Global Scientific JOURNALS

GSJ: Volume 10, Issue 3, March 2022, Online: ISSN 2320-9186
www.globalscientificjournal.com

Solving N-body problems using the Barnes Hut Algorithm

*Rohaan Advani (student, College of Engineering Pune)

----------**ABSTRACT**----------

An N-body problem is the problem of predicting individual motions of a group of objects interacting with each other through physical forces. The Barnes Hut Algorithm is an effective solution to approximate physical forces a system of objects has on a given point. This paper attempts to use the Barnes Hut Algorithm to solve two N-body problems – Estimation of Newtons' Gravitational Force at a given point using a set of live tracked satellites in a three-dimensional environment and Estimation of Coulombs' Electrical Force at a given point using a set of charged particles on a two-dimensional plane.

Date of Submission: - -2022 Date of Acceptance: - -2022

----

1. INTRODUCTION

The conventional method to calculate N-body forces was to consider all pairs of individual objects and add up the contributions of each interaction. This method leads to a quadratic time complexity - as the number of points n increases, the running time grows proportionally to n^2, quickly leading to intractably long calculations.

To tackle this problem the Barnes Hut Algorithm was introduced. It accelerated computation and made large-scale simulation possible. In exchange for a small amount of error the algorithm significantly speeds up calculation making running time proportional to n*log(n). The Algorithm has 3 main steps – constructing the spatial index, calculating center of mass / charge of system of objects and estimating forces caused by system at a given point.

To solve spatial problems, we use efficient data structures such as Quadtrees and Octrees which helps computing two-dimensional and three-dimensional problems respectively. These data structures will provide us with an environment to add the satellites / charged particles to. Moreover, parent nodes of respective child nodes in trees will be used in order to store center of mass / charge values, further storing the center of mass / charge of a system of objects in the root node. These coordinates will then be used to estimate the force the system of objects applies on a given point.

For the purpose of this paper, the coordinates of satellites in three-dimensional space will be denoted by live latitude, longitude and altitude values and the coordinates of charged particles in planar space will use the two-dimensional cartesian coordinate system. Using this calculator, we can estimate gravitational forces applied by a set of satellites and electrical forces applied by a set of charged particles at a given point.

2. IMPORTANT TERMINOLOGY

- Quadtree - A quadtree is a tree data structure in which each internal node has exactly four children. Quadtrees are the two-dimensional analog of octrees and are most often used to partition a two-dimensional space by recursively subdividing it into four quadrants or regions.

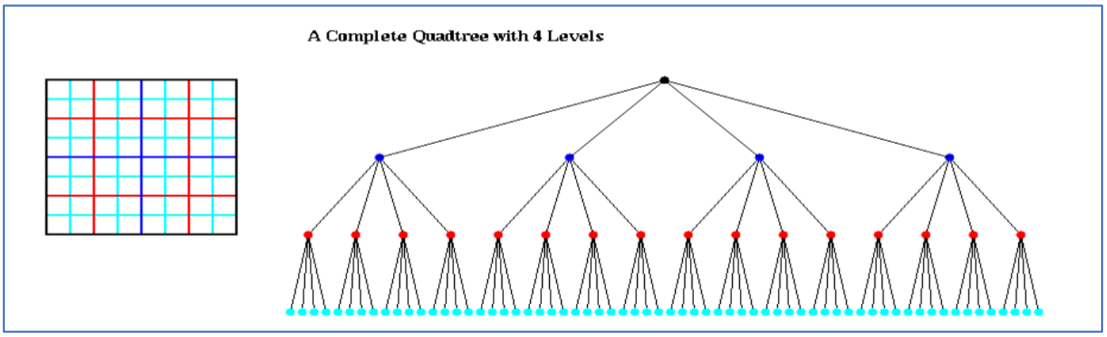


Fig 1: Quadtree Spatial Representation

- Octree - An octree is a tree data structure in which each internal node has exactly eight children. Octrees are most often used to partition a three-dimensional space by recursively subdividing it into eight octants. Octrees are the three-dimensional analog of quadtrees.

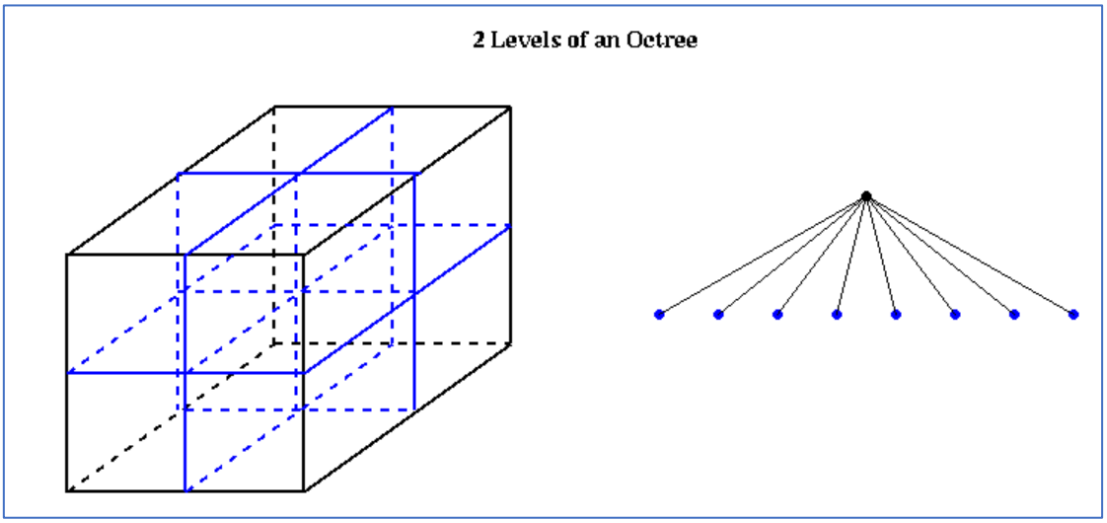


Fig 2: Octree Spatial Representation

- Center of mass - The center of mass is a position defined relative to an object or system of objects. It is the average position of all the parts of the system, weighted according to their masses.

$$X_{com} = \frac{\sum_{i=0}^{n} m_i x_i}{M} \; Y_{com} = \frac{\sum_{i=0}^{n} m_i y_i}{M} \; Z_{com} = \frac{\sum_{i=0}^{n} m_i z_i}{M}$$

- Center of charge – The center of charge is a position defined relative to a system of electrical charges. It is the average position of all partys of the system, weighted according to their charges.

- Newtons' Gravitational force – Force is directly proportional to product of masses and inversely proportional to square of distance between them. It is equated using the universal gravitational constant G which is 6.67408×10^{-11} m^3 kg^{-1} s^{-2}.

$$\begin{aligned}\mathbf{F}_i &= \sum_{j\neq i}\mathbf{F}_{ij} = \sum_{j\neq i} G m_i m_j \frac{\mathbf{r}_{ij}}{|\mathbf{r}_{ij}|^3} \\ &= \sum_{j\neq i} G m_i m_j \frac{(x_j - x_i)\hat{\mathbf{i}} + (y_j - y_i)\hat{\mathbf{j}} + (z_j - z_i)\hat{\mathbf{k}}}{[(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2]^{\frac{3}{2}}}.\end{aligned}$$

- Coulombs' electrical Force – Force is directly proportional to product of charges and inversely proportional to square of distance between them. It is equated using coulombs' constant which is 8.988×10^9 $N\cdot m^2\cdot C^{-2}$.

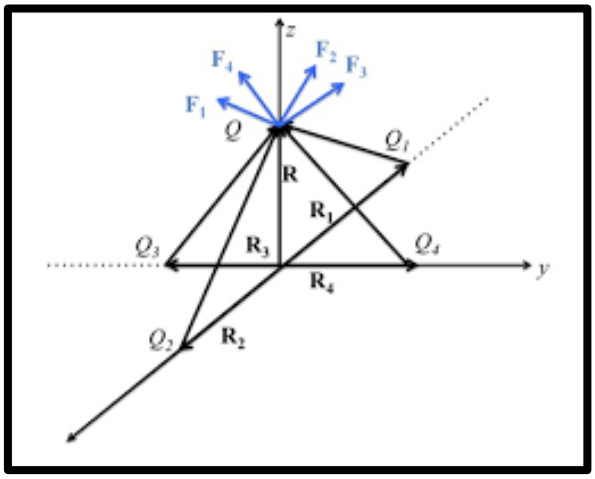


Fig 3: Forces applied by charges in a plane on a point

3. PROPOSED METHOD

Step 1: Design Quadtree / Octree Node

The Quadtree / Octree nodes should contain the following variables for the purpose of solving our problems:

1. Pointers to children – 4 pointers in case of Quadtree and 8 pointers in case of Octree.
2. Number of Leaves of Node – To keep track of limit for further splitting of special coordinates.
3. Center of mass / charge values – X/Y for 2D and X/Y/Z for 3D Space respectively.
4. Key Value – Charge in case of Quadtree and Mass incase of Octree for respective applications.
5. Bounds – Upper / Lower and Middle bounds of node for spatial splitting.

Step 2: New node Function

The new node function will have the following functionality:

1. Allocate memory for storing node.
2. Set lower and upper bounds – For the root node the lower and upper bounds will be available as world size entered by user / programmer. For child nodes, the lower and upper bounds will be set to the appropriate octant with respect to the parent node.
3. Set middle bound – (Lower bound + Upper bound) / 2 for all nodes.
4. Initialize center of mass / charge, child pointers and number of leaves of node.
5. Return node.

Step 3: Destroy Tree Function

After use trees created must be destroyed in order to prevent wastage of space. The destroy tree function will recursively free all child nodes.

Step 4: Add and Subtree Functions

The add function algorithm will work as follows:

1. If tree empty center of mass / charge and key values are added to root node.
2. If node is a leaf place its values in an appropriate child node making it an internal node.
3. Add newly created node to appropriate child node using subtree functionality.
4. Increment number of leaves by 1 if child node added.

The subtree function is used to decide the appropriate child node the values should be added to based of relative spatial coordinates.

On comparison of the bounds of the newly created node with a node available in the system the appropriate quadrant / octant is chosen and the new bounds of the newly created node are defined.

Step 5: Tree Calculation using the Barnes Hut Algorithm

The center of mass / charge values of child nodes of a particular node is calculated using the center of mass / charge formulas and stored in parent nodes recursively. This is done until the root node has the positional center of mass / charge value of the system as a whole. Incase the node is a leaf node the center of mass / charge value is its actual positional coordinates in the system.

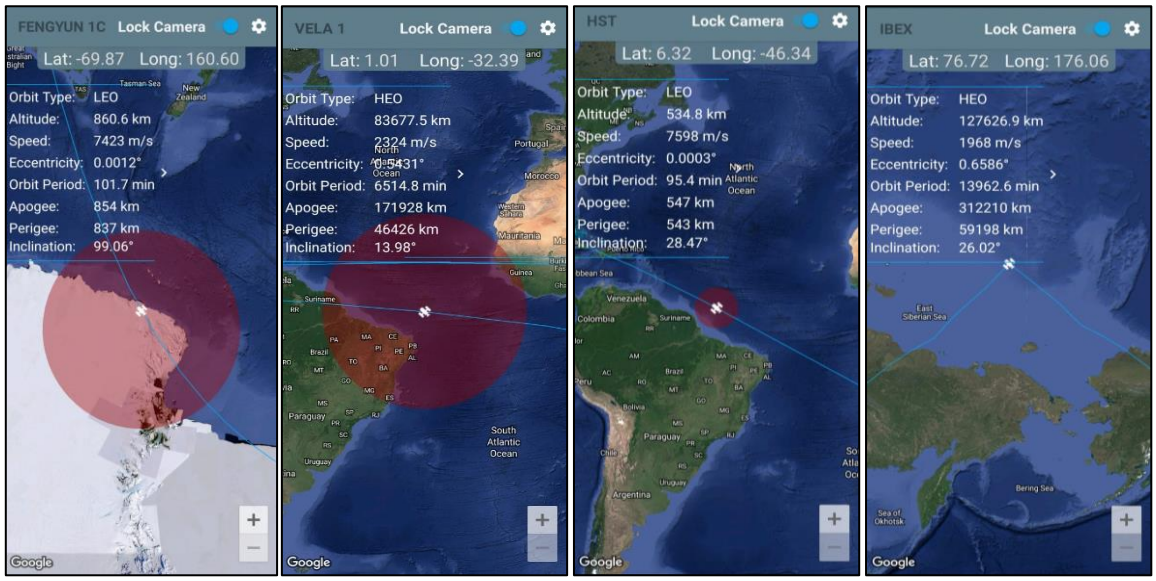
Step 6: Calculate Force the system of objects applies on a given point

After running the Tree Calculation Function the center of mass / charge values of the system are stored in the root node of the tree. If a set of coordinates is given by the user / programmer, the force applied on these coordinates can now be calculated using Newtons' Gravitational Force and Coulombs' Electrical Force formulae for X/Y/Z and X/Y directional respectively. The net force then can be calculated by taking root of sum of squares of the above calculated directional forces.

Step 7: Traverse Tree Functionality

The purpose of this function is to test if the tree is getting built correctly as per requirement for calculations. Also, this function can be used to verify the positional center of mass / charge values calculated by the Tree calculation function.

Step 8: Perform Live Tracking of a set of satellites in order to build system



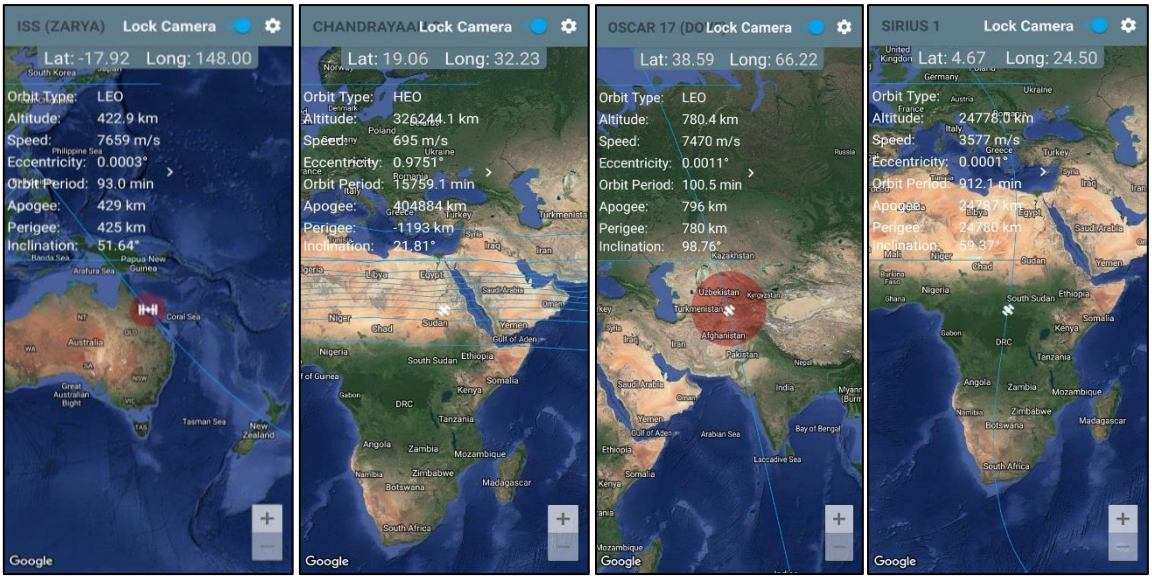


Fig 4: Live Satllite Tracking

Step 9: Build Test Dataset for satellite system

South and West Hemisphere is –'ve whereas North and East Hemisphere is +'ve.

Name of satellite	Latitude	Longitude	Altitude(km)	Mass(kg)
International Space Station	-17	148	422	419700
Hubble Space Teliscope	6	-46	534	11110
Fengyun 1c	-70	160	860	950
IBEX	76	176	127626	80
Chandrayaan 2	19	32	326244	2380
Vela 1	1	-32	83677	150
Sirius 1	4	24	24778	3800
Oscar 17	38	66	780	13

Table: Satellite Dataset for testing

Step 10: Make check functionality

In order to verify that Latitude, Longitude and Altitude variables entered by user / programmer is valid we will have to check them with appropriate ranges.

1. Latitude between -90 and 90
2. Longitude between -180 and 180
3. Altitude above 100 Km – min Altitude satellites should be at.
4. Mass greater than 1 Kg

Step 11: Standardize Coordinates

Latitude and Longitude values are global coordinates. But for the purpose of this paper, we require a consistent coordinate system with units of all 3 dimensions equal. Thus, Latitude and Longitude values are converted to km as follows:

1. Latitude * 110 – Each Latitude is 110 Km apart.
2. Longitude * 85 – Average distance between longitudes is 85 Km at 40 N and 40 S.

Step 12: Build Charge System for Testing

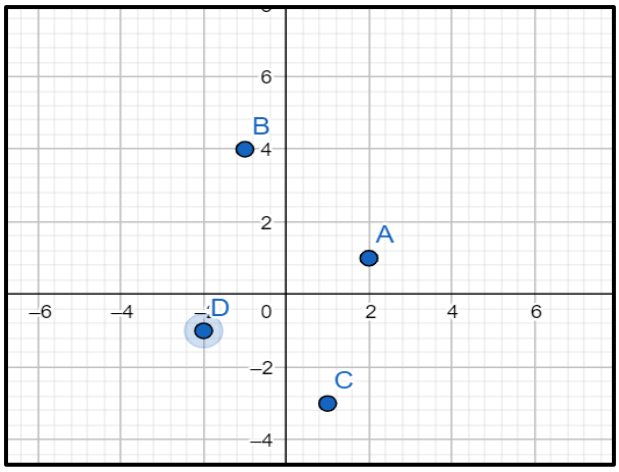


Fig 5: Charge System Built on a 2-Dimensional Plane

EXAMPLE SYSTEM:

A = +2Q

B = +4Q

C = -Q

D = -2Q

Step 13: Test functions with Test Datasets

If above procedure is followed, the system of masses / charges is built in 3 dimensions / 2 dimensions respectively with a world size which is defined by the user / programmer and the test point coordinates specified by user / programmer in order to calculate forces between the system and test point.

While running functions make sure to run new node function in the beginning with bounds of the world, then the add function iteratively adding nodes for test dataset to system and the tree calculation function before force calculation.

By maintaining this order, we have effectively computed Newtons' Gravitational Force and Coulombs' Electrical Force applied by a system of Satellites / Charges bounded in a 3 dimensional / Planar Environment on a test point.

4. REFERENCES

[1] https://en.wikipedia.org/wiki/N-body_problem

[2] https://jheer.github.io/barnes-hut/

[3] https://en.wikipedia.org/wiki/Quadtree

[4] https://en.wikipedia.org/wiki/Octree

[5] https://www.khanacademy.org/science/physics/linear-momentum/center-of-mass/a/what-is-center-of-mass

[6] https://en.wikipedia.org/wiki/Newton%27s_law_of_universal_gravitation