N-Body Simulations of Barred Galaxies

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

Thomas B Malthouse

Summer 2017

Approved for the Division
(Physics)

J Powell

Acknowledgements

I want to thank a few people.

Preface

This is an example of a thesis setup to use the reed thesis document class.

List of Abbreviations

You can always change the way your abbreviations are formatted. Play around with it yourself, use tables, or come to CUS if you'd like to change the way it looks. You can also completely remove this chapter if you have no need for a list of abbreviations. Here is an example of what this could look like:

ABC	American Broadcasting Company
CBS	Columbia Broadcasting System
CDC	Center for Disease Control
CIA	Central Intelligence Agency
CLBR	Center for Life Beyond Reed
CUS	Computer User Services
FBI	Federal Bureau of Investigation
NBC	National Broadcasting Corporation

Table of Contents

List of Tables

List of Figures

Abstract

The preface pretty much says it all.

Dedication

You can have a dedication here if you wish.

Introduction

The Milky Way is an entirely unextraordinary galaxy. Its hallmark spiral arms—visible in the night sky as a bright smearing of stars, stretching from horizon to horizon—are shared by about 60% of galaxies in our universe (?). A bar—a large collection of stars passing through the galactic center, prominent in renderings of the Milky Way (such as Fig. ??)—is also found around two in three other spiral galaxies. Under the Hubble classification system commonly used to sort and organize galaxies, the Milky Way is classified as an *Sb* type galaxy, along with 40% of known galaxies (for more information, see ??). These extensive similarities mean that studying the evolution and structure of the Milky Way can provide insights about galactic behavior in general, and that observing other galaxies can reveal the past and future of out own.

Disks

The Milky Way has two main disks which hold the vast majority of visible matter in the galaxy and give rise to its spiral structure. The *thin disk* is the more visible of the two, composed mainly of main-sequence stars and clouds of gas and dust (?). Its vertical density scale height—the distance over which its density decreases by a factor of e—is about $350\,\mathrm{pc}$—very thin compared to its radius of about $25\,\mathrm{kpc}$. This thinness comes from its young age, since the stars that compose it are less likely to have had their orbits perturbed—especially in the chaotic period about $9\,\mathrm{Gyr}$ ago. The thin disk accounts for about 97% of the galaxy's (normal) mass and holds nearly all galactic dynamism and stellar formation.

The other disk, referred to as the *dark disk* or *thick disk*, is far older and less dynamic (?). Composed of stars formed $10\,\mathrm{Gyr}$ to $12\,\mathrm{Gyr}$ ago, it is very faint and hard to detect—all the bright stars burned out long ago, and the only ones left are low-magnitude red dwarfs and K-class stars. These stars' orbits also tend to be less regular, since they've had time to be perturbed and pushed into new orbits, especially during the chaotic initial organization of the Milky Way $10\,\mathrm{Gyr}$ ago—resulting in a scale height of about $1\,\mathrm{kpc}$. Because its stars are so steady-burning and the complete lack of gas and dust, the thick disk is very stable and exhibits none of the dynamism seen in the thin disk.

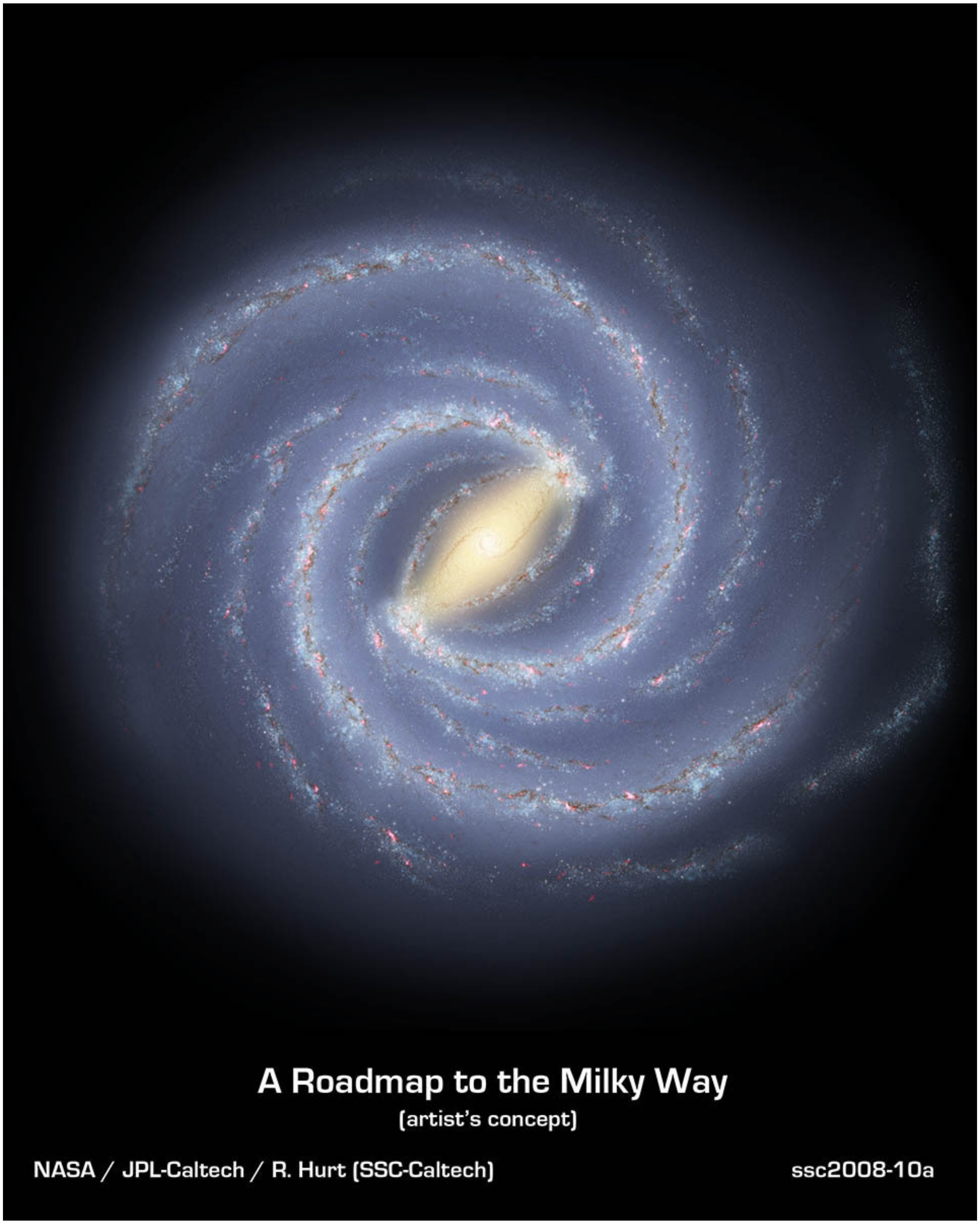

Figure 1: This rendering of the Milky Way clearly shows the stellar bar going through the center of the galaxy, and the extensive spiral system. *Image courtesy NASA/JPL*

It accounts for only about 3% of the regular matter in the galaxy, with a total mass of about $1 \times 10^{10}\,M_\odot$. Note that the name *dark disk* refers to the low luminosity of the stars within, and has no relation to the dark matter discussed below.

Metallicity and Age

The easiest way to determine which disk a star is in is to look at its metallicity (the amount of metal in the star.) This can be measured by looking at the strength of various emission spectra, since elements like iron have very distinct emission lines. These heavy elements are only formed when large stars reach the end of their life, and so their concentration has steadily increased over time as more large stars form and die. Old stars dating back to the formation of the galaxy (like those of the thick disk) tend to have very "clean" emission spectra, with very little other than hydrogen and helium, while younger stars have strong magnesium and iron lines.

Metallicity isn't a perfect way to measure the age of a star. Metal concentrations vary widely across both space and time, and two stars forming at the same time may have very different metallicities. However, when looking at a large and statistically representative sample of stars, a high metallicity indicates a younger age (?, pp. 885).

The Stellar Halo

The disks extend to about 25 kpc from the galactic center, and contain practically all the mass within that radius. Past that point, however, stars are distributed far more chaotically. The thin plane disappears, and stellar orbits become more spherically distributed. The composition of individual stars in the halo is similar to those in the thick disk—old, dim, and unchanging. However, the halo is also home to many globular clusters, groups of tens or hundreds of thousands of stars that act like small galaxies in their own right. These clusters continue to create new stars, and most of the light coming from the halo is from globular clusters.

Measurements of stellar velocities have long predicted that the mass of the halo dominates the mass of the galaxy—about 95% of galactic mass must be in the halo for the observed velocity curves to hold. Since the halo was known not to be made up of gas and dust (otherwise its extinctive properties would be easy to measure), astronomers long though that the halo contained vast numbers of dense, dark bodies, such as lone planets, dim stars, black holes, and neutron stars—referred to as MACHOs, or Massive Astrophysical Compact Halo Object. Gravitational lensing observations disproved this theory, however, when they capped the mass percentage of MACHOs at about

15% of the mass of the galaxy. The remainder of the mass was some strange material, spherically distributed throughout the universe, that interacts with nothing but gravity.

Dark Matter

As that strange material was further studied (as much as it is possible to study something so elusive), more of its properties were discovered. This *dark matter* seems to be made up of WIMPS (Weakly Interacting Massive Particles), which only interact via gravity and the weak force. Being collisionless (since it does not interact with the electromagnetic force), it does not coalesce and form clouds and stars like regular matter, which is how it has maintained its spherical distribution for so long.

This dark matter has a density of

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

where ρ_0 is the maximum density and a is proportional to the size of the galaxy. This function is referred to as the NFW profile (named after its creators, Navarro, Frank, and White), and is highly accurate for all observed spiral galaxies. For the Milky Way, the simplified equation

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

also returns satisfactory results. However, both these equations appear to suffer from a major problem. If we use them to calculate the total mass of the dark matter in a galaxy, integrating from 0 to ∞, it appears that a galaxy has an infinite mass, as follows:

$$\int_0^\infty \rho(r) 4\pi r^2 \, \mathrm{d}r = \infty \tag{3}$$

Since we know this is not true, there must be a cutoff point where the law no longer holds. As it turns out, in local groups like our own, the dark matter halos are so large that they border one another—providing a natural cutoff point and a solution to our problem (?, pp. 196).

Coordinate Systems

To identify and keep track of objects in the sky, we need to create a coordinate system. A number of natural possibilities spring to mind. The three most useful are detailed below.

Equatorial Coordinates

An equatorial coordinate has two components—a *right ascension*, and a *declination*. To find this coordinate, find the location on the Earth's surface where the object of interest is directly overhead, in the very middle of the sky. The right ascension is the angle between the nearest point on the equator and the vernal equinox (which is defined to be the point where the equator crosses the ecliptic). The declination is then the angle between the current point and that nearest equatorial point. This coordinate system is very ancient, dating back millennia. Figure ?? shows the process of finding such a coordinate.

Galactic Coordinates

This coordinate system is similar to the equatorial system, but the declination is measured from the galactic plane instead of the equatorial plane. The right ascension is then defined to be the angle between the projection of the body of interest on the galactic plane and the vector between the Sun and galactic center. Figure ?? shows this process in more detail. Although harder to calculate from the surface of the earth, this system is more natural when studying bodies traveling close to the Sun. The standard notation and conversions between the two systems are given below:

Measurement	Equatorial Notation	Galactic Notation
Right Ascension	δ	b
Declination	α	ℓ

$$\sin b = \sin \delta_{NGP} \sin \delta + \cos \delta_{NGP} \cos \delta \cos(\alpha - \alpha_{NGP}) \tag{4}$$

$$\sin \delta = \sin \delta_{NGP} \sin b + \cos \delta_{NGP} \cos b \cos \ell_{NCP} - \ell \tag{5}$$

Where $\delta_{NGP} = 27°7'41.7''$ and $\ell_{NCP} = 123°55'55.2''$, as determined by the tilt of the earth and its orientation relative to the galactic plane. These equations can also be inverted to find ℓ and α (?, pp. 900).

Cylindrical Coordinates

The two coordinate systems discussed earlier are well-suited for positional observations from the Earth at a given point in time, but perform poorly over long timeframes. As the sun travels around its orbit, the coordinates of an object change even if it has not moved at all—not an ideal behavior from a reference system. The cylindrical coordinate system, with a reference point at the center of the universe solves these concerns. Unlike

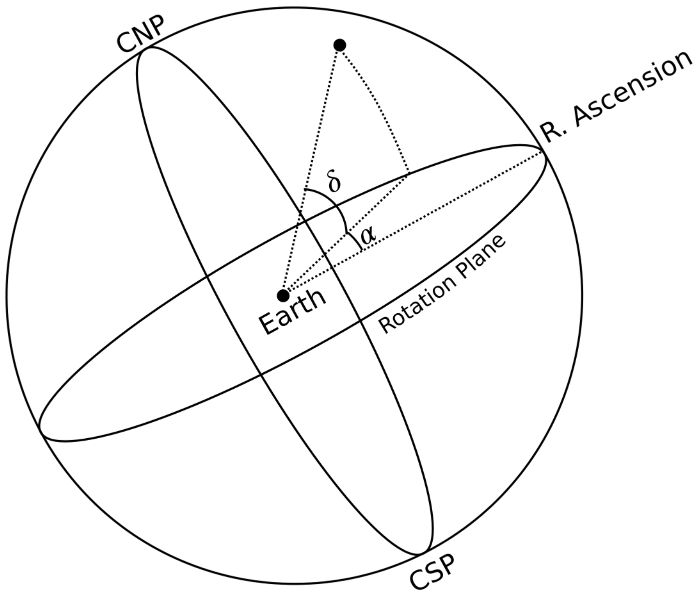

Figure 2: This figure shows the process of finding an equatorial coordinate. CNP and CSP refer to the celestial north and south poles, respectively. R. Ascension refers to the right ascending node, where the Earth's rotational plane crosses its orbital plane.

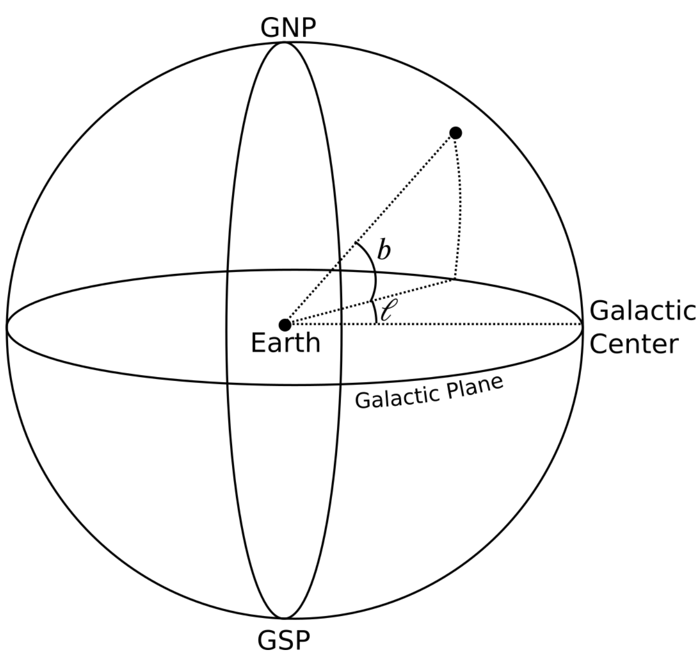

Figure 3: This figure shows the process of finding a galactic coordinate. GNP and GSP refer to the galactic north and south poles, respectively.

the others, it is a three-component coordinate system: R is the radial distance along the plane, increasing outwards; θ is the angular position, and increases in the direction of rotation; and z is the height above (or below) the plane, increasing towards the north, as shown in figure ??. These coordinates also produce a natural (and commonly used) velocity coordinate system, as described below (?):

$$\Pi \equiv \frac{dR}{dt} \qquad \Theta \equiv R\frac{d\theta}{dt} \qquad Z \equiv \frac{dz}{dt} \tag{6}$$

Note that, because the galaxy rotates clockwise when viewed from the north pole, this is a left-handed coordinate system rather than a more-standard right-hand system. Fortunately, we do not need to take any cross-products, so this does not cause any problems.

Local Standard of Rest

Now that we have a definition of the cylindrical velocity coordinates, we can define the Local Standard of Rest (LSR), an important concept in astrophysics. The LSR at a given moment is defined to be the velocity of a body in the sun's position, in a perfectly circular and on-plane orbit—which in practice means the Θ-component of the sun's velocity, with Π and Z set to zero.

The velocity of a nearby star relative to the LSR is known as its *peculiar velocity*, and approximates the velocity of that star relative to the sun. Its coordinates are typically designated (u, v, w), where

$$u = \Pi - \Pi_{LSR} \tag{7}$$
$$v = \Theta - \Theta_{LSR} \tag{8}$$
$$w = Z - Z_{LSR} \tag{9}$$

The average peculiar velocity for stars in the solar neighborhood is approximately zero, since the universe is mostly axisymmetric. However, individual peculiar velocities vary widely, with young main-sequence stars like the sun having low velocities and old, metal-poor red dwarfs having higher velocities. As discussed earlier, this is due to the additional orbital perturbations experienced by old stars, especially during the chaotic period of formation $9\,$Gyr ago.

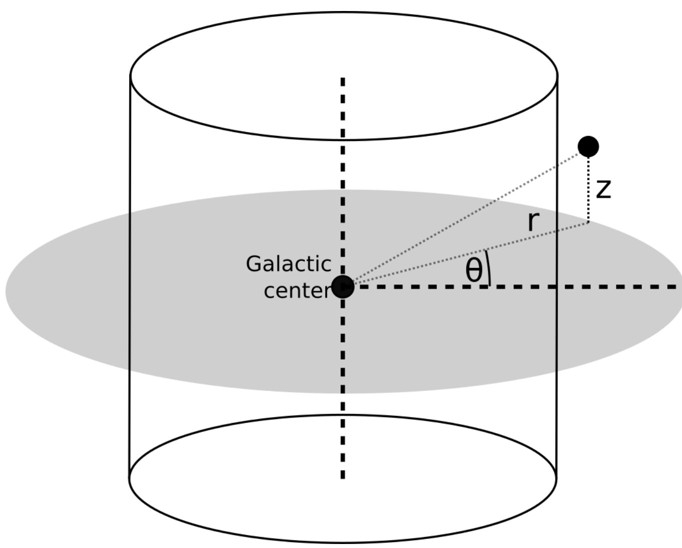

Figure 4: This figures shows the process of finding a cylindrical coordinate.

Chapter 1

Computational Optimizations

1.1 Hardware-based Vectors

Traditionally, to do vector-based computations on the CPU, a programmer would have to create their own definition of a vector through a class or a struct. They would be responsible for implementing all needed operations in software, and these operations would be run like any other code—serially, with the normal overhead of a function call. However, recent x86-64 CPUs (anything made by Intel in the last decade) have a SIMD coprocessor that allows for native, hardware-accelerated vector computation.

Listing 1.1 : An nearly-equivalent declaration

```
typedef struct {
    double x;
    double y;
    double z;
} vec3;
```

1.1.1 SIMD

SIMD, short for *single-instruction, multiple-data*, is the simplest form of parallelism. A special piece of hardware executes the same command on more than one piece of data at a time, making it very useful for tasks like array processing or graphics rendering. For tasks like array processing, the SIMD optimizations are often generated by the compiler, with no programmer input necessary (other than setting the correct flags). However, they also work well for vector operations (which often involve performing the same operation on each scalar element). Since the compiler cannot reliably determine if a certain datatype is meant to represent a vector, this functionality requires an explicit declaration, as seen in ??

Listing 1.2 : The declaration for EXT vector types

```
typedef double vec3 __attribute__((ext_vector_type(3)));
```

1.1.2 Properties of Hardware-based Vectors

Although these types are meant to serve as vectors, many useful vector operations—absolute value, the cross and dot products, and similar—are not implemented. All the C operators can be used on vectors (even if they don't make much sense). For example, as seen in listing ??, the > operator is defined for vector types, and returns a vector of 3 `bool`s. All these operators return a vector themselves, so any operators returning a scalar must be implemented in software (such as seen in listing ??).

Listing 1.3 : Some operations aren't especially useful

```
(vec3){0,5,8} > (vec3){3,4,8} == (vec3){false, true, false};
```

Listing 1.4 : Implementing the absolute value function in software

```
#include<math.h>
inline double vabs (vec3 v) {
    return sqrt(v.x*v.x + v.y*v.y + v.z*v.z);
}
```

1.1.3 Performance Advantages

The best way to measure the performance of SIMD vectors is to write a simple program and time it, both with and without hardware vectors enabled. However, looking at the assembly output of gcc can also provide an idea of relative performance. For example, listing ?? shows the code for a performance critical loop that calculates force, compiled with the option `-Ofast`—maximum optimization, but with software-based vectors.

Listing ??, on the other hand, has been compiled with vector support—with options `-Ofast -march=native` telling the compiler to take advantage of any and all CPU features that could be useful. It is about two-thirds as long as the fastest possible software version, which generally corresponds to a 33% performance increase—a huge boon in code that runs tens of millions of times per second. Rows and rows of near-identical `add` and `mov` calls are replaced by a single specialized call to `vfmadd231sd`, which performs all those additions simultaneously.

Listing 1.5 : The optimized assembly for a critical loop, without SIMD vector support

```
LBB2_27:                                 ##   in Loop: Header=BB2_23 Depth=1
        movapd   144(%rsi), %xmm2
        movapd   160(%rsi), %xmm3
        movsd    128(%rsi), %xmm4        ## xmm4 = mem[0],zero
        subpd    48(%rsp), %xmm3         ## 16-byte Folded Reload
        subpd    64(%rsp), %xmm2         ## 16-byte Folded Reload
        movapd   %xmm2, %xmm5
        mulsd    %xmm5, %xmm5
        movapd   %xmm2, %xmm6
        shufpd   $1, %xmm6, %xmm6        ## xmm6 = xmm6[1,0]
        mulsd    %xmm6, %xmm6
        movapd   %xmm3, %xmm7
        mulsd    %xmm7, %xmm7
        addsd    %xmm5, %xmm7
        addsd    %xmm6, %xmm7
        xorps    %xmm5, %xmm5
        sqrtsd   %xmm7, %xmm5
        addsd    %xmm10, %xmm7
        divsd    %xmm7, %xmm4
        movddup  %xmm4, %xmm6            ## xmm6 = xmm4[0,0]
        movddup  %xmm5, %xmm7            ## xmm7 = xmm5[0,0]
        divsd    %xmm5, %xmm3
        divpd    %xmm7, %xmm2
        mulpd    %xmm6, %xmm2
        mulpd    %xmm4, %xmm3
LBB2_28:                                 ##   in Loop: Header=BB2_23 Depth=1
        addpd    %xmm3, %xmm0
        addpd    %xmm2, %xmm1
```

Listing 1.6 : The same code, with SIMD vectors

```
LBB2_25:                                 ##   in Loop: Header=BB2_21 Depth=1
        vmovupd 160(%rsi), %ymm2
        vmovsd   144(%rsi), %xmm3        ## xmm3 = mem[0],zero
        vsubpd   (%rsp), %ymm2, %ymm2    ## 32-byte Folded Reload
        vpermilpd       $1, %xmm2, %xmm4 ## xmm4 = xmm2[1,0]
        vmulsd   %xmm4, %xmm4, %xmm4
        vfmadd231sd     %xmm2, %xmm2, %xmm4
        vextractf128    $1, %ymm2, %xmm5
        vfmadd231sd     %xmm5, %xmm5, %xmm4
        vaddsd   %xmm0, %xmm4, %xmm6
        vdivsd   %xmm6, %xmm3, %xmm3
        vbroadcastsd    %xmm3, %ymm3
        vsqrtsd %xmm4, %xmm4, %xmm4
        vmovddup        %xmm4, %xmm6     ## xmm6 = xmm4[0,0]
        vdivpd   %xmm6, %xmm2, %xmm2
        vdivsd   %xmm4, %xmm5, %xmm4
        vinsertf128     $1, %xmm4, %ymm2, %ymm2
        vfmadd231pd     %ymm2, %ymm3, %ymm8
```

1.1.4 Disadvantages

There is a good reason SIMD support is not enabled by default. Because its instructions are not part of the `x86-64` standard, various chipset manufacturers may implement the feeature differently (or not enable it at all.) The `-march=native` flag we passed to the compiler voids any guarantee that the resulting binary be able to run on any computer running the same operating system and instruction set. We could build the non-SIMD code in 32-bit mode and load the resulting binary onto a computer from the early 90s, and it would (probably) run. Trying the same with the SIMD-enabled binary may throw an error similar to the one seen in ??, especially on old or lower-end CPUs.

Listing 1.7 : An error thrown by an unsupported instruction on OS X

```
45584 illegal hardware instruction  ./a.out
```

1.2 The Tree Algorithm

When trying to calculate the behavior of a body in a field, the simplest way is to calculate the force on that body from every other body that makes up that field. This strategy is easy to implement—the only relevant equations are from Physics 100. However, as the number of bodies to be considered grows, the required number of computations skyrockets, becoming untenable on even the most powerful computers. With modern astrophysical simulations often including over a million bodies, this naïve approach is unworkable on all but the largest computers. In technical terms, this algorithm is said to have a *time efficiency* of $O(n^2)$, which means that, as the number of bodies doubles, the number of calculations quadruples.

We clearly need to find a way to reduce the number of calculations. Using the system in figure ?? as an example, we see that each body requires 79 calculations—a good baseline to work from. Recall from Physics 100 that, as one moves further away, the forces exerted by two identical bodies near each other converge. It follows that, if distant collections of bodies could be replaced with singular large bodies, the number of calculations could be sharply reduced without significantly impacting accuracy. Figure ?? shows an example of this, replacing a cluster of 10 distant bodies and reducing the number of calculations by about 15%—with only one replacement.

Our next task is to find an algorithm to do all these merges and accuracy judgements as quickly as possible. Representing space as a kind of tree structure makes this easy to do. To build this tree, space is divided into nodes along each axis (so a 2-dimensional node will have 4 children, while a 3-dimensional one will have 8). Each of these child nodes is then bisected in the same manner, and the process repeats recursively until

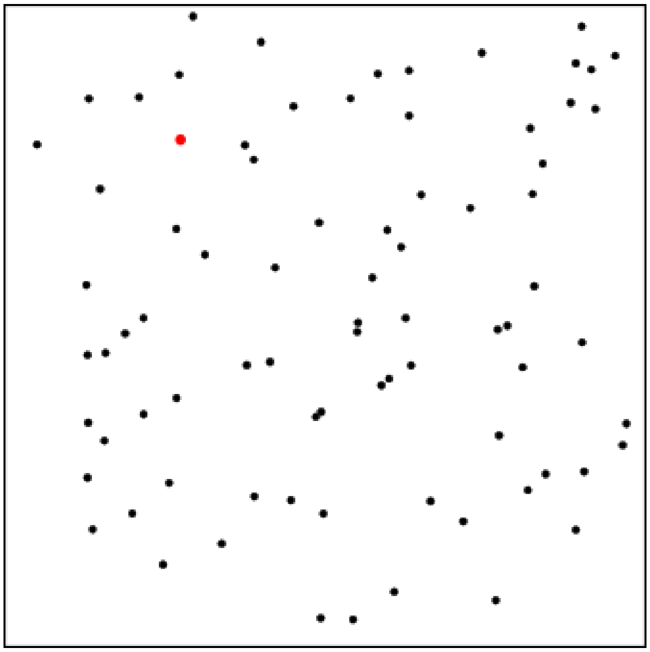

Figure 1.1: We want to calculate the force on the red dot in this system, which involves 79 calculations (since there are 80 bodies). These 79 calculations must be repeated for every other body, for a total of $79 * 80 = 6320$ calculations.

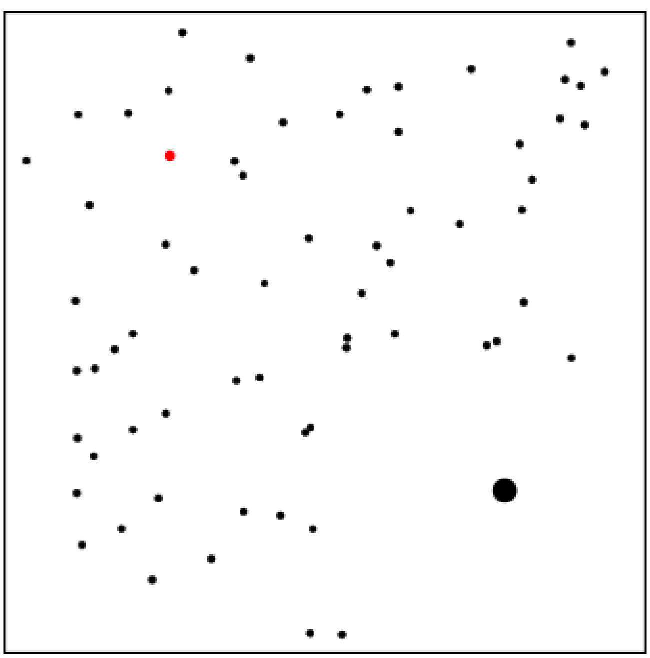

Figure 1.2: The force on the red body in this system is similar to the one seen earlier. However, the number of calculations has been significantly reduced, because about 10 bodies have been replaced with one large body.

each bottom-level node has 0 or 1 bodies in it. The resulting structure is referred to as a quadtree (in 2D space) or an octotree (in 3D space). The first two levels of a quadtree can be seen in figure ??.

Then, to create a system that approximates the original, we simply have to "walk" through the tree, deciding whether each node is a suitably accurate approximation for the bodies contained within, or whether its children should be considered individually. This decision is based on the mass of the node and its distance from the body of interest, as well as an "accuracy factor" that controls the allowable level of error. Eventually, a list of nodes is created that contains each body exactly once, and the net force can be found by summing up the forces from each node. Figure

The algorithm discussed here has a time complexity of $O(n \log n)$ in the average case, and $O(n^2)$ in the very worst case. This means that it will practically always beat the naïve algorithm for systems of more than a few bodies, and becomes very valuable for large simulations. The creationand analysis of the tree does add some overhead, but a well written implementation will rarely take longer than a few dozen force calculations—while eliminating hundreds or even thousands.

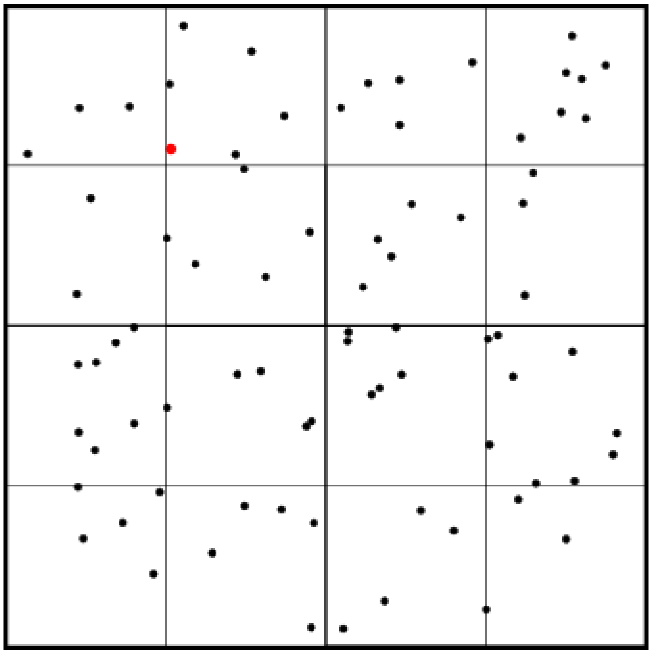

Figure 1.3: The quadtree after two levels of bisections. Note the 4 child nodes of each node, and the ease of scaling this structure to an arbitrary dimension.

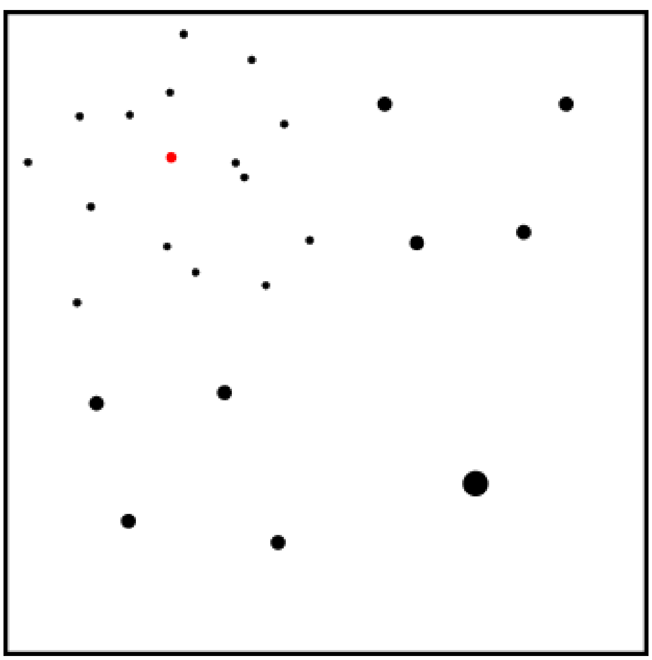

Figure 1.4: This system is a good approximation of the one considered early, and only requires 24 calculations—nearly 75% less than before. Of course a real algorithm would be far more rigorous, and likely find even more opportunities to merge bodies.

Conclusion

The process by which the structure and dynamics of the MW were discovered was by no means trivial: Linblad [ref.] was a hero along with the other pioneers. Buried in the galaxy they had some advantages compared to understanding, say, the Andromeda, but being in the disk causes huge difficulties, not the least of which is the dust.

4.1 More info

And here's some other random info: the first paragraph after a chapter title or section head *shouldn't be* indented, because indents are to tell the reader that you're starting a new paragraph. Since that's obvious after a chapter or section title, proper typesetting doesn't add an indent there.

Appendices

Appendix A

A.1 The Hubble Classification System

The Hubble Classification System, or Hubble Sequence, is the most commonly used scheme for classifying galaxies. Nicknamed *The Fork*, it goes from elliptical galaxies on the left, to the two kinds (barred and unbarred) of spiral galaxies on the right. See fig. ??

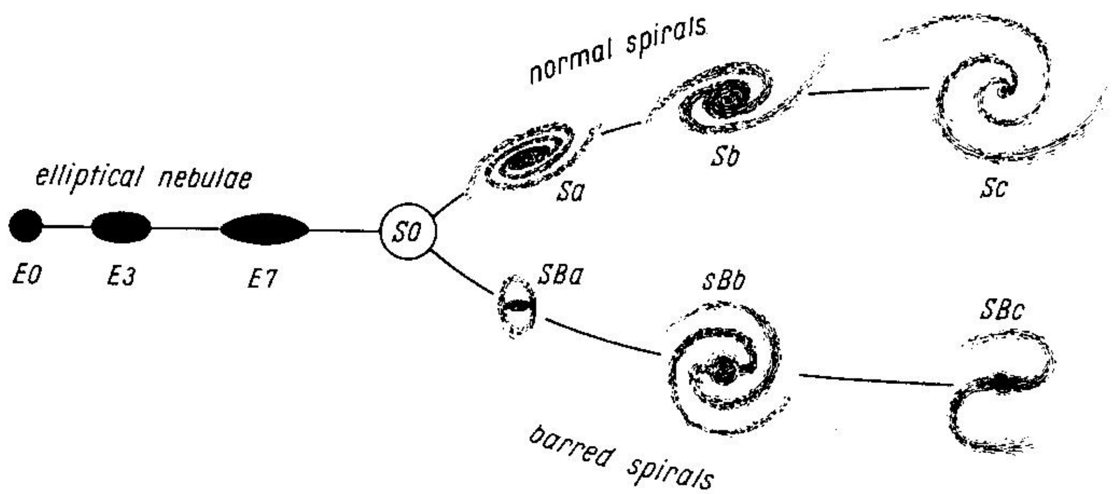

Figure A.1: A diagram of the Hubble tuning fork. *Image courtesy Allan Sandage/CalTech*