

Tree-code algorithms

This algorithm allows an *approximate* calculation of the forces and potential in an N -body problem in $\mathcal{O}(N \log n)$ steps, which is a huge improvement on the straightforward $\mathcal{O}(N^2)$ enumeration of all pairs of bodies. The approximation uses the multipole expansion of the potential and forces, and it can be improved by increasing the number of multipoles that are taken into account.

Barnes-Hut tree-code algorithm for astrophysics

This N -body problem involves point masses moving in 3-dimensional space and interacting via gravitational forces. The algorithm works roughly as follows:

Consider a mass at some point in space. All of the other masses are divided into clusters that do not overlap with one another. The size of a cluster increases with distance from the point mass being considered. Each cluster is approximated by a point particle whose mass is the sum of the masses in the cluster, and whose position is at the center of mass of the cluster. The net force on the point particle is approximated by the sum of the forces exerted on it by each of the cluster considered as another point mass. This procedure must be repeated for each of the N bodies in the system!

The tree-code algorithm is a clever way of dividing the N particles into clusters simultaneously. It makes use of a data structure called an *octree*. A *tree* is a data structure which consists of *nodes*. A node has at most one *parent* node. A node can have any number of *child* nodes or *children*. A tree has

- one node called the *root* which has no parent, and
- zero or more additional nodes, each of which is a descendent of the root node.

In an octree, every node has either eight or zero children.

In the Barnes-Hut algorithm, the root node in the tree of clusters is taken to be all of a cubic volume of space which contains the N bodies. Slice this cube in half in each of the 3 dimensions x , y , and z . This yields the 8 children of the root cluster, each of which is a cube with $1/8$ the volume of the whole system. This procedure is repeated on each of the eight cubes to give $8 \times 8 = 64$ cubes at the next generation of children.

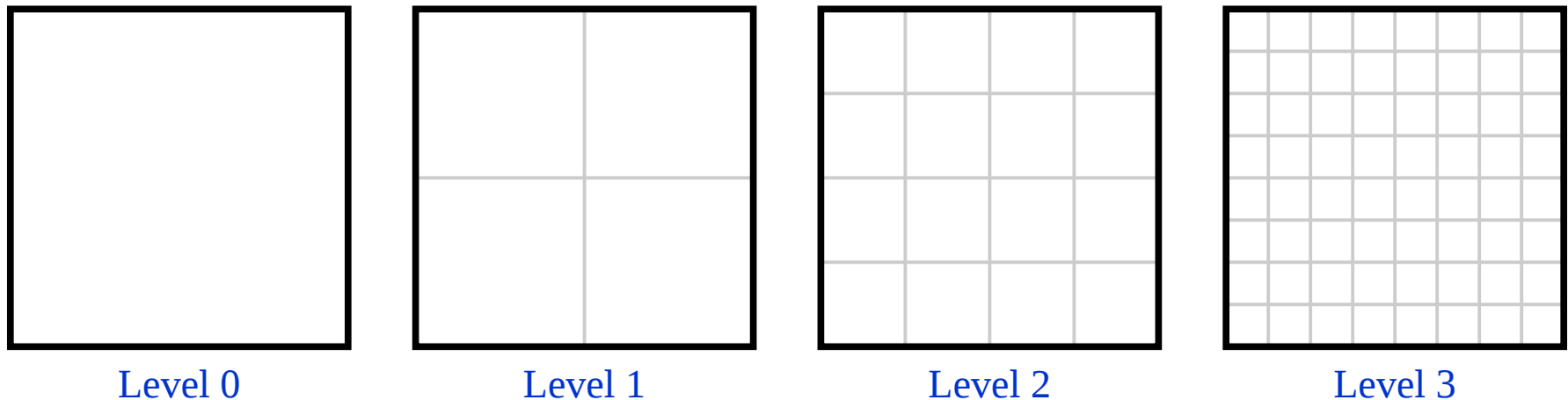
This procedure is repeated for some fixed number of generations. Alternatively, the subdivision can be repeated until each terminal cube contains at most one body: along the way, cubes which contain no bodies can be pruned from the tree so that the force calculation does not waste time on empty cubes.

After the tree has been constructed, an efficient way of computing the masses and centers-of-mass of clusters, and of evaluating the forces between each particle and its surrounding hierarchy of clusters, must be implemented.

A tree-code algorithm in two dimensions

We consider the simpler two-dimensional problem of N charges in a plane, which has a convenient representation in terms of complex numbers. This problem is described in §8.7.2 of the textbook, and in more detail in the reference given to L. Greengard, *Computers in Physics*, 4, 142 (1990).

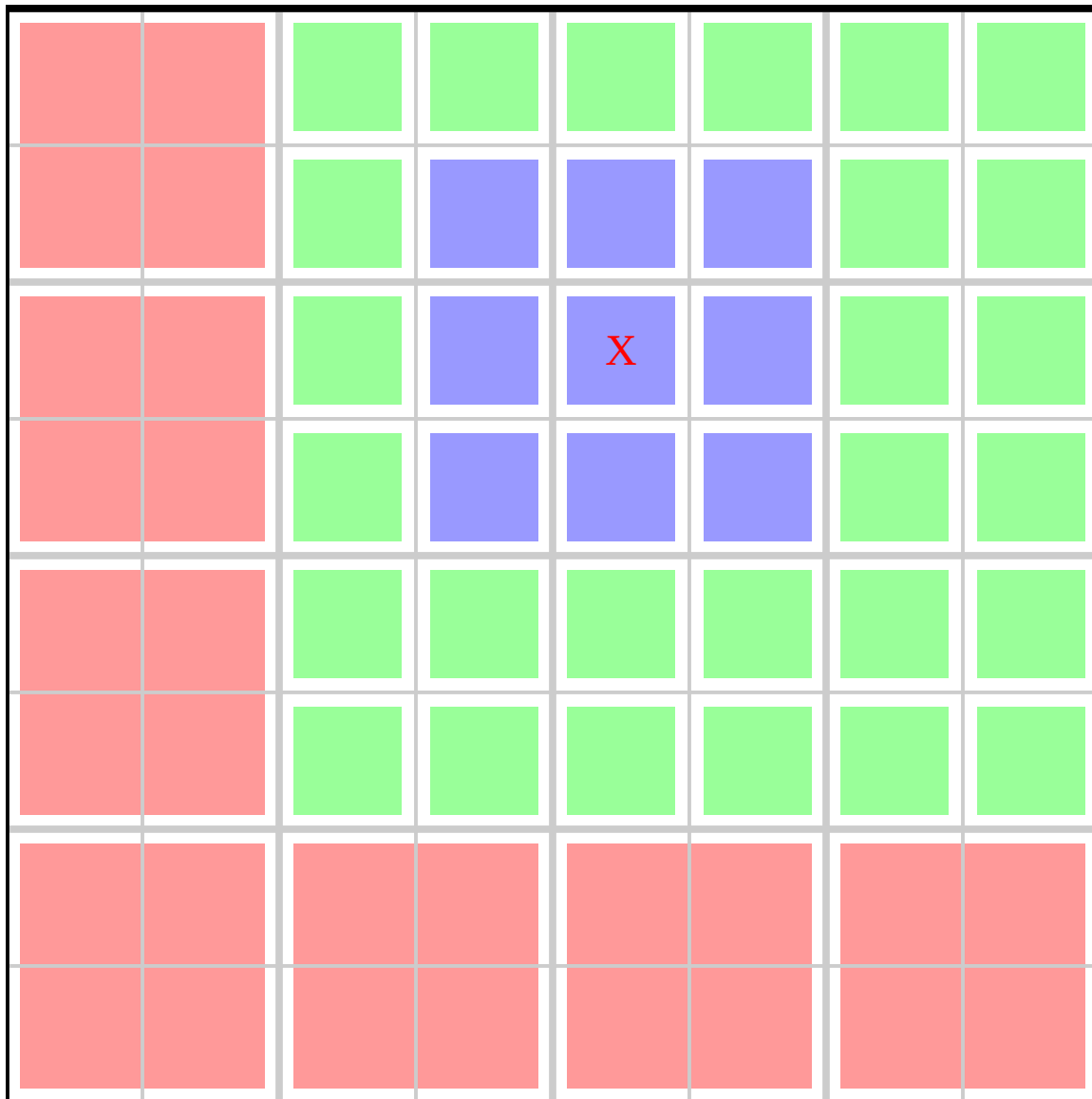
The charges are contained in a square region in 2-D space. The square is repeatedly divided into smaller squares of $1/4$ the area. This produces a *quadtrees* structure in which each node has either zero or 4 children.



In the figure, level 0 represents the root node, and the subdivision is carried out to 3 generations of children with $4^1 = 4$, $4^2 = 16$, and $4^3 = 64$ children.

The basic *divide and conquer* strategy which leads to $\mathcal{O}(N \log N)$ behavior can be understood from these numbers. Suppose that $N = 64$, and the charges are distributed roughly uniformly in the square. Then after 3

levels of subdivision, each terminal square will contain one charge on average. It therefore makes sense to terminate the subdivision at this stage. In general, the number of subdivision levels can be taken to be $\lceil \log_4 N \rceil$.



Consider a charge located at the point marked “X” in the figure. We need a strategy to divide the other charges

into clusters. Distant clusters can contain a large number of charges, while nearby clusters must be smaller. This hierarchical division must be made for *each* of the charges in the system! To do this, some definitions are needed.

Consider the squares or *boxes* at any particular level $\ell = 0, 1, \dots, \lceil \log_4 N \rceil$. There are 4^ℓ boxes at this level.

- **Definition 1:** Two boxes are said to be *near neighbors* if they are at the same level and they share a boundary point. A box is considered to be a near neighbor of itself.
- **Definition 2:** Two boxes are said to be *well separated* if they are at the same level and are *not* near neighbors.
- **Definition 3:** With each box is associated an *interaction list*, which consists of the children of the near neighbors of its parent which are well separated from it.

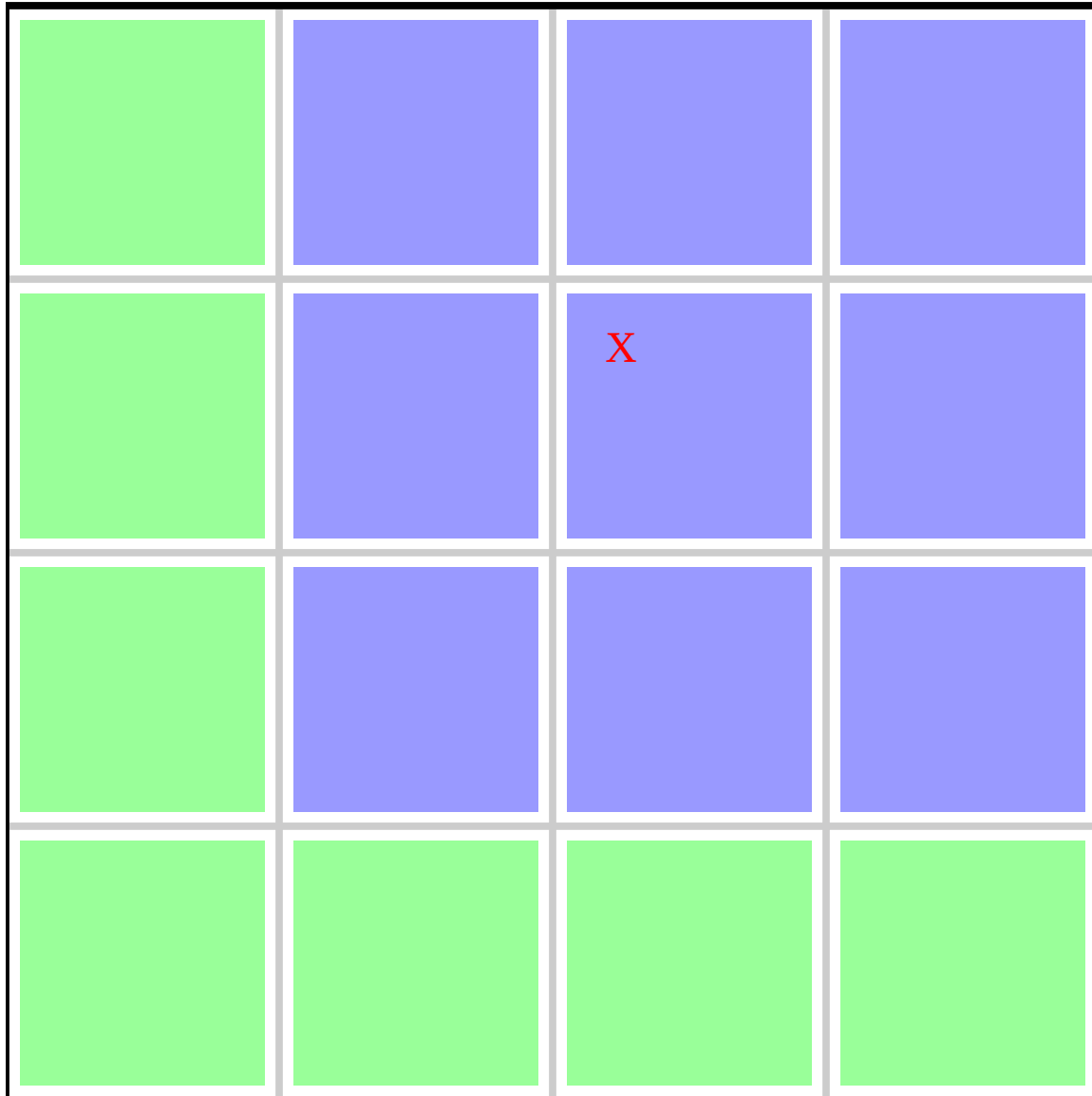
Consider level 3 boxes in the figure. The box marked “X” has 9 *near neighbors* (including itself) which are shaded blue. The other $64 - 9 = 55$ level 3 boxes in the figure are *well separated* from “X”. The *interaction list* of box “X” consists of the 27 boxes that are shaded green in the figure. The *parent* of box “X” consists of “X” and the three neighbor boxes to the East, South, and South-East of it. This parent box at level 2 has 9 near neighbors, and these 9 level-2 boxes have a total of 36 children, which are the level 3 boxes shaded green and blue. The 27 green boxes are *not* near neighbors of box “X”, but *are* children of its parent’s near neighbors, and therefore belong to its *interaction list*.

Next, consider level 2 boxes in the figure on the next page. The box marked “X” has 9 near neighbors shaded blue, and an interaction list which consists of the 7 boxes shaded green which are the children of its parent’s near neighbors that are well separated from it.

We can now specify a procedure for computing forces on any particle, for example a charge located in the level-3 box marked “X”. The general rule is that forces are computed *only* between the particle and other particles which belong to a box in an interaction list. In the example discussed in the figures above, there are only two interaction lists, one at level 3, and one at level 2. Furthermore, these two lists are *disjoint*, i.e., there is no particle which belongs to both of them. This is important to ensure that forces between any two particles are only computed once! Notice also that there are no interaction lists at levels 0 and 1 because all boxes at these two top levels are near neighbors.

However, there are particles which do not belong to either of these two interaction lists, namely the particles

in the near neighbor boxes at level 3: including these accounts for all particles in the system!



The general algorithm for computing forces is as follows:

- Divide the particles into a quadtree of boxes with some number of levels, which is usually fixed at $\lceil \log_4 N \rceil$ for uniform systems.
- For each particle in the system, visit all levels ≥ 2 . At each level,
 - find the multipole moments of each box in the interaction list, and
 - compute the force (or potential energy) of the particle using this multipole approximation. For example, at zeroth multipole order, the particle simply interacts with the next charge of the box located at its center.
- Finally, at the deepest level $\ell = \lceil \log_4 N \rceil$, compute the interaction of the particle with *all* of the charges in the near neighbor boxes.

Scaling of the tree-code algorithm.

At each level, the amount of computation that needs to be done scales like $\mathcal{O}(N)$: If M is the order of the multipole expansion used, then $\mathcal{O}(M \times N)$ operations are needed to compute the multipole coefficients; and each particle interacts with at most 27 boxes, which is the maximum number in its interaction list. At the deepest level, the interactions of each particle with the charges in its near neighbor boxes also need to be computed: for a roughly uniform system, each box at the deepest level contains only one particle on average, so the number of operations required is of $\mathcal{O}(8 \times N)$ (there are 9 near neighbor boxes, but a particle does not interact with itself). Since there are $\mathcal{O}(\log N)$ levels to be visited, the whole computation scales like $\mathcal{O}(N \log N)$.

A tree-code algorithm program

We will develop a C++ code that implements this algorithm for the 2-D Coulomb problem. Following the suggestions in the textbook, we will not implement a full quadtree data structure. A simpler procedure is to iterate over the levels and use C++ STL `lists` to manage the lists of particles in the various boxes.