RADIATIVE TRANSFER IN GALAXIES

RADIATIVE TRANSFER IN GALAXIES

By
RORY WOODS, B.Sc., M.Sc.

A Thesis
Submitted to the School of Graduate Studies
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

McMaster University

DOCTOR OF PHILOSOPHY (2015) McMaster University
(Physics and Astronomy) Hamilton, Ontario

TITLE: Radiative Transfer in Galaxy Formation

AUTHOR: Rory Woods, B.Sc. (Mount Allison University), M.Sc. (McMaster University)

SUPERVISOR: Professor James Wadsley

NUMBER OF PAGES: 1

Abstract

In this thesis, we present a novel algorithm for computing the radiation field in astrophysical simulations.

Dedicated to...

Acknowledgements

Thank you to all that helped.

"Some sort of quote?"

Albert Einstein (1879-1955)

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Chapter 2

Radiative Transfer

This chapter will contain an overview of current radiative transfer methods and where we stand.

2.1 The Radiative Transfer Problem

$$j = \frac{dE}{dV d\Omega dt} \tag{2.1}$$

$$dI = jds \tag{2.2}$$

$$dI = -\alpha I ds = -n\sigma I ds = -\rho\kappa I ds \tag{2.3}$$

$$\frac{dI}{ds} = -\alpha I + j \tag{2.4}$$

$$I(s) = I(s_0) + \int_{s_0}^{s} j(s')ds' \tag{2.5}$$

$$I(s) = I(s_0) \exp\left[-\int_{s_0}^{s} \alpha(s')ds'\right] = I(s_0) \exp\left[-\tau(s)\right] \tag{2.6}$$

$$\tau(s) = \int_{s_0}^{s} \alpha(s')ds' = \int_{s_0}^{s} \rho(s')\kappa(s')ds' \tag{2.7}$$

2.2 Current Methods

Chapter 3

The Numerical Method

In the absence of absorbing material, the problem of radiative transfer reduces down to that of gravity. As such, the tree-algorithm for calculating gravity can be used [2].

- A tree can be used to partition space.
- Each level of the tree holds finer partitions of the volume. See figure 3.1
- Each node of the tree contains accumulated information about the tree below it (total mass, etc.).
- In order to calculate gravity on a particular leaf (bucket), you can interact with the moment of another cell (3.1).
- To decide what level of the tree to interact with, you can define an opening angle/radius, θ. If a cell is smaller than this opening angle (the distribution of matter inside the cell is contained within a small enough angle on the sky), the entire cell can be used in the force calculation. If not, you must consider the child nodes separately. See equation 3.3.

- On average, the number of interactions a each particle will have is $\log N$, where N is the total number of particles. Thus, the force calculation for the whole simulation scales as $N \log N$. Note that lowering θ shifts the number of calculations that are approximated by large cells to smaller cells, and thus if θ is very small, the code approached scaling of order N^2.

- In the case of radiation, the math is very similar (See eq 3.2). However, since radiation does not cancel like forces, the dipole moment does not disappear and a rougher approximation is possible (wording wrong, fix this).

- In this case, the interaction scales as $N_{\rm sink} \log N_{\rm source}$. However, assuming the full tree is still used, the tree-build still scales as $N \log N$.

$$\Phi = \frac{G}{4\pi} \sum_{n=0}^{\infty} \frac{1}{r^{(n+1)}} \int (r')^n P_n(\cos\theta') \rho(\boldsymbol{r}') d\tau' \tag{3.1}$$

$$F = \frac{L}{4\pi r^2} \tag{3.2}$$

3.1 Tree Data Structures

In order to understand the radiative transfer algorithm that we are presenting, it is important to understand tree data structures.

- Terminology: Node, root node, leaf node, interior node, child, parent, sibling, tree build, walk the tree, ascend the tree, descend the tree.

- In computer science, a tree is a hierarchical data structure. Typically the tree starts at a single point, usually called the root node, and branches out to many other "child" nodes.

- Each node in the tree stores some sort of data, and the relative location of the node in the tree indicates the relation of the data in the node to the data in other nodes.

- GASOLINE uses a "k-d tree" for gravity. This is an example of a binary space-partitioning tree. Every node contains 2 children, and each node of the tree represents a particular volume of space. kd-Trees and octrees represent the majority of trees used in astrophysical simulations. See figure 3.1 for visual examples of trees.

3.2 Building a Radiation Tree

- While the algorithm we present is general enough to work with any volume-filling tree, the following sections will introduce the algorithm as we have developed it. GASOLINE uses a k-d tree for its gravity solver, and as such, our version of the algorithm uses this tree type in order to make use of existing tools in the code base.

- The recursive pseudocode for the tree-build is presented below (note - change to nice pseudocode format):

    1. if number of data elements greater than n_{leaf}, partition data
    2. recursively call build tree on each partition of the data set

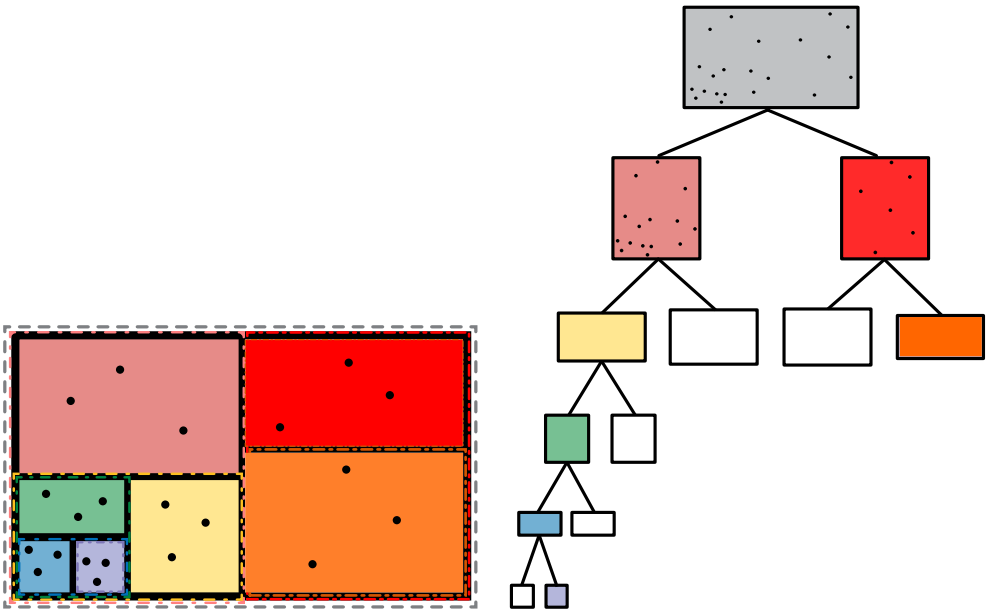


Figure 3.1: This is an example of a binary tree. The volume is represented by a tree node, and each volume is then split into two subvolumes, which are represented by two "child" nodes of the original node. This splitting can continue indefinitely on either side, making the tree an effective way at splitting volumes.

3. else, if number of data elements less than n_{leaf}, calculate basic cell properties

4. After if statement, calculate accumulated cell properties (higher moments, etc)

- In our case, the partition data step involves finding the longest axis of the data contained on the current node and dividing particles to the upper and lower halves of the midway point of that axis.

- Each volume and its corresponding list of particles is then passed recursively to the tree build function again. This terminates when build tree receives a list of particles that is sufficiently short (less than a user set

parameter, n_{leaf}). At this point, basic cell properties such as center of luminosity and total luminosity are calculated.

- Once the leaf nodes have been calculated, more complicated average properties, such as higher moments, can be calculated by looping through all particles in the node. This applies to both leaf and interior nodes.

- The initial partition function does not require that the data be fully sorted, only that it be divided to either side of an intermediate value. This is an order n operation.

- The tree will be roughly of depth $\log(N)$, meaning that the partition will need to be performed $\log N$ times. Therefore, the tree build should scale as roughly $N \log N$.

- For radiation, average cell properties that are used are average density, average opacity, standard deviation of opacity, total luminosity, and center of luminosity.

- Note that we have calculated center of luminosity without taking into account absorption within the cell.

3.3 Exchanging Radiation

Once the tree has been built, calculating the radiation (gravity) at any particular point can be accomplished by traversing the tree structure, a process called a "tree walk."

- First, a "post-order" tree walk is performed in which the children of a node are always checked before its sibling. The walk continues until it

arrives at a leaf node, at which point the radiation (gravity) arriving at that leaf is calculated. This leaf node will be called the receiving leaf.

- The second walk occurs during the radiation (gravity) calculation. We must check what cells are acceptable to interact with based on the opening angle criteria mentioned in the chapter 3 introduction. We can make use of the fact that no children below a cell that has already accepted the θ criterion need be checked, as they are contained within the cell and so automatically satisfy the criteria of their parent.

- The algorithm looks like:

    1. Given a cell (starting with the root cell), check if the distance, r, from the current leaf to the cell is shorter than the opening radius (given by equation 3.4).

    2. If r is shorter than the opening radius, the cell must be "opened," meaning that we return to step 1, but now passing in each child node to check.

    3. If r is longer than the opening radius, the cell is acceptable to interact with. The radiation may be calculated to the bucket using equation 3.5. See figure 3.2.

    4. In the case that the distance to the cell is smaller than the opening radius, and the node has no children (e.g. for leaves of the tree that are very spatially close to the receiving leaf), the interaction should not be approximated, and the direct n^2 summation over all particles in each leaf is performed.

5. Once radiation has been calculated from the current cell, the tree-walk is allowed to skip all children of the current cell and move on to its sibling or parent's sibling (parent's sibling in the case that we are the right-hand child of the parent node).

- Once radiation has been calculated for the receiving bucket, we move on to the next bucket, which is accomplished by moving to the sibling if the current bucket is the left child of the parent node, or to the sibling of the parent node if we are the right child. An example radiation exchange is shown in figure 3.3.

- The above algorithm will run in $N \log N$ time, as with gravity. However, unlike gravity, not all objects emit radiation. Thus, technically the more specific scaling is $N_{\text{sink}} \log N_{\text{source}}$. The slow growth rate of computation time with the number of sources makes the algorithm a very strong candidate for cosmological applications in which there are often similar numbers of star particles to gas particles. In fact, some codes have already made use of this basic idea [3**?** , 4]. As well, the algorithm may allow for the treatment of gas particles as sources (more caerful considerations must be made to tie the radiation to the cooling of the gas in this case), which is rarely done due to incredibly high computational cost.

$$\theta_{\text{crit}} = \frac{b_{\text{max}}}{r} \tag{3.3}$$

$$r_{\text{crit}} = \frac{b_{\text{max}}}{\theta_{\text{crit}}} \tag{3.4}$$

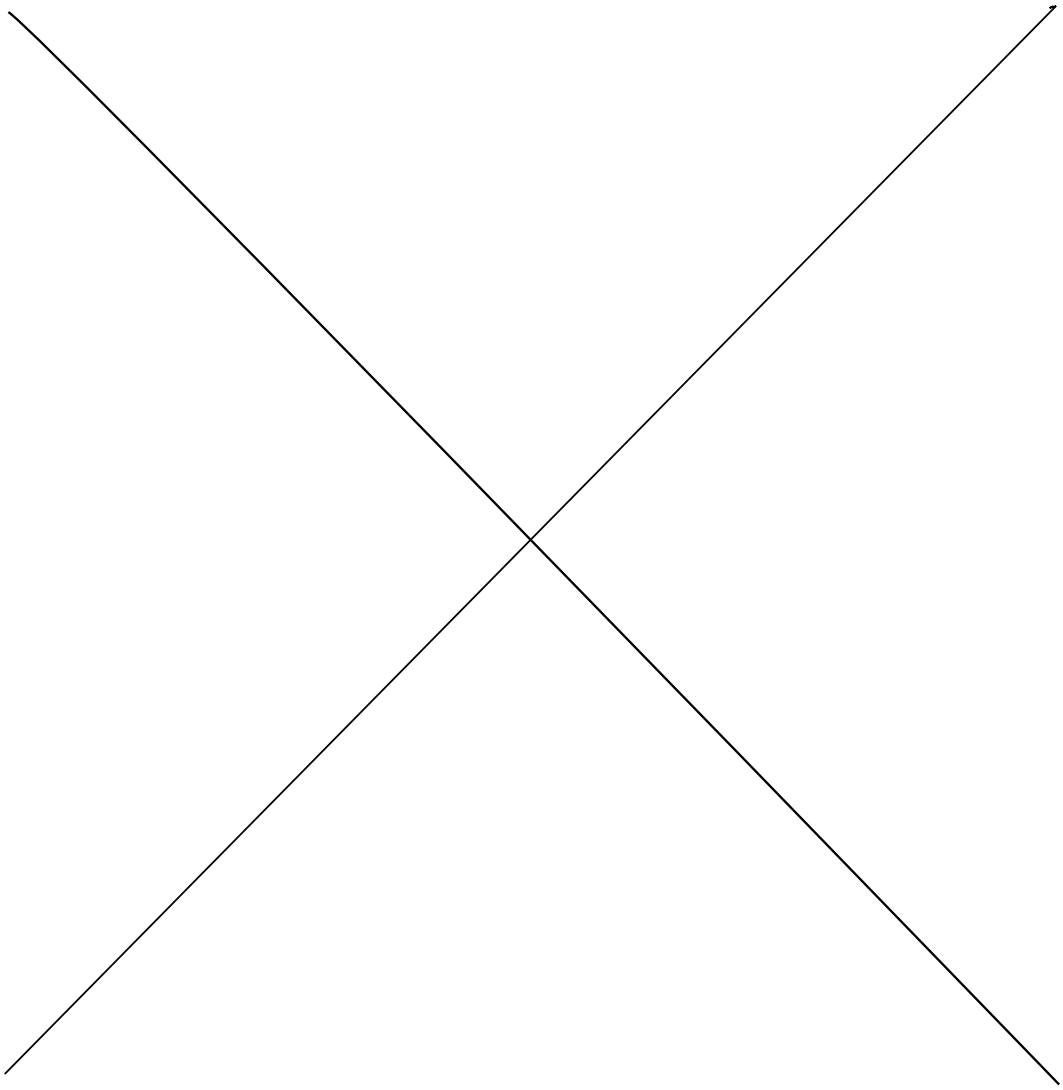


Figure 3.2: Cell A in this image is the receiving cell, while cells B, C, and D are cells that A will receive flux from. Cell A is close enough so that it should be opened, but is a leaf and so it requires a direct n^2 summation. Cell C is close enough and is not a leaf, so it will have its two children checked for the same criteria (the left child will be too close, the right child will be acceptable to interact with). Cell D is not a leaf, but is sufficiently far away that leaf A can interact with the full cell.

$$F_{\text{bucket}} = \frac{L_{\text{tot}}}{4\pi r^2} e^{-\tau} \tag{3.5}$$

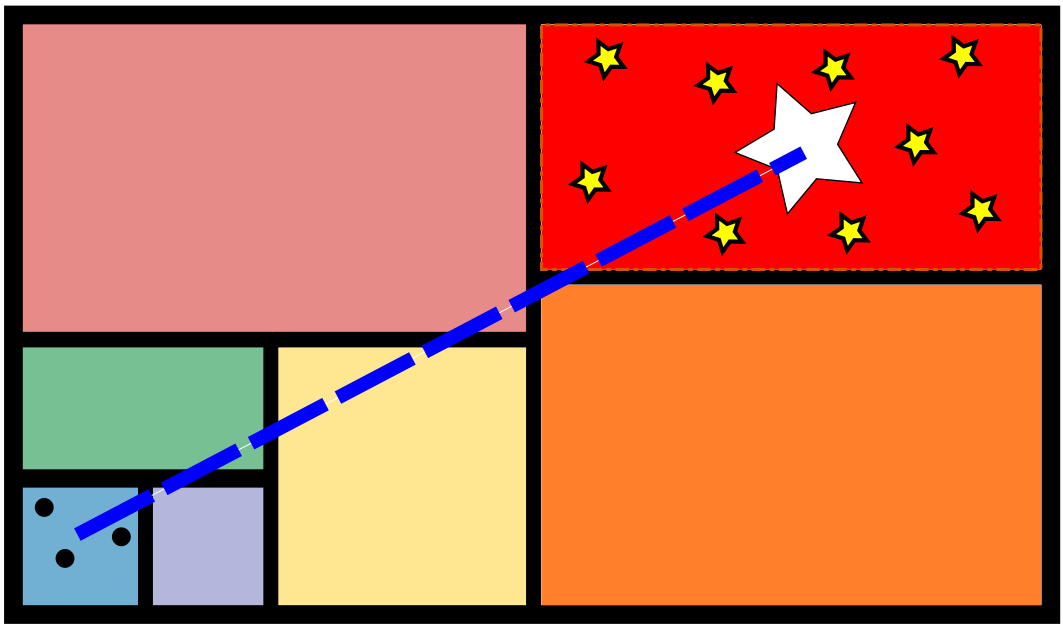


Figure 3.3: In this image, cell A is receiving radiation from cell B. Cell B is sufficiently far away that we can find the center of luminosities of all the sources inside of it, and calculate flux based on that single value rather than summing each one individually.

3.4 Absorption

The algorithm presented in sections 3.2 and 3.3 assumes that no change to the radiation (force) happens in between the sending and receiving buckets. In gravity, this is acceptable because forces are not "absorbed" in any way. However, radiation tends to be absorbed and scattered by intervening material and thus the intensity of the radiation at a point is not only due to the sending source, but to all material in between the source and the sink.

- As mentioned in chapter 1, most current radiative transfer codes either completely ignore intervening material or do very detailed tracing of photons throughout the medium. The former option produces very bad radiation fields while the latter is incredibly computationally expensive.

- In order to reproduce the behavior of equation 2.4, we must modify the algorithm to be able to find the optical depth between any two points. However, we must be careful not to lose the performance afforded by the tree. In order to do this, we have developed the algorithm to make use of the tree during the optical depth calculation as well.

- The crucial point to the algorithm lies in the fact that for any two interacting cells, there exists a common parent node. Thus, all intervening space between the cells must lie within the subtree in which the common parent is the root.

- If we traverse up the depth of the tree (hereafter referred to as a tree climb) once from each interacting node to the common parent node, we will have performed roughly $\log(N)$ extra operations per interaction. If we do no other work than this, then our scaling for radiative transfer changes to $N_{\rm sink} \log N_{\rm source} \log N$. While the extra factor of $\log N$ is certainly worth noting, it does not tend to increase scaling by a significant amount.

- Our goal then becomes to perform order (1) amount of work during this additional tree climb.

- In order to accomplish this, we need only make use of the average properties recorded for each cell during the tree build.

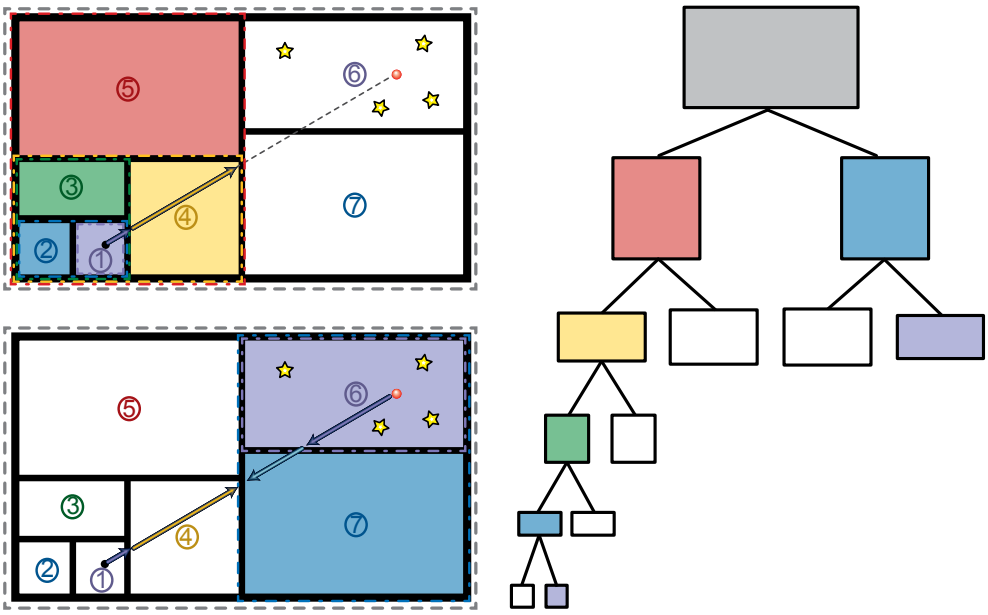
- At each higher cell during the tree climb, we obtain a larger representative volume from that cell. The new volume contains the previous volume as well as a new contribution from the previous cell's sibling. This sibling's volume may or may not lie on the vector connecting the

two interacting cells. This can be determined by calculating the distance to the edge of the current volume along the vector from the centers of the original interacting cells. This operation is completed in order (1) time. (introduce this algorithm?).

- At each new higher cell, if the calculated line segment is longer than the accumulated distance so far, than the difference is the amount of the vector contained in the additional volume. By recording this new line segment, the average density of the cell, and the average opacity of the cell (both values that were accumulated during the tree build), then we have everything needed to calculate the optical depth of the line segment according to equation 2.7. By summing the optical depth of each line segment, we will have obtained the full optical depth between the interacting cells in order $logN$ time. The algorithm is depicted graphically in figure 3.4

3.5 Refinement

While section 3.4 introduces a very fast algorithm for calculating a radiation field, it relies heavily on the geometry of the underlying tree. In volumes with very smooth density/opacity, the above algorithm performs very well. However, in cases with sharp density/opacity gradients, the density/opacity gradient is discretized into widths of order the cell size at the current tree depth. This can become problematic, causing the tree structure to be imposed into the calculated radiation field. In order to solve this, we introduce a refinement process to the algorithm that allows a descent back down the tree



(a) A representation of the algorithm in real space, with the tree overlaid.

(b) The colors correspond to the volumes on the left.

Figure 3.4: The absorption algorithm.

during the tree climb in order to obtain a more detailed description of the medium.

- The refinement is a fairly straightforward addition to the algorithm. At the point where the average properties of the cell would normally be considered, we simply check if the current cell passes a refinement criteria.

- If the cell passes the criteria to refine, rather than recording the average properties, we recursively check the children of section of the tree we did *not* ascend from.

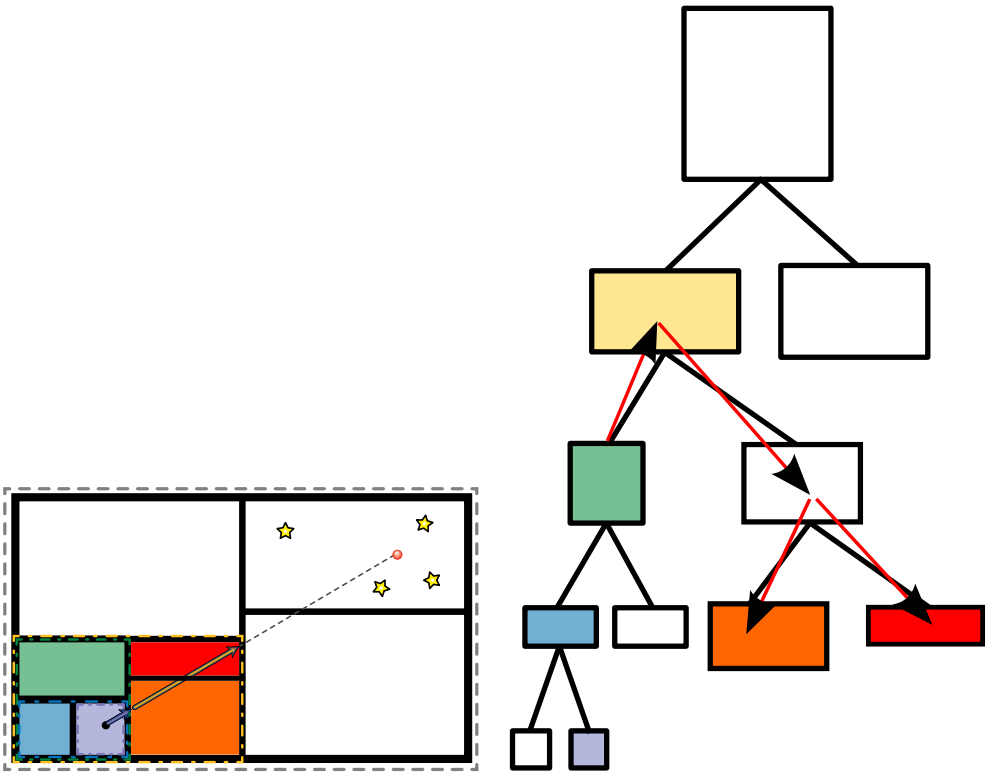
- Once we arrive at a cell that fails the criteria to refine (or at a leaf and can no longer refine), we record the line segment within the cell and the average properties as normal, and return up the recursive call. See figure

3.5 for a visual representation.

- The specific refinement criteria has deliberately been left vague until this point. In principle, one can refine on any cell property desired.
- For the purposes of this paper, we have decided to use an opacity refinement criteria. Within any cell, if a constant times the standard deviation of the average opacity is larger than the average opacity, the cell is refined. We find this produces a reasonable amount of refinement in code tests.
- Note that this is not necessarily the ideal criteria for physical simulations. It would be wise not only to look at the variation in opacity, but also the absolute value. In cases where the optical depth is very high, most of the radiation will be absorbed anyway, and the algorithm can be terminated since this particular vector yields a negligible flux of photons to the receiving cell.

Extension of the refinement to ray-tracing

- If very high accuracy is required, the refinement routine is flexible enough that sub-leaf refinement is possible. While this has not currently been tested since it leaves the regime of low computational expense, it could easily be implemented.
- If a leaf was reached during refinement and still passed the criteria to be refined on, the individual particles inside the cell could be considered.
- A ray tracing scheme through the cell similar to SPHRay [**?**] could be performed. The machinery to do this ray trace is already established for use within the receiving and sending cells (see section 3.6 and figure 3.6).



(a) The tree splits the line segment into the red and orange sections.

(b) The colors correspond to the volumes on the left.

Figure 3.5: When the line segment is too rough in some physical sense, refinement can be triggered. Visually, the algorithm descends back down the tree the opposite direction it came from until the criteria to refine is no longer satisfied or until a leaf is reached.

3.6 Resolving the Receiving Cells

During testing, we ran into issues with ionization fronts "stalling" in certain cells. If a sharp ionization front is passing through a receiving bucket, then the effects of averaging can cause issues if the optical depth of the bucket is of order unity or higher. (below section needs re-wording and more specifics).

- Consider an ionization front that has passed halfway through a leaf node (half of the particles are ionizaed, half are not).

- The average opacity will be $\kappa/2$, where κ is the opacity of the unionized particles.
- The ionized particles will use an opacity that is much too large, therefore reducing the flux that particles at the "rear" of the leaf see.
- This means that particles at the rear of the leaf are harder to ionize than at the front, and the propagation speed of the front is drastically reduced.
- In order to combat this, more detailed tracing is required *only in the receiving leaf.*
- This is easily accomplished by implementing a scheme similar to SPHRay [**?**]. (introduce simpler method that we use where we order particles along vector and linearly add optical depth?). See figure 3.6.

Introducing the ray tracing machinery for the above purpose also creates the ability to ray trace within leaves during the refine mentioned in section 3.5. In principle, this means the code can easily be forced into a full ray trace if this behavior is desired.

3.7 Summary of Algorithm

We have presented a flexible and computationally inexpensive algorithm for calculating the radiation field within a simulation. The algorithm affords many benefits (note: need to introduce many of stated benefits below in previous sections):

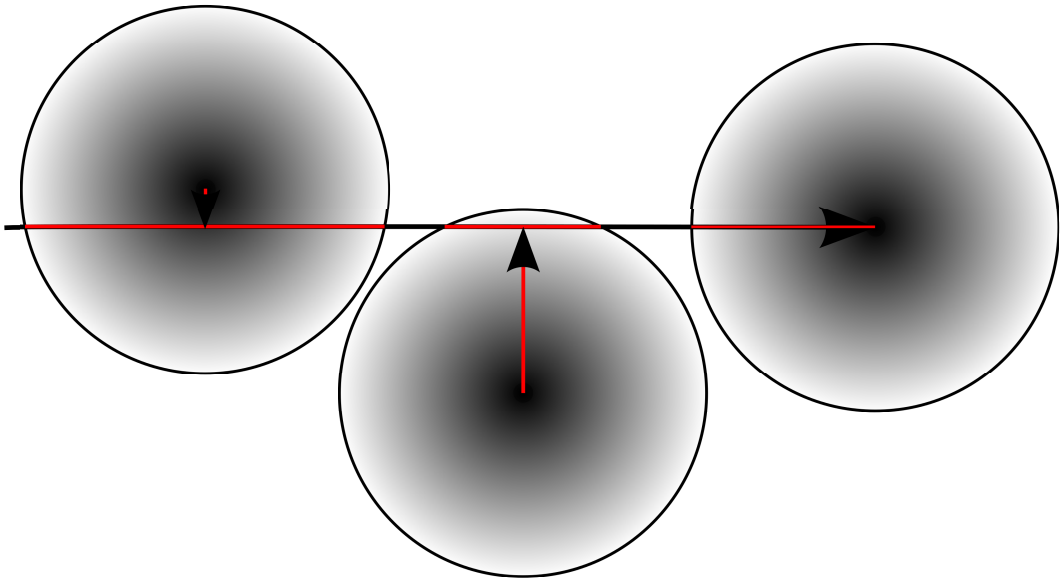


Figure 3.6: Two ray tracing schemes. The first is the scheme of **?**] in which the photons are diminished by optical depth along each particle's smoothing length that they pass through. The second scheme is much simpler and yields very similar results. It relies on the fact that the ray actually represents a very large cone of photons, and that most particles in the cell will probably contribute to the absorption of photons at the receiving particle. It essentially simplifies to setting b = 0 for every particle in the receiving leaf.

- It is flexible enough to allow a wide range of accuracy depending on the application. Speed starts at $N \log N \log N$ and approaches that of ray tracing (check this...) when the algorithm is tuned to that level of refinement.

- Because radiation is transferred instantaneously, the speed of light does not become a limiting time step. If ionization dynamics are important, then the propagation of the ionization front becomes the limiting time step. If only end behavior is required, then the there is very little the algorithm does to limit the time step.

- There is no scan dependence. Because flux is accumulated at each receiv-

ing bucket without explicitly depositing the photons into the intervening material, ionization/heating/cooling is performed completely separate to radiation. This means that the solution will not change based on the order in which the sources are visited.

- The algorithm is independent of wavelength or even number of wavelengths. The algorithm need only perform the tree walk and tree climb a single time in order to obtain the line segments in each cell. Performing different wavebands simply equates to recording multiple average opacities. This enables multi-band radiative transfer at little additional cost.

However, it is important to keep in mind the limitations and assumptions of this algorithm.

- Photons are not explicitly conserved. In order to save computational time, we can not keep track of the photons deposited in intervening material during an exchange. We obtain an optical depth and simply assume that the photons lost in the process have been deposited in the intervening material. When the intervening material is the receiving bucket at a later point in the algorithm, it should receive roughly the correct number of photons due to a matching initial segment (wording...).

- Light is transferred instantaneously, meaning that photon fronts could travel faster than allowed, and that sinks could receive photons from a source too far away to have sent photons there yet.

- Very large opacities in single particles can be problematic for both cooling (calculating the emitted flux from a particle is complex if the par-

ticle itself is optically thick) and ionization propagation. Particle self-absorption can impart the same "stall" in a single particle that was mentioned in section 3.6 for leaves.

- Extra computation time can be required in the heating and cooling code due to intense local radiation fields. However, if the goal is to obtain a radiation field, this is already a built in cost to any algorithm. We simply mention it to suggest that increased computation time is due not only to the radiation algorithm, but the increased computation time for the cooling integrations (remove this point?).

Chapter 4

Discussion

Chapter 5

Conclusions

Appendix A

Appendix A

Bibliography

[1] G. Altay, R. A. C. Croft, and I. Pelupessy. SPHRAY: a smoothed particle hydrodynamics ray tracer for radiative transfer. *MNRAS*, 386:1931–1946, June 2008.

[2] J. Barnes and P. Hut. A hierarchical O(N log N) force-calculation algorithm. *Nature*, 324:446–449, December 1986.

[3] N. Y. Gnedin and T. Abel. Multi-dimensional cosmological radiative transfer with a Variable Eddington Tensor formalism. *New Astron.*, 6:437–455, October 2001.

[4] R. Kannan, G. S. Stinson, A. V. Macciò, J. F. Hennawi, R. Woods, J. Wadsley, S. Shen, T. Robitaille, S. Cantalupo, T. R. Quinn, and C. Christensen. Galaxy formation with local photoionization feedback - I. Methods. *MNRAS*, 437:2882–2893, January 2014.