Light cones in ABACUS

Lehman Garrison

July 30, 2018

TODO: the floating point roundoff issues in §6.4 need more consideration before this is implemented.

1 Introduction

The following document outlines a scheme for implementing light cone outputs in ABACUS. Light cones are an N-body simulation data product that mimic the effect of the finite speed of light: more distant structures are observed earlier in the universe's history. As a data product, light cones are much like an ordinary 3D time slice (containing a set of dark matter particles), but one side of the box will be "older" than the other, depending on where the mock observer was placed.

2 Definitions

To define the desired light cone (LC) output, we imagine a continuous sweep of a spherical surface through the primary periodic zone of the comoving simulation box. The sphere is centered on the LC observer at $z = 0$, and the radius is given by the comoving distance from the observer to the current simulation redshift (so the sweep is inwards). The observer may be inside or outside the box. We imagine the particles moving continuously, and the position of a particle is recorded at the moment the light cone sweeps past.

Note that this definition allows for particles to be seen more than once if a particle crosses a periodic wrap that brings it back inside the light cone. We never want a particle to be seen more than once at a given redshift, though. In the continuous model, we can say that we want to record all of the unique intersections of the particle's world line with the light cone.

Of course, the particles and light cone move in discrete time steps in the real simulation. The following methodology describes how to tell whether a particle intersects the light cone during a time step and how to estimate the exact position of intersection via interpolation. We will pay special attention to periodic boundaries and the various pathologies they introduce and discuss implications for LC stacking.

The time step starts at time t_0. Comoving particle positions $\mathbf{x}$ are defined at t_0 in $[-B/2, +B/2)$[1] and canonical velocities $\mathbf{v}$ at $t_{-1/2}$. Before the LC processing step, accelerations are computed and velocities kicked to $t_{+1/2}$, which is the correct leapfrog velocity to use when drifting the particle to t_1. The LC starts the step at comoving distance R_0 from the observer and ends at distance R_1. In ABACUS, we can compute these as

$$R_{0,1} = (\eta_K(z = 0) - \eta_K(z = z_{0,1}))\frac{c}{H_0}, \tag{1}$$

which will give the radius in comoving Mpc. We can also use $H_0 = 100$ km/s to get the radius in our preferred h^{-1}Mpc units. $\eta_K(z)$ is the conformal time from $t = 0$ to t in our standard time units of $1/H_0$:

$$\eta_K(z) = \int_0^t \frac{dt}{a} \tag{2}$$

and is already computed by ABACUS for time stepping.

[1] We actually store the particle positions in unit-box, cell-centered format. But we'll pretend they're stored as global positions with box size B for clarity.

3 Algorithm

The basic idea is to check if a particle is inside the light cone at the beginning of the time step and outside at the end. If so, the particle must have crossed the LC, so we know to interpolate.

Note that this is a tag-less scheme. To avoid seeing a particle a two different, well-separated redshifts one does indeed need tags, but we don't try to avoid this in our scheme. In fact, we want to see both intersections, since it makes box boundaries "clean" (see §6.3).

At the time of the LC output, we already have the acceleration of every particle, but nothing has been moved. Thus we can exactly compute where the particle will end up at the next time step with no approximation.

3.1 Cell opening

LC membership is defined by the interpolated position of a particle, but we want to avoid doing the interpolation if we know a cell is too far from the LC. The LC sweeps from outer radius R_0 to inner radius R_1. If a cell is entirely outside of R_0, meaning

$$|\mathbf{x}_{cc} - \mathbf{x}_{obs}| - \frac{\sqrt{3}C}{2} > R_0, \tag{3}$$

then no particles can be in the LC because particles can never "catch up" to the light cone, which moves at the speed of light. Here, $\mathbf{x}_{cc}$ is the cell center, $\mathbf{x}_{obs}$ is the LC observer, $C = B/\texttt{CPD}$ is the cell edge length (B is the box size), and the $\sqrt{3}$ factor gives the distance to the corner of the cell. Particle are guaranteed to be fully contained in their cells because nothing has been drifted yet.

Note that the distance to the LC observer is always computed without a periodic wrap. The observer's position should be in global units and may be outside the primary periodic wrap. This is so we can place an observer very far away from a box to observe the box at an early time.

If a cell is in front of the light cone, it still may contain particles that will move into the LC before the step is complete. Using the fact that ABACUS has a "Courant" condition in which particles cannot drift more than one cell, we can limit this possible movement and exclude cells with

$$|\mathbf{x}_{cc} - \mathbf{x}_{obs}| + \frac{\sqrt{3}C}{2} + C < R_1. \tag{4}$$

We can combine the two preceding conditions and accept cells with

$$R_1 - \left(1 + \frac{\sqrt{3}}{2}\right) C \le |\mathbf{x}_{cc} - \mathbf{x}_{obs}| \le R_0 + \frac{\sqrt{3}}{2} C. \tag{5}$$

Above we claimed that we don't want to do a periodic wrap of the distance to the observer. But if the LC sweep straddles the box boundary, we need to include cells on the other side of the box, since particles there may interpolate across the wrap before the LC sweeps by. We don't want a true periodic wrap, which would cause the LC to open cells unnecessarily in all periodic replicas, or even a single wrap, which would still result in many false positives. Again using our Courant condition, we can see that we only need to include a single layer of cells surrounding the primary periodic zone. Imagine the box, with its $\texttt{CPD}^3$ cells, padded with one extra layer of cells (that come from the other side of the box). It's probably best to "move" these cells by a box length and then evaluate them exactly as normal cells; this will simplify the interpolation logic because no further periodic wrapping will be needed.

Note that some cells will be processed twice as a result — once on each side of the wrap. This is fine. Particles will only be included if they intersect the light cone while in the primary periodic zone $[-B/2, B/2)$.

Here's how we should construct this external layer of cells. Face cells (cells where at least one index is 0 or $\texttt{CPD} - 1$) should be displaced by $\pm B$ in each applicable dimension and processed as normal, probably immediately after processing it in its original location. Edge and corner cells (two or three indices 0 or $\texttt{CPD} - 1$) should be processed as face cells (displaced in one dimension at a time) then displaced in two or three dimensions at a time. We can unify these cases by taking all variable-length combinations of indices that are at the box boundary.

3.2 Interpolation

For each particle in an opened cell, we find the time of intersection $t_0 + \Delta t$ with the LC:

$$x_r + v_r \Delta t = R_0 + c\Delta t \tag{6}$$

$$\Delta t = \frac{x_r - R_0}{c - v_r} \tag{7}$$

where x_r and v_r are the position and velocity projections onto the LoS ($\hat{\mathbf{x}}$) of the observer. c should be converted from km/s to canonical units to be in the same units as $\mathbf{v}$; Δt will then be in the same units as the drift factor η_D. The location of the intersection is then

$$\mathbf{x}' = \mathbf{x} + \alpha \Delta\eta_D \mathbf{v} \tag{8}$$

$$\alpha = \frac{\Delta t}{\Delta \eta_D}, \tag{9}$$

where we have introduced the interpolation fraction α which we will reuse for the velocities.

To be recorded, the intersection must be within the light cone and the primary periodic zone:

$$R_0 \leq x_r' < R_1; \qquad\qquad \mathbf{x}_i' \in [-B/2, B/2). \tag{10}$$

An equivalent form of the first condition is

$$0 \leq \alpha < 1, \tag{11}$$

which is faster to evaluate than computing x_r.

The velocity above was defined at $t_{1/2}$. However, we want to output velocities that are synchronous with the positions; that is, somewhere between t_0 and t_1. But in order to kick from $t_{1/2}$ to t_1, we technically need $\mathbf{a}_1$, which we won't have until the next time step. For now, we will ignore this error and use $\mathbf{a}_0$. This looks like

$$\mathbf{v}' = \mathbf{v} + (\alpha - 0.5)\Delta\eta_K \mathbf{a}. \tag{12}$$

Now $\mathbf{v}'$ is synchronous with $\mathbf{x}'$ and both are ready to be outputted.

We could imagine a scheme in which we detect LC intersections between $t_{-1/2}$ and $t_{1/2}$ instead of t_0 and t_1; we should have enough information to interpolate correctly both positions and velocities in this range. However, the interpolation/intersection detection would become more complicated (i.e. piecewise), and we don't have R_{-1} on-hand. The above scheme is probably accurate enough for our purposes. In the microstepping version of the code where the global steps may be large even at late times, we should consider what error this implies.

Note that none of the above quantities need periodic wrapping. This is dealt with when we move the boundary cells.

We could also imagine doing kinematic, rather than linear, interpolation on the position for slightly more accuracy. Since the inclusion of a particle in the LC is based on its final position, the method of interpolation we choose has (almost) no bearing on a particle's inclusion[2]. The velocity interpolation would remain unchanged (but see the above comments on the approximation in our velocity interpolation).

Similarly, there are probably more accurate interpolation schemes for the LC position (Eq. 2), like logarithmic interpolation.

3.2.1 Old interpolation scheme

The current implementation of interpolation in ABACUS looks different than the above:

$$\mathbf{x}' = \mathbf{x} + (\Delta\eta_D \Delta r)\mathbf{v}, \tag{13}$$

$$\Delta r = \left(\frac{x_r - R_0}{R_0 - R_1}\right) \frac{1}{1 + v_r/c}. \tag{14}$$

It's not immediately obvious where this comes from or that it's equivalent to the new formulation.

[2] See §6.1 for the one exception.

4 Output

The LC outputs are at least the size of a time slice, so we definitely want to use pack14. Particles may leave their cells during interpolation, however, which is illegal in pack14[3]. One solution is to create singlet cells — cells containing only one particle — whenever a stray particle is found. The pack14 header is 14 bytes, so it doubles the storage cost but only for a very small fraction of particles. We could also imagine insert-list-like schemes where stray particles are pushed onto a list then grouped into cells only at the end of a slab. That would save some space. The on-the-fly group finding code may have some helpful data structures for this task. A full, cross-slab insert list would be the optimal space-saving scheme, but probably not worth the trouble.

Particles may also exceed the maximum cell velocity during interpolation. We could create singlet cells for these particles, too, or compute the maximum possible velocity by considering the interpolated velocity of every particle at t_0 and t_1 and taking the max. That would require two passes through the particles in each cell, though. A quicker way would be to take the maximum velocity and kick it by the maximum acceleration for the maximum time (width of the time step). We'd overshoot, but it'd be guaranteed safe and fast.

`AppendArena` takes cell-centered positions (unit-box) and canonical velocities. In a few places we probably have to convert back and forth between global and cell-centered positions. For those intermediate variables we may wish to use `double`s, since pack14 is more accurate than `float`.

5 Stacking

We may wish to use multiple observers spaced by a box length to create a long, continuous light cone. This new scheme provides "clean" box boundaries that can be trusted to the edge, meaning we don't have to overlap boxes to get an uninterrupted LC. There shouldn't be any problems with particle doubling near the boundaries, since the observers are set up such that the LC surfaces mesh exactly, so the interpolation solution will be the same for both observers.

6 Potential Pathologies

The following discusses a few potential scenarios for trouble and why this scheme is not susceptible to them.

6.1 Dropped particles

A particle moving against the LC cannot skip over it — that is, start the time step ahead of the LC sweep and end it behind. To do so, a particle would have to drift by more than the width of the LC. But the LC moves at the speed of light, so particles always move less than the sweep width.

The only case where we can "drop" a particle is when it starts inside the LC but outside the box, and enters the box but misses the primary periodic zone due to interpolation inaccuracy. The definition of the correct behavior is blurry here, since it's hard to know what the continuous version of the system would have done (indeed this is ill-defined). So, in some sense, this isn't even an error, especially if we're using kinematic interpolation.

Note that this type of "error" can only occur relative to the box boundaries and never the LC boundaries.

6.2 Double-counted particles

A particle moving in the same direction as the LC might appear behind the leading edge then in front of the trailing edge in two consecutive time steps. This can never cause double-counting, though. If an intersection occurs in the first time step, the particle cannot end up in front of the LC for the next time step (because the LC moves faster than the particle, and they were in the same location at the intersection). Thus, the particle will only ever be considered once (at that redshift).

[3] Small excursions are allowed by pack14, but in our early tests, the LC interpolation violated this buffer zone.

6.3 Dirty boundaries

The previous tag-based light cone implementation would have missing or double-counted particles near the edges. This proposed implementation should have none of those problems. If a particle's interpolated LC intersection crosses a periodic wrap, it will be included with no caveats.

For example, previously a particle could cross the wrap in the step just before the leading edge of the LC arrived. In the next step, it would be on the other side of the box and missed completely, even though in the "continuous" version of the world its intersection would have been included. This doesn't happen in the new implementation because the intersection will be seen occur before the wrap and inside the light cone.

6.4 Round-off error

Since we have the actual velocities that will be used to drift the positions, we can construct the final particle positions that are bitwise identical to what will happen in the drift. This should alleviate most floating point concerns. Appropriately using this information may require an explicitly constructing the initial and final particle positions and checking against the LC boundaries, instead of just computing the α ratio.

There may be a number of other cases to worry about with regards to cell centering. Specifically, since particles are stored as offsets relative to cell centers, the final position we compute for a particle may not be exactly what happens in the drift and rebin step. Thus, the next timestep may see the particle at a slightly different position than the LC code expected it to.

Similar issues may appear with the layer of boundary cells.

I think this could get us into trouble. We could pretend to rebin the particle, but I'm not sure we want to go down the rabbit hole of chasing every possible floating point difference. This will require some more thought and may push us back to a tag-based scheme. But then tags will make the boundaries dirty again... ugh.

7 To-do

This scheme has yet to be implemented and tested. Also, we haven't considered the microstepped code yet. The basic idea is that we'll want to output halos as one unit, rather than considering their particles independently. This is because the high accelerations within a halo and large global timesteps will make the linear interpolation used here inaccurate, causing "shearing" of cohesive structures in the LC output.