The N-Body Simulation of Galaxy Dynamics

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Noah Muldavin

May 2013

Approved for the Division
(Physics)

_____

Johnny Powell

Acknowledgements

First and foremost I want to thank my advisor, Johnny Powell, for letting me jump into the deep end with this crazy project, and for giving such solid guidance the whole way through. I don't think I could have gotten more out of this experience, or had more fun. It's been awesome. Thanks to Emily Brown for all the advice from a recent physics graduate and for being such a level headed, wonderful person. Thanks to my wonderful parents for giving me so much support, I really couldn't have done it without you. Thanks to Joel Franklin for the computer stuff. Thanks to Connor, Lukas and Max for being a blast in our underground lair. Thanks to Zach Brown for planning good pubbasements. Thanks to all the people who have enriched my life with amazing experiences, conversations, and ideas during my four years at Reed. This is an amazing community to have been a part of, may it never loose its spark. Last but certainly not least I want to thank my grandfather, Herbert Keller for financing my education. He would have loved reading this thesis, and I'm sure would have a lot to tell me about numerical analysis.

Preface

This thesis is the result of more than a year spent studying the dynamics of galaxies and the N-Body simulation. It began when Johnny sent me an excited email during the winter break of my junior year, wondering if I wanted to do an independent study project on galactic astrophysics. Of course I said yes, launching this 15 month endeavor. It became clear after a few weeks that I was drawn to the theoretical study of the *dynamics* of galaxies more than anything else so after spending some time getting familiar with basic observational facts about galaxies, I dove into the study of galactic dynamics from a couple more advanced textbooks.

The N-Body simulation entered the picture after I got rejected from all of the REUs (Research Experience for Undergraduates) I applied to for the summer of my junior year. In stead, Johnny offered some of his faculty research funds for me to continue my study over the summer. It seemed natural to use the time to build an N-Body simulation code, since it is the most widely used tool in the study of galactic dynamics. By the end of summer I had a working simulation code written in Mathematica. Feeling limited by Mathematica's computational capabilities, I decided that the next step should be to rewrite the code in C++, a language which I was not familiar with at the time. It turned out to take the better part of the fall semester to learn C++ and implement the code, in part because I got wrapped up with MPI (Message Passing Interface) trying to run the code in parallel. In hindsight that probably wasn't the best use of my time, but I did get it to work eventually. Before the end of the semester I was able to run a simulation of an Sc class galaxy without a dark matter halo (the first trial presented in Chapter 4). Not long into the spring semester I ran a second test *with* a dark matter halo (the second trial in Chapter 4).

There was a moment when I was staring at the results of my simulations, in a daze from weeks of frantic code-debugging, when I realized that I had *no idea* what the result *meant*. To be sure, there were bits and pieces of information floating around in my brain but I didn't have a way to organize it in to a cohesive picture. In a very intentional way, the process of writing this thesis was an attempt to organize my own thoughts, to figure out what's actually going on when those black dots appear on a screen. As such, what I've written here is not a complete description of what I've learned or done but a cohesive *narrative*contextualizes the main result. I think this will benefit the reader as well, since true understanding only results when ideas are accompanied by a framework in which they may be understood.

Table of Contents

Abstract

The N-Body simulation is discussed and applied as a tool for studying the dynamics of disk galaxies, with particular attention paid to the formation of spiral arms and their relation to dark matter. As a comparison and in order to build intuition about the dynamical processes at work, analytical results on the statistical mechanics and stability of galactic disks are presented. The rationale for the N-Body simulation is presented in relation to these analytical results, and an algorithm is developed. Two simulations are performed: one of a typical Sc class disk galaxy *without* an encompassing dark matter halo, and one of a typical Sc class disk galaxy *with* an encompassing dark matter halo. The results show that without a dark matter halo the disk is violently unstable. Furthermore, in the case with a halo a spiral pattern develops which is persistent for $20 - 30$ crossing times.

Introduction

> *Examining the record of past research from the vantage of contemporary historiography, the historian of science may be tempted to exclaim that when paradigms change, the world itself changes with them. Led by a new paradigm, scientists adopt new instruments and look in new places. Even more important, during revolutions scientists see new and different things when looking with familiar instruments in places they have looked before. It is rather as if the professional community had been suddenly transported to another planet where familiar objects are seen in a different light and are joined by unfamiliar ones as well.*
>
> - Thomas Kuhn

The story of how we know what we know about galaxies is a story of the extension of the human eye and the ongoing attempt to interpret that new field of vision. We are stuck for now and forever with a single perspective; from a rock circling a rather unremarkable star in one galaxy among billions, we must interpret the universe as it is revealed to us by those unlikely photons which, having travelled from the farthest reaches of space, finally strike the retina of an earthbound observer. With each registered photon we reformulate and update our understanding of the world; as our understanding evolves we learn where to look and what questions to answer. From this cooperation of mind and eye emerges a picture of the universe in which we reside, in all of its mystery and beauty. Resolved among the sights contained therein is the most majestic of all celestial objects, home of stars and nebulae, black holes and interstellar gas: the galaxy.

The most important galaxy, the Milky Way, is visible as a thick band of light in the night sky interspersed with blotchy dark clouds (Fig. 1). The center, located in the constellation Sagittarius, is the brightest patch in the night sky. Most cultures have some interesting story explaining the existence of this milky celestial band. In Greek mythology it is the breast milk of the goddess Hera. As the story goes, she awoke one day to discover Zeus' illegitimate (and mortal) son Heracles suckling her in order to gain godlike abilities. Disgusted, she pushed him away, spraying milk all over the sky. It is no accident that the English word "galaxy" is derived from the Greek word "galactos" [1] which literally means milk! The Incans, who were privileged with a particularly bright view of the milky way due to their high elevation, saw a massive river connecting the world of the living to the world of the dead. In the black clouds they saw animal figures who drank from the bright river, blocking its luminescence

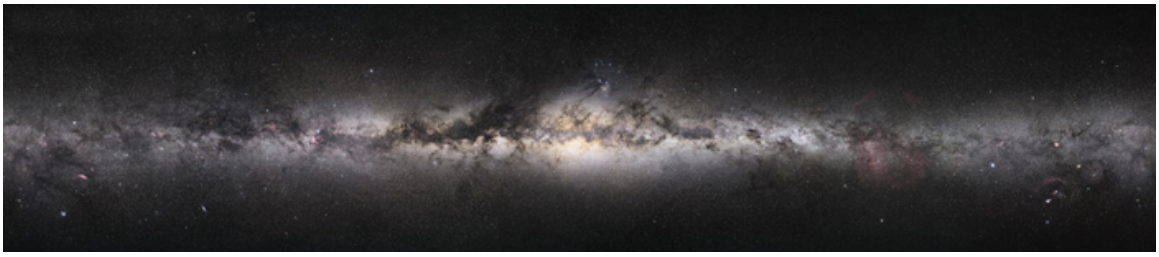

Figure 1: Panorama of the Milky Way galaxy. Black dust clouds can be seen among the bright patches. Image credit: Digital Sky LLC.

with their figures. Near the center was a mother and baby llama, threatened on either side by a snake and a fox [2].

One of the first scientific discussions of the Milky Way occurs in Aristotle's *Meteorologica* (350 BC).[1] In it, he rejects the view held by the great philosopher Democritus that the Milky Way was composed of the light of distant stars. Democritus, a father of the atomic theory of matter, believed that the universe was infinite and uniformly populated with stars. Aristotle, on the other hand, thought that the universe was a series of concentric spheres centered around Earth. Each sphere, he thought, contained a distinct set of astronomical objects, rotating in order to explain their movement in the sky. Strangely, Aristotle's explanation of the Milky Way takes up only five lines of his entire work on the physical form of the natural world. He concludes that it must be the "dry exhalation" of the marshy areas of Earth, which each night makes its way to the region between the sky-sphere and the Moon-sphere where it spreads out to form the familiar milky shape.

As absurd as his explanation is, it isn't entirely surprising that the physical form of the Milky Way wasn't more thoroughly questioned. Democritus and a few other advocates of the theory of the infinite universe were outsiders; Aristotle's geocentric cosmology (sometimes called the Aristotelean cosmology) was accepted almost uniformly among philosophers of the time. Physicality was not of immediate concern because all objects were thought to be lodged in sky-spheres of finite extent. Astronomers of the day were chiefly concerned with producing accurate predictions of celestial events, typically using some intricate pattern of intersecting circles, not with postulations of physical form.

Not much changed for nearly two millennia, in part due to the ubiquity of Aritotle's *Meteorologica*. From the Hellenic period through the Middle Ages it was among the most widely distributed and discussed works of natural philosophy. The philosopher himself was idolized to the point where new ideas needed to defend themselves against not only the scrutiny of peers but the offense of contradicting Aristotle. There were occasional murmurs of discontent with the implausible evaporation theory of the Milky Way, but most alternate theories were either unremarkable or equally as implausible. Most of the time the subject was completely disregarded. Even Titus Lucretius' *De Rerum Natura* (50 BC) which momentarily popularized the atomic

[1]For an extended discussion of the history of the Milky Way, see Reference [3].

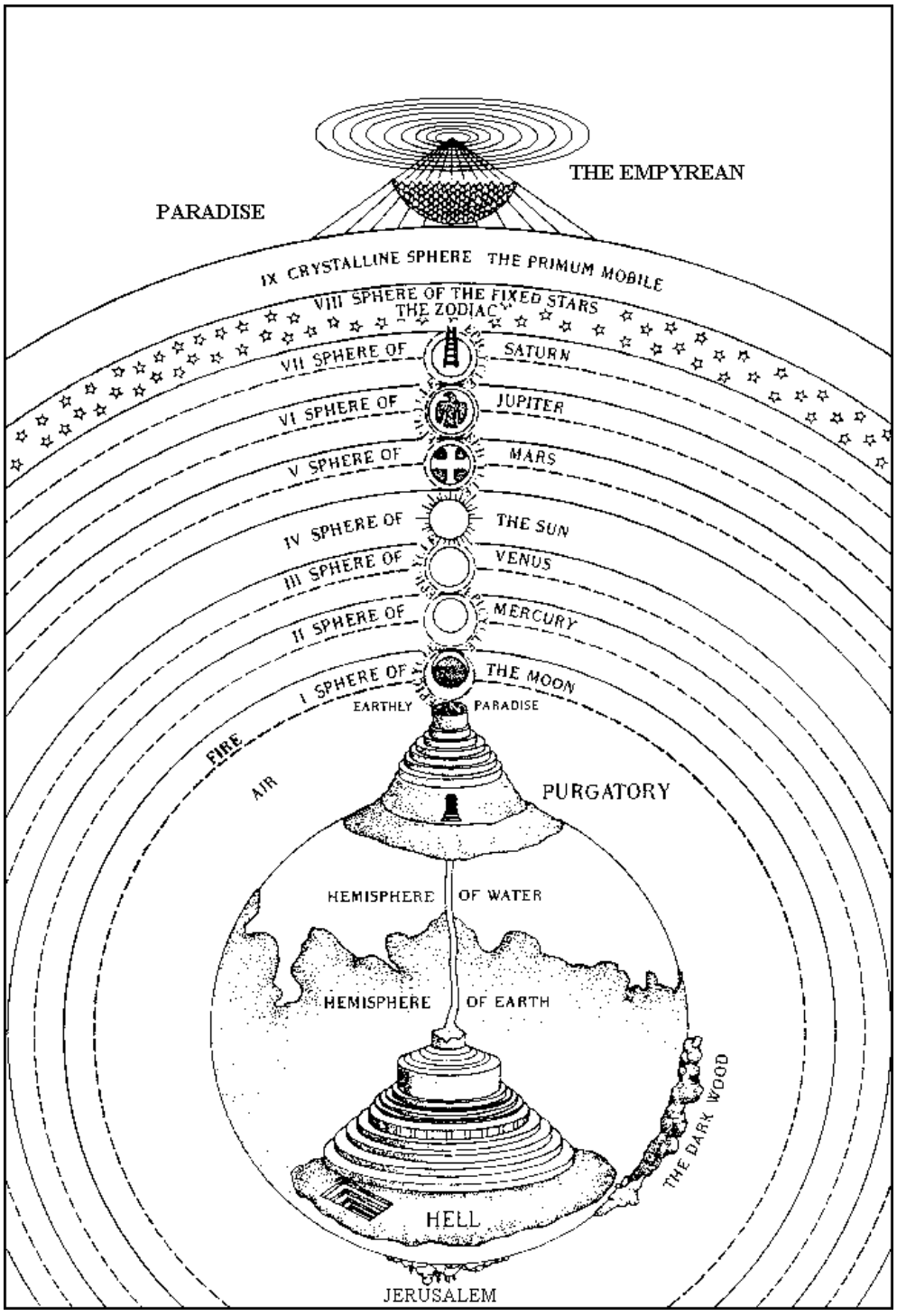

Figure 2: The universe as imagined in Dante's *Divine Comedy*, with Jerusalem at the bottom of the Earth. The fire sphere between the air and Moon sphere is where the Milky Way would have formed according to Aristotle. Taken from J. R. Primack, *From cosmology and culture*, `http://ircamera.as.arizona.edu/NatSci102/NatSci/images/extcosmo.htm` (2002).

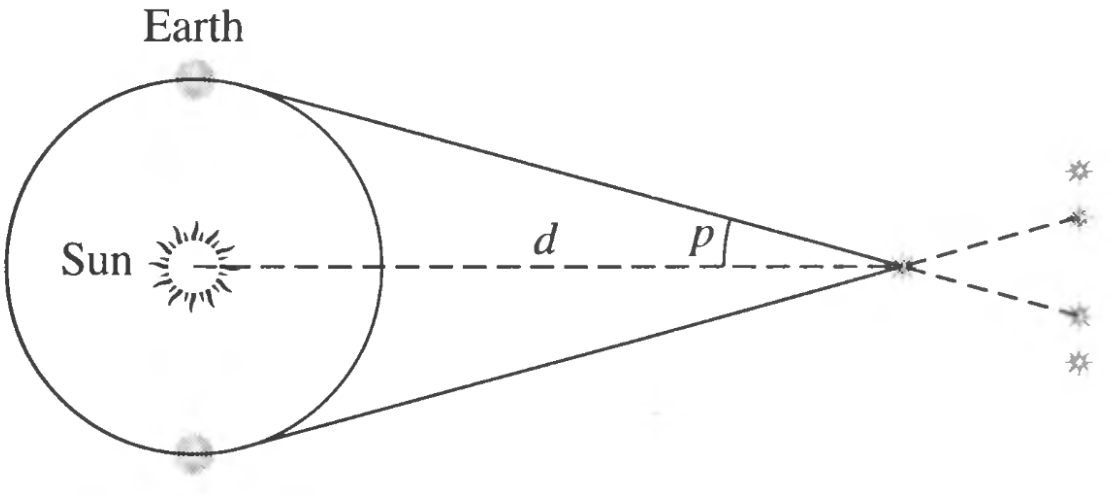

Figure 3: Trigonometric Parallax. The solid lines represent the line of sight from the Earth on either side of its orbit around the Sun to an object located at the vertex on the right. The dotted lines extend to the apparent position of the object in relation to more distant stars. The farther away an object, the less it will appear to move during the Earth's yearly rotation about the Sun. Taken from B. W. Carroll and D. A. Ostlie, *An Introduction to Modern Astrophysics* (Pearson Education, Inc, San Francisco, CA, 2007), 2nd ed., p. 58.

theory of matter contained no such discussion.

The Catholic church found Aristotelian cosmology particularly agreeable. The concentric spheres were seen as a manifestation of God's perfection, with Man in his rightful place in the center and God just beyond the outermost dome. Some maps even managed to maneuver Jerusalem to the center of the Earth so that all would follow the order laid out in the bible (Fig. 2). Through the Middle Ages there were occasional discussions of the Milky Way but they were often more theological than scientific.

In 1543 the entire world was thrown off its axis (so to speak) with the publication of Nicolaus Copernicus' *De Revolutionibus Orbium Coelestium* (On the Revolutions of the Heavens). The Aristotelian cosmology was replaced by a more convincing heliocentric model in which the Earth and all other planets orbit the Sun. In the history of paradigm shifts there isn't one much greater than this dramatic de-throning of the Earth from its glorified place at the center of the universe. For the most part it gets the credit it deserves, but one of its most important consequences is often overlooked. The fact that the Earth orbits the Sun once a year made it necessary to accept that stars in the sky should appear to *move* ever so slightly over the course of that orbit. This is a phenomenon known as trigonometric parallax: the apparent relative position of an object will change depending on the direction from which it is viewed (Fig. 3). The more distant an object, the less its position appears to change. Since the stars in the sky don't move at all over the course of a year, the Copernican revolution implied that the distance to the nearest stars was much, much larger than

Figure 4: On the left is shown the constellation of Andromeda with a large Arabic fish over her body from a manuscript written by Abu-al-Husain Al Sufi in 964 AD. The "nebulous" spot (the Andromeda nebula) is shown as a cluster of stars in the fish's mouth. On the right is the same drawing from a later text in the same tradition, this time looking down on Earth from the heavens. The Andromeda nebula can be seen clearly to the right of the fish's mouth. Taken from P. Kunitzch, The Messenger of the European Southern Observatory **49**, 42 (1987).

anyone previously imagined.

The expanded universe brought with it an expanded set of possibilities. The opening of the closed universe sparked interest in the physical form of astronomical objects. Of particular note to this narrative, there was renewed interest in a class of objects known as nebulae, which though luminous like a star, appeared hazy and unresolved. First noted in the second century BC by the ancient Greek astronomer Claudius Ptolemaeus in his catalog *Mathematiké Syntaxis* [4], nebulae were mostly ignored by astronomers because they assumed that with clear enough conditions they could be resolved into individual stars.[2] After the Copernican revolution some astronomers began to question this assumption. Though not universally accepted, a new theory that both the Milky Way and nebulae were composed of the same"luminous, nebulous material" became popular.

Concurrent with these revolutions of mind, the human eye was experiencing a revolution of its own. By the early 17th century the first telescopes were being

[2]There is one very cool exception which occured in 964 AD. In his *Catalog of the Fixed Stars*, the great Arabic astronomer Abu-al-Husain Al Sufi repeatedly mentions a "nebulous spot" appearing in the constellation Andromeda (Fig. 4). As it turns out this "nebulous spot" is the Milky Way's nearest neighbor the Andromeda Galaxy.

fashioned and put to use. In the 1610 the famous Italian astronomer and martyr of the Copernican revolution Galileo Galilei pointed his telescope at a random patch of the Milky Way. He describes what he saw [3]:

> *Third, I have observed the nature and the material of the Milky Way. With the aide of the telescope this has been scrutinized so directly and with such ocular certainty that all the disputes which have vexed philosophers through so many ages have been resolved, and we are at last freed from wordy debates about it. The galaxy is, in fact, nothing but a congeries of innumerable stars grouped together in clusters. Upon whatever part of the telescope it is directed, a vast crowd of stars is immediately presented to view. Many of them are rather large and quite bright, while the number of smaller ones is quite beyond calculation.*

This discovery proved all the more fascinating for it suggested that the universe itself might have some intrinsic shape. If the Milky Way was composed of stars just like the Sun, they must arrange themselves in such a way as to explain the strange ribbon in the sky.[3] This meant there could not be a uniform sky-dome in which all stars existed and neither could the region of stars extend out to infinity in all directions.

In 1687 Isaac Newton's *Principia Mathematica* was published in London. His work revealed the strange and remarkable fact that the physical workings of the universe could be explained in the language of mathematics. His most impressive (in my opinion) contribution was the universal law of gravitation which explained the motion of celestial objects: two massive objects will exert an attractive force on one another according to the relation

$$F_g = -G\frac{m_1 m_2}{d^2}, \tag{1}$$

where m_1 and m_2 are their respective masses, d is the distance between them, and G is a proportionality constant. This remarkable law explained with precision how celestial objects become bound to one another (like the Earth and the Moon) and therefore a framework for analyzing the physical structure of the universe as a whole.

At this point the stage was set. The human eye was steadily extending outwards into the reaches of space, the prevailing cosmology was now flexible enough to generate questions about the shape and form of the galaxy, the new field of mathematical physics provided a framework in which this questioning could occur.

The modern picture of the Milky Way emerged quietly in 1785 when William Herschel published his prodigious empirical map of the galaxy. The map was constructed by literally counting the stars in 683 regions of the sky. He assumed that 1) all stars are of the same luminosity and therefore dimmer stars are further away, 2) the density of stars is the same in all parts of the galaxy, 3) there is nothing obscuring our view

[3]This was determined by what has become known as Olber's Paradox: If the universe is infinite in all directions and populated by stars in all parts then there would lie a star in every patch of sky and thus the night sky would be white. In essence: the fact that the night sky is black is proof that the universe is not infinite (or that it is infinite in size but not infinitely old and that there is a finite speed at which light propagates).

Figure 5: The Milky Way as imagined by William Herschel in 1785. The Sun can be seen as a thick dot just to the right of center. Taken from W. Herschel, Philosophical Transactions of the Royal Society of London **75**, 213 (1785).

of the stars, and 4) one can see to the edges of the galaxy. All of these assumptions were wrong, strictly speaking, but somehow he managed to produce a fairly accurate picture of the galaxy. As he described it, it was a great stratum of stars in which our Sun was probably placed not far from the center. His diagram (Fig. 5) is one of the most famous in all of science.

Herschel's slightly lesser known contribution was an extended catalog of nebulae which later became known as the General Catalog. With the rapid improvement of the telescope during the 15th century, many nebulae turned out to resolve into individual stars. Yet others still appeared smoky and cloudy even when viewed through the largest telescopes of the era. Thirty years before the General Catalog was published the German philosopher Immanuel Kant postulated that these nebulae were in fact "island universes" similar to the Milky Way, containing billions of stars of their own. Herschel himself was particularly taken by this theory. His successful mapping of the Milky Way gave him the necessary ammunition to postulate the workings of these distant nebulae. Among his many theories, he hypothesized that they must be bound together by their mutual gravity, and that they must be relatively young since their forms were distinctly nonuniform.

Unfortunately there was absolutely no way of verifying these claims, which left room for critics (almost everyone) to deny them. Herschel's description of the Milky Way was so grand in scope and precise in its geometry that it was easy to accept. Not so for the theory of island universes: it reopened the problematic possibility of an infinite universe and once again decentered the Earth. Who can blame them for doubting? It's easy to forget how hard it is to release one's most closely held truths. The infinity or non-infinity of the universe is a weighty matter, revolutions of thought should not be rushed. For the time being at least, the astronomical community rejected the theory that the Milky Way was one of many galaxies in favor of the more comfortable hypothesis that the entirety of creation could be seen in that mysteriously floating band in the sky.

Figure 6: Lord Rosse's sketch of the first known spiral nebula (M51) from 1845. Taken from *M51 as recorded by Lord Rosse, 1845*, `http://amazing-space.stsci.edu/resources/explorations/groundup/lesson/basics/g44/`

By the mid 19th century, however, observational evidence was mounting against the single-island theory. A large number of nebulae simply hadn't resolved into individual stars, despite the general expectation that they would as the telescope allowed ever more distant objects to be observed. Consequently the boundary of the "single island" was pushed farther and farther outwards. More surprising still, the internal structure of these unresolved nebulae turned out to be more strange and exotic than anyone had anticipated. In 1845 William Parsons, the third Earl of Rosse, pointed his 72-inch telescope (nicknamed Leviathan) at a known unresolved nebula. To his surprise, he discovered a spiral pattern in its cloudy form (Fig. 6), as if the entire nebula were spinning about an axis. As it turned out there were an extremely large number of such objects, which became known as spiral nebulae.

A very pleasing explanation of the nebula problem seemed to present itself with the discovery that many unresolvable nebulae in fact consisted of luminous gas, not of individual stars [3]. This would explain why a great many remained unresolved despite the ever-increasing optical resolution provided by the telescope. Unfortunately this did not fully explain the predominance of spiral structure. If all nebulae were composed of gas they should take shapes which occur by the chance movement of a cloud of gas interacting with itself. Why then, were spiral patterns so common? Indeed, it was soon after confirmed that spiral nebulas were not composed of gas but

of stars, since the light they emit could not be that of a cloud of gas.

The astronomical community was split by this great and weighty question: were spiral nebulae part of the Milky Way or were they galaxies of their own? Without some reliable way of measuring the distance to these objects (one of the most notoriously difficult tasks in astronomy) there was no way of answering this question.

Two exciting prospects for measuring distance emerged in the late 19th and early 20th century. The total influx of radiation F_{rad} received from an object of intrinsic luminosity L at a distance d is given by

$$F_{\text{rad}} = \frac{L}{4\pi d^2}. \tag{2}$$

Therefore if F_{rad} can be measured (it usually can), the problem of determining distance is reduced to the problem of determining intrinsic luminosity. Luckily there are certain classes of astrophysical objects (sometimes referred to as "standard candles") for which intrinsic luminosity is well defined.

One such object is a nova, which occurs when hydrogen being stripped off of a larger star by the gravity of a nearby small, very dense star ignites in a runaway reaction. This reaction produces an immense amount of light, and a *relatively* constant amount in each nova (see Reference [5] chapter 15). In 1885 astronomers at the Mount Wilson observatory in southern California accidentally observed a nova occurring in the great spiral nebula of Andromeda. This proved that novae could be used as distance markers, if an inaccurate one.

A second and much more reliable class of objects are known as Cepheid Variable stars. Astronomers had been aware of these objects, which slowly change brightness over a period between 1 and 50 days, since at least the 15th century (see Reference [5] chapter 14). It was not until 1912 that their true value was discovered by an unsung hero of astronomy named Henrietta Swan Leavitt.[4] While working at the Harvard University Observatory as a "computer" tasked with the painstaking search for Cepheids among fields of stars, she noticed that more luminous stars took longer to go through their pulsation cycles. Since she was measuring stars which were known to be roughly the same distance away, the apparent brightness difference must reflect an intrinsic difference. This period-luminoscity relationship turned out to be very well defined and therefore presented a very accurate option for measuring the distance to spiral nebulae. Unfortunately, observing a cepheid variable in a spiral galaxy was next to impossible since Cepheids, unlike novae, are no more bright than an average star.

Armed with these new methods for determining distance, the debate over the island universe theory only heated up. Things came to head on April 26th, 1920 at the National Academy of Sciences in Washington D.C. when Harlow Shapley of the Mount Wilson Observatory and Heber Curtis of the Lick Observatory met to argue the merits of each point of view.[5] Now known as the "Great Debate," it is one of those

[4] for a wonderful biography, see reference [6].

[5] It reminds me of the legendary standoffs between the Dodgers and the Giants, but instead of the national league pennant they're fighting over the size of the universe. Does this put baseball in context? I'm not sure, all I know is that northern California won the debate.

events in the history of science which has made the leap from reality into the realm of mythology, an immortalized moment in the romanticized history physicists repeat *ad infinitum* as a rallying cry to the broken soldiers on the battlefields of algebra. What actually happened should be kept in mind (see References [7], [8]) but all the hubbub seems justified. What could be more mythological than two giants of astrophysics engaged in public debate over the size and form of the universe?

Shapley argued that the Milky Way was the *only* island universe and that all spiral nebulae were contained within. One of his major arguments was that if the Andromeda Nebula were as large as the Milky Way (he estimated it was 300,000 light-years across), its small angular size in the sky would require it to be so far away that all recently observed novae in the nebula would need to be *impossibly* bright. Curtis argued in favor of the theory of island universes. To him, it was necessary that the nearest nebulae be at least 450,000 light-years away in order for the novae to have intrinsic brightnesses comparable to those in the Milky Way. At this distance, the Andromeda Nebula would be similar in size to the Milky Way (his estimate of the Milky Way's size was much smaller than Shapley's). The debate offered no resolution; mostly it served to clarify the issues at hand.

The answer finally came in 1923 when Edwin Hubble observed Cepheid Variable stars in the Andromeda Galaxy using the 100 inch telescope at the Mount Wilson Observatory. Based on their period he determined that the distance to the galaxy must be roughly 930,000 light-years. This is 2.7 times smaller than the currently accepted value, but it was big enough to show that the Andromeda Galaxy could not be part of the Milky Way. The island universe theory was true: the galaxy exists.

We are told that science advances through the matching of models to facts, but as I hope to have illustrated in this introduction, it works the other way as well: models influence the availability of facts. Of all the natural sciences, physics is most subject to this reverse influence because the questions at hand are so fundamental. Often times, the subject of inquiry is so divorced from tangible reality that models are the *only* way to find things out. The central irony of physics is that these models are often as incomprehensible as the reality which they are meant to explain. What happens to the loop of models and facts when facts about the models themselves are lacking?

Two thousand years after the brief discussion in Aristotle's *meteorologica*, thought on the structure and composition of galaxies occurs through a new kind of empiricism made possible by the incredible computational abilities of the modern computer. Its subject is as much physical theory as it is of physical fact.

This thesis is concerned with a tool of astrophysical inquiry which is a major part of this trend: the N-Body simulation. In short, an N-Body simulation is an attempt to approximate the meaning embedded in the physical laws of nature using a numerical algorithm which can be implemented on a computer. Yet it is often used as a tool to learn more about the *real* universe in the sky. To me what makes it most interesting, as a phenomenon, is that one cannot make concrete claims about the similarity of an

N-Body simulation to *either* theory or truth. To be sure, there are plenty of reasons to believe that it isn't too far off, but there is no way to be absolutely positive about it. Despite this inherent uncertainty, the N-Body simulation is a major scientific tool in astrophysics and claims *are* made based on the results.

The primary goal of this thesis is to understand the intricacies of the N-Body simulation and what it means to use it as a scientific tool. My secondary goal is to participate in the history of scientific thought on the nature of galaxies as described in this chapter. Specifically, I will attempt to analyze the dynamics of a spiral galaxy in an effort to gain some insight into the formation of spiral arms and their relation to dark matter. These seemingly disparate projects are actually quite compatible, as an N-Body simulation is the only way to properly analyze galactic dynamics. More importantly, the best way to understand the intricacies of the N-Body simulation is to roll up your sleeves and do it.

Chapter 1

Observational Overview

In order to properly contextualize the discussion of galaxy dynamics which will follow, I feel it is necessary to briefly overview some of the basics of galactic astronomy. In part this is an effort to cover the bases in case I accidentally drop some jargon later on, but it is relevant and certainly interesting on its own terms. In any case, the next two sections will cover 1) common observational techniques of galactic astronomy and 2) general morphological properties of galaxies. This is not meant to be even remotely complete, only to give a general sense of *what we know* and *how we know what we know*. For more complete overviews see References [5] chapters 24 - 26 and [9] chapter 1.

1.1 Astronomical Techniques

In astronomy there is not the luxury of multiple perspectives. At the end of the day, the only information we have about the universe around us is the information which comes to Earth in the form of electromagnetic radiation,[1] traveling from distant astronomical sources, long ago. Luckily, there is an incredible amount of information encoded in those photons. The task is to extract and analyze that information as best we can.

Electromagnetic radiation is usually defined in terms of its wavelength λ or its frequency ν such that $\lambda\nu = c$, where c is the speed of light (2.9989×10^8 m/s). There are no theoretical bounds on λ, but observing a very long or very short wavelength is very unlikely. The longest wavelengths generally observed are long wave radio waves ($\lambda \approx 10^8$ m). Proceeding from long to short wavelengths, some general classifications are: AM radio, FM radio, microwaves, infrared, visible light, ultraviolet, x-rays, and γ-rays. Of this total spectrum, visible light occupies only a small segment (Fig. 1.1). The energy E carried by radiation is related to the frequency by the relation $E = h\nu$ or the wavelength by $E = \frac{hc}{\lambda}$, where h is the Planck constant (6.626×10^{-34} kg m^2/ s).

The easiest way to get information from electromagnetic radiation is through the wonderful interpretational faculties of the visual cortex. It used to be that the only

[1] I will use the terms radiation, waves, and photons interchangeably.

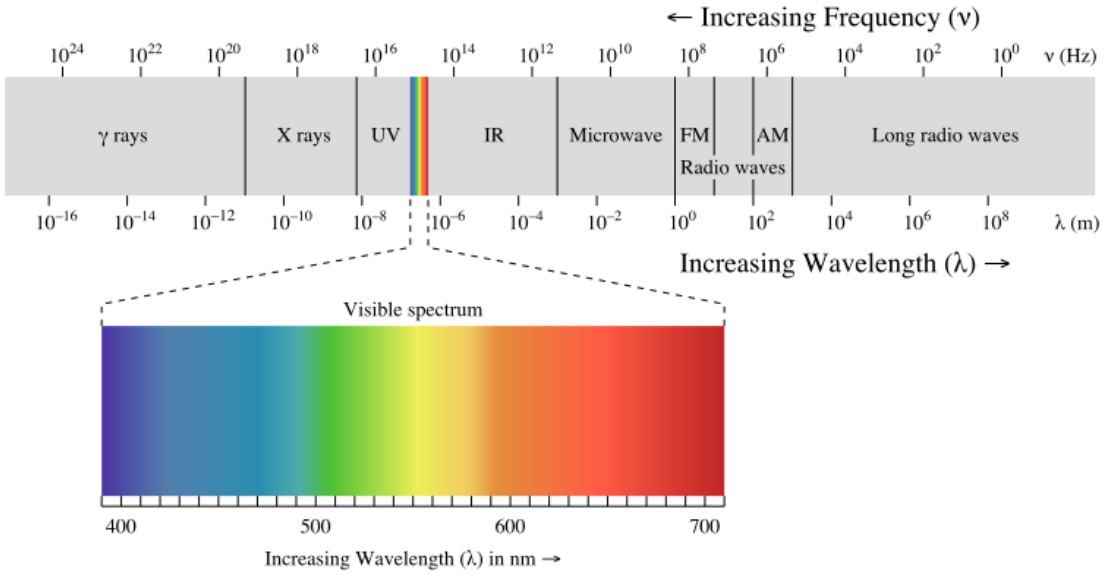

Figure 1.1: The electro-magnetic spectrum. Image Credit: Philip Ronan, Wikimedia Commons.

way to do this was to actually look at the sky, either directly or through a telescope. The invention of photography, however, expanded the possibilities for visual observation greatly. A photographic plate (or more recently, a digital camera) can collect incoming light for much longer than the eye, producing much more detailed images. Modern telescopes are equipped with state-of-the-art photographic equipment designed to wring the sky of as much light as possible. It is also possible for images to be created from light outside of the visual spectrum. By associating visual light wavelengths with other wavelengths in the observed spectrum, an image can be created that is essentially "what we would see" if our eyes observed in that range.

As mentioned in the historical introduction, it is often useful to know exactly how bright an object appears in the sky. Since the invention of CCD (Charge Coupled Device) cameras this process has gotten a lot easier. The relative brightness at each pixel can be easily retrieved from CCD data. Normally brightness is measured across a large range of wavelengths, but sometimes it is useful to measure the intensity in specific ranges in order to give some indication of color.

The astronomical term used to indicate brightness is *magnitude*. The apparent magnitude m is measured on an inverse logarithmic scale such that a difference of 5 magnitudes between two objects corresponds to the smaller magnitude star producing 100 times more flux than the other:

$$\frac{F_2}{F_1} = 100^{(m_1 - m_2)/5}. \tag{1.1}$$

It is a strange scale whose function was more clear historically than it is now, but it is worth covering because I am bound to mention a "smaller magnitude" star later on and I want it to be clear what I mean.

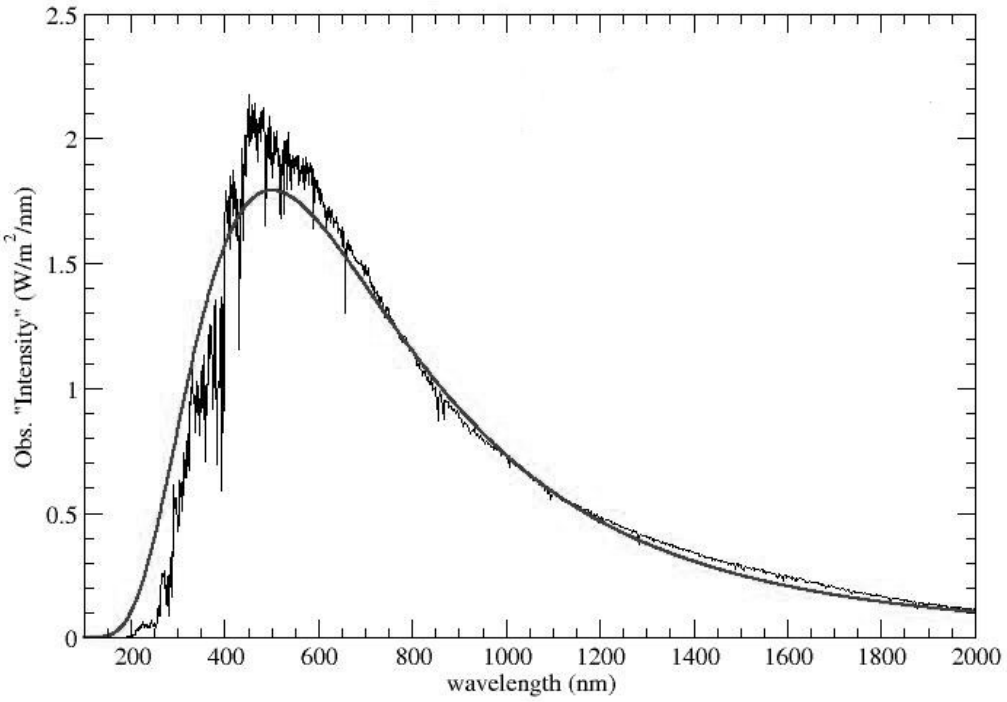

Figure 1.2: The true solar spectrum as well as a smooth 5777 Kelvin blackbody spectrum. Deep absorption and emission lines are clearly seen in the true solar spectrum. Image credit: Kirk T. Korista, Western Michigan University.

The most important way in which magnitude is applied in astrophysics is in determining distance. Distance in astrophysics is usually measured in *parsecs* (pc). The parsec is formally defined as *the distance to a star with a parallax angle of one arc-second*. Parallax angle is the amount the star moves in the sky when viewed from opposite sides of the Earth's rotation around the Sun . . . but if I were you I wouldn't worry about it too much. The parsec is also more historical than practical, but unfortunately it caught on and stuck. One parsec is equal to 3.26 light-years and 3.085×10^{16} meters. The "intrinsic brightness" of a star is usually characterized by the *absolute magnitude* M, which is defined to be *the apparent magnitude of the star if it were located at a distance of 10 parcecs from Earth*[2]. In a galaxy one is often interested in the spatial distribution of stars.

There is yet more information encoded in electromagnetic radiation. Possibly the

[2]It is common practice to characterize distance by the difference between the apparent and absolute magnitude

$$m - M = 5 \log_{10}(d) - 5. \tag{1.2}$$

This quantity is known as the *distance modulus*.

most widely used observational tool in astrophysics is the *spectrum*, which measures the *amount of radiation received at each wavelength*. The spectrum is incredibly useful because theoretical models often predict the amount of radiation a system should emit or absorb at specific wavelengths given certain characteristics. Temperature is a prime example: objects which are hotter will appear more blue, in the sense that blue corresponds to shorter wavelengths, while cooler objects will appear more red. Additionally, experiments on Earth have revealed that every molecule has a unique set of wavelengths at which it will absorb and emit photons. When observing a spectrum from an object in which a molecule is present, there will appear *spectral lines* at those specific wavelengths. The lines will appear bright if the molecule is emitting photons of that wavelength, and dark if the molecule is absorbing photons of that wavelength. Since we assume that the atomic structure of the universe is the same everywhere, it is possible to infer the chemical composition of a star from the position of its spectral lines. Incidentally, it was the spectrum of spiral nebulae which ultimately verified that they are composed primarily of stars, not gas.

Spectral lines can also be used to measure the velocity of astronomical objects relative to Earth. It is a well documented phenomenon known as the Doppler effect that sound coming from an object traveling towards the listener will have a higher pitch while sound traveling from a receding object will have a lower pitch. When the object is moving towards the listener, the distance between successive wave crests is reduced because the object moves closer to the listener over one oscillation. When the object is moving away from the listener the opposite is true. The same phenomenon occurs in astronomical observations with electromagnetic waves. When the object is moving towards the observer the wavelength (read "distance between wave crests") is shortened or *blue shifted*. If the object is moving away from the observer the wavelength will be longer, or *red shifted*. In a spectrum, red and blue shifts can be measured by the slight movement of a spectral line. Since the rest wavelength λ_{rest} of a specific line is known from observation on earth, any variation in the observed wavelength λ_{obs} indicates velocity. Usually this change in wavelength is described in terms of a *redshift parameter*:

$$z = \frac{\lambda_{\text{obs}} - \lambda_{\text{rest}}}{\lambda_{\text{rest}}}. \tag{1.3}$$

If the object is moving along the line-of-sight to the observer with velocity v_r then

$$z = \sqrt{\frac{1 + v_r/c}{1 - v_r/c}} - 1. \tag{1.4}$$

Sometimes, especially when observing distant galaxies, a collection of objects in view will be moving randomly both towards and away from the observer. This will cause the spectral lines to spread out in either direction. The "spread" of velocities is usually characterized in terms of the statistical *velocity dispersion*:

$$\sigma = \sqrt{\overline{v^2} - \overline{v}^2} \tag{1.5}$$

where an an overline indicates an average over all observed values.

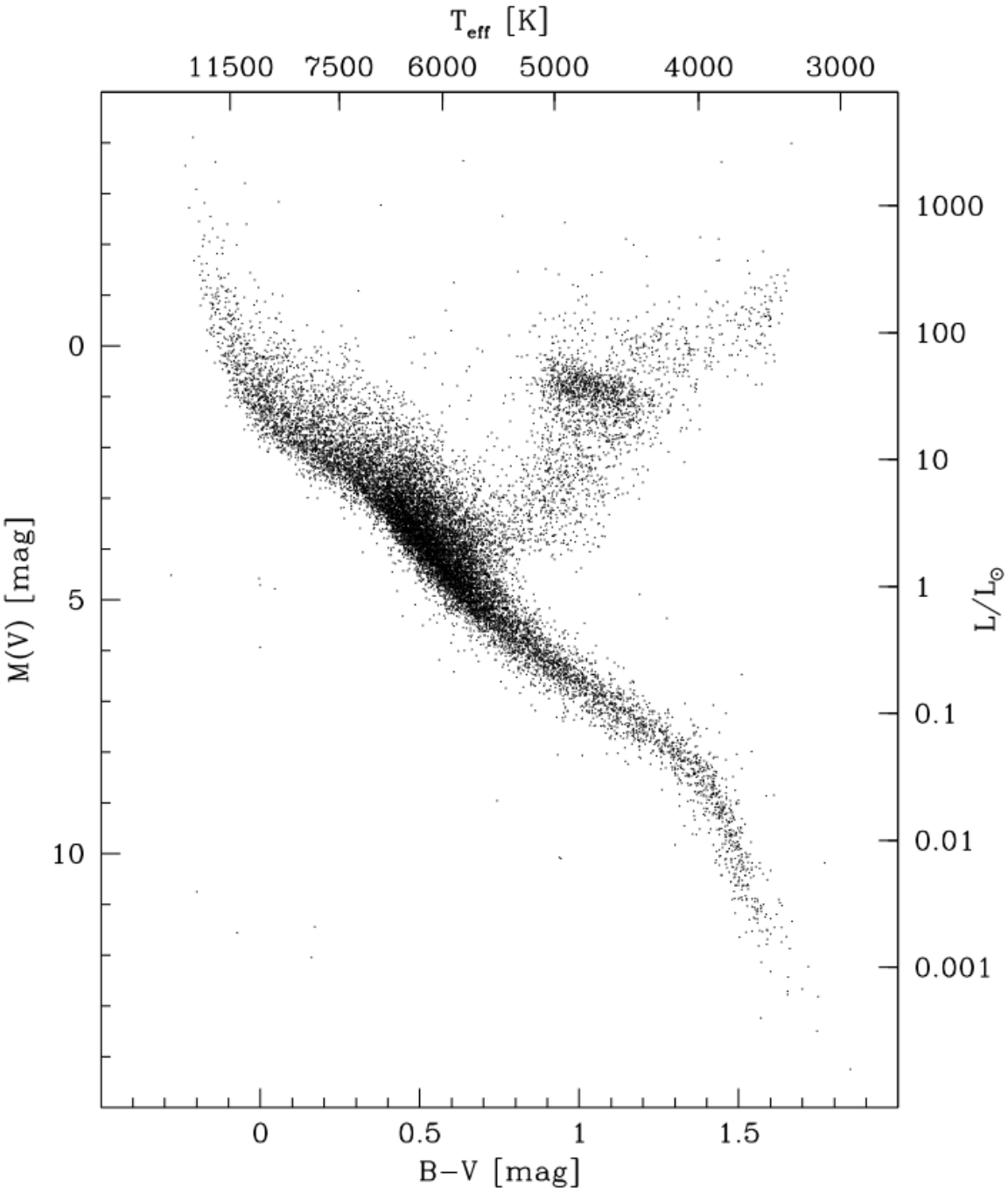

Figure 1.3: The Hertzsprung-Russell Diagram. The left and right axes represent magnitude and luminosity, respectively. The top and bottom represent temperature and color, as indexed by the difference in magnitude between blue light and visible light.Taken from J. Binney and S. Tremaine, *Galactic Dynamics* (Princeton University Press, Princeton, NJ, 2008), 2nd ed.

Much information about galaxies can be discerned from studying the stars which compose them. A complete description of stellar astrophysics would constitute an entire chapter on its own, but I will list a few important facts about the life cycles of stars which are especially relevant to galactic astrophysics. Stars form when clumps of hydrogen gas become dense enough to ignite nuclear fusion in their cores. In general, the rate at which stars form in a cloud of gas is greater if the density of that gas is greater. There has been much theoretical work attempting to accurately define a star formation rate as a function of density but it has proved exceptionally difficult (see Reference [10]).

Stars are generally categorized in terms of their position on the *Hertzsprung-Russell Diagram* (Fig. 1.3) which plots color (blue to red) vs magnitude. The thick band running from the upper left to the bottom right is known as the *main sequence*. This is where stars will spend the majority of their lifecycles. Stars in the upper left corner are more massive, brighter, and bluer. Because they are hotter they also exhaust their fuel very quickly and thus have very short lifespans, some less than 100 mega-years (Myr). Stars towards the bottom right corner are smaller, less bright, and redder. Because they burn fuel much slower, their lifespans are comparable to the age of the universe. Because blue stars burn away much quicker, in general the color of a population of stars is a good indication of its age. The stars branching upwards from the main sequence are known as giants and red giants. These are main sequence stars which have exhausted the supply of hydrogen in their core and are now burning hydrogen in the areas surrounding the core. When they reach the tip of the red giant branch they begin to burn helium, producing heavier elements in the process. At this point the most massive stars can become unstable, resulting in a *supernova* which blasts their contents back out in to space.

Any element other than hydrogen and helium was produced in some stellar process and subsequently ejected by supernova or normal stellar mass ejection. Thus it is useful to quantify the fraction of the total mass of a star composed of heavier elements. This quantity, known as the *metalicity*, can be determined from analysis of spectral lines and reveals important information about the age of a stellar population.

1.2 Galactic Morphology

In order to orient the dizzying discussion that is to come, it will be useful to describe some general morphological properties of galaxies. Most simply stated, a galaxy is a large, gravitationally bound assembly of stars or other masses. This is a problematic definition, however, because a large number of celestial objects would qualify which we don't *really* consider galaxies. More specifically, galaxies have to be *really* big (1 to 100 kilo-parsecs (kpc) across), have *a lot* of stars (10^7 to 10^{14} stars), and be separated from one another by distances on the order of millions of parcecs. Kant was somewhat correct: a galaxy must be enough of an island to be considered as a unit of large-scale cosmological structure.

Edwin Hubble developed a morphological classification scheme for galaxies that is still in use today (Fig. 1.4). Broadly, he divided galaxies into three categories:

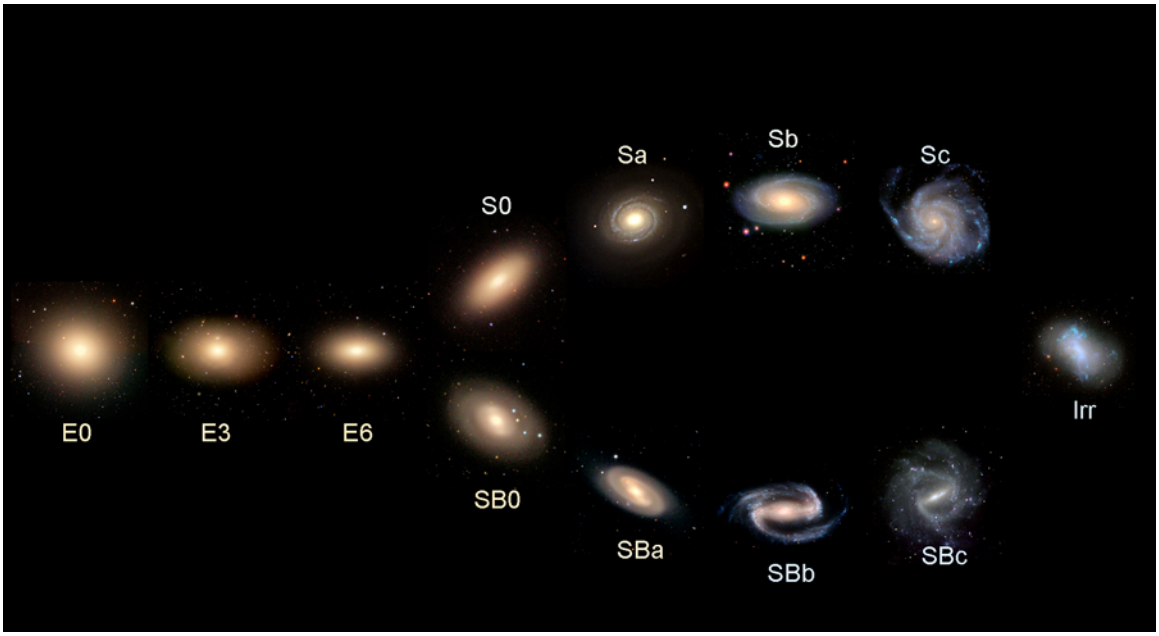

Figure 1.4: The Hubble Sequence. Taken from *Normal Galaxies*, http://hendrix2.uoregon.edu/ imamura/123/lecture-3/ lecture-3.html.

ellipticals, *spirals*, and *irregulars*. Elliptical galaxies were divided into nine numerical categories depending on how elongated they were. On one end, E0 galaxies are nearly circular while on the other end E9 galaxies are very flattened. Spiral galaxies were subdivided into two parallel sequences (S and SB) depending on whether or not a straight bar was present in the spiral pattern. Within each sequence there were three categories: Sa and SBa galaxies which have the most prominent central bulges and most tightly wound spiral arms, Sb and SBb galaxies which have moderately sized bulges and slightly less tightly wound spirals, and Sc and SBc galaxies which have small bulges and spiral arms which resolve into clumps of stars and gas. Additionally, there is a transitional category between ellipticals and spirals known as the lenticulars (S0 or SB0). Hubble falsely believed that this was an evolutionary sequence. Consequently galaxies on the left of the diagram are often referred to as "early type" galaxies while galaxies on the right are referred to as "late type."

These categories are somewhat arbitrary. For instance, the Hubble classification of elliptical galaxies might not be related to the true ellipticity of a galaxy because we can only see it along the line of sight to Earth. Nonetheless, they successfully organized the study of galaxies, especially of late-type spiral galaxies.

The main visible component of spiral galaxies is the stellar disk. According to the most widely accepted theory of galaxy formation, disks form if the cloud of primordial matter which collapsed into a galaxy had some net angular momentum. To see why this is true, consider what happens when you pulse a blender. The blender blades impart a net angular momentum to your smoothie by causing it to spin in a uniform direction. The contents of your smoothie (if properly liquified) fly outwards towards the wall of the blender and stay there for a while after you stop the pulse. An

Figure 1.5: NGC 1376, a classic Sc class spiral galaxy. Taken from R. Thompson, *Spiral galaxy ngc 1376*, `http://www.spacetelescope.org/images/opo1000a/`

isolated system must conserve both energy and angular momentum. Objects farther away from the center can "take care of" more of the net angular momentum while having a negligible impact on the total energy.

Stellar disks are usually described in *galactic coordinates*, which is just a fancy name for cylindrical coordinates with the origin placed at the center of the galaxy and the z axis pointing straight up and down from the disk. They are often characterized in terms of their *surface brightness* $I(r, \phi)$, or *the total luminosity emitted per unit area of the disk*. Observations of the Milky Way and other disk galaxies suggest that in general I can be approximated as an exponential function of radius:

$$I(r) = I_0 e^{-r/r_0}, \tag{1.6}$$

where I_0 is the brightness at the center and r_0 is some scale length typically on the order of 2 to 3 kilo-parcecs (kpc). More exotic models are used depending on the specific system of interest.[3] In the direction perpendicular to the plane the luminosity often takes the functional form of a hyperbolic secant so that the total luminosity density is modeled as

$$L(r, z) = L_0 e^{-r/r_0} \mathrm{sech}^2(z/z_0) \tag{1.7}$$

[3]In fact, the main result of this thesis (Chapter 4) uses a non-exponential model of an Sc class galaxy.

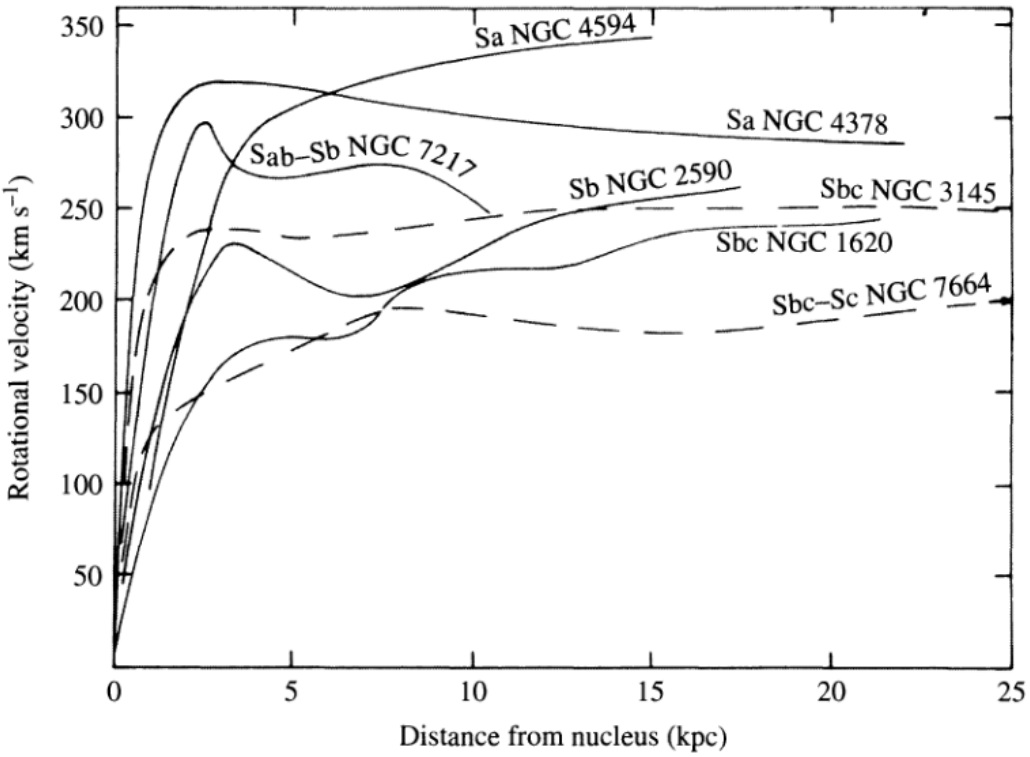

Figure 1.6: Rotation curves of several spiral galaxies. Notice the curves are relatively flat past ~ 4 kpc. Taken from B. W. Carroll and D. A. Ostlie, *An Introduction to Modern Astrophysics* (Pearson Education, Inc, San Francisco, CA, 2007), 2nd ed.

where z_0 is a vertical scale height (usually 100 - 1000 pc). It is also common to model the vertical distribution with a Gaussian so that

$$L(r, z) = L_0 e^{-r/r_0} e^{-z^2/(2z_0^2)}. \tag{1.8}$$

Observations show that the scale height z_0 is remarkably independent of r.

Usually the stellar disk is composed of two independent populations: a thick disk composed of older, redder stars which formed in the early history of the galaxy, and a thin disk composed of younger, bluer stars.[4] In this case the distribution can be modeled as the sum of two decreasing exponentials.

Most stars in the stellar disk travel in nearly circular orbits around the center of the galaxy. The average speed v_c of a star in a circular orbit at radius r is a quantity of great theoretical interest because it depends directly on the distribution of mass. It's easy to see why: in a simplified system which is spherically symmetric, the centripetal force $F_c = mv_c^2/r$ must balance the gravitational attraction from the total mass M_r which lies interior to the radius r:

$$\frac{mv_c^2}{r} = \frac{GM_r m}{r^2}.$$

[4]The Sun is a member of the thin disk in the Milky Way. We know this because it has very high metalicity compared to the usual thick disk population.

Thus M_r can be found directly from v_c:

$$M_r = \frac{v_c^2 r}{G}. \tag{1.9}$$

In more realistic systems which are not spherically symmetric the math is more complicated but the principle is the same. The plot of v_c vs. r is known as the *rotation curve*. In most cases almost all stars in the stellar disk orbit in the same direction (clockwise or counterclockwise). This is not required, however. In some cases there are populations of stars which orbit in both directions.

Interspersed in the stellar disk are large patches of gas, usually atomic and molecular hydrogen, concentrated in clouds of widely ranging shape and size. In most cases these hydrogen clouds are neutral and therefore referred to as HI regions after the spectral line which indicates its presence. In some cases when the gas surrounds a very bright, hot star it becomes ionized. These regions are referred to as HII regions after another characteristic spectral line. In general the total mass of interstellar gas is less than 15% of the mass of the stellar disk, yet it plays an essential role in galactic chemistry since all new stars are formed from clouds of gas.

Almost all disk galaxies display some sort of *spiral structure* (Fig. 1.5). Spirals are the iconic filaments of dust and stars which give galaxies their distinctive shape. The most dramatic spiral galaxies, known as *grand design spirals* have only two, very well defined arms. Others have multiple arms, or in some cases just broken up pieces which still form some kind of spiral pattern. Because the density of gas is higher in the arms, they are generally regions of active star formations. Therefore they are populated by bright young stars which dominate optical images. They are thought to be a phenomenon mainly, though perhaps not exclusively, of the thin disk. Additionally, they are thought to play a critical role in the redistribution of angular momentum and to increase random velocity, which as we'll see later can lead to greater stability.

Intuitively you might expect that spiral arms are composed of a fixed collection of stars and gas, but unfortunately this leads to a contradiction known as the *winding problem*. The dilemma is best understood by considering a set of stars which are initially oriented along a single line (Fig. 1.7). Since masses at different radii do not rotate at the same angular rate, any arm-like collection of mass will *wind up* tighter than any observed spiral over a period of just a couple rotations. There must be another mechanism to explain the presence of spiral structure.

In the mid-1960s the American astronomers C.C. Lin and Frank Shu suggested that spiral arms might actually be *quasistatic density waves* which propagate through the stellar disk much like a sound wave would propigate through air. Though not consisting of any constant group of masses, a spiral arm could exist in a relatively constant form for a very long time. The Lin-Shu hypothesis *must* be true in some sense to allow for the presence of spiral arms without the winding-catastrophe. However the mechanisms by which a density wave may propagate through a uniformly rotating disk are not well understood. Hence the density-wave theory of spiral arms is an active area of research [11].

More than half of all spiral galaxies also display a straight central *bar* (Fig. 1.8).

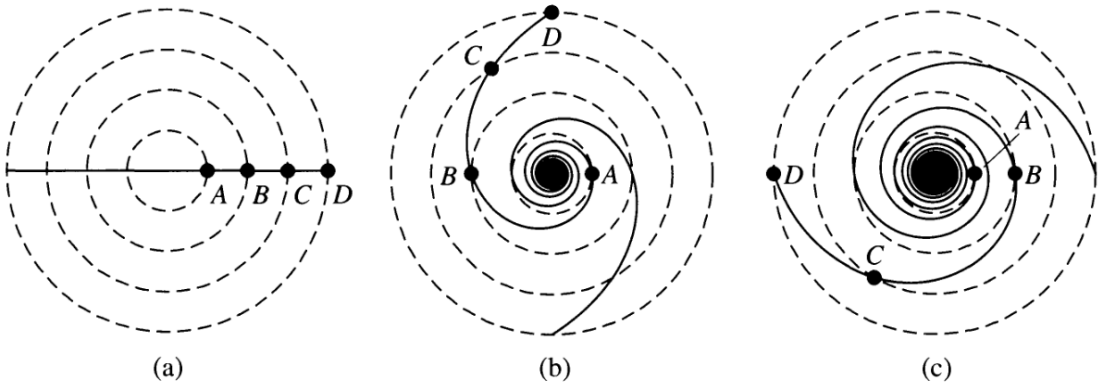

Figure 1.7: The winding problem for material arms. In (a) it is assumed that masses A, B, C, are D are members of a fixed collection of mass initially oriented in a straight horizontal line. As an approximation to the rotation curves in Fig. 1.6 it is assumed that each mass is traveling with the same circular speed along orbits represented by the dotted lines. In (b) the system is shown some time later, the collection of masses has stretched into a spiral pattern. In (c) the system is shown later still, the spiral has wound up tighter than any typically observed in the sky. Taken from B. W. Carroll and D. A. Ostlie, *An Introduction to Modern Astrophysics* (Pearson Education, Inc, San Francisco, CA, 2007), 2nd ed.

Bars range greatly in size, some are nearly as wide as the entire disk. They are remain rigid as they rotate, and generally rotate at a relatively quick pace. Within the bar, the surface brightness is relatively constant but drops steeply at the bar edges. Bars, like spirals, are thought to play a significant role in the redistribution of angular momentum. It is also thought that they could *drive* spiral patterns in the outer disk.

Many spiral galaxies contain a roughly spherical *central bulge*. The bulge is characterized by a high degree of random motion, as opposed to the nearly-circular orbits of the disk. It is usually responsible for less than 15% of the total luminosity. In most if not all galaxies there is a supermassive black hole in the center with a mass on the order of 10^{12} solar masses. Though supermassive black holes play a significant role near the center, their effect on the global dynamics of the disk is negligible.

About 1% of the total stellar mass in a typical spiral galaxy is part of a *stellar halo* which is roughly spherical and extends far beyond the stellar disk (out to a radius of about 50 kpc). The stars in this halo usually have very low metalicity and are often members of globular clusters (large groups of stars).

The *vast* majority of the total mass of a galaxy is contained in mysterious matter that we can't see and don't understand: dark matter. It was through the study of the rotation curve $v_c(r)$ in spiral galaxies that dark matter was first "discovered." Typically $v_c(r)$ increases linearly near the origin, which is what you would expect from a rigid body rotating at a constant angular rate, but levels out past a few kpc (Fig. 1.6).

The fact of the matter is that we *simply cannot observe* enough mass to account

Figure 1.8: NGC 6127, a classic bar spiral galaxy. Taken from J. Voisey, *Psa: Bars kill galaxies*, `http://www.universetoday.com/77810/psa-bars-kill-galaxies/` (November 2010).

for these rotation curves. It's not just a little bit off, it's *two orders of magnitude* off. Thus there must be a large amount of mass in the universe which we cannot see.

We now believe that every galaxy is nested in it's own *dark matter halo* that extends far (up to 200 kpc) beyond the edges of luminous matter. One popular model derived from studying dark matter clustering on a cosmological scale is known as the NFW (Navarro, Frenk, and White) profile [12],

$$\rho_{NFW}(r) = \frac{\rho_0}{\left(\frac{r}{r_0}\right)\left(1+\frac{r}{r_0}\right)^2}. \tag{1.10}$$

It is not clear whether the NFW profile is a valid model for the dark matter halo of a single galaxy, but it is reasonable and well studied so it is often used anyways. Dark matter typically composes 95% of all mass in a typical galaxy. The other 5% is the stuff we get to see.

A stated goal of this thesis is to study the formation and evolution of spiral arms and their relation to dark matter. These processes each involve the *motion* of galaxies as they interact with themselves and with other galaxies. Unfortunately the timescales

over which galaxies evolve are unimaginably large compared to a human lifetime: motion cannot be observed. Not permitted the luxury of sight, these questions must be addressed through the creation of a *model*.

The next chapter is concerned with the construction of this model using the tools made available by the fundamental theories of physics. As is the nature of physics, this will require a great number of simplifications and assumptions as we move from the realm of astronomy to the realm of mathematics. The hope is that once the dust clears we might look around to discover in that model something new about the universe in the sky.

Chapter 2

Theory

The task at hand is to study the structure and evolution of spiral patterns and their relation to dark matter by creating a simplified theoretical model. The weapons in hand are the well-verified theories of theoretical physics. Before any headway is made we must determine the angle of attack: what simplifications are necessary to make this problem tractable in the scope of this thesis?

To start with, let's do away with all consideration of interstellar gas as a dynamical entity. Gas composes less than 15% of the mass in a galactic disk, so stars are the much more important dynamical entity. Let's also throw out all matters concerning the internal dynamics and evolution of stars. This seems reasonable since stars are very, very small compared to a galaxy. As one last simplification, let's assume that all stars have the same mass m. Again, on the galactic scale mass differences among stars are negligible. Naturally with each simplification we're loosing some amount of potentially important information, but the practical costs of these extra considerations outweigh the loss of realism.

Given this imaginary galaxy of identical stars which exist in their current state now and forever, which physical theories will be necessary to describe its dynamics? As far as gravity is concerned, Newtonian gravity should be very sufficient. General relativity only differs from Newton's theory very close to large massive objects and on a cosmological scale. Luckily galaxies avoid either extreme. Even in the case of a supermassive black hole at the center of a galaxy, the region affected by general relativistic weirdness is small compared to the size of the galaxy. It follows that classical mechanics should be completely adequate to describe the motion of stars. As far as other theories are concerned, galaxies are far too large for electric or magnetic fields to play a significant role, not to mention any more exotic forces.

So there you have it: in order to gain insight into the origin and evolution of spiral structure in disk galaxies, we will be studying a simplified model consisting of identical point-mass stars which move classically according to Newtonian gravity. Don't be fooled, there is still a trillion lifetimes worth of information to wring out of this simplified model. What has been achieved is focus and direction: by limiting the scope of phenomena considered, it becomes clear what types of analysis will be necessary to proceed.

2.1 Collisonless Stellar Systems

A typical disk galaxy contains between 200 and 400 *billion* stars. There is no hope whatsoever of describing exactly the motion of each individual star and even if there was I'm not sure what we'd do with all that information. In general we are more interested in *statistical* information about the movement of stars. This is nothing revolutionary, physicists are well versed at studying systems using statistical mechanics when exact information is unknowable. There is a fundamental difference, however, between galaxies and the systems typically studied in statistical mechanics because of the nature of the forces involved. A typical molecule in a gas has little or no net charge so unless they are very close together, intermolecular forces are very small. Consequently they are subject to short, violent accelerations as they "collide" in between much longer periods in which they travel with nearly constant velocity.

The gravitational forces between stars in a galaxy, on the other hand, act over a very long range. Since mass is always positive, a star is subject to the gravitational influence not only of its nearest neighbors but of the entire distribution of mass. It might even seem reasonable to expect that the *majority* of the force on a given star is due to the overall distribution of mass rather than near neighbors. If this were true we could treat the system as a continuous density distribution and a resulting smooth gravitational potential rather than a collection of point-mass stars.

This sounds like a great idea, but it will be necessary to get a more quantitative sense of how accurate a continuous approximation would be. Naturally if there were to occur significant deviation from the continuous model it would occur through the repeated close encounters of pairs of stars. One quantitative measure of the effect of these encounters is known as the *relaxation time* which measures the time required for the velocity of a typical star to change significantly due to close encounter. More specifically, it is defined as the time at which the sum of the *squares* of the velocity changes δv due to close interactions is equal to the original squared-velocity v^2:

$$\Delta v^2(t_{\text{relax}}) = \sum \delta v^2 = v^2. \tag{2.1}$$

This is a bit of a complicated concept. The idea is to keep track of the total amount that close encounters have effected the velocity. The square is used so that the total change is always positive and never cancels. The condition $\Delta v^2 = v^2$ is a benchmark for measuring when the total change is *significant*.

In order to estimate the relaxation time it will first be necessary to solve the gravitational two-body problem to find the change in velocity δv. Intuitively we expect that δv will be related to *how close together* two stars come in a typical interaction. Therefore δv should be related to some measure of initial separation which can be compared to those

Consider an isolated gravitational interaction of two masses m_1 and m_2 located

at positions $\mathbf{x}_1$ and $\mathbf{x}_2$ respectively.[1] The Lagrangian of the system is,

$$\mathcal{L}_{\text{tot}} = \frac{1}{2}\left(m_1\dot{\mathbf{x}}_1^2 + m_2\dot{\mathbf{x}}_2^2\right) + G\frac{m_1 m_2}{|\mathbf{x}_1 - \mathbf{x}_2|}. \tag{2.2}$$

Because the system is isolated, the center of mass should move in a straight line with constant velocity while the *relative position* contains the interesting physics. Thus it seems like a good idea to introduce a new coordinate system which separates these two pieces of information. Let

$$\mathbf{R} = \frac{m_1\mathbf{x}_1 + m_2\mathbf{x}_2}{m_1 + m_2} \quad , \quad \mathbf{r} = \mathbf{x}_1 - \mathbf{x}_2 \tag{2.3}$$

where $\mathbf{R}$ is the center of mass, $\mathbf{r}$ is the separation vector between the two masses. Then

$$\mathcal{L}_{\text{tot}} = \frac{1}{2}(m_1 + m_2)\dot{\mathbf{R}}^2 + \frac{1}{2}\frac{m_1 m_2}{m_1 + m_2}\dot{\mathbf{r}}^2 + G\frac{m_1 m_2}{r}. \tag{2.4}$$

It seems natural to define the quantities

$$M = (m_1 + m_2) \quad , \quad \mu = \frac{m_1 m_2}{m_1 + m_2}, \tag{2.5}$$

where μ is known as the *reduced mass* of the system and M is of course the total mass, so that

$$\mathcal{L}_{\text{tot}} = \frac{1}{2}M\dot{\mathbf{R}}^2 + \frac{1}{2}\mu\dot{\mathbf{r}}^2 + G\frac{\mu M}{r}. \tag{2.6}$$

In this form the terms involving $\mathbf{R}$ and the terms involving $\mathbf{r}$ are completely independent. In fact, the total Lagrangian can be written

$$\mathcal{L}_{\text{tot}} = \mathcal{L}_{\text{CM}} + \mathcal{L}$$

where $\mathcal{L}_{\text{CM}}$ describes the motion of the center of mass (in a straight line with constant velocity) and $\mathcal{L}$ describes all the juicy relative motion stuff:

$$\mathcal{L} = \frac{1}{2}\mu\dot{\mathbf{r}}^2 + G\frac{\mu M}{r}. \tag{2.7}$$

Because two-body gravitational interaction conserves angular momentum, motion occurs entirely in a plane. It is therefore natural to use cylindrical coordinates (r, ϕ, z) with z oriented normal to the plane of motion so that

$$\mathcal{L} = \frac{1}{2}\mu[\dot{r}^2 + (r\dot{\phi})^2] + G\frac{\mu M}{r}. \tag{2.8}$$

Since the equations of motion are unchanged by the presence of a total constant, μ can be divided out so that the Lagrangian in its final form is

$$\mathcal{L} = \frac{1}{2}[\dot{r}^2 + (r\dot{\phi})^2] + G\frac{M}{r}. \tag{2.9}$$

[1] I will be using bold-face symbols to denote vector quantities and corresponding unbold symbols to represent the norm of those vectors. For example, v is the norm of the vector $\mathbf{v}$. Squared vector quantities indicate a dot product, $\mathbf{v}^2 = \mathbf{v}\cdot\mathbf{v} = v^2$.

What we have here is a Lagrangian *per unit mass*. This is a feature of gravity: the motion of a test particle in a gravitational potential is independent of its mass.

The next task is to solve the equations of motion. From the Euler-Lagrange equations they are

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{r}}\right] - \frac{\partial \mathcal{L}}{\partial r} = 0 \longrightarrow \ddot{r} - r\dot{\phi}^2 = -G\frac{M}{r^2}, \tag{2.10a}$$

$$\frac{d}{dt}\left[\frac{\partial \mathcal{L}}{\partial \dot{\phi}}\right] - \frac{\partial \mathcal{L}}{\partial \phi} = 0 \longrightarrow \frac{d}{dt}\left(r^2\dot{\phi}\right) = 0. \tag{2.10b}$$

The second equation implies that there is a conserved quantity

$$L \equiv r^2\dot{\phi}. \tag{2.11}$$

It's not hard to see that this is just the magnitude of the vector $\mathbf{r} \times \mathbf{v}$, L is the angular momentum *per unit mass* so that (2.10b) is just a restatement of the conservation of angular momentum.[2] Solving for each coordinate as a function of time is messy and not very useful but luckily (2.11) implies that

$$\frac{d}{dt} = \frac{L}{r^2}\frac{d}{d\phi} \tag{2.12}$$

so the angle ϕ can replace t as the independent variable. With this substitution (2.10a) becomes

$$\frac{L^2}{r^2}\frac{d}{d\phi}\left(\frac{1}{r^2}\frac{dr}{d\phi}\right) - \frac{L^2}{r^3} = -G\frac{M}{r^2}. \tag{2.13}$$

One more substitution will make this whole thing a lot prettier: let $u = 1/r$ and note that

$$dr = -\frac{1}{u^2}du.$$

Then (2.13) becomes

$$-L^2u^2\frac{d^2u}{d\phi^2} - L^2u^3 = -GMu^2, \tag{2.14}$$

or more simply

$$\frac{d^2u}{d\phi^2} + u = \frac{GM}{L^2}. \tag{2.15}$$

In the end all the substitutions pay off, the equation is simple and there exists a solution! Specifically,

$$u = \frac{1}{r} = C\cos(\phi - \phi_0) + \frac{GM}{L^2} \tag{2.16}$$

where C and ϕ_0 are constants determined by the initial conditions.

The physical situation of interest is a scattering interaction between two stars in a galaxy. Remember that we're trying to determine the total change in velocity given some measurement of the separation between the stars in order to determine

[2] Since we used a Lagrangian *per unit mass*.

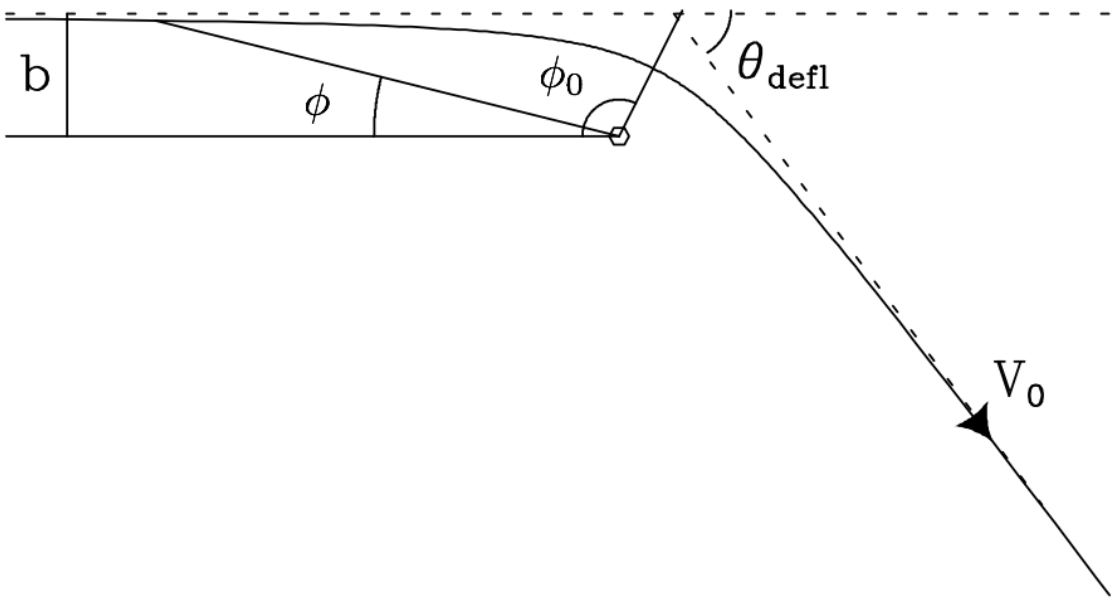

Figure 2.1: The motion of the reduced mass during a close gravitational encounter. Modified from J. Binney and S. Tremaine, *Galactic Dynamics* (Princeton University Press, Princeton, NJ, 2008), 2nd ed. p. 154.

the relaxation time. Suppose that at $t = -\infty$ the initial relative velocity of the two masses is $\mathbf{v}_0$ and that the component of the initial separation vector r which is perpendicular to $\mathbf{v}_0$ has length b (Fig. 2.1). b is known as the *impact parameter* and is a good measurement of how close together two stars will come. In fact, if there were no gravitational interaction, b would be the separation of the two masses at closest pass. The magnitude of the conserved angular momentum is related to these quantities since

$$L = |\mathbf{r} \times \mathbf{v}| = r_\perp v_0 = b v_0. \tag{2.17}$$

Plugging this in to (2.16) gives the condition

$$\frac{1}{r} = C\cos(\phi - \phi_0) + \frac{GM}{b^2 v_0^2}. \tag{2.18}$$

Another equation can be generated by differentiating the above with respect to time,

$$\frac{dr}{dt} = Cr^2\dot{\phi}\sin(\phi - \phi_0), \tag{2.19}$$

or since $r^2\dot{\phi} = L = bv_0$,

$$\frac{dr}{dt} = Cbv_0\sin(\phi - \phi_0). \tag{2.20}$$

Note that at $t = -\infty$, $dr/dt = -v_0$ and $1/r = 0$. If we define $\phi = 0$ to point towards the particle as $t \longrightarrow -\infty$ then (2.20) evaluated at $t = -\infty$ gives

$$-v_0 = Cbv_0\sin(-\phi_0). \tag{2.21}$$

Additionally, (2.18) evaluated at this same time gives

$$0 = C\cos\phi_0 + \frac{GM}{b^2 v_0^2}. \tag{2.22}$$

Eliminating C between these two equations leads to the condition

$$\tan\phi_0 = -\frac{bv_0^2}{GM}. \tag{2.23}$$

It is clear from (2.18) that the separation between the masses is least when $\phi = \phi_0$. Since the orbit is symmetric about this point,[3] the angle θ_{defl} through which the relative velocity is deflected is equal to $2\phi_0 - \pi$ (Fig. 2.1). With only a little massaging of (2.23) comes the result

$$\theta_{\text{defl}} = 2\tan^{-1}\left(\frac{GM}{bv_0^2}\right). \tag{2.24}$$

This is a good place to be: we've managed to find the amount that the masses are deflected given the parameters b and v_0. It is useful to define the quantity

$$b_{90} = \frac{GM}{v_0^2} \tag{2.25}$$

so that

$$\theta_{\text{defl}} = 2\tan^{-1}\left(b_{90}/b\right). \tag{2.26}$$

Qualitatively, b_{90} is the impact parameter at which $\theta_{\text{defl}} = 90^\circ$ so it serves to define an important scale for the interaction. If b is in the region of b_{90} the masses will deflect widely, otherwise they will not. If we assume that $M = 2M_\odot$[4] and $v_0 = 30$ km/s, a typical relative-velocity between stars in the Milky Way, then $b_{90} = 1.47 \times 10^{11}$ m or 1.01 AU.[5] On the galactic scale this length is *extremely* small. For comparison, the distance to the Sun's nearest neighbor Alpha Centauri is 266,078 AU. Thus it is safe to assume that for a typical interaction, $b \gg b_{90}$.

Because $b \gg b_{90}$ the total change in velocity will be small and mostly in the direction perpendicular to the original velocity $\mathbf{v}$. The change in perpendicular velocity $\Delta\mathbf{v}_\perp$ is easy to calculate given θ_{defl}:

$$\begin{aligned}
|\Delta\mathbf{v}_\perp| = v_0\sin\theta_{\text{defl}} &= v_0\sin\left[2\tan^{-1}\left(b_{90}/b\right)\right] \\
&= 2v_0\sin\left[\tan^{-1}\left(b_{90}/b\right)\right]\cos\left[\tan^{-1}\left(b_{90}/b\right)\right] \\
&= 2v_0\frac{b_{90}/b}{\sqrt{1+b_{90}^2/b^2}}\cdot\frac{1}{\sqrt{1+b_{90}^2/b^2}} \\
&= 2v_0\frac{b_{90}/b}{1+b_{90}^2/b^2}
\end{aligned} \tag{2.27}$$

[3] By this I mean that (2.18) is even about $\phi = \phi_0$

[4] $M_\odot$ is the mass of the Sun.

[5] An AU or Astronomical Unit is the distance from the Earth to the Sun.

In the limit $b \gg b_{90}$

$$|\Delta \mathbf{v}_\perp| \approx 2v_0 \frac{b_{90}}{b} = \frac{2GM}{bv_0}. \tag{2.28}$$

Of course we're really interested in the change in the velocity of *one* star, while until now we've been solving for the change in relative velocity. Since any two stars in our model galaxy have the same mass m, the change in the velocity $\Delta \mathbf{v}_m$ of either star is one half of the relative velocity change

$$|\Delta \mathbf{v}_m| = \frac{1}{2}|\Delta \mathbf{v}|. \tag{2.29}$$

If we replace the relative velocity v_0 with v, the typical speed of a star then the total velocity change δv due to the interaction is

$$\delta v = |\Delta \mathbf{v}_{m,\perp}| \simeq \frac{2Gm}{bv}. \tag{2.30}$$

The first task is done: we've estimated the change in velocity δv imparted to a star in a typical close encounter. Now the task is to consider the effect of *repeated* close encounters given the size and number of stars in the system. Out of necessity this will be a very rough, order-of-magnitude type calculation. Even though we're interested in disk galaxies, to make the math easier let's consider a uniform spherical system of N stars and radius R. At the order-of-magnitude level, the result should provide insight into the more complicated processes of a disk if the size of the system is similar.

As a star moves through the system, the number of encounters δn with impact parameter between b, $b + db$ occurring in time t is equal to the volume in which those encounters can occur times the number density of stars in the system,

$$\delta n = (2\pi\, b\, db\, vt) \cdot \left(\frac{N}{\frac{4}{3}\pi R^3} \right) = \frac{3Nvt\, b\, db}{2R^3}. \tag{2.31}$$

This is a bit of a complicated argument: the idea is that all stars for which the impact parameter with the test star is b are located in a ring around the path of the star with radius b. As the star moves through space that volume is traced out along a path of length vdt. The total number of encounters is equal to the volume of the hollow cylinder times the number density of stars which is assumed to be uniform. Figure 2.2 shows it best.

As an approximation, the total change in squared velocity due to encounters with impact parameter b is just the average velocity change from the previous derivation times the number of encounters. Thus at a time t,

$$\sum \delta v^2 \simeq \delta v^2 \delta n = \left(\frac{2Gm}{bv} \right)^2 \frac{3Nvt}{2R^3} b\, db = 6N \left(\frac{Gm}{Rv} \right)^2 \frac{vt}{R} \frac{db}{b}. \tag{2.32}$$

Integrating this result over all impact parameters from $b_{\min}$ to $b_{\max}$ gives the total

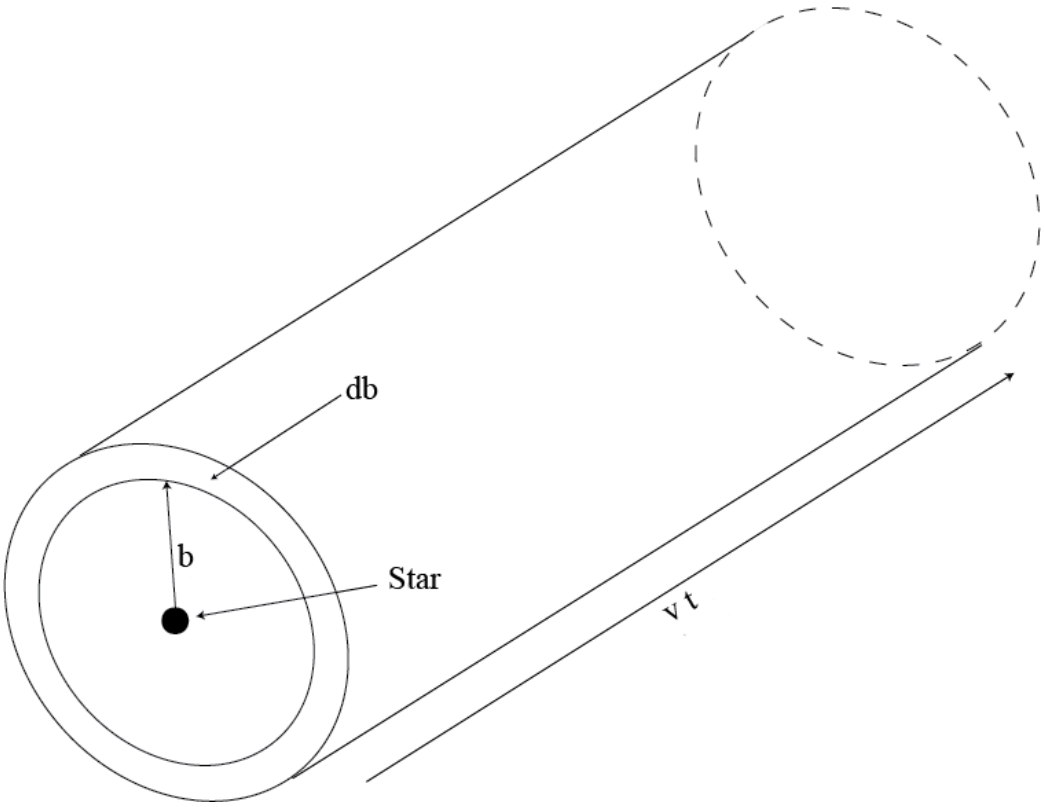

Figure 2.2: The volume of encounters with impact parameter b traced out as a star moves through the system.

change in squared velocity

$$\Delta v^2 = \int_{b_{\min}}^{b_{\max}} \sum \delta v^2 \simeq 6N \left(\frac{Gm}{Rv}\right)^2 \frac{vt}{R} \int_{b_{\min}}^{b_{\max}} \frac{db}{b}$$
$$= 6N \left(\frac{Gm}{Rv}\right)^2 \frac{vt}{R} \ln \Lambda, \tag{2.33}$$

where

$$\ln \Lambda = \ln \left(\frac{b_{\max}}{b_{\min}}\right) \tag{2.34}$$

is known as the Coulomb logarithm. As an absolute upper bound, set $b_{\max} = R$ since the impact parameter can't be bigger than the system. As a lower bound, it makes sense to set $b_{\min} = b_{90}$ since, as discussed above, b_{90} is very small on the scale of stellar interactions. In that case

$$\ln \Lambda = \ln \left(\frac{R}{b_{90}}\right) = \ln \left(\frac{Rv^2}{2Gm}\right). \tag{2.35}$$

Suppose that typical speed of a star is roughly that of a particle in a circular orbit at the edge of the galaxy. Then for all stars,

$$v^2 \approx \frac{GNm}{R}. \tag{2.36}$$

This is another weird assumption, but again I remind you that this is just an order-of magnitude affair. This can clear v^2 in (2.35) so that

$$\ln \Lambda = \ln \left(\frac{N}{2} \right) \approx \ln N. \tag{2.37}$$

We may also use (2.36) to clear Gm/R in (2.33) so that

$$\Delta v^2 = 6N \left(\frac{v}{N} \right)^2 \frac{vt}{R} \ln N \tag{2.38}$$

or written another way,

$$\frac{\Delta v^2}{v^2} = \frac{6 \ln N}{N} \frac{vt}{R}. \tag{2.39}$$

Recall from way back at (2.1) that the whole point of this lengthy endeavor was to solve for the time at which $\Delta v^2 = v^2$. Well there it is staring us in the face! Setting (2.39) equal to 1 and solving for t gives the result

$$t_{\text{relax}} \simeq \frac{N}{6 \ln N} \frac{R}{v}. \tag{2.40}$$

The quantity R/v on the right has units of time, indicating that it serves as some kind of time-scale for the system. Indeed, a commonly used time-scale is the *crossing time*

$$t_{\text{cross}} = \frac{R}{v}, \tag{2.41}$$

or one half of the time it would take for a star to pass from one side of the system to the other, if it were traveling in a straight line. Thus finally

$$t_{\text{relax}} \simeq \frac{N}{10 \ln N} t_{\text{cross}} \tag{2.42}$$

where I have rounded to 10 from the factor of 6 to be consistent with the result in Reference [9] Chapter 1.

Here are some numbers: for a normal sized galaxy with $N = 10^{11}$, $t_{\text{relax}}/t_{\text{cross}}$ is on the order of 10^8. This is *enormous* considering that galaxies are usually only a couple hundred crossing times old. Thus even over the age of the universe a galaxy will deviate only *very* slightly from the continuous approximation. This is a great result, one that is dramatic enough (I hope) to subsume all of the weird approximations used in its derivation. Systems with large relaxation times are generally referred to as *collisionless* systems. For comparison a globular cluster usually has $N \sim 10^5$ so $t_{\text{relax}}/t_{\text{cross}} \sim 800$. A globular cluster is typically several thousand crossing-times old, so the continuous approximation is not valid. Systems like this are generally referred to as *collisional* systems.

2.2 Gravity of Continuous Distributions

The importance of the result from the previous section cannot be overstated. Because a galaxy is approximately continuous on the large scale we may replace the collection of discrete stars with a continuous density distribution $\rho(\mathbf{x})$. A picture of the total dynamics of the galaxy can be obtained by studying the orbits of mass under the gravitational influence of $\rho(\mathbf{x})$.

Gravitational potential theory for continuous distributions should be familiar enough as an analogue to electricity and magnetism, but it is worth discussing briefly if only so I can reference the equations later on. According to Newton's law of universal gravitation, the total force $\mathbf{F}(\mathbf{x})$ on a mass m_s may be obtained by summing over the contributions

$$d\mathbf{F}(\mathbf{x}) = Gm_s \frac{\mathbf{x}' - \mathbf{x}}{|\mathbf{x}' - \mathbf{x}|^3} \rho(\mathbf{x}') d^3\mathbf{x}' \tag{2.43}$$

from a volume $d^3\mathbf{x}'$ at a location $\mathbf{x}'$. More often we're interested in the force per unit mass (otherwise known as the gravitational field) $\mathbf{g}(\mathbf{x}) = \mathbf{F}(\mathbf{x})/m_s$ so that

$$\mathbf{g}(\mathbf{x}) = G \int \frac{\mathbf{x}' - \mathbf{x}}{|\mathbf{x}' - \mathbf{x}|^3} \rho(\mathbf{x}') d^3\mathbf{x}'. \tag{2.44}$$

Because gravity is a conservative force, $\mathbf{g}$ may be written as the negative gradient of a scalar field Φ,

$$\mathbf{g}(\mathbf{x}) = -\nabla\Phi \tag{2.45}$$

which is related to the density $\rho(\mathbf{x})$ as

$$\Phi(\mathbf{x}) = -G \int \frac{\rho(\mathbf{x}')}{|\mathbf{x}' - \mathbf{x}|} d^3\mathbf{x}'. \tag{2.46}$$

It is often useful and always insightful to relate $\mathbf{g}$ and Φ to ρ as differential rather than integral equations. It turns out that (2.44) can be written as a total divergence:

$$\nabla \cdot \mathbf{g}(\mathbf{x}) = -4\pi G\rho(\mathbf{x}). \tag{2.47}$$

Substituting $-\nabla\Phi = \mathbf{g}$ leads to the differential form of (2.46),

$$\nabla^2\Phi = 4\pi G\rho(\mathbf{x}). \tag{2.48}$$

This is of course Poisson's equation for the gravitational potential. Note that in all cases the gradient operator ∇ is the gradient with respect to the *unprime* coordinates. For derivations and further discussion of the above equations see reference [9] pages 56-58.

In general $\Phi(\mathbf{x})$ is both more computationally tractable and analytically useful than $\mathbf{g}(\mathbf{x})$, though it contains the same information. Given a known mass distribution, Poisson's equation (2.48) can (in principle at least) be solved by integrating (2.46). From $\Phi(\mathbf{x})$ the orbits of particles can be derived from classical equations of motion.

You may have already spotted the problem with this whole process: in general the exact form of $\rho(\mathbf{x})$ for a galaxy in question is not known, so how can we find $\Phi(\mathbf{x})$ or especially the orbits of particles? There isn't really a solution to this problem, only some workarounds. In general analysis proceeds by the identification of potential-density pairs which approximate the total potential of the system. For disk galaxies the natural first choice is an axially symmetric potential-density pair $\Phi_0(r,z)$, $\rho_0(r,z)$. A secondary approximation might consider small, non-axisymmetric perturbations Φ_1, ρ_1 to the axisymmetric approximation so that

$$\Phi(r,\phi,z) = \Phi_0(r,z) + \Phi_1(r,\phi,z). \tag{2.49}$$

Yet a third approximation might consider a case in which the small perturbations ρ_1, Φ_1 *rotate* at a constant angular speed Ω_p so that

$$\Phi(r,\phi,z,t) = \Phi_0(r,z) + \Omega_p t\, \Phi_1(r,\phi,z). \tag{2.50}$$

In this case it is useful to consider the motion of objects in the *non-inertial frame* which rotates with the perturbing potential at an angular rate Ω_p. If large galactic structures such as bars and spiral arms are stable and persistent, the non-axisymmetric perturbations they introduce would lead to orbits which *reinforce* the original perturbation. This is a fascinating and illuminating story, but sadly I will not have time to tell it with my time left at Reed. I refer you to Reference [9] chapter 3.

2.3 Statistical Mechanics

Even with more time and energy, there are limits to the amount of information one can get by studying the specific orbits of stars in potentials of known form. One limit is imposed by the fundamental nonlinearity of the system at hand. If the distribution of stars $\rho(\mathbf{x},t)$ evolves along the trajectories or orbits in the potential $\Phi(\mathbf{x},t)$ then any density distribution you were able to write down would last only an instant. In order to completely solve the system it would be necessary to integrate (2.46) to find $\Phi(\mathbf{x},t)$, calculate the trajectory of every mass element due to Φ, and update $\rho(\mathbf{x},t)$ *at every moment in time*. Evidently the universe does not do math. Another limit is imposed by our ability to make meaningful predictions about galactic phenomena even given perfect solutions to our equations. Any testable prediction *needs* to be statistical in nature, or else we would need to solve a new problem for every galaxy in the sky.

The main statistical object used to study collisionless systems is the *distribution function* f, defined such that $f(\mathbf{x},\mathbf{v},t)d^3\mathbf{x}d^3\mathbf{v}$ is the probability that a randomly chosen star has phase space coordinates in the volume $d^3\mathbf{x}d^3\mathbf{v}$ located at $(\mathbf{x},\mathbf{v})$. For systems with identical stars (as we've assumed ours is), this probability is the same for any randomly chosen star. Naturally f must be normalized so that

$$\int f(\mathbf{x},\mathbf{v},t)d^3\mathbf{x}d^3\mathbf{v} = 1 \tag{2.51}$$

where the integral is over all of phase space.

The distribution function is a useful mathematical construct because it is simple to write down, yet nearly every observable quantity of interest can be derived from it. Integrating f over all velocities reduces the distribution to a position-space probability distribution

$$\nu(\mathbf{x}) = \int f(\mathbf{x}, \mathbf{v}) d^3\mathbf{v} \tag{2.52}$$

which gives the probability of finding a randomly selected star at location $\mathbf{x}$ regardless of its velocity. Multiplying ν by the total number of stars N gives the real-space number density,

$$n(\mathbf{x}) = N\nu(\mathbf{x}), \tag{2.53}$$

while multiplying by the total mass M gives the mass density,

$$\rho(\mathbf{x}) = M\nu(\mathbf{x}). \tag{2.54}$$

Observationally, one might be interested in the luminosity density $j(\mathbf{x}) = L\nu(\mathbf{x})$ where L is the total luminosity.

The probability distribution of velocities at a location $\mathbf{x}$, $P_\mathbf{x}(\mathbf{v})$, is obtained by dividing f by ν,

$$P_\mathbf{x}(\mathbf{v}) = \frac{f(\mathbf{x}, \mathbf{v})}{\nu(\mathbf{x})}. \tag{2.55}$$

From this it is easy to calculate the mean velocity at a location $\mathbf{x}$,

$$\overline{\mathbf{v}}(\mathbf{x}) = \int P_\mathbf{x}(\mathbf{v})\, \mathbf{v}\, d^3\mathbf{v} = \frac{1}{\nu(\mathbf{x})} \int f(\mathbf{x}, \mathbf{v}) \mathbf{v} d^3\mathbf{v}. \tag{2.56}$$

Another very important quantity of interest is the "spread" in velocities around $\overline{\mathbf{v}}(\mathbf{x})$. In its most general form, this spread is characterized by the velocity dispersion tensor

$$\sigma^2_{ij}(\mathbf{x}) = \frac{1}{\nu(\mathbf{x})} \int (v_i - \overline{v_i})(v_j - \overline{v_j}) f(\mathbf{x}, \mathbf{v}) d^3\mathbf{v} \tag{2.57}$$

where v_i and v_j are *components* of velocity. Evaluating each integral returns a more familiar definition of velocity dispersion,

$$\sigma^2_{ij}(\mathbf{x}) = \overline{v_i v_j} - \overline{v_i}\,\overline{v_j}. \tag{2.58}$$

It is clear that σ^2_{ij} is symmetric, therefore there are six independent components. The diagonal components represent the usual statistical dispersion in each direction. The interpretation of the off diagonal elements is more tricky: the main purpose of the tensor definition is mathematical flexibility. If the coordinate system you're using is related to the symmetries of the potential the off diagonal elements are zero. In other cases they can assumed to be small.

One important question to answer is whether or not f is invariant of the choice of coordinates. It'd be really nice if it was, and there is no reason to expect that it isn't but it's worth a proof nonetheless. Let $\mathbf{w} = (\mathbf{x}, \mathbf{v})$ be the usual Cartesian coordinates

and velocities and let $\mathcal{V}$ be some arbitrary region of phase space. The probability P of finding a star in $\mathcal{V}$ is

$$P = \int_{\mathcal{V}} f(\mathbf{w}) d^6\mathbf{w}. \tag{2.59}$$

Now let $\mathbf{W}$ be some arbitrary set of phase space coordinates and let $F(\mathbf{W})$ be the distribution function in those coordinates so that

$$P = \int_{\mathcal{V}} F(\mathbf{W}) d^6\mathbf{W}. \tag{2.60}$$

If $\mathcal{V}$ is small enough, both f and F will be essentially constant across it so

$$P = f(\mathbf{w}) \int_{\mathcal{V}} d^6\mathbf{w} = F(\mathbf{W}) \int_{\mathcal{V}} d^6\mathbf{W}. \tag{2.61}$$

If the coordinate system in question is canonical then

$$\int_{\mathcal{V}} d^6\mathbf{w} = \int_{\mathcal{V}} d^6\mathbf{W} \tag{2.62}$$

and consequently $F(\mathbf{W}) = f(\mathbf{w})$. This invariance allows us to treat $\mathbf{w} = (\mathbf{q}, \mathbf{p})$ as an arbitrary set of canonical coordinates, and suggests that Hamiltonian formalism might be an appropriate language for the coming analysis.

The probability of randomly selecting a star at a given location in phase space will evolve in time. Therefore if the distribution function is to be of any use it will be necessary to determine a differential equation governing its evolution. An important constraint on this evolution is that probability must be conserved, since the system in question contains real particles which don't jump around.

Consider an arbitrary volume V with a boundary surface S. The probability of finding a star in V at time t is given by

$$P(t) = \int_V f(\mathbf{w}, t) d^6\mathbf{w}, \tag{2.63}$$

so naturally

$$\frac{dP}{dt} = \int_V \frac{\partial f}{\partial t} d^6\mathbf{w}. \tag{2.64}$$

If $P(t)$ is changing in time then the probability must have either left or entered the volume through the surface S. Thus dP/dt is also related to the "flow" of probability,

$$\frac{dP}{dt} = -\oint_S f\dot{\mathbf{w}} \cdot d^5\mathbf{S}. \tag{2.65}$$

where $d^5\mathbf{S}$ is a unit vector normal to S. Equating (2.64) and (2.65) gives the condition

$$\int_V \frac{\partial f}{\partial t} d^6\mathbf{w} + \oint_S f\dot{\mathbf{w}} \cdot d^5\mathbf{S} = 0. \tag{2.66}$$

The divergence theorem may be applied to the second integral above so that

$$\int_V \left[\frac{\partial f}{\partial t} + \nabla \cdot (f\dot{\mathbf{w}})\right] d^6\mathbf{w} = 0 \tag{2.67}$$

Since the volume V is arbitrary the integrand itself must equal zero. Hence,

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial \mathbf{w}} \cdot (f\dot{\mathbf{w}}) = 0. \tag{2.68}$$

This equation is known as the continuity equation. Conceptually, it guarantees that the amount of change in f at any point in space $\partial f/\partial t$ is equal and opposite to the amount that the flow $f\dot{\mathbf{w}}$ diverges from that point. In other words, probability cannot be spontaneously created or destroyed, only moved from one place to another. That's a good physical constraint if I ever heard one.

In order to squeeze more insight out of (2.68), let's use Hamilton's equations,

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}} \qquad , \qquad \dot{\mathbf{p}} = -\frac{\partial H}{\partial q}, \tag{2.69}$$

to eliminate $\dot{\mathbf{w}} = (\dot{\mathbf{q}}, \dot{\mathbf{p}})$ from the right-hand side of (2.68):

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}} \cdot (f\dot{\mathbf{w}}) &= \frac{\partial}{\partial \mathbf{q}} \cdot (f\dot{\mathbf{q}}) + \frac{\partial}{\partial \mathbf{p}} \cdot (f\dot{\mathbf{p}}) \\
&= \frac{\partial}{\partial \mathbf{q}} \cdot \left(f\frac{\partial H}{\partial \mathbf{p}}\right) - \frac{\partial}{\partial \mathbf{p}} \cdot \left(f\frac{\partial H}{\partial \mathbf{q}}\right) \\
&= \frac{\partial f}{\partial \mathbf{q}} \cdot \frac{\partial H}{\partial \mathbf{p}} + f\frac{\partial^2 H}{\partial \mathbf{q}\partial \mathbf{p}} - \frac{\partial f}{\partial \mathbf{p}} \cdot \frac{\partial H}{\partial \mathbf{q}} - f\frac{\partial^2 H}{\partial \mathbf{p}\partial \mathbf{q}}.
\end{aligned} \tag{2.70}$$

The second derivative terms cancel by the equality of mixed partial derivatives so what's left is

$$\frac{\partial}{\partial \mathbf{w}} \cdot (f\dot{\mathbf{w}}) = \frac{\partial f}{\partial \mathbf{q}} \cdot \frac{\partial H}{\partial \mathbf{p}} - \frac{\partial f}{\partial \mathbf{p}} \cdot \frac{\partial H}{\partial \mathbf{q}} \tag{2.71}$$

$$= \dot{\mathbf{q}} \cdot \frac{\partial f}{\partial \mathbf{q}} + \dot{\mathbf{p}} \cdot \frac{\partial f}{\partial \mathbf{p}}. \tag{2.72}$$

Substituting (2.72) into (2.68) gives the *collisionless Boltzmann equation*,

$$\frac{\partial f}{\partial t} + \dot{\mathbf{q}} \cdot \frac{\partial f}{\partial \mathbf{q}} + \dot{\mathbf{p}} \cdot \frac{\partial f}{\partial \mathbf{p}} = 0. \tag{2.73}$$

The collisionless Boltzmann equation is generally thought of as the governing equation, or "equation of motion" if you will, of the distribution function f. If the form of f is known at some time t_0 then it will give the solutions at all later times.

Equation (2.73) may be rewritten in several alternative forms, each which is useful in different contexts. For one, regrouping $\mathbf{p}$ and $\mathbf{q}$ into $\mathbf{w} = (\mathbf{q}, \mathbf{p})$ gets rid of a few symbols:

$$\frac{\partial f}{\partial t} + \dot{\mathbf{w}} \cdot \frac{\partial f}{\partial \mathbf{w}} = 0. \tag{2.74}$$

The right hand side of (2.71) is simply the Poisson-Bracket $[f, H]$, so another form is

$$0 = \frac{\partial f}{\partial t} + [f, H]. \tag{2.75}$$

The partial derivative $\partial f/\partial t$ represents the time evolution of the distribution function *at a fixed point in phase space*, but what what if we are interested in the how f changes *along the orbit of a single mass*? To capture this information it is necessary to define the *convective derivative* df/dt which represents the local change in the probability f as seen by an observer who travels *with a star*. The relation of the new derivative to $\partial f/\partial t$ is easy to derive. The change in f along the path of a star is simply the change at one point in space $(\partial f/\partial t)dt$ plus the difference in f between two points separated by an infinitesimal distance $d\mathbf{w} = \dot{\mathbf{w}}dt$ along the motion of a mass. The infinitesimal change in f between two points in space is simply the gradient $\partial f/\partial \mathbf{w}$ so

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + \dot{\mathbf{w}} \cdot \frac{\partial f}{\partial \mathbf{w}}. \tag{2.76}$$

Of course, all that really happened here is the chain rule, but to me at least the interpretation isn't obvious. The total derivative is defined along particle trajectories because the coordinates themselves move along with particles according to Hamilton's equations.

Wait a second, (2.76) looks an awful lot like (2.74) ... oh wait they are the same! This means that the collisionless Boltzman Equation is simply

$$\frac{df}{dt} = 0. \tag{2.77}$$

Things are looking pretty sharp. What this means is that f is actually a conserved quantity of the Hamiltonian, the same way angular momentum or energy is. Less abstractly, it means that f is constant along the paths traced out by particles in phase space. Here's an interesting consequence: if a star moves along its orbit into a region with a higher spatial density of stars, the corresponding velocity-density must decrease in order to keep f constant. In other words, more dense regions of a galaxy must have greater velocity dispersion.

Inadvertently we've just proved a really general result about functions on phase space. Comparison of (2.74), (2.75) and (2.76) necessitates that

$$\frac{df}{dt} = \frac{\partial f}{\partial t} + [f, H]. \tag{2.78}$$

Notice that the only thing we assumed about f to derive the collisionless Boltzmann equation was that it is a function on phase space that obeys the continuity equation (2.68). This means that in fact everything we've done could be applied to a wide range of objects, not just probability distributions. The result (2.78) can be derived in general, and holds for any function on phase space. The continuity equation (2.68) simply enforced the condition that $df/dt = 0$.

Before we're done it's useful to see where gravity comes into the picture. Suppose we are working in the usual Cartesian coordinates $\mathbf{w} = (\mathbf{x}, \mathbf{v})$ with a Hamiltonian (per unit mass) $H = \frac{1}{2}v^2 + \Phi(\mathbf{x}, t)$. Then Hamilton's equations are

$$\dot{\mathbf{x}} = \frac{\partial H}{\partial \mathbf{v}} = \mathbf{v} \quad , \quad \dot{\mathbf{v}} = -\frac{\partial H}{\partial \mathbf{x}} = -\frac{\partial \Phi}{\partial \mathbf{x}} \tag{2.79}$$

so (2.73) becomes

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} - \frac{\partial \Phi}{\partial \mathbf{x}} \cdot \frac{\partial f}{\partial \mathbf{v}} = 0. \tag{2.80}$$

Hence the evolution of the distribution function depends on the potential in an essential way. This is good news since if it didn't the statistical formulation wouldn't have much to say about galaxies!

2.4 The Jeans Equations

One might ask, "to what extent does the evolution of the distribution function f mirror the evolution of an ideal fluid?" It seems like a reasonable comparison since standard fluid dynamics is similarly concerned with the evolution of a continuous medium of mass as a statistical approximation of the random motion of an enormous number of particles. The state of an ideal fluid (one with no viscosity) at a given point in space is given in terms of the density $\rho(\mathbf{x}, t)$, pressure $p(\mathbf{x}, t)$, and a velocity field $\mathbf{v}(\mathbf{x}, t)$ which describes the total velocity of a fluid element located at $\mathbf{x}$.

An analogous argument to the one which derived the equation of continuity for the distribution function (2.68) leads to a continuity condition for the density ρ,

$$\frac{\partial \rho}{\partial t} + \frac{\partial}{\partial \mathbf{x}} \cdot (\rho \dot{\mathbf{x}}) = 0. \tag{2.81}$$

This equation is considered to be the first fundamental equation of fluid dynamics. Another important equation can be derived pretty quickly by considering the forces acting on a single fluid element. One force component might come from some gravitational potential Φ so that the force exerted on a fluid element is $-\rho\nabla\Phi$. Another might come from the pressure which the fluid exerts on itself. Consider a volume V in space with boundary S. The total force on that volume is the negative of the pressure exerted on the surface S,

$$\mathbf{F}_p = -\oint_S p \, d^2\mathbf{S}. \tag{2.82}$$

Using the divergence theorem this can be turned into a volume integral

$$\mathbf{F}_p = -\int_V \nabla p \, d^3\mathbf{x}. \tag{2.83}$$

Since the volume V can be made arbitrarily small we can say that a force $-\nabla p$ is exerted on the infinitesimal fluid element. The equation of motion, known as Euler's

Equation, follows from Newton's second law. The total force on a fluid element is equal to the mass ρ times the acceleration of that element $d\mathbf{v}/dt$ so

$$\rho\frac{d\mathbf{v}}{dt} = -\nabla p - \rho\nabla\Phi. \tag{2.84}$$

The derivative $d\mathbf{v}/dt$ is a convective derivative similar to the one from the previous section, so it describes the change in velocity of a single fluid element as it moves about in space. In position space the convective derivative expands as

$$\frac{d\mathbf{v}}{dt} = \frac{\partial\mathbf{v}}{\partial t} + \left(\dot{\mathbf{x}}\cdot\frac{\partial}{\partial\mathbf{x}}\right)\mathbf{v}, \tag{2.85}$$

so another form of (2.84) is

$$\rho\frac{\partial\mathbf{v}}{\partial t} + \rho(\mathbf{v}\cdot\nabla)\mathbf{v} = -\nabla p - \rho\nabla\Phi. \tag{2.86}$$

In order to compare the evolution of f as described by the collisionless Boltzman equation to the evolution of a fluid as described by (2.81) and (2.86) it will be necessary to restrict the description of f from $6D$ phase space to $3D$ position space. Consider the collisionless Boltzmann equation in Cartesian coordinates (2.80), rewritten in component form,

$$\frac{\partial f}{\partial t} + \sum_{i=1}^{3}\left(v_i\frac{\partial f}{\partial x_i} - \frac{\partial\Phi}{\partial x_i}\frac{\partial f}{\partial v_i}\right) = 0. \tag{2.87}$$

This equation may be restricted to position space by integrating over all possible velocities

$$\int\frac{\partial f}{\partial t}d^3\mathbf{v} + \sum_{i=1}^{3}\int v_i\frac{\partial f}{\partial x_i}d^3\mathbf{v} - \sum_{i=1}^{3}\frac{\partial\Phi}{\partial x_i}\int\frac{\partial f}{\partial v_i}d^3\mathbf{v} = 0. \tag{2.88}$$

The first term may be simplified by noting that the time derivative may be taken outside of the integral since the region of integration is independent of time. Recalling the definition (2.52) of the position-probability density $\nu(\mathbf{x})$,

$$\int\frac{\partial f}{\partial t}d^3\mathbf{v} = \frac{\partial}{\partial t}\int f d^3\mathbf{v} = \frac{\partial\nu}{\partial t}. \tag{2.89}$$

Similarly, the second term may be simplified by noting that the derivative with respect to x_i may be taken outside of the integral because v_i and x_i are independent. Recalling the definition of the mean velocity (2.56),

$$\int v_i\frac{\partial f}{\partial x_i}d^3\mathbf{v} = \int\frac{\partial\left(v_i f\right)}{\partial x_i}d^3\mathbf{v} = \frac{\partial}{\partial x_i}\int v_i f d^3\mathbf{v} = \frac{\partial\left(\nu\overline{v_i}\right)}{\partial x_i}. \tag{2.90}$$

The last term in (2.88) may be simplified by application of the divergence theorem. If S is a surface in velocity-space at absolute velocity v then

$$\frac{\partial\Phi}{\partial x_i}\int\frac{\partial f}{\partial v_i}d^3\mathbf{v} = \lim_{v\to\infty}\frac{\partial\Phi}{\partial x_i}\oint_S f\cdot d^2\mathbf{S}. \tag{2.91}$$

Since f is normalizable $f \longrightarrow 0$ as $v \longrightarrow \infty$. Thus

$$\frac{\partial \Phi}{\partial x_i} \int \frac{\partial f}{\partial v_i} d^3\mathbf{v} = 0. \tag{2.92}$$

Putting it all back together gives a probability continuity equation in position space

$$\frac{\partial \nu}{\partial t} + \sum_{i=1}^{3} \frac{\partial \left(\nu \overline{v_i}\right)}{\partial x_i} = 0. \tag{2.93}$$

Since the observable density $\rho = M\nu$ where M is the total mass, (2.93) is equivalent to (2.81) if the average stellar velocity $\overline{\mathbf{v}}$ is associated with the the velocity of a fluid element. This is really good news since there would be a serious problem with this whole model if the mass density of stars wasn't conserved. Equation (2.93) is known as the first Jeans equation since it was first applied to stellar dynamics by the English physicist James Jeans (1877-1946).

Right off the bat it is clear that the Euler equation (2.84) will not apply to a collisionless stellar system since in the absence of collisions there will be no pressure! Unfortunately it isn't as simple as setting the ∇p term equal to zero. Pressure encodes a vast amount of statistical information about the system. Most importantly, it is related to the amount of random motion in a fluid. Though there are no collisions in galaxies, there must still be some amount of random motion that may have an important effect. Luckily, the distribution function encodes this random motion by virtue of its statistical definition. In fact it is possible to derive an analogue of the Euler equation for a collisionless system straight from the collisionless Boltzmann equation.

To do so, begin by multiplying (2.88) through by the velocity component v_j,

$$\int v_j \frac{\partial f}{\partial t} d^3\mathbf{v} + \sum_{i=1}^{3} \int v_i v_j \frac{\partial f}{\partial x_i} d^3\mathbf{v} - \sum_{i=1}^{3} \frac{\partial \Phi}{\partial x_i} \int v_j \frac{\partial f}{\partial v_i} d^3\mathbf{v} = 0. \tag{2.94}$$

The first term simplifies because v_j is independent of t,

$$\int v_j \frac{\partial f}{\partial t} d^3\mathbf{v} = \frac{\partial}{\partial t} \int v_j f d^3\mathbf{v} = \frac{\partial \left(\nu \overline{v_j}\right)}{\partial t} = \overline{v_j} \frac{\partial \nu}{\partial t} + \nu \frac{\partial \overline{v_j}}{\partial t}. \tag{2.95}$$

The second term simplifies because $\mathbf{v}$ is independent of $\mathbf{x}$,

$$\int v_i v_j \frac{\partial f}{\partial x_i} d^3\mathbf{v} = \frac{\partial}{\partial x_i} \int v_i v_j f d^3\mathbf{v} = \frac{\partial \left(\nu \overline{v_i v_j}\right)}{\partial x_i}. \tag{2.96}$$

The third term is a little more tricky. Note that

$$\frac{\partial}{\partial v_i} \left(v_j f\right) = \frac{\partial v_j}{\partial v_i} f + v_j \frac{\partial f}{\partial v_i} \tag{2.97}$$

so that

$$\int v_j \frac{\partial f}{\partial v_i} d^3\mathbf{v} = \int \frac{\partial \left(v_j f\right)}{\partial v_i} d^3\mathbf{v} - \int \frac{\partial v_j}{\partial v_i} f d^3\mathbf{v}. \tag{2.98}$$

If S is a surface in volume-space at absolute velocity v then

$$\int \frac{\partial\left(v_j f\right)}{\partial v_i} d^3\mathbf{v} = \lim_{v\to\infty} \oint_S v_j f \; d^2\mathbf{S} = 0 \tag{2.99}$$

since $f \longrightarrow 0$ as $v \longrightarrow \infty$. Thus

$$\int v_j \frac{\partial f}{\partial v_i} d^3\mathbf{v} = -\int \frac{\partial v_j}{\partial v_i} f d^3\mathbf{v} = -\delta_{ij} \int f d^3\mathbf{v} = -\delta_{ij}\nu. \tag{2.100}$$

Using this, the third term of (2.94) can be rewritten,

$$-\sum_{i=1}^{3} \frac{\partial\Phi}{\partial x_i} \int v_j \frac{\partial f}{\partial v_i} d^3\mathbf{v} = -\sum_{i=1}^{3} \frac{\partial\Phi}{\partial x_i} \left(-\delta_{ij}\nu\right) = \nu \frac{\partial\Phi}{\partial x_j}. \tag{2.101}$$

In the end all this mess puts (2.94) into the form

$$\overline{v_j} \frac{\partial\nu}{\partial t} + \nu \frac{\partial \overline{v_j}}{\partial t} + \sum_{i=1}^{3} \frac{\partial\left(\nu \overline{v_i v_j}\right)}{\partial x_i} + \nu \frac{\partial\Phi}{\partial x_j} = 0. \tag{2.102}$$

This is sometimes called the second Jeans equation, though with a little more work it can be written in a much more familiar form. Subtracting $\overline{v_j}$ times the continuity equation (2.93) from (2.102) leaves

$$\nu \frac{\partial \overline{v_j}}{\partial t} + \sum_{i=1}^{3} \frac{\partial\left(\nu \overline{v_i v_j}\right)}{\partial x_i} - \overline{v_j} \sum_{i=1}^{3} \frac{\partial\left(\nu \overline{v_i}\right)}{\partial x_i} = -\nu \frac{\partial\Phi}{\partial x_j}. \tag{2.103}$$

The average velocity $\overline{v_i v_j}$ showed up before in this chapter, way back at the definition of the velocity-dispersion tensor σ_{ij}^2 (2.57). Just for fun, let's try to use the definition of σ_{ij}^2 to simplify (2.103). Since $\overline{v_i v_j}$ appears in a partial derivative with respect to x_i, σ_{ij}^2 must be put in the same form:

$$\begin{aligned} \sum_{i=1}^{3} \frac{\partial\left(\nu\sigma_{ij}^2\right)}{\partial x_i} &= \sum_{i=1}^{3} \frac{\partial\left(\nu \overline{v_i v_j}\right)}{\partial x_i} - \sum_{i=1}^{3} \frac{\partial\left(\nu \overline{v_i}\;\overline{v_j}\right)}{\partial x_i} \\ &= \sum_{i=1}^{3} \frac{\partial\left(\nu \overline{v_i v_j}\right)}{\partial x_i} - \nu \sum_{i=1}^{3} \overline{v_i} \frac{\partial \overline{v_j}}{\partial x_i} - \overline{v_j} \sum_{i=1}^{3} \frac{\partial\left(\nu \overline{v_i}\right)}{\partial x_i}. \end{aligned} \tag{2.104}$$

Subtracting (2.104) from (2.103) results in the third Jeans equation,

$$\nu \frac{\partial \overline{v_j}}{\partial t} + \nu \sum_{i=1}^{3} \overline{v_i} \frac{\partial \overline{v_j}}{\partial x_i} = -\nu \frac{\partial\Phi}{\partial x_j} - \sum_{i=1}^{3} \frac{\partial\left(\nu\sigma_{ij}^2\right)}{\partial x_j}. \tag{2.105}$$

It is provocative to rewrite this in bold vector notation. Let $\boldsymbol{\sigma}^2$ represent the velocity dispersion tensor σ_{ij}^2, then (2.105) looks like

$$\nu \frac{\partial \overline{\mathbf{v}}}{\partial t} + \nu\left(\overline{\mathbf{v}} \cdot \nabla\right) \overline{\mathbf{v}} = -\nu \nabla\Phi - \nabla \cdot \left(\nu\, \boldsymbol{\sigma}^2\right). \tag{2.106}$$

It is clear upon comparison to (2.86) that this is a very direct analogue to Euler's equation, with mass density replaced by probability density and fluid velocity replaced by mean stellar velocity. The pressure gradient term in Euler's equation is replaced by divergence of the velocity dispersion tensor. The physics here is actually quite different since pressure is a scalar quantity while velocity dispersion is tensorial. The comparison provides some intuitive insight, however. A standard gas under the influence of its own gravity will tend to collapse in on itself. As it condenses into a denser region, collisions become more frequent so the pressure increases. The resulting pressure gradient will act *against* gravity to *support* the cloud of gas.

A self-gravitating collisionless fluid will also "want" to collapse into smaller volume. As mentioned above, higher density regions must have a higher velocity dispersion since f is constant along orbits, so as the density of a region increases so does the velocity dispersion. As the velocity dispersion increases, random velocity will start to "dissipate" any small perturbations of higher density before they can collapse any further. In other words, though the mechanism is much different, random velocity in a collisionless fluid has a similar "supportive" effect to that of pressure in an ideal fluid.

This interpretation suggests that if some structure (say, a spiral galaxy) is to be stable, velocity dispersion must play some very important role. It is possible to analyze the stability of stellar structures using the collisionless Boltzman equation, though there isn't time to do so here.[6] First, one would seek a steady state for which the potential Φ is constant. To model disk galaxies one would seek steady state solutions which are axisymmetric, flat, and uniformly rotating. Once a solution is known, the stability may be tested by considering the response of the the system to a small perturbation around the steady state.

In 1964 the mathematician Alar Toomre [13] showed that the local stability of a uniformly rotating disk may be parametrized in terms of the quantity

$$Q = \frac{\sigma_r \kappa}{3.36\, G\, \Sigma(r)} \tag{2.107}$$

where σ_r is the velocity dispersion in the radial direction, $\Sigma(r)$ is the surface-density of the disk, G is the gravitational constant, and κ is the epicyclic frequency of oscillations around a perfectly circular orbit at a radius r given by

$$\kappa^2(r) = r\frac{d\Omega^2}{dr} + 4\Omega^2 \tag{2.108}$$

where Ω is the angular frequency circular orbits. Q can be thought of as a temperature scale for the disk since it is directly correlated to the velocity dispersion σ_r. A "hot" disk has a large velocity dispersion and a high Q, a "cool" disk has a small velocity dispersion and a low Q, while a "cold" disk has zero velocity dispersion and $Q = 0$. The latter case corresponds to a disk in which every mass rotates about the center on a perfectly circular orbit.

Toomre showed that a disk is violently unstable if $Q \leq 1$. This makes a lot of sense given our interpretation of the third Jeans equation (2.105): without velocity

[6] See Reference [9] Chapter 5.

dispersion there is no support against the runaway collapse of higher density fluctuations. On the other hand, if $Q > 2$ small non-axisymmetric perturbations will not grow since the large velocity dispersion will sweep any fluctuations away.[7] If there are to be persistent non-axisymmetric structures such as spiral arms or bars, the sweet spot is

$$1 < Q < 2. \tag{2.109}$$

In this range, there is enough velocity dispersion to support against catastrophic collapse but not enough to sweep away any interesting perturbations.

Before this section is over it is worth noting that, though I presented Jeans equations (2.93) and (2.105) as an analogy to fluid dynamics, in practice they aren't used as fundamental equations of collisionless systems the way that the continuity equation (2.81) and the Euler equation (2.84) are for an ideal fluid. The problem is that the Jeans equations are not complete. Supposing we know the potential $\Phi(\mathbf{x}, t)$ and the density $\nu(\mathbf{x}, t)$, there are nine quantities left to calculate: three components of the average velocity $\overline{\mathbf{v}}$ and six components of the tensor $\mathfrak{o}$. Unfortunately there are only four equations to work with: the scalar continuity equation (2.93) and three components of the third Jeans equation (2.105).

In practice the Jeans equations are most useful for filling in missing observational information. One problem with the distribution function formulation of collisionless mechanics is that information about f can't necessarily be observed in all circumstances. It might be straightforward to observe the position-density $\nu(\mathbf{x})$ but not straightforward to observe the streaming velocity $\overline{\mathbf{v}}$ or the velocity dispersion $\mathfrak{o}$. The Jeans equations allows for information to be partially filled in as needed.

2.5 What does it mean?

Things are getting a bit esoteric, so let's return to the problem at hand. What good does the collisionless Boltzmann equation actually do for us if we're trying to describe the motion of the stars in a galaxy? Recall the Cartesian form of the collisionless Bolzmann equation,

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} - \frac{\partial \Phi}{\partial \mathbf{x}} \cdot \frac{\partial f}{\partial \mathbf{v}} = 0. \tag{2.110}$$

If you think about it, that third term is looking problematic because at every instant in time $\Phi(\mathbf{x}, t)$ depends on $\rho(\mathbf{x}, t)$ by Poisson's equation (2.48). Just to give a sense of the true complexity, let's plug in the definition of $-\partial\Phi/\partial\mathbf{x} = -\nabla\Phi = \mathbf{g}(\mathbf{x})$ from (2.44),

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} + \left(G \int \frac{\mathbf{x}' - \mathbf{x}}{|\mathbf{x}' - \mathbf{x}|^3} \rho(\mathbf{x}') d^3\mathbf{x}' \right) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0. \tag{2.111}$$

To make it even more complete, recall that ρ is related to f by (2.54) so that in full

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{x}} + \left[MG \int \frac{\mathbf{x}' - \mathbf{x}}{|\mathbf{x}' - \mathbf{x}|^3} \left(\int f(\mathbf{x}', \mathbf{v}') d^3\mathbf{v}' \right) d^3\mathbf{x}' \right] \cdot \frac{\partial f}{\partial \mathbf{v}} = 0. \tag{2.112}$$

[7] The real argument is a lot more rigorous than this, it comes from considering the wave mechanics of the disk. I'm just trying to give some intuitive notion of why this is the case.

Of course this is a bit ridiculous because no one in their right mind would actually write it like this, or especially to try to solve it in that form. The point I'm trying to make is that, though the statistical formulation of galaxy dynamics is more flexible and mathematically rigorous than a potential density formulation, it does not absolve the essential nonlinearity of the system at hand. The beast we're trying to wrestle with has got a lot up its sleeve no matter which way you spin it.

Fortunately we are blessed in this modern age with the slavery of a machine race, one which is particularly good at basic arithmetic operations. This chapter constituted the construction of a physical model of a galaxy, and the analytical exploration of that model using the available tools of mathematical analysis. The next chapter will constitute the construction of another model, this time starting with the governing equations of collisionless dynamics and hopefully ending with an algorithm which can be implemented on a modern computer.

Chapter 3

N-Body Simulation

As derived in Section 2.3, the equations governing the evolution of a very large stellar system are the collisionless Boltzmann equation,

$$\frac{\partial f}{\partial t} + \dot{\mathbf{q}} \cdot \frac{\partial f}{\partial \mathbf{q}} + \dot{\mathbf{p}} \cdot \frac{\partial f}{\partial \mathbf{p}} = 0,$$

and Poisson's equation,

$$\nabla^2 \Phi = 4\pi G \rho.$$

These equations are coupled and extremely nonlinear (remember how ugly the full form was from the previous section?) so they don't yield analytic solutions except in specific cases with a high degree of symmetry. Luckily it is 2013 and we have the machine.

But hold your horses, computers aren't technological genies waiting to grant solutions to your favorite three coupled nonlinear equations. In fact they only really know how to do one thing: manipulate finite bits of information. They don't know how to integrate or take derivatives, they can't deal with things that are continuous or infinite, and most importantly they can't tell you what it all means. For being so good at arithmetic, they're pretty stupid really. In order for them to be of any use it will be necessary to dumb down the governing equations into a form they can handle in their narrow little minds.

3.1 N-Body Simulation and its Justification

Since computers can only deal with discrete entities, the first step must be to come up with some way of discretizing the distribution function f. Recall that one version of the collisionless Boltzman equation was

$$\frac{df}{dt} = 0, \tag{3.1}$$

meaning that the distribution function is constant along single particle trajectories of the Hamiltonian $H = \frac{1}{2}v^2 + \Phi(\mathbf{x}, t)$. Here's an idea for discretization: since orbits are paths of constant f, a large but finite sampling of single particle orbits should

give some characteristic sense of how the distribution evolves as a whole. The only catch is this: at every moment in time, the motion of individual particles depends on the potential $\Phi(\mathbf{x}, t)$ which is a function of the mass distribution $\rho(\mathbf{x}, t)$. If you're sampling discrete orbits, you don't necessarily know what ρ is at any time after $t = 0$. If this method is going to make any sense there must be some way in which Φ can be determined in relation to the discrete orbits themselves.

First things first, we need to determine Φ in terms of the distribution function itself. A relation can be determined easily by substituting the statistical definition of mass density (2.54) into the integral definition of Φ from (2.46):

$$\begin{aligned}\Phi(\mathbf{x}, t) &= -G \int \frac{\rho(\mathbf{x}', t)}{|\mathbf{x}' - \mathbf{x}|} d^3\mathbf{x}' \\ &= -MG \int \int \frac{f(\mathbf{x}', \mathbf{v}', t)}{|\mathbf{x}' - \mathbf{x}|} d^3\mathbf{x}' d^3\mathbf{v}' \\ &= -MG \int \frac{f(\mathbf{w}', t)}{|\mathbf{x}' - \mathbf{x}|} d^6\mathbf{w}' \end{aligned} \tag{3.2}$$

where $\mathbf{w}' = (\mathbf{x}', \mathbf{v}')$. Even if the functional form of f is known (it usually isn't), this is going to be a hard integral to evaluate since it lives in six dimensions. There's got to be a way to do this integral quickly and easily.

Let $g(\mathbf{w})$ be a generic function on phase space. If we wanted to know the average of $g(\mathbf{w})$, one way to find it would be to sum g over a very large number of randomly selected points $\mathbf{w}_i$. In this case,

$$\overline{g(\mathbf{w})} = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} g(\mathbf{w}_i). \tag{3.3}$$

If $p(\mathbf{w})$ represents the probability of sampling at a location $\mathbf{w}$ then also

$$\overline{g(\mathbf{w})} = \int g(\mathbf{w}) p(\mathbf{w}) d^6\mathbf{w}. \tag{3.4}$$

Now suppose that we're interested in evaluating the integral

$$I = \int h(\mathbf{w}) d^6\mathbf{w}. \tag{3.5}$$

With a small slight of hand this can rewritten in the form

$$I = \int \frac{h(\mathbf{w})}{p(\mathbf{w})} p(\mathbf{w}) d^6\mathbf{w} \tag{3.6}$$

where now p can be *any* normalized probability distribution you can think of. Then by comparison to (3.3) and (3.4),

$$\int h(\mathbf{w}) d^6\mathbf{w} = \lim_{N \to \infty} \frac{1}{N} \sum_{i=1}^{N} \frac{h(\mathbf{w}_i)}{p(\mathbf{w}_i)} \tag{3.7}$$

where the points $\mathbf{w}_i$ are randomly chosen according to p. This method of integration by random sampling is known as Monte Carlo integration. Applied to (3.2),

$$\Phi(\mathbf{x},t) \simeq -\frac{GM}{N}\sum_{i=1}^{N}\frac{f(\mathbf{w},t)/p(\mathbf{w},t)}{|\mathbf{x}-\mathbf{x}_i|} \tag{3.8}$$

for sufficiently large N. Remarkably, this is simply the gravitational potential of a collection of N point particles,

$$\Phi(\mathbf{x},t) = -G\sum_{i=1}^{N}\frac{m_i(t)}{|\mathbf{x}-\mathbf{x}_i(t)|} \tag{3.9}$$

with masses

$$m_i(t) = \frac{M}{N}\frac{f(\mathbf{w},t)}{p(\mathbf{w},t)}. \tag{3.10}$$

Things are looking pretty promising, since we now have a way to approximate Φ in terms of a set of discrete masses. The original project of tracing out characteristic orbits may proceed.

Suppose we are given the initial distribution function $f(\mathbf{w},0)$. A set of N masses may be positioned randomly according to an arbitrarily chosen probability distribution $p(\mathbf{w},0)$ with masses assigned according to (3.10). Now supposed that they are treated as real particles and advanced some infinitesimal distance along the trajectories determined by the Hamiltonian H. Here's the crux: by the collisionless Boltzmann equation, the distribution function along these trajectories is constant so we will have "solved" for $f(\mathbf{w},t)$ at the location of each mass.

In order to continue advancing the particles, Poisson's equation must be solved again. Equation (3.8) approximates $\Phi(\mathbf{x},t)$ by sampling $f(\mathbf{w},t)$. Unfortunately after $t=0$ the only points at which we know the values of f are the locations of the sampling masses $\mathbf{w}_i$ so we *must* sample at these points. This requirement is problematic because we need to know what probability distribution $p(\mathbf{w},t)$ the masses are assigned by in order to assign the masses $m_i(t)$. After $t=0$ the masses are placed according to the trajectories of the Hamiltonian, not a probability distribution. The easiest way out is to simply note that the masses are *always* sampling the distribution function itself so we can just set $p(\mathbf{w},t) = f(\mathbf{w},t)$. In that case all the particles will have equal mass M/N which is independent of time.

That was a rather convoluted argument, but when the dust settles it becomes clear that something amazing just happened. We started with the collisionless Boltzmann equation and Poisson's equation. We sought to solve for a characteristic sampling of orbits which have a constant value of f. It turned out that Φ can be approximated as the gravitational potential of the system of point particles on those orbits. In other words, the evolution of the distribution function as described by the collisionless Boltzmann equation is approximated by the evolution of a system of N discrete point masses. The equation of motion for such a system is well defined by Newton's law of gravitation,

$$\ddot{\mathbf{x}}_j = -\frac{GM}{N}\sum_{i=1\ i\neq j}^{N}\frac{\mathbf{x}_i-\mathbf{x}_j}{|\mathbf{x}_i-\mathbf{x}_j|^{3/2}} \quad , \quad j=(1\,,2\,,\ldots\,,N). \tag{3.11}$$

The problem of collisionless dynamics has been reduced to the gravitational N-Body problem. This problem is one of the oldest in physics, one which is extremely well studied, and most importantly one that can be tackled by computers.

It is worth thinking about the progression that this thesis has taken. Our simplified model began as a system of 400 billion point masses interacting gravitationally. In chapter 2 we developed a statistical formulation of the mechanics of such a system. In order to solve the governing of equations by computer, we've ended up back at a system of point masses, this time only $N \gg 400$ billion of them. In some sense this is really obvious: *of course* a smaller system of particles has similar statistical properties to a larger system of particles. I think it was worth the careful consideration, for the statistical interpretation of the N-Body simulation is not quite what it seems. Though it is tempting to think of the sampling masses as "real stars," since they have constant mass and travel on the same trajectories that real particles would, they aren't *really*. In fact in the next section we're going to kill any resemblance they had to real particles anyways. The more nuanced interpretation is that an N-Body calculation is solving the collisionless Boltzmann equation by the method of characteristics, and integrating Poisson's equation by Monte Carlo sampling. For this reason this numerical method is generally referred to as an N-Body *simulation*.

3.2 Softening

I glossed over it for the sake of a coherent argument, but there is actually a huge problem with the method of Monte-Carlo integration described above. In fact, I totally lied to you in equation (3.9) for in fact the full integral (3.2) is not well represented by the sum over discrete points. The problem lies in the singularities present at the location of each sampling mass. f is continuous by assumption and therefore Φ must be as well, it should never be infinite. Though the Monte Carlo sampling effectively approximates Φ on a large scale, near a sampling mass it is *wildly* inaccurate.

The problem may also be thought of in terms of the relaxation time (2.42). We established that for $N \sim 400$ billion the relaxation time is extremely large in relation to the age of the system. But what about for N on the order of what might be possible to achieve on a computer? For $N = 20,000$, a very achievable value, $t_{\text{relax}}/t_{\text{cross}} \sim 200$. This is pretty low, on the order of the age of a galaxy, so it is safe to say that the system is not collisionless.

One solution is to simply *remove* the singularity by modifying the normal point-particle potential. The simplest modification is a potential of the form

$$\Phi(r) = -G\frac{m}{\sqrt{r^2 + \epsilon^2}} \tag{3.12}$$

where ϵ is some small "softening length" and r is the distance from the particle's location. In the limit $r \gg \epsilon$ this modified potential is identical to that of a true point particle. At $r = 0$ it takes a finite minimum value $\Phi_{\text{min}} = -Gm/\epsilon$. Naturally ϵ can assigned to whatever value one likes, one that is large enough to solve the problem of

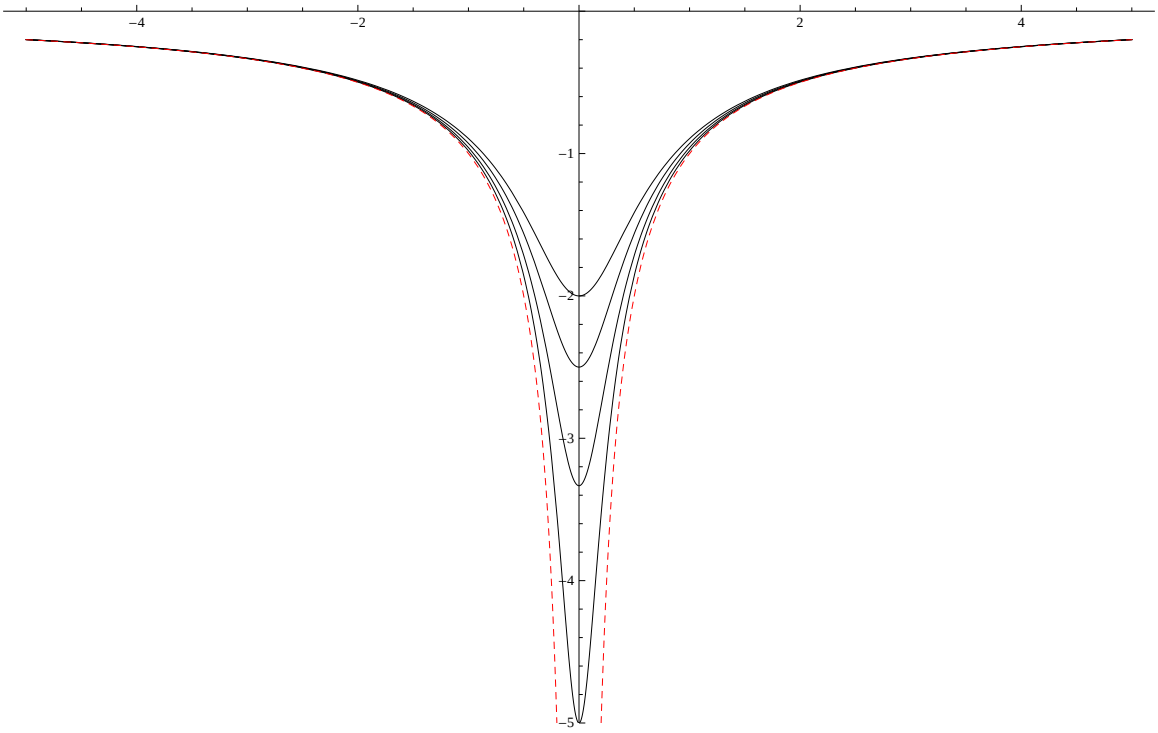

Figure 3.1: The potential $\Phi(r)$ for a Plummer-softened point particle plotted for various values of epsilon. The value of the softening length ϵ increases from bottom to top. The unsoftened case is shown in dotted red, diverging at $r = 0$.

divergent force but small enough to maintain some level of detail in the simulation. The modified equation of motion resulting from a potential of this form is,

$$\ddot{\mathbf{x}}_j = -\frac{GM}{N} \sum_{i=1\, i\neq j}^{N} \frac{\mathbf{x}_i - \mathbf{x}_j}{\left(|\mathbf{x}_i - \mathbf{x}_j|^2 + \epsilon^2\right)^{3/2}} \quad , \quad j = (1\,, 2\,, \ldots\,, N) \tag{3.13}$$

One might wonder what the global effects of a softened potential are. In fact, I spent a long time worrying about this myself. It seemed like *modifying gravity* would change the dynamics of the system we're trying to study in the first place. I thought, "there are some circumstances in which modifying gravity is appropriate, but the Newtonian approximation is very good on the scale of a single galaxy, it shouldn't be modified!"

Such thoughts were naive, and after much existential angst I know better. My mistake was forgetting that the particles in an N-Body simulation aren't *real* particles. They're sampling points which are used to integrate (3.2) by Monte-Carlo integration, as well as the points used to trace out a characteristic sampling of orbits. It doesn't matter if we're "modifying gravity," only that the numerical method is an accurate approximation of the true evolution as described by the collisionless Bolzmann equation and Poisson's equation.

A collection of softened point particles is actually much better suited to evaluate the integrand in (3.2) since the Monte-Carlo sampling masses have some continuous

distribution of their own. To see why, let's solve for the mass distribution $\rho(r)$ which would result in a potential of the form (3.12). The gradient of Φ is easy to calculate,

$$\nabla^2\Phi(r) = \frac{1}{r}\frac{d}{dr}\left(r^2\frac{d\Phi}{dr}\right) = \frac{3Gm\epsilon^2}{(r^2+\epsilon^2)^{5/2}}, \tag{3.14}$$

so by Poisson's equation,

$$\rho(r) = \frac{3m}{4\pi\epsilon^3}\left(1+\frac{r^2}{\epsilon^2}\right)^{-5/2}. \tag{3.15}$$

This kind of spherical distribution is known as a Plummer sphere. When a softening length is introduced, instead of solving (3.12) by Monte Carlo sampling with point particles, we're solving it by Monte-Carlo sampling with Plummer spheres. It's easy to see why this is actually much more accurate. It's worth noting that the Plummer sphere analogy only goes so far. (3.13) describes the motion of point particles in the presence of a potential created by Plummer spheres at the locations of those point particles, *not* the motion of N Plummer spheres interacting with themselves. That system would be substantially more complicated.

There is still some amount of systematic error introduced by softening, for it must effect total dynamics on some level. It is difficult to determine exactly how important this systematic error is. Nevertheless there is literature (References [14], [15], and [16]) which addresses this question to the extent which it can be addressed. In general it is thought that a softening length which is too large will destroy the stability properties of the disk [16]. The softening length ϵ should be large enough to provide appropriate relaxation characteristics but small enough to maintain realistic local potentials. In general it should be smaller if more sampling particles are used. One study [17] found that the optimum value is

$$\epsilon = 0.98N^{-0.26}, \tag{3.16}$$

where N is the number of sampling masse. I will assign ϵ as such in my simulation code.

3.3 Numerical Method

After all the hard work to derive a discrete approximation of a collisionless stellar system, it's now finally time to figure out how to actually put this stuff on a computer. From here on out we'll be working in non-dimensional units in which $G = 1$ and every particle has mass 1 so that $M = N$. The absolutely final (I swear I'm not joking this time) equation of motion that will be put on a computer is

$$\ddot{\mathbf{x}}_j = -\sum_{i=1\, i\neq j}^{N} \frac{\mathbf{x}_i - \mathbf{x}_j}{\left(|\mathbf{x}_i - \mathbf{x}_j|^2 + \epsilon^2\right)^{3/2}} \quad , \quad j = (1\,, 2\,, \ldots\,, N). \tag{3.17}$$

The numerical method for solving the above equation may be broken up into two parts. First, there must be a method of integrating the positions and velocities of

every mass forward in time over some grid of discrete times separated by an amount Δt. In order to do so it will be necessary to know the acceleration of each mass at those discrete times as given by the right hand side of the equation above.

3.3.1 Force Calculation

The easiest way to calculate the force $\mathbf{F}_j$ on every mass $j = (1\,, 2\,, \ldots\,, N)$ is to literally perform the sum on the right-hand side of (3.17). For each of the N masses, the inverse-distance-squared to every other mass must be summed. Thus the number of computations required is $\mathcal{O}(N^2)$. This is really bad, it's going to take a long time to do for even a moderate number of particles.

There are other methods of force calculation which are $\mathcal{O}(N\,\mathrm{log}N)$. One of them, known as a treecode, takes advantage of the fact that the force from groups of particles may be approximated using its multipole moments. Unfortunately this whole thesis thing happens over a pretty limited amount of time and through a series of choices the treecode was put on the back burner. A direct summation approach is limiting in general but totally acceptable given the scope of this thesis. I regret nothing.

3.3.2 Integration

The basic idea for solving a differential equation numerically is to approximate the true solution to the function at discrete points using some simple arithmetic algorithm which can be handled by a computer. In this case the differential equation we're solving is just Newton's second law

$$\ddot{\mathbf{x}}_j = \mathbf{a}_j \tag{3.18}$$

where $\mathbf{a}_j$ replaces $\mathbf{F}_j$ because we're working in units where all masses are set to 1. A vast number of methods exist for solving Newton's second law numerically. Different methods are useful for solving different sorts of systems, depending on how quickly the force is varying with time, or how expensive it is to calculate the force. In my N-Body simulation I will be using the leapfrog, or Verlet, method for solving Newton's second law. There are a number of ways to formulate and conceptualize the method. It will be most useful to formulate it in terms of the phase space coordinates $(\mathbf{x}, \mathbf{v})$, both in keeping with the phase-space-centric theme of this thesis and because the discussion of individualized timesteps in the next section depends on the "Drift-Kick-Drift" formulation.

To keep the notation clean, let's restrict the consideration of Newton's second law to one particle. It should be pretty clear that the result for one particle generalizes to many. The Hamiltonian of a single particle is

$$H = \frac{1}{2}v^2 + \Phi(\mathbf{x}) \tag{3.19}$$

and the Hamilton equations of motion are

$$\dot{\mathbf{x}} = \mathbf{v} \qquad , \qquad \dot{\mathbf{v}} = -\nabla\Phi(\mathbf{x}) = \mathbf{a}(\mathbf{x}). \tag{3.20}$$

These equations are just a reformulation of Newton's second law as two first order ordinary differential equations as opposed to one second-order ordinary differential equations. Now, suppose we replace the true Hamiltonian (3.19) with an approximate Hamiltonian

$$\tilde{H} = \frac{1}{2}v^2 + \Phi(\mathbf{x})C(t) \tag{3.21}$$

where

$$C(t) = \Delta t \sum_{k=-\infty}^{\infty} \delta(t + 1/2 - k\Delta t) \tag{3.22}$$

is a periodic "comb function" of Dirac delta spikes and Δt is some small timestep. It is clear that as $\Delta t \longrightarrow 0$, $\tilde{H} \longrightarrow H$ so the trajectories determined by $\tilde{H}$ should therefore approach those determined by H. The equations of motion for this modified Hamiltonian are

$$\dot{\mathbf{x}} = \mathbf{v} \qquad , \qquad \dot{\mathbf{v}} = -\nabla\Phi\, C(t) = \mathbf{a}(\mathbf{x})\, C(t). \tag{3.23}$$

These equations can now be integrated from $t = 0$ to $t = \Delta t$. Suppose at $t = 0$ the mass is at $(\mathbf{x}_0, \mathbf{v}_0)$. During the interval from $t = 0$ to $t = \Delta t/2 - \epsilon$ where ϵ is very small, the potential is zero so the velocity does not change. The position "drifts" with constant velocity so that at $t = \Delta t/2 - \epsilon$,

$$\mathbf{x}_{1/2} = \mathbf{x}_0 + \frac{1}{2}\mathbf{v}_0\Delta t. \tag{3.24}$$

Over the interval from $t = \Delta t/2 - \epsilon$ to $t = \Delta t/2 + \epsilon$ the position is essentially constant but the velocity suffers a "kick" delivered by the delta spike at $t = \Delta t/2$. Thus the new velocity $\mathbf{v}_1$ is updated as

$$\mathbf{v}_1 = \mathbf{v}_0 + \mathbf{a}(\mathbf{x}_{1/2})\, \Delta t. \tag{3.25}$$

From $t = \Delta t/2 + \epsilon$ to $t = \Delta t$ the position "drifts" again with constant velocity so that

$$\mathbf{x}_1 = \mathbf{x}_{1/2} + \frac{1}{2}\mathbf{v}_1\Delta t. \tag{3.26}$$

Equations (3.24) and (3.26) can be combined for an interesting result,

$$\mathbf{x}_1 = \mathbf{x}_0 + \frac{1}{2}\left(\mathbf{v}_0 + \mathbf{v}_1\right)\Delta t \tag{3.27}$$

In other words, the position is updated at the end of each timestep using the average of original and final velocities, while velocity is updated in the middle of the timestep using the acceleration calculated with the position $\mathbf{x}_{1/2}$. This is why the method is known as the leapfrog method: the updates of position and velocity are offset by $\Delta t/2$.

The leapfrog method will produce a list of positions and velocities at a discrete set of times t_n separated by Δt. If $\mathbf{x}_n = \mathbf{x}(t_n)$ and $\mathbf{v}_n = \mathbf{v}(t_n)$ then in total

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n\Delta t + \frac{1}{2}\mathbf{a}(\mathbf{x}_{n+1/2})\Delta t^2 \tag{3.28a}$$

$$\mathbf{v}_{n+1} = \mathbf{v}_n + \mathbf{a}(\mathbf{x}_{n+1/2})\Delta t. \tag{3.28b}$$

The accuracy of this method may be determined by comparison of the above to the Taylor expansions

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\mathbf{a}(t)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{3.29a}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t + \frac{1}{2}\frac{d\mathbf{a}}{dt}\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{3.29b}$$

Any arbitrary term of order Δt^3 or higher may be pulled out of the $\mathcal{O}(\Delta t^3)$ terms as needed so the acceleration terms may be rewritten in terms of an expansion,

$$\mathbf{x}(t+\Delta t) = \mathbf{x}(t) + \mathbf{v}(t)\Delta t + \frac{1}{2}\left(\mathbf{a}(t) + \frac{1}{2}\frac{d\mathbf{a}}{dt}\Delta t + \cdots\right)\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{3.30a}$$

$$\mathbf{v}(t+\Delta t) = \mathbf{v}(t) + \left(\mathbf{a}(t) + \frac{1}{2}\frac{d\mathbf{a}}{dt}\Delta t + \frac{1}{8}\frac{d^2\mathbf{a}}{dt^2}\Delta t^2 + \cdots\right)\Delta t + \mathcal{O}(\Delta t^3). \tag{3.30b}$$

This expansion is just the Taylor expansion of $\mathbf{a}(t + \Delta t/2) = \mathbf{a}(\mathbf{x}_{n+1/2})$, returning exactly the result from (3.29a) and (3.29b)! Thus the leapfrog method is $\mathcal{O}(\Delta t^2)$ accurate in both position and velocity.

The leapfrog method has a number of advantages. First, it is $\mathcal{O}(\Delta t^2)$ but only requires one force evaluation per timestep. This is a huge advantage in an N-Body simulation where the force is extremely expensive to calculate (see the previous section). Second, it is time reversible in the sense that the procedure for integrating forwards from $(\mathbf{x}_n, \mathbf{v}_n)$ to $(\mathbf{x}_{n+1}, \mathbf{v}_{n+1})$ is identical to the procedure for integrating backwards from $(\mathbf{x}_{n+1}, -\mathbf{v}_{n+1})$ to $(\mathbf{x}_n, -\mathbf{v}_n)$. For this reason it will conserve the energy of a system extremely well because energy is the conserved quantity associated with time translation symmetry.

To use this method to integrate equation (3.17) one would simply perform the calculations (3.24), (3.25), and (3.26) once each timestep for each of the N masses in the system. Only $\mathcal{O}(N)$ calculations are necessary to perform the integration, but for each integration the acceleration must be calculated from the positions $\mathbf{x}_{n+1/2}$ by direct summation which is $\mathcal{O}(N^2)$. The bulk of computational cost will result from the force calculation.

3.3.3 Individual Timesteps

When performing a numerical integration using the leapfrog method or any other, the timestep Δt must be small enough to capture important dynamical effects and minimize the error occurring as a result of the numerical approximation. In an integration of Newton's second law the factor which determines the dynamical timescale of a particle's motion is the magnitude of the acceleration $\mathbf{a}$. If $\mathbf{a}$ is small and relatively constant, the velocity won't vary much in time and to a good approximation, the position advances in a straight line over small timescales. If $\mathbf{a}$ is large or not very constant the velocity might change dramatically in a short timescale, the particle's path might be very far from a straight line even over a short timescale. The way to mitigate the damage is to reduce the timestep Δt until you are satisfied with the accuracy of the integration.

In an N-body simulation this process is going to be problematic. If there are thousands of masses, all interacting with each other in endlessly complicated ways, the dynamical timescale might vary wildly from particle to particle. There might be a particle which is out at the edge of the distribution for which the dynamical timescale is quite long. Another particle in a dense region might be experiencing dramatically strong forces from its thousands of near-neighbors and evolve on a timescale orders of magnitude smaller than the one at the edge. In order to accurately integrate the system you would need to integrate every particle on the *smallest* timestep required for sufficient accuracy of any single particle. If there's one thing we don't want to happen more than it needs to, it's a force calculation. Every time a particle is integrated over a timestep shorter than it needs to be we are *wasting* computation.

It would be ideal for each mass to be integrated on its own individual timestep, so that the computer doesn't waste precious computation time evaluating forces that don't need to be evaluated. Such a method would be computationally problematic for it would mean that every mass must be integrated individually rather than collectively. In order to select the appropriate mass to advance, several $\mathcal{O}(N)$ search operations would need to be performed in addition to the force calculation. The computational cost would be overwhelming.

For such a method to be computationally viable, there must be a finite number of allowed timesteps so that large groups of particles may be advanced simultaneously. Such quantization may be achieved by restricting the allowed timesteps to factors of two of some maximum timestep $\Delta t_{\max}$,

$$\Delta t_k = \frac{\Delta t_{\max}}{2^k} \quad , \quad k = 0,\ 1,\ 2, \ldots \tag{3.31}$$

All the particles with a single timestep value may be advanced simultaneously. Figure 3.2 shows a schematic of this quantized timestep structure. Each arc represents the integration of a single group of particles, with the horizontal axis representing time. The vertical line in the center of each arc represents the time at which the force is calculated and the velocities of that group are updated. In a way one can think of the vertical lines as barriers: in between the vertical lines, the system can be advanced backwards and forwards relatively easily since all the particles are traveling in straight lines. At the vertical lines, the system cannot be advanced without an expensive force calculation. The groups of particles must be integrated in the order in which the velocity updates occur, chronologically.

Algorithmically this is achieved by keeping track of the "prediction time" $\tau_j = t_j + \Delta t_j/2$[1] at which the next velocity update should occur for each mass j, if t_j is the time after its last integration and Δt_j is its individual timestep. Only particles for whose prediction time is equal to the *minimum* prediction time out of the entire set are advanced. When the force is calculated on particle j it is done using the positions $\mathbf{x}_i'$ of *all* masses at the time when the velocity update occurs,

$$\mathbf{x}_i^{1/2} = \mathbf{x}_i + (\tau_{\min} - t_i)\mathbf{v}_i. \tag{3.32}$$

[1]Now that we're talking about the full set of N particles I am using subscripts to denote a mass's index as opposed to the timestep number. So, t_i is the current time of the i^{th} particle.

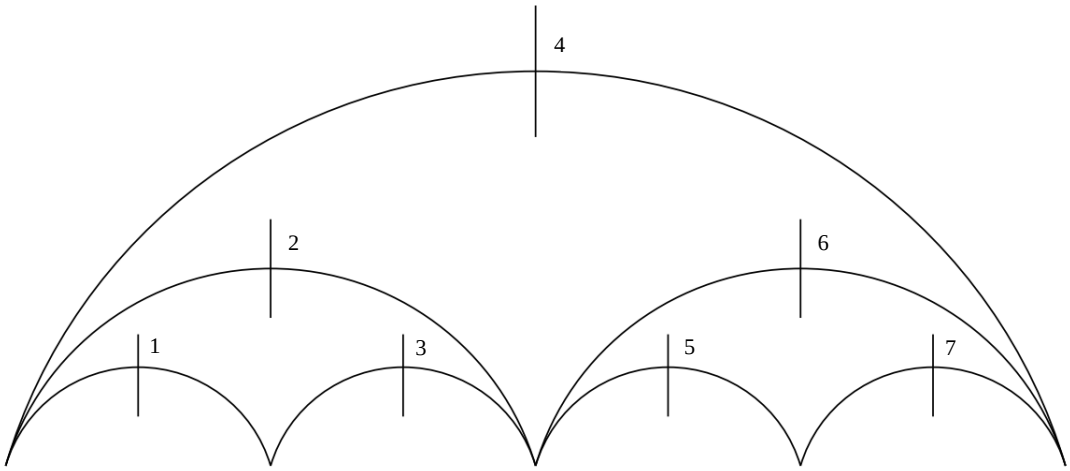

Figure 3.2: Schematic of the quantized individual timestep routine. Arcs represent the advancement of groups of particles while vertical lines represent force calculations. The advancements are numbered in the order that they would be performed. Modified from T. R. Quinn, N. Katz, J. Stadel, and G. Lake, Astrophys.J. (1997), astro-ph/9710043.

Once a particle is integrated, its prediction time is updated and a new minimum time is selected. Thus the integration proceeds through time for all groups of masses, with each group integrated twice as often as the one above it.

This seems like a pretty reasonable and achievable scheme for incorporating individual timesteps into the leapfrog integration scheme. How then, shall we choose an appropriate timestep for each mass? As mentioned before, the dynamical timescale on which a mass evolves might have something to do with the magnitude of the force it is experiencing. There are many ways one could characterize this relationship. One way is by equation (3.28a), written here again:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{v}_n \Delta t + \frac{1}{2}\mathbf{a}(\mathbf{x}_{n+1/2})\Delta t^2. \tag{3.33}$$

In order for the integration to be accurate, the acceleration term must be small compared to the position and velocity terms. Suppose we put a constraint on the magnitude of the trailing term,

$$\left|\frac{1}{2}\mathbf{a}(\mathbf{x}_{n+1/2})\Delta t^2\right| < \delta \tag{3.34}$$

where δ is some small length. This condition puts a constraint on Δt,

$$\Delta t < \sqrt{\frac{2\delta}{|\mathbf{a}|}}. \tag{3.35}$$

The parameter δ limits the amount by which the path of a particle may deviate from a straight line to the influence of the force over one timestep. It is arbitrary but it

Algorithm 3.1 Selecting a new timestep Δt_{new} given the previous timestep Δt_p and an ideal timestep Δt_{ideal}

if $\Delta t_{\text{ideal}} < \Delta t_p$ **then**
  $\Delta t_{\text{new}} = \Delta t_p/2$
else if $\Delta t_{\text{ideal}} > 2\Delta t_p$ AND $\text{Mod}(t/\Delta t_p, 2) = 0$ **then**
  $\Delta t_{\text{new}} = 2\Delta t_p$
else
  $\Delta t_{\text{new}} = \Delta t_p$
end if

should be related to some characteristic scale length of the system you're trying to integrate. A good choice is the softening length ϵ since it *already* defines what is a small length in the simulation. One could just set $\delta = \epsilon$, but there should still be some user control over the accuracy of the simulation so a better choice is to set $2\delta = \alpha_{\text{tol}}\epsilon$ where α_{tol} is a tolerance parameter. Thus the ideal timestep would be

$$\Delta t_{\text{ideal}} = \sqrt{\frac{\alpha_{\text{tol}}\epsilon}{|\mathbf{a}|}}. \tag{3.36}$$

You may have noticed the catch with this method: it requires that we know the magnitude of the acceleration $|\mathbf{a}|$ *before* the integration in order to determine the appropriate timestep. Of course, the acceleration itself depends on the timestep insofar as it relates to the position of the masses at the time the force is calculated. Without using some method that predicts the force before it is known,[2] it must suffice to use the magnitude of the force from the *previous* timestep. This shouldn't be so far off since the force on a single particle is a smoothly varying function of time.

Once the ideal timestep is selected, it must be sorted into the largest quantized step Δt_k such that $\Delta t_k < \Delta t_{\text{ideal}}$. In order for the method to maintain its organizational structure it must be ensured that the timestep groups remain commensurate to powers of two. In other words, that every timestep is subdivided equally by two smaller timesteps. This can be achieved by requiring that a particle's timestep may increase only *every other* integration when the current time of the particle is commensurate with the current time of the particles in a higher group. Algorithm 3.1 describes the method for selecting a new timestep Δt_{new} given the previous one Δt_p and the ideal timestep Δt_{ideal}.

There are a number of other ways to select the ideal timestep based other criterion. For example, limiting the higher order terms in an expansion of the system's total energy suggests a criterion

$$\Delta t = \alpha'_{\text{tol}}\frac{\sigma}{|\mathbf{a}|} \tag{3.37}$$

where σ is the local velocity dispersion. Others use the mass local density ρ as a parameter (see References [19] and [10]). I have chosen to use the first method because it is formulated entirely in terms of known quantities and thus requires no

[2]These actually do exist, they're known as predictor corrector methods. For full discussion see reference [18] Chapter 2.

Algorithm 3.2 Schematic of leapfrog integration including individual timesteps.

Select $\tau_{\min}$

for $j = 1 \rightarrow N$ **do**
  $\mathbf{x}_j^{1/2} = \mathbf{x}_j + (\tau_{\min} - t_j)\mathbf{v}_j$
end for

for $j = 1 \rightarrow N$ **do**
  if $\tau_j = \tau_{\min}$ **then**
    Compute $\mathbf{a}_j(\mathbf{x}^{1/2})$
    Update $\mathbf{x}_j$ and $\mathbf{v}_j$ by leapfrog
    Calculate Δt_{ideal}
    Sort and assign Δt_j according to Algorithm 3.1
    Update t_j, τ_j
  end if
end for

extra computation. In general it yields larger timesteps than other methods. This can be compensated for by selecting a relatively low value of $\Delta t_{\max}$.

The advantages of an individualized timestep integration routine are plain and clear. The force is calculated on a mass only when it needs to be: masses in dense regions can be integrated frequently while those floating at the edge of the distribution may be integrated infrequently. Moreover, the timestep selection allows the algorithm to account for its own accuracy as needed, thus removing the burdensome task of determining an appropriate timestep from the human user.

The disadvantages are significant, however. Individual timesteps violate just about everything we said was good about the leapfrog integrator in the previous section. The method for timestep selection is strongly non-time-reversible (remember, it depends on the force from the previous step). Consequently integrals of motion such as momentum and energy will not be conserved, in general.

There is an extended body of literature on the numerical analysis of variable timestep integrators. It is thought that by selecting the timestep in a time-reversible manner produces integrators with the best conservation properties (see reference [9] page 201). However, there is also reason to believe that for a collisionless N-Body simulation, strict conservation requirements are not as important as they might seem. The systems in question are statistical in nature and therefore the processes which occur are *inherently* non-time reversible so the time reversibility of the integrator might be a trivial matter.

3.3.4 Code and Pseudocode

Algorithm 3.2 presents the full variable timestep leapfrog integration scheme over one iteration. I based the algorithm loosely on the one used in the freely available GADGET cosmological simulation code (see references [20], [19]) put out by the Max

Plank Institute for Astrophysics, though without a lot of its fancy extra features. The code, written in C++, is presented in Appendix A. It is implemented in parallel using MPI (Message Passing Interface).

3.4 Sperical Shell Collapse

Table 3.1: Spherical shell collapse results. The reported time is the average of the time at which each mass passed $r = r_0/2$, with uncertainties equal to one half of the timestep. Each is compared to the true time $t = 0.5748$.

Integrator	Δt	Time	Percent Error
Variable Timestep	$0.017 \to 0.008$	0.577 ± 0.004	0.427 %
Leapfrog	0.017	0.582 ± 0.008	0.867 %
Leapfrog	0.008	0.577 ± 0.004	0.427 %

It would constitute an entire thesis of its own to fully vet this algorithm, but it is worthwhile to perform at least one accuracy test to make sure this whole thing isn't completely crazy. A uniformly distributed spherical shell of mass M and radius r which is subject to its own gravity will obey the differential equation

$$\ddot{r} = -G\frac{M}{2r^2}. \tag{3.38}$$

Suppose we sample a shell with N point masses. Then the shell will have total mass N so in units with $G = 1$

$$\ddot{r} = -\frac{N}{2r}. \tag{3.39}$$

The time required for the shell to collapse from an initial radius r_0 (if initially at rest) to $r_0/2$ can be found easily from (3.39),

$$t = \frac{r_0^{3/2}}{4\sqrt{N}}\left(2 + \pi\right). \tag{3.40}$$

The accuracy of an N-Body simulation code may be tested by comparing this exact result to one given by the N-Body modeling of the same system.

The first thing I wanted to test was the deviation of the variable timestep integrator from a normal leapfrog integrator. The regular leapfrog integrator is extremely well tested so it would be a great confidence boost if the results were consistent. I sampled a spherical shell of radius $r_0 = 10$ with 5000 masses. For these parameters the true collapse time should be $t = 0.5748$ in nondimensionalized units. The softening length was assigned as usual to $\epsilon = 0.98N^{-0.26}$. The maximum timestep was set high, to $\Delta t_{\max} = 34.52$,[3] and a tolerance parameter $\alpha_{\text{tol}} = 0.1$.

[3]Chosen by the "hit the keypad randomly and see what happens" method.

Table 3.2: Spherical shell collapse results. The reported time is the average of the time at which each mass passed $r = r_0/2$, with uncertainties equal to one half of the timestep. Each is compared to the true time $t = 0.5748$.

α_{tol}	Time	Percent Error
50	0.68 ± 0.09	18.9 %
10	0.66 ± 0.02	14.7 %
5	0.56 ± 0.02	-2.3 %
1	0.57 ± 0.01	-1.9 %
0.5	0.583 ± 0.006	0.87 %
0.1	0.583 ± 0.003	1.40 %
0.05	0.581 ± 0.002	1.13 %

I ran six tests total. First, a test with the full variable timestep integrator. In collapsing from r_0 to $r_0/2$ the timestep quantization for each mass should change quantization once, so for the second and third tests I ran a normal leapfrog integrator with the highest and lowest timesteps used by the variable integrator. The reported time is the average of the times at which each mass passed $r = r_0/2$, with uncertainties equal to one half of the timestep.

The results of these tests are presented in Table 3.1. Remarkably, the variable timestep integrator and the regular leapfrog integrator integrated over the smaller timestep report exactly the same result! I find this extremely encouraging. Not only does the variable timestep integrator perform exactly as well as the normal leapfrog but it did so with less force calculations, by making itself as accurate as it needs to be.

All of the reported times are greater than the true time by some amount so there must be some systematic error. One source of this error might be the softening length since it decreases the radial force due to nearby particles.[4] I don't find this error too concerning, since a razor-thin shell of mass is more artificial than anything that would occur in a real system. The global effects of softening would need to be tested in a different context.

I ran another series of tests using *only* the variable integrator with decreasing values of the tolerance parameter α_{tol}, with all other parameters the same as above. The accuracy increases dramatically from $\alpha_{\text{tol}} = 50$ to $\alpha_{\text{tol}} = 1$ but below this the returns are much smaller. Based on this test, I estimate that $\alpha_{\text{tol}} = 0.1$ is sufficient to provide accurate results and will assign it as such for all simulations. A more thorough test of timestep accuracy might involve energy conservation in an arbitrary system.

[4]Incidentally, if you try to run this test without a softening length you will find that the average time for a mass to reach $r_0/2$ is ∞! This is because some of the masses get flung to infinity during close interactions so they *never* cross the midpoint.

Chapter 4

What Happened

So far this thesis has constituted a succession of refinements. In the first chapter, the vast complexity of astronomical phenomena was refined into a series of basic observational assumptions. In the next chapter these assumptions were refined into a simple statistical model, including a set of governing equations. Chapter 3 constituted another refinement, this time from the governing equations of a collisionless system to their approximation by N-Body simulation, including an an algorithm for its implementation on a computer.

I am happy to inform you that there will be no more refinements. Once the N-Body code is implemented on a computer, it can handle any collisionless system we throw at it. Thus we return to the original questions: what conditions are necessary for the formation of spiral arms and how do they relate to the existence of dark matter? How does the presence of such structures relate to the presence of a dark matter halo? In this final chapter I will attempt to address this question by simulating the evolution of two systems: a typical Sc class disk galaxy *with* and *without* an encompassing dark matter halo.

4.1 Building a Galaxy

The construction of appropriate initial conditions for an N-Body simulation is not very straightforward. As outlined in Section 3.1, the N-Body simulation method is predicated on the assumption that one knows the initial distribution function $f(\mathbf{x}, \mathbf{v}, 0)$ in order to sample it with discrete masses. Unfortunately for most realistic systems it is not possible to write down the distribution function in a manner that can be sampled numerically since the non-spatial components of a model are difficult to track down. Luckily it is possible to characterize the distribution in terms of other observational quantities (see Section 2.4), namely the density distribution $\nu(\mathbf{x}, 0)$, the average velocity $\overline{\mathbf{v}}(\mathbf{x})$, and the velocity dispersion tensor $\mathfrak{v}(\mathbf{x})$ which can be assumed to be diagonal to a first approximation.

Since we are interested in studying the *formation* of non-axisymmetric structures such as spiral arms, our toy galaxy should be initially axisymmetric. The spatial distribution of a very thin disk can be divided into two parts: a top-down surface

density distribution $\Sigma(r)$ and a vertical distribution $h(z)$.[1]

If a disk galaxy is at or near an equilibrium configuration, each star should be on a nearly circular orbit. Though not always true observationally, it is generally assumed that the masses rotate about the center in a uniform manner. Thus the average velocity $\overline{\mathbf{v}}_c(r)$ is initialized such that every mass is on a circular orbit about the center. Computationally this is achieved by calculating the radial acceleration a_r on each mass such that the circular velocity $v_c = \sqrt{a_r\, r}$ (again with the assumption that all particles have $m = 1$). The radial and azimuthal velocity dispersions are initialized in terms of the Toomre Q parameter (section 2.4),

$$\sigma_\phi = \sigma_r = \frac{3.36\Sigma(r)Q}{\kappa}. \tag{4.1}$$

where κ is determined from (2.108) using numerical derivative approximations. The vertical dispersion σ_z is usually much smaller than σ_r.[2]

The routines for initializing a disk galaxy are contained in the file `Initialize.cpp` listed in Appendix A. It is designed so that the most important parameters $\Sigma(r)$ and Q can be edited with ease.

4.2 Trial 1: No Dark Matter

4.2.1 Initial Conditions

For both trials, $N = 20,000$ masses are initialized according to a surface density distribution typical of an Sc class spiral galaxy [21],

$$\Sigma(r) = \Sigma_0 \left(1 + \frac{20r}{a}\right)^{-3/4} \left(1 + \frac{r}{5a}\right)^{-5/2} \tag{4.2}$$

where a is a scale length for the distribution and the disk is truncated at $r = 7a$. Σ_0 is set by normalization given N. The vertical distribution $h(z)$ is Gaussian with a dispersion $z_0 = 0.07a$. Q is set moderately to 1.2, while the vertical dispersion σ_z is set to $0.07\sigma_r$. All these parameters are listed in a more viewer-friendly format in Table 4.1.

The following figures present the state of the simulation at a handful of representative times from a top-down perspective and a side-on perspective. Time is expressed in terms of the crossing time $t_{\text{cross}} = R/\overline{v}$ where $R = 7a$ and $\overline{v}$ is the average speed of the outermost masses.

[1] Recall that the vertical thickness of a disk galaxy is generally independent of radius.

[2] In general it isn't *quite* true that radial and azimuthal velocity dispersion are equal. They are generally quite similar, however, so this isn't a bad estimate. There are more thorough treatments which involve the third Jeans equation (2.105).

Table 4.1: Important parameters for the first N-Body simulation trial of an Sc class galaxy with no dark matter.

Parameter	Value
Particles	$N = 20,000$
Softening Length	$\epsilon = 0.98N^{-0.26} = 0.075$
Surface Density	$\Sigma(r) = \Sigma_0 \left(1 + \frac{20r}{a}\right)^{-3/4} \left(1 + \frac{r}{5a}\right)^{-5/2}$
Radial Scale Length	$a = 20\epsilon$
Disk Cutoff	$r_{\max} = 7a$
Vertical Distribution	$h(z) = h_0 e^{-z^2/2z_0^2}$
Vertical Scale Length	$z_0 = 0.07a$
Toomre Q Parameter	$Q = 1.2$
In-plane Velocity Dispersion	$\sigma_\phi = \sigma_r = \frac{3.36\Sigma(r)Q}{\kappa}$
Vertical Velocity Dispersion	$\sigma_z = 0.07\sigma_r$
Timestep Tolerance Parameter	$\alpha_{\text{tol}} = 0.1$
Dark Matter Halo	None

4.2.2 Results

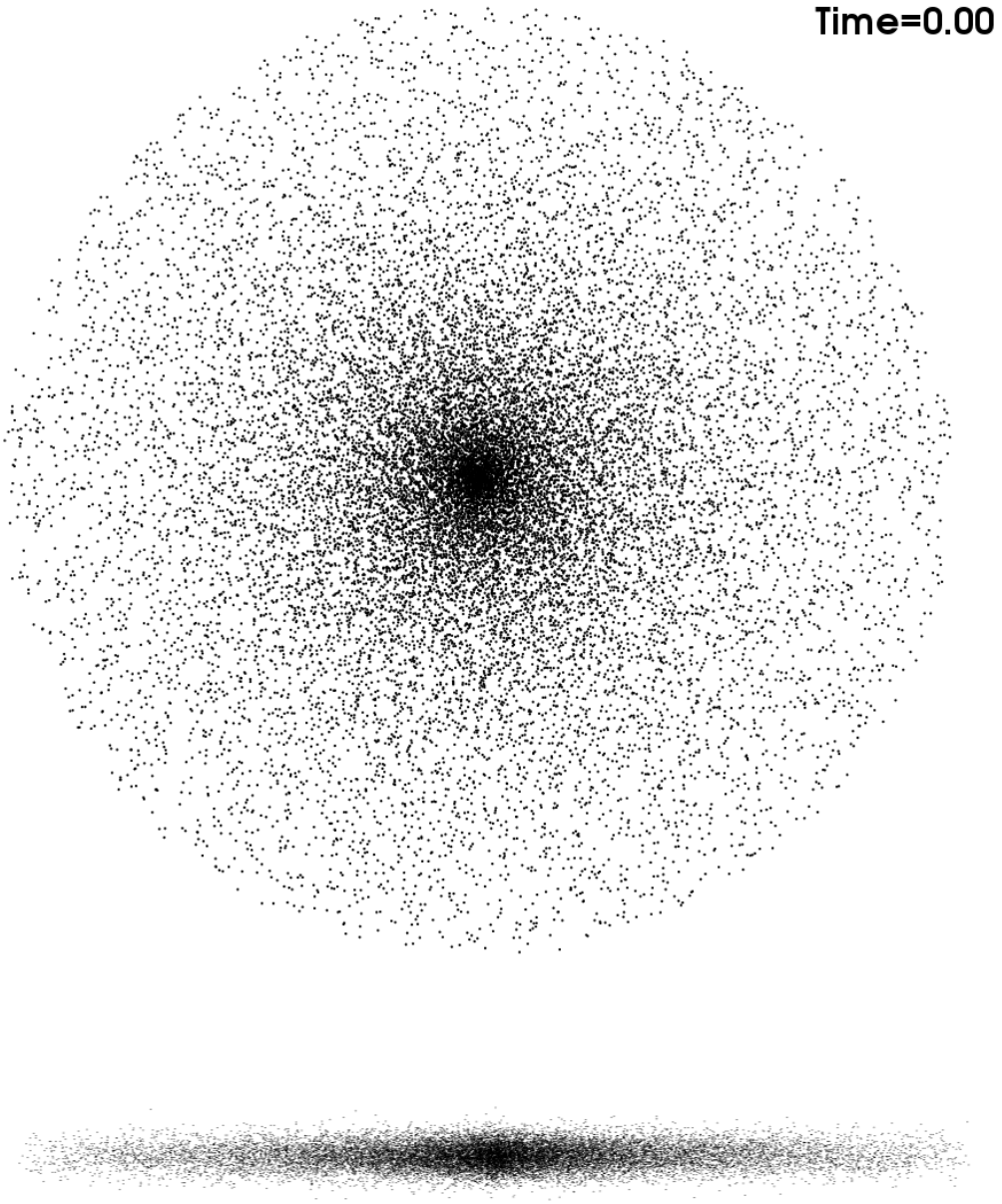

Figure 4.1: The initial configuration. The masses were placed in an axisymmetric distribution with Gaussian thickness. Each mass is on a nearly circular orbit, with random velocity set according to a Toomre stability parameter $Q = 1.2$. See Table 4.1 for the full list of parameters.

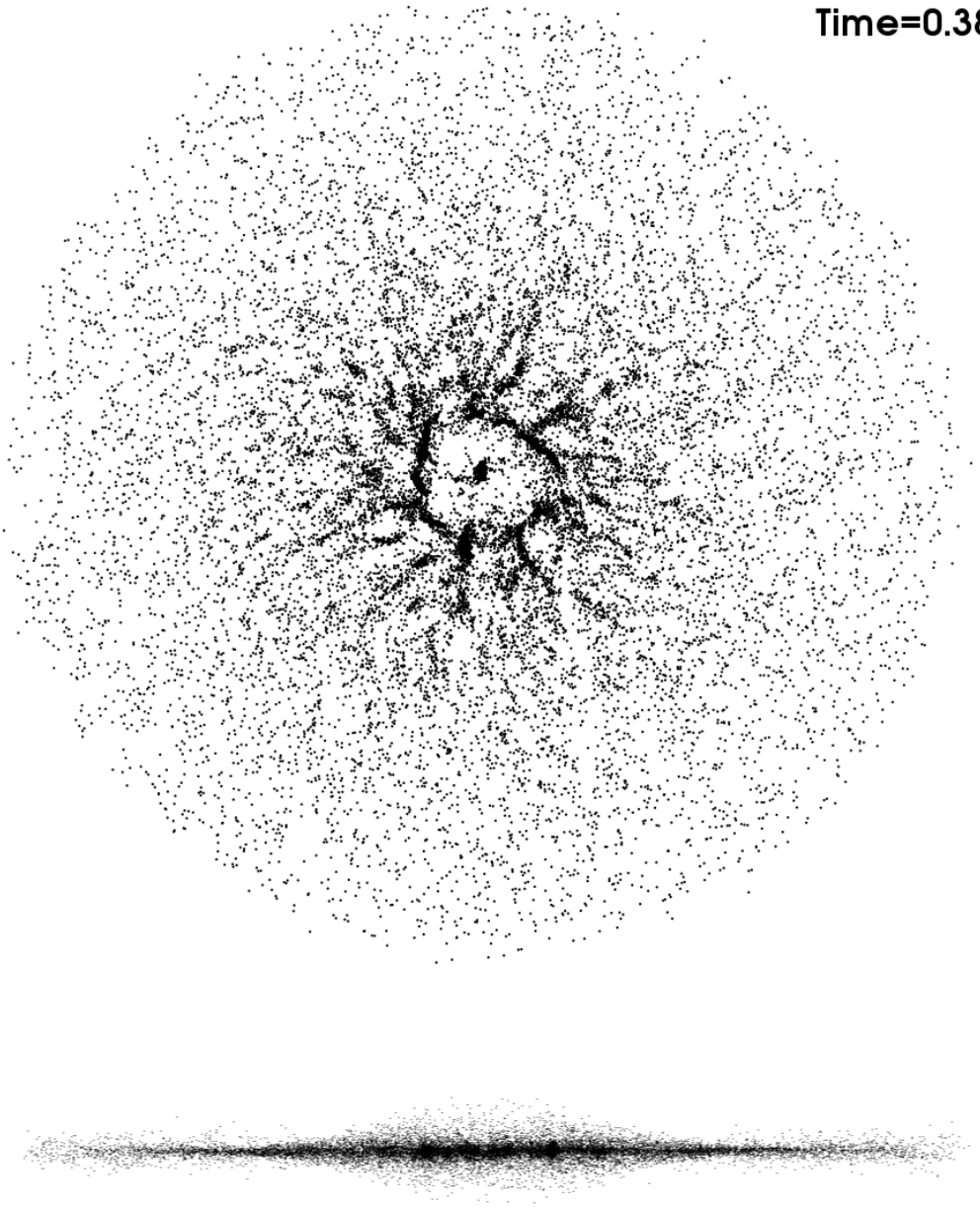

Figure 4.2: The system after 0.38 crossing times. The most central masses have collapsed into clumps surrounding the center of the disk, indicating that the inner disk was unstable in some manner. The side-on perspective shows that the vertical distribution is no longer independent of radius: in general the masses have condensed into a more narrow band in the plane of the disk. In the center, however, the distribution is about as thick as it was originally.

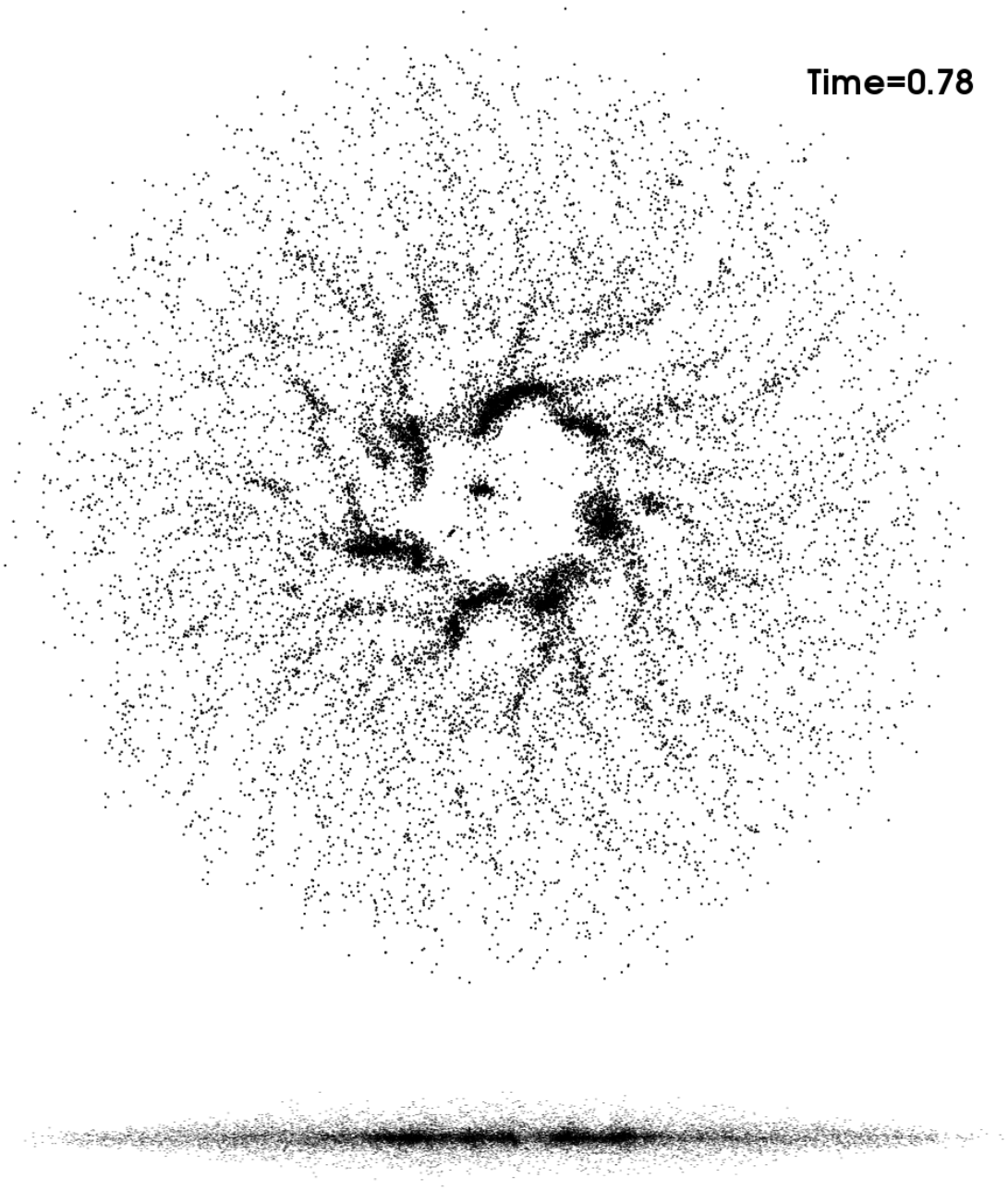

Figure 4.3: The system after 0.78 crossing times. The instability in the central disk is propagating outwards as the clusters aggregate mass. There are faint arm-like structures but nothing you would call a spiral arm. The vertical distribution seems to have evened out a bit. It is clear already that this model doesn't match reality: no galaxies have ever been observed in a form like this.

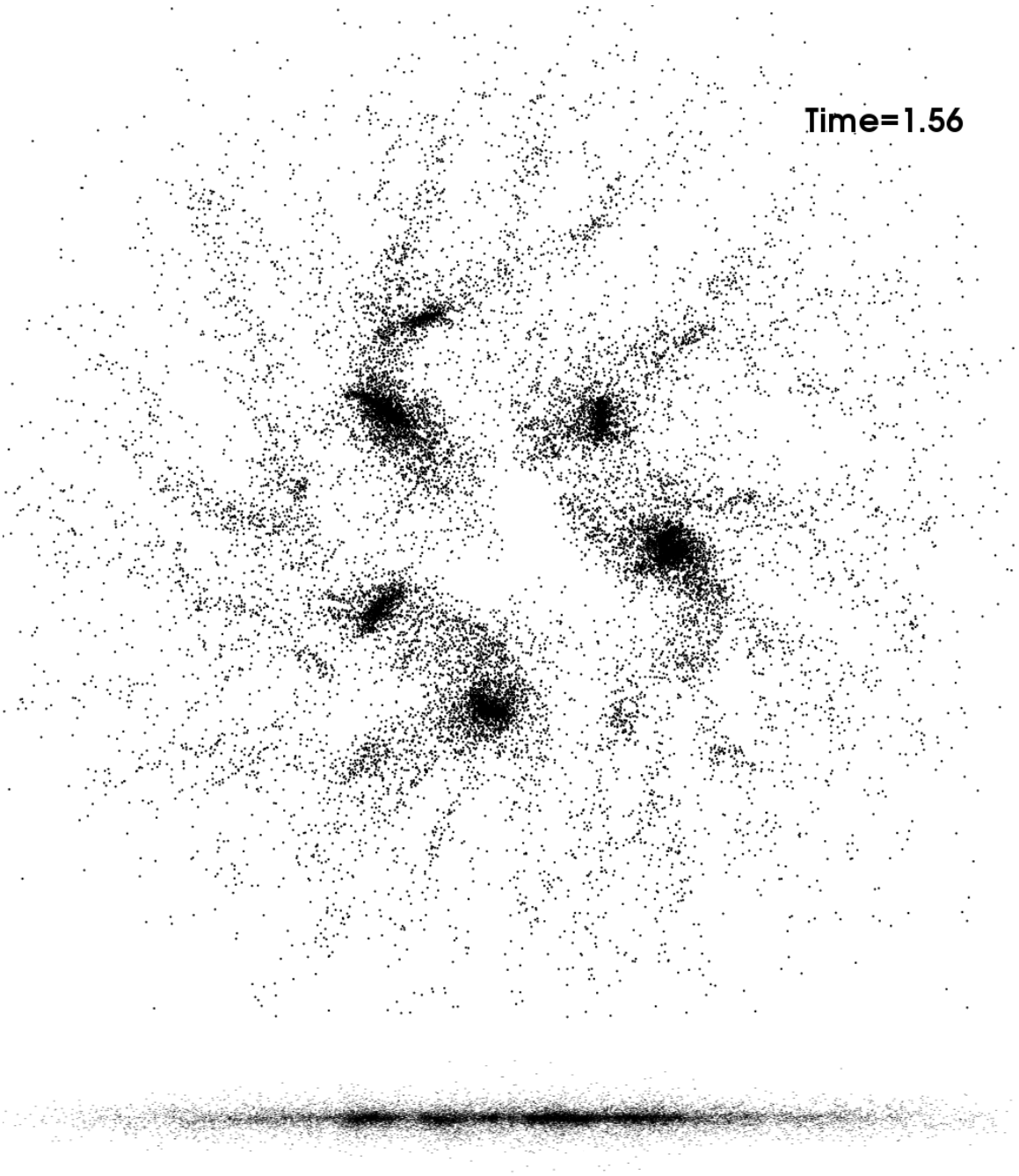

Figure 4.4: The system after 1.56 crossing times. The clusters have coalesced into five distinct groups which slowly orbit one another about halfway out from the center. Each cluster spins in a uniform direction like a disk galaxy of its own.

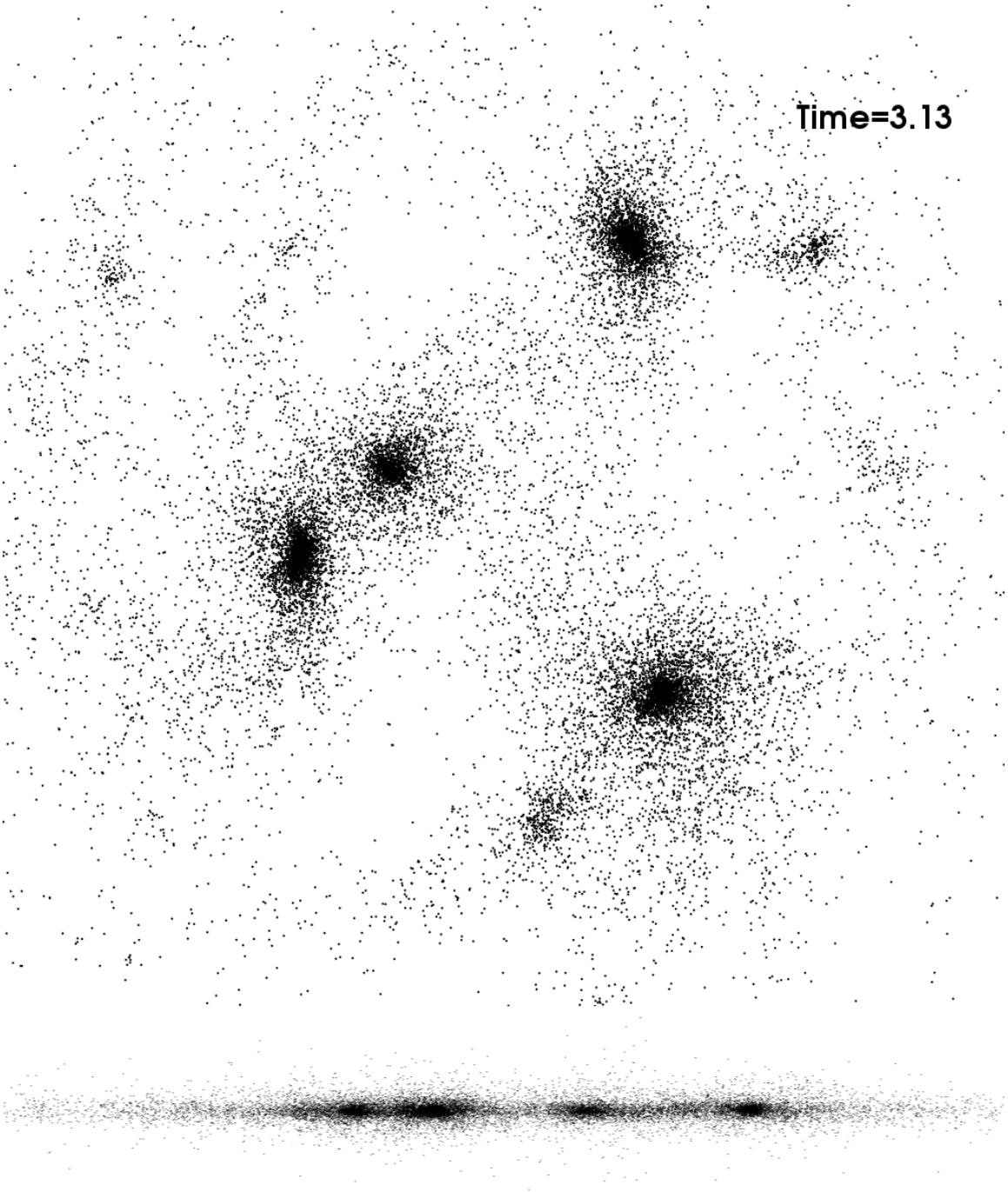

Figure 4.5: System after 3.13 crossing times. The clusters are beginning to merge as they collide due to their mutual gravitational attraction.

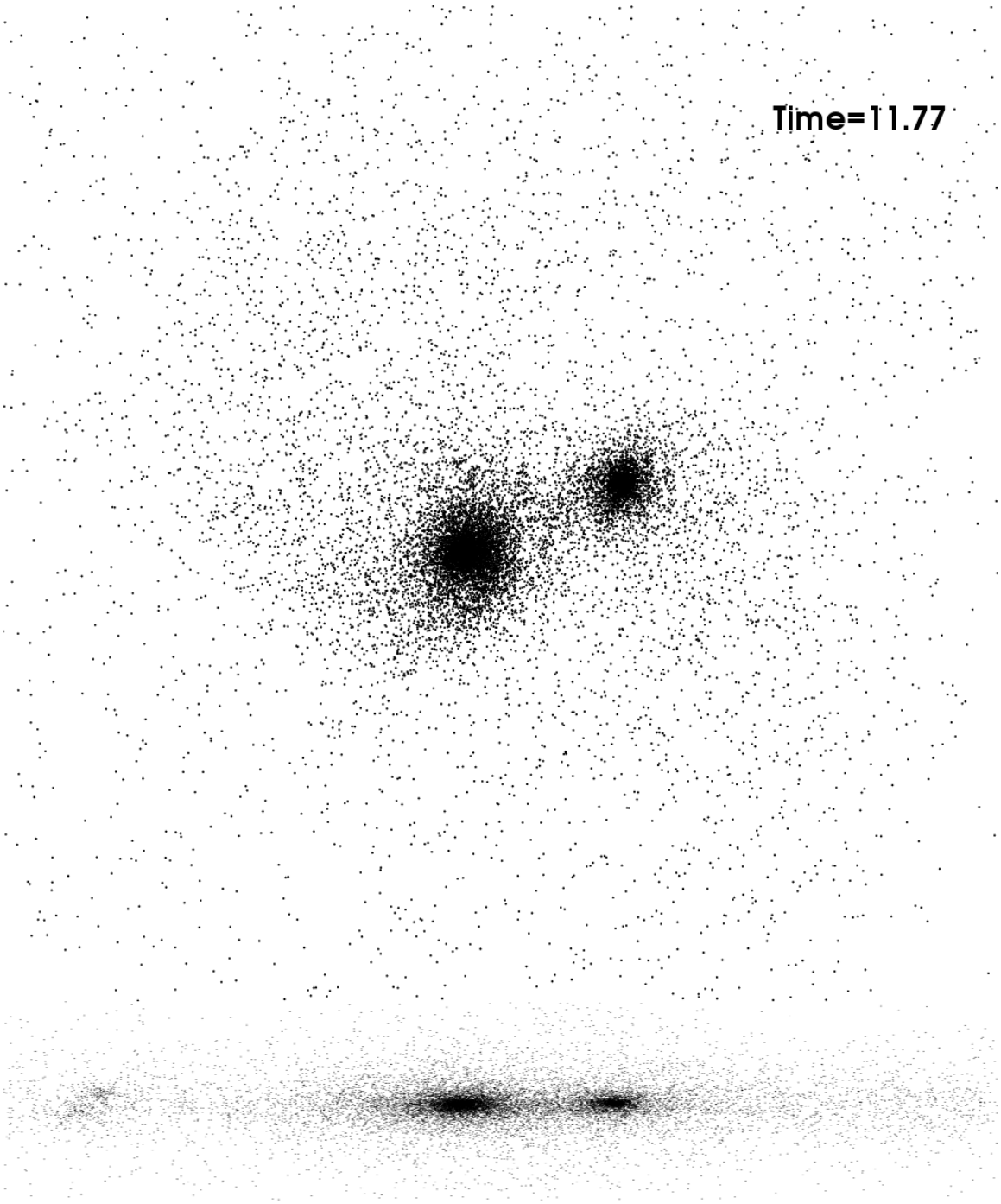

Figure 4.6: System after 11.77 crossing times. The five clusters have merged into two which now orbit one another. The vertical distribution has spread out considerably, most likely due to the repeated merging of clusters.

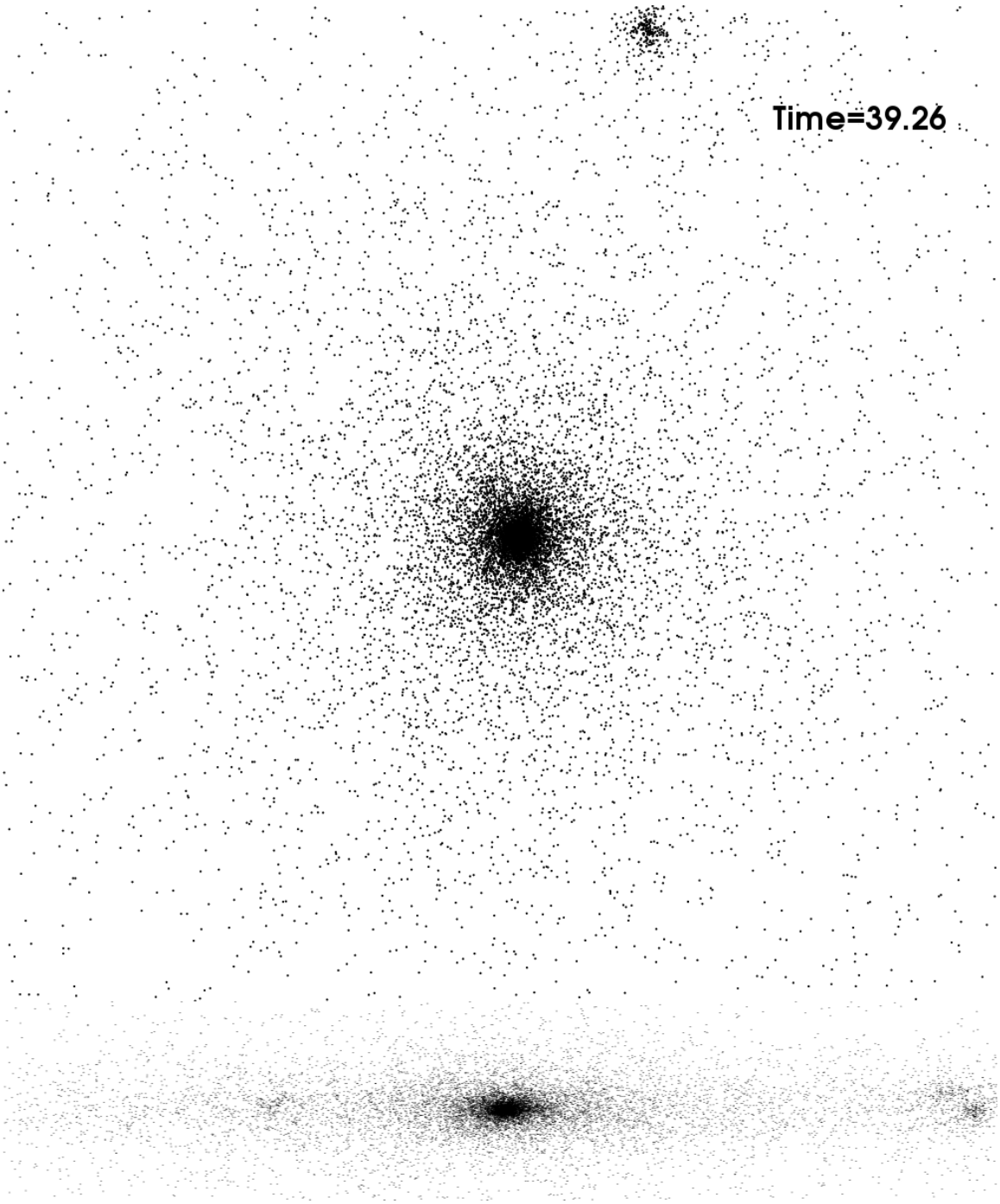

Figure 4.7: The system after 39.26 crossing times. Finally the clusters merge into a single disk again. The equilibrium configuration for the system is clearly a disk, just not one of the form it was initialized in.

4.2.3 Analysis

Well that was dramatic. The initial distribution, though constructed to model the observed distribution in an Sc class galaxy, was extremely unstable without the presence of dark matter. There is intentionally some disequilibrium in the disk, set by the stability parameter Q. This initial disequilibrium could have been increased by other artificial factors: the relatively small number of sampling masses, my hand-wavey initialization of the vertical coordinates, or extra noise introduced by numerical integration.

Even the most thoroughly constructed simulations will include some initial disturbances which the system must work out. In fact I claim that such disturbances are actually *realistic* since reality is never as ordered as physicists want it to be. The central question is the *response* of the system to these disturbances. Whatever the cause, the major point to take away is that the system wasn't locally *or* globally stable against these initial disturbances. I suspect that this is due to the absence of dark matter: there simply isn't enough mass present to hold the galaxy together in this configuration. The only way to validate this claim is to do another test.

4.3 Trial 2: With Dark Matter

There are two main ways to get information about dark matter. The most reliable way is to study the gravitational lensing of light caused by an astrophysical mass distribution. The other is to study the motion of mass in the universe. When studying dark matter distributions in a cluster of galaxies you might study the motion of galaxies; when studying dark matter *within* a galaxy you would study the motion of stars. Enter the project of this thesis: information about the distribution of dark matter may be inferred from the dynamics of stars.

On the scale of galaxy clusters, it is thought that the distribution of dark matter halos may be modeled accurately by a spherical distribution of the form

$$\rho(r) = \frac{\rho_0}{\left(\frac{r}{r_0}\right)\left(1+\frac{r}{r_s}\right)^2} \tag{4.3}$$

where r_s is a scale length. This is known as the NFW profile, you might remember it from Chapter 1. It is unclear whether this model is valid on the scale of an individual galaxy, since dark matter observations on this scale are much more difficult. On the other hand, it's about as good as any other model so it's worth a try.

In order to limit the total computational cost, I will model the dark matter distribution as a *rigid halo*, by which I mean that the evolution of its shape and composition are not considered in the dynamics of the system. As such, its dynamical contribution can be reduced entirely to an extra term $\mathbf{a}_{DM}(r)$ added to the acceleration on each mass. Newton's shell theorem tells us that the total gravitational acceleration due to a spherically symmetric mass distribution is that of a point mass,

$$\mathbf{a} = -G\frac{M(r)}{r^2}\hat{r} \tag{4.4}$$

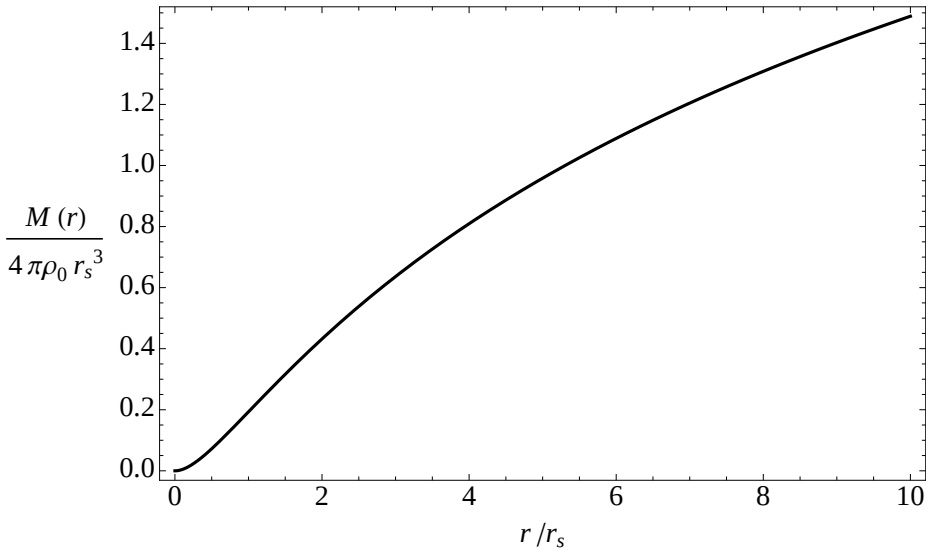

Figure 4.8: The characteristic shape of $M_{DM}(r)$ for an NFW profile dark matter halo plotted. The vertical axis is scaled by $1/(4\pi\rho_0 r_s^3)$ and the horizontal by $1/r_s$. It is clear that the total mass diverges as $r \longrightarrow \infty$.

where $M(r)$ is the total mass contained within a sphere drawn at radius r. $M_{DM}(r)$ for an NFW profile halo can be found easily from (4.3):

$$\begin{aligned} M_{DM}(r) &= 4\pi \int_0^r \rho(r') r'^2 dr' \\ &= 4\pi\rho_0 r_s^3 \left[\ln\left(1 + \frac{r}{r_s}\right) - \frac{r/rs}{1 + r/rs} \right]. \end{aligned} \tag{4.5}$$

Fig. 4.8 shows the chracteristic shape of $M_{DM}(r)$. Unfortunately it appears that the total mass of the halo is infinite since $M_{DM}(r) \longrightarrow \infty$ as $r \longrightarrow \infty$. This problem can be avoided as always by cutting off the halo at some radius $r_{\max}$. The relationship between $r_{\max}$ and r_s can be characterized in terms of a "concentration" parameter c such that

$$r_{\max} = c\, r_s. \tag{4.6}$$

The total mass of the halo is thus

$$M_{DM} = 4\pi \int_0^{r_{\max}} \rho(r') r'^2 dr' = 4\pi\rho_0 r_s^3 \left[\ln(1+c) - \frac{c}{1+c} \right]. \tag{4.7}$$

Studies of galaxy formation suggest that for a typical galaxy $c \sim 10 - 17$ [22]. It is thought that dark matter composes $95 - 99\%$ of a galaxy's total mass, thus we should set M_{DM} to be somewhere between 20 and 100 times the total stellar mass of the disk (numerically, just N). A bound may be placed on $r_{\max}$ as well since the dark matter halo extends to $7 - 9$ times the radius of the stellar disk [5].

4.3.1 Initial Conditions

In the second N-Body trial I included a moderately massive NFW profile halo with $r_{\max} = 56a$, $M_{DM} = 20M_{\text{disk}}$ and a concentration $c = 10$. All parameters relating to the initial distribution of the disk were left unchanged, though it is important to note that the average speed of each particle will have increased significantly in order to achieve a nearly circular orbit in the presence of the extra radial acceleration. The whole shabang is listed in Table 4.3.1.

Parameter	Value
Particles	$N = 20,000$
Softening Length	$\epsilon = 0.98N^{-0.26} = 0.075$
Surface Density	$\Sigma(r) = \Sigma_0 \left(1 + \frac{20r}{a}\right)^{-3/4} \left(1 + \frac{r}{5a}\right)^{-5/2}$
Radial Scale Length	$a = 20\epsilon$
Disk Cutoff	$r_{\max} = 7a$
Vertical Distribution	$h(z) = h_0 e^{-z^2/2z_0^2}$
Vertical Scale Length	$z_0 = 0.07a$
Toomre Q Parameter	$Q = 1.2$
In-plane Velocity Dispersion	$\sigma_\phi = \sigma_r = \frac{3.36\Sigma(r)Q}{\kappa}$
Vertical Velocity Dispersion	$\sigma_z = 0.07\sigma_r$
Timestep Tolerance Parameter	$\alpha_{\text{tol}} = 0.1$
Dark Matter Halo	$\rho(r) = \frac{\rho_0}{\left(\frac{r}{r_0}\right)\left(1+\frac{r}{r_s}\right)^2}$
Dark Matter Cutoff Radius	$r_{\max} = 56a$
Dark Matter Concentration	$c = 10$

4.3.2 Results

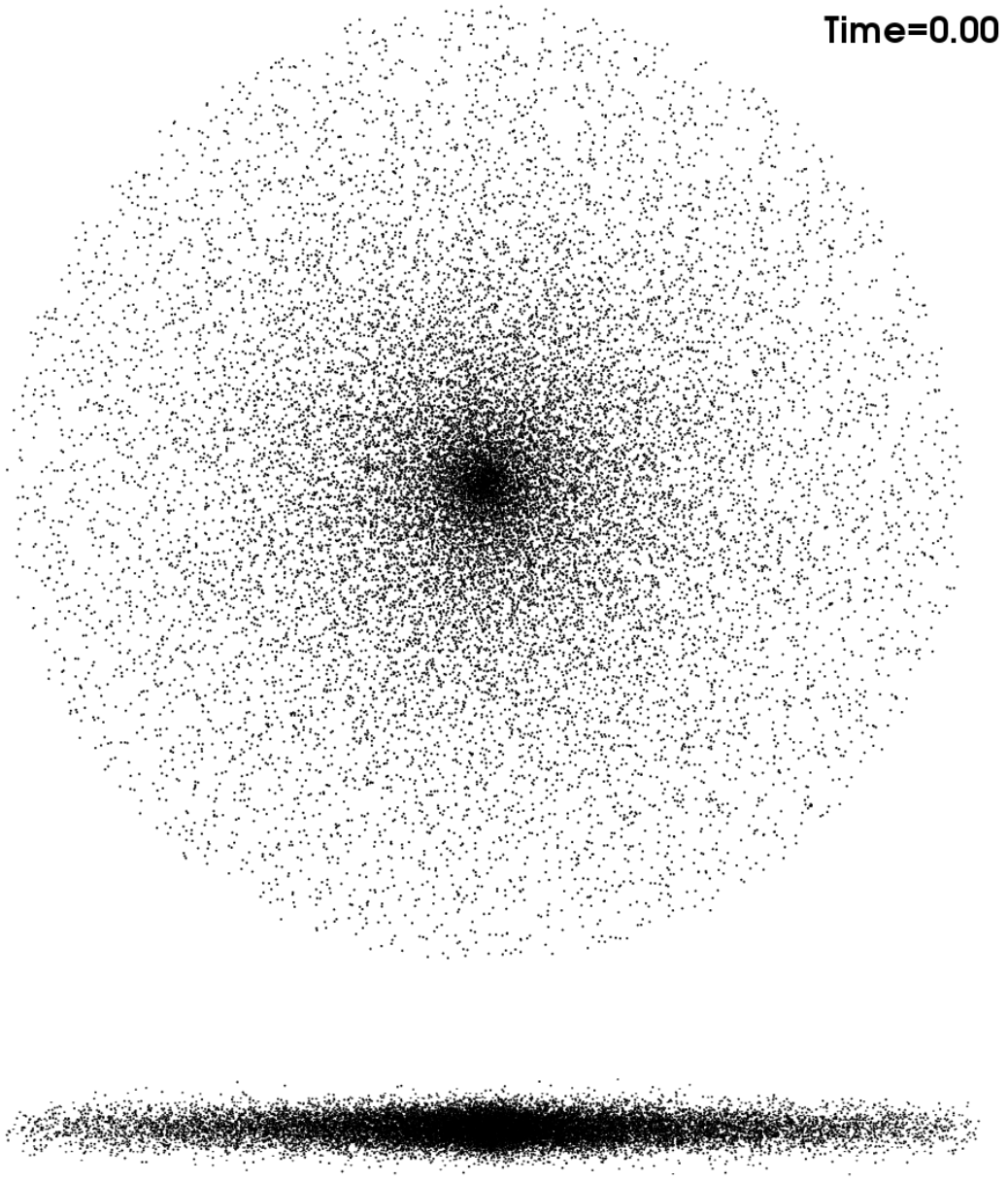

Figure 4.9: The initial configuration. The masses were placed in an axisymmetric distribution with Gaussian thickness. Each mass is on a nearly circular orbit, with random velocity set according to a Toomre stability parameter $Q = 1.2$. In addition, a rigid NFW profile Dark Matter Halo is present. See Table 4.3.1 for the full list of parameters.

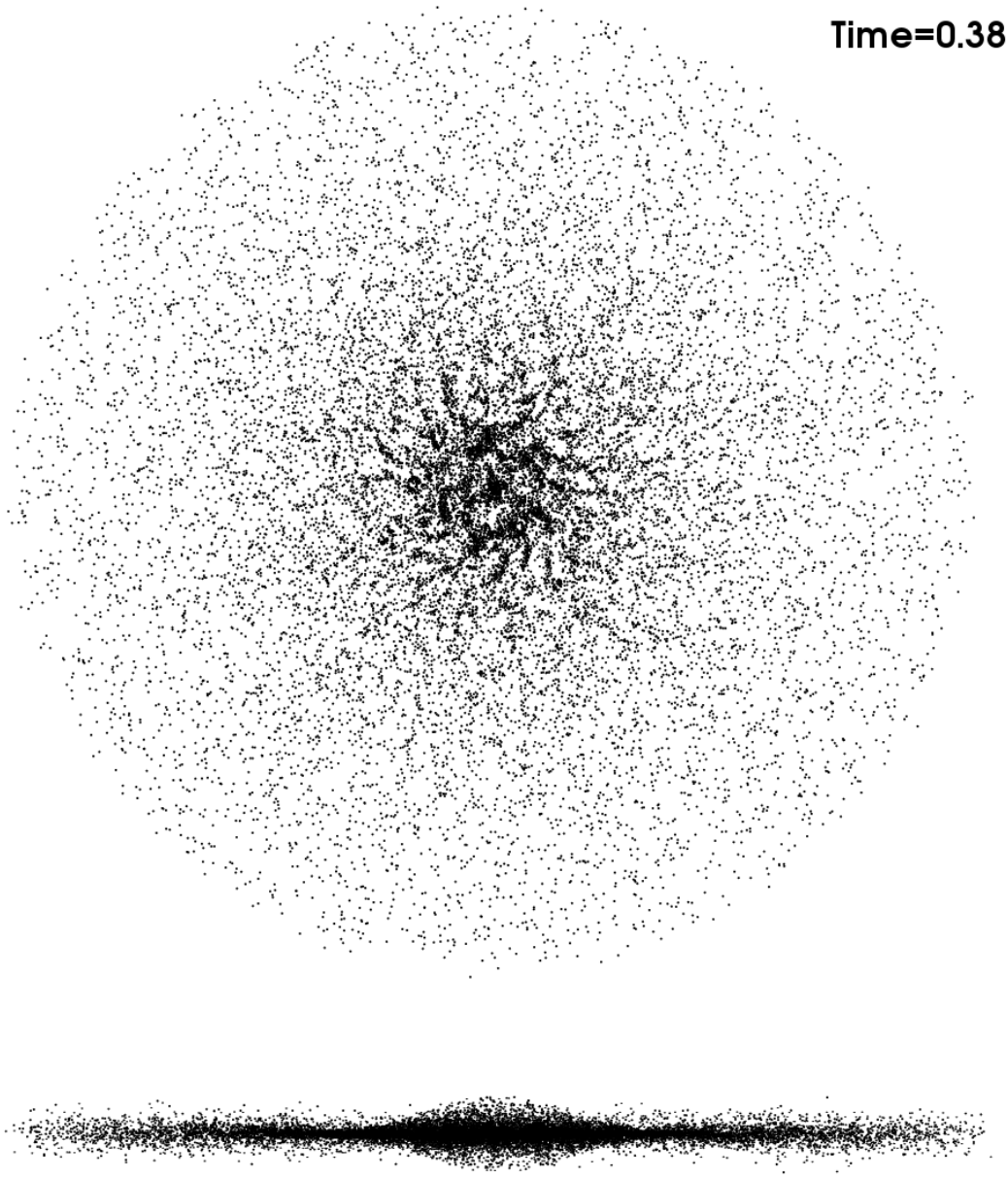

Figure 4.10: The distribution after 0.38 crossing times. The inner disk is beginning to clump as in the first simulation, though not nearly as dramatically. Some of the clumps appear to be arm-like, but as of yet they are very faint. There is evidence of vertical disequilibrium here as well since the disk is no longer uniformly thick. In general it has flattened but the inner disk is starting to swell back out.

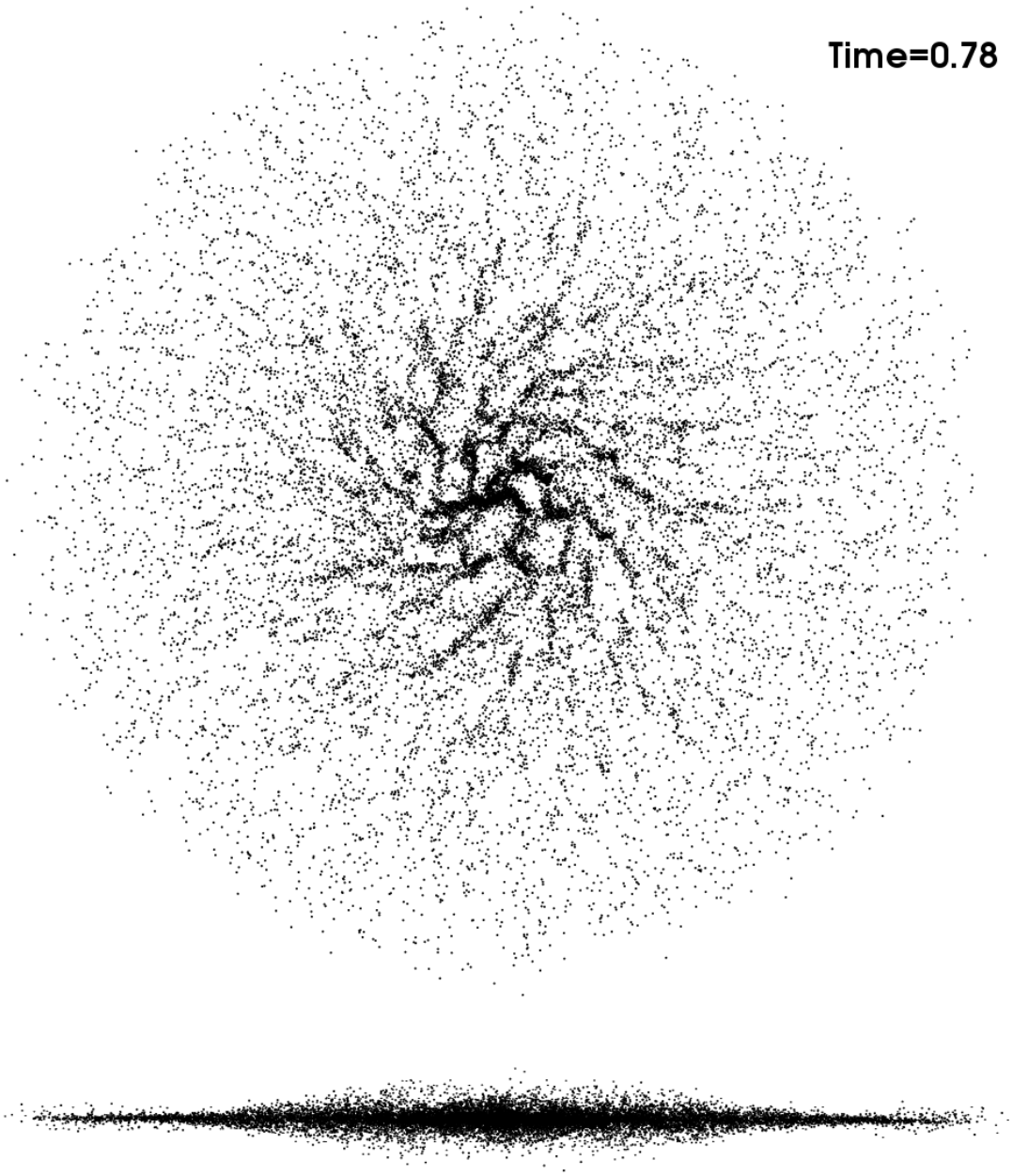

Figure 4.11: The distribution after 0.78 crossing times. By god, there's a spiral! It isn't very defined yet, but it's there. I can't convey to you my excitement when I first saw these images, I genuinely wasn't expecting this to happen. The extra radial force due to the dark matter halo manages to keep the clumps from propagating outwards. Instead, they get "stirred up" into the beginnings of a spiral pattern. The vertical distribution is slowly evening out as in the first simulation.

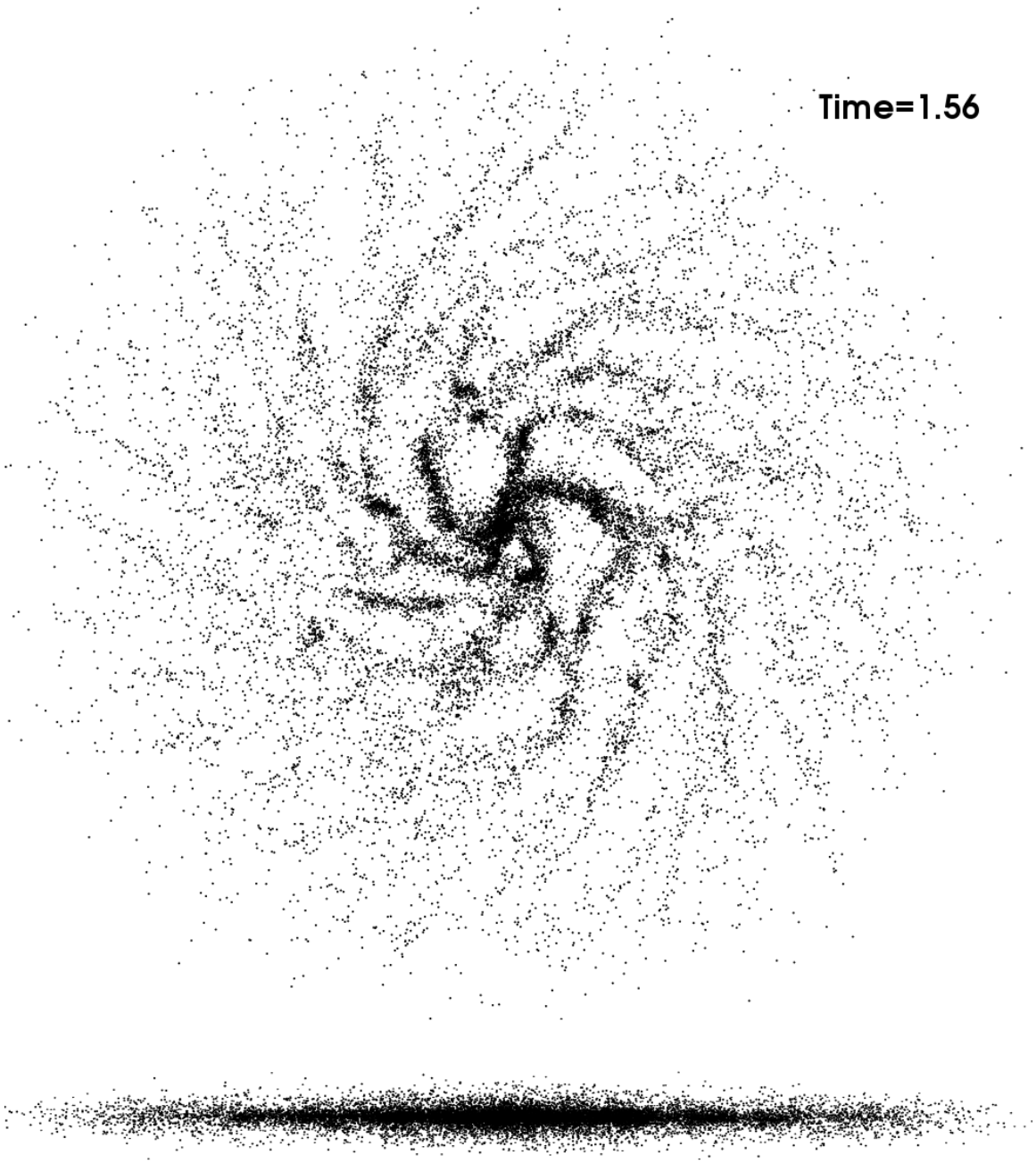

Figure 4.12: The distribution after 1.56 crossing times. The spiral pattern is now much more pronounced. The arms extend all the way to the edge of the disk, though they are most visible in the center. I count seven arms, though there are many more arm-like structures. One thing you're missing in these still frames is a visual verification that spiral arms aren't fixed groups of stars. Masses stay in an arm for a while but not forever. Often two arms will merge, or a clump from one will break off to become part of another. The vertical distribution has evened out almost entirely.

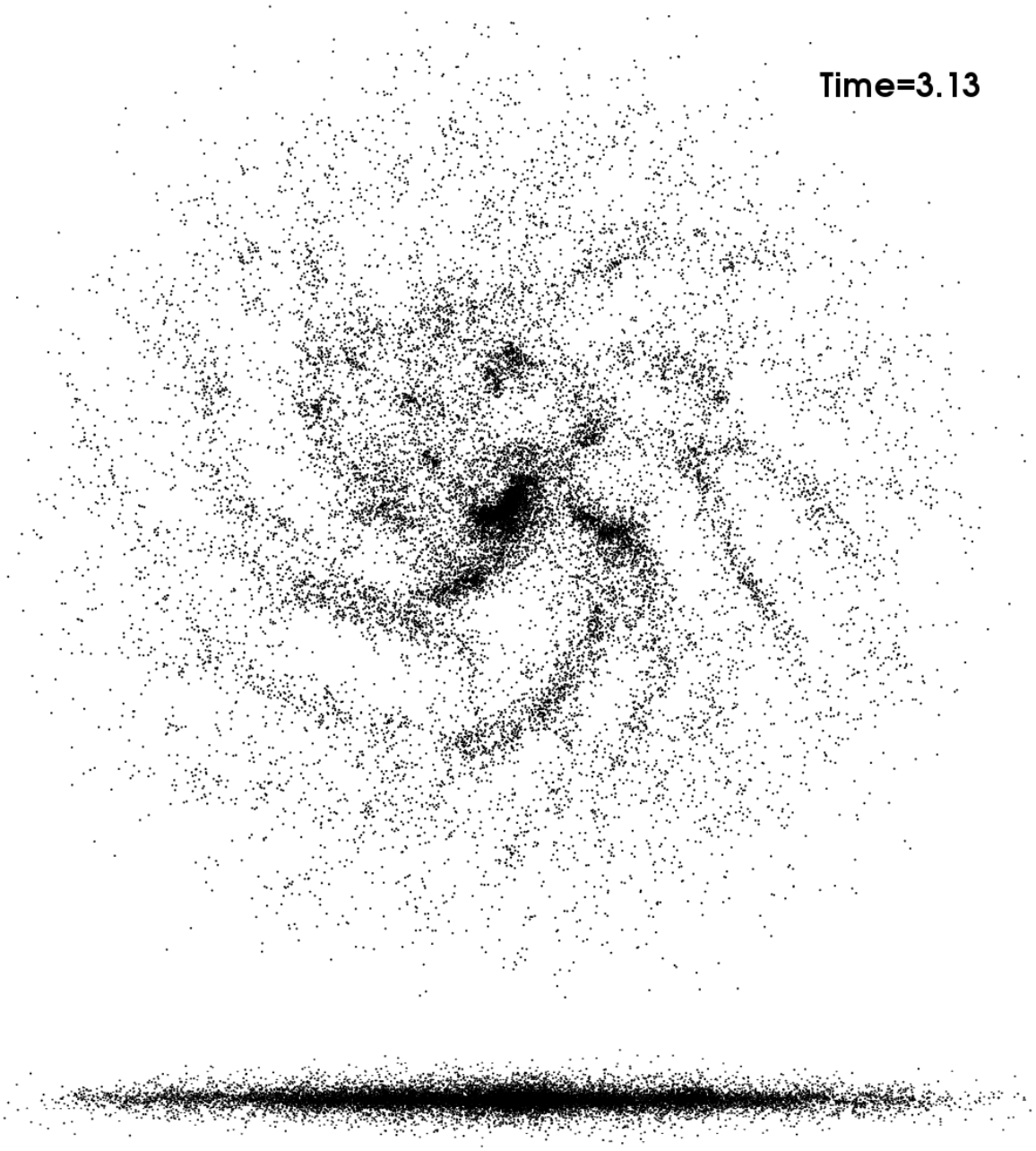

Figure 4.13: The distribution after 3.13 crossing times. The center of the galaxy has stabilized significantly but prominent spiral arms are still visible in the mid to outer disk. Clearly the arms aren't distributed in a symmetric manner at all. In fact in the upper portion of the disk the spiral is only faintly seen, traced out by clumps of mass.

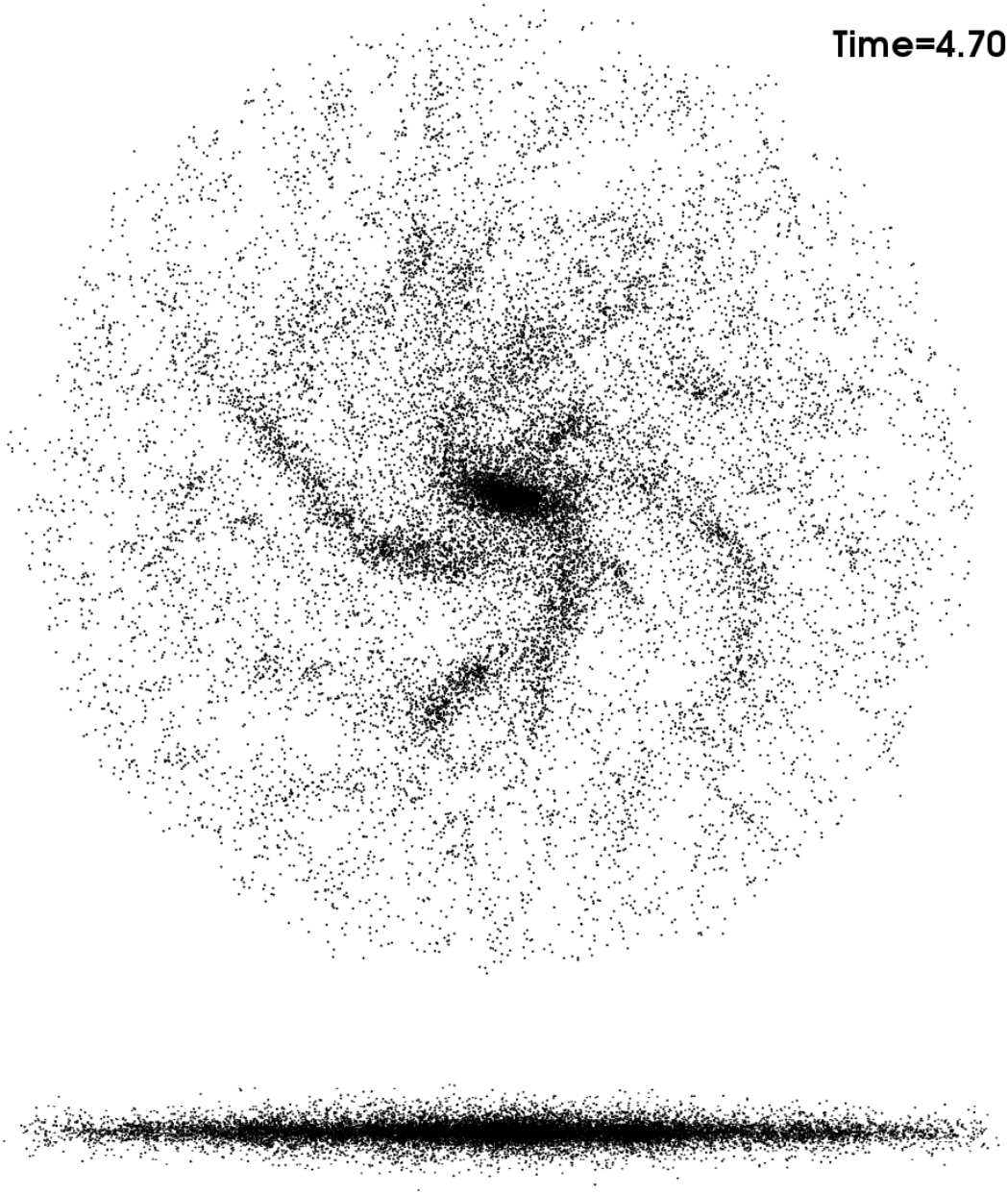

Figure 4.14: The distribution after 4.7 crossing times. The spiral arms, visible very distinctly here, are quite persistent compared to the structures in the previous simulation. There is a bar-like structure present in the center of the disk, though it doesn't last long enough to merit much consideration.

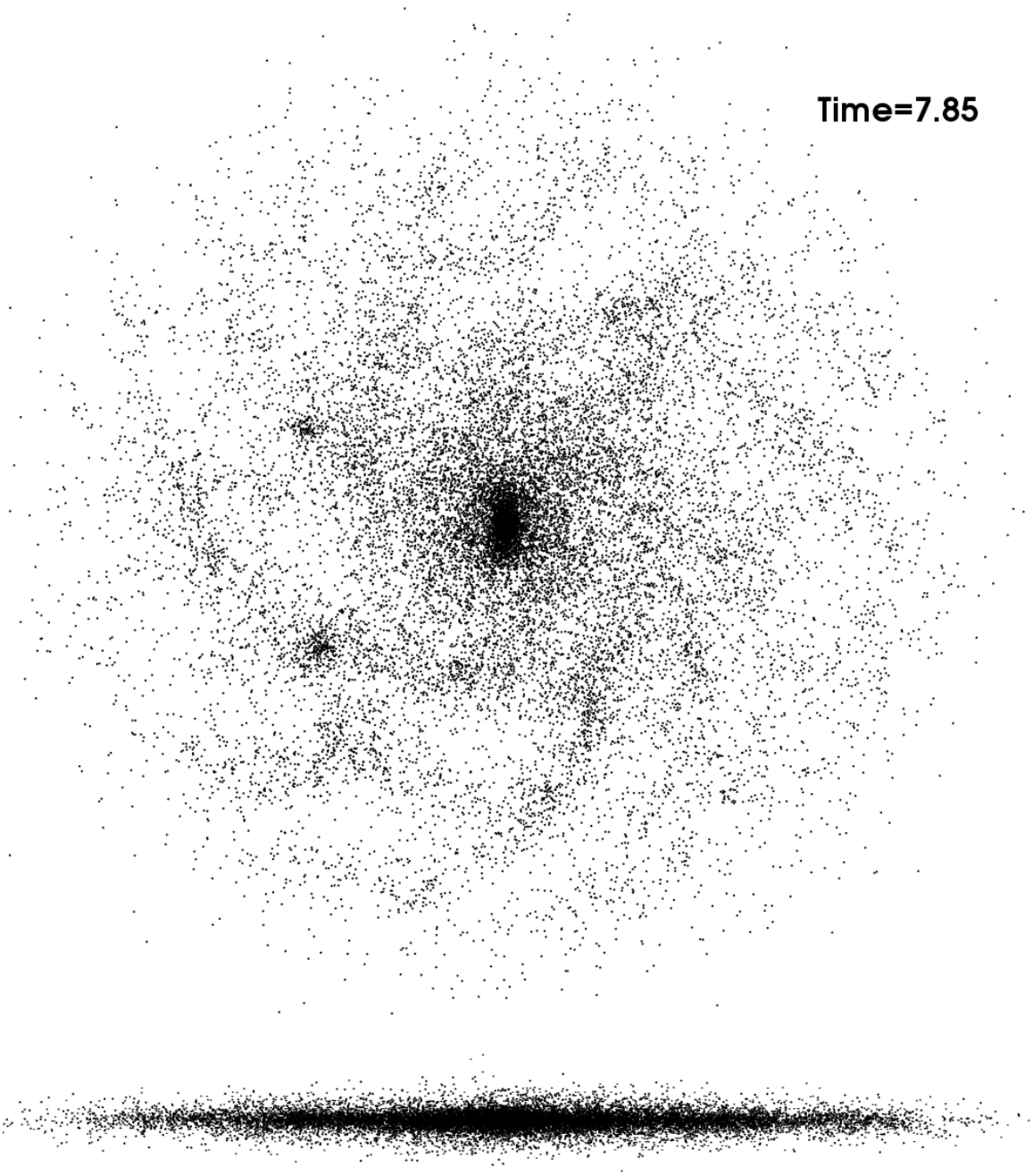

Figure 4.15: The distribution after 7.85 crossing times. A spiral pattern is still visible, though much fainter than in the previous frames. It seems that as the disk equilibrates, the spiral disappears. This suggests that a persistent spiral pattern is only possible if the disk has some mechanism by which it can avoid fully equilibrating.

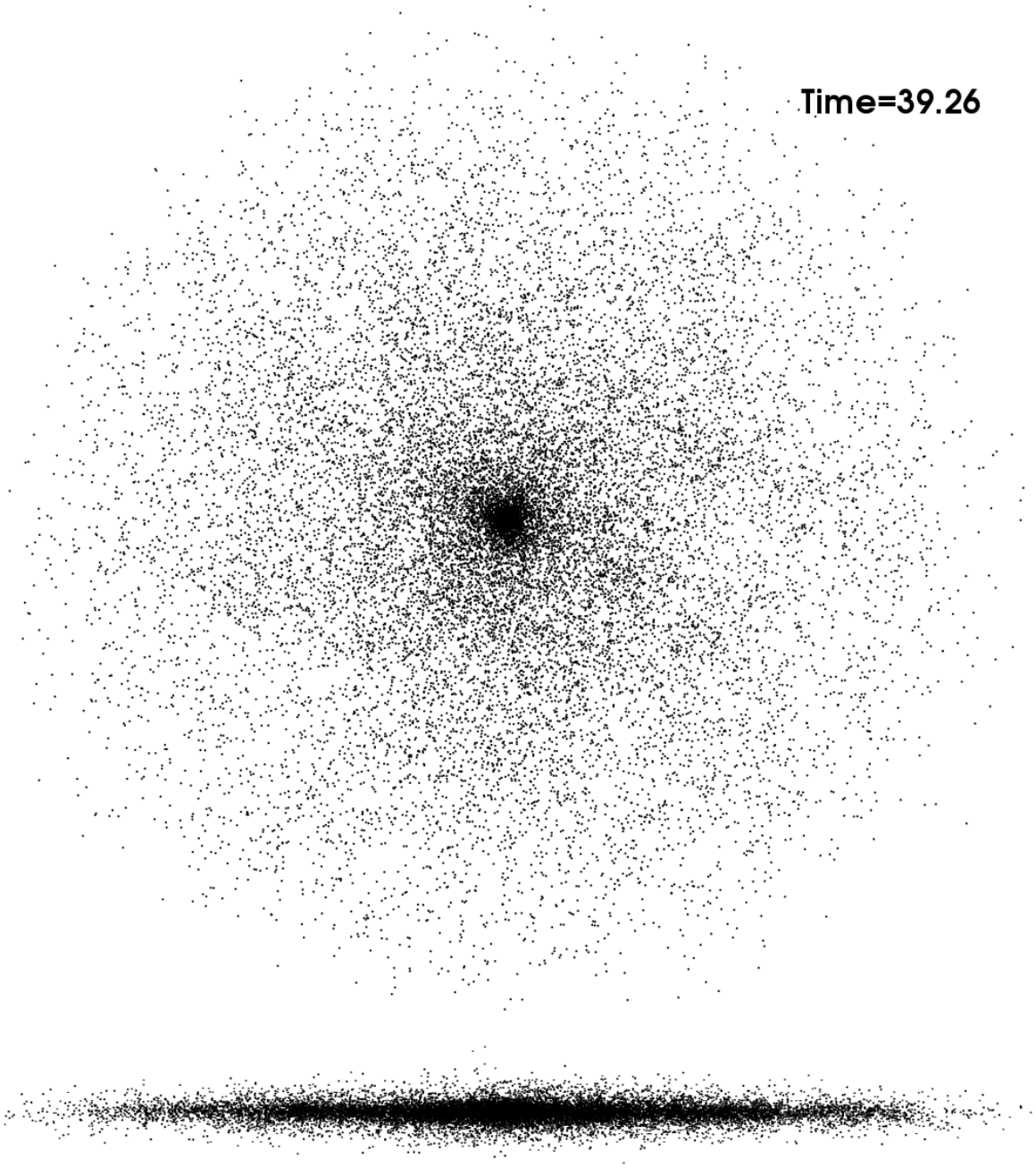

Figure 4.16: The distribution after 39.26 crossing times. At this point the spiral is completely gone, though it was very faintly visible until only slightly before this frame. The equilibrium configuration is a disk not unlike the one we started with, axisymmetric with no spiral arms.

4.3.3 Analysis

There are two results to discuss here. The first and more tangible is the simple fact that the disk stayed a disk with the addition of the rigid dark matter halo. It's pretty intuitive: the way to prevent a disk from flying apart is to add some restoring force to make sure it doesn't. To reiterate the argument from the previous analysis: random deviation from equilibrium is to be expected in any real system, the point is that the system wasn't globally or locally stable against such perturbations. A rigid dark matter halo adds the necessary stability. Therefore, as the primary result of this thesis I verify the existence of dark matter, in this case in order to ensure the stability of galactic disks.

The more nuanced result has to do with the presence of the spiral pattern. The spiral pattern was a product of initial disequilibrium. Whatever the cause, the central disk wasn't entirely happy with the configuration it was in so it began to clump into regions of higher density just outside the center. Those clumps got stretched by the rotation of the disk into a spiral pattern. The pattern grew more defined for about 5 crossing times, after which it began to fade. It remained faintly visible for nearly 30 crossing times before it disappeared. This is consistent with previous results on spiral structure which suggest that spirals are only persistent if there is some method (cooling, for example) by which the disk can avoid equilibrating [21].

This result also verifies the hypothesis that spiral arms are not composed of a fixed group of mass: though arms are present persistently for more than 25 crossing times, their composition varies in time. The arms themselves also appear and disappear with some regularity, though some spiral pattern is consistently present. It was clearly a transient phenomenon resulting from disequilibrium or the process of equilibrating.

The most striking result of all is that, remarkably, the spiral pattern is quite similar to one you might see in a real SC class galaxy (Fig. 1.5, for instance) particularly once it's fully formed. There is no reason to expect that this fake galaxy should look anything like a real one, but miraculously it does.

Conclusion

At the end of the introduction, I said that there were two goals for this thesis. The first was to understand the intricacies of N-Body simulation, and what it means as a scientific tool. The second was to analyze dynamics of disk galaxies, with the specific aim of understanding spiral arms and their relation to the presence of dark matter. I consider the second goal to be complete, with a much better outcome than I could have imagined when I started. The results of the N-Body simulations showed that without an encompassing dark matter halo, a disk galaxy (at least an Sc galaxy) is not stable. With a halo, a transient spiral pattern appeared during the initial equilibration of the disk. It seemed to confirm the predominant explanations of spiral patterns: that they are not composed of fixed sets of stars, and that unless there is some way of keeping the disk out of equilibrium they are transient.

In my opinion the more valuable insights into the dynamics of galaxies came in the process of getting to this final result. The relaxation-analysis from Section 2.1 showed that to a very good approximation a galaxy behaves as a continuous, collisionless medium of mass moving under the influence of its own gravity. In Section 2.4 it was shown that the stability of galaxies is intimately related to the *amount of random velocity* as characterized by the velocity dispersion tensor $\mathfrak{o}$. This was a prime example of exploration through physical model, for that kind of understanding would never have happened without the distribution function formalism which was developed in Section 2.3. These kinds of results deepen our understanding of the processes at work in the sky in ways that transcend the bounds of a stated result.

What about the first goal? If this thesis was an analysis of the N-Body simulation as a scientific tool, what was discovered? The most compelling argument in favor of the N-Body simulation as a scientific method is the fact that when implemented to model an Sc class galaxy, *a spiral arm showed up.* Let's think about all the approximations that resulted in the still frames from chapter four and our ability to discern in them a spiral pattern.

The first chapter was a discussion on the morphological properties of galaxies, the "truth" at least as it's understood and organized by astronomers. In the first part of Chapter two during the creation of the physical model of the galaxy we said "since gas and bulges and black holes and supernovas are too much of a headache, let's throw them all out so that all we need to think about are the stars." Not only that, but we said all the stars are exactly the same. All these assumptions are crazy, but necessary for any meaningful work to be done. In the next part of that chapter we determined that to a good degree the distribution of stars in a galaxy can be approximated

as a continuous medium of mass. Subsequently a statistical interpretation and its accompanying equations of motion were developed.

The next chapter constituted the development of a discrete approximation to the equations of motion from Chapter two. The N-Body simulation was formulated as a method which "solves" the collisionless Boltzmann equation by sketching out characteristic orbits of the distribution function and integrates Poisson's equation by Monte Carlo sampling. In order to sketch out the characteristic orbits, a numerical integration scheme was developed which approximates the true trajectories at a series of discrete times. Even the ideal numerical integrator was modified to a less theoretically pleasing form in order to accommodate the possibility of individual timesteps.

The funny thing about each of these approximations is that *for the most part* it is impossible to make any *concrete* claims about the accuracy of the approximations. To be sure some of them were very thoroughly justified, most notably in the derivation of the relaxation time which justified the continuous approximation of the stellar distribution. In every case they were *extremely well reasoned*. But in general the best we could say was "this seems pretty reasonable, let's give it a try and see what happens."

Yet in Fig. 4.14 there's a spiral arm, one that looks enough like the ones seen in photos of Sc galaxies to give you the chills. Somehow, despite the loss of realism with each approximation, those dots on that page managed to capture a fleeting glimpse of some truth at the bottom of everything.

I think it is appropriate to think of an N-Body simulation as a form of empiricism, one that is not concerned with immutable physical fact, but with the emergent order embedded in physical theory. It is a different sort of science, for it requires that the scientist relinquish control of their own theories, to allow theory to be "sketched out" in less certain ways by a computer. I think this transition is fruitful, for once control is surrendered the universe is allowed to unravel itself before your eyes, if only in approximate form, through the unknowable patterns of flipping transistors.

Appendix A

N-Body Simulation Code

Below is presented the full N-Body simulation code as it was implemented in parallel for this thesis. The code is written in C++ with the parallelization implemented with MPI (Message Passing Interface). To be honest I'm a little embarrassed about this code, since it's not quite as clean and efficient as I would like. I'm hoping that no-one actually tries to understand it. I wrote a similar variable-timestep leap integration scheme in Mathematica the summer before I started writing this thesis. Given that Mathematica can only handle like, 100 particles, I decided I needed to learn C++ in order to get N up to a reasonable number. A couple months into my thesis I had a working non-parallel version written in C++. It was well organized and easy to understand.

At this point I had to decide between writing a treecode (an $N\log N$ force calculation method) and trying to implement the code in parallel. The treecode would have provided a much larger speedup (it's about a thousand times faster) but I was finding it difficult to understand how a treecode *actually* works on an algorithmic level. It seems that computer scientists are more interested in protecting industry secrets than in the free flow of information. So, lacking the necessary information I decided to implement my code in parallel using MPI since parallelization is a simple idea (distribute information among processors, compute, recombine). I was hoping the speedup would be about $60 - 80$ if I used the whole computing cluster at Reed.

Parallelization ended up becoming a total nightmare. In fact I probably should have quit while I was ahead, but what can you do. The trouble is that MPI isn't written to work well with user-defined classes or data structures such as the ones I wrote for my non-parallel code. It is relatively simple to send arrays of data around, but not complicated classes of information. It is theoretically possible to elegantly integrate user defined classes into MPI but it seems to require a computer-scientist level of expertise. Anyways, I eventually got it to work but the elegance of my code was sacrificed in the process. Given infinite time and energy I could try to clean it up, but the end is near and I'm sprinting for the finish.

That said, I've tried to include in-code commentary about what's going on, in case some poor soul decides to read it.

A.1 Vector3.hpp/Vector3.cpp

These files contain methods and definitions for a class of three-vectors `Vector3`. It's nothing terribly interesting, but necessary since vectorized operations will be used in the rest of the code. I based this class on the general C++ vector class presented in Reference [23].

```
//
// Vector3.hpp
// Header file for Vector3 class
//
// Noah Muldavin
// Fall 2013
/*****************************************************************************/


#ifndef Classes__Vector_h
#define Classes__Vector_h

class Vector3
{
public:
    double* mData;
    Vector3(const Vector3& otherVector);
    Vector3();
    ~Vector3();
    double& operator[](int i);
    double Read(int i) const;
    double& operator() (int i);
    Vector3& operator=(const Vector3& otherVector3);
    Vector3 operator+() const;
    Vector3 operator-() const;
    Vector3 operator+(const Vector3& v1) const;
    Vector3 operator-(const Vector3& v1) const;
    Vector3 operator*(double a) const;
    double operator*(const Vector3& otherVector) const;
    double CalculateNorm() const;
    double * GetData() const;
    void SetData(double * data);
};

#endif


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

```
//
//  Vector3.cpp
//  Methods and definitions for Vector3 class
//
//  Noah Muldavin
//  Fall 2013
/*****************************************************************************/


#include <iostream>
#include <cmath>
#include "Vector3.hpp"
```

```
//  Copy constructor. Allocates memory for a new 3-Vector and copies in entries
//  from another vector

Vector3::Vector3(const Vector3& otherVector)
{
    mData = new double [3];
    for (int i = 0; i < 3; i++)
    {
        mData[i] = otherVector.mData[i];
    }
}


//  Default constructor. Allocates memory and sets all entries to 0.0

Vector3::Vector3()
{
    mData = new double [3];
    for (int i = 0; i < 3; i++)
    {
        mData[i] = 0.0;
    }
}


//  Destructor to properly free memory

Vector3::~Vector3()
{
    delete[] mData;
}


//  Overloading the square bracket operator to return a vector entry at a
//  given index

double& Vector3::operator[](int i)
{
    return mData[i];
}


//  Read-only variant of [] for those times when the program wants to mess
//  with your data

double Vector3::Read(int i) const
{
    return mData[i];
}


//  Assignment operator assigns all entries to those of another

Vector3& Vector3::operator=(const Vector3& otherVector)
{
    for (int i=0; i<3; i++)
    {
        mData[i] = otherVector.mData[i];
    }
    return *this;
}


//  Unary + operator

Vector3 Vector3::operator+() const
{
    Vector3 v;
```

```
    for (int i=0; i<3; i++)
    {
        v[i] = mData[i];
    }
    return v;
}


//  Unary - operator

Vector3 Vector3::operator-() const
{
    Vector3 v;
    for (int i=0; i<3; i++)
    {
        v[i] = -mData[i];
    }
    return v;
}


//  Binary + operator

Vector3 Vector3::operator+(const Vector3& v1) const
{
    Vector3 v;
    for (int i=0; i<3; i++)
    {
        v[i] = mData[i]+v1.mData[i];
    }
    return v;
}


//  Binary - operator

Vector3 Vector3::operator-(const Vector3& v1) const
{
    Vector3 v;
    for (int i=0; i<3; i++)
    {
        v[i] = mData[i] - v1.mData[i];
    }
    return v;
}


//  Overloaded * operator for scalar multiplication

Vector3 Vector3::operator*(double a) const
{
    Vector3 v;
    for (int i=0; i<3; i++)
    {
        v[i] = a*mData[i];
    }
    return v;
}


//  Overloaded * operator for the dot product

double Vector3::operator*(const Vector3& otherVector) const
{
    double out = 0.0;
    for (int i=0; i<3; i++)
    {
        out += data[i]*otherVector.data[i];
```

```
    }
    return out;
}


//  Method to calculate norm

double Vector3::CalculateNorm() const
{
    double norm_val, sum = 0.0;
    for (int i=0; i<3; i++)
    {
        sum += pow(mData[i], 2);
    }
    norm_val = pow(sum, 0.5);
    return norm_val;
}


//  GetData() returns the pointer to vector data. This is used in MPI transfer

double * Vector3::GetData() const
{
    return mData;
}


//  SetData method assigns vector entries given an array of three doubles. This
//  is also used in MPI transfer

void Vector3::SetData(double * data)
{
    for(int i = 0; i < 3; i++)
    {
        mData[i] = data[i];
    }
}


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

A.2 Particle.hpp/Particle.cpp

These files define a Particle class which contains all the quantities typically associated with a particle: position, velocity, current time, the prediction time and the ideal timestep for the next integration. Additionally, it contains an integer ID used to distinguish between particles. Lastly, it contains a `dump` element which is an array of double precision numbers of length nine used to store the particle's data for MPI transfer. This was implemented as a workaround since the MPI transfer of an array is *much* more straightforward than the transfer of data stored in various elements of a nice logical class. It works, though it's a bit clumsy and wastes memory. Included is a `SetDump()` method which assigns the elements of `dump` and a constructor which builds a particle class out of an array of nine doubles.

```
//
// Particle.cpp
// Header for the Particle class
//
// Noah Muldavin
// Fall 2013
```

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

#ifndef Classes_Particle_h
#define Classes_Particle_h

#include "Vector3.hpp"

class Particle
{
public:
    Particle();
    Particle& operator=(const Particle& otherParticle);
    void SetDump() const;
    Particle(double * dump);
    int particleid;
    Vector3 position;
    Vector3 velocity;
    double time;
    double predictiontime;
    double timestep;
    double * dump;

};

#endif


/*****************************************************************************/




//
// Particle.cpp
// Methods and definitions for the Particle class
//
// Noah Muldavin
// Fall 2013
/*****************************************************************************/


#include <iostream>
#include "Particle.hpp"
#include "Vector3.hpp"

//  Default constructor. Allocates memory for all entries, and sets each
//  equal to zero

Particle::Particle()
{
    particleid = 0;
    Vector3 position = Vector3::Vector3();
    Vector3 velocity = Vector3::Vector3();
    time = 0.0;
    predictiontime = 0.0;
    timestep = 0.0;
    dump = new double[9];
    for (int i = 0; i < 10; i++)
    {
        dump[i] = 0.0;
    }
}


//  Overidden assignment operator. Sets all particle entries equal to those of
//  another particle
```

```
Particle& Particle::operator=(const Particle& otherParticle)
{
    position = otherParticle.position;
    velocity = otherParticle.velocity;
    time = otherParticle.time;
    predictiontime = otherParticle.predictiontime;
    timestep = otherParticle.timestep;

    return *this;
}


//  SetDump() assigns each entry of the 9-double array "dump" a specific piece
//  of the the particle's data. This is meant for MPI transfer.

void Particle::SetDump() const
{
    dump[0] = position.Read(0);
    dump[1] = position.Read(1);
    dump[2] = position.Read(2);
    dump[3] = velocity.Read(0);
    dump[4] = velocity.Read(1);
    dump[5] = velocity.Read(2);
    dump[6] = time;
    dump[7] = predictiontime;
    dump[8] = timestep;

}


//  Constructor which assigns entries of a particle according to a 9-double
//  array as assigned above. This is meant for MPI transfer.

Particle::Particle(double * dump)
{
    position[0] = dump[0];
    position[1] = dump[1];
    position[2] = dump[2];
    velocity[0] = dump[3];
    velocity[1] = dump[4];
    velocity[2] = dump[5];
    time = dump[6];
    predictiontime = dump[7];
    timestep = dump[8];
}


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

A.3 Functions.hpp/Functions.cpp

These files contain a large amount of miscellaneous functions used in the N-Body simulation code. Some are pretty mundane (`randomdouble` for instance is just a random number generator). Some are really essential to the integration (`advance` is the function responsible for integrating a particle according to the leapfrog method, as well as updating and sorting its next timestep).

```
//
// Functions.hpp
// Header file containing miscellaneous functions used in
// the N-Body simulation
//
// Noah Muldavin
```

```
// Fall 2012
/*****************************************************************************/


#ifndef FUNCTIONS_H
#define FUNCTIONS_H

#include "Particle.hpp"

std::string inttostring (int x);
void writetofile (int N, int step, Particle *particles, Vector3 bottomleft,
                  Vector3 size, std::string fileloc);
void readfromfile (int& N, int step, Particle * Particles, Vector3& bottomleft,
                   Vector3& size, std::string fileloc);
double randomdouble(double min, double max);
Vector3 calculateforce(int N, Vector3 *r12, double eps, int i, double c,
                       double p0, double rmax);
Vector3 DMforce(Vector3 ri, double c, double p0, double rmax, double eps);
double provisionaltimestep(Vector3 force, double eps,
                           double toleranceparameter);
double sorttimesteps(double dtmax, double dtprovisional);
void initialize(int N, Particle *particles, double dtmax,
                double eps, double tolerance);
double minpredictiontime(int N, Particle *Particles);
double mintimestep(int N, Particle * Particles);
void createadvancelist(double tpmin, Particle *Particles,
                       Particle * tobeadvanced, int N, int& length);
Particle advance(Particle Particlein, Vector3 *r12, int N, double eps,
                 double atol, double dtmax);
int roundtoint (double rounded);
double energy(int N, const Particle * Particles, double eps);
Vector3 cylindricaltocartesian(Vector3 cylindrical);
Vector3 cartesiantocylindrical(Vector3 cartesian);
double Dot(Vector3 one, Vector3 two);
double GaussianRandom(double sigma);
void sphere(int N, double radius, Particle * Particles);

#endif


/*****************************************************************************/




//
// Functions.cpp
// Defines miscellaneous functions used in the N-Body simulation
//
// Noah Muldavin
// Fall 2013
/*****************************************************************************/


#include <iostream>
#include <iomanip>
#include <string>
#include <sstream>
#include <iostream>
#include <fstream>
#include <cassert>
#include <ctime>
#include <cmath>
#include <algorithm>
#include "Functions.hpp"
```

```
//  inttostring function takes an integer input and outputs a string with four
//  digits. If the integer is less than four digits, it will fill in 0s so that
//  that the string is always in the form "NNNN"

std::string inttostring (int x)
{
    std::stringstream xstream;
        //allows string "xstream" to be manipulated as an output
    xstream << std::setw(4) << std::setfill( '0' ) << x;
        //sets character # to 4, fills in 0s, imports x into xstream variable
    return xstream.str();
}


//  writetofile writes ALL particle data to a file named NNNN.txt where NNNN is
//  the timestep. The first output line contains the number of particles, the
//  second the location of the bottom left corner of the volume to be rendered
//  (x y z) followed by the size of the box (x y z). All following lines
//  contain particle data, (x y z vx vy vz time predictiontime timestep). The
//  file is designed to be used with the IFRIT particle visualization software
//  and also serve as a save-file in case something goes wrong

void writetofile (int N, int step, Particle *particles,
                  Vector3 bottomleft, Vector3 size, std::string fileloc)
{
    std::string nstring = inttostring (step);
    std::string filetype = ".txt";
    nstring+=filetype;
    std::ofstream write_output(nstring.c_str());
        //Creates file "NNNN.text"
    write_output << N << "\n";
        //Writes number of particles (N) on first line
    write_output.setf(std::ios::showpoint);
    write_output.precision(2);
        //Sets output format to one-decimal point
    for(int i = 0; i < 3; i++)
    {
        write_output << bottomleft[i] << " ";
    }
        //writes coordinates of bottom left corner of rendering volume

    for(int i = 0; i < 3; i++)
    {
        write_output << size[i] << " ";
    }
        //writes dimensions of render volume
    write_output << "\n";

    write_output.setf(std::ios::scientific);
    write_output.setf(std::ios::showpos);
    write_output.precision(7);
        //Sets format to scientific notation with 7 decimal places and a +/-
    for(int i = 0; i < N; i++)
    {
        for(int j = 0; j<3; j++)
        {
            write_output << particles[i].position[j] << " ";
        }
        for(int j=0; j < 3; j++)
        {
            write_output << particles[i].velocity[j] << " ";
        }
        write_output << particles[i].time << " ";
        write_output << particles[i].predictiontime << " ";
        write_output << particles[i].timestep << "\n";
    }
        //writes particle data
```

```
    write_output.close();
}


//  readfromfile is the opposite of writefromfile. It takes a save-file of the
//  form described above and sets all data in a Particle array

void readfromfile (int& N, int step, Particle * Particles,
                   Vector3& bottomleft, Vector3& size, std::string fileloc)
{
    std::string nstring = fileloc + inttostring(step);
    std::string filetype = ".txt";
    nstring+=filetype;
    std::cout << "Reading from " << nstring.c_str() << "\n";
    std::ifstream read_file(nstring.c_str());
    assert(read_file.is_open());
    read_file >> N;
    read_file >> bottomleft[0] >> bottomleft[1] >> bottomleft[2];
    read_file >> size[0] >> size[1] >> size[2];
    for (int i = 0; i < N; i++)
    {
        read_file >> Particles[i].position[0] >> Particles[i].position[1]
                >> Particles[i].position[2] >> Particles[i].velocity[0]
                >> Particles[i].velocity[1] >> Particles[i].velocity[2]
                >> Particles[i].time >> Particles[i].predictiontime
                >> Particles[i].timestep;
    }
    read_file.close();
}


//  randomdouble function creates a pseudo-random double precision number
//  between a specified minimum and maximum. Requires a random seed:
//  srand((double) time(NULL)) at the beginning of the file

double randomdouble(double min, double max) {
    double random = ((double) rand()) / (double) RAND_MAX;
        // returns a random double number in [0 , 1]
    double diff = max - min;
    double r = random * diff;
        // expands [0, 1] to [0, max-min]
    return min + r;
        // adjusts to [min, max] and returns
}


//  calculateforce is the all-important force-calculator. It takes as inputs
//  the position data of all particles in the form of an array of vectors r12,
//  an epsilon softening term, and the index i of the particle for which the
//  force is to be calculated. The contribution of dark matter is added at the
//  end (if you need it) according to the function DMForce below.

Vector3 calculateforce(int N, Vector3 *r12, double eps, int i, double c,
                       double p0, double rmax)
{
    double distancesquared = 0.0;
    Vector3 inc;
    Vector3 rij;
    Vector3 force;
        // initialization of internal variables
    for (int j = 0; j < N; j++)
    {
        if(j != i)
        {
            rij = r12[j] - r12[i];
            distancesquared = rij*rij + eps*eps;
            inc = rij*(1.0/pow(distancesquared, 1.5));
            force = force + inc;
```

```
        }
    }
    force = force + DMforce(r12[i], c, p0, rmax, eps);
        // sums force from all particles.
    return force;
}


//  DMforce function adds a dark matter contribution to the total force. It
//  takes as an input the position of the particle, as well as a number of
//  other parameters. As defined below the halo is an NFW profile

Vector3 DMforce(Vector3 ri, double c, double p0, double rmax, double eps)
{
    double rs = rmax/c;
    const double PI = 3.14159265359;
    double r = sqrt(ri.CalculateNorm() + eps*eps);
    double scale;
    scale = -4.0*p0*PI*pow(rs, 3.0)/(sqrt(3.0)*pow(r, 2.0));
    scale *= (log(1.0 + r/rs) - (r/rs)/(1.0 + r/rs));
    Vector3 out = ri*(1.0/ri.CalculateNorm());
    out = out*scale;
    return out;
}


//  provisional timestep function calculates the "ideal" timestep for the next
//  integration of a particle based on the timestep selection criterion. It
//  takes as input the force from the last time the particle was integrated,
//  the softening parameter, and the timestep tolerance parameter

double provisionaltimestep(Vector3 force, double eps, double tolerance)
{
    double normf = force.CalculateNorm();
    double dt = sqrt((tolerance*eps)/normf);
    return dt;
}


//  sorttimesteps assigns a quantized timestep dtk such that dtk < dt_ideal
//  as given by the provisional timestep function. This function is only used
//  once in the initialization of timesteps, for thereafter the timestep is
//  reassigned in comparison to the previous one

double sorttimesteps(double dtmax, double dtprovisional)
{
    double dtfinal = dtmax;
    if(dtfinal > dtprovisional)
    {
        do
        {
            dtfinal = 0.5*dtfinal;
        } while (dtfinal > dtprovisional);
    }
        //recursive definition iterates until dtprovisional > dtfinal
    return dtfinal;
}


//  initialize assigns timestep values for for the FIRST iteratoin of the N
//  body integration by calculating the force due to the initial particle
//  particle configuration, using this to find the ideal timestep, then sorting
//  into categories dtmax/2^n

void initialize(int N, Particle *particles, double dtmax,
                double eps, double tolerance)
{
    Vector3 r12[N];
```

```
    double dtprov;
    double dtfinal;
    Vector3 force;
    for(int j = 0; j < N; j++)
    {
        r12[j] = particles[j].position;
    }
    for(int i = 0; i < N; i++)
    {
        force = calculateforce(N, r12, eps, i, c, p0, rmax);
        dtprov = provisionaltimestep(force, eps, tolerance);
        dtfinal = sorttimesteps(dtmax, dtprov);
        particles[i].timestep = dtfinal;
        particles[i].predictiontime = 0.5*dtfinal;
    }
}


//  minpredictiontime scans a list of particles to select the minimum
//  predictiontime

double minpredictiontime(int N, Particle *Particles)
{
    double min = Particles[0].predictiontime;
    for(int i=1; i<N; i++)
    {
        if(min > Particles[i].predictiontime)
        {
            min = Particles[i].predictiontime;
        }
    }
        return min;
}


//  mintimestep scans a list of particles to select the smallest timestep. This
//  is only used in analysis of the N-Body method

double mintimestep(int N, Particle * Particles)
{
    double min = Particles[0].timestep;
    for(int i=1; i<N; i++)
    {
        if(min > Particles[i].timestep)
        {
            min = Particles[i].timestep;
        }
    }
    return min;
}


//  roundtoint rounds a double precision number to the nearest integer

int roundtoint(double toberounded)
{
    return floor(toberounded+0.5);
}


//  createadvancelist creates a list of the partilces to be advanced in the
//  next time integration

void createadvancelist(double tpmin, Particle * Particles, Particle * tobeadvanced, int N, int& length)
{
    length = 0;
    for(int i = 0; i < N; i++)
    {
```

```
        if((Particles[i].predictiontime -tpmin) < 0.000001)
        {
            tobeadvanced[length] = Particles[i];
            tobeadvanced[length].particleid = Particles[i].particleid;
            length += 1;
        }

    }

}


//  advance is the big kahuna function which advances a single particle, given
//  a list of the positions of all particles and other parameters. It first
//  calculates the force, then updates positions and velocities according to
//  the leapfrog method. Next it uses the force to update the particle's
//  by calculating the ideal timestep and sorting into the appropriate category.
//  the timestep is only allowed to increase every other step, while it can
//  decrease every step.

Particle advance(Particle Particlein, Vector3 *r12, int N, double eps,
                 double atol, double dtmax)
{
    Particle Particleout;
    double oldtimestep = Particlein.timestep;
    Vector3 force = calculateforce(N, r12, eps, Particlein.particleid);
    Particleout.velocity = Particlein.velocity + force*oldtimestep;
    Particleout.position = Particlein.position +
        (Particlein.velocity + Particleout.velocity)*(0.5*oldtimestep);
    Particleout.time = Particlein.time + oldtimestep;
    double dtprov;
    dtprov = provisionaltimestep(force, eps, atol);
    if(dtprov > dtmax)
    {
        dtprov = dtmax;
    }
    if (dtprov < oldtimestep)
    {
        Particleout.timestep = (0.5*oldtimestep);
    }
    else if((dtprov > (2.0*oldtimestep)) &&
            (roundtoint(Particleout.time/oldtimestep) % 2 == 0))
    {
        Particleout.timestep = (2.0*oldtimestep);
    }
    else
    {
        Particleout.timestep = oldtimestep;
    }
    Particleout.predictiontime = Particleout.time + (0.5*Particleout.timestep);
    Particleout.particleid = Particlein.particleid;

    return Particleout;
}


//  energy function calculates the total energy of a collection of particles

double energy(int N, const Particle * Particles, double eps)
{
    double K = 0;
    double U = 0;
    double dist;
    Vector3 rij;
    for(int i = 0; i < N; i++)
    {
        K += 0.5*Particles[i].velocity.CalculateNorm();
    }
```

```
    for (int j = 0; j < N; j++)
    {
        for (int i = 0; i < N; i++)
        {
            if(i > j)
            {
                rij = Particles[i].position - Particles[j].position;
                dist = sqrt(pow(rij.CalculateNorm(), 2.0)+pow(eps, 2.0));
                U -= 1.0/dist;
            }
        }
    }
    return K+U;

}


//  cylindricaltocartesian transforms a vector from cylindrical to cartesian
//  coordinates

Vector3 cylindricaltocartesian(Vector3 cylindrical)
{
    Vector3 out;
    out[0] = cylindrical[0] * cos(cylindrical[1]);
    out[1] = cylindrical[0] * sin(cylindrical[1]);
    out[2] = cylindrical[2];
    return out;
}


//  cartesian to cylindrical transforms a vector from cartesian to cylindrical
//  coordinates

Vector3 cartesiantocylindrical(Vector3 cartesian)
{
    Vector3 out;
    out[0] = sqrt(pow(cartesian[0], 2.0)+pow(cartesian[1], 2.0));
    out[1] = atan(cartesian[1]/cartesian[0]);
    out[3] = cartesian[3];
    return out;
}

//  Dot takes the dot product of two vectors

double Dot(Vector3 one, Vector3 two)
{
    double out = 0.0;
    for (int i = 0; i < 3; i++)
    {
        out +=one[i]*two[i];
    }
    return out;
}


//  GaussianRandom generates a random number according to a gaussian
//  distribution with dispersion sigma

double GaussianRandom(double sigma)
{
    const double PI = 3.14159265359;
    double random1 = ((double) rand()) / (double) RAND_MAX;
    double random2 = ((double) rand()) / (double) RAND_MAX;
    double out  = sqrt(-2 * sigma * log(random1)) * cos (2*PI*random2);
    return out;
}
```

```
//  sphere function distributes N particles semi-evenly in a spherical shell
//  with a given radius. The positions are assigned according to the golden
//  spiral method, which divides the sphere up into a series of N slices. A
//  mass is placed in each slice at an angle which is offset each slice by the
//  golden section of a circle

void sphere(int N, double radius, Particle * Particles)
{
    const double PI = 3.14159265359;
    double inc = PI*(3.0 - sqrt(5.0));
    double off = 2.0/((double)(N));
    double y, r, phi;
    for (int i = 0; i < N; i++)
    {
        y = ((double)(i))*off - 1.0 + (0.5*off);
        phi = ((double)(i))*inc;
        r = sqrt(1.0-y*y);
        Particles[i].position[0] = radius*r*cos(phi);
        Particles[i].position[1] = radius*y;
        Particles[i].position[2] = radius*r*sin(phi);
        Particles[i].particleid = i;
    }
}


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

A.4 Initialize.cpp/Initialize.hpp

```
//
// Initiaize.hpp
// header file defining initialization routines for a disk galaxy
//
// Noah Muldavin
// Fall 2013
/*****************************************************************************/


#ifndef INITIALIZE_H
#define INITIALIZE_H

#include "DMFunctions.hpp"

double surfacedensity(double r, double a);
void Initialize(int N, double Q, double tolerance, double c, double p0,
    double rmax, Particle * Particles, double eps, int& realN, double& dtmax);

#endif


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

```
//
// Initialize.cpp
// initialization routine for a disk galaxy
//
// Noah Muldavin
// Fall 2013
/*****************************************************************************/
```

```
#include <iostream>
#include <string>
#include <ctime>
#include <cmath>
#include <mpi.h>
#include <algorithm>
#include "DMInitialize.hpp"


//  User-defined surface density distribution function Sigma(r)
double surfacedensity(double r, double a)
{
    double c1 = 1.0+((20.0*r)/a);
    double c2 = 1.0+(r/(5*a));
    double out1 = pow(c1, -(3.0/4.0));
    double out2 = pow(c2, -(5.0/2.0));
    return out1*out2;
}

/*****************************************************************************
 *  Initialize function takes as arguments the desired number of particles N,   *
 *  the desired toomre stability parameter Q, and a timestep tolerance          *
 *  parameter atol. It returns a fully initialized galaxy with roughly N        *
 *  particles distributed according to a defined surface density distribution   *
 *  (above). Initially, all particles are located at discrete radii following   *
 *  the distribution exactly, but small gaussian random deviations are          *
 *  introduced. The particle velocities are initialized such that each particle *
 *  is traveling in a circular orbit in the counterclockwise direction.         *
 *  Gaussian random deviations from these circular trajectories are added such  *
 *  that the local velocity dispersion is defined by the Toomre stability       *
 *  parameter Q. Finally, initial timesteps are calculated.                     *
 *****************************************************************************/

void Initialize(int N, double Q, double tolerance, double c, double p0, double
            rmax, Particle * Particles, double eps, int& realN, double& dtmax)
{
    // Play star trek next generation themesong
    system("open -a QuickTime\\ Player.app ~/tng.mp3");
    //  Define PI
    const double PI = 3.14159265359;


    std::cout << "Beginning initialization of " << N << " particles\n";
    std::cout << "Stability parameter Q = " << Q << "\n";
    std::cout << "Timestep tolerance parameter = " << tolerance << "\n";


    //  Calculating softening term epsilon
    std::cout << "Softening scale epsilon = " << eps << "\n";


    //  Setting scale length a
    double a = eps/0.05;


    /*  Normalizing the surface density distribution over a discrete set of     *
     *  100 radii spaced 7a/100 apart                                           */
    std::cout << "Normalizing surface density\n";

    double * unnormalized = new double[100];
    double normalizationconst = 0.0;
    double r;

    for (int i = 0; i < 100; i++)
    {
        r = ((7.0*a)/100.0)*((double)(i+1)); // Radius i
        unnormalized[i] =surfacedensity(r, a)*2.0*PI*r;
```

```
    normalizationconst += unnormalized[i];
}


/*  Setting absolute magnitude Sigma0 such that the total number of masses  *
 *  is N, the number of particles                                           */
double Sigma0 = (((double)(N))/normalizationconst)*surfacedensity(0, a);


/*  Caluclating the number of masses at each radius. Since each value will  *
 *  be rounded from an approximation, the total number of particles might   *
 *  deviate slightly from N. Thus the total number of particles is summed   *
 *  and reset at the end                                                    */
realN = 0;

int * numatrad = new int[100];

for (int i = 0; i < 100; i++)
{
    // Rounds normalized distribution
    numatrad[i] = roundtoint(Sigma0*unnormalized[i]);
    // sums total
    realN += numatrad[i];
}

//  Resetting N
N = realN;

std::cout << "Final number of particles = " << N << "\n";


//  Freeing "unnormalized" memory
delete[] unnormalized;


/*  Declaring two vector arrays (one cartesian and one polar) to describe   *
 *  position of the particles                                               */
Vector3 * PolarGalaxy = new Vector3[N];
Vector3 * CartesianGalaxy = new Vector3[N];


/*  Initializing the polar galaxy such that the number of masses at each    *
 *  discrete radius follows the distribution "numatrad" found above. the    *
 *  masses at each radius are spread evenly over the azimuthal angles. The  *
 *  starting point of the distribution is offset by Pi/2 at each radius in  *
 *  order to prevent an asymmetric distribution from developing             */
std::cout << "Setting Initial distribution\n";

int index = 0;

for (int i = 0; i < 100; i++)
{
    r = ((7.0*a)/100.0)*((double)(i+1));
    for(int j = 0; j < numatrad[i]; j++)
    {
        PolarGalaxy[index][0] = r;
        PolarGalaxy[index][1] = ((double) (i))*0.5*PI+
                (((2.0*PI)/((double) (numatrad[i]))))*((double)(j));
        PolarGalaxy[index][2] = 0.0;
        index += 1;
    }
}


// Transforming into cartesian coordinates so that force can be calculated
for (int i = 0; i < N; i++)
{
    CartesianGalaxy[i] = cylindricaltocartesian(PolarGalaxy[i]);
```

```
}


/*  Calculating the force on each mass, as well as the magnitude of the     *
 *  force component in the radial direction                                 */
std::cout << "Calculating radial force \n";
Vector3 * force = new Vector3[N];

double * radialcomp = new double[N];
for (int i = 0; i < N; i++)
{
    force[i] = calculateforce(N, CartesianGalaxy, eps, i, c, p0, rmax);
    radialcomp[i] = Dot(force[i], CartesianGalaxy[i])*(1.0/PolarGalaxy[i][0]);
}


//  Freeing "force" memory
delete[] force;


/*  Averaging the radial force component at each discrete radius and        *
 *  calculating the angular velocity Omega associated with the circular     *
 *  orbit defined by that average force                                     */
std::cout << "Calculating angular velocity of circular orbits Omega[r]\n";

double avgforce;
double * Omega = new double[100];
int lowlim = 0; //  lower limit of summation
int uplim;  //  upper limit of summation

for (int i = 0; i < 100; i++)
{
    avgforce = 0.0; //  Resets summation
    //  sets uplim - lowim  = number of masses at radius i
    uplim = (lowlim+numatrad[i]);
    //  sums radial force components at radius i
    for (int j = lowlim; j < uplim; j++)
    {
        avgforce+=radialcomp[j];
    }
    //  dividing by number of masses at radius i to obtain average
    avgforce = avgforce/((double)(uplim - lowlim));
    //  calculates angular velocity of circular orbit given average force
    Omega[i] = sqrt(-avgforce/(((7.0*a)/100.0)*((double)(i+1))));
    lowlim = uplim;
}


//  Freeing "radialcomp" memory
delete[] radialcomp;


/*  Setting maximum timestep equal to the period of orbit of pretty-far     *
 *  out star divided by 100                                                 */
dtmax = (2.0*PI/Omega[95])/200.0;
std::cout << "dtmax = " << dtmax << "\n";


/*  Bootstrapping to a (very approximate) list of numerical derivatives of  *
 *  the angular rotation speed Omega                                        */
std::cout << "Differentiating Omega[r]\n";

double * deltaOmega = new double[100];

for (int i = 0; i < 99; i ++)
{
    deltaOmega[i] = (Omega[i+1]-Omega[i])/((7.0*a)/100.0);
}
```

```
//  Last entry is set to the average of the previous two.
deltaOmega[99] = (deltaOmega[98]+deltaOmega[97])/2;


//  Calculating k[r], the epicyclic frequency
std::cout << "Calculating epicyclic frequency k[r]\n";

double * k = new double[100];

for(int i = 0; i < 100; i++)
{
    r = ((7.0*a)/100.0)*((double)(i+1));
    k[i] = 2*Omega[i]*sqrt(fabs(1+(r/(2*Omega[i]))*deltaOmega[i]));
}

//  Freeing "deltaOmega" memory
delete[] deltaOmega;


// finally, creating a list of veloctiy dispersions sigma
std::cout << "Calculating Velocity Dispersions sigma[r]\n";

double * sigma = new double[100];

for(int i = 0; i < 100; i++)
{
    r = ((7.0*a)/100.0)*((double)(i+1));
    sigma[i] = (Q*3.36*Sigma0*surfacedensity(r, a))/k[i];
}

//  Freeing "k" memory
delete[] k;

/*  Now creating a random velocity distribution in polar coordinates with   *
 *  radial and azimuthal dispersions equal. The vertical dispersion is set  *
 *  to 0.3 times the radial dispersion.                                     */
std::cout << "Initializing Velocity\n";

Vector3 * PolarVelocity = new Vector3[N];

index = 0;
for (int i = 0; i < 100; i++)
{
    for(int j = 0; j < numatrad[i]; j++)
    {
        PolarVelocity[index][0] = GaussianRandom(sigma[i]);
        PolarVelocity[index][1] = Omega[i] + GaussianRandom(sigma[i]);
        PolarVelocity[index][2] = GaussianRandom(0.05*sigma[i]);
        index += 1;
    }
}

//  Freeing "Omega", "Sigma" and "numatrad" memory.
delete[] Omega;
delete[] sigma;
delete[] numatrad;


/*  Slightly perturbing the position of each particle with a gaussian       *
 *  random number distribution. The radial dispersion is equal to 1/10th of *
 *  the separation between discrete radii. The angular dispersion is equal  *
 *  to one  200th of the circumference at the radius of the particle. The   *
 *  vertical dispersion is set to 1/100th of the radius of the disk.        */
std::cout << "Randomizing position\n";

for (int i = 0; i < N; i++)
{
    PolarGalaxy[i][0] += GaussianRandom(((7.0*a)/100.0)/20.0);
```

```
        PolarGalaxy[i][0] = fabs(PolarGalaxy[i][0]);
        PolarGalaxy[i][1] += GaussianRandom((2.0*PI*PolarGalaxy[i][0])/2000.0);
        PolarGalaxy[i][2] = GaussianRandom(0.01*7.0*a);
    }

    /*  Allocating data for the Galaxy Dataset and transfering in data in       *
     *  cartesian coordinates                                                   */
    std::cout << "Building a galaxy\n";
    for(int i = 0; i < N; i++)
    {
        Particles[i].position = cylindricaltocartesian(PolarGalaxy[i]);
        Particles[i].velocity[0] = PolarVelocity[i][0]*cos(PolarGalaxy[i][1])
                - PolarGalaxy[i][0]*sin(PolarGalaxy[i][1])*PolarVelocity[i][1];
        Particles[i].velocity[1] = PolarVelocity[i][0]*sin(PolarGalaxy[i][1])
                + PolarGalaxy[i][0]*cos(PolarGalaxy[i][1])*PolarVelocity[i][1];
        Particles[i].velocity[2] = PolarVelocity[i][2];
        Particles[i].particleid = i;
    }

    //  Freeing "PolarGalax", "CartesianGalaxy", and "PolarVelocity" memory
    delete[] PolarGalaxy;
    delete[] CartesianGalaxy;
    delete[] PolarVelocity;


    // Initializing timesteps
    std::cout << "Initializing timesteps\n";
    initialize(N, Particles, dtmax, eps, tolerance, c, p0, rmax);

    std::cout << "Initialization Complete\n";
    std::cout << "Engage!\n";

}
```

A.5 main.cpp

This is the main routine for N-Body simulation. A large portion of the code is concerned with parallelization via MPI (Message Passing Interface). The integration is controlled and organized by a master process (`rank 0`). Each iteration it scans the full list of particles to select the minimum prediction time and generate a list of particles to be advanced. The force calculation and leapfrog integration of those particles is performed by the slave processes. In order to advance a particle they must have access to a list of the positions of *all* particles at the prediction time as well as the relevant data for the particle to be advanced. Thus the slave process sends this information to each slave as needed.

```
//
// main.cpp
// Main integration routine for N-Body simulation
//
// Noah Muldavin
// Fall 2012
/*****************************************************************************/


#include <iostream>
#include <string>
#include <ctime>
#include <mpi.h>
#include <cmath>
#include <algorithm>
#include "Vector3.hpp"
#include "Particle.hpp"
```

```
#include "DMFunctions.hpp"
#include "DMInitialize.hpp"


int main (int argc, char * argv[])
{
        //   PI, the most important number of all
    const double PI = 3.14159265359;
        //  starting random seed
    srand ((double) time(NULL));
        //  number of particles. realN is slightly different fron N because of
        //  rounding errors occuring when assigning particles in initialization
    int realN = 20000;
    int N = 20000;
        //  Allocating memory for the array of particles
    Particle * Particles = new Particle[realN];

        //  Declaring rank and size MPI quantities. rank is an index
        //  corresponding to a processor, size is the number of processor
    int rank, size;
        //  starting MPI
    MPI::Init(argc, argv);
        //  declaring MPI status variable
    MPI::Status status;
        //  setting size and rank
    size = MPI::COMM_WORLD.Get_size();
    rank = MPI::COMM_WORLD.Get_rank();

        //  Below are all the important physical and numerical parameters:
    double atol = 0.1;  //  timestep tolerance
    double Q = 1.2;     //  Toomre stability parameter
    double eps = 0.98*pow(N, -0.26);    // softening length
    double dtmax;       //  maximum timestep
    double a = eps/0.05;    //  Disk cutoff radius
        //  Here are the parameters associated with the dark matter halo:
    double c = 10.0;    //  concentration parameter
    double p0;          //  central density
    double rmax = 56.0*a;   //  maximum radius
        //  below the central density is set such that the dark matter halo
        //  is 20 times more massive than the disk
    p0 = 20.0*((double)(N))/(4.0*PI*pow(rmax/c, 3.0)*(log(1.0 + c)-c/(1.0 +c)));
    std::cout << "p0 = " << p0 << "\n";


        //  below the initialization routine is called on the root process
        //  (rank 0), then the relevant global paramters are distributed
        //  among the slaves
    if (rank ==0)
    {
        //  run the initialize routine to make a disk galaxy with N masses
    Initialize(N, Q, atol, c, p0, rmax, Particles, eps, realN, dtmax);
        //  reset N to the rounded value
    N = realN;
        //  Send newly assigned global parameters to the slave node.
        for(int j = 1; j < size; j++)
        {
            MPI::COMM_WORLD.Send(&N, 1, MPI::INT, j, 1);
            MPI::COMM_WORLD.Send(&dtmax, 1, MPI::DOUBLE, j, 2);
            MPI::COMM_WORLD.Send(&eps, 1, MPI::DOUBLE, j, 3);
        }
    }
    else
    {
        //  slave processes receive data
        MPI::COMM_WORLD.Recv(&N, 1, MPI::INT, 0, 1, status);
        MPI::COMM_WORLD.Recv(&dtmax, 1, MPI::DOUBLE, 0, 2, status);
        MPI::COMM_WORLD.Recv(&eps, 1, MPI::DOUBLE, 0, 3, status);
    }
```

```
std::cout << N << "\n";

    //  declaring variables used in particle transfer
double * data;
double out[3];
double Particleout[10];
double ans[9];
    //  numtosend variable keeps track of the number of particles left to
    //  send to the slave processes
int numtosend;
    //  tpmin is the mimumum prediction time, assigned by a search
    //  operation at each iteration
double tpmin;

    //  numsent is the number of particles sent from the master process
    //  to the slave processes
int numsent;
    //  numgot is the number of particles received from the slave processes
int numgot;

    //  iteration keeps track of how many times the integration routine
    //  has been run
int iteration = 1;
    //  maxiterations sets the maximum number of times the integration
    //  will run
int maxiterations = 100000;
    //  advanceme is a placeholder particle used by the slave processes
    //  when advancing particles
Particle advanceme;
    //  index is a variable keeping track of the particle ID of each
    //  particle sent to a slave process by MPI transfer
int index;
    //  sender keeps track of which slave process an advanced particle
    //  was sent from
int sender;
    //  Advancelist is a list of all particles to be advanced at the next
    //  integration. Since the maximum number which could be advanced is
    //  N, N particles are allocated
Particle * Advancelist = new Particle[N];
    //  r12 is a list of the positions of all partilces at the time the
    //  velocity is updated for the relevant particles
Vector3 * r12 = new Vector3[N];

    //  bottomleft is the location of the bottom left corner of the
    //  rendering box in IFRIT visualization. Below i've set it to be
    //  outside the maximum disk extent
Vector3 bottomleft;
bottomleft[0] = -9.0*a;
bottomleft[1] = -9.0*a;
bottomleft[2] = -9.0*a;
    //  rendersize is the size of the rendering box for IFRIT visualization.
    //  A quirk of the visualization software is that it will stretch any
    //  rectangular rendering box into a cube. Thus if you want your
    //  galaxy to look realistic the box should already be a cube
Vector3 rendersize;
rendersize[0] = 18.0*a;
rendersize[1] = 18.0*av;
rendersize[2] = 18.0*a;
    //  location of the file where the data will save
std::string fileloc =
    "/Network/Servers/Knowlton.local/Users/noahmuldavin/Data3/";

    //  MPI barrier to make sure all processes wait here before progressing
    //  to the main integration routine.
MPI::COMM_WORLD.Barrier();


    //  main integration routine. Iterates maxiteration times.
```

```
for(int k = 0; k < maxiterations; k++)
{
        //  syncrhonize processes, reset variables
    MPI::COMM_WORLD.Barrier();
    numgot = 0;
        //  below are the instructions for the master process, which
        //  organizes all global data and sends particles to the slave
        //  processes to be advanced
    if (rank == 0)
    {
            //  print iteration number
        std::cout << "Iteration " << k << "\n";
            //  scan list of partilces to find minimum prediction time
        tpmin = minpredictiontime(N, Particles);
            //  print time in units of the maximum timestep
        std::cout << "time = " << tpmin/dtmax << "\n";
            //  assign r12 vectors the location of each particle at tpmin
        for (int j = 0; j < N; j++)
        {
            r12[j] = Particles[j].position + Particles[j].velocity*
                (tpmin - Particles[j].time);
        }
            //  send r12 to each slave process.
        for (int j = 0; j < N; j++)
        {
                //  assigning "data" variable relevant information
            data = r12[j].GetData();
                //  send "data" to each slave
            for (int i = 1; i < size; i++)
            {
                MPI::COMM_WORLD.Send(data, 3, MPI::DOUBLE, i, j+1);
            }
                //  wait until all have received
            MPI::COMM_WORLD.Barrier();
        }
            //  create a list of all particles to be advanced
        createadvancelist(tpmin, Particles, Advancelist, N, numtosend);
            //  print the number to be sent
        std::cout << "Number advanced: " << numtosend << "\n";
            //  reset number set to 0
        numsent = 0;

            //  a conditional is required here because if there are less
            //  particles to send than the number of slave processes we
            //  need to tell some of the slave proceses to shut up.
        if(numtosend < (size -1))
        {
                // first send all the particles
            for(int i = 0; i < numtosend; i++)
            {
                    //  setting data dump for the particle to be sent
                Advancelist[numsent].SetDump();
                    //  the MPI data will be tagged with the index of the
                    //  particle
                index = Advancelist[numsent].particleid;
                    //  sending
                MPI::COMM_WORLD.Send(Advancelist[numsent].dump, 9,
                                     MPI::DOUBLE, i+1, index+1);
                    //  increasing numsent
                numsent = numsent + 1;
            }
                //  the "shut up" code is to send a message tagged with "0"
                //  to a slave process. Thus for the rest of the particles
                //  a 0 is sent.
            for(int i = numtosend; i < (size-1); i++)
            {
                MPI::COMM_WORLD.Send(MPI::BOTTOM, 0, MPI::DOUBLE, i+1, 0);
            }
```

```
        //  now receiving the advanced particle data from each
        //  slave process
    for(int i = 0; i < numtosend; i++)
    {
            //  receive particle data
        MPI::COMM_WORLD.Recv(&ans, 9, MPI::DOUBLE, MPI::ANY_SOURCE,
                             MPI::ANY_TAG, status);
            //  note which slave process it came from
        sender = status.Get_source();
            //  note the index from the tag
        index = (status.Get_tag()-1);
            //  construct a particle from the 9-array of doubles
        advanceme = Particle(ans);
            //  asssign the Particle data of appropriate index
            //  this data
        Particles[index] = advanceme;
            //  tell the relevant slave process to shut up
        MPI::COMM_WORLD.Send(MPI::BOTTOM, 0, MPI::DOUBLE, sender, 0);
    }
}

    //  now the normal routine for when there are more particles to
    //  advance than the number of processes
else
{
        // send the first (numeber of slave processes) particles
    for(int i = 0; i < (size - 1); i++)
    {
        Advancelist[numsent].SetDump();
        index = Advancelist[numsent].particleid;
        MPI::COMM_WORLD.Send(Advancelist[numsent].dump, 9,
                             MPI::DOUBLE, i+1, index+1);
        numsent = numsent + 1;
    }

        //  afterwards the send and receive operations must be
        //  syncrhonized: A new particle is sent only when an
        //  advanced particle is received from a slave
    for (int i = 0; i < numtosend; i++)
    {
            //  receive particle
        MPI::COMM_WORLD.Recv(&ans, 9, MPI::DOUBLE, MPI::ANY_SOURCE,
                             MPI::ANY_TAG, status);
            //  add it to the total
        numgot += 1;
            //  note which process it came from
        sender = status.Get_source();
            //  construct a particle out of 9-array of doubles
        advanceme = Particle(ans);
            //  note index from MPI message tag
        index = (status.Get_tag() - 1);
            //  reassign appropriate particle data
        Particles[index] = advanceme;
            //  if there are more particles to send, send one to
            //  the relevant slave process
        if (numsent < numtosend)
        {
            Advancelist[numsent].SetDump();
            index = Advancelist[numsent].particleid;
            MPI::COMM_WORLD.Send(Advancelist[numsent].dump, 9,
                                 MPI::DOUBLE, sender, index+1);
            numsent = numsent + 1;
        }
            //  if not, tell it to shut up
        else
        {
            MPI::COMM_WORLD.Send(MPI::BOTTOM, 0,
```

```
                                        MPI::DOUBLE, sender, 0);
                }
            }
        }


            //  if the time passed one fourth of the maximum timestep,
            //  write to file
        if(tpmin/dtmax >= (((double)(iteration))/4.0))
        {
            writetofile(N, iteration, Particles, bottomleft,
                        rendersize, fileloc);
            std::cout << "Wrote file # " << iteration << "\n";
            iteration +=1;
        }

    }



        //  below are the instructions for the slave processes:
    else
    {
            //  receive r12 data
        for (int j = 0; j < N; j++)
        {
            MPI::COMM_WORLD.Recv(&out, 3, MPI::DOUBLE, 0, j+1, status);
                //  set vector data from 3-array of doubles
            r12[j].SetData(out);
                //  wait
            MPI::COMM_WORLD.Barrier();
        }

            //  receive the first particle to advance
        MPI::COMM_WORLD.Recv(&Particleout, 9,
                             MPI::DOUBLE, 0, MPI::ANY_TAG, status);
            //  if the tag wasn't a zero
        if(status.Get_tag() > 0)
        {
            //  advance the particle
            do
            {
                    //  note particle index from MPI tag
                index = status.Get_tag()-1;
                    //  create a particle out of 9-array
                advanceme = Particle(Particleout);
                    //  set particle's ID
                advanceme.particleid = index;
                    //  advance particle
                advanceme = advance(advanceme, r12, N, eps, atol,
                                    dtmax, c, p0, rmax);
                    //  reset data dump
                advanceme.SetDump();
                    //  send data back to master process
                MPI::COMM_WORLD.Send(advanceme.dump, 9,
                                     MPI::DOUBLE, 0, index+1);
                    //  wait for new particle
                MPI::COMM_WORLD.Recv(&Particleout, 9, MPI::DOUBLE, 0,
                                     MPI::ANY_TAG, status);
            } while (status.Get_tag() > 0); //  repeat if tag wasn't 0
        }
    }
}

    //  close processes
MPI::Finalize();

return 0;
```

```
}

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

References

[1] K. Masters, *Curious about the astronomy? ask an astronomer: Where does the name "milky way" come from?*, `http://curious.astro.cornell.edu/question.php?number=94` (March 2001).

[2] A. Aveni, *Stairway to the Stars: Skywatching in Three Great Ancient Cultures* (Wiley & Sons, Hoboken, NJ, 1999), 2nd ed.

[3] S. L. Jaki, *The Milky Way: An Elusive Road for Science* (Neale Watson Academic Publications, Inc, New York, NY, 1972), 1st ed.

[4] P. Kunitzch, The Messenger of the European Souther Observatory **49**, 42 (1987).

[5] B. W. Carroll and D. A. Ostlie, *An Introduction to Modern Astrophysics* (Pearson Education, Inc, San Francisco, CA, 2007), 2nd ed.

[6] G. Johnson, *Miss Leavitt's Stars: The Untold Story of the Woman Who Discovered How to Measure the Universe* (W.W. Norton & Company, Inc., New York, NY, 2005).

[7] R. W. Smith, *The Expanding Universe: Astronomy's 'Great Debate' 1900 - 1931* (Cambridge University Press, Cambridge, UK, 1984).

[8] M. A. Hoskin, J. Hist. Astron. **7**, 169 (1976).

[9] J. Binney and S. Tremaine, *Galactic Dynamics* (Princeton University Press, Princeton, NJ, 2008), 2nd ed.

[10] H. Mo, F. van den Bosch, and S. White, *Galaxy Formation and Evolution* (Cambridge University Press, Cambridge, UK, 2010).

[11] J. A. Sellwood, ArXiv e-prints (Jun. 2010), 1006.4855.

[12] J. F. Navarro, C. S. Frenk, and S. D. White, Astrophys.J. **462**, 563 (1996), astro-ph/9508025.

[13] A. Toomre, Astrophys. J. **139**, 1217 (May 1964).

[14] A. B. Romeo, Astron. Astrophys. **324**, 523 (1997).

[15] R. A. Gerber, Astron. J. **466**, 724 (1996).

[16] A. B. Romeo, Astron. Astrophys. **286**, 199 (1994).

[17] E. Athanassoula, J. C. L. E Fady, and A. Bosma, Mon. Not. R. Astron. Soc. **000**, 1 (1999).

[18] S. J. Aarseth, *Gravitational N-body Simulations: Tools and Algorithms* (Cambridge University Press, 2003).

[19] V. Springel, Monthly Notices of the Royal Astronomical Society **364**(4), 1105 (2005), http://dx.doi.org/10.1111/j.1365-2966.2005.09655.x.

[20] V. Springel, N. Yoshida, and S. D. White, New Astron. **6**, 79 (2001), astro-ph/0003162.

[21] J. A. Sellwood and R. G. Carlberg, Astrophys. J. **282**, 61 (Jul. 1984).

[22] A. Klypin, H. Zhao, and R. S. Somerville, Astrophys. J. **573**, 597 (Jul. 2002), arXiv:astro-ph/0110390.

[23] J. Pitt-Francis and J. Whiteley, *Guide to Scientific Computing in C++* (Springer-Verlag, London, 2011).

[24] *Normal galaxies*, http://hendrix2.uoregon.edu/~imamura/123/lecture-3/lecture-3.html.

[25] J. Voisey, *Psa: Bars kill galaxies*, http://www.universetoday.com/77810/psa-bars-kill-galaxies/ (November 2010).

[26] R. Thompson, *Spiral galaxy ngc 1376*, http://www.spacetelescope.org/images/opo1000a/.

[27] *M51 as recorded by lord rosse, 1845*, http://amazing-space.stsci.edu/resources/explorations/groundup/lesson/basics/g44/.

[28] J. R. Primack, *From cosmology and culture*, http://ircamera.as.arizona.edu/NatSci102/NatSci/images/extcosmo.htm (2002).

[29] W. Herschel, Philosophical Transactions of the Royal Society of London **75**, 213 (1785).

[30] T. R. Quinn, N. Katz, J. Stadel, and G. Lake, Astrophys.J. (1997), astro-ph/9710043.

[31] T. S. Kuhn, *The Structure of Scientific Revolutions* (The University of Chicago Press, Chicago, IL, 1996), 3rd ed.