n;

        if (*nbin % 100 == 0) (void) printf("%li binaries remaining
                                            (%li/%li active/total
                                            particles)\n",
                                            *nbin,ncmp,*npart);

        if (++nloops == nmax) {
            (void) fprintf(stderr,"Maximum number of loops
                                   exceeded\n");
            break;
            }
        }

    free((void *) new);
    free((void *) add);
    free((void *) del);
    }

int main(int argc,char *argv[])
{
    extern int optind;

    COMPDATA *part;
    NODE *root;
    BINARY *list;
    PARAMS params;
    VECTOR root_center;
    double m_tot,root_size;
    long list_size,list_posn;
    long n_part,n_orig,i;

    /*Defaults*/
    set_defaults(&params);

    /* Parse command line arguments */
    parse_in(argc,argv,&params);

    for(;optind<argc;optind++) {

        (void) printf("Reading data...\n");
```

```
if (read_data(&params,argv[optind],&part,&n_part,&m_tot,
              root_center,&root_size) != 0)
    return 1;

(void) printf("Building tree...\n");
make_node(root_center,root_size,&root);
for (i=0;i<n_part;i++)
    add_to_tree(root,&part[i]);

(void) printf("Computing center of mass...\n");
get_com_vel(root);
assert(root->n_part == n_part); /* particle conservation check */

(void) printf("Beginning satellite search...\n");
list = NULL;
list_size = list_posn = 0;
for (i=0;i<n_part;i++)
    find_companion(&params,root,&part[i],&list,&list_size,
                   &list_posn);
(void) printf("%li binar%s found.\n",list_posn,
               list_posn==1?"y":"ies");

if (list_posn == 0) goto done; /* no point in continuing */

if (params.ExtIdx >= 0 && params.ExtIdx < n_part) {
    char ext_outfile[MAXPATHLEN];

    if (myNewExt(argv[optind],FileTypeStr[params.FileType],
        ext_outfile,EXT_EXT)) {
        (void) fprintf(stderr,"Unable to generate output
                               filename for %s.\n",
                               argv[optind]);
        return 1;
        }

    (void) extract(&params,ext_outfile,list,list_posn);
    }

if (params.Hier == TRUE) {
    n_orig = n_part;
    find_systems(&params,&part,&n_part,root_center,root_size,
                 &root,&list,&list_size,&list_posn);
    /*add hierarchy extraction here*/
    if (params.HierExtIdx >= 0) {
```

```
            if (params.HierExtIdx >= n_orig && params.HierExtIdx
                < n_part) {
                char hier_ext_outfile[MAXPATHLEN];

                if (myNewExt(argv[optind],
                             FileTypeStr[params.FileType],
                             hier_ext_outfile,HIER_EXT_EXT)) {
                    (void) fprintf(stderr,"Unable to generate
                                           output filename
                                          for %s.\n",argv[optind]);
                    return 1;
                    }

                if (hier_extract(&params,hier_ext_outfile,
                    &part[params.HierExtIdx])) {
                    (void) fprintf(stderr,"Unable to extract
                                           hierarchy system
                                           of particle %ld\n",
                                           params.HierExtIdx);
                    return 1;
                    }
                }
            else {
                (void) fprintf(stderr,"Hierarchy extraction index
                                       must be a center of
                                       mass particle\n");
                (void) fprintf(stderr,"Index >= %ld\n",n_orig);
                return 1;
                }
            }
        /* cuts applied */
        (void) write_hier_output(&params,argv[optind],part,n_part,
                                 m_tot,n_orig);
        }
    else { /* do a normal cull */

        (void) printf("Sorting...\n");
        qsort((void *) list,list_posn,sizeof(BINARY),sort_bin);

        /*applies any cuts stored in params*/
        cut_list(&params,list,&list_posn);
        (void) printf("%li binar%s survived the cut.\n",list_posn,
                      list_posn==1?"y":"ies");
```

```
        /* output normally */
        (void) write_output(&params,argv[optind],list,list_posn,
                            m_tot);

        if (params.TipsyFile == TRUE) {
            char tip_outfile[MAXPATHLEN];

            if (myNewExt(argv[optind],FileTypeStr[params.FileType],
                         tip_outfile,VEC_EXT)) {
                (void) fprintf(stderr,"Unable to generate output
                                       filename for %s.\n",
                               argv[optind]);
                return 1;
                }

            (void) write_tipsy(tip_outfile,n_part,list,list_posn);
            }

        }

done:

    /* all done */

    free((void *) list);
    kill_node(root);
    free((void *) part);
    }

return 0;
}
```

Bibliography

Agnor, C. B. & Ward, W. R. 2002, ApJ, 567, 579

Alonso, R., Brown, T. M., Torres, G., Latham, D. W., Sozzetti, A., Mandushev, G., Belmonte, J. A., Charbonneau, D., Deeg, H. J., Dunham, E. W., O'Donovan, F. T., & Stefanik, R. P. 2004, ApJ, 613, L153

Araki, S. & Tremaine, S. 1986, Icarus, 65, 83

Asphaug, E. & Benz, W. 1996, Icarus, 121, 225

Asphaug, E. & Melosh, H. J. 1993, Icarus, 101, 144

Asphaug, E., Ostro, S. J., Hudson, R. S., Scheeres, D. J., & Benz, W. 1998, Nature, 393, 437

Asphaug, E., Ryan, E. V., & Zuber, M. T. 2002, in Asteroids III, ed. W. F. Bottke, A. Cellino, P. Paolicchi, & R. P. Binzel (Univ. of Arizona Press, Tuscon AZ), 463–484

Barnes, J. & Hut, P. 1986, Nature, 324, 446

Beaugé, C. & Aarseth, S. J. 1990, MNRAS, 245, 30

Belton, M. & Carlson, R. 1994, IAU Circ., 5948, 2

Belton, M., Chapman, C., Thomas, P., Davies, M., Greenberg, R., Klaasen, K., Byrnes, D., D'Amario, L., Synnott, S., Merline, W., Petit, J.-M., Storrs, A., & Zellner, B. 1995, Nature, 374, 785

Bentley, J. L. & Friedman, J. H. 1979, Comput. Surv., 11, 397

Benz, W. 2000, Space Science Reviews, 92, 279

Benz, W. & Asphaug, E. 1999, Icarus, 142, 5

Binney, J. & Tremaine, S. 1987, Galactic Dynamics (Princeton Univ. Press, Princeton, NJ)

Blum, J. & Muench, M. 1993, Icarus, 106, 151

Blum, J. & Wurm, G. 2000, Icarus, 143, 138

Boss, A. P. 1998, ApJ, 503, 923

Boss, A. P., Wetherill, G. W., & Haghighipour, N. 2002, Icarus, 156, 291

Bottke, W. F. & Melosh, H. J. 1996a, Icarus, 124, 372

—. 1996b, Nature, 381, 51

Bottke, W. F., Richardson, D. C., & Love, S. G. 1997, Icarus, 126, 470

Bottke, W. F., Richardson, D. C., Michel, P., & Love, S. G. 1999, AJ, 117, 1921

Chauvin, G., Lagrange, A. M., Dumas, C., Zuckerman, B., Mouillet, D., Song, I., Beuzit, J. L., & Lowrance, P. 2004, A&A, 425, L29

Chauvineau, B. & Farinella, P. 1995, Icarus, 115, 36

Davies, M. B., Benz, W., & Hills, J. G. 1991, ApJ, 381, 449

Davis, M., Efstathiou, G., Frenk, C. S., & White, S. D. M. 1985, ApJ, 292, 371

Durda, D. D. 1996, Icarus, 120, 212

Durda, D. D., Bottke, W. F., Enke, B. L., Merline, W. J., Asphaug, E., Richardson, D. C., & Leinhardt, Z. M. 2004, Icarus, 170, 243

Durda, D. D., Greenberg, R., & Jedicke, R. 1998, Icarus, 135, 431

Goldreich, P., Lithwick, Y., & Sari, R. 2002, Nature, 420, 643

—. 2004, ARA&A, 42, 549

Goldreich, P. & Ward, W. R. 1973, ApJ, 183, 1051

Governato, F., Moore, B., Cen, R., Stadel, J., Lake, G., & Quinn, T. 1997, New Astronomy, 2, 91

Greenberg, R., Hartmann, W. K., Chapman, C. R., & Wacker, J. F. 1978, Icarus, 35, 1

Harris, A. W. 1996, in Lunar and Planetary Institute Conference Abstracts, 493–+

Holsapple, K. A. 1994, Planet. Space Sci., 42, 1067

Housen, K. R., Holsapple, K. A., & Voss, M. E. 1999, Nature, 402, 155

Housen, K. R., Schmidt, R. M., & Holsapple, K. A. 1991, Icarus, 94, 180

Johnson, B. M. & Gammie, C. F. 2003, ApJ, 597, 131

Kokubo, E. & Ida, S. 1995, Icarus, 114, 247

—. 1996, Icarus, 123, 180

—. 1998, Icarus, 131, 171

—. 2000, Icarus, 143, 15

—. 2002, ApJ, 581, 666

Konacki, M. & Wolszczan, A. 2003, ApJL, 591, L147

Lecar, M. & Aarseth, S. J. 1986, ApJ, 305, 564

Leinhardt, Z. M. & Richardson, D. C. 2002, Icarus, 159, 306

—. 2005a, Icarus, in press

—. 2005b, ApJ, in press

Leinhardt, Z. M., Richardson, D. C., & Quinn, T. 2000, Icarus, 146, 133

Lissauer, J. J. 1993, ARA&A, 31, 129

Lodders, K. & Fegley, B. 1998, The Planetary Scientist's Companion (Oxford University Press, New York)

Love, S. G. & Ahrens, T. J. 1996, Icarus, 124, 141

Love, S. G., Hörz, F., & Brownlee, D. E. 1993, Icarus, 105, 216

Makino, J., Fukushige, T., Funato, Y., & Kokubo, E. 1998, New Astronomy, 3, 411

Margot, J. L., Nolan, M. C., Benner, L. A. M., Ostro, S. J., Jurgens, R. F., Giorgini, J. D., Slade, M. A., & Campbell, D. B. 2002, Science, 296, 1445

Mayer, L., Quinn, T., Wadsley, J., & Stadel, J. 2002, Science, 298, 1756

Merline, W. J. 2001, Bull. Am. Astron. Soc., 33, 1133

Merline, W. J., Close, L. M., Dumas, C., Chapman, C. R., Roddier, F., Menard, F., Colwell, W., Slater, D. C., Duvert, G., Shelton, C., & Morgan, T. 1999, Bull. Am. Astron. Soc., 31, 1106

Michel, P., Benz, W., Tanga, P., & Richardson, D. C. 2001, Science, 294, 1696

Miller, M. C. & Hamilton, D. P. 2001, ApJ, 550, 863

Ostro, S. J., Pravec, P., Benner, L. A. M., Hudson, R. S., Šarounová, L., Hicks, M. D., Rabinowitz, D. L., Scotti, J., Tholen, D. J., Wolf, M., Jurgens, R. F., Thomas, M. L., Giorgini, J. D., Chodas, P. W., Yeomans, D. K., Rose, R., Frye, R., Rosema, K. D., Winkler, R., & Slade, M. A. 1999, Science, 285, 557

Petit, J.-M. & Hénon, M. 1987, Astron. Astrophys., 188, 198

Pravdo, S. H., Shaklan, S. B., Henry, T., & Benedict, G. F. 2004, ApJ, 617, 1323

Pravec, P., Šarounová, L., Hicks, M. D., Rabinowitz, D. L., Wolf, M., Scheirich, P., & Krugly, Y. N. 2002, Icarus, 158, 276

Pravec, P., Šarounová, L., Rabinowitz, D. L., Hicks, M. D., Wolf, M., Krugly, Y. N., Velichko, F. P., Shevchenko, V. G., Chiorny, V. G., Gaftonyuk, N. M., & Genevier, G. 2000, Icarus, 146, 190

Rice, W. K. M., Armitage, P. J., Bonnell, I. A., Bate, M. R., Jeffers, S. V., & Vine, S. G. 2003, MNRAS, 346, L36

Richardson, D. C. 1994, MNRAS, 269, 493

—. 2001, Bull. of the Am. Astron. Soc., 33, 1351

Richardson, D. C., Asphaug, E., & Benner, L. 1995, Bull. Am. Astron. Soc., 27, 1114

Richardson, D. C., Bottke, W. F., & Love, S. G. 1998, Icarus, 134, 47

Richardson, D. C., Elankumaran, P., & Sanderson, R. E. 2005, Icarus (in press)

Richardson, D. C., Leinhardt, Z. M., Melosh, H. J., Bot tke, W. F., & Asphaug, E. 2002, in Asteroids III, ed. W. F. Bottke, A. Cellino, P. Paolicchi, & R. P. Binzel (Univ. of Arizona Press, Tuscon AZ), 501–515

Richardson, D. C., Quinn, T., Stadel, J., & Lake, G. 2000, Icarus, 143, 45

Ryan, E. V., Hartmann, W. K., & Davis, D. R. 1991, Icarus, 94, 283

Ryan, E. V. & Melosh, H. J. 1998, Icarus, 133, 1

Safronov, V. S. 1969, Evolution of the protoplanetary cloud and formation of the Earth and planets (Nauka, Moscow), transl. 1972 NASA TT F-677

Sigurdsson, S., Richer, H. B., Hansen, B. M., Stairs, I. H., & Thorsett, S. E. 2003, Science, 301, 193

Stadel, J. G. 2001, PhD thesis, University of Washington, Seattle, WA

Tanga, P., Babiano, A., Dubrulle, B., & Provenzale, A. 1996, Icarus, 121, 158

Thorsett, S. E., Arzoumanian, Z., Camilo, F., & Lyne, A. G. 1999, ApJ, 523, 763

Veverka, J., Thomas, P., Harch, A., Clark, B., Bell, J. F., Carcich, B., Joseph, J., Chapman, C., Merline, W., Robinson, M., Malin, M., McFadden, L. A., Murchie, S., Hawkins, S. E., Farquhar, R., Izenberg, N., & Cheng, A. 1997, Science, 278, 2109

Wasson, J. T. 1985, Meteorites–Their Record of Early Solar System History (Freeman, New York)

Watanabe, S. & Miyama, S. M. 1992, ApJ, 391, 318

Weidenschilling, S. J. 1977, MNRAS, 180, 57

—. 1995, Icarus, 116, 433

—. 2002, Icarus, 160, 212

Weidenschilling, S. J. & Cuzzi, J. N. 1993, in Protostars and Planets III (University of Arizona Press, Tuscon), 1031

Wetherill, G. W. & Stewart, G. R. 1989, Icarus, 77, 330

—. 1993, Icarus, 106, 190

Wolszczan, A. 1994, Science, 264, 538

Wolszczan, A. & Frail, D. A. 1992, Nature, 355, 145

Yeomans, D. K., Barriot, J.-P., Dunham, D. W., Farquhar, R. W., Giorgini, J. D., Helfrich, C. E., Konopliv, A. S., McAdams, J. V., Miller, J. K., Owen, W. M., Scheeres, D. J., Synnott, S. P., & Williams, B. G. 1997, Science, 278, 2106

Youdin, A. N. & Chiang, E. I. 2004, ApJ, 601, 1109

Youdin, A. N. & Shu, F. H. 2002, ApJ, 580, 494

Zappalà, V., Cellino, A., Farinella, P., & Knezevic, Z. 1990, AJ, 100, 2030