DYNAMIC MADE-TO-MEASURE:
A METHOD OF MAKING DYNAMICALLY
SELF-CONSISTENT TRIAXIAL DARK MATTER HALOS

by

NATHAN JAMES DEG

A thesis submitted to the
Department of Physics, Engineering Physics, and Astronomy
in conformity with the requirements for
the degree of Master of Science

Queen’s University
Kingston, Ontario, Canada
April 2010

Copyright © Nathan James Deg, 2010
Abstract

In this thesis we modify the Made-To-Measure (M2M) algorithm to be dynamically self-consistent and apply it to the problem of generating equilibrium collisionless systems with non-spherical halos. Our M2M algorithm systematically adjusts the masses of particles in a system slowly, keeping the system in equilibrium. The adjustments are performed according to some given constraints and proceed until pseudo-observations of the system match the constraints. We use this algorithm to generate isolated triaxial dark matter halos and disk-halo systems with prolate halos. The isolated triaxial dark-matter halo simulations provide a test for the algorithm. These tests show that our algorithm can generate equilibrium collisionless systems with non-spherical halos, but we also find that our algorithm requires a large amount of computational time to converge to the final target system. The disk-halo simulations show that prolate halos modify the morphology and velocity profile of dark matter dominated disks that cause errors in the measurement of the inclination and understanding the rotation curve. As a result of these errors, a mass estimate from the observed rotation curve of a disk in a prolate halo will depend on the observers position relative to the disk. The mass estimates from the same disk observed at different positions may vary by up to a factor of three.
Acknowledgments

Thank you to Larry, Pascal, astro grads, family, friends, the Universe.
Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>ii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>iii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Common Acronyms</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 1:</td>
<td></td>
</tr>
<tr>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>Chapter 2:</td>
<td></td>
</tr>
<tr>
<td>Literature Review</td>
<td>4</td>
</tr>
<tr>
<td>2.1 Schwarzschild Method</td>
<td>5</td>
</tr>
<tr>
<td>2.2 Iterative Method</td>
<td>7</td>
</tr>
<tr>
<td>2.3 M2M</td>
<td>9</td>
</tr>
<tr>
<td>2.4 Cosmological Simulations</td>
<td>23</td>
</tr>
<tr>
<td>Chapter 3:</td>
<td></td>
</tr>
<tr>
<td>Self-Consistent M2M</td>
<td>25</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>3.1</td>
<td>Self-Consistency</td>
</tr>
<tr>
<td>3.2</td>
<td>Basic Algorithm</td>
</tr>
<tr>
<td>3.3</td>
<td>Kernels</td>
</tr>
<tr>
<td>3.4</td>
<td>Grid</td>
</tr>
<tr>
<td>3.5</td>
<td>Momentum Correction</td>
</tr>
<tr>
<td>3.6</td>
<td>Adiabatic Evolution</td>
</tr>
<tr>
<td>3.7</td>
<td>Turning M2M on and Off</td>
</tr>
<tr>
<td>3.8</td>
<td>Implementation</td>
</tr>
<tr>
<td>Chapter 4:</td>
<td><a href="#">Isolated Halos</a></td>
</tr>
<tr>
<td>4.1</td>
<td>The Initial and Target System</td>
</tr>
<tr>
<td>4.2</td>
<td>Analysis</td>
</tr>
<tr>
<td>4.3</td>
<td>Particle Number</td>
</tr>
<tr>
<td>4.4</td>
<td>Form of $\epsilon$</td>
</tr>
<tr>
<td>4.5</td>
<td>Entropy</td>
</tr>
<tr>
<td>4.6</td>
<td>Final Halo Properties</td>
</tr>
<tr>
<td>Chapter 5:</td>
<td><a href="#">Disk-Halo Systems</a></td>
</tr>
<tr>
<td>5.1</td>
<td>Motivation</td>
</tr>
<tr>
<td>5.2</td>
<td>Initial System and Target</td>
</tr>
<tr>
<td>5.3</td>
<td>Extra Analysis Tools</td>
</tr>
<tr>
<td>5.4</td>
<td>Results</td>
</tr>
</tbody>
</table>
Chapter 6:

Discussion and Conclusion ........................................ 87

Bibliography .......................................................... 90
List of Figures

3.1 A graphical representation of the transformation from $R_1$ to $R_2'$. . . . . . . 31

4.1 A comparison of the initial numerical density to the analytic density. . . . . 45

4.2 The effect of particle number on M2M. . . . . . . . . . . . . . . . . . . . . 49

4.3 The axis ratios as a function of $r$ for three simulations in the particle number tests. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 51

4.4 The average mass per particle for three different simulations in the particle number tests. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 53

4.5 The form of $\epsilon_i'$ as a function of $r$. . . . . . . . . . . . . . . . . . . . . 55

4.6 The effect of the form of $\epsilon_i$ on M2M. . . . . . . . . . . . . . . . . . . . . 56

4.7 The axis ratios as a function of $r$ for two simulations in the $\epsilon$ form tests. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 57

4.8 The mass dispersion from the pseudo-entropy tests. . . . . . . . . . . . . . . 61

4.9 The effect of the pseudo-entropy on M2M. . . . . . . . . . . . . . . . . . . . . 62

4.10 The properties of the final halo. . . . . . . . . . . . . . . . . . . . . . . . . . 64

4.11 The density profile of the final halo. . . . . . . . . . . . . . . . . . . . . . . 65

4.12 The axis ratios as a function of $r$ for the best halo. . . . . . . . . . . . . 66

4.13 The mass dispersion of the final halo. . . . . . . . . . . . . . . . . . . . . . . 67

4.14 The anisotropy profile of the final halo. . . . . . . . . . . . . . . . . . . . . 68

5.1 Rotation curve of the initial disk-halo model. . . . . . . . . . . . . . . . . . . 74
5.2 The analytic rotation curves compared to the theoretical inner slopes.

5.3 The analytic rotation curves compared to the numerical rotation curves.

5.4 That axis ratios of the disk and halo as a function of time.

5.5 The orientation of the disk relative to the halo.

5.6 A face-on contour plot of the disk in the 1 : 0.8 : 0.8 halo system.

5.7 Contour maps of the projected images of the disk in the 1 : 0.8 : 0.8 system.

5.8 The inclination correction as a function of the viewing angle.

5.9 The rotation curves of the two final disks compared to the model rotation curves.

5.10 The inclination corrected rotation curves of the disk in the 1 : 0.8 : 0.8 halo.

5.11 The mass estimate from the observed rotation curves.
List of Common Acronyms

• M2M : Made-To-Measure

• ST : Syer and Tremaine

• NMAGIC : The name of the M2M algorithm developed in de Lorenzi et al. (2007)

• DM : Dark Matter

• LSB : Low-Surface-Brightness galaxy

• LOS : Line-of-sight

• LOSVD : Line-of-sight velocity dispersion

• CBE : Collisionless Boltzmann Equation

• DNC : The N-body code used to evolve the particles in this work

• GalactICS : A program to make initial, axisymmetric systems (Widrow et al. (2008))

• NFW : Navarro, Frenk, and White density profile, described in Navarro et al. (1996)
Chapter 1

Introduction

Cosmological simulations suggest that dark matter (DM) halos are non-spherical in shape (Frenk (1988), Warren et al. (1992), Jing and Suto (2002), Allgood et al. (2006)). These simulations have studied the structure, density profile, and morphology of the halos extensively. In addition, there are studies of disk formation in cosmological simulations (Abadi et al. (2003), Abadi et al. (2009)) which examine the effect that disks have on the halos and vice-versa. These studies are usually performed by examining the difference between cosmological simulations consisting of only dark matter and ones with dark matter and baryonic physics. However, there are few non-cosmological studies on the effect of a non-spherical halo on a disk. Such studies would provide controlled numerical experiments that allow the effects of the halo on the disk to be isolated and investigated. The reason that these studies have not been performed in great detail is that it is very difficult to generate isolated disk-halo systems with non-spherical halos, as most methods of generating multi-component galaxies rely on the analytic distribution functions, and we currently do not know of many analytic distribution functions for non-spherical halos.

One approach to this problem is to use methods of generating collisionless systems
with specified constraints to create equilibrium systems with non-spherical halos. These methods include orbital Schwarzschild methods, particle-based methods, and solutions of the Jeans Equations. One of the most interesting, and seemingly most general, methods is the M2M algorithm developed by Syer and Tremaine (1996). This method takes an initial N-body system and adjusts the weights of various particles in order to match some given constraints. If the constraints consist of a non-spherical density distribution, then the M2M algorithm should be able to evolve an initially spherical system with a known distribution function to a triaxial system with a density distribution that matches the constraints.

However, the M2M algorithm requires that the potential of the system is known **a priori**, and, for the disk-halo systems that we are interested in, the potential is unknown. Thus, we must first develop a version of M2M that allows the potential to change. If we can develop such an algorithm, we can begin with an equilibrium system that has an initially spherical halo and slowly evolve it to an equilibrium system with a non-spherical halo.

The development of a self-consistent M2M algorithm (one in which the potential is generated solely by the particles making up the system) is the major goal of this thesis. We will first review methods of generating models with some given constraints. In this review, we will also examine the three major implementations of the M2M algorithm and review their development, similarities, and improvements. We will then proceed to develop our self-consistent algorithm. Unfortunately, the development of a self-consistent M2M algorithm gives rise new difficulties, such as momentum conservation, adiabatic evolution, and shocking the system when M2M is turned on and off. We will show the origin of these problems and present our solutions to them.

Following the development of our self-consistent algorithm, we will test our algorithm by generating isolated, triaxial DM halos. We will examine the rate of convergence for
our system and the stability of the final systems. In addition we will briefly investigate the properties of triaxial DM halos.

The main purpose of developing a self-consistent M2M algorithm is to investigate the effects of a non-spherical halo on an embedded disk. As such, once we are satisfied with the workings of our M2M algorithm, we will proceed to generate disk-halo systems with prolate DM halos. We choose to use prolate halos as they are relatively simple and can allow us to investigate the effects a simple change in the morphology of a halo has on the embedded disk. The disk-halo systems we will generate will be DM dominated so that we can see the largest possible effects of the prolate halo. However, it is thought that Low-Surface-Brightness (LSB) galaxies are DM dominated (Bothun et al. (1997), Impey and Bothun (1997)). As a result, any effects that we see in our test system may be occur in observations of LSB’s. Therefore, we will analyze our disks using a simulated observer and we will examine how the observations change with position. These effects may have profound implications for the study of LSBs.
Chapter 2

Literature Review

A common problem is astrophysics is modelling galaxies. Galactic models can reveal interesting facets of their structure and internal dynamics and are necessary to investigate important phenomena like encounters between different galaxies. Galaxies are collisionless systems, and the modelling of galaxies can be thought of as the quest to find a solution to the Collisionless Boltzmann Equation (CBE). The CBE is

\[
\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} - \nabla \Phi \cdot \frac{\partial f}{\partial \mathbf{v}} = 0 ,
\]

(2.1)

where \( f(\mathbf{r}, \mathbf{v}) \) is the distribution function of the system, \( \mathbf{r} \) and \( \mathbf{v} \) are the position and velocity vectors, and \( \Phi \) is the gravitational potential. If the system is in a steady-state, that is, an equilibrium model, then we only need to solve the time-independent CBE:

\[
\mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} - \nabla \Phi \cdot \frac{\partial f}{\partial \mathbf{v}} = 0 ,
\]

(2.2)

There are a number of ways to find solutions to this equation. They include using the Jean’s theorem and the Jeans equations, orbit-based methods, and particle-based methods.

The Jeans Equations are a set of equations derived from the CBE. They relate velocity moments to the density and potential. Thus, if the projected velocity moments are observed
for a system, they can be deprojected and used to find the density and potential of the system.

The Jeans theorem comes in both the weak and strong forms. The weak form says that any distribution function depends on the phase-space coordinates only though the integrals of motion in the given potential and that any function of the integrals will yield a steady state solution to the CBE. The strong form of the Jeans theorem says that a distribution function of any steady-state system in which all the orbits are regular with non-resonant frequencies may be presumed to be a function of only three independent isolating integrals. In short, if three integrals of motion are known for a system, then any distribution function described by the integrals of motion will be a solution to Eq. 2.1. Modelling techniques using the Jean’s theorem take known integrals of motion and use them to construct a distribution function and generate a model of the system. These techniques are very robust and are quite successful in generating models where the integrals of motion are know. However, these techniques can not be applied when the third integral is unknown, which is a common occurrence.

In this chapter we will briefly review the orbit-based Schwarzschild Method and the particle-based Iterative Method. The Schwarzschild Method is the inspiration of M2M and the Iterative Method has a number of similarities to M2M. In addition, we will review the three major implementations of the M2M algorithm by Syer and Tremaine (1996), de Lorenzi et al. (2007) and Dehnen (2009). In addition, we will include a brief discussion of the results of cosmological simulations.

### 2.1 Schwarzschild Method

Schwarzschild (1979) developed one of the first methods to model stellar systems based
CHAPTER 2. LITERATURE REVIEW

on observational constraints. The core ideas of the Schwarzschild Method are the basis of many other methods of modelling systems based on some constraints. The goal of the Schwarzschild Method is to find a set of orbits within a known potential that satisfy some set of observational constraints. In order to do this, a library of orbits are generated that span the available phase space within the potential. The orbit library is stored in an array, $B(I, J)$, where $B$ is the amount of time that a particle in the $I$’th orbit would stay in the $J$’th cell of an observational grid. Each cell is given some observational constraint, $D(J)$, and each orbit is given some weight, $C(I)$. This gives the set of linear equations:

$$\sum_{I=1}^{N} C(I) B(I, J) = D(J).$$

(2.3)

These equations can be solved for $C(I)$, giving the relative occupation of each orbit. Once the weighted orbit library is constructed, it can be studied to see what are the observational effects of a system that satisfies the target constraints. For instance it could be used to find the anisotropy profile for a system with a triaxial density.

Unfortunately, there are some difficulties with the Schwarzschild Method. Firstly, the orbit libraries require a large number of orbits as they must fully sample the phase space. Each of these orbits must be integrated over a significant period of time in order to build up $B(I, J)$. The construction and integration of the orbits is both time-consuming and susceptible to errors, particularly in sampling the phase space.

In addition, the Schwarzschild Method assumes a fixed known potential and finds the structure of test orbits within that potential. However, if the potential is unknown, the Schwarzschild Method cannot be used. In addition, if the orbits are to generate the potential, that is, if the system is to be dynamically self-consistent, the Schwarzschild Method cannot be used.
2.2 Iterative Method

Rodionov and Sotnikova (2006) have developed a new method for generating equilibrium N-body systems that match some prescribed constraints. This method, known as the Iterative Method, has since been refined in Rodionov et al. (2008) and Sotnikova and Rodionov (2008). The basis of the Iterative Method is the idea that any non-equilibrium system will evolve towards virial equilibrium. The way that the Iterative Method uses this idea is rather novel. It takes an initial N-body system with a prescribed mass distribution and arbitrary velocities. This system is evolved in a way that keeps the mass distribution constant on average, which forces the velocity profile to evolve to equilibrium with that mass distribution and match any additional kinematical constraints like the line-of-sight velocity dispersion (LOSVD).

The Iterative Method generates equilibrium systems in three main steps. First, it generates an N-Body system with the target mass distribution and arbitrary velocities. For simplicity, the velocities may be set to zero initially, though there may be cases where the system will converge faster if the system is given a small rotational velocity. Next, the N-body system is evolved for a short time using a standard N-body code. The third major step is to fix the constraints while retaining the memory of the system’s evolution. In simpler language, after the N-body evolution, the evolved velocity dispersion (with any modifications necessary to match any kinematical constraints) is transferred to the initial mass distribution. Once the evolved velocity dispersion is transferred, the system is evolved for a small amount of time again. The evolution and velocity transfer steps are repeated until the velocity dispersion converges to virial equilibrium with the mass distribution.

The simplest case for transferring the velocity dispersion is when there are no kinematical constraints. The algorithm for transferring the velocity dispersion can be found in
Rodionov et al. (2009) and is based on transferring the velocities of individual particles. First, each particle in the evolved system is given a counter, \( n_{\text{used}} \) that is initially set to zero. If a particle in the evolved system is used for a velocity transfer \( n_{\text{used}} \) will increase by one. To transfer the velocities each particle in the new system finds a group of nearest neighbours in the evolved system. This group is then refined to a subgroup consisting of nearby particles with the lowest values of \( n_{\text{used}} \). Then the particle in the subgroup of lowest \( n_{\text{used}} \) that is closest to the particle in the new system will transfer its velocity to the new system’s particle. This will spread out the velocity transfers between the particles in the evolved system, helping to retain all the velocity information of the evolved system.

If kinematical constraints are specified, then the velocities will be modified slightly during the velocity transfer. The exact form of the modification depends on the kinematical constraint, but the process of modification proceeds identically for all possible constraints. First, pseudo-observations are made of the evolved system and these are compared to the constraints. The comparison between the pseudo-observations and the constraints determines the strength of the modification to the velocity. For instance, if the constraint is the radial velocity dispersion, \( \sigma_r(r) \), then pseudo-observations of the evolved system’s radial velocity dispersion, \( \sigma'_r(r) \) would be made. Then the radial velocity transferred would be

\[
v_{r,i} = v'_{r,i} \frac{\sigma'_j}{\sigma'_r},
\]

where \( i \) is the particle and \( j \) is the specific observation, rather than just \( v'_{r,i} \).

An interesting property of the Iterative Method is that it is possible to scale up the model between steps. In the first few iterations, it may be faster to use a coarse N-body realization of the mass distribution. Once the Iterative Method starts getting close to the equilibrium system it is possible to increase \( N \), so as to achieve a greater resolution. This scaling up as the system evolves allows the Iterative Method to converge fairly rapidly towards the target
The Iterative Method is relatively new but it has been used to successfully construct equilibrium models of triaxial systems, multi-component disk galaxies (Rodionov et al. (2009)), anisotropic, spherically symmetric dark matter halos (Sotnikova and Rodionov (2008)), and models of the Milky Way Disk (Rodionov and Orlov (2008)). It is a relatively simple technique but it seems to be a very effective and powerful method of constructing equilibrium systems for a given mass distribution. However, this technique cannot be used for systems where the mass distribution is unknown. This means that it would be very difficult to generate a model for a distant galaxy with an unknown spatial density.

2.3 M2M

The M2M algorithm is a method of constructing N-body systems that satisfy various constraints. It was originally developed in Syer and Tremaine (1996). Later, Bissantz et al. (2004) used the Syer and Tremaine implementation (hereafter referred to as ST) to study the Galactic Bulge. A new implementation of M2M, called NMAGIC, was developed in de Lorenzi et al. (2007). This version includes a few modifications to the original ST algorithm and has been used to generate various spherical and triaxial halos and study the DM halos of actual galaxies. A third, even more recent, implementation of M2M has been developed in Dehnen (2009). This implementation has a number of interesting differences from the ST and NMAGIC implementations and has been used to generate triaxial halos. Each of these three implementations of the M2M algorithm will be discussed in detail in the following sub-sections.
2.3.1 Syer and Tremaine

As Syer and Tremaine (1996) explain, the M2M algorithm is based on the Schwarzschild Method of modelling collisionless systems. The M2M algorithm constructs equilibrium N-body systems that solve the time-independent CBE (Eq. 2.2) and satisfy various prescribed constraints. The M2M algorithm solves the time-independent CBE by taking a set of observational constraints, \( Y_j \), and comparing them to numerical pseudo-observations, \( y_j \), of an N-body system. The comparison of the pseudo-observations to the constraints is used to systematically modify the N-body system to one in which the pseudo-observations match the constraints. The final system will be a numerical realization of a distribution function that satisfies the CBE and has properties that match the constraints.

In general, any observable of a system with a distribution function, \( f \), can be expressed as

\[
Y_j = \int K_j(z) f(z) \, d^6z ,
\]  

(2.5)

where \( K_j \) is a known kernel and \( z = (r, v) \). Observables might include the surface brightness, the space density, and the LOSVD. A number of target observables are given as constraints while the target distribution function is unknown. Instead an initial N-body system is given, consisting of \( N \) particles, each having a weight \( w_i(t) \) The distribution function of an N-body system can be written as

\[
f = \sum_{i}^{N} w_i(t) \delta^6(z - z_i) .
\]  

(2.6)

In this notation, any pseudo-observation of the N-body system is

\[
y_j(t) = \sum_{i=1}^{N} w_i(t) K_{i,j} ,
\]  

(2.7)

where \( K_{i,j} \equiv K_j[(z_i(t)] \).
In order to match the pseudo-observations with the constraints, the weights of the particles are systematically evolved. The evolution of the weights changes the distribution function of the N-body system. Initially Syer and Tremaine (1996) give a weight change evolution of

\[
\frac{dw_i}{dt} = -\epsilon w_i(t) \sum_{j=1}^{M} \frac{K_{i,j}}{Z_j} \Delta_j(t),
\]  

(2.8)

where \( M \) is the number of observables, \( \epsilon \) is a small constant that determines the rate of change, \( Z_j \) is an arbitrary positive factor yet to be determined, \( \Delta_j \) is a measure of how close the pseudo-observation is to the constraint, and the subscript \( i \) indicates the \( i \)th particle. In Syer and Tremaine (1996), \( \Delta_j \) is given by

\[
\Delta_j(t) = \frac{y_j(t) - Y_j}{Y_j}.
\]  

(2.9)

The factor \( K_{i,j}/Z_j \) in Eq. 2.8 can be thought of as the amount that the \( i \)th particle contributes to the \( j \)th observable.

It is necessary to show that the weight evolution of Eq. 2.8 will lead the pseudo-observations to converge with the target constraints. The proof of convergence can be shown by examining how \( \Delta_j \) changes as a function of time:

\[
\frac{d\Delta_j}{dt} = \frac{1}{Y_j} \frac{dy_j}{dt},
\]  

(2.10)

\[
= \frac{1}{Y_j} \frac{d}{dt} \sum_{i=1}^{N} w_i(t) K_{i,j},
\]  

(2.11)

\[
= \frac{1}{Y_j} \left( \sum_{i=1}^{N} K_{i,j} \frac{dw_i}{dt} + \sum_{i=1}^{N} w_i(t) \frac{dK_{i,j}}{dt} \right),
\]  

(2.12)

\[
= \frac{1}{Y_j} \left( \sum_{i=1}^{N} K_{i,j} \left( -\epsilon w_i(t) \sum_{k=1}^{M} \frac{K_{i,k}}{Z_k} \Delta_k \right) + \sum_{i=1}^{N} w_i(t) \frac{dK_{i,j}}{dt} \right),
\]  

(2.13)

\[
= -\epsilon \frac{1}{Y_j} \left( \sum_{i,k} w_i(t) \frac{K_{i,j} K_{i,k}}{Z_k} \Delta_k - \frac{1}{\epsilon} \sum_{i=1}^{N} w_i(t) \frac{dK_{i,j}}{dt} \right).
\]  

(2.14)
We can rewrite Eq. 2.14 in matrix notation as
\[ \frac{d\Delta}{dt} = -\epsilon (A(t) \cdot \Delta + B(t)), \quad (2.15) \]
where the elements of \( A(t) \) are
\[ A_{kj} = \sum_{i=1}^{N} w_i(t) \frac{K_{i,k}K_{i,j}}{Y_jZ_k}, \quad (2.16) \]
and the elements of \( B \) are
\[ B_j = \frac{1}{\epsilon} \sum_{i=1}^{N} w_i(t) \frac{dK_{i,j}}{dt}. \quad (2.17) \]
Far from convergence the \( \Delta_j \) will be much larger than any fluctuations in the kernels making up the \( j \)th observable. Therefore, far from convergence, \( \Delta \) will evolve according to
\[ \frac{d\Delta}{dt} = -\epsilon A(t) \cdot \Delta. \quad (2.18) \]
Since \( A(t) \) is positive definite, \( \Delta \) will evolve towards zero, indicating the pseudo-observations are evolving to match the constraints.

Nearer convergence, \( \Delta \) will become small enough that \( A(t) \cdot \Delta \) will be of the same order as \( B \). This suggests that there is a limit to how well the M2M algorithm converges to the target constraints. However, it is still possible to examine the general behaviour of \( \Delta \) even when we are near convergence. The orbit-averaged change in \( \Delta \) can be written as
\[ \left\langle \frac{d\Delta}{dt} \right\rangle = -\epsilon \left\langle A(t) \cdot \Delta + B(t) \right\rangle, \quad (2.19) \]
where \( \left\langle \cdot \right\rangle \) denotes an orbit-averaged quantity. If many particles make up each observation, the sum of the fluctuations in a given observation should be small and essentially random. Thus \( \left\langle B(t) \right\rangle \approx 0 \), so
\[ \left\langle \frac{d\Delta}{dt} \right\rangle = -\epsilon \left\langle A(t) \cdot \Delta \right\rangle. \quad (2.20) \]
From this we can see that even near convergence when particle noise begins to become important, $\Delta$ will tend to evolve towards zero, although at any given step it may undergo random fluctuations from $\mathbf{B}$. Therefore, it is clear that Eq. 2.8 drives the pseudo-observations to converge with the target constraints.

It is important to note that this proof of convergence is different than the proof shown in Syer and Tremaine (1996). Syer and Tremaine (1996) show that M2M converges by first arguing that, since $\epsilon$ is small, the weight will only change over many orbits. Thus, the weight change equation can be rewritten as

$$\frac{dw_i}{dt} = -\epsilon w_i(t) \sum_{j=1}^{M} \frac{\langle K_{i,j} \rangle}{Z_j} \langle \Delta_j(t) \rangle,$$

where it is assumed that fluctuations in $K_{i,j}$ are not correlated with $\Delta_j$, which is reasonable when the $j$’th observable is made of many particles. In addition it is assumed that the orbit averaged kernels are time-independent. Using this modified weight change equation, Syer and Tremaine (1996) arrive at

$$\frac{d\langle \Delta \rangle}{dt} = -\epsilon \sum_{i,k} w_i(t) \frac{\langle K_{i,j} \rangle \langle K_{i,k} \rangle}{Y_j Z_k} \langle \Delta_k \rangle.$$

Then, since the weights are varying slowly, near convergence, the changes in $\langle \Delta \rangle$ will be dominated by the $\langle \Delta_k \rangle$ terms. Thus, it is possible to replace the varying $w_i$ terms with constant, $w_i^o$, terms. Then, in matrix notation

$$\frac{d\langle \Delta \rangle}{dt} = -\epsilon \langle \mathbf{A}(t) \cdot \Delta \rangle.$$

where the elements of $\mathbf{A}$ are

$$A_{kj} = \sum_{i=1}^{N} w_i^o \frac{\langle K_{i,k} \rangle \langle K_{i,j} \rangle}{Y_j Z_k}.$$

This version of $\mathbf{A}$ is independent with time, which suggests that the system converges in $\sim \epsilon^{-1}$ orbital periods.
Our proof differs from the one presented in [Syer and Tremaine (1996)] for a number of reasons. The first of these is to show that the system will converge without invoking orbit-averaging, which is somewhat confusing. Secondly, it seems the reason for using orbit-averaging in [Syer and Tremaine (1996)] is simply to arrive at a definition of \( A \) that is time-independent. It is this time-independence that allows [Syer and Tremaine (1996)] to assign \( \epsilon \) a physical meaning. In the algorithm we develop in chapter 3, we do not use the same \( \epsilon \) for all particles and we find a slightly different physical meaning for our values of \( \epsilon_i \). Therefore it is not necessary to define \( A \) as being time-independent. Finally, in our algorithm, our definition of \( Z_j \) is not time-independent, and it is correlated with \( \Delta_j \) so we cannot separate the orbit-averaging of \( \Delta \) with the orbit-averaging of \( A \). Our proof of convergence is valid for both our modified algorithm and the ST algorithm and it does not involve a redefinition of Eq. 2.8 that invokes orbit-averaging.

Thus far, \( Z_j \) has been defined as an arbitrary positive quantity. Since it is arbitrary, the ST algorithm simply defines \( Z_j = Y_j \). Rewriting Eq. 2.8 with this definition of \( Z_j \) gives

\[
\frac{dw_i}{dt} = -\epsilon w_i(t) \sum_{j=1}^{M} \frac{K_{i,j}}{Y_j} \Delta_j(t).
\]

Unfortunately this formulation of the weight-change equation has issues with N-body shot noise and the uniqueness of the final system. In our proof of convergence we discussed how the fluctuations in the kernels and individual weights give a limit to how close the system can converge to the targets. These fluctuations arise due to shot noise from the particle number and the shot noise is larger where \( N/M \) is small, as the noise is on the order of \( \sim (M/N)^{1/2} \). The issue of uniqueness arises from there being more particles than observables. One way of thinking about M2M is that it is a method for solving the set of
linear equations given by
\[ \sum_{i=1}^{N} w_i(t) K_{i,j}(t) = Y_j . \] (2.26)

Since \( N \) is greater than \( M \), the system of equations is ill-conditioned and the pseudo-observations may converge to the target constraints while the weights continue to evolve, which is undesirable.

The shot noise can be reduced in two ways. The simple way is to add more particles, as additional particles will reduce the shot noise. However, additional particles will also increase the computational time of the algorithm. Instead, Syer and Tremaine (1996) implement temporal smoothing to lower the particle noise. Temporal smoothing effectively smears a particle backwards along its orbit, thereby reducing the shot noise. A way to think of time smoothing is that it introduces a number of ghost particles that have a half-life of \( t_{1/2} = \ln 2/\alpha \). These ghost particles increase the effective particle number to
\[ N_{eff} = N \frac{t_{1/2}}{\delta t} , \] (2.27)
where \( \delta t \) is the time step.

The smoothing is not performed on a particle-by-particle basis, as that would be computationally quite difficult and time-consuming. Instead the time-smoothed difference between the pseudo-observations and the constraints is defined as:
\[ \tilde{\Delta}_j(t) = \alpha \int_0^\infty \Delta_j(t - \tau)e^{-\alpha \tau}d\tau , \] (2.28)
where \( \alpha \) is small and positive. The time-smoothed difference takes into account both the pseudo-observations made of the particles at their current positions and the observations made at earlier times and is equivalent to smearing every particle backwards along their orbits.
In order to solve the uniqueness problem, Syer and Tremaine (1996) introduce a pseudo-entropy term:

\[ S = - \sum_i w_i(t) \log \left( \frac{w_i(t)}{m_i} \right), \]  

(2.29)

where \( m_i \) are a set of pre-determined prior target weights, called priors. The pseudo-entropy term provides a 'force' that pushes the weights to evolve towards the priors. Typically, these priors are set to equal the initial weights, but, if there is a better guess as to the final weights, the priors can be set to that guess instead. The entropy term is given a strength of \( \mu \) that determines the importance of satisfying the target constraints compared to the importance of having the weights equal the priors. In order to take the entropy into account on a particle by particle basis it is necessary to calculate \( \partial S/\partial w_i \). Then the weight change equation can be rewritten to take both the entropy and the temporal smoothing into account:

\[ \frac{dw_i(t)}{dt} = \epsilon w_i(t) \left[ \mu \frac{\partial S}{\partial w_i} - \sum_{j=1}^{M} K_{i,j} \tilde{\Delta}_j(t) \right]. \]  

(2.30)

Equation 2.30 was derived from the arbitrary initial weight equation by the Eq. 2.8 to account for various issues. However, it is possible to derive Eq. 2.30 by first considering a function \( F \), defined as

\[ F = \mu S - \frac{1}{2} C, \]  

(2.31)

where \( C \) is:

\[ C = \sum_{j=1}^{M} \tilde{\Delta}_j^2. \]  

(2.32)

It is clear from \( F \) that the weight change equation is actually

\[ \frac{dw_i}{dt} = \frac{\partial F}{\partial w_i}, \]  

(2.33)

and that the M2M algorithm is simply the process of maximizing the function \( F \).
The implementation of the ST algorithm is quite simple. The particles are initially advanced by a time-step using some N-body code. Then weighted pseudo-observations are made of the system. These observations are used to adjust the weights. Once this is done an extra renormalization step is performed as Syer and Tremaine (1996) restrict the weights so that the sum of the weights is constant. These three steps are repeated until the pseudo-observations converge with the target constraints.

The ST implementation of M2M is elegant, simple and has been used to study the Milky Way Bulge with limited success by Bissantz et al. (2004). However, this implementation still contains a number of issues. There is an issue with the convergence time-scale being proportional to the orbital time. Since the orbital times are different in different regions of a system, it is difficult to know how long M2M must be run in order for the entire system to converge. There are also numerical difficulties when $Y_j$ is near zero due to the $K_{i,j}/Y_j$ factor in Eq. 2.30. In addition it would be useful to be able to include information on observational errors if the constraints are from real observations. Finally, the ST version of M2M, like the Schwarzschild Method, examines the behaviour of a set of test particles in a given potential. If the potential of the target system is unknown, it is difficult to use the ST implementation of M2M.

2.3.2 NMAGIC

The NMAGIC algorithm was developed by de Lorenzi et al. (2007). The most important improvement in the NMAGIC algorithm is the inclusion of errors on the constraints. The errors are included by replacing $Z_j$ in Eq. 2.8 with $\sigma_j$ where $\sigma_j$ is the error on the constraint $Y_j$. In addition, $\Delta_j$ is also modified to include the error term. The modification is written
as
\[
\Delta_j(t) = \frac{y_j(t) - Y_j}{\sigma_j}.
\]  

(2.34)

Including \(\sigma_j\) is quite useful as it allows for the use of real observational data as constraints and it does not hit a singularity when \(Y_j\) is near zero. Additionally, it gives Eq. 2.31 a more physical meaning as, in this formulation, \(C\) is a time-smoothed \(\chi^2\) statistic. Ultimately, the inclusion of \(\sigma_j\) changes Eq. 2.30 to

\[
\frac{dw_i(t)}{dt} = \epsilon w_i(t) \left[ \mu \frac{\partial S}{\partial w_i} - \sum_{j=1}^{M} \frac{K_{ij}}{\sigma_j} \tilde{\Delta}_j(t) \right].
\]  

(2.35)

Another difference between the NMAGIC algorithm and the original ST algorithm is the inclusion of kinematical constraints. While Syer and Tremaine (1996) mention that it is possible to use any sort of constraint, the examples in Syer and Tremaine (1996) only use density constraints. In de Lorenzi et al. (2007), both density and kinematical constraints are used. The density is expressed in spherical harmonics and the LOSVD is expressed in a truncated Gauss-Hermite series. The moments of these series are used as the constraints.

The NMAGIC algorithm also includes the possibility of using a self-consistent potential. As mentioned in Sec. 2.3.1, the ST algorithm uses test particles in a constant potential. In order for the algorithm to be dynamically self-consistent, the potential must be generated by the particles and as the particle weights change, the potential must also change. In the NMAGIC algorithm, the potential is recalculated using a multipole expansion of the density every 25 M2M steps. This change in potential will then change the orbits, allowing for a more dynamically self-consistent evolution.

The NMAGIC algorithm is robust and has had significant success. It has been used most recently to study the properties of elliptical galaxies, particularly the shape of their dark matter halos in de Lorenzi et al. (2008) and mass-shape-anisotropy degeneracies in
de Lorenzi et al. (2009). However, it is still not completely self-consistent in that it uses a shock and relax method of changing the potential. As well, the use of density and velocity moments as constraints ensures symmetry that may be incorrect when used to model real systems.

### 2.3.3 Dehnen’s Algorithm

A third implementation of M2M was developed recently by Dehnen (2009). This implementation is designed to solve three issues common to both the ST and NMAGIC algorithms. The first issue is that the time-scale for convergence is different at different radii. The second issue is that both the ST and NMAGIC algorithms conserve the particle weights, but there is no explanation of the necessity of this conservation, or how it is implemented. The final issue is the use of time-smoothing in the other algorithms. The proof of convergence is done without temporal smoothing and it is possible that the smoothing can disrupt the convergence properties of M2M.

The time-scale for convergence is different at different radii due to the pseudo-observations converging to the constraints in $\epsilon^{-1}$ orbital times. Unfortunately, the orbital time scales can vary by orders of magnitude within an N-body system. This means that the interior region will converge rapidly while the outer regions will take much longer. This behaviour is not desirable as it makes it difficult to determine how long the algorithm needs to run. Dehnen addresses this problem by introducing a dimensionless time unit, $\tau$, where $\tau$ is given by

$$\tau = t/T_i,$$

and $T_i$ is the period of the particle. Using this dimensionless time allows for the weight change to be written as

$$\frac{dw_i(t)}{d\tau} = \epsilon w_i(t) U_i$$

(2.37)
If the orbital periods of the particles are approximated using the epicycle approximation, it is possible to evolve the weights using Eq. 2.37 instead of Eq. 2.30. In this equation $\epsilon$ is a dimensionless constant that determines the rate of change for each orbit.

To actually implement this, the Dehnen algorithm has the particles orbit in a constant analytic potential and uses an adaptive time-step for the orbit integration. However, it is possible to use an orbit integrator that uses constant time-steps and instead calculates $d\tau_i$ for each particle. Nonetheless, in the algorithm used in Dehnen (2009), an adaptive time-step is used and the step is chosen so that the M2M steps occur at equal $\tau$ for each particle.

The second issue that Dehnen addresses is the weight renormalization found in the ST and the NMAGIC algorithms. In those algorithms the sum of the weights is renormalized to one after each M2M step. Yet, neither Eq. 2.30 or Eq. 2.35 indicate that this renormalization is necessary or should even occur. Nor do the algorithms indicate precisely where, or how, the renormalization occurs. The most logical way of renormalizing is to rescale the weights after the weight adjustment. Instead of performing a renormalization, Dehnen’s algorithm incorporates the weight conservation as a constraint on the weight change calculations.

The incorporation of the weight renormalization constraint starts with the definition of $U_i$ as

$$U_i = \mu \frac{\partial S}{\partial w_i} - \sum_{j=1}^{M} \frac{K_{i,j}}{\sigma_j} \Delta_j.$$  \hspace{1cm} (2.38)

$F$ can be changed to a function $F^*$, which is

$$F^*(w) = F(w) + \ln \left( \sum_{i=1}^{N} w_i \right) - \sum_{i=1}^{N} w_i,$$  \hspace{1cm} (2.40)
where

\[ F(w) = \mu S - \frac{1}{2} C \]  

(2.41)

and

\[ C = \sum_{j=1}^{M} \Delta_j^2. \]  

(2.42)

It is important to note that \( C \) does not use the time-smoothed values of \( \Delta_j \), which is different than in Eq. 2.32. Then, if we use \( F^* \) instead of \( F \) in Eq. 2.39, \( U_i \) will become

\[ U_i = \frac{\partial F^*}{\partial w_i} = \frac{\partial F}{\partial w_i} + \left( \frac{1}{\sum_{k=1}^{N} w_k} - 1 \right). \]  

(2.43)

This modification constrains the weight evolution by pushing the net weight change towards zero. However, it does not force the sum of the weights to be one. As such, if weight conservation is important, a renormalization step will still be required, although, this renormalization step will be much smaller than the renormalization in other algorithms.

The third issue addressed by Dehnen’s algorithm is the use of the time-smoothed observables. The reason why the use of time-smoothed observables is an issue is that the proof of convergence shown in Sec. 2.3.1 examines \( \langle \frac{d\Delta_j}{dt} \rangle \) and not \( \langle \frac{d\tilde{\Delta}_j}{dt} \rangle \). While one might expect that the time-averaged time-smoothed measure of difference to be equal to the time-averaged measure of difference, this is not necessarily true. Additionally, time-smoothing is introduced to reduce the shot noise, which is excellent when trying to model real systems. However, if M2M is being used to generate models for the use in numerical experiments, the shot noise is important as it gives the limit to the trustworthiness of the model. As such, smoothing out the noise is incorrect.

Rather than using time-smoothed pseudo-observations, Dehnen uses a time-smoothed constraint function:

\[ \frac{dU_i}{d\tau} = \eta \left( w_i \frac{\partial F^*}{\partial w_i} - U_i \right), \]  

(2.44)
where $\eta$ is a constant that controls the amount of time-smoothing. Replacing $w_i$ with $\phi_i$ in (2.37) where $\phi_i = \ln w_i$, gives

$$\frac{d\phi_i}{d\tau} = \epsilon U_i.$$

Taking the second derivative of this gives:

$$\frac{d^2\phi_i}{d\tau^2} = \epsilon \frac{dU}{d\tau},$$

$$= \epsilon \eta \left( w_i \frac{\partial F^*}{\partial w_i} - U_i \right).$$

(2.45) (2.46) (2.47)

Finally, $\partial F^*/\partial w_i$ can be replaced with $\partial F^*/\partial \phi_i$ giving

$$\frac{d^2\phi_i}{d\tau^2} = \epsilon \eta \frac{\partial F^*}{\partial \phi_i} - \eta \frac{d\phi_i}{d\tau}.$$

(2.48)

Eq. 2.48 is analogous to a force equation with some frictional term. In this analogy, $\phi_i$ will evolve according to a force generated by a ‘potential’ given by $\epsilon \eta F^*$, with an additional frictional force that has a strength proportional to $\eta$. The frictional term will damp the weight changes due to shot noise once the system has converged to the constraints. Since this formulation of temporal smoothing does not utilize time-smoothed observations, it will not interfere with the proof of convergence shown in Sec. 2.3.1.

Dehnen used this implementation of M2M to successfully generate triaxial halos. However, this implementation is similar to the Schwarzschild Method in that it works on test particles in a known potential. The necessity of knowing the potential before running the algorithm makes it somewhat difficult to apply Dehnen’s implementation of M2M to real observations, as the potential of real galaxies may be unknown.
2.4 Cosmological Simulations

Much of our information on the structure of DM halos arises from cosmological simulations. Therefore we will briefly discuss the methodology and results of cosmological simulations.

The key idea behind cosmological simulations is to take a co-moving volume with a nearly uniform density and evolve it in time. Often the evolution will be purely gravitational, thereby simulating dark matter, however a great deal of effort is now being spent to include baryonic physics in these simulations.

The nearly uniform density has very small, random density perturbations. As the co-moving volume evolves, these perturbations collapse into small halos. These merge to make larger and larger halos in a hierarchical fashion. These halos seem to have a universal density profile (Navarro et al. (1996)) and are non-spherical in shape (Warren et al. (1992)).

One of the most surprising results of cosmological simulations is that the density profile can be characterized by a general formula. This formula, found in Navarro et al. (1996) is

$$\rho(r) \propto \frac{1}{r/s(1 + r/s)^2},$$  \hspace{1cm} (2.49)

where $s$ is some scale radius. This density profile is a simple fitting equation to the observed density measurements. There has been questions about the accuracy of the power of the inner slope, and some evidence that the inner density profiles are cored rather than cuspy. However, this phenomenological profile fits the density of halos in cosmological simulations quite well.

The density profile described by Navarro et al. (1996) assumes spherical symmetry. However, it is important to note that most halos do not appear to be spherical. Instead, as shown in Figure 6 of Warren et al. (1992), the majority of halos are non-spherical in shape, with the majority being more prolate than oblate. This is quite important because,
CHAPTER 2. LITERATURE REVIEW

it suggests that real DM halos are non-spherical, and the non-spherical shape of the halos may have important dynamical effects on embedded galaxies.

In addition, cosmological simulations have been performed that include baryonic physics. These simulations allow for the study of star formation and disk formation (Abadi et al. (2003), Abadi et al. (2009)). These simulations show how the disk forms as a DM halo collapses to its final shape. They also show the effect of interactions and the merging history. One of the major results of these studies is that, as shown in Abadi et al. (2009), the presence of a baryonic disk tends to make the halo more axisymmetric.

However, it is important to note that many of these simulations must include recipes for the baryonic physics (particularly for the feedback due to supernovae). As such, there may be some issues with them. However, they are good approximations and well thought out. Unfortunately, since they include everything, it may be difficult to pinpoint the particular cause of a particular observation. Thus, it is useful to also generate simplified numerical experiments to investigate what the effects of each piece of the simulation are. These include the shape of the halo, merging history, feedback, etc. The effects from each piece of the cosmological simulation may be studied by generating numerical models of the galaxies and investigating each piece separately. That is, to investigate merging through one experiment, merging history in another experiment, and halo shape in yet another experiment. The goal of this thesis is develop a tool that will allow us to perform numerical experiments that test the effect of the halo shape.
Chapter 3

Self-Consistent M2M

In this chapter we will discuss the development and implementation of a dynamically self-consistent version of M2M. There is only one major change necessary to make M2M self-consistent. We will show this change, and then present a number of smaller changes that we use in our algorithm. We will also show the kernels and error terms that we use in this work. Unfortunately, the change to a self-consistent algorithm generates a number of new problems. These problems consist of momentum conservation, large weight changes, and shocking the system when M2M is turned on and off. We will show why these problems arise and our solutions to them. Finally we will present the actual implementation of our dynamically self-consistent M2M algorithm.

3.1 Self-Consistency

As discussed in Sec. 2.3, the original formulation of M2M is not dynamically self-consistent. Instead it uses test particles in a known prior potential. If the potential is unknown, M2M is difficult to use. One solution to the problem is the NMAGIC approach of updating the
potential every 25 M2M steps using a multipole expansion of the observed density. Instead we make our M2M algorithm dynamically self-consistent using a slightly different technique. We set the masses of the particles to be proportional to the weights, and use the particle masses to calculate the forces at every step. If the weight change is slow, then the system will evolve adiabatically and stay in dynamical equilibrium at each time-step. The actual mass of each particle is

\[ m_i(t) = w_i(t) M_{ini}, \]  

(3.1)

where \( M_{ini} \) is the mass of the initial system. This definition of the mass requires that the initial weights sum to one so that the sum of the initial masses equals \( M_{ini} \).

The change to a self-consistent algorithm from a non-self-consistent algorithm is mechanically simple, but the two are conceptually quite different. The non-self-consistent M2M algorithm is very similar to the Schwarzschild Method in that it simply emphasizes certain test particles on particular orbits within a given potential. By contrast, the self-consistent algorithm uses the weighted system of particles to generate the potential and allows the orbital structure of the system to change.

### 3.2 Basic Algorithm

The basic equations used in our implementation are very similar to the NMAGIC equations. Our weight change equation is

\[ \frac{dw_i(t)}{dt} = \epsilon_i w_i(t) \left[ \mu \frac{\partial S}{\partial w_i} - \sum_{j=1}^{M} \frac{K_{i,j}}{\sigma_j} \Delta_j(t) \right]. \]  

(3.2)

There are a few changes between this weight change equation and the NMAGIC weight change equation. Firstly, we do not include any time smoothing as it is conceptually inconsistent with a self-consistent algorithm. Time-smoothing is meant to reduce the shot noise
by averaging each particle backwards along its orbit. But in a self-consistent algorithm the orbits can change with time. That means that the particle will not be smoothed along its current orbit, but rather along its previous orbit. The orbital difference may be small, but, to be completely self-consistent, any smoothing really should be along the particle’s current orbit. Such a time-smoothing would be difficult to implement. Since the time-smoothing is not conceptually self-consistent, we do not use it in our algorithm. The second key difference in our basic weight change equation is that we set $\epsilon$ to be different for each particle. This change in $\epsilon$ is discussed in detail Sec. 3.6. For now, it is sufficient to say that $\epsilon_i$ is small, positive, and constant in time for each particle.

In addition to these obvious changes, there is another hidden difference. In this algorithm $\sigma_j$ is not a given observational error. Rather the error term is calculated at each step based on the particles comprising a given observation. The full details of the error term are given in Sec. 3.3, but, at this point, it is sufficient to note that the error can vary with time.

Finally, we do not renormalize the weights in our M2M algorithm. It is possible that the set of constraints points to a system with a different total mass than the initial system. Therefore it would be inappropriate to renormalize the weights, and a renormalization may hinder convergence.

### 3.3 Kernels

Two key aspects of any implementation of M2M are the constraints and the kernels used to calculate the pseudo-observations. In this work we only use the density as a constraint. The density is calculated on the three-dimensional grid described in Sec. 3.4. The density constraint is calculated by integrating the target density profile over the volume of each grid
cell:

\[ Y_j = \frac{\int_j \rho(x^3) d^3x}{\int_j d^3x}, \quad (3.3) \]

where the \( j \) subscript indicates the \( j \)’th cell.

The pseudo-observations are calculated using

\[ y_j(t) = \sum_{i=1}^{N} \frac{m_i(t)}{V_j} \Theta_{i,j}, \quad (3.4) \]

where \( \Theta_{i,j} \) is equal to one if the particle is in the \( j \)th cell and zero if it is not in that cell, and \( V_j \) is the volume of the cell. Since any observation is given as

\[ y_j(t) = \sum_{i=1}^{N} w_i(t) K_{i,j}, \quad (3.5) \]

and \( m_i(t) = w_i(t) M_{ini} \), then the kernel must be

\[ K_{i,j} = \frac{M_{ini}}{V_j} \Theta_{i,j}. \quad (3.6) \]

Each pseudo-observation has an error, \( \sigma_j \). The typical error for the density calculated from \( n_j \) particles is

\[ \sigma_j = \frac{y_j}{\sqrt{n_j}}, \quad (3.7) \]

where \( n_j \) is the number of particles contributing to the \( j \)th observation. Unfortunately, if \( y_j \) is very small, \( \sigma_j \) will also be very small. This is not a problem except when \( Y_j \) is much larger than \( y_j \). This situation can occur when the \( n_j \) particles making up \( y_j \) all have an exceptionally low weight relative to the average particle weight. Ordinarily, this situation will cause \( \Delta_j \) to become very large and cause the particle weights in the \( j \)’th cell to change rapidly. However, in our self-consistent algorithm, it is necessary to implement a scaling that prevents the maximum weight change from being larger than a factor, \( f_{max} \) (see Sec. 3.6 for details). This scaling decreases all the other weight changes proportionally to the
largest weight change. As such, if one $\Delta_j$ is exceptionally large, the weight changes in all the other cells will be scaled down, which effectively turns off M2M in those cells.

The strange situation of all the particles in a particular cell having a very low weight occurs due to the movement of particles from cell to cell in their orbits. If a particle spends a significant amount of time in a cell where $Y_k$ is much less than $y_k$, those particles will have their weights drastically decreased. Then if they move into the $j$’th cell and the particles that have an average weight move out of that cell, the measurement of $y_j$ will be very small.

In order to deal with this issue, we modify our density error term to

$$\sigma_j = \frac{Y_j}{\sqrt{n_j}}.$$  

(3.8)

This error term will not cause $\Delta_j$ to become exceedingly large when a cell is filled with phantom particles. Near convergence, where $y_j$ is near $Y_j$, this error term will be approximately the same as the standard error. As well, this non-standard error term will represent the noise of the system due to N-body fluctuations. Having an error term that can represent the shot noise is useful for the class of problems that are studied in this work. As such, this error term is somewhat better than a constant error on the constraint, as that error term would not represent the shot noise at any given time.

### 3.4 Grid

All the constraints and pseudo-observations used in our algorithm are grid-based. The system is divided into cells of various sizes, and pseudo-observations are made based on the particles in a particular cell. Similarly, the constraints are constructed by integrating appropriate functions over that cell. The grid used should be appropriate to the constraint used. Some appropriate grids might be projected circles for surface brightness, a long slit
for LOSVD, or spherical shells for density.

In chapters 4 and 5 we use the M2M algorithm to generate prolate and triaxial halos using density constraints. This suggests that the appropriate grid consists of spherical shells that are subdivided into angular bins. However, there is a different grid choice that simplifies the calculation of the constraint, and is slightly simpler to understand.

A triaxial density is a function of three coordinates in physical space, \((r, \phi, \theta)\). However, the density can be written as a function \(r'\), where

\[
r' = \sqrt{\left(\frac{x'}{a}\right)^2 + \left(\frac{y'}{b}\right)^2 + \left(\frac{z'}{c}\right)^2},
\]

and \(a/b\) and \(a/c\) give the target axis ratios for the system. In this notation, \(x'\), \(y'\), and \(z'\) are aligned with the major, semi-major, and minor axis of the system. If the grid is set up in the \(R'\) frame, the calculation of the density constraint is quite simple.

Based on this, the grid that is used for the density constraints consists of a central sphere and a series of radial shells in the \(R'\) frame that is fixed at the origin. Each shell is subdivided into a number of angular bins that are designed so that the volume of each cell in a given shell is the same.

To calculate the pseudo-observations at each step, the particles must be placed in the appropriate cells. To do this we get each particle’s position in the \(R'\) frame. Once the particles are in the \(R'\) frame we use the cell limits to place the particles into the appropriate cells. A transformation matrix, \(T\), can be constructed using \(a, b,\) and \(c\) that will give a particle’s position in the \(R'\) frame. If the target system is aligned to the system’s coordinates the transformation matrix is

\[
T = \begin{pmatrix}
\frac{1}{a} & 0 & 0 \\
0 & \frac{1}{b} & 0 \\
0 & 0 & \frac{1}{c}
\end{pmatrix}.
\]
Then the coordinates of any particle can be transformed from the physical, \( R \), frame to the \( R' \) frame using

\[
TR = R',
\]  

(3.11)

If we are to have the same total mass in the final system as in the initial system than it is necessary to have

\[
abc = 1.
\]  

(3.12)

However, this is not necessary as we do not implement any mass conservation and allow the total mass to change as necessary.

The transformation of the system from the \( R \) frame to the \( R' \) frame is shown in Fig. 3.1. This figure clarifies that a spherical system in the \( R \) frame will be triaxial in the \( R' \) frame and the triaxial grid will be spherical in the \( R' \) frame. Then, since the target density is spherically symmetric in the \( R' \) frame, the M2M algorithm will attempt to sphericalize the triaxial system in the \( R' \) frame. As the system becomes spherical in the primed frame, it will become triaxial in physical space.


3.5 Momentum Correction

A problem that arises in the self-consistent M2M algorithm that does not occur in the non-self-consistent algorithms is the conservation of momentum. In general, any N-body code conserves the overall system momentum \( p \), where \( p \) is given as

\[
p = \sum_{i=1}^{N} m_i v_i.
\]

(3.13)

Then, at each step the momentum is changed by

\[
\delta p = \sum_{i=1}^{N} m_i \delta v_i,
\]

(3.14)

where \( \delta v_i = a_i \delta t \). Thus

\[
\delta p = \sum_{i=1}^{N} m_i a_i \delta t,
\]

(3.15)

which is zero by Newton’s Third Law. However, in M2M, both the masses and velocities are changed. As such, there will first be a momentum adjustment from M2M given by

\[
\delta p_{M2M} = \sum_{i=1}^{N} \delta m_i v_i,
\]

(3.16)

followed by the momentum adjustment from the change in velocities:

\[
\delta p_{Nbody} = \sum_{i=1}^{N} (m_i + \delta m_i) \delta v_i.
\]

(3.17)

While \( \delta p_{Nbody} \) is zero by Newton’s Third Law, \( \delta p_{M2M} \) may not be zero and the total momentum may not be conserved between steps. If the total momentum is not conserved, the system will drift from the origin. Since our observational grid is centered on the origin, this drifting may cause significant problems.

In order to conserve momentum, we will subtract an extra term, \( V \), from the velocity, so that

\[
\delta v_i = a_i \delta t - V.
\]

(3.18)
Then the change in momentum from the velocity advancement will be

$$ \delta p_{N_{\text{body}}} = \sum_{i=1}^{N} (m_i + \delta m_i) (a_i \delta t - V), \quad (3.19) $$

and the total change in momentum for a step is

$$ \delta p = \sum_{i=1}^{N} \delta m_i v_i - V \sum_{i=1}^{N} (m_i + \delta m_i). \quad (3.21) $$

Since $V$ is defined so that the total momentum change is zero, we find that $V$ must be

$$ V = \frac{\sum_{i=1}^{N} \delta m_i v_i}{\sum_{i=1}^{N} (m_i + \delta m_i)}. \quad (3.22) $$

In our work, the initial momentum is zero, so we can add it to the numerator to get

$$ V = \frac{\sum_{i=1}^{N} (m_i + \delta m_i) v_i}{\sum_{i=1}^{N} (m_i + \delta m_i)}, \quad (3.23) $$

or

$$ V = \frac{\sum_{i}^{N} m'_i v_i}{\sum_{i}^{N} m'_i}. \quad (3.24) $$

In other words we must adjust the velocities by the center of mass velocity after M2M in order to conserve momentum.

In addition to the momentum correction, a centre of mass correction is also implemented. Just as the total momentum is not necessarily conserved, the centre of mass does not necessarily stay at the origin when the mass adjustments are made. The math involved in this correction is identical to the momentum correction, as the centre of mass and the total momentum equations have the same form. Therefore, the centre of mass can be corrected by a factor $R$ given by

$$ R = \frac{\sum_{i}^{N} m'_i r_i}{\sum_{i}^{N} m'_i}. \quad (3.25) $$
Both the momentum correction and the centre of mass correction are implemented after each M2M step. The effect of these corrections is to simply keep the system in an inertial frame centred on the centre of mass.

### 3.6 Adiabatic Evolution

In a self-consistent M2M algorithm, both the particle masses and orbits evolve with time. Unfortunately, the mass evolution may be much faster than the orbit evolution. A difference in evolution rates will cause the mass distribution to not correspond to the velocity distribution. To avoid this situation, the mass distribution must evolve at the same rate as the velocity distribution. In other words, the system must evolve adiabatically, meaning that the weight changes in one orbit cannot be larger than the adiabatic limit, $A$:

$$ A \geq \left| \frac{\Delta w_i}{w_i} \right|. \quad (3.26) $$

While the adiabatic limit is not known \textit{a priori}, its existence can be used to scale our system in two distinct ways. The first way is to develop a scaling limit in each M2M step, and the second way is to use the adiabatic limit to get a form of $\epsilon_i$ such that the weight changes will be adiabatic for each particle over a full period.

One of the consequences of the scalings is that $\epsilon$ will not have the same value for each particle. Instead each particle will have its own $\epsilon_i$ that is positive and constant in time. The inclusion of differing $\epsilon_i$’s has already been shown in our weight equation. However, we have not been shown that our weight change equation will still cause the pseudo-observations to converge with the constraints. So, before deriving the scaling factors, it is necessary to show that the use of Eq. 3.2 to modify the weights will still cause convergence.

The form of Eq. 3.2 is very similar to the form of Eq. 2.8. As such, the proof on
convergence will proceed exactly as shown in Sec. 2.3.1 with the one difference that \( \epsilon_i \) cannot be pulled out from the summation. Far from convergence we arrive at a slightly modified Eq. 2.15:

\[
\frac{d\Delta}{dt} = -A \cdot \Delta + B,
\]

where the elements of \( A \) are

\[
A_{kj} = \sum_{i=1}^{N} \epsilon_i w_i(t) \frac{K_{i,k}K_{i,j}}{\sigma_j\sigma_k},
\]

and the elements of \( B \) are still given by Eq. 2.17. We can use the same assumptions about \( B \) being small relative to \( \Delta \). Thus, far from convergence we get

\[
\frac{d\Delta}{dt} = -A \cdot \Delta,
\]

and near convergence we get

\[
\left\langle \frac{d\Delta}{dt} \right\rangle = -\left\langle A \cdot \Delta \right\rangle.
\]

From Eq. 3.29 and Eq. 3.30, it is apparent that \( \Delta \) will converge to zero. This means that the pseudo-observations will still converge to the target constraints.

With the question of the convergence when using \( \epsilon_i \) having been dealt with, it is now possible to return to the discussion of scaling the weight changes so as to remain within the adiabatic limit. In order to stay within the adiabatic limit we wish to find a form of \( \epsilon_i \) that will cause the weight changes over a period to be below this limit. We will begin the search for the correct scaling of \( \epsilon_i \) by rewriting Eq. 3.2 as

\[
\frac{dw_j}{w_j} = \epsilon_j g_j(t) dt,
\]

where

\[
g_j(t) = \mu \frac{\partial S}{\partial w_j} - \sum_{i=1}^{M} \frac{K_{i,j}}{\sigma_i} \Delta_j(t).
\]
This equation is quite useful as it shows that the fractional weight change, $f_j$, in any given
is equal to $\epsilon_j g_j(t)dt$.

Integrating both sides of Eq. 3.31 over one period gives

$$\int_{w_{j,i}}^{w_{j,f}} \frac{dw_j}{w_j} = \int_0^{\tau_j} \epsilon_j g_j(t)dt,$$

(3.33)

$$\ln \left( \frac{w_{j,f}}{w_{j,i}} \right) = \int_0^{\tau_j} \epsilon_j g_j(t)dt,$$

(3.34)

where $\tau_j$ is the period of the particle in question, $w_{j,i}$ is the initial weight, and $w_{j,f}$ is the
final weight.

Since the adiabatic limit, $A$ in Eq. 3.26 is just an unknown constant, we can write the
equivalent expression

$$\ln(A) \geq \left| \ln \left( \frac{w_{j,f}}{w_{j,i}} \right) \right|,$$

(3.35)

which implies that the fastest that the system can be driven towards equilibrium occurs
when the absolute value of the right-hand side of Eq. 3.34 is equal to $\ln(A)$. That is

$$\ln(A) = \left| \left( \int_0^{\tau_j} \epsilon_j g_j(t)dt \right) \right|.$$

(3.36)

The rate of the weight changes is driven by $\epsilon_j$. Since $\epsilon_j$ is constant and positive it can be
pulled out from the integral and Eq. 3.36 can be solved to find the largest $\epsilon_j$ that remains
within the adiabatic limit. This gives

$$\epsilon_j = \frac{\ln(A)}{\left| (\int_0^{\tau_j} g_j(t)dt) \right|}.$$

(3.37)

Unfortunately the integral in Eq. 3.37 is difficult to solve as the form of $g_j(t)$ is un-
known, and implicitly depends on $\epsilon_j$. However, it can be assumed that $g_j(t)$ is a decreasing
function with time since the M2M algorithm converges with time. As the algorithm con-
verges the $\Delta_j$ terms go towards zero, and, as they go to zero, $g_j(t)$ will also go to zero.
This behaviour suggests that, over a significant period of time, the absolute value of \( g_j(t) \) behaves as

\[
|g_j(t)| \propto Ct^{-\alpha},
\]

(3.38)

where \( C \) is some unknown constant. The origin of the power-law description of \( g_j(t) \) is purely phenomenological. We know that \( g_j(t) \) must decrease and this decrease will be asymptotic. A power-law has the same behaviour, so we choose it to model \( g_j(t) \). However, any other decreasing function that asymptotically approaches zero could also be used as a model. Since \( g_j(t) \) is unitless we introduce the constant, \( \delta t \), with units of time, in order to cancel units of the \( t^{-\alpha} \) term. This gives

\[
\left| \left( \int_0^{\tau_j} g_j(t) dt \right) \right| \approx \int_0^{\tau_j} C \left( \frac{t}{\delta t} \right)^{-\alpha} dt,
\]

(3.39)

which can then be used in Eq. 3.37 giving

\[
\epsilon_j = \frac{\ln(A)}{\left( \int_0^{\tau_j} C \left( \frac{t}{\delta t} \right)^{-\alpha} dt \right)}.
\]

(3.40)

The integral in the denominator is solvable, giving

\[
\epsilon_j = \frac{(1 - \alpha) \ln(A)}{C \delta t^{-\alpha} \tau_j^{1-\alpha}},
\]

(3.41)

as long as \( \alpha \neq 1 \). Before proceeding with further study of Eq. 3.41 it is interesting to note that, if \( \alpha = 0 \), the functional form of \( \epsilon_j \) is identical to the form of \( \epsilon_j \) one would get from Dehnen’s Eq. 2.37 if his \( d\tau \) where replaced with \( dt/\tau_j \) and he defined an \( \epsilon_j \) as \( \epsilon/\tau_j \).

While Eq. 3.41 can be used to get a form for \( \epsilon_i \), there are still a lot of unknowns. The first, and perhaps most important, is the value of \( \alpha \). It seems obvious that \( \alpha = 0 \) is incorrect as it suggest that \( g_j(t) \) is constant in time. Since the M2M algorithm converges, \( g_j(t) \) must decrease with time, meaning that \( \alpha > 0 \). That being said, \( \alpha = 0 \) gives a good first estimate as it is guaranteed to be below the adiabatic limit. Since there is no way of knowing exactly
how \( g_j(t) \) actually decreases with time, the only way to choose \( \alpha \) is to try a number of values and find what works. In Sec. 4.4, these trials are performed to find the optimal value for the systems that we generate in this work.

Another complication with Eq. [3.41] is the use of \( \tau_i \). As discussed in Sec. 3.2, as a self-consistent M2M algorithm runs, the orbits of the particles will change. This means that the initial period of a particle is not necessarily the same as the final period of a particle. However, the goal of Eq. [3.41] is to find values of \( \epsilon_i \) that are below the adiabatic limit. Since the largest weight changes occur at the outset, it seems appropriate to use the initial particle periods to calculate \( \epsilon_i \). In this work, our M2M algorithm is exclusively applied to halos, so we can approximate the initial periods by

\[
\tau_r = 2\int_{r_1}^{r_2} \frac{dr}{\sqrt{2[E - \Phi(r)] - L^2/r^2}},
\]

from Binney and Tremaine [1987] where \( E \) is the energy of the particle, \( \Phi(r) \) is the potential of the system, \( L \) is the total angular momentum of the particle, and \( r_{1,2} \) are the turning points of the orbit.

In addition, the constants in Eq. [3.41] are also unknown. The \( \delta t \) constant is arbitrary, so we choose to set \( \delta t \) to be equal to the time-step over which the system is integrated. The rest of the constants are unitless, so they can be combined them into one unknown constant, \( n \). The unknown nature of \( n \) seems to leave the \( \epsilon_i \) scaling at an impasse, as without a value for \( n \), \( \epsilon_i \) cannot be calculated. However, there is a solution to this problem. If \( n \) is assigned arbitrarily, a second, single-step scaling can be implemented to keep the weight changes below the adiabatic limit.

The single-step scaling is motivated by the idea that the adiabatic condition suggests that the fractional weights cannot change by more than a factor \( f_{\text{max}} \) in a single step. This is a simple modification of the adiabatic limit being for a limit on a particle’s weight change...
over one period. In order to enforce the single step limit, $\epsilon_i$ can be multiplied by a scaling factor, $f$, every time step, where $f$ is given by

$$
f = \min \left\{ \frac{1}{f_{\max} (\epsilon_i g_i \delta t)_{\max}} \right\}.
$$

(3.43)

This sets the scaling factor to be either one or, if the maximum fractional change of any particle is larger than $f_{\max}$, to a factor such that $f \epsilon_i g_i (t) \delta t = f_{\max}$. Then the weight change equation can be redefined as

$$
\frac{dw_i}{dt} = w_i (f \epsilon_i g_i (t)),
$$

(3.44)

This ensures that a particle can only change its fractional weight by, at most, $f_{\max}$ in a single time step. It then scales down all the other weight changes proportionally.

When M2M is integrated over one step, the weights are changed by

$$
\delta w_i (t) = f w_i \epsilon_i g_i (t) \delta t.
$$

(3.45)

Based on this, we will define a quantity $\epsilon'_i$ as

$$
\epsilon'_i = \epsilon_i \delta t = \frac{1}{n} \left( \frac{\delta t}{\tau_i} \right)^{1-\alpha}.
$$

(3.46)

This allows for a simple implementation of all the scalings, as $n$ is arbitrary, $\delta t$ is the length of the time-step, and $\alpha$ is found by various trials. Before any M2M steps, the full set of $\epsilon'_i$ are calculated. Then, at each M2M step, $g_i$ is calculated and used to get the scaling factor. Finally the weights are adjusted according by

$$
\delta w_i (t) = f \epsilon'_i w_i g_i (t).
$$

(3.47)
3.7 Turning M2M on and Off

M2M algorithms are generally turned on and off completely in the first and final steps of a simulation. However, this may not be the best method of turning M2M on and off. One way of thinking of \( \frac{dw_i}{dt} \) is as the velocity of the weight changes. This suggests that it would be best to smoothly accelerate the weight changes from a stand-still when a simulation starts and then to smoothly slow down to a stop when a simulation is finished. As well, a slow turn off can help with the problem of ill-conditioning as slowly turning off M2M will settle the weights into one configuration.

The acceleration of the weight changes is implemented by taking advantage of the single-step scaling factor introduced in Sec. 3.6. The single-step scaling ensures that the fractional weight change in one step is less than some fraction \( f_{max} \). To turn the algorithm on and off smoothly, \( f_{max} \) is made into a function of time that begins at zero, increases to some standard value, remains constant, and then slowly decreases back to zero. This can be written as

\[
f_{max}(t) = \begin{cases} 
    f_{\text{grow}}(t) & 0 \leq t < t_g \\
    f_{\text{max, st}} & t_g \leq t < t_o \\
    f_{\text{decay}}(t) & t_o \leq t < t_o + t_g \\
    0 & t \geq t_o + t_g 
\end{cases},
\]

(3.48)

where \( f_{\text{max, st}} \) is the standard scaling factor, \( t_g \) is the time over which the function grows or decays and \( t_o \) is the time when M2M is turned off.

For the acceleration to be smooth the derivative of both \( f_{\text{grow}} \) and \( f_{\text{decay}} \) should be zero at the limits and continuous at all times. Taking inspiration from the drag term introduced in Holley-Bockelmann et al. (2001), \( f_{\text{grow}} \) and \( f_{\text{decay}} \) are defined as

\[
f_{\text{grow}}(t) = f_{\text{max, st}} \left[ 3 \left( \frac{t}{t_g} \right)^2 - 2 \left( \frac{t}{t_g} \right)^3 \right],
\]

(3.49)
and
\[
\begin{align*}
  f_{\text{decay}}(t) &= f_{\text{max, st}} \left[ 3 \left( \frac{t_o + t_g - t}{t_g} \right)^2 - 2 \left( \frac{t_o + t_g - t}{t_g} \right)^3 \right].
\end{align*}
\]  
(3.50)

These functions go smoothly from zero to one for \( f_{\text{grow}} \) and one to zero for \( f_{\text{decay}} \). Now, all that must be chosen is a value for \( f_{\text{max, st}} \) that is appropriate when considering the adiabatic limit. In this work we always choose \( f_{\text{max, st}} = 0.01 \), so that the largest fractional change in one step is 1%.

### 3.8 Implementation

With the algorithm having been described in great detail it is worthwhile to now discuss the actual implementation of the entire algorithm. To begin using the algorithm an initial equilibrium N-body system must be generated. This is done using the GalactICS code (see [Widrow et al.](2008) for a description of GalactICS). Once the system is generated, an N-body code is needed to advance the particles in time. The N-body code (called the DNC code) used in this work is the implementation of the Dehnen algorithm ([Dehnen](2001)) by [Stiff et al.](2001).

The DNC code uses a fast algorithm with \( O(n) \) operations. The algorithm works by dividing the system into cells, in a similar manner to standard tree codes. The main difference between the DNC code and a standard tree-code is that, when two cells are well separated, it is possible to calculate the force of the first cell on the second cell and vice versa in a single step. A full explanation of this calculation and the criteria for two cells being well separated is beyond the scope of this work. The reader is encouraged to read [Dehnen](2001) for details of the algorithm. It is sufficient to note that the algorithm conserves momentum, and that the single step calculations reduce the order of operations from
One important aspect of our M2M algorithm is that it can be applied to the entire system, or just to one component within the system. So, if we wish to examine the response of a disk in a prolate halo, it is possible to use M2M to adjust only the halo particles to match some prolate density constraint, while the disk particles simply respond to the new potential without any M2M adjustments. As such, we must include a step that separates the particles that will be adjusted by M2M from the rest of the particles.

The actual M2M algorithm is written as a series of subroutines that are inserted into the DNC code. Before advancing the system a single step, one subroutine constructs the grid, calculates the target constraints, and reads in the \( T \) matrix for transforming the particle’s positions. In addition, the particle periods are generated prior to any simulation and are read in by the initial subroutine. These periods are then used to calculate \( \epsilon_i' \) for all the particles being adjusted.

Once the initialization is completed, the system is advanced by the N-body code. After the system is advanced, the particles that will undergo weight adjustments are passed to the M2M subroutine. In the M2M subroutine, the pseudo-observations are calculated using the appropriate kernels. These are compared to the target observations to generate the \( \Delta_j \) terms. The weights are then adjusted according to Eq. 3.47. The masses are then adjusted and the momentum and centre of mass corrections are performed. Once M2M has run for a specified length of time, it slowly turns off and the system is allowed to relax.
Chapter 4

Isolated Halos

In this chapter our self-consistent M2M algorithm will be used to generate isolated, triaxial DM halos. The first section will discuss the properties of the initial system and define the target system. The second section will describe the tools used for the analysis of the halos. The next sections deal with the effects of particle number, the form of $\epsilon_i$, and the inclusion of entropy. Finally the general properties of the generated triaxial halo will be shown.

4.1 The Initial and Target System

We generate the initial systems, using the GalactICS code. This code is designed to generate multi-component galaxies that are axisymmetric using a self-consistent distribution function that satisfies both the CBE and the Poisson equations. In this chapter we wish to test the M2M code and investigate triaxial halos, so we generate systems that consist of only a DM halo.

The halos generated by GalactICS have a modified NFW density profile. The NFW
profile was found by Navarro et al. (1996) and described by

\[ \rho(r) = \frac{\rho_o}{\left(\frac{r}{a}\right)^2 \left(1 + \frac{r}{a}\right)^2}. \] (4.1)

The GalactICS halo density profile is modified slightly to include a power for the inner cusp and a truncation radius that gives the system a definite size:

\[ \rho(r) = \frac{2^{2-\gamma} \sigma_h^2}{4\pi a_h^2} \left(\frac{r}{a_h}\right)^\gamma \left(1 + \frac{r}{a_h}\right)^{3-\gamma} C(r; r_h, \delta r_h), \] (4.2)

where \( \sigma_h \) is the scale velocity dispersion, \( a_h \) is the scale radius, \( \gamma \) is the power of the inner cusp, and \( C \) is a truncation factor. This truncation factor is described by

\[ C(r; r_h, \delta r_h) = \frac{1}{2} \text{erfc} \left( \frac{r - r_h}{\sqrt{2\delta r_h}} \right), \] (4.3)

where \( r_h \) is the halo’s extent and \( \delta r_h \) is the truncation scale.

The halos generated for investigation are isotropic with no rotation. All the halos are generated using the same parameters. The only variation between different trials is the particle number. The parameters of the initial halos are

\[ \begin{align*}
\sigma_h &= 1, & \gamma &= 1, \\
\sigma_h &= 1, & a_h &= 1, \\
\delta r_h &= 5, & r_h &= 25,
\end{align*} \] (4.4)

where the units of velocity are 100 km/s and the units of distance are kpc. These generates a simple generic halo with a mass of 2.27 units (the units of mass are \( 2.32 \times 10^9 \text{ M}_\odot \)). The density profile for these parameters is shown in Fig. 4.1.

Just as the initial halos are all the same in this chapter, the same target is used for all the simulations in this chapter. A significant portion of this work is the testing of our self-consistent M2M algorithm, so it is useful to always use the same target until the performance of the algorithm is satisfactory. The target halo has axis ratios of 1 : 0.7 : 0.5.
Chapter 4. Isolated Halos

Figure 4.1 The initial density of the system compared to the analytic density. The blue points are the density of the system in each cell within a 21 by 8 by 8 grid while the red curve is the analytic density calculated for each cell. The measured density appears to be discrete in $x$ and have a range of values in $y$. The discreetness in $x$ is due to the location of the radial shells, while the spread in $y$ is due to the small fluctuations of $\rho$ between the different angular cells. The radial bins are not spaced equally in either log-space or physical space, and so the radial measurements do not appear to be equally spaced in this logarithmic graph. The $x$-axis is in kpc and the $y$-axis is in units of $1 \times 10^9 \, \text{M}_\odot/\text{kpc}^3$. 
reach this target the transformation matrix from $R$ to $R'$ coordinates is

$$T = \begin{pmatrix} 0.7 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1.4 \end{pmatrix}. \quad (4.5)$$

The system will converge faster if the initial system is close to the target system so we set the target density profile to be the same as Eq. 4.2, except with $r'$ rather than $r$.

The density is calculated using a grid of 20 radial shells and one inner sphere in the $R'$ frame. The inner sphere meets the innermost radial shell at $R'_{\text{min}} = 0.5$ kpc. The radial shells then extend to an outer radius $R'_{\text{max}} = 30$ kpc. The radial shells are spaced equally in $r''$ where

$$r'' = \log \left( 1 + \frac{r'}{a'} \right), \quad (4.6)$$

and $a'$ is just a constant. This spacing allows for sufficient sampling in both the inner and outer regions of the system. In all our runs we use $a' = 10R'_{\text{min}}$. The centre sphere and each radial shell are subdivided into an $8 \times 8$ angular grid. The size of the angular cells is designed so that every cell in a given radial shell has an equal volume in the the $R'$ frame.

### 4.2 Analysis

In order to determine the success of our algorithm and the properties of our triaxial halos we use a number of simple tools. They are the density profile, $\chi^2$, the axis ratios, and the anisotropy profile, $\beta$. The density profile and $\chi^2$ are calculated every M2M step, where $\chi^2$ is given by

$$\chi^2 = \frac{1}{P} \sum_{j=1}^{P} \Delta_j. \quad (4.7)$$
We use $P$ rather than $M$ to indicate the number of observations in order to avoid any confusion with mass. In order to examine the convergence as a function of time we periodically output the the density profile and $\chi^2$.

The morphology of the system is analyzed using the method suggested in Dubinski and Carlberg (1991). This consists of diagonalizing the moments of the inertia tensor

$$
\bar{I}_{i,j} = \sum_n \frac{x_{i,n}x_{j,n}}{r_n^2}
$$

(4.8)

where

$$
r_n^2 = x_n^2 + \left(\frac{y_n}{q}\right)^2 + \left(\frac{z_n}{s}\right)^2,
$$

(4.9)

and $q = b/a$ and $s = c/b$. The diagonalized inertia tensor can then be used to get a new $q$ and $s$. This calculation can be iterated until the values of $q$ and $s$ used in the calculation of $r_n$ converge with the axis ratios calculated from the diagonalized inertia tensor. It is possible to calculate the inertia tensor for the entire system, or just a part of the system. Therefore the axis ratios can be found for the entire system or for shells within the system.

It is impossible for a triaxial system to have a completely isotropic velocity dispersion. Thus, the anisotropy profile of the systems generated by our M2M algorithm will be informative. The anisotropy parameter, $\beta$, is found by

$$
\beta(r) = 1 - \frac{\sigma_\phi(r)^2 + \sigma_\theta(r)^2}{2\sigma_r(r)^2},
$$

(4.10)

and $\sigma^2$ is the velocity dispersion in spherical coordinates. The one slight complication is that it is necessary to remember that the particles have differing masses, and thus a mass-weighted value of $\sigma^2$ must be calculated. In this work we calculate $\beta$ in spherical shells about the origin.
4.3 Particle Number

We will first examine how our algorithm depends on particle number. If the particle number is small, the system will not converge as tightly due to having larger values of $\sigma_j$, and the system may undergo processes like two-body relaxation. However, the computational time increases with particle number, so finding the minimum number of particles necessary for convergence will allow for the most efficient simulations.

In order to study the effects of particle number, a number of simulations are run with differing numbers of particles. The initial halos are generated with the parameters given in Eq. 4.4 using $N = 10^4 - 5 \times 10^5$. For simplicity, we set $\alpha = 1$, $\mu = 0$, and $n = 10$, so that

$$
\epsilon_i' = \frac{1}{10} \frac{\delta t}{\tau_i}.
$$

Each system is evolved using the M2M algorithm for 4000 time units (where one time unit is $\approx 1 \times 10^7$ yr) and then allowed to relax for an additional 2000 time units with a turn-on and off period of $t_g = 100$. This evolution length is approximately 40 circular orbits at the half-mass radius. The long relaxation time allows us to check how stable our resulting systems are and examine any relaxation effects.

The axis ratio evolution of the simulations, shown in Fig. 4.2, have a number of interesting phenomena. First, none of the systems seem to reach the target. However, as the particle number increases the systems seem to converge closer to the target. As well, there is a rise in the axis ratios when M2M is turned off that seems to be inversely proportional to the particle number. Each of these phenomena can be understood through careful examinations of the axis ratios and properties of the systems.

Let us begin by examining the lack of axis ratio convergence shown in Fig. 4.2. While none of the systems have converged to the target axis ratios, it is clear that, for the systems
Figure 4.2 The effect of particle number on M2M. The black curves are for the $10^4$ particle run, the blue curves are for the $2.5 \times 10^4$ particle run, the red curve are for $5 \times 10^4$ particle run, the green curves are for the $10^5$ particle run, the magenta curves are for the $2 \times 10^5$ particle run, and the cyan curves are for $5 \times 10^5$ particle run. The dashed horizontal black lines show the target axis ratios, while the solid vertical black lines show when M2M begins to be turned off and when it is completely turned off. None of the curves appear to reach the target axis ratios, and the majority appear to increase after M2M is turned off.
with larger $N$, the axis ratios are still decreasing when M2M is turned off. This decrease means that the lack of convergence is not due to M2M failing, but rather due to the evolution taking longer than the simulations have been run. It also appears that the axis ratios of each system approach the target axis ratios asymptotically. This asymptotic behaviour is not unexpected due to the fact that $\frac{dw_i(t)}{dt} \propto \sum_{j=1}^{M} \Delta_j$. As the system gets closer to the target, the $\Delta_j$ terms decrease, thereby decreasing the rate of change, and the rate at which the axis ratios evolve. The asymptotic behaviour of the axis ratio evolution indicates that, while M2M will converge with this form of $\epsilon'_i$, the convergence will take a prohibitively long time.

It is clear from Fig. 4.2 that the simulations with larger $N$ were able to get closer to the target. This increased convergence with $N$ is also due to the $\Delta_j$ terms. The error terms, $\sigma_j$, will decrease as $N$ increases. Since $\Delta_j$ is inversely proportional to $\sigma_j$, $\Delta_j$ will increase, and that increase will increase $\frac{dw_i(t)}{dt}$. The larger weight change will speed the system’s convergence to the target constraints. As such, it is easy to understand why the systems with larger $N$ get closer to the target constraints in the same length of time.

The rise in the axis ratios seen in Fig. 4.2 when M2M is turned off indicates that the system relaxes to a spherical shape and this relaxation is more pronounced for systems with small $N$. To understand this relaxation, it is useful to examine the axis ratios as function of $r$ both when M2M is turned off and at the end of the simulation. The axis ratios as a function of $r$ for the $10^4$, $10^5$, and $5 \times 10^5$ particle runs are shown in Fig. 4.3. The axis ratios show that the system is closer to the target in the inner region than the outer region and that the relaxation seems to primarily occur in the inner region. The convergence from the inside out is consistent with $\epsilon'_i$ being larger in the inner region than the outer region. As such the rate of change is larger in the inner region and the system will converge to the
Figure 4.3 The axis ratios as a function of $r$ in kpc for three different simulations. A) is for the $10^4$ particle run, B) is for the $10^5$ particle run, and C) is for the $5 \times 10^5$ particle run. The blue curves are the system when M2M is turned off and the red curves are after relaxing. The dashed black horizontal lines are the target axis ratios. The solid lines are the semi-major/major ratios and the dashed lines are the minor/major ratios.
constraints faster. The increase in axis ratios from the inside out indicates the process that causes the relaxation occurs primarily in the inner region.

In any large system that evolves for a significant amount of time, some particles will undergo multiple two-body interactions. Over a long enough period these interactions will cause two-body relaxation. However, even on a shorter time scale, these interactions can cause mass segregation. Mass segregation is seen in globular clusters where the more massive stars fall towards the centre of the cluster due to the equipartition of energy. The more massive particles tend to settle into smaller orbits with a lower eccentricity, while the lighter particles will scatter outwards onto orbits with large eccentricities. Since there are more interactions in the inner, high-density regions, than in the outer regions, we will first see the effects of mass segregation in the inner regions. Our axis ratios are mass weighted, which means that the settling of massive particles onto less eccentric orbits will ultimately increase the measurements of the axis ratios in the inner regions. In addition, if the particle number is high, the effect of mass segregation on the axis ratios will decrease, as the increased number of massive particles will scatter off each other, keeping the axis ratios from rising significantly.

To check whether mass segregation is occurring we plot the average mass per particle per radial bin when M2M is turned off and after the system has relaxed in Fig. 4.4. This plot shows that, for the $10^4$, $2 \times 10^4$, and $5 \times 10^5$ particle runs, the average mass per particle increases in the inner region as the system relaxes. This mass increase is a definite sign of mass segregation, which, based on the previous arguments, suggests that mass segregation is the cause of the rise in the axis ratios seen in Fig. 4.2.

The end result of the particle number tests show that increased particle number helps both the convergence properties of M2M and the stability of the final system. We want to
Figure 4.4 The average mass per particle for three different simulations. The A) pair is for the $10^5$ particle run, the B) pair is for the $2 \times 10^5$ particle run, and the C) pair is for the $5 \times 10^5$ particle run. The upper figure shows the average mass per shell in physical frame and the lower panel shows it for the $R'$ frame. The blue curve shows the ratio of the final average mass per particle to the initial average mass per particle when M2M is turned off and the red curve shows the same ratio for the final configuration. The small vertical bars show the error bars of the average density measurements using $\sigma_j = \bar{M}_j / \sqrt{n_j}$. The units of $r$ are kpc.
perform our analysis on the most stable systems possible, so we will generate our models using $5 \times 10^5$ particles. However, we still have a significant number of tests to perform and increasing the particle number also increases the computational time required significantly. In the tests, we merely need to find the general behaviour of the system with various parameters, so we do not require the same degree of stability. As such we will perform our tests of $\alpha$ and pseudo-entropy using systems that only consist of $5 \times 10^4$ particles.

4.4 Form of $\epsilon$

One of the key aspects of our implementation of M2M is the power law index, $\alpha$, in the calculation of

$$\epsilon'_i = \frac{1}{n} \left( \frac{\delta t}{\tau_i} \right)^{1-\alpha}. \tag{4.12}$$

As discussed in Sec. 3.6, the optimum value of $\alpha$ cannot be known a priori and must be found through experimentation. In this section we will attempt to find that optimum value using four simulations with $\epsilon'_i$ set to

$$
\begin{align*}
\epsilon'_i &= 0.05 \delta t, \\
\epsilon'_i &= 0.01 \left( \frac{\delta t}{\tau_i} \right)^{1/2}, \\
\epsilon'_i &= 0.025 \left( \frac{\delta t}{\tau_i} \right)^{3/4}, \\
\epsilon'_i &= 0.1 \left( \frac{\delta t}{\tau_i} \right). 
\end{align*} \tag{4.13}
$$

where $\delta t$ is set to be the length of the time-step. The differing values of $n$ are chosen to scale the $\epsilon'_i$ forms to roughly the same level at the scale radius. These different simulations will be referred to by the $\alpha$ constant in $(\delta t/\tau_i)^{1-\alpha}$. One thing to mention is that the first simulation, corresponding to $\alpha = 1$ in Eq. 3.41, is not actually a solution of Eq. 3.37. However, it does correspond to a constant $\epsilon$ and is a good comparison to the standard M2M
Figure 4.5  The form of $\epsilon_i'$ as a function of $r$ with $r$ in kpc. The thin black line is for the value of used in the $\alpha = 1$ run. The blue points are for $\alpha = 0$ run, the red points are for the $\alpha = 0.25$ run, and the green are for the $\alpha = 0.5$ run.

algorithms. The plot of $\epsilon_i'$ shown in Fig. 4.5 shows that the $\alpha = 1$ run has the largest values of $\epsilon_i'$ in the outer region and the magnitude of $\epsilon_i'$ decreases with $\alpha$.

The axis ratio evolution shown in Fig. 4.6 exhibits some very interesting behaviour. Firstly, the simulations with a smaller $\alpha$ are farther from the target when M2M is turned off than those with larger $\alpha$ values. Secondly, and more interestingly, there is a jump in the axis ratios for the simulations with large values of $\alpha$, and the size of this jump seems to be proportional to the size of $\alpha$. After this period of rapid axis ratio evolution, the systems seem to become more triaxial and the axis ratios decrease towards the target values. Finally a longer, more steady increase in the axis ratios occurs that matches the relaxation seen in Sec. 4.3 and is caused by the same mechanism.

The reason for the large $\alpha$ simulations getting much closer to the target, while M2M is
Figure 4.6 The effect of the form of $\epsilon_i$ on M2M. The dashed black horizontal lines are the target axis ratios and the solid black vertical lines show when M2M begins being turned off and when it is completely turned off. The black curves are for the $\alpha = 1$ run, the blue curves are for $\alpha = 0.5$, the red curves are for $\alpha = 0.25$, and the green curves are for $\alpha = 0$. 
Figure 4.7  The axis ratios as a function of $r$ in kpc for two different simulations. A) is for the $\alpha = 1$ run and B) is for the $\alpha = 0.5$ run. The blue curves are the system when M2M is turned off and the red curves are at the early peak in the axis ratios. The dashed black horizontal lines are the target axis ratios. The solid lines are the semi-major/major ratios and the dashed lines are the minor/major ratios.

on, than the small $\alpha$ simulations is quite simple. The large $\alpha$ simulations have larger values of $\epsilon_i'$, and cause larger weight changes at each step. These larger weight changes will push the system to converge to the target constraints much faster.

The jump in the axis ratios when M2M is turned off is much more difficult to understand. An examination of the axis ratios as a function of $r$ in Fig. 4.7 shows that the increase in the axis ratios primarily occurs in the outer regions of the systems. This behaviour, along with the understanding of the $\alpha$ constant from Sec. 3.6 gives the cause of the jump in axis ratios. In Sec. 3.6 we defined $\alpha$ as the approximate power by which $g_i$ decreases over one period. But, since we do not actually know how $g_i$ decreases, and simply choose a test $\alpha$. If the test $\alpha$ suggests that $g_i$ decreases faster than it actually decreases, it will result in a value of $\epsilon_i$ that may violate the adiabatic limit. The violation of the adiabatic limit will preferentially occur in the outer regions of the system as the value of $\epsilon_i$ varies far more
dramatically in the outer regions for different value of $\alpha$.

If $\epsilon_i$ is large enough to violate the adiabatic limit, the masses of the particles will change faster than the orbits. This means that the system will have a velocity dispersion that does not correspond to the mass distribution. While the mass distribution may have the required triaxiality, the velocity profile will not maintain that shape. Once M2M is turned off, the system will relax according to the velocity dispersion. This relaxation will cause the axis ratios to increase, and to primarily increase in the outer region. As such, it seems that the violation of the adiabatic limit in the outer region causes the axis ratio jumps seen in Fig. 4.6 and explains why the high $\alpha$ runs have a larger jump.

In addition, the mismatch between the mass distribution and the velocity dispersion also explains the re-triaxialization of the system that occurs following the peak of the axis ratio jump. We can think of the mass distribution of the system as being ‘in phase’ with the target mass distributions when M2M is turned off. As the particles move along their orbits, we would expect that, if the orbital periods were all equal, the system would return to being ‘in phase’ and the axis ratios would return to the values that had when M2M were turned off. Of course, the orbital periods are different for all particles, but since the axis ratio jump occurs primarily in the outer region, and since the density falls off significantly in the outer region, we know that there is a region that must dominate the increase of the total axis ratios. In that region the periods are similar, and after half a period, we expect the system to return to being ‘in phase’ and the axis ratios to re-triaxialize and decrease. Based on this we would expect oscillations in the axis ratios until sufficient mixing occurs. We do see one rise and fall of the axis ratios, but, before any more oscillations occur, the relaxation seen in Sec. 4.3 takes over and causes the axis ratios to increase.

The results in this section show that there is a trade-off between a speedy convergence
and the stability of the final system. The $\alpha = 1$ simulation converges swiftly but relaxes significantly, while the $\alpha = 0$ takes a long time to converge but it is much more stable. Based on these results we find that, due to time considerations and the small size of the relaxation, $\alpha = 0.5$ is the optimum value for future simulations.

### 4.5 Entropy

Thus far we have not included the pseudo-entropy term in any simulations in order to isolate the effects of particle number and $\alpha$. However, it is possible that the inclusion of entropy will solve the problem of increasing axis ratios due to mass segregation. The pseudo-entropy can cause the mass dispersion to shrink, and if the mass dispersion is smaller, the amount of mass segregation will also decrease, which should reduce the increase in axis ratios while the system relaxes.

The pseudo-entropy term is

$$ S = -\sum_i w_i(t) \log \left( \frac{w_i(t)}{w_{i,o}} \right), \quad (4.14) $$

where $w_{i,o}$ represents the priors rather than $m_i$ as in Sec. 2.3.1. We make this change in variables in order to avoid confusion with the particle masses. The pseudo-entropy will drive particle weights towards the priors with a strength given by the constant $\mu$. Thus, if the priors are equal for all particles, the pseudo-entropy will attempt to reduce the mass dispersion.

There is a possible drawback to including a pseudo-entropy term. Instead of minimizing $\chi^2$, M2M will be minimizing the entropy plus $\chi^2$. This minimization may create a final system that is significantly farther from the target constraints than is desired. If the final system is too far from the constraints it would suggest that it is inadvisable to include
pseudo-entropy in future simulations.

To investigate the effects of the pseudo-entropy term, three simulations are run using differing values of $\mu$: $\mu = 0$, $\mu = 100$, and $\mu = 10^4$. Each simulation consists of $5 \times 10^4$ particles and is run using $\alpha = 0.5$. The priors are set to equal the initial weights of the system as we have no better initial guess. The resulting mass dispersion is shown in Fig. 4.8. The mass dispersion shows that the $\mu = 100$ simulation has a negligible effect on the mass dispersion, while the $\mu = 10^4$ simulation narrows the mass dispersion significantly. Based on these results, we would expect that the axis ratio evolution will be the same for the $\mu = 0$ and $\mu = 100$ simulations, and the $\mu = 10^4$ simulation will have a smaller increase in axis ratios. An examination the axis ratio evolution shown in Fig. 4.9 reveals that the $\mu = 0$ and $\mu = 100$ simulations do indeed evolve nearly identically. It also reveals that the $\mu = 10^4$ simulation converges to a system very different than the target system, has a larger jump in axis ratios due to violating the adiabatic limit and that the axis ratios still increase after the jump, albeit to a slightly lesser degree than the smaller pseudo-entropy simulations.

The results from the pseudo-entropy simulations suggest that it should not be included in rest of this work. When the pseudo-entropy is large enough to reduce the relaxation from the mass dispersion, it prevents the system from converging to the target constraints. While it may be possible to construct a set of priors that keep the system stable when M2M is turned off and still allow the system to converge to the constraints, it does not seem particularly likely. Based on these results, we do not include any pseudo-entropy in the rest of this work.
Figure 4.8 The mass dispersion for the three simulations. Each curve shows the mass dispersion when M2M is turned off. The blue curve shows the system for the $\mu = 0$ run, the red curve for the $\mu = 100$, and the green curve for the $\mu = 10^4$ run. The black curve shows the initial mass dispersion.
Figure 4.9  The effect of the pseudo-entropy on M2M. The blue curves are for $\mu = 0$, the red curves are for $\mu = 100$, the green curves are for $\mu = 10^4$. The dashed horizontal black lines are show the target axis ratios, while the solid vertical black lines show when M2M begins to be turned off and when it is completely turned off.
CHAPTER 4. ISOLATED HALOS

4.6 Final Halo Properties

Now that we have investigated the effects of particle number, $\alpha$, and pseudo-entropy on our implementation of M2M, we can generate a triaxial halo. We will use a system with $5 \times 10^5$ particles (based on the results of Sec. 4.3) and evolve it using $\alpha = 0.5$ and we will not include any pseudo-entropy. Since our results seem to suggest that the convergence occurs faster when using $\alpha = 0.5$, we will only evolve the system for 2000 time units, and allow it to relax for 2000 time units. Figure 4.10 shows the main results of this simulation.

The axis ratio evolution in the upper panel of Fig. 4.10 shows both the axis ratios of the system, and the axis ratios of the ‘reduced’ system. The ‘reduced’ system is generated by the pseudo-observations of the density. The density is calculated by taking the mass of all the particles in a cell and dividing it by the volume of that cell. This process is equivalent to creating a particle in the centre of the cell that has a mass equal to the total mass of the $n_j$ particles within that cell. In other words, each observation of the density can be thought of as a reduction of the system of $N$ particles to a smaller system with $P$ particles. It is this ‘reduced’ system that generates $\chi^2$ and drives the M2M evolution.

The axis ratio evolution shows that the system rapidly converges to the target axis ratios. There is a small difference between the ‘reduced’ system and the actual system, but it is not too significant. There is also a small jump in the axis ratios when M2M is turned off. This is not unexpected due to the use of $\alpha = 0.5$, and the jump is not large. To see how close we are to the target within numerical noise, we plot $\chi^2$ as a function of time in the Lower right panel of Fig. 4.10. Based on the $\chi^2$ value and the axis ratio evolution we are justified in claiming that we have generated a stable, dynamically self-consistent, triaxial halo.

In Fig. 4.11 we plot the density of our system when M2M is turned off and when we finish the simulation. This serves as a final confirmation of the success of our algorithm.
Figure 4.10 Upper Panel: The axis ratios of the system as a function of time. The thin horizontal lines are the targets and the thin vertical lines are when M2M begins turning off and when it is completely turned off. The black and blue curves show the axis ratios of the system, while the red and green curves show the axis ratios of the reduced system.

Lower Left Panel: $\chi^2$ for the system as a function of time.

Lower Right Panel: The weights of 5 random particles as a function of time.
CHAPTER 4. ISOLATED HALOS

Figure 4.11  The density as a function of $r'$ in kpc'. A) is the density of the halo when M2M is turned off and B) is the density of the halo when the simulation is finished. The red curve is the target density, and the blue points are the measured density in each cell.

We can see from this plot that we achieve a very good fit to the target density at all radii.

Finally, the lower right panel of Fig. 4.10 shows the weight evolution of 5 random particles. The weight evolution is fairly smooth, with some large-scale jumps and drops in the weights. The large-scale weight changes are not completely unexpected due to the finite size of the observational cells. While a particle is in one cell, it will evolve according to the value of $\Delta$ in that cell. When the particle moves from one cell to another, $\Delta$ will change and the particle will suddenly start evolving differently. If the particle has an orbit that moves it between cells with a negative value of $\Delta$ and a positive value of $\Delta$ the weights will have large-scale jumps and drops with a period that is proportional to that particle’s period.

To see how the system converges the axis ratios are plotted as a function of $r$ in Fig. 4.12. This plot shows that the system converges much more closely in the inner region than the outer region and that the the axis ratios of the final system have increase by roughly the
same amount at all radii. The axis ratio increase at all radii is not unexpected as the outer region has the rapid jump in axis ratios seen in Sec. 4.4 and the inner region relaxes due to mass segregation seen in Sec. 4.3.

In order to further investigate the final system, we plot the mass histogram in Fig. 4.13. We can see that the mass dispersion covers approximately two orders of magnitude and peaks at a mass slightly less than that of the initial mass. The movement of the peak is interesting as it indicates that the majority of the particles have had their mass decreased while the total mass of the system is conserved.

For a system to be triaxial, the velocity dispersion must have some anisotropy. The extent of the anisotropy can be explored by examining $\beta$ as a function of $r$. The naive expectation would be that the anisotropy is constant throughout the system. However, Fig. 4.14 shows that there is an increase in $\beta$ in the inner region to some constant value and a
Figure 4.13 The mass histogram for the final system. The black curve shows the initial mass while the blue curve shows the dispersion of masses after running M2M.

The final, dramatic increase in the outer region. The final increase may be due to low particle number in the outer region, and the inner rise may be a remnant of the initial spherical halo. Nonetheless, the behaviour is quite different from the initial expectation.

The anisotropy profile suggests an interesting avenue for further research. It is likely that the \( \beta \) profile for a simple triaxial halo depends on the morphology. It may be useful to develop a number of isolated halos with differing morphologies in order to find what relationship \( \beta \) has with the morphology. If a relationship between the two is found, then that relationship may allow observers to determine the morphology of a halo by the \( \beta \) profile of a tracer population. This idea has many assumptions and is beyond the scope of this work. It is likely that this idea is far too difficult to implement practically. However, the idea of a relationship between the anisotropy and the morphology is worth further investigation.
Figure 4.14 The anisotropy profile of the system as a function of $r$ in kpc. The black curve indicates the initial system, the blue curve indicates the system when M2M is turned off, and the red curve indicates the final system.
Chapter 5

Disk-Halo Systems

In this chapter we will use our self-consistent M2M algorithm to generate disk-halo systems with prolate halos. First we will explore some reasons for generating disk-halos systems with non-spherical halos. Then the initial and target systems will be described. Next, some extra tools for analyzing disk-halo systems will be presented and explained. Finally the results of the disk-halo systems will be shown.

5.1 Motivation

Cosmological simulations have shown that the majority of DM halos are non-spherical in shape. This means that most disk galaxies lie within a non-spherical halo. Our M2M algorithm can generate dynamically self-consistent equilibrium systems where the halo has a specified morphology. As such, it is a perfect tool to study the effects of a non-spherical halo on a disk. In addition, LSB galaxies are thought to be DM dominated and consist of only a disk and a halo. As such, if we generate DM dominated disk-halo systems with non-spherical halos, they will be analogues to LSBs. We mention this because LSBs have
been studied extensively and their inner rotation curves are often used as indicators of the inner density profile of their DM halos (Flores and Primack (1994), de Blok et al. (2001), de Blok (2005)). The inner rotation curves exhibit a wide variety of shapes that seem to indicate that the inner profile of real DM halos have a core rather than the NFW cusp.

However, Hayashi and Navarro (2006) point out that calculations of the inner profile of the halo from the rotation curves assume that the halo is spherical. Since cosmological simulations suggest that DM halos are actually triaxial, the calculation of the inner profile from the rotation curve may be incorrect. To illustrate this error, Hayashi and Navarro (2006) generate a slightly prolate NFW halo and examine the effect that this halo would have on the rotation curve of a mass-less disk. Their results show that the non-axisymmetry of the halo causes the rotation curves to look like those from a cored halo rather than a cuspy halo.

Since we can generate an N-body model with a non-spherical halo, we can attempt see whether the Hayashi and Navarro (2006) results hold true for disks with self-gravity. More importantly, we can see what other effects a non-spherical halo has on a disk, such as the velocity profile and morphology. We will generate disk-halo systems with prolate halos as they are simplest systems to generate and understand.

5.2 Initial System and Target

The initial system will be a spherical halo and axisymmetric disk generated by the GalactICS code. The disk generated by the GalactICS code has a density profile that falls off exponentially in $R$ with a scale length of $R_d$. The vertical component of the density falls off as sech$^2$ with a scale length of $z_d$. The disk is truncated at $R_{out}$ with a sharpness of $\delta R_d$. The disk must also have some random motions to keep it stable. As such the radial velocity
dispersion is given by

\[ \sigma^2_R = \sigma^2_{R0} e^{R/R_\sigma}. \] (5.1)

The parameters chosen for the disk are

\[ \begin{align*}
M_d &= 0.2, & R_d &= 1, \\
z_d &= 0.1, & R_{out} &= 5, \\
\delta R_d &= 5, & \sigma_{R0} &= 0.06, \\
R_\sigma &= 5,
\end{align*} \] (5.2)

while the halo parameters are the same as those in chapter 4. The mass is in units of $2.32 \times 10^9 M_\odot$, the velocities in 100 km/s, and the distances in kpc. The final system will be an axisymmetric equilibrium disk-halo system with a stable disk that has roughly 10% of the mass of the halo.

The goal of these simulations is to generate an equilibrium disk-halo system with a non-spherical halo. Therefore, only the halo particles will be passed to the M2M algorithm, and the target constraints will be for the halo’s density profile. The target profile will have the same parameters as the target in chapter 4 with one exception. Rather than generating a triaxial halo, the halo generated will be a simpler prolate halo. We will generate two systems, one with halo axis ratios of $1:0.9:0.9$ and the other with $1:0.8:0.8$. The long axis of the halos will be aligned with the y-axis. For all the simulations in this chapter, the halo will consist of $5 \times 10^5$ particles and the disk will consist of $1.5 \times 10^5$ particles.

### 5.3 Extra Analysis Tools

We can analyze the halos of our systems using the same tools that were used in chapter 4. The effects of the halo on the disk can be investigated in a variety of ways, the most
interesting of which is to analyze the disk as a simulated observer. By analyzing the disk as an observer we can verify the results of [Hayashi and Navarro (2006)]. In addition, we will be able to simulate observations of LSBs, which may reveal some interesting properties.

In order to simulate an observer, we must make a projected image of the disk based on the simulated observer’s position. Then, using the line-of-sight velocities, rotation curves must to be made for the projected image. In addition, it is necessary to find the observed morphology of the disk so that the ‘observer’ can calculate the apparent inclination of the disk and correct their rotation curves for this inclination.

The projected image of the disk is easily found given an observing position relative to the disk. The observing position gives the inclination of the observer, \( i \), and the viewing angle, \( \theta \). In this work, the viewing angle is defined as \( 0^\circ \) if the observer is along the positive x-axis and \( 90^\circ \) if the observer is along the positive y-axis and the inclination is defined as \( 0^\circ \) if the disk is viewed face-on and \( 90^\circ \) if the disk is viewed edge-on. The system is rotated using \( i \) and \( \theta \) so as to place the observer on the negative y-axis. Then, the projected coordinates \((X_{\text{proj}}, Z_{\text{proj}})\) of each particle are found. Finally, the LOS velocity for each particle is calculated.

The major-minor axis ratios of the projected image can be found using a modified form of Eq. 4.8. The inertial tensor remains the same, except that there are two dimensions instead of three. Therefore, \( r_n \) in Eq. 4.8 becomes

\[
r_n^2 = X_{\text{proj},n}^2 + \left( \frac{Z_{\text{proj},n}}{q} \right)^2,
\]

where \( q \) is the ratio of the minor axis to the major axis. As before, the calculation of the axis ratios can be performed iteratively until the axis ratios of the projected system are found.

The calculation of the rotation curve is relatively simple. First a thin slit is placed along the projected X-axis. The size of the slit is \( 2z_{\text{slit}} \), and all particles within \( \pm z_{\text{slit}} \) are found.
These are then grouped into bins based on $X_{\text{proj}}$. The bulk velocity is found within each cell using

$$v_j = \sum_{i}^{n_j} v_{\text{LOS},i} / n_j,$$

(5.4)

where $n_j$ is the number of particles in each cell. The bulk velocity is used to create the rotation curve.

One might ask whether the bulk velocity actually matches the expected circular velocity for a given system. The bulk velocity is calculated using disk particles which have an asymmetric drift that will lower $v_j$ from the expected value of $v_{\text{circ},j}$. As such, it is important to be able to compare the rotation curves from the generated disks to model rotation curves. This comparison will show whether this method of calculating rotation curves is actually successful.

The model rotation curves can be calculated from the potential of the system. If the potential is known, then the force at some radius is known, and the force can be used to get the circular velocity using

$$v^2_c(r) = -F_r r.$$

(5.5)

In the GalactICS code, the potential of the system is calculated using the initial parameters. We then use this potential to calculate the model rotation curves. The model rotation curve for the initial system is shown in Fig. 5.1. This curve shows that the disk rotation is indeed dominated by the halo.

To check the numerical rotation curves from the disk, three spherical disk-halo models are constructed. They all have the same parameters except for the cuspiness of the halo. One is an NFW type halo with $\gamma = 1$ (which we use as the initial system in Sec. 5.4), one is a cuspy halo with $\gamma = 1.5$, and one is a cored halo with $\gamma = 0.4$. If the disk were massless, the inner slope of the logarithmic rotation curves would be 0.5, 0.25, and 0.8 respectively.
CHAPTER 5. DISK-HALO SYSTEMS

Figure 5.1 The rotation curve of the initial model. The red curve is the total circular velocity, the green curve is the halo component and the blue curve is the disk component. The units of $r$ are kpc and the units of $v$ are 100 km/s.

In this case, the disks are not massless, although they are dominated by the halo. As such, a plot of the logarithmic rotation curves is expected to have those slopes, which is confirmed in Fig. 5.2.

The model curves can be used to check the numerical curves from the disk particles. This comparison is shown in Fig. 5.3 where the numerical curves are calculated using $z_{slit} = 0.05$ viewed at an inclination of $60^\circ$. The numerical curves and the model curves show good agreement in the outer regions of the disk. In the inner regions random motions from $\sigma_r$ and $\sigma_z$ lower the numerical curve below that of the model rotation curves. Based on these results we can trust the numerical rotation curves in the outer regions, but not in the inner $\sim 0.2$ kpc of the disk, as the noise from random motions becomes too large.
Figure 5.2 The logarithmic rotation curves from the analytic calculations. The black curve corresponds to the $\gamma = 1$ model, the blue to the $\gamma = 0.4$, and the red to the $\gamma = 1.5$. The dashed lines are sample lines with slopes equal to 0.5, 0.8, and 0.25 for the black, blue, and red lines respectively. The units of $r$ are kpc and the units of $v$ are 100 km/s.

Figure 5.3 The logarithmic rotation curves from the analytic calculations compared to the numerical calculations. The solid curves are the analytic models while the dashed curves are the numerically calculated rotation curves. Black corresponds to the $\gamma = 1$, blue to $\gamma = 0.4$, and red to $\gamma = 1.5$. The units of $r$ are kpc and the units of $v$ are 100 km/s.
5.4 Results

We apply the M2M algorithm to two identical initial disk-halo systems in order to generate two final systems with prolate halos that have their long axis along the y-axis. One system has target axis ratios of $1 : 0.9 : 0.9$ while the other has target axis ratios of $1 : 0.8 : 0.8$. To check whether we are successful, we plot the axis ratios as a function of time in the upper panel of Fig. 5.4. It is clear from this plot that the systems reach the target constraints within $\sim 100$ time units. Since it has reached the target that swiftly, all further analysis will be performed on the systems at $t = 100$ as the systems are already stable.

Initially the disk is circular in the x-y plane. However, by $t = 100$ the disk is no longer circular. Instead, as the lower panel of Fig. 5.4 shows, it is slightly elongated along the x-axis. In order to clearly show the orientation of the disk relative to the halo, we have generated an exaggerated view of the $x - y$ plane of the system in Fig. 5.5. It is clear that the disk has elongated perpendicular to the long axis of the halo, and that particles moving parallel to the x-axis have larger velocities than those moving parallel to the y-axis. The different orientations of the disk and halo are not unexpected. The long axis of the halo generates a greater force along the y-axis that slows down the particles relative to those moving along the x-axis. Since, the particles moving parallel to the x-axis have a slightly greater velocity, they can move out further along the x-axis than they can along the y-axis.

The fact that the disk is not axisymmetric has implications on the calculation of the inclination from a projected image of the disk. Generally, the inclination of a disk can be found by finding the axis ratios of the projected image. Then, by assuming the face-on disk is circular, the projected axis ratios can be used to find the apparent inclination, $i_{app}$, by

$$i_{app} = \cos^{-1} (q)_{app}.$$  \hspace{1cm} (5.6)
Figure 5.4 Upper Panel: The axis ratios of the halos as a function of time. The black and red curves are for the $1 : 0.9 : 0.9$ halo, while the blue and cyan curves are for the $1 : 0.8 : 0.8$ halo. The black and blue curves are the major axis ratios, $X/Y$, and the red and cyan curves are the minor axis ratios, $Z/Y$. The thin vertical lines are when M2M begins being turned off and when it is completely turned off.

Lower Panel: The $Y/X$ axis ratio of the disks as a function of time. The black curve is for the disk in the $1 : 0.9 : 0.9$ halo and the blue curve is for the disk in the $1 : 0.8 : 0.8$ halo. The thin vertical lines are when M2M begins being turned off and when it is completely turned off.
Figure 5.5 An exaggerated view of the orientation of the long axis of the disk relative to the long axis of the halo as viewed in the $x - y$ plane. The arrows on the disk show the velocity vectors of the disk. These show that the velocities are greater along the $x$-axis than the $y$-axis.

The inclination will be used to correct the rotation curve to the 'true' velocity by

$$v = \frac{v_{\text{observed}}}{\sin(i_{\text{app}})}.$$  \hfill (5.7)

However, since the face-on disk is not circular, the calculation of the inclination will be incorrect. In addition, the viewing angle will change the projected shape of the disk. This means that the measured inclination will change as an observer moves around the disk.

In order to demonstrate this clearly we show a rough contour map of the face-on disk of the $1 : 08 : 08$ system in Fig. 5.6 and in Fig. 5.7 we show rough contour maps of the projected image of the $0.80.8$ disk at inclinations of $30^\circ$ and $60^\circ$ when the observer is along the $x$-axis and when the observer is along the $y$–axis. We can see quite clearly in Fig. 5.6 that the disk is only slightly elongated. However, this slight elongation causes significant differences in the axis ratios of the projected disk depending on the location of the observer relative to the long axis of the disk. The effect of the changing projected axis ratios on the inclination is shown in Fig. 5.8. We can see that the inclination errors for
the nearly edge-on disks are almost negligible. This is not unexpected as the slight change in projected morphology with viewing angle will not hide the fact that the disk is nearly edge-on. However, when the disk is viewed nearly face-on the inclination errors are quite significant. This means that the inclination-corrected rotation curves for nearly face-on LSB galaxies may have large errors due to the shape of the DM halo.

The non-axisymmetric nature of the disk indicates that the velocity dispersion must also be non-axisymmetric. In order to investigate the changes to the velocity profile, we measure the rotation curves of the two disks at various viewing angles. The disks are observed at an inclination of 60° and are corrected by the true inclination. The rotation curves are shown in Fig. 5.9 with the model curves for the cuspy, NFW, and cored halo systems as a comparison.

An examination of the rotation curves of the two systems leads to two conclusions. Firstly, there is an intrinsic difference in the peak velocity of the rotation curve depending
Figure 5.7  Contour maps of the projected images of the disk in the 1 : 0.8 : 0.8 system. A) shows the image of the disk for an observer located at an inclination of 30° viewing the disk along the $x$-axis, while B) shows the image of the disk for an observer at the same inclination, viewing the disk along the $y$-axis. C) shows the image of the disk when viewed at an inclination of 60° along the $x$-axis, and D) shows the image of the disk when viewed at an inclination of 60° along the $y$-axis. The images of the disk viewed along the $y$-axis appear significantly more elongated than those observed along the $x$-axis. The $x$ and $z$ axis are units of projected kpc.
Figure 5.8 The inclination correction as a function of the viewing angle in degrees. The upper panels show the ratio of $i_{\text{app}}$ to $i_{\text{true}}$ and the bottom panels show $\frac{\sin(i_{\text{app}})}{\sin(i_{\text{true}})}$. The left-hand panels are for the disk in the $1:0.9:0.9$ halo, while the right-hand panels are for the disk in the $1:0.8:0.8$ halo. The black curves are the measurements when the disk is viewed at a true inclination of $60^\circ$ and the red curves are for the disk viewed at a true inclination of $30^\circ$. 
CHAPTER 5. DISK-HALO SYSTEMS

Figure 5.9 The rotation curves for the two disks compared to the model rotation curves. The left-hand panels are for the disk in the $1 : 0.9 : 0.9$ halo, while the right-hand panels are for disk in the $1 : 0.8 : 0.8$ halo. The upper panels are the logarithmic rotation curves, while the bottom panels are the physical rotation curves with $r$ measured in kpc and $v$ measured in 100 km/s. Each curve is observed at inclination of $60^\circ$. In each panel the solid black curve corresponds to the $\gamma = 1$ model, the blue to the $\gamma = 0.4$, and the red to the $\gamma = 1.5$. The dashed black curve are what an observer on the x-axis would see, the red at $\theta = 45^\circ$, and the blue at $\theta = 90^\circ$. 

A) 

B) 

C) 

D)
on the viewing angle of the observer. The largest velocity is seen when the observer is
located parallel to the long axis of the disk and the smallest when the observer is along the
minor axis of the disk. This result is expected when one considers the morphology of the
disk. While it seems that the velocities fall off to the same value, in the outer regions the
particle number is low, making the rotation curves untrustworthy. The intrinsic difference
in the peak velocities of the rotation curves is very important for mass estimates. If an
observer is using the rotation curve to estimate the mass of the system based on assumptions
of spherical symmetry, they will arrive at different mass estimates simply due to observing
the system at different angles. So, even if they manage to get the correct inclination, the
mass estimate of a disk in a prolate system will depend on how the observer is aligned to
the disk.

The second thing that can be noted from the logarithmic rotation curves is that the
inner slope seems to be constant with viewing angle. No matter where the observer is,
the inner slope of the logarithmic velocity curve is the same. It also appears that the inner
slopes of both disks match the initial NFW profile. However, this is difficult to say without
fitting some velocity function to the curves. Nonetheless, if there is any change in the inner
slope, it would appear to be much smaller than the results of Hayashi and Navarro (2006).
The differences between our results and their results may be explained by the inclusion of
disk self-gravity in our simulations, or the fact that we evolve to a non-axisymmetric disk
rather constructing a non-axisymmetric disk based on the halo potential. One method to
distinguish between these possibilities is to create a disk-halo system with a massless disk.
If the inner portions of the rotation curves of the massless disks have the differing slopes,
that would indicate that the disk self-gravity causes the discrepancies between our results
and those of Hayashi and Navarro (2006).
CHAPTER 5. DISK-HALO SYSTEMS

Figure 5.10 The logarithmic rotation curves of the disk in the $1 : 0.8 : 0.8$ halo when viewed at an inclination of $30^\circ$ and corrected for the apparent inclination compared to the cored, NFW, and cuspy analytic curves. The black solid curve corresponds to the $\gamma = 1$ model, the blue to the $\gamma = 0.4$, and the red to the $\gamma = 1.5$. The black dashed curve are what an observer on the x-axis would see, the red at $\theta = 45^\circ$, and the blue at $\theta = 90^\circ$.

The rotation curves shown in Fig. 5.9 are, in a certain sense, the intrinsic rotation curves. While they depend on the viewing angle, they have been corrected by the true inclination. However, from Fig. 5.8 it is clear that an observer would see a different inclination than the true inclination. It would be informative to see the rotation curves as the observer would see them. In order to show the most extreme case the logarithmic rotation curve of the disk embedded in the $1 : 0.8 : 0.8$ halo observered at an inclination of $30^\circ$ is shown in Fig. 5.10. The rotation curves have been corrected for the observed inclination, and it is clear that the effects on the different intrinsic velocity and the inclination error are profound.

These wildly different rotation curves will lead to very different estimates of the mass of the system. If the observer were to estimate the mass assuming spherical symmetry and
circular velocities using

\[ v(r)^2 = \frac{M(r)}{r}, \] (5.8)

where the gravitational constant has been set to one, they will find drastically different results depending on their viewing angle. A rough mass estimate for the $1 : 0.8 : 0.8$ system is shown in Fig. 5.11. This plot shows that the mass estimates from an observer along the x-axis is between two and three times more massive than the mass estimate of an observer along the y-axis.

The variation in the mass estimates shown in Fig. 5.11 seem quite incredible. However, this is the worst case scenario of the simulations that we have run. If the disk is observed nearly an edge-on, there would not be nearly as large of a difference. In addition, if the halo were less prolate, the difference would also be much smaller. Finally it is important to note that observers generally measure rotation curves for systems that are closer to edge-on, as they have less noise. It is impossible to measure the rotation curve of a face-on galaxy, and it is also difficult to measure the rotation curve of an edge-on galaxy. Thus, rotation curves are only measured for a range of inclinations, ranging from $\sim 25^\circ$ to $\sim 75^\circ$. Nonetheless, these mass estimates do illustrate a very important point. They show that the rotation curves of nearly face-on LSB’s are not the best estimates of the DM halo mass as the shape of the halo leads to large errors in the mass estimates.
Figure 5.11 The mass estimate from the observed rotation curves of a disk in a 1 : 0.8 : 0.8 halo observed at 30° and corrected for the observed inclination. These mass estimates are compared to the mass estimate from the analytic rotation curves for the cored, NFW, and cuspy models. The solid black curve corresponds to the $\gamma = 1$ model, the blue to the $\gamma = 0.4$, and the red to the $\gamma = 1.5$. The dashed black curve are what an observer on the x-axis would see, the red at $\theta = 45^\circ$, and the blue at $\theta = 90^\circ$. The units of $r$ are kpc and the units of mass are $2.32 \times 10^9 \, M_\odot$. 
Chapter 6

Discussion and Conclusion

In this work we have successfully created a dynamically self-consistent implementation of the M2M algorithm. Using this algorithm we generated isolated triaxial halos and disk-halo systems with prolate halos. We were able to show that our dynamically self-consistent M2M algorithm can, in fact, generate a stable equilibrium system with a non-spherical halo without knowledge of the potential or an analytic form of the distribution function. Unfortunately our algorithm requires a large number of particles and the computational time is quite large. We may be able to reduce the computational time required by our algorithm by parallelizing our M2M algorithm. Nonetheless, our algorithm is still quite useful as it can be used to generate equilibrium systems with arbitrary morphologies.

The most important science results from this work came from our simulations of disk-halo systems with prolate halos. These simulations show that the mass estimates from rotation curves of DM dominated disks vary depending on the observers position. A prolate halo changes the intrinsic peak velocities as well as the morphology. If the disk is nearly face-on, the non-circular morphology can cause large errors in the calculation of the inclination of the disk. The inclination error is correlated with the change in the intrinsic
peak velocity, causing differences in the mass estimation of up to a factor of three. These results suggest that any mass estimates from the rotation curves of nearly face-on LSB’s must be considered very carefully. Since the mass estimate can vary greatly depending on the disk’s orientation, it is important to attempt to measure the mass of nearly face-on LSB’s using methods that do not depend on the rotation curve.

A fully self-consistent M2M algorithm that is capable of modifying one or all components of a galactic model is quite useful. The main application is generating multi-component systems with non-spherical halos. However, there are many potential other applications, such as generating models of the Galactic Bulge that match the observed surface brightness and velocity dispersion, or generating spherical systems with a given anisotropy. Another possible application is the generation of a slowly tumbling halo, possibly with an embedded disk. In addition, our algorithm can use real observations as constraints in order to construct models of actual galaxies.

The logical next step to this work is to use our M2M algorithm to model real LSB galaxies using non-spherical halos. These models will help constrain the uncertainties that arise from the non-sphericity of the DM halos. In addition, it may also be interesting to investigate the effect of the halo shape on more common galaxy types. It is possible that the presence of a bulge will keep the inner region of the disk circular, and a larger disk self-gravity may prevent changes in the disk morphology. It would be interesting to see how large a disk and bulge (relative to the halo) would be needed in order to make the effects of a non-spherical halo negligible. At this point all we know is that if galaxy has no bulge and is DM dominated, then the shape of the halo has a large effect on the disk.

Overall this work has been very successful in developing a new and very useful tool for modelling galaxies. We are able to develop multi-component galaxies with non-spherical
halos of arbitrary morphologies. In addition we have found that the effects of the halo shape on the disks of DM dominated galaxies are profound and can affect observations of the systems considerably.
Bibliography


