

Computer Simulation of Dark Matter Effects on Galaxy Rotation

New Mexico
Supercomputing Challenge
Final Report
April 5, 2011

Team Number 72
Los Alamos Middle School

Team Members

Cole Kendrick

Teachers

Brian Kendrick

Project Mentor

Brian Kendrick

Summary

The main goal of this project is to develop a computer program to model the rotation of a galaxy including dark matter. The computer program will be used to answer these questions: (1) How does dark matter affect rotational curves in galaxies; (2) how accurately can this be modeled; (3) what will happen when the dark matter and galaxy masses are changed; and (4) How well can this method work for different galaxies. The computer program was built-up over time. The first step was writing a computer program using Python to model a mass attached to a spring. This model was extended to two-dimensions and then to a solar system model with both sun and interplanetary gravity interactions. Finally, the galaxy program was constructed using C by adding hundreds of “stars” to my solar system model in place of the planets and replacing the sun’s mass with a large central galaxy “core” mass. Dark matter is implemented in this model by treating it as an additional large mass point located at the center of the galaxy. Newton’s laws of motion were solved using a velocity Verlet method. The forces due to gravity were computed using two different methods: (1) nearest neighbor, a method that decreases calculation time by drawing a circle around each star and seeing whether a star is inside or not. Only the force due to gravity between the star and its neighbors will be computed (with the exception of the central mass); and (2) N-body, a method that computes the gravitational force between the current star and every other star in the galaxy and the central mass. The N-body approach is very slow. Both methods gave the essentially the same results but nearest neighbor is much faster and capable of using more stars. Five different calculations were run for the Andromeda galaxy. Nearest neighbor with dark matter, without dark matter, and less dark matter using 4400 stars, and N-body with and without dark matter using 520 stars. Two other galaxies NGC 2403 and NGC 3198 were also modeled using 4400 stars with dark matter. From these simulations I was able to successfully match the experimental data measured for Andromeda, NGC 2403 and 3198. My results show that dark matter is needed to maintain a stable galaxy. Dark matter also causes the galaxy to rotate much faster so that the rotational velocity remains constant out to the edge of the galaxy. These “flat” rotational curves are experimentally observed for nearly all spiral galaxies.

Table of Contents

Introduction.....	4
Problem.....	5
Galaxy Model	6
Results	7
OpenCL.....	21
Conclusion.....	22
Future Work.....	22
References.....	24
Appendix 1.....	25
Appendix 2	27

Introduction

What is dark matter and why do we need it?

No one knows what dark matter really is, although it has been one of the many theories that exist to help explain the flat rotational curve of all galaxies. Dark matter was first introduced in the 1930's but many people did not really support it until later in the

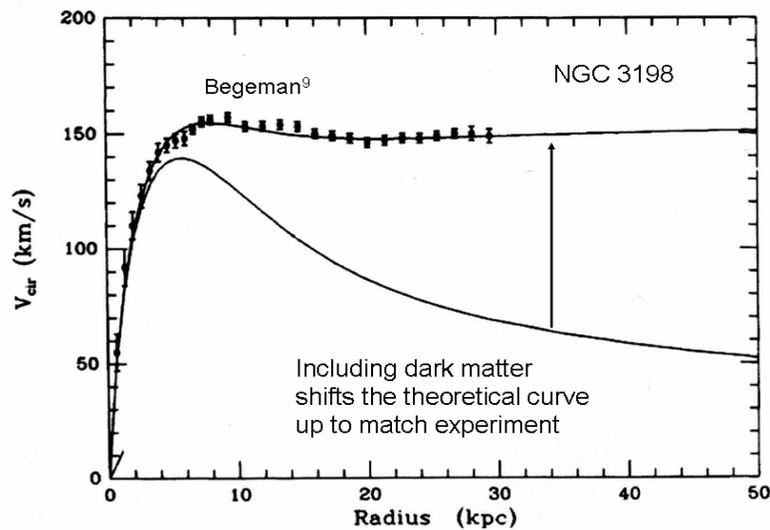


Figure 1: Rotational curves and experimental data for NGC 3198

1970's. In the 1970's people had begun collecting experimental data from galaxies by taking the brightness of the galaxy. They also made computer programs to test the dark matter theory. According to Newton's laws of motion, a galaxy rotation curve should dip down towards the edge of the galaxy. However, the experimental curve is relatively flat (see Fig. 1). Dark matter is one of the theories to resolve the difference between the two curves. Another theory that exists is called MOND (Modified Newtonian Dynamics) which changes Newton's laws of gravity to fit the flat rotational curve. By using the dark matter and MOND theory we can better understand how galaxies function and learn more about our tiny section of the universe.

How are galaxy rotational curves measured?

Galaxy rotational curves are measured from the Doppler shifts of the Hydrogen 21 cm line. The shift is broken down into red and blue, the red shift is going away and blue shift is coming towards you.

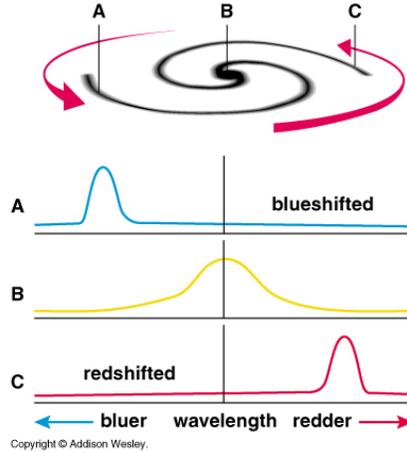


Figure 2: Schematic showing the blue and red Doppler shifts which are used to measure a galaxy's rotational velocity.

Problem

Can I successfully create a computer model to simulate dark matter effects on a galaxy's rotation? How accurately can I model the effects? These are the two primary questions that my project will address. The dark matter model is based on the Navarro-Frenk-White mass distribution. It is a static, spherically symmetric mass distribution (see Figure 3 below). Newton's law of motion $F=ma$ (total force = mass times acceleration) will be used to move the visible matter ("stars") in my simulations. The force is due to gravity acting between all of the visible matter plus a large central force at the center of the galaxy due to dark matter. The force between any two mass points m_1 and m_2 is given by Newton's law of gravity $F=G \frac{m_1 m_2}{r^2}$ where G is the universal gravitational constant and r is the distance between the two points. My simulation will be based on the Python and C programming languages.

Dark Matter Model

Dark matter mass distribution (the blue sphere) is static, spherically symmetric, and surrounds entire galaxy

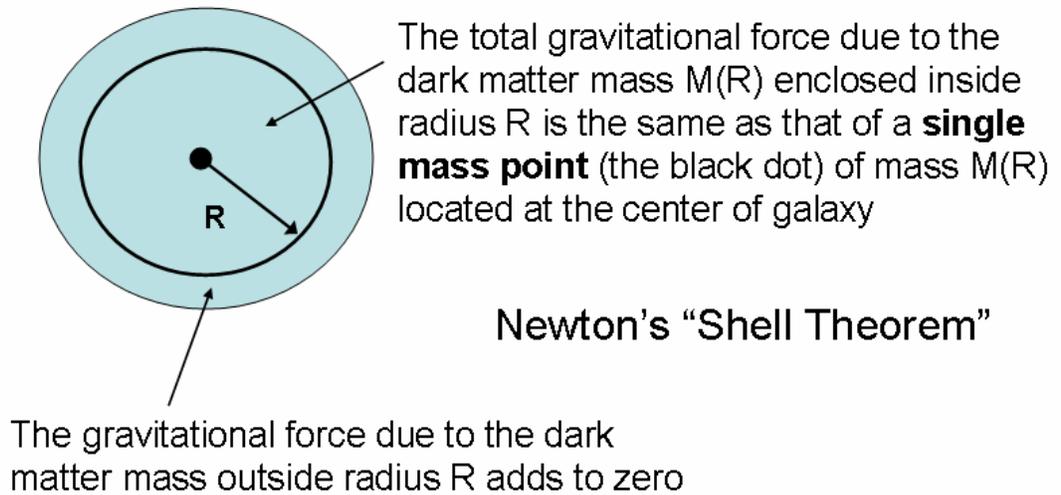


Figure 3: Schematic showing how dark matter is treated in the computer model.

Galaxy Model

My galaxy model is very similar to our solar system, instead of planets orbiting the sun, stars orbit the dark matter and black hole of the galaxy. Also, instead of 8 planets, there are trillions of stars. My galaxy model takes those stars and represents many stars as one because doing a trillion stars on single computer will take forever. My galaxy model calculates the mass of the stars by taking the total mass of the core or disk (M_{core} or M_{disk}) and dividing that by the number of stars in the core ($N - N_0$) or disk (N_0). The total number of stars is N . The formula in my model is $M = M_{\text{disc}}/N_0$ or $M_{\text{core}}/(N - N_0)$ to get the mass of a star. The figure 4 below shows how my galaxy model is similar to a solar system.

Galaxy Model

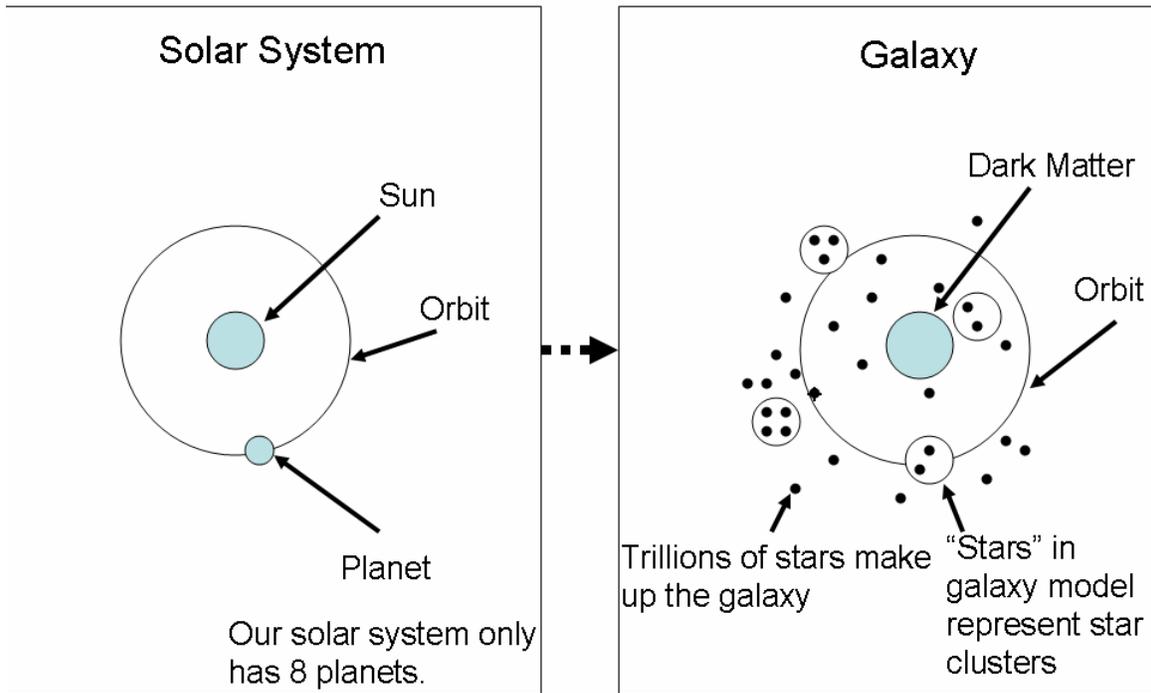


Figure 4: Schematic showing how my galaxy model resembles the solar system.

Results

Eleven simulations were run using nearest neighbor and N-body methods both with and without dark matter. Nearest neighbor creates a radius around each star and determines whether or not another star is in that radius. If the star is inside of the radius then the program stores that stars ID in an index and computes the gravitational force between it

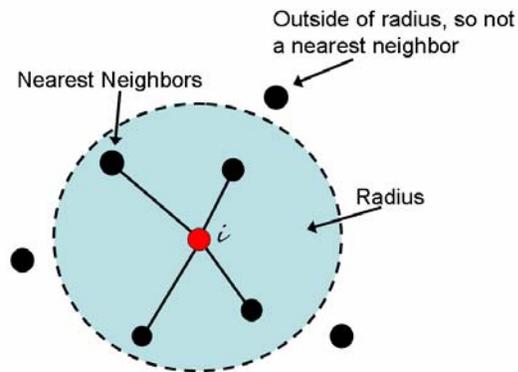


Figure 5: Schematic showing how the nearest neighbor method works.

and the star i (red point). If the star is not inside the radius, then the program does not compute the gravitational force between it and the star (see Figure 5). The force due to the central mass is included plus all of the points outside the nearest neighbor radius and less than the star's radius are grouped together and treated as a single mass point at the center of the galaxy (using Newton's shell theorem). Also, the nearest neighbor radius increases as the distance from the center of the galaxy increases. The typical values varied between 1 and 11 kpc so that the number of neighbors was always around 100. The N-body method calculates the gravitational force between itself and every single star. Since the N-body method is so computationally demanding, the N-body simulations have only 520 mass points. Nearest neighbor is a much more efficient because it speeds up the calculation greatly compared to N-body. Figure 6 below shows how the CPU time scales for N-body and nearest neighbor methods.

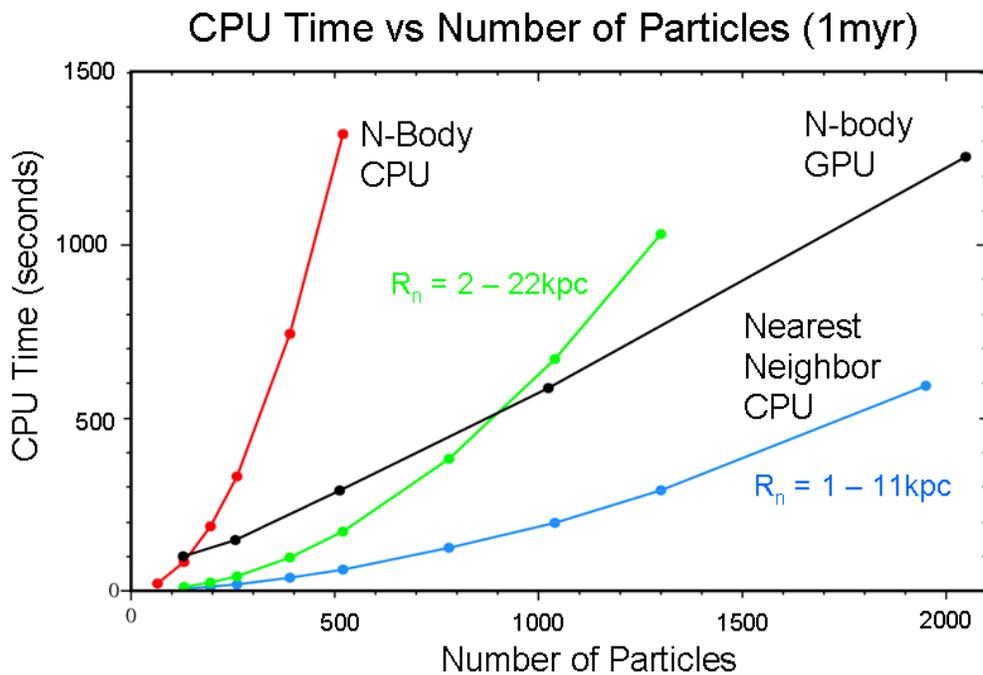


Figure 6: CPU time vs the number of particles for N-body and nearest neighbor methods for a 1 million year simulation with a time step $dt = 10$ years. R_n is the nearest neighbor radius which increases linearly with the distance from the center of the galaxy.

Nearest neighbor gave similar results as N-body simulations did and was more stable without dark matter than N-body because of its 4400 mass points. Although nearest neighbor is not a standard method for gravity computations there are similar methods like the tree algorithm, which are designed to work especially for galaxy related objects.

The initial mass distribution was chosen based on the experimentally measured visible light intensity which is well described by an exponential function (see Ref. [1]). The initial velocities for each star were chosen based on setting the centripetal force for uniform circular motion equal to the gravitational force: $m_1 v^2 / r = G m_1 M / r^2$ where M is the total mass enclosed within the star's radius r (it includes both the galaxy's visible and dark matter). A small random velocity was also added to the initial v (see Ref. [1]). Newton's equations of motion were then solved as a function of time using the velocity Verlet method. Other methods were also tried (see Appendix 1). The flow chart of my galaxy simulation program is shown in Figure 7 and the corresponding Python code is in Appendix 2. In order to speed up the calculations, Shedskin was used initially to convert the Python code to C but the translated C code was not always compatible with other subroutines (especially OpenCL). So, the galaxy code was rewritten in C and the C version was used in all of my final galaxy simulations.

Flow Chart of Main Computational Loop

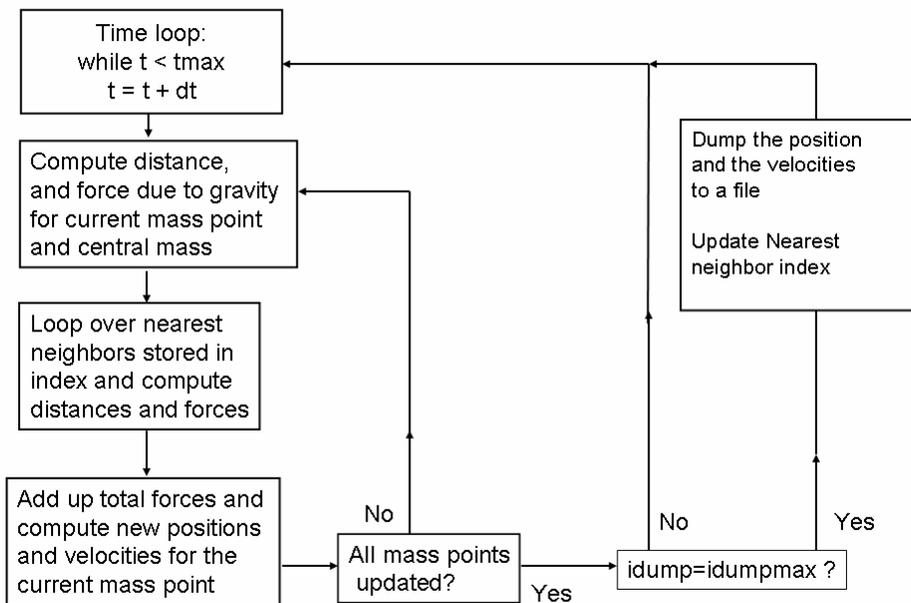


Figure 7: Flow chart of the main computational loop used in all simulations to calculate the force due to gravity and store the data every million years.

Nearest neighbor results are plotted in Figs. 8 and 9 for the Andromeda galaxy simulation. The results with dark matter show that dark matter causes the galaxy to rotate faster (red points) because of the large mass in the center. Nearest neighbor results without dark

matter shows that the galaxy rotates slower (black points) than dark matter and does not have as much structure. The N-body results plotted in Figs. 10 and 11 with dark matter show essentially the same results as nearest neighbor which proves that the nearest neighbor method works respectably. The N-body results without dark matter become unstable quickly due to the smaller number of particles and the lack of a large central mass (no dark matter). In Fig. 11 the galaxy splits into two different galaxies.

Andromeda Simulation WITH Dark Matter

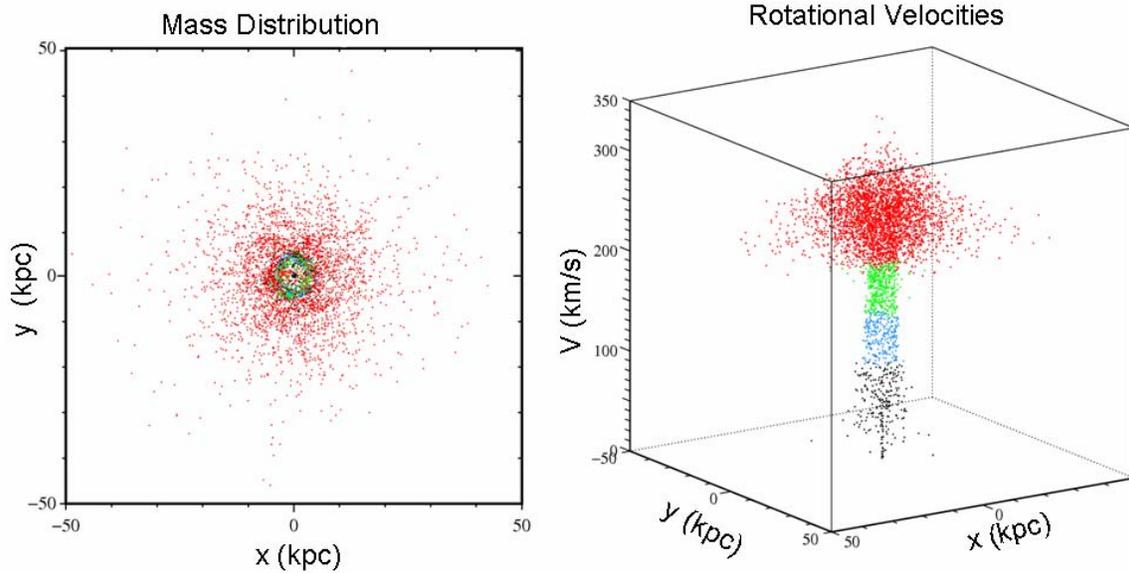


Figure 8: Nearest Neighbor, with dark matter, 1 billion years, 4400 mass points. Red (black) points correspond to high (low) rotational velocities.

Andromeda Simulation NO Dark Matter

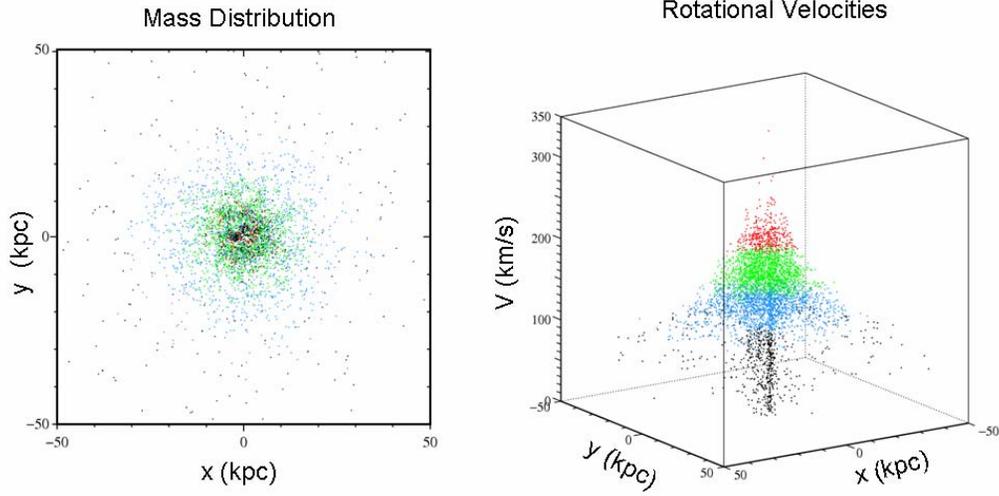


Figure 9: Nearest Neighbor, without dark matter, 1 billion years, 4400 mass points. Red (black) points correspond to high (low) rotational velocities.

Andromeda Simulation WITH Dark Matter

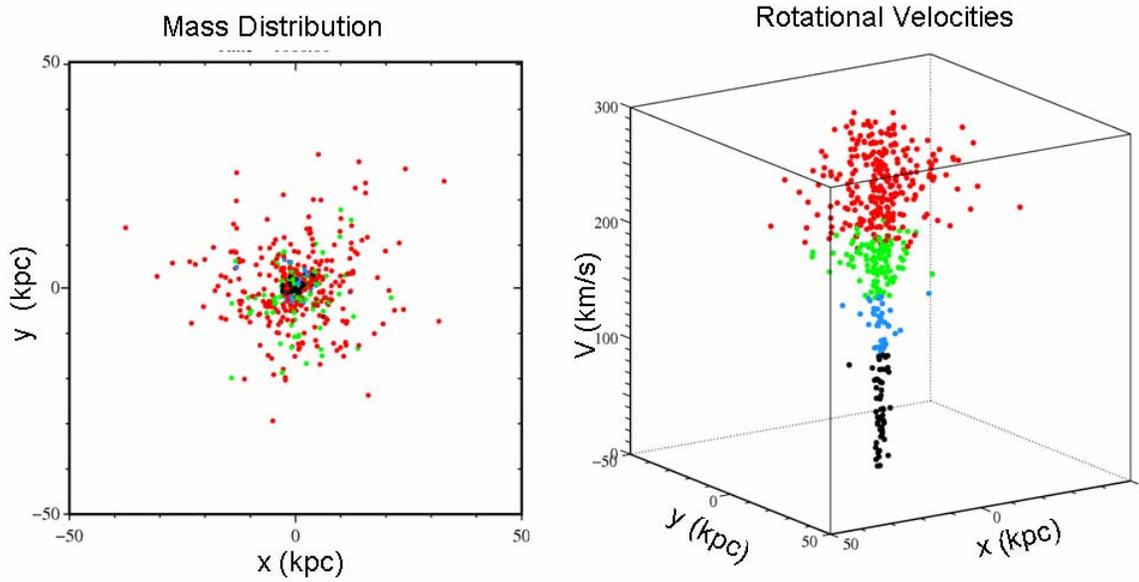


Figure 10: N-body, with dark matter, 1 billion years, 520 mass points. Red (black) points correspond to high (low) rotational velocities.

Andromeda Simulation NO Dark Matter

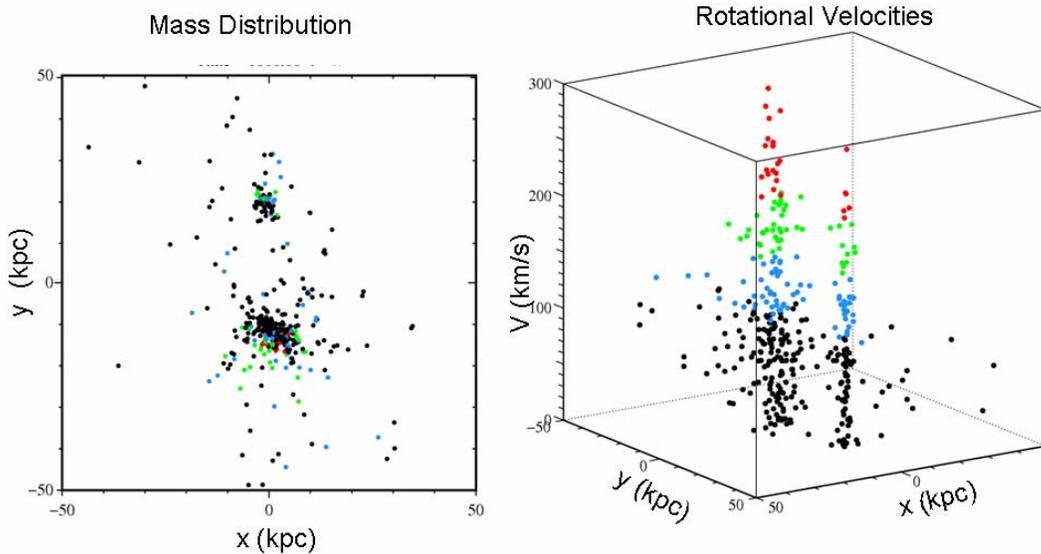


Figure 11: N-body, without dark matter, 1 billion years, 520 mass points. Red (black) points correspond to high (low) rotational velocities.

The rotational curves for each of the nearest neighbor and N-body simulations for Andromeda are plotted in Figs. 12-15. The rotational velocity at a radius r was calculated by adding up the velocities of all of the stars that lie within a range of $r + 0.5$ and $r - 0.5$ kpc and then dividing by the total number of stars in that band. The rotational velocities are plotted at every 10 myr (open squares). The average over all these time steps is plotted as a solid black curve. The experimental results are plotted on the figure in red with error bars. My simulation results with dark matter match the experimental data well. The reduced number of points used in the N-body simulation results (Figs. 14 and 15) cause the scatter in the calculated rotational velocities to increase. The unstable case (Fig. 15) without dark matter has a very large amount of scatter. Figure 15 shows how unstable the N-body simulation is without dark matter. Normally the rotational curve would dip just below the experimental data (red), like in Fig. 13. Since the galaxy split into two different bodies, the rotational curve is way off from the experimental data and where it would normally be without dark matter.

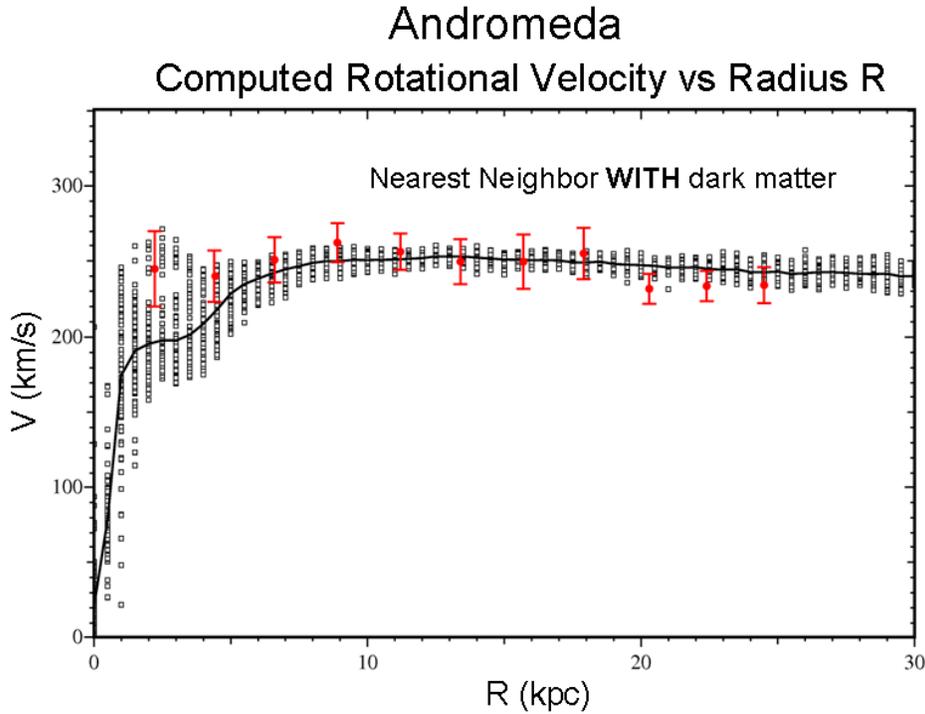


Figure 12: The rotational velocities for Andromeda from the nearest neighbor simulation, with dark matter, 1 billion years, 4400 mass points. The black data points and average solid curve are the calculated results and the red data with error bars are the experimental data.

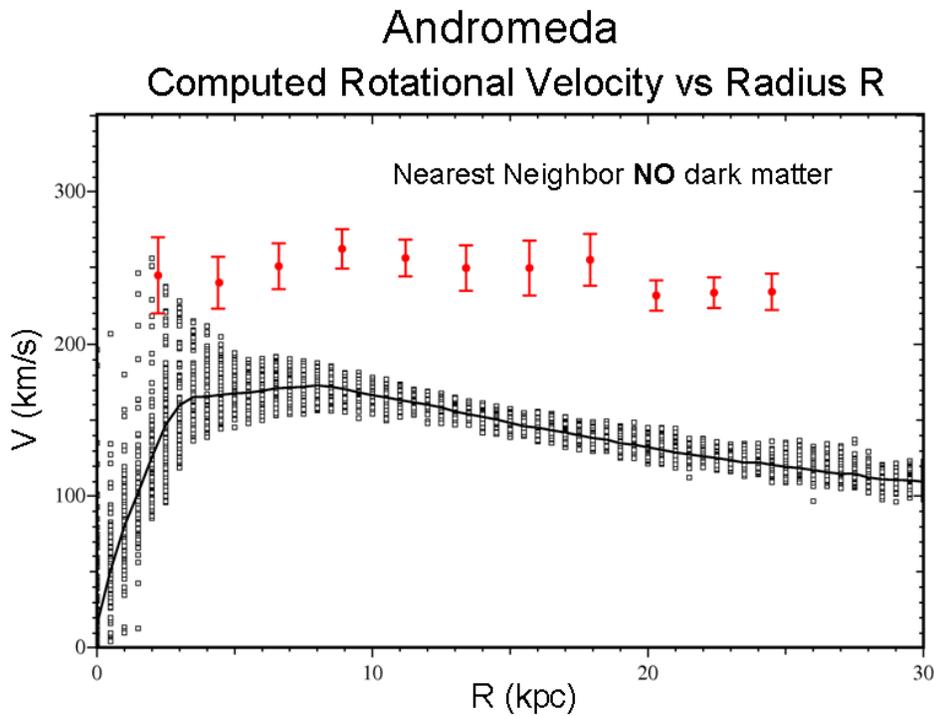


Figure 13: The rotational velocities for Andromeda from the nearest neighbor simulation, without dark matter, 1 billion years, 4400 mass points. The black data points and average solid curve are the calculated results and the red data with error bars are the experimental data.

Andromeda Computed Rotational Velocity vs Radius R

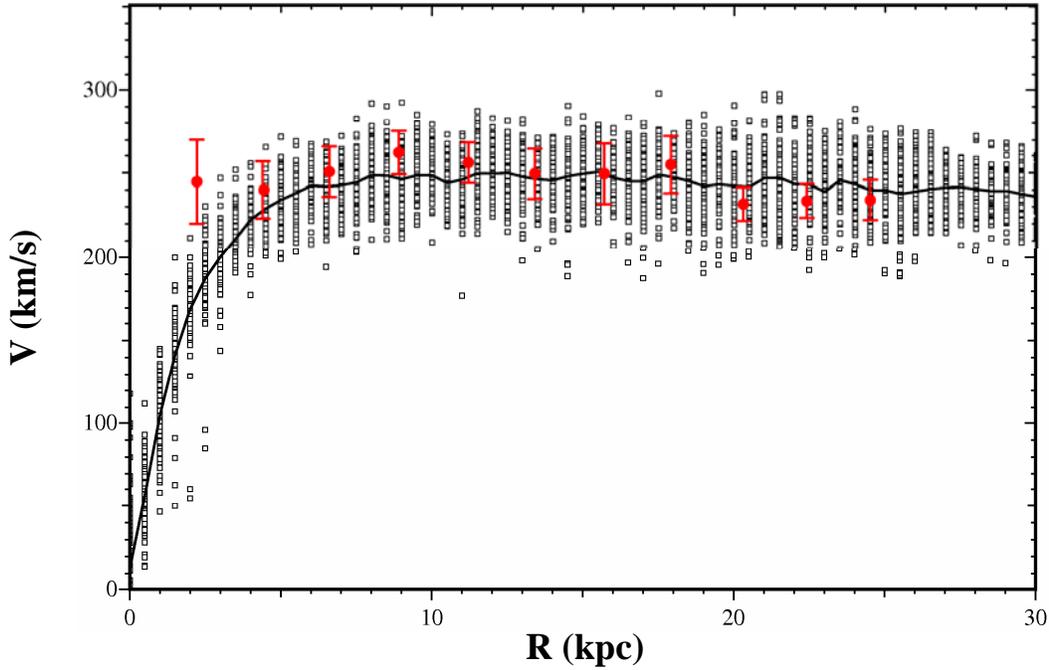


Figure 14: The rotational velocities for Andromeda from the N-body simulation, with dark matter, 1 billion years, 520 mass points. The black data points and average solid curve are the calculated results and the red data with error bars are the experimental data.

Andromeda Computed Rotational Velocity vs Radius R

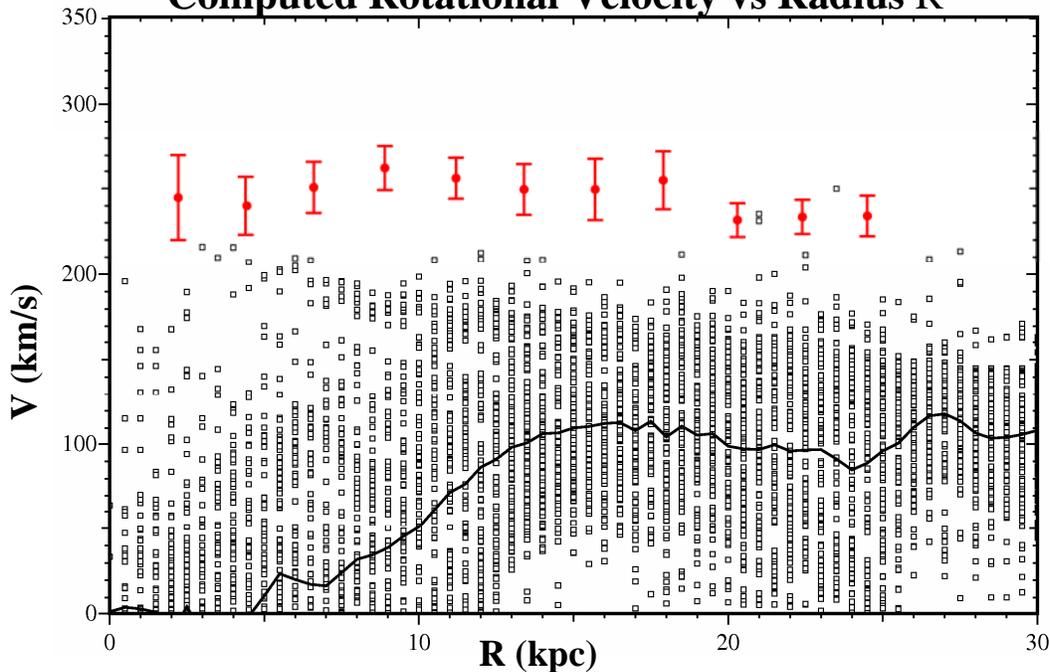


Figure 15: The rotational velocities for Andromeda from the N-body simulation, without dark matter, 1 billion years, 520 mass points. The black data points and average solid curve are the calculated results and the red data with error bars are the experimental data.

The effect of decreasing the amount of dark matter and increasing the disk mass in Andromeda is shown in Figures 16 - 17. The reduced dark matter causes slight instabilities to occur which produces spiral arm structure. The rotational curve in Figure 17 is still in good agreement with the experimental data but this model is ruled out since Andromeda does not have prominent spiral arms.

Andromeda Simulation WITH LESS Dark Matter

Disk mass > Dark Matter mass produces spiral arms

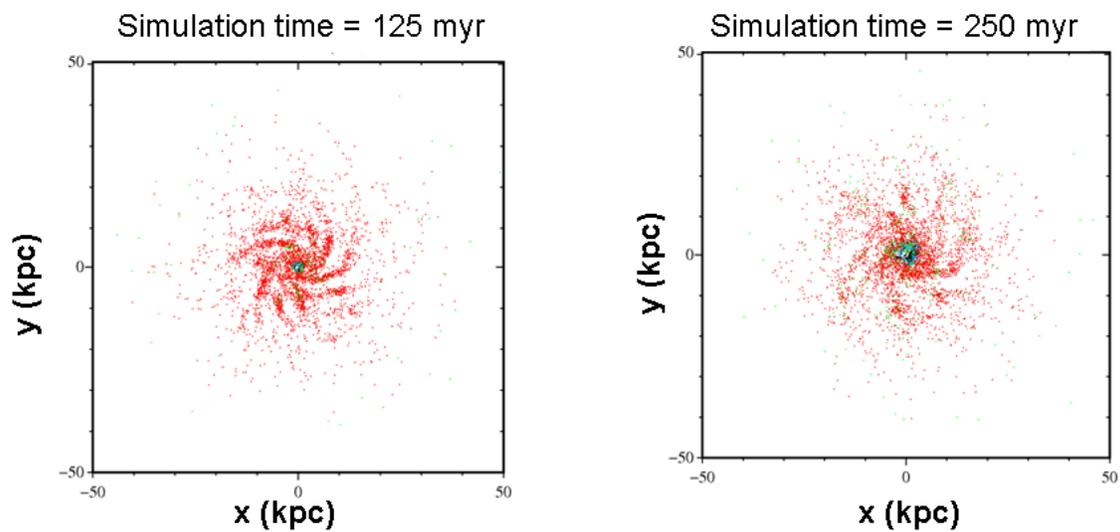


Figure 16: Mass distributions for the nearest neighbor simulation, 4400 mass points. The reduced dark matter and increased disk mass produces spiral arms.

Andromeda

Computed Rotational Velocity vs Radius R

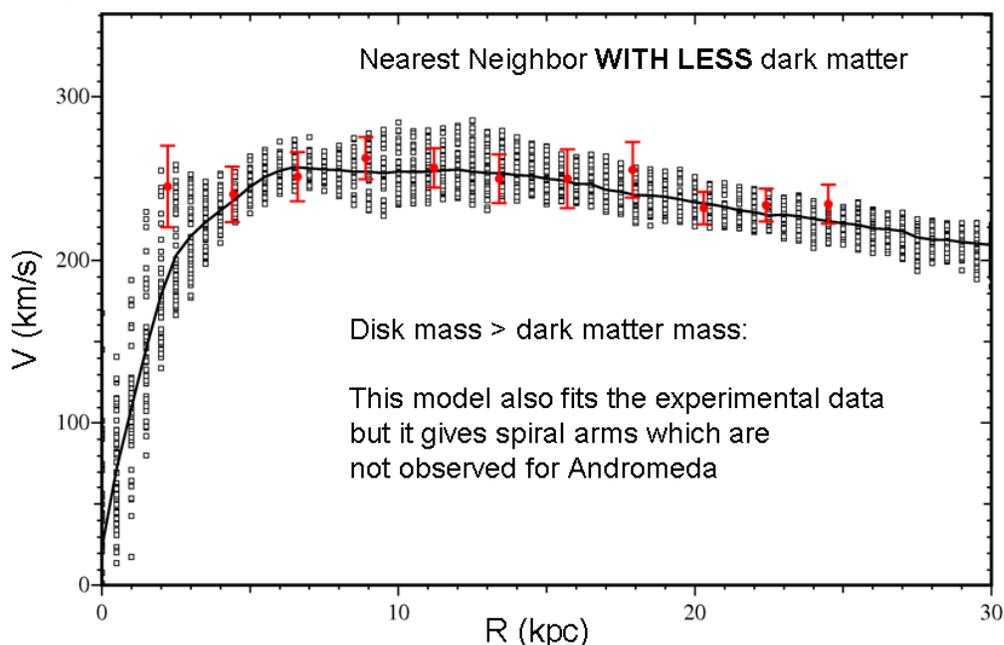


Figure 17: The velocity rotation curve for the nearest neighbor simulation, 4400 mass points with reduced dark matter. The black data points and average solid curve are the calculated results and the red data with error bars are the experimental data.

NGC 3198 & 2403

Additional nearest neighbor simulations using 4400 mass points were run for NGC 3198 and 2403 to test my galaxy simulation code for other galaxies as well. Simulations with these galaxies matched experimental data well. Figure 18 plots the mass distribution of NGC 3198 and 2403. Both galaxies have much smaller mass and therefore rotate slower than Andromeda. In NGC 2403, the core is not packed in because of its small mass. However NGC 3198 has more mass than 2403 so its core is denser. Figures 19 and 20 plot the corresponding rotational curves which are in good agreement with the experimental data (especially 3198). In Fig. 20 the rotational curve from 5 – 10 kpc is shifted down slightly because there may not be enough mass in the core of the galaxy. Additional simulations with increased core mass might correct this.

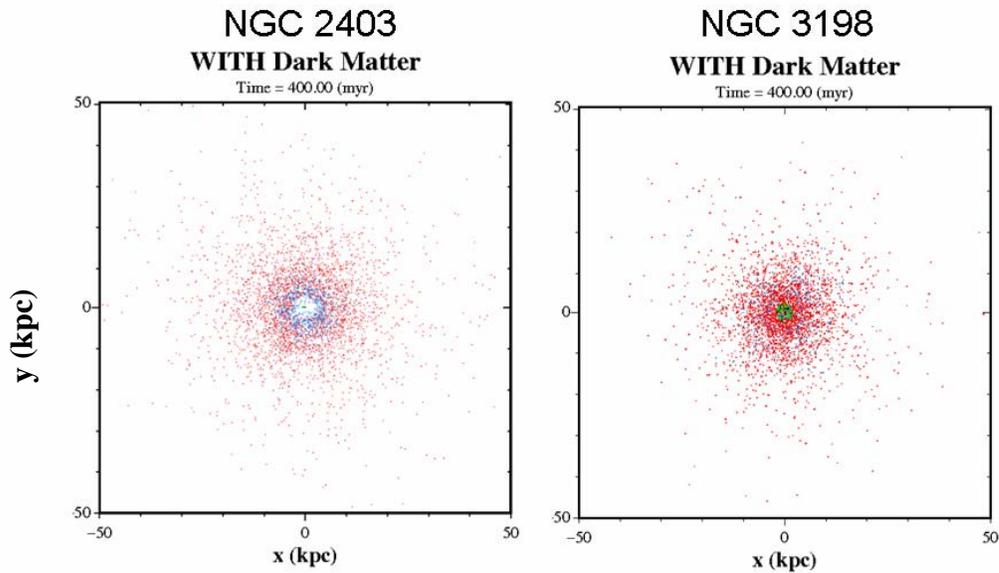


Figure 18: The calculated mass distribution plots for NGC 2403 and 3198 at 400 myr. Red (black) data points represent higher (lower) rotational velocities.

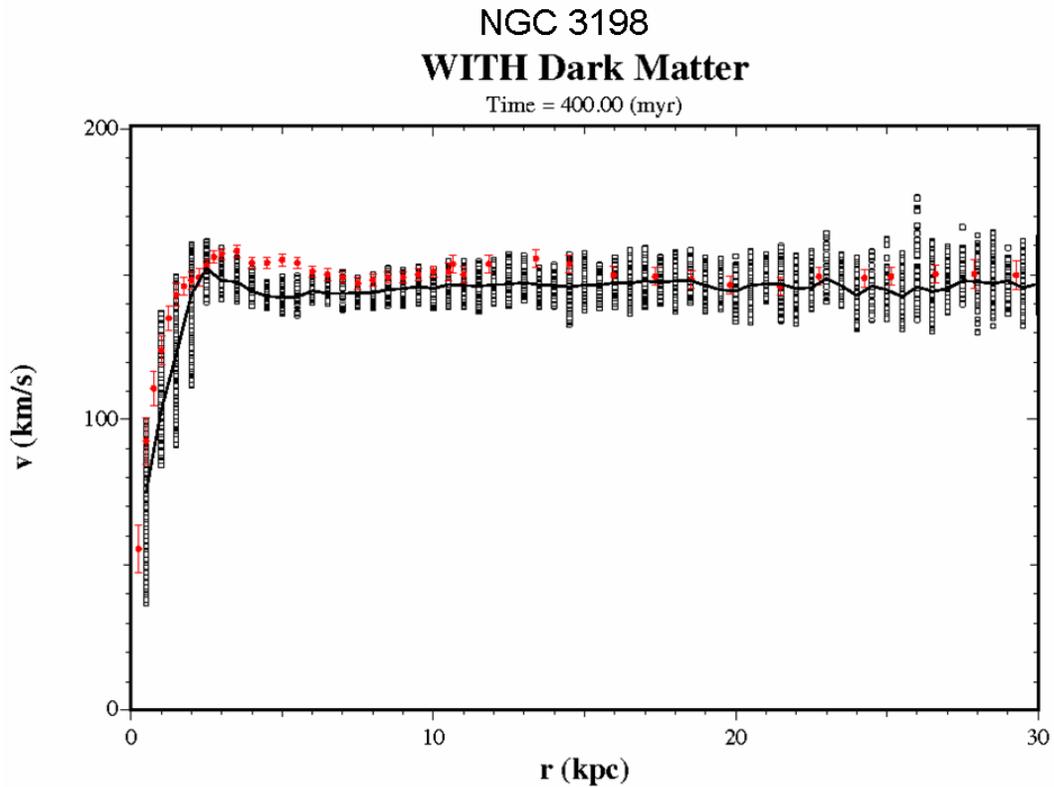


Figure 19: The velocity rotation curve for the nearest neighbor simulation, 4400 mass points with dark matter. The black data points and average solid curve are the calculated results and the red data with error bars are the experimental data.

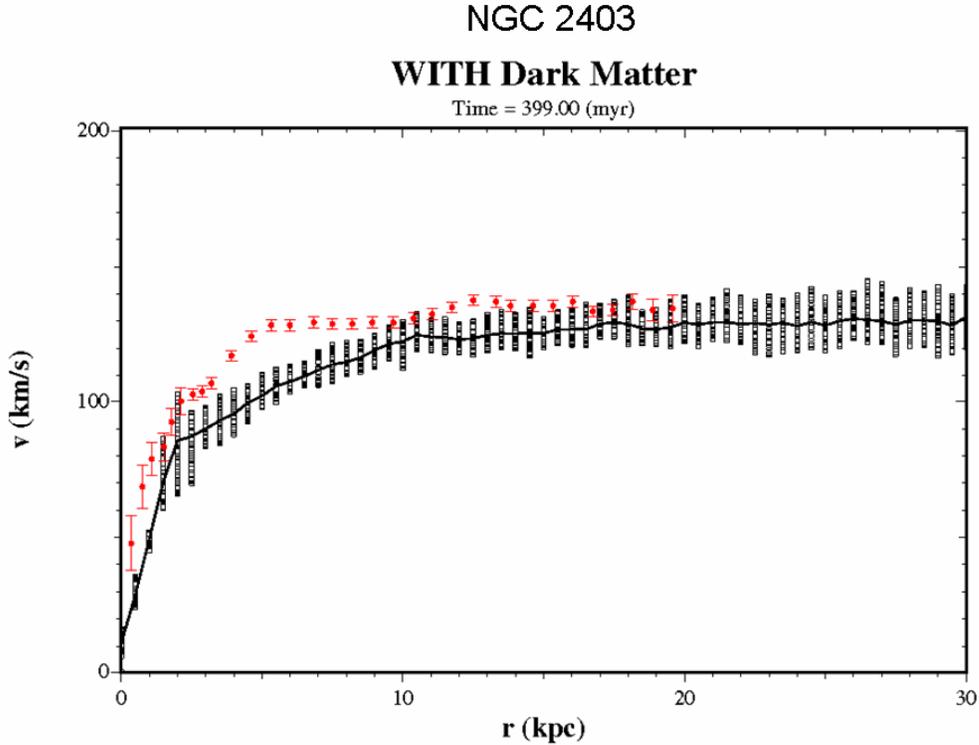


Figure 20: The velocity rotation curve for the nearest neighbor simulation, 4400 mass points with dark matter. The black data points and average solid curve are the calculated results and the red data with error bars are the experimental data.

Parameter Optimization Studies

In both the nearest neighbor and N-body methods a variable called dr_{min} (the force cut off distance) is used to limit the force between two stars. If one star is less than 1 kpc from another star the gravitational force between the two stays the same. This stops 2-body interactions between stars as seen in Figs. 21 and 22 from becoming too large. Several calculations were run to determine which value conserved the most energy. Figure 21 shows that the pink curve ($dr_{min}=1\text{kpc}$) is optimal because it conserves the most energy (E_n/E_0 is closest to 1, E_0 is the initial energy at time zero).

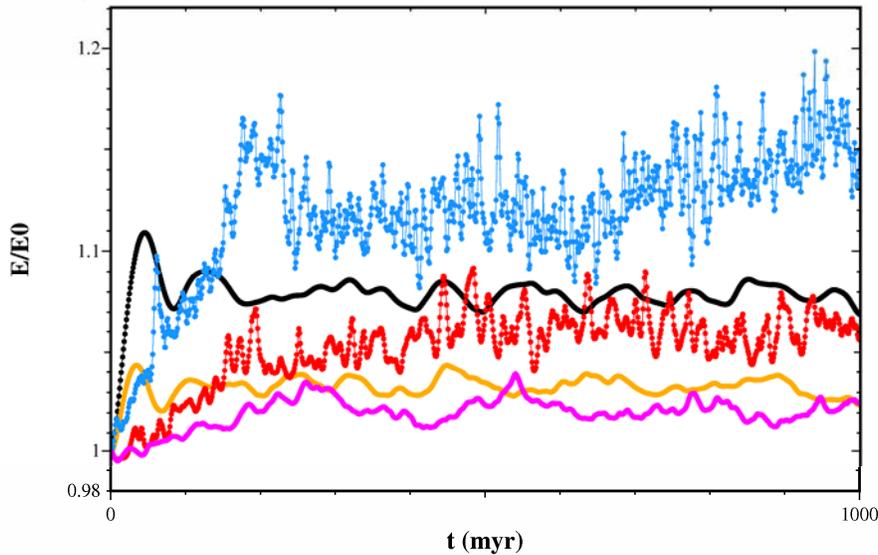


Figure 21: The normalized energy is plotted for different drmin values vs time. Blue – 0.25kpc, Black – 4kpc, Red – 0.5kpc, Orange – 2kpc, Pink – 1kpc (optimal – used in all simulations).

Figure 21 also shows that the fast oscillations in the blue curve (.25kpc) has dominate 2-body interactions which are shown in the figure below (Fig. 22). A drmin value of .25 is not optimal because 2-body interactions cannot be “tracked” accurately enough with a 10yr time step. Also, because of these errors the energy is not conserved as well. The interactions seen below are due to two stars becoming close to each other and the gravitational force greatly increases and causes the stars to become a double star and orbit each other. Drmin is basically like a switch, when two stars get within the drmin distance from each other the gravitational force cannot grow until they move apart again.

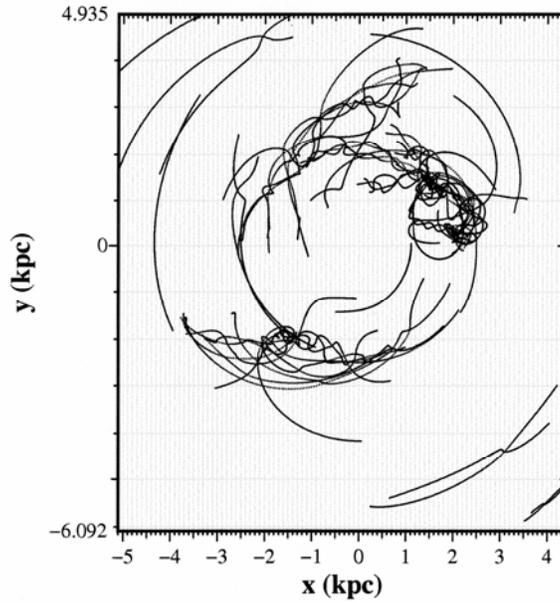


Figure 22: The star trajectories are plotted for the $dr_{min}=0.25$ case. These dominate 2-body interactions take place mainly in the core where stars are closer together.

The optimal time step (dt) was determined by running several calculations with different dt and comparing the positions and velocities to those using a much smaller time step. Figure 23 plots the velocity error as a function of dt . The error decreases quadratically since the velocity Verlet method is a 2nd-order method. An optimal value of $dt=10$ yr was determined and this value was used in all of the simulations. The effect of increasing the nearest neighbor radius was also studied. The results using $r_n = 1 - 11$ kpc were essentially the same as those using $r_n = 2 - 22$ kpc. So $r_n = 1 - 11$ kpc was used in all of the simulations.

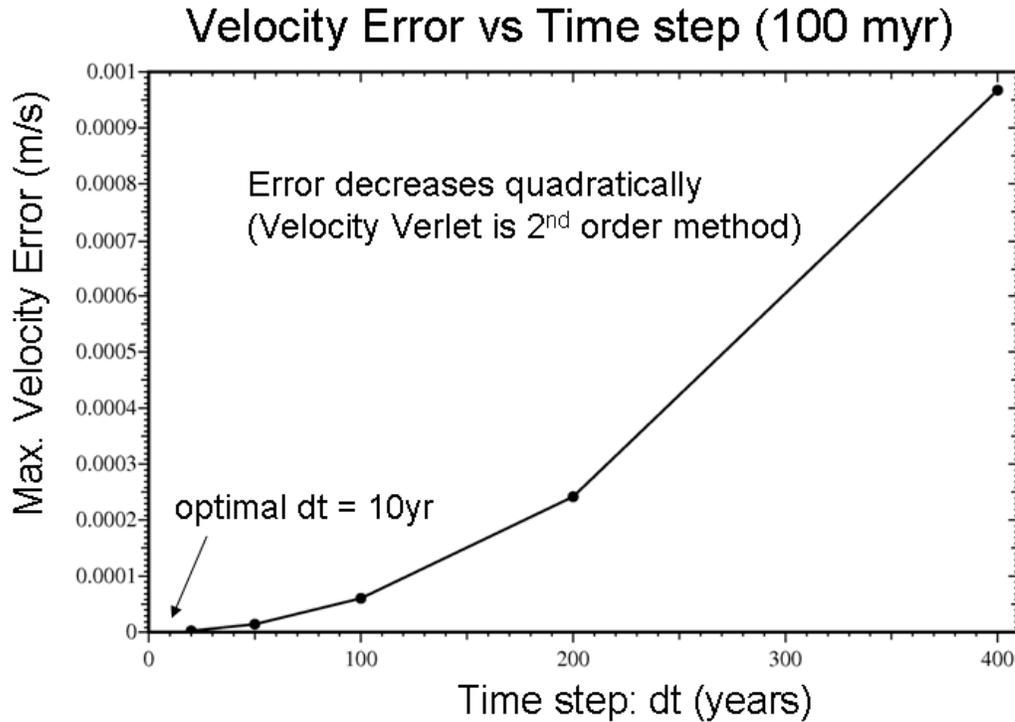
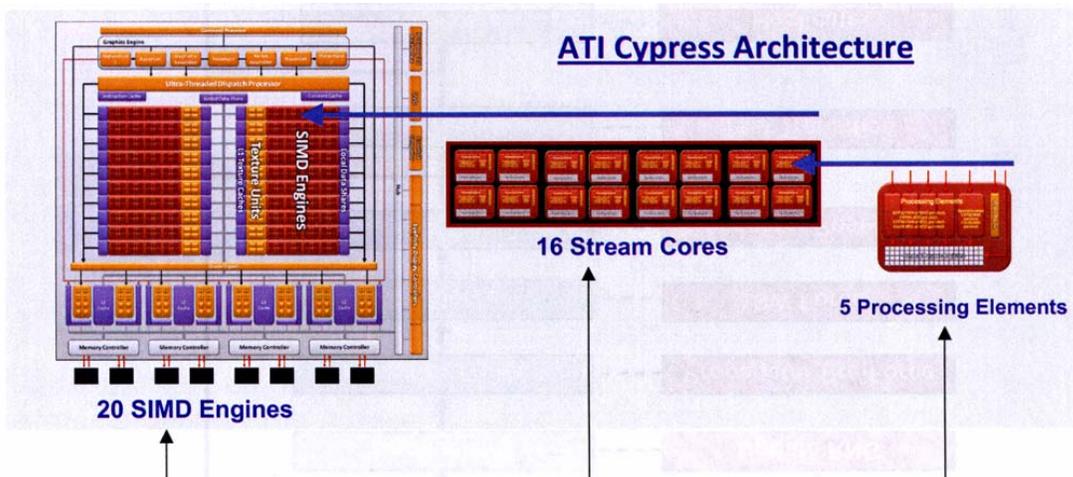


Figure 23: The optimal time step was determined by plotting the velocity error as a function of dt.

OpenCL

OpenCL is a program written by the Khronos group to enable certain programs to be run on GPUs (Graphical Processing Unit). Since graphics cards today have many processors, running a program on a GPU is much more time efficient. OpenCL does not only focus on GPUs however, they can also enable multi-threading (the use of multiple CPU cores) so instead of just utilizing one core you can use two or more. OpenCL works for both Nvidia and ATI graphics cards. The cards used in this project were the ATI Raedon HD4870 (x2) – 1GB vram (512mb global, 512mb local) and the Nvidia GeForce G210M – 512MB (256mb global, 256mb local). The GPUs work by sending kernels (the program you want to run) to all of its threads. The threads then carry out the problem as if it were a CPU. Synchronization can be a problem with GPUs but OpenCL has functions such as global and local memory fences, this waits for everyone to finish to a point (global) or waits for a specific workgroup to finish to a point in the code (local). Each processor has

access to private, local, and global memory. Private memory is up to 100x faster than global memory. This diagram below shows and explains how GPUs are structured and how to OpenCL utilizes them.



View as 20 CPU cores with 16-way threading and SSE for single-precision

(From Brown Deer Technology, "GPU-Accelerated Computing for Chemistry and Material Simulations using ATI Stream Technology", David Richie, 2010.)

When running the galaxy program on the GPUs a factor of 30-100x speed increase was noticed. This means that when running a N-body 512 star run it would take about 16.6 hours to complete. On the CPU however, a N-body 230 star run would take 3 days just for a 230 star run. Both of these benchmarks ran to 1 billion years. However, while running on the GPU may sound very promising currently double precision is not entirely supported so single precision is required for OpenCL. Single precision does cause a lot of error when doing this type of problem. When doing N-body with 4096 stars it takes about 10 minutes per timestep of 1 myr (million years). This means that the GPU can do about 60 myr in one hour and 1,200 myr (1.2 by) in one day. A N-body with 8192 stars takes about 12-16 minutes per timestep of 1 myr.

Conclusions

I was able to successfully create a galaxy simulation code using Python and C. My results for Andromeda are consistent with the experimental data and similar to professional

models which show that dark matter is needed to explain the observed flat rotational curves in galaxies. The simulations also showed that dark matter helps to stabilize the galaxy's rotation. Simulations can be run for longer periods to see how dark matter can effect the galaxy rotation over many billions of years. Using my simulation code, I can do these tests with other galaxies not just Andromeda. In conclusion this project can be used to help understand how dark matter affects galaxies and what it really is.

Future Work

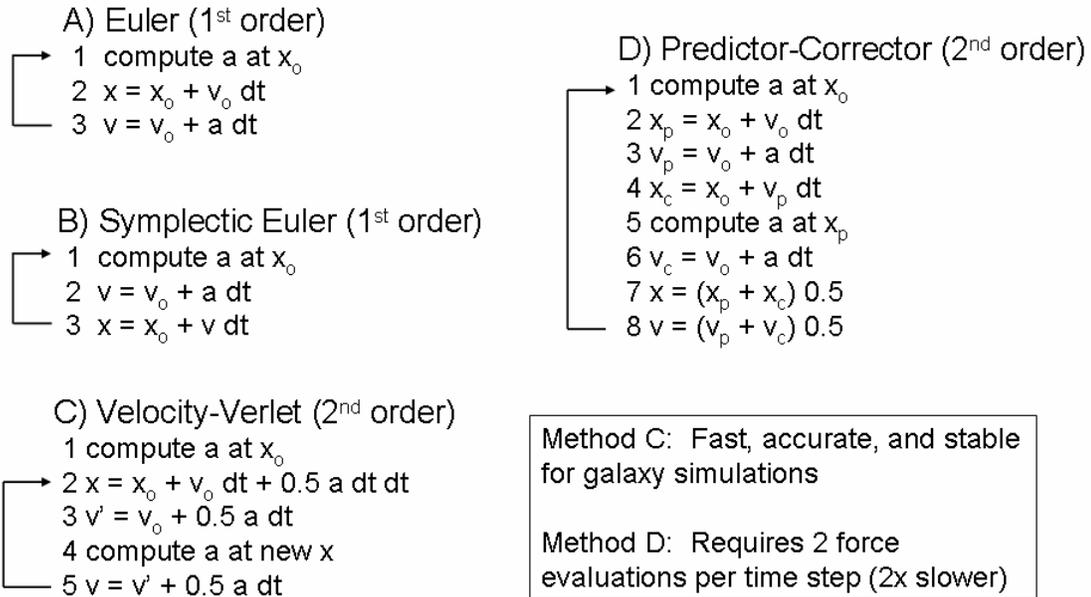
Future work includes using OpenCL to make the program run on the graphics card (GPUs – Graphical Processing Unit) and also using OpenCL to do multi-threading (running the program on multiple CPUs). Although some work has been done on OpenCL, work is still needed to be able to get results from the GPUs. Also future work includes adding more mass points and seeing how my results might change. Adding 3D to the program will help make my model more realistic and also more accurate. More tests can be run to find the optimal cutoff distances, time step values, and number of mass points in the galaxy. Also, seeing how different calculation methods such as the predictor corrector and Runge-Kutta affect the results (especially in the core region). Although using these methods could come at a disadvantage because the predictor corrector method requires an additional force calculation (a total of 2) at each time step and the Runge-Kutta method requires three additional force calculations (a total of 4).

References

- 1) Widrow M. Laurence, Perrett M. Kathryn, and Suyu H. Sherry. "Disk-Bulge-Halo Models for the Andromeda Galaxy." *The Astrophysical Journal* **588** (2003): 311-325. Print.
- 2) Carignan Claude, Chemin Laurent, Huchtmeier K. Walter, and Lockman J. Felix. "The Extended H I Rotation Curve and Mass Distribution of M31." *The Astrophysical Journal* **641** (2006): L109-L112. Print.
- 3) Bahcall N. John, and Soneira M. Raymond. "On the Interpretation of Rotation Curves Measured at Large Galactocentric Distances." *The Astrophysical Journal* **258** (1982): L23-L27. Print.
- 4) Sellwood A. J. "The global stability of our Galaxy." *Royal Astronomical Society* **217** (1985): 127-148. Print.
- 5) Ostriker P. J., and Peebles E. J. P. "A Numerical Study of the Stability of Flattened Galaxies: or, can Cold Galaxies Survive?" *The Astrophysical Journal* **186** (1973): 467-480. Print.
- 6) Hernquist Lars. "An Analytical Model for Spherical Galaxies and Bulges." *The Astrophysical Journal* **356** (1990): 359-364. Print.
- 7) Williams R. P and Nelson H. A. "Numerical simulation of the formation of a spiral galaxy." *Astronomy & Astrophysics* **374** (2001): 839-860. Print.
- 8) Navarro J. F., Frenk C.S, and White S.D. "A Universal Density Profile from Hierarchical Clustering." *The Astrophysical Journal* **490** (1997): 493-508. Print.
- 9) Begeman, K.G., "HI Rotation Curves of Spiral Galaxies. I. NGC 3198", *Astronomy and Astrophysics* **223** (1989), 47-60. Print.
- 10) Zwicky, F., "Die Rotverschiebung von extragalaktischen Nebeln", *Helvetica Physica Acta* **6** (1933): 110–127. Print.
- 11) Milgrom, M., "A modification of the Newtonian dynamics as a possible alternative to the hidden mass hypothesis". *Astrophysical Journal* **270** (1983): 365–370. Print.
- 12) Bennett, J.O., Donahue, M., Schneider, N., and Voit, M., "The Essential Cosmic Perspective", Pearson; Addison Wesley, (3rd edition) 2005. Print.

Appendix 1

Four different methods were tried for solving Newton's equation of motion (1D examples):



Four N-body simulations were run at different dt 's; 10, 20, 50, 100, 200, and 400 years. Each simulation ran to 10 million years using a different method to determine the optimal method for galaxy simulations. Method A is Euler's method, B is Symplectic Euler, C is Velocity Verlet, and D is Predictor Corrector. The velocity and position errors were computed for each simulation relative to the Predictor Corrector with $dt=10$ yr. Figure A1 shows the position error vs. dt (timestep). The second graph (Fig. A2) shows the velocity error for the different methods vs. dt (timestep). The velocity Verlet method C was essentially identical to the Predictor Corrector but is two times faster. So the velocity Verlet (method C) was chosen for all of my simulations.

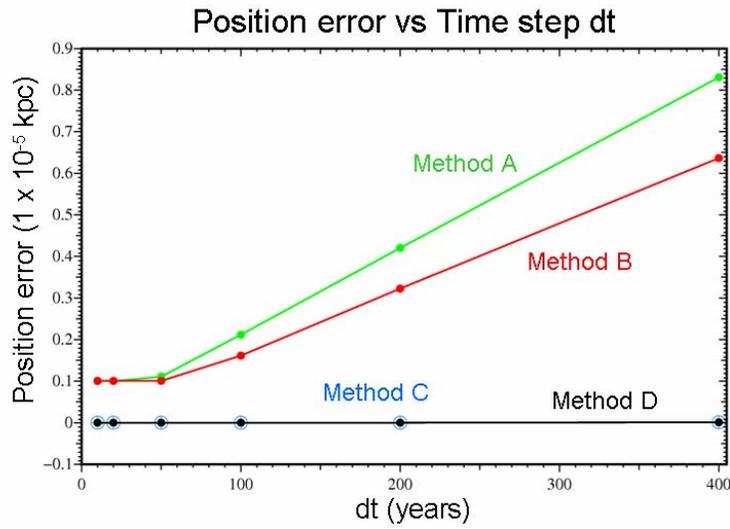


Figure A1: Plots the position error (for each method) as a function of the time step dt.

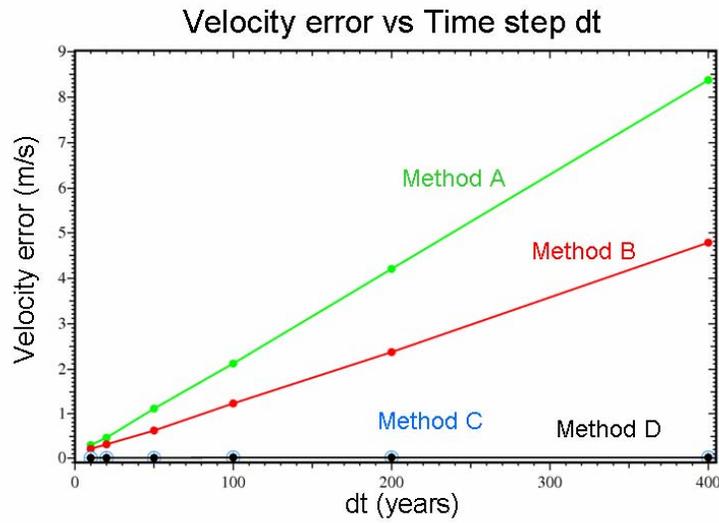


Figure A2: Plots the velocity error (for each method) as a function of the time step dt.

Appendix 2

```
#PYTHON Main Computational Loop
while t<tmax: #Run while time is less than the final time
    t=t+dt #Increment time in program
    for j in range(0,n): #Loops over stars with j counter
        rsq = x[j]**2 + y[j]**2
        r[j]=sqrt(rsq) #Distance to galaxy's center
        rkpc=r[j]/kpc #Converts from meters to kpc
#NFW CDM Mass (Basically Linear Mass, see CDM M(r) in Fig)
        mcdm[j]=mcdm0*(-rkpc/(A + rkpc)-log(A) +log(A+rkpc))
#Total mass =black hole + core + disk + CDM
        mass=m0 + mt[j] + mcdm[j]
#Velocity-Verlet (Method C):
#Compute positions & velocities at current Ax,Ay (steps 2,3)
        x[j] = x[j] + vx[j]*dt + 0.5*Ax*dt*dt #Position of jth
        y[j] = y[j] + vy[j]*dt + 0.5*Ay*dt*dt
        vxn[j] = vx[j] + 0.5*Ax*dt #Velocity of jth star in X
        vyn[j] = vy[j] + 0.5*Ay*dt #Velocity of jth star in Y
#Compute new values for Ax and Ay at new x and y (step 4)
        f = (g*m[j]*mass)/rsq #Force due to Gravity
        ex = x[j] / r[j] #Cosine
        ey = y[j] / r[j] #Sine
#Compute NEAREST NEIGHBOR force INTERACTIONS
        Fintx=0.0 #Initialize to zero
        Finty=0.0
        if (nn[j] != 0):#Check number of nearest neighbors
#Loops over nearest neighbor stars with i counter
            for i in range(0,nn[j]):
                ind=j*n + i #Index for ith neighbor of j
                iv=jn[ind] #Global index for ith neighbor
#Computes distance2 between stars; drmin keeps distance nonzero
                dr2=(x[j]-x[iv])**2+(y[j]-y[iv])**2+drmin
#Computes gravitational force between star j and neighbor iv
                fint=g*m[j]*m[iv]/dr2
                dr=sqrt(dr2) #The distance
                exi = (x[j]-x[iv]) / dr #Cosine
                eyi = (y[j]-y[iv]) / dr #Sine
#Add up interactions between j and iv in x and y directions
                Fintx=-fint*exi + Fintx
                Finty=-fint*eyi + Finty
#Compute Total Gravitational force including NN interactions
                Fx = -f * ex + Fintx #Total Force on jth star in X
                Fy = -f * ey + Finty #Total Force on jth star in Y
                Ax = Fx / m[j] #Acceleration in X (Newton's law)
                Ay = Fy / m[j] #Acceleration in Y
#Update velocity with new accelerations (step 5)
                vx[j] = vxn[j] + 0.5*Ax*dt #Velocity of jth star in X
                vy[j] = vyn[j] + 0.5*Ay*dt #Velocity of jth star in Y
```