**REPORT**

**ON**

**THE STATISCAL STUDY OF OORT CLOUD AND ITS EFFECTS ON THE INNER SOLAR SYSTEM.**

**BY**

ID NAME

2009A7PS126G ARUNAV SANYAL

2009B5A3468G ADITYA NAIK

**PREPARED IN PARTIAL FULFILLMENT OF**

**STUDY ORIENTED PROJECT**

****

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI(RAJASTHAN)**

**REPORT**

**ON**

**THE STATISCAL STUDY OF OORT CLOUD AND ITS EFFECTS ON THE INNER SOLAR SYSTEM.**

**BY**

ID NAME DISCIPLINE

2009A7PS126G ARUNAV SANYAL COMPUTER SCIENCE

2009B5A3468G ADITYA NAIK PHYSICS + ELECTRICAL &ELECTRONICS

**PREPARED IN PARTIAL FULFILLMENT OF**

**STUDY ORIENTED PROJECT**

****

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI(RAJASTHAN)**

**ACKNOWLEDGEMENT**

To say that this report is the result of the efforts taken by just the aforementioned two members would be a gross injustice. Injustice to fellow BITSians who helped us out through invaluable opinions and constant co-operation. Special thanks are reserved for Mr. Aditya Raj Pant for his help in the simulation programming. And of course, we thank with our collective hearts, Dr. Tarun K Jha, for giving us such an opportunity to realize our potential, along with the incentive of studying the field of astrophysics, in our own small and limited capacity.

**BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI(RAJASTHAN)**

**STUDENTS:**

ID NAME DISCIPLINE

2009A7PS126G ARUNAV SANYAL COMPUTER SCIENCES

2009B5A3468G ADITYA NAIK PHYSICS + ELECTRICAL &ELECTRONICS

FACULTY:

Dr. Tarun K Jha

PHYSICS FACULTY, BITS-PILANI KK BIRLA GOA CAMPUS.

ABSTRACT:

The giant cometary sphere surrounding the solar system is named the Oort cloud, after Jan Oort who postulated its existence in 1950. Comets arrive in the solar system from all directions, often from as far away as 100,000 AU. An AU, or Astronomical Unit, is the distance from earth to the sun. Pluto, for comparison, is 39 AU from the sun. This report tries to estimate the statistical data of comets, originating from the Oort cloud, hitting the Earth or reaching its orbit and the effects it would have on the inner solar system. It also summarizes the causes of dislocation of these comets. Finally the report summarizes the statistical analysis of the Oort cloud via a computer simulation to better understand the dynamics of the cloud under various disturbances.

**CONTENTS**

1. Introduction

2. History and Theoretical Background of Oort Cloud

2.1. History

2.2. Origin

## 2.3. Star Perturbations and Nemesis Theory

3. Composition and temperature estimation.

3.1. Oort cloud Objects

3.2. Comparison with solar system

4. Simulation and Relevant Calculations using Python

4.1. Assumptions in the Oort cloud Model

4.2. Initialization of Oort cloud

4.3. Acting Forces

4.4. Numerical Methods used for Calculations

4.5. Dynamics of the Model

5. Results and scope of the project

5.1. Correctness of model

5.2. Sample runs and extrapolated values

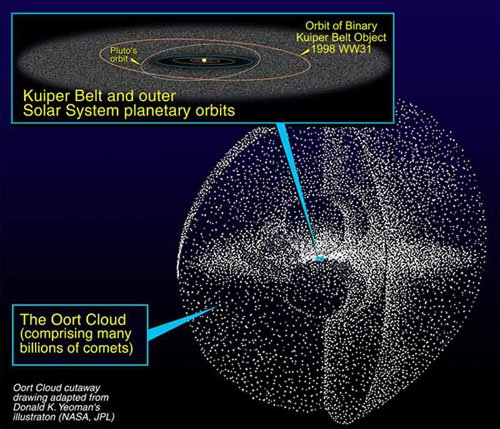
5.3. Scope for further improvement

5.4. Possible applications of the simulation

6. References

7. Simulation code

**1. INTRODUCTION**

****

The Oort cloud is a massive spherical cloud; the size of this cloud is disputed by different astronomers. Some believe.

e that it begins at 2000 or 5000 astronomical units–an astronomical unit (AU) equals the distance between the Earth and the Sun–and ends at 50,000 AU, which is almost a light-year. Others think that it may extend to over 100,000 AU, which would mean its edge would extend to nearly the end (gravitational extreme) of the Solar System.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |

The sphere was named after the astronomer Jan Oort who hypothesized its existence in 1950. Although its existence has not yet been proven through direct observation, the reality of the Oort cloud is widely accepted in the scientific community. The Oort cloud is filled with icy objects/comets composed of ammonia, water and methane.

Although a few comets in our solar system come from the Kuiper Belt, most are thought to originate from the Oort cloud. The comets enter the inner solar system when passing stars knock some of the objects in the Oort cloud out of their orbits. These long-period comets then travel into the inner solar system. Short-period comets have orbits that last up to two hundred years while the orbits of long-period comets can last for thousands of years. Most short-period comets come from the Kuiper Belt, while long-period comets come from the Oort cloud. There are a few exceptions to this rule though. Halley’s Comet is short-period comet that originated in the Oort cloud.

The contents of both Kuiper Belt and the Oort cloud are known as Trans-Neptunian Objects (TNOs) because the objects of both regions have orbits that that are further from the Sun than Neptune’s orbit is. Because the Oort cloud is so much farther out than the Kuiper Belt, it has not explored unlike the Kuiper Belt. Additionally, astronomers have been unable to identify objects to the degree that they have in the Kuiper Belt. In fact, aside from long-period comets, astronomers have only found four objects that, from the objects’ orbits, they believe came from the Oort cloud.

**2. HISTORY AND THEORETICAL BACKGROUND OF OORT CLOUD**

**2.1 History**

In 1932, Ernst Öpik, an [Estonian](http://www.newworldencyclopedia.org/entry/Estonia) astronomer, proposed that comets originate in an orbiting cloud situated at the outermost edge of the solar system. In 1950 the idea was revived and proposed by [Dutch](http://www.newworldencyclopedia.org/entry/Netherlands) astronomer Jan Hendrick Oort to explain an apparent contradiction: comets are destroyed by several passes through the inner [solar system](http://www.newworldencyclopedia.org/entry/Solar_system), yet if the comets we observe had really existed for billions of years (since the generally accepted origin of the solar system is of a comparable time scale), all would have been destroyed by now. According to the hypothesis, the Oort cloud contains millions of comet nuclei, which are stable because the sun's radiation is very weak at their distance. The cloud provides a continual supply of new comets, replacing those that are destroyed. It is believed that if the Oort cloud exists and supplies comets, in order for it to supply the necessary volume of comets, the total mass of comets in the Oort cloud must be many times that of [Earth](http://www.newworldencyclopedia.org/entry/Earth). Estimates range between five and 100 Earth masses although the number of comets in this cloud has been and still is a subject of speculation.

There is also a theory of a denser, inner part of the Oort cloud coined the Hills cloud; it would have a well-defined outer boundary at 20,000 to 30,000 AU, a less well defined inner boundary at 50 to 3,000 AU, and would be about 10 to 100 times denser than the remainder.

The Oort cloud is thought to be a remnant of the original solar nebula that collapsed to form the [Sun](http://www.newworldencyclopedia.org/entry/Sun) and [planets](http://www.newworldencyclopedia.org/entry/Planet) approximately 4.6 billion years ago, and is loosely bound to the [solar system](http://www.newworldencyclopedia.org/entry/Solar_system).

**2.2. Origin**

The most widely-accepted hypothesis of its formation is that objects in the Oort cloud initially formed much closer to the Sun as part of the same process that formed the [planets](http://www.newworldencyclopedia.org/entry/Planet) and [asteroids](http://www.newworldencyclopedia.org/entry/Asteroid), but that gravitational interaction with young gas giants such as Jupiter ejected them into extremely long elliptical or parabolic orbits. This process also served to scatter the objects out of the ecliptic plane, explaining the cloud's spherical distribution. While on the distant outer regions of these orbits, gravitational interaction with nearby stars further modified their orbits to make them more circular.

It is thought that other [stars](http://www.newworldencyclopedia.org/entry/Star) are likely to possess Oort clouds of their own, and that the outer edges of two nearby stars' Oort clouds may sometimes overlap, causing perturbations in the comets' orbits and thereby increasing the number of comets that enter the inner solar system. Although an exotic idea, this is highly unlikely for our solar system, as the nearest star is proxima centauri approximately 4.2 light years away and with a much weaker gravitational field than the sun making it incredibly unlikely to perturb the oort cloud in this manner.

**2.3 Start Perturbations and Nemesis Theory**

The known star with the greatest possibility of perturbing the Oort cloud in the next 10 million years is Gliese 710. However, physicist Richard A. Muller and others have postulated that the [Sun](http://www.newworldencyclopedia.org/entry/Sun) has a heretofore undetected companion star in an elliptical orbit beyond the Oort cloud based on the records of heavy bombardments on the [Earth](http://www.newworldencyclopedia.org/entry/Earth) that caused [mass extinctions](http://www.newworldencyclopedia.org/entry/Mass_extinction)( for example the jurrasic period bombardment that caused the extinction of the dinosaurs).[[6]](http://www.newworldencyclopedia.org/entry/Oort_cloud#cite_note-5) This star(brown dwarf to be precise), known as Nemesis, is theorized to pass through a portion of the Oort cloud approximately every 26 million years, bombarding the inner [solar system](http://www.newworldencyclopedia.org/entry/Solar_system) with comets. Although the theory has many proponents, no direct proof of the existence of Nemesis has been found.

**2.4 Inner solar system dynamics and the oort cloud**

The inner solar system has been speculated to be effected in the past by the Oort cloud. Periodic mass extinction events in Earth are often speculated to be caused by massive perturbations in the Oort cloud. Often these perturbations are periodic (eg. the nemesis hypothesis) and often random.

Even for small period comets a particular effect is prominent which significantly reduces the danger Earth is put into. This is called the screening effect. It is actually the phenomenon in which the largest gas giants tend to gravitationally capture comets within the inner solar system that get too close to them. This leads to lesser number of comets reaching Earth. A particularly notable example of this is the shoemaker levy comet that struck into Jupiter in 1996.

**3. COMPOSITION AND TEMPERATURE ESTIMATION.**

**3.1.Oort Cloud Objects**

So far, only three potential Oort cloud objects have been discovered: 90377 Sedna, [2000 OO67](http://www.newworldencyclopedia.org/p/index.php?title=%2887269%29_2000_OO67&action=edit&redlink=1), and [2000 CR105](http://www.newworldencyclopedia.org/p/index.php?title=2000_CR105&action=edit&redlink=1).

90377 Sedna's orbit that ranges from roughly 76 to 925 AU, does not carry it completely out to the assumed position of the Oort cloud and is too far out for it to be truly considered as a Kuiper belt object. If Sedna indeed belongs to the Oort cloud, this may mean that the Oort cloud is both denser and closer to the Sun than previously thought.

Some astronomers include the objects [2000 CR105](http://www.newworldencyclopedia.org/p/index.php?title=2000_CR105&action=edit&redlink=1) and [2000 OO67](http://www.newworldencyclopedia.org/p/index.php?title=%2887269%29_2000_OO67&action=edit&redlink=1) as part of the Oort cloud. The object [2000 CR105](http://www.newworldencyclopedia.org/p/index.php?title=2000_CR105&action=edit&redlink=1) has a perihelion of 45 AU, an aphelion of 415 AU and an orbital period of 3,241 years while the object [2000 OO67](http://www.newworldencyclopedia.org/p/index.php?title=%2887269%29_2000_OO67&action=edit&redlink=1) has a perihelion of 21 AU, an aphelion of 1,000 AU and an orbital period of 12,705 years.

The outer Oort cloud is believed to contain several trillion individual objects larger than approximately 1 km (0.62 mi) (with many billions with [absolute magnitudes](http://en.wikipedia.org/wiki/Absolute_magnitude#Solar_System_bodies_.28H.29) brighter than 11—corresponding to approximately 20 km (12 mi) diameter), with neighboring objects typically tens of millions of kilometers apart. Its total mass is not known with certainty, but, assuming that Halley's comet is a suitable prototype for all comets within the outer Oort cloud, the estimated combined mass is 3×1025 kg (7×1025 lb or roughly five times the mass of the Earth). Earlier it was thought to be more massive (up to 380 Earth masses), but improved knowledge of the size distribution of long-period comets has led to much lower estimates. The mass of the inner Oort cloud is not currently known.

If analyses of comets are representative of the whole, the vast majority of Oort-cloud objects consist of various ices such as water, [methane](http://en.wikipedia.org/wiki/Methane), [ethane](http://en.wikipedia.org/wiki/Ethane), [carbon monoxide](http://en.wikipedia.org/wiki/Carbon_monoxide) and [hydrogen cyanide](http://en.wikipedia.org/wiki/Hydrogen_cyanide). However, the discovery of the object 1996 PW, an asteroid in an orbit more typical of a long-period comet, suggests that the cloud may also be home to rocky objects. Analysis of the [carbon](http://en.wikipedia.org/wiki/Carbon) and [nitrogen](http://en.wikipedia.org/wiki/Nitrogen) [isotope](http://en.wikipedia.org/wiki/Isotope) ratios in both the Oort cloud and Jupiter-family comets shows little difference between the two, despite their vastly separate regions of origin. This suggests that both originated from the original protosolar cloud, a conclusion also supported by studies of granular size in Oort cloud comets and by the recent impact study of Jupiter-family comet.

**3.2. Comparison with Solar system**

The following table and the graph give detailed information about the temperatures of the planets in solar system, their distance from the sun, orbital eccentricity and their states of matter. This data gives us a rough estimation of the temperature and the shape of the Oort cloud if we extend and extrapolate the graphical lines.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **PLanet** | **Distance from sun (Million Kms)** | **Max temp(**°C) | **Min temp(°C)** | **Orbital eccentricity** | **States of matter** |
| Mercury | 57 | 465 | -184 | 0.206 | Rocky,mostly iron |
| Venus | 107 | 480 | -184 | 0.0068 | Rocky |
| Earth | 150 | 60 | -90 | 0.0167 | Rocky |
| Mars | 229 | 20 | -140 | 0.0934 | Rocky |
| Jupiter | 777 | 21 | -148 | 0.0485 | Gas giant |
| Saturn | 1429 | -130 | -178 | 0.0556 | Gas giant |
| Uranus | 2871 | -197 | -224 | 0.0472 | Ice giant |
| Neptune | 4496 | -214 | -229 | 0.0086 | Ice giant |
| Pluto | 5869 | -223 | -234 | 0.25 | Rocky |

At Extremely large distances temperatures tend to reach absolute zero. However due to remaining cosmic microwave background radiation temperatures tend to even out at 2.73 K and stay constant, unless heated up by some other sources. Almost all elements solidify at these temperatures and thus the possibility of condensation among these objects increase.

**4. Simulation and Relevant Calculations using Python.**

We have made an attempt to simulate a simplified solar system along with the Oort cloud and a significantly large perturbing object dubbed the ‘floating mass’. We have simulated the cloud under gravitational interactions over extended periods of the time. We tried to simulate the effect of screening of comets by inner solar system planets and get an estimate over the number of comets reaching up to the Earth’s orbit. The main goal of this simulation is to estimate the actual size and composition of the Oort cloud under a hypothesized perturbation and known observational data. We have based our simulation under a number of idealistic assumptions and constraints. However what is really nice about this model is that the mass of the comet gets cancelled on both sides of the equations and thus is not taken as an input or randomized parameter.

**4.1.Assumptions taken in oort cloud model:-**

1. All comets follow perfectly spherical orbits initially. This is true especially of the outer Oort cloud objects.

2. The initial velocities of the comets are orthogonal to the radius vector.

3. As of now we have included forces of Sun, Jupiter, Earth and the floating masses disturbing the cloud.

4. The floating mass velocity components are being randomly generated subject to certain constraints

5. Earth and Jupiter both follow perfectly circular orbits.

6. We can at will change the output/calculation duration and the amount of time the simulation runs. However our simulation remains accurate for small durations. We shall discuss the reasons later.

7. The numerical integrator we will use is Runge-Kutta and it loses its accuracy as we increase the calculation duration time or the force acting at that point is large. However it will reduce the computational load on the computer. We shall talk about the dynamics later.

8. We have neglected other planets but they can be included by similar logic that is used to simulate Jupiter's effects. ( Only the number of variables will increase, no more fundamental logic is introduced )

9. All particle masses are considered point masses.

10. We are considering the Oort cloud to be sparsely populated. We are neglecting collisions between the comets and any other bodies.

11. The planets and floating masses are considerably heavier than the comets. Hence we are neglecting changes of their paths due to forces. (This is grossly inaccurate if the floating mass is considerably lighter than the sun or for longer simulation times). However this model is accurate for the workaround method we are using

12. The entire simulation follows Newtonian Mechanics. No relativistic equations are used.

**4.2.Initialisation of the Oort cloud:-**

Oort cloud:-

To generate the Oort cloud we have taken the number of comets and two radii values r(inner) and router from the user and initialized radius vectors randomly in this sphere using r(inner)<r<r(outer) and x and z azimuths(angles) randomly assigned. To initialize velocity we have used Newton's law to derive speed of comet at a given distance.  
  
 0.5mv ^ 2 = GmM/r

Then we have used the fact that x1v1 + x2v2 + x3v3 =0(the velocity of the comet is orthogonal to its radius vector) where 1, 2, 3, indices stand for x, y, z axes respectively. We have randomly assigned two of the three velocities between 0 and 1 and calculated the third from the equation given above. Then we divided each of the velocities by the root mean square sum of the three components and multiplied the maximum speed generated previously to arrive at the required velocity components. Then we use standard polar to Cartesian co-ordinate changes to arrive at velocity components in Cartesian form.

**Floating masses:-**

We have added only single floating mass for interaction. The initial mass value has to be comparable to the Sun’s mass, or else either the perturbations are too small thereby not affecting the cloud, or it is too large and will just blow the cloud away. Possible candidates for this are dark matter halos, the brown dwarf of the nemesis hypothesis (although in this case we have to include the curvature of the trajectory of this brown dwarf) etc.

The floating mass follows a straight line trajectory and is unaffected by the Sun. This is grossly inaccurate for longer times, but due to the divergent behavior of the model(discussed later) it is a fair approximation.

Adding extra masses is just extending the code and no new fundamental logic is used for this purpose. Also the direction is randomized with a component of velocity along the negative radial direction from the sun. In other words, it tends to approach the sun. This is achieved by adding constraints to velocity directions depending on azimuthal angles.

Planets:-

We have considered planets Earth and Jupiter for screening effect (i.e. gravitational capture). Their orbits are initialized at zero azimuthal angles and follow uniform circular motion with their actual radial velocities.

**4.3. Acting forces:-**

Floating mass: - This mass is the primary source of perturbation and is given as in input. The forces exerted are purely gravitational in nature and collisions with comets and other bodies are neglected.

Sun: - The sun exerts gravitational force on the Oort cloud and keeps it in stable orbit.

Inner planets: - In this case we are considering Jupiter and Earth but we can add other planets or bodies for a more realistic model. The planets follow their usual orbits and hence the radius vector joining a comet and the planet dynamically changes. The forces are purely gravitational in nature.

Galactic bulge and galactic disk: - We are not applying this force to the model but for more realistic simulations it is preferred.

**4.4. Numerical methods used:-**

Uniform randomization:-Almost in every initialization we have applied uniform randomization by the random module in python.

Runge Kutta integrator:-. It is the principle technique by which we will advance the velocities and position vectors of the comets. It is actually a numerical method to recalculate velocities and thus positions of the comets.

vx(tn+1) = vx(tn) + 1/6(k1 + 2 k2 + 2 k3 + k4)

k1 = hFx(tn, r(tn))

k2 = hFx(tn + h/2 , r(tn) + (k1)/2 )

k3 = hFx(tn + h/2 , r(tn) + (k2)/2 )

k4 = hFx(tn + h, r(tn) + k3)

Here Fx(t,r(t)) is the sum total of all forces along the x direction at time t and radius vector r, h is the interval of time chosen over which to recalculate velocities.

Finally x(t+h) = x(t) + v(tn +1) \* h will give us the new position of the comet. We can similarly calculate the new y and z coordinates by replacing Fx by Fy and Fz respectively.

**4.5.Dynamics of the model:-**

**Run time:-**

We can at will change the simulation time and interval of time after which the forces are recalculated.

The simulation is actually modeled on a simplified solar system with the forces recalculated at discrete intervals of time. The main aim of the simulation is to track the Oort cloud’s behavior at significant perturbations and estimate the number of comets reaching the earth. There are two significant flaws with this simulation and both are computational limitations not physical errors.

One of them is the fact is that any realistic long term simulations will take a stupendous amount of time on any computer. Reason- For say 50000 comets we have to recalculate 4 forces, update them with each updating itself taking 2 recalculations and we have to simulate this over a large time( consider 1 second interval for 1000000 seconds) . This particular simulation itself took about 3 hours and it is actually a miniscule amount of time is astronomical terms from which no conclusions can be made. One could say increase the interval of time over which we recalculate forces, but that leads to the second problem, the divergence of solutions.

**Divergent behavior:-**

The simulation diverges for the following two cases:-

1. The intervals of time for each iteration (force recalculation time interval) taken is long

2. The force recalculated is too large

Reason- The model is that of a continuous system being modeled using discrete math. In reality the forces in the natural solar system continually change in instants of time, in stark contrast to the model created here. The amount of computation required to perfectly describe such systems is tremendously vast and beyond the scope of computation today. Regardless, at long intervals of time over which forces are recalculated, the velocities of the comets stay essentially constant. This results in significant errors as the time duration is increased. Even for smaller time durations and large forces, velocity changes are often not implemented accurately and could lead to significant errors. This is the reason we could not see an actual screened comet (the comet behaves incorrectly when it comes too close to a planet). These two flaws coupled together have not allowed the authors to carry out any realistic simulations with precision. Four workarounds had been thought of:-

1. Using Runge Kutta 4th order method- This gives a further accuracy over two more decimal points but does not give a significant advantage over the divergent behavior (only the model can tolerate higher time values and forces but still will diverge after a threshold).
2. Extrapolation- This is the method we have actually employed. We have carried out the simulation for a small amount of time, binned it (got the radial distances) at specific intervals of time and extrapolated values have been obtained for large value. While grossly inaccurate, especially in case of the screening effect this is the only feasible solution we could possibly employ.
3. Scale down the solar system:- Proportionately decrease all distance by a factor of k and increase all masses by square root of k. The behavior will be exactly the same, only scaled down and faster. However the downside of this method was that the divergent behavior will be even more prominent and lead to even more errors.
4. Employ more computers: - Clustering the program across different machines or multithreading the program seemed like a good choice. The authors could only manage six computers at a time and the speed up that could have been achieved( assuming perfect parallelization and no communication delay) would still only give a 6 times boost. To achieve a perfect simulation, we would require considerably much more computation than that. The amount of extra coding required does not justify the effect on the end result (see section 5). Thus this idea was not implemented.

**5.Results and scope of the project: -**

We have not been able to fully simulate the solar system along with the Oort cloud for even modest intervals of time due to the computational constraints

**5.1. Correctness of the model**

By the following test runs we were able to show that the model is correct:-

1. We did not include any perturbations and simulated comets at various distances (1 AU to 100000AU). In all cases the orbits are stable irrespective of the number of comets. (This is true only if we don’t give parameters that would lead to divergent behavior, e.g.- comets at 0.001 AU or recalculations times as long as a year)
2. Turning the mass of the Sun as zero and introducing a floating mass with zero velocity and mass = the mass of the sun. The behavior of the comets was exactly the same case 1.
3. Changing case 2’s floating mass’s mass value = 100 times the mass of the sun. For outer Oort cloud objects, they begin to fall inwards (at distances of 1 AU however, the comets diverge).
4. Turning both the sun and floating mass’s mass as zero, but keeping Jupiter’s orbit the same. Comets tend to go outward if kept at 1AU distance

From these arguments we conclude that the model behaves accurately for all forces taken individually. As the sum of forces along each component is just a scalar addition of these forces, that too should be accurate.

**5.2. Sample runs and extrapolated values:-**

Observations: - We ran a total of six simulations on a varied mix on the number of comets. None of the simulations resulted in any comet actually reaching the Earth’s orbit (simply because the run time of the runs was not long enough for massive changes in trajectories). We have however interpolated values over extended time periods (it is inaccurate but is the best we could do).

The simulations were run on a dual core Intel Pentium 2.4 GHz computer. The best run (the correctness run was for 45 minutes) and the worst run (the run for 1000 comets) ran for about 8 hours and 32 minutes.

Instead of showing the output as directly generated by the program, we shall show the distribution of the comets at different time intervals by 3d models and graphs generated from the data of the program and Microsoft Excel.

**Correctness run:-**

This run was for 35 comets (out of which only 10 had all three coordinates as positive) and was basically a test for the correctness ran for 1000000 time units with 10 units of interval per iteration.

The following figures (for the correctness run, we took only the comets in the first quadrant, or else the figure does not remain understandable) were obtained:-

Fig. Initial and final distribution of comets that were in the first octant

We included both final and initial coordinates of both comets to show change in orbit (this simulation ran for 45 minutes)

Considering the amount of clutter that we encountered in viewing just 10 comets, we have taken a different approach towards real systems (comets). We have sampled 100 comets and represented their max distances from the sun as a function of comet number after sorting them in ascending order then represented(the comet numbers being the same) them. This particular simulation ran for 8 hours and 32 minutes.

**Specifics of the run:-**

Floating mass’s mass= 1 Solar Mass

Speed = 100000 m/s

Component speed along the Sun (derived from azimuthal angles of velocity) = 86778 m/s

Run time= 10^10 time units over 1 second intervals (had to resample the output 20 times due to memory error of list data structure in python).

X-axis: comet number (marker) Y-Axis: meters

Oort cloud radial distance bounds:-

Inner=10000AU

Outer= 50000AU

Initial distance of floating mass from Sun = 60000AU

Initial graph:-

Final graph after distortions:-

We used linear interpolation techniques (based on ten different sample times that we output for comparison). We interpolated the values to 300 million years and the following graph was obtained: -

10 out of these 100 comets sampled reached or crossed the Earth’s orbit.

**Comparison with real data:-**

Till date about 4850 known comets have long elliptic orbits and have reached Earth’s orbit. Using some extrapolation method we can calculate the number of comets in the Oort cloud using this data and the simulation results.

**5.3. Scope for further improvement**

Although the model is physically sound, it is actually a bit inefficient and a lot of factors have been neglected to simplify calculations. Some of these are:-

1. The model does not take into account planets (except Earth and Jupiter) and other massive bodies of the solar system. It is actually a trivial matter, as including other planets is just an extension of the existing code for the forces of Earth and Jupiter.
2. It does not take into account complicated forces like that of the Galactic Bulge. Again the reason for this was to simplify the model. For more realistic models this could however be needed.
3. At a time only one floating mass was considered and that too was gravitationally unaffected by the Sun. While this is acceptable for the small duration of the code, the logic fails for actual realistic extensions of the model.
4. The simulations were taking a stupendously large amount of time. The authors had to leave their computers running for hours at a time, with them often closing down due to them overheating. Probably running this model on computers or clusters faster than dual core laptops would give better results.

**5.4 Possible applications of the simulation**

While our original intention was to get an accurate estimate of the real Oort cloud’s parameters by comparing the multiple simulations’ results with observational data, we have been able to achieve this goal to a very limited extent.

The following could be the applications of the simulation:-

1. One could study the dynamics of the Solar system by appropriately modifying the code as an educational resource.
2. Extended codes could be used to generate actual realistic simulations in powerful computing environments.
3. Could be used to study both the asteroid and Kuiper belts by varying the input parameters and renaming the Oort cloud.

**6.References**

1) [What Are Asteroids And Comets?"](http://neo.jpl.nasa.gov/faq/#ast). Near Earth Object Program FAQ. [NASA](http://en.wikipedia.org/wiki/NASA).

2) Single Space thesis. University of California. Daniel Tierney.

3) Chebotarev, G.A. (1964). "Gravitational Spheres of the Major Planets, Moon and Sun". Astronomy

4) Jan Oort (1950). "The structure of the cloud of comets surrounding the Solar System and a hypothesis concerning its origin". [Bulletin of the Astronomical Institutes of the Netherlands](http://en.wikipedia.org/wiki/Bulletin_of_the_Astronomical_Institutes_of_the_Netherlands).

5) Julio A. Fernéndez (1997). ["The Formation of the Oort Cloud and the Primitive Galactic Environment"](http://www.gps.caltech.edu/classes/ge133/reading/oort.pdf)

6) Paul R. Weissman (1998). ["The Oort Cloud"](http://www.sciamdigital.com/index.cfm?fa=Products.ViewIssuePreview&ISSUEID_CHAR=8DB2FB44-6B4B-47AF-B46B-791A911764D&ARTICLEID_CHAR=B294C211-98B8-4374-92AB-158C4866AB1). [Scientific American](http://en.wikipedia.org/wiki/Scientific_American). Retrieved 2007-05-26.

7) Sebastian Buhai. ["On the Origin of the Long Period Comets: Competing theories"](http://66.102.1.104/scholar?hl=en&lr=&safe=off&q=cache:oco9aJ2k6UoJ:www.tinbergen.nl/%7Ebuhai/pictures/UCU/Physics_AppliedMathematics/Astrophysics/long_period_comets.pdf+380+mass+earth+oort-cloud). Utrecht University College.

8) Yeoman’s, Don. ["Near Earth Object Search Programs"](http://neo.jpl.nasa.gov/programs/). NASA. Retrieved 2008-04-15.

**7. Simulation code:-**

The following code has been written in python (without SymPy or any other complex module) using only basic math and randomization modules and has been tested both on idle (Windows) and fedora (Linux) platforms. Python was chosen because we wanted a simple object oriented language with good documentation for debugging.

The code starts from the next line:-

#we are assuming a uniform spherical shell shaped Oort cloud to get idealized computations so theta and phi (the azimuths)

#these masses are treated as point masses

#we have considered one floating mass per simulation, since doing a multiple comets and floating mass scenario will lead to O(n^3) complexity, and at high values is infeasible

Import random

Import math

EarthMass = 5.9723e24 #units: kg

JupiterMass = 1.8987e27 #units: kg

SunMass = 1.98844e30 #units: kg

AU = 149597870660.0 #AU = Astronomical Unit

G = 6.6742e-11 #grav const in AU^3 / (kg s^2)

Pi = 3.141592654

year = 60 \* 60 \* 24 \* 365 #one year in seconds

#only have to cross check units once­-

class comet():

def \_\_init\_\_(self,rin,rout):

self.rad\_dist=random.uniform(rin,rout) #polar coordinate representations of the comet's position

self.x\_azimuth=random.uniform(0,2\*Pi) #angles in radians

self.z\_azimuth=random.uniform(0,Pi)

self.x=self.rad\_dist\*math.sin(self.z\_azimuth)\*math.cos(self.x\_azimuth)

self.y=self.rad\_dist\*math.sin(self.z\_azimuth)\*math.sin(self.x\_azimuth)

self.z=self.rad\_dist\*math.cos(self.z\_azimuth)

self.max\_velocity=math.sqrt((G\*SunMass)/(self.rad\_dist\*AU)) #velocity obtained in meters per second

self.slack=random.randint(1,3)

if self.slack==1:

self.y\_v=random.uniform(0,1)

self.z\_v=random.uniform(0,1)

self.x\_v= -(self.y\*self.y\_v+self.z\*self.z\_v)/self.x

self.v=math.sqrt(self.x\_v\*self.x\_v+self.y\_v\*self.y\_v+self.z\_v\*self.z\_v)

self.x\_v=self.x\_v/self.v

self.y\_v=self.y\_v/self.v

self.z\_v=self.z\_v/self.v

self.x\_v=self.x\_v\*self.max\_velocity

self.y\_v=self.y\_v\*self.max\_velocity

self.z\_v=self.z\_v\*self.max\_velocity

elif self.slack==2:

self.x\_v=random.uniform(0,1)

self.z\_v=random.uniform(0,1)

self.y\_v= -(self.x\*self.x\_v+self.z\*self.z\_v)/self.y

self.v=math.sqrt(self.x\_v\*self.x\_v+self.y\_v\*self.y\_v+self.z\_v\*self.z\_v)

self.x\_v=self.x\_v/self.v

self.y\_v=self.y\_v/self.v

self.z\_v=self.z\_v/self.v

self.x\_v=self.x\_v\*self.max\_velocity

self.y\_v=self.y\_v\*self.max\_velocity

self.z\_v=self.z\_v\*self.max\_velocity

else:

self.x\_v=random.uniform(0,1)

self.y\_v=random.uniform(0,1)

self.z\_v = - (self.x\*self.x\_v+self.y\*self.y\_v)/self.z

self.v = math.sqrt(self.x\_v\*self.x\_v+self.y\_v\*self.y\_v+self.z\_v\*self.z\_v)

self.x\_v=self.x\_v/self.v

self.y\_v=self.y\_v/self.v

self.z\_v=self.z\_v/self.v

self.x\_v = self.x\_v\*self.max\_velocity

self.y\_v = self.y\_v\*self.max\_velocity

self.z\_v = self.z\_v\*self.max\_velocity

#self.x\_v=0

#self.y\_v=0 #debugging lines

#self.z\_v=0

self.x=self.x\*AU #converting to meters

self.y=self.y\*AU

self.z=self.z\*AU

self.Earth\_rad\_dist=1\*AU #initialisation of the Earth and Jupiter screeners, here in meters

self.Jupiter\_rad\_dist=5.46\*AU

self.Earth\_x\_azimth=random.uniform(0,2\*Pi)

self.Earth\_z\_azimtuh=Pi/2

self.Jupiter\_x\_azimth=random.uniform(0,2\*Pi)

self.Jupiter\_z\_azimtuh=Pi/2

def orbits(self,time):

self.omega\_Earth= math.sqrt(G\*EarthMass/pow(self.Earth\_rad\_dist,3)) #angular velocity

self.Earth\_x\_azimuth= self.omega\_Earth\*time #time over which it interacts

self.Earth\_z\_azimuth=Pi/2

self.omega\_Jupiter= math.sqrt(G\*JupiterMass/pow(self.Earth\_rad\_dist,3)) #angular velocity

self.Jupiter\_x\_azimuth= self.omega\_Jupiter\*time #time over which it interacts

self.Jupiter\_z\_azimuth=Pi/2

self.Jupiter\_x=self.Jupiter\_rad\_dist\*math.sin(self.Jupiter\_z\_azimuth)\*math.cos(self.Jupiter\_x\_azimuth)

self.Jupiter\_y=self.Jupiter\_rad\_dist\*math.sin(self.Jupiter\_z\_azimuth)\*math.sin(self.Jupiter\_x\_azimuth)

self.Jupiter\_z=self.Jupiter\_rad\_dist\*math.cos(self.Jupiter\_z\_azimuth)

self.Earth\_x=self.Earth\_rad\_dist\*math.sin(self.Earth\_z\_azimuth)\*math.cos(self.Earth\_x\_azimuth)

self.Earth\_y=self.Earth\_rad\_dist\*math.sin(self.Earth\_z\_azimuth)\*math.sin(self.Earth\_x\_azimuth) #i need to call this function every iteration of the main loop of the program

self.Earth\_z=self.Earth\_rad\_dist\*math.cos(self.Earth\_z\_azimuth)

def forces(self,time,mass,x,y,z): #i have to call this function after every iteration of the main loop

self.orbits(time)

self.SunForce=(G\*SunMass)/(pow((self.rad\_dist\*AU),2)) #force of the sun

self.SunForce\_x=-self.SunForce\*math.sin(self.z\_azimuth)\*math.cos(self.x\_azimuth) #exaclty in reverse of r direction

self.SunForce\_y=-self.SunForce\*math.sin(self.z\_azimuth)\*math.sin(self.x\_azimuth) #these calculations are force per unit mass

self.SunForce\_z=-self.SunForce\*math.cos(self.z\_azimuth)

#EarthForce

self.EarthForce\_x=self.Earth\_x-x #as of now only force direction calclulations

self.EarthForce\_y=self.Earth\_y-y

self.EarthForce\_z=self.Earth\_z-z

self.rms\_Earth=math.sqrt(self.EarthForce\_x\*self.EarthForce\_x + self.EarthForce\_y\*self.EarthForce\_y + self.EarthForce\_z\*self.EarthForce\_z)

self.EarthForce=(G\*EarthMass)/(pow((self.rms\_Earth),2))

self.EarthForce\_x=-self.EarthForce\*math.sin(self.Earth\_z\_azimuth)\*math.cos(self.Earth\_x\_azimuth) #exaclty in reverse of r direction

self.EarthForce\_y=-self.EarthForce\*math.sin(self.Earth\_z\_azimuth)\*math.sin(self.Earth\_x\_azimuth) #these calculations are force per unit mass

self.EarthForce\_z=-self.EarthForce\*math.cos(self.Earth\_z\_azimuth)

#JupiterForce

self.JupiterForce\_x=self.Jupiter\_x-x #as of now only force direction calclulations

self.JupiterForce\_y=self.Jupiter\_y-y

self.JupiterForce\_z=self.Jupiter\_z-z

self.rms\_Jupiter=math.sqrt(self.JupiterForce\_x\*self.JupiterForce\_x + self.JupiterForce\_y\*self.JupiterForce\_y + self.JupiterForce\_z\*self.JupiterForce\_z)

self.JupiterForce=(G\*JupiterMass)/(pow((self.rms\_Jupiter),2))

self.JupiterForce\_x=-self.JupiterForce\*math.sin(self.Jupiter\_z\_azimuth)\*math.cos(self.Jupiter\_x\_azimuth) #exaclty in reverse of r direction

self.JupiterForce\_y=-self.JupiterForce\*math.sin(self.Jupiter\_z\_azimuth)\*math.sin(self.Jupiter\_x\_azimuth) #these calculations are force per unit mass

self.JupiterForce\_z=-self.JupiterForce\*math.cos(self.Jupiter\_z\_azimuth)

#force of floating mass

self.force\_x=mass.x\_coord-x #need to check for directions once

self.force\_y=mass.y\_coord-y

self.force\_z=mass.z\_coord-z

self.rms=math.sqrt(self.force\_x\*self.force\_x + self.force\_y\*self.force\_y + self.force\_z\*self.force\_z)

self.force=(G\*mass.mass\*SunMass)/(pow((self.rms),2))

self.force\_x=self.force\*math.sin(mass.z\_azimuth)\*math.cos(mass.x\_azimuth) #exaclty in reverse of r direction

self.force\_y=self.force\*math.sin(mass.z\_azimuth)\*math.sin(mass.x\_azimuth) #these calculations are force per unit mass

self.force\_z=self.force\*math.cos(mass.z\_azimuth)

#sum of forces, for this particular scenario, it can handle only one floatingmass at a time

self.sumofforces\_x=-self.force\_x+self.SunForce\_x+self.EarthForce\_x+self.JupiterForce\_x

self.sumofforces\_y=-self.force\_y+self.SunForce\_y+self.EarthForce\_y+self.JupiterForce\_y

self.sumofforces\_z=-self.force\_z+self.SunForce\_z+self.EarthForce\_z+self.JupiterForce\_z

#other forces can be similarly implemented but we defer it for now

def evaluate(self,deltatime,time,x,y,z,mass): #runge kutta evaluator

self.forces(time,mass,x,y,z)

self.k1x=deltatime\*self.sumofforces\_x

self.k1y=deltatime\*self.sumofforces\_y

self.k1z=deltatime\*self.sumofforces\_z

self.forces(time+deltatime/2,mass,x+self.k1x/2,y+self.k1y/2,z+self.k1z/2) #setup for k2 evaluation

self.k2x=deltatime\*self.sumofforces\_x

self.k2y=deltatime\*self.sumofforces\_y

self.k2z=deltatime\*self.sumofforces\_z

self.forces(time+deltatime/2,mass,x+self.k2x/2,y+self.k2y/2,z+self.k2z/2) #setup for k2 evaluation

self.k3x=deltatime\*self.sumofforces\_x

self.k3y=deltatime\*self.sumofforces\_y

self.k3z=deltatime\*self.sumofforces\_z

self.forces(time+deltatime,mass,x+self.k2x,y+self.k2y,z+self.k2z)

self.k4x=deltatime\*self.sumofforces\_x

self.k4y=deltatime\*self.sumofforces\_y

self.k4z=deltatime\*self.sumofforces\_z

self.x\_v=self.x\_v+(self.k1x + 2\*self.k2x + 2\*self.k3x + self.k4x)/6

self.y\_v=self.y\_v+(self.k1y + 2\*self.k2y + 2\*self.k3y + self.k4y)/6

self.z\_v=self.z\_v+(self.k1z + 2\*self.k2z + 2\*self.k3z + self.k4z)/6

self.x=self.x+self.x\_v \* deltatime

self.y=self.y+self.y\_v \* deltatime

self.z=self.z+self.z\_v \* deltatime

def output(self):

print self.x, self.y, self.z, math.sqrt(self.x \* self.x + self.y \* self.y + self.z \* self.z)

class floatingmass:

def \_\_init\_\_(self,mass,r,v):

self.mass=mass

self.rad\_dist=r

self.v=v

self.x\_azimuth=random.uniform(0,2\*Pi) #angles in radians

self.z\_azimuth=random.uniform(0,Pi)

self.x\_coord=self.rad\_dist\*math.sin(self.z\_azimuth)\*math.cos(self.x\_azimuth)

self.y\_coord=self.rad\_dist\*math.sin(self.z\_azimuth)\*math.sin(self.x\_azimuth)

self.z\_coord=self.rad\_dist\*math.cos(self.z\_azimuth)

if self.x\_azimuth<2\*Pi and self.x\_azimuth>Pi:

self.y\_v=-1 #deciding direction

else:

self.y\_v=1

if self.z\_azimuth<Pi/2 and self.z\_azimuth>0:

self.z\_v=-1

else:

self.z\_v=1

if self.x\_azimuth>Pi/2 and self.x\_azimuth<3\*Pi/2:

self.x\_v=1

else:

self.x\_v=-1 #directions settled by this point

self.v\_sq=self.v\*self.v

self.v\_x\_sq=random.uniform(0,self.v\_sq)

self.v\_y\_sq=random.uniform(0,self.v\_sq - self.v\_x\_sq)

self.v\_z\_sq=random.uniform(0,self.v\_sq - self.v\_x\_sq - self.v\_y\_sq)

self.x\_v=self.x\_v \* math.sqrt(self.v\_x\_sq) #final initialisation of velocities,we can unbias this using similar slack procedure as previously used

self.y\_v=self.y\_v \* math.sqrt(self.v\_y\_sq)

self.z\_v=self.z\_v \* math.sqrt(self.v\_z\_sq)

def move(self,deltatime):

self.x\_coord = self.x\_coord + deltatime \* self.x\_v

self.y\_coord = self.y\_coord + deltatime \* self.y\_v

self.z\_coord = self.z\_coord + deltatime \* self.z\_v

def main():

countreached=0 #count of comets reaching earth's orbit

print "Hello and welcome to the oort cloud simulator"

no\_of\_comets=input("Enter number of comets you want to process in this simulation:")

rin=input("Enter the inner radius of the oort cloud in AU:")

rout=input("Enter the outer radius of the oort cloud in AU:")

comets=[]

for i in range(no\_of\_comets):

temp=comet(rin,rout)

comets.append(temp)

#in this version we are treating the floatingmass as something that can cause significant variation in oort cloud's trajectory, also itself is not perturbed by anything

m=input("Enter floating mass's mass as a multiple of Sun's Mass:")

v=input("Enter floating mass's velocity in meters per second:")

r=input("Enter floating mass's distance from sun in AU:")

mass=floatingmass(m,r,v)

bin\_values=[]

time=input("Enter time over which you wish to run this simulation in years:")

interval\_time=input("Enter interval of time over which you wish to calculate the simulator's results in years:") #the higher the bin time the more faster the calclutation but less accurate the result

#time = time \* year

#interval\_time = interval\_time \* year

tr\_ratio=time/interval\_time

it=0

for i in range(tr\_ratio): #we will use this list for the computation purposes

bin\_values.append(it)

it=it+interval\_time

k=0

counter=0

for i in range(no\_of\_comets):

comets[i].output()

print "Number of iterations needed = ",tr\_ratio

for i in bin\_values: #the main loop

print "Iteration number",k,"beginning"

k=k+1

for j in range(no\_of\_comets):

comets[j].orbits(i) #resetting the orbits

comets[j].evaluate(interval\_time,i,comets[j].x,comets[j].y,comets[j].z,mass) #the real evaluator, need to learn runge kutta method for this

if j%100==1:

comets[j].output() #binning routine

mass.move(interval\_time)

for i in range(no\_of\_comets):

comets[i].output()

if(math.sqrt(comets[i].x \* comets[i].x + comets[i].y \* comets[i].y + comets[i].z \* comets[i].z)<=1\*AU):

counter=counter+1

print "Number of comets reaching earth is:", counter

main() #the final function call