RAY-RAMSES: a code for ray tracing on the fly in N-body simulations

Alexandre Barreira,[1,2,3,*] Claudio Llinares,[3] Sownak Bose,[3] and Baojiu Li[3]

[1] *Institute for Computational Cosmology, Department of Physics, Durham University, Durham DH1 3LE, U.K.*
[2] *Institute for Particle Physics Phenomenology, Department of Physics, Durham University, Durham DH1 3LE, U.K.*
[3] *Max-Planck-Institut für Astrophysik, Karl-Schwarzschild-Str. 1, 85748 Garching, Germany*

We present a ray tracing code to compute integrated cosmological observables on the fly in AMR N-body simulations. Unlike conventional ray tracing techniques, our code takes full advantage of the time and spatial resolution attained by the N-body simulation by computing the integrals along the line of sight on a cell-by-cell basis through the AMR simulation grid. Moroever, since it runs on the fly in the N-body run, our code can produce maps of the desired observables without storing large (or any) amounts of data for post-processing. We implemented our routines in the RAMSES N-body code and tested the implementation using an example of weak lensing simulation. We analyse basic statistics of lensing convergence maps and find good agreement with semi-analytical methods. The ray tracing methodology presented here can be used in several cosmological analysis such as Sunyaev-Zel'dovich and integrated Sachs-Wolfe effect studies as well as modified gravity. Our code can also be used in cross-checks of the more conventional methods, which can be important in tests of theory systematics in preparation for upcoming large scale structure surveys.

I. INTRODUCTION

Observations of large scale structure in the Universe have been playing a crucial role in getting ever tighter constraints on competing theoretical cosmological models. Perhaps the most classical type of such observations consists in mapping the three-dimensional distribution of galaxies (which trace with some bias the total matter distribution) with spectroscopic surveys [1–3]. These surveys measure the baryon acoustic oscillations (BAO) signal [4, 5] and clustering anisotropies due to redshift space distortions (RSD) [6–8], which allow to put constraints on the rate at which the Universe expands and the rate at which structure grows in it. A complementary approach to traditional galaxy surveys (and which is the focus of this paper) is to infer the large scale structure of matter by measuring its integrated effect on light that travels from background sources towards us. These include shifts in the temperature of cosmic microwave background (CMB) photons caused by inverse Compton scattering with high energy electrons inside galaxy clusters – the so-called Sunyaev-Zel'dovich (SZ) effect [9–12], shifts in the temperature of CMB photons as they cross time-evolving gravitational potentials – the so-called integrated Sachs-Wolfe (ISW) effect [13–16], and magnification and distortions of background light sources as their emitted light bends due to strong and weak gravitational lensing effects [17–20].

As observational surveys of large scale structure keep attaining higher precision, it is important that our theoretical understanding of the relevant physical processes keeps evolving as well. This helps in the interpretation of current data, as well as in the planning of future missions. In computing these theoretical predictions, theorists typically recourse to N-body simulation methods since these are currently the most accurate way to predict the clustering of matter on intermediate and small scales, where the density fluctuations have become nonlinear. N-body simulations allow also to include recipes to model the effects of baryonic physics and can be used in the generation of mock data sets to calibrate observational pipelines. Using N-body simulations to predict integrated effects along lines of sight that cover a redshift range is, in general, not as straightforward as getting predictions at fixed redshift values. For the latter, it often suffices to analyse the simulation output at a given snapshot of the particle distribution, whereas for the former it is required that the simulation results are analysed in a continuous range of redshift values. As a result of that, N-body methods for integrated observables are often subject to a number of approximations that are more or less valid depending on the exact observable studied. For instance, one of the most popular methods for cosmological weak lensing simulations consists of projecting the continuous matter distribution along the line of sight into a series of lens planes [21–34]. This naturally erases the details of the time evolution of the fields along the line of sight. Furthermore, these projections assume that the superposition principle holds for the lensing effects of gravity, which is not necessarily true in theories beyond General Relativity that have nonlinear equations. Simulations of the ISW effect also make certain simplifying assumptions in the calculation of the time derivative of the gravitational potentials (see e.g. [35]).

A great deal of effort is normally put into assessing the validity of the approximations made in these numerical methods, and in general, they seem to be robust enough. However, given the ever higher precision observations that lie ahead, it is desirable that the same observables can be computed with different methods, especially those which are subject to fewer approximations. This can allow for important checks of any residual theory systematics that could still be present. Moreover, current N-body methods to probe the clustering of matter along the line of sight sometimes require substantial amounts of data to be stored before it is post-processed to compute the desired signal. This provides extra motivation to develop new numerical techniques that are lighter in data storage, especially in light of the upcoming generation of surveys, which will require large simulations and mock data sets for calibration purposes.

This paper is precisely about a numerical method for in-

*Electronic address: barreira@mpa-garching.mpg.de

tegrated cosmological observables that goes beyond existing techniques in the number of approximations made and data storage concerns. Our method, which is based on the original idea of Refs. [36, 37], is designed to trace rays from some source redshift to the observer, on the fly in the N-body simulation. Our algorithm is implemented in the adaptive mesh refinement (AMR) RAMSES N-body code [38] and performs the integrations along the line of sight on a cell-by-cell basis, fully exploiting the spatial and time resolution provided by the N-body code. Moreover, since it runs on the fly in the N-body simulation, it can produce the desired maps (lensing, ISW, etc) without having to output the particle snapshots for post-processing analysis. The goal of this paper is to introduce the code implementation and illustrate its application in cosmological weak lensing simulations.

The outline of this paper is as follows. We start in Sec. II by describing in more detail the reasons that motivated us to develop the ray tracing code presented in this paper. Section III explains the main aspects of the implementation of the algorithm in the RAMSES code. In Sec. IV, we explain the formalism to perform weak lensing studies with our code and test our implementation in Sec. V for a fixed Gaussian potential. Section VI is devoted to weak lensing cosmological simulations. We analyse our code results for one- and two-point statistics of the lensing convergence, where we assess the impact of N-body resolution and different integration methods. We also investigate the lensing signal around dark matter haloes in our simulations. Finally, we summarise and analyse the prospects for future developments and work in Sec. VII.

II. MOTIVATING A NEW RAY TRACING CODE

In general, numerical studies of integrated observables employ the following three step strategy. First, one runs a N-body simulation for a given cosmology and stores the particle data at a specified number of redshift values. Second, the output from the simulations is used to generate a mock lightcone from some observer to a given source redshift[1]. Here, one often needs to employ some interpolation scheme to construct a continuous matter distribution from the simulation results that are available only at a finite number of redshift values. Finally, rays are traced across the lightcone to probe the distribution of matter along the line of sight. This strategy has been employed most notably in weak lensing studies (see e.g. Refs. [21–34] and references therein), but also in ISW [35, 39, 40] and SZ [41–45] related work.

One can identify, however, two less appealing aspects of this strategy. The first one is practical and relates to the large amounts of data that are needed to generate lightcones for post-processing. The second is related to the loss in resolution along the line of sight that follows from analysing a lightcone that has been constructed from a finite number of snapshots. To give a concrete example, conventional weak lensing studies usually employ the so-called multiple lens-plane approximation, in which the observables are calculated only on a series of planes, onto which the density field has been projected[2]. Although one can always perform convergence tests on the number of planes used (e.g. Ref. [32]), some of the detailed information on the time evolution along the line of sight is in general lost.

Our main motivation to develop the code presented in this paper was to overcome the two above-mentioned aspects. Namely, we aimed to implement a numerical method that (i) computes the integrated observables on the fly in the simulation, thereby avoiding the need to store large amounts of data; and (ii) takes full advantage of the spatial and time resolution of the N-body run to compute the integrals along the line of sight. Our numerical implementation is based on the original idea of Ref. [36] for weak lensing simulations, which was later optimized in Ref. [37]. In particular, in the latter work, the authors realized that in particle-mesh (PM) N-body simulations, the integrated quantities can be computed analytically on a cell-by-cell basis as the simulation is running. These authors implemented their method in regular grid PM codes. In this paper, we follow a similar approach, but implement the algorithm in the publicly available RAMSES code [38], which can achieve a far greater resolution due to its AMR nature.

When designing this code, it was also our goal to make it general enough so that it could be used as a platform to perform studies of other types of integrated observables, and not just lensing. As a result, even though in this code presentation paper we illustrate the code operation for lensing, we stress that the algorithm is more general than that. In short, the code we present in this paper can calculate integrals of the form

$$I = \int K(\chi) Q(x, y, z) \mathrm{d}\chi, \tag{1}$$

where χ is the comoving distance along some ray trajectory, K is an integration kernel and Q is any field that can be determined inside the simulation box at coordinates x, y and z. The calculation of different observables corresponds to different expressions for K and Q. For instance, for thermal and kinetic SZ studies, Q is related to the density-weighted temperature and bulk velocity of electrons in clusters, respectively; for ISW studies, Q is given by the time derivative of the lensing gravitational potential; and for lensing studies, Q would be associated with second transverse derivatives of the lensing potential (cf. Sec. IV). For the case of lensing, rays may also get their trajectories bent (although we anticipate here that this is not the case in this first version of the code). As we commented above, there is already a substantial body of work available in the literature on these topics. The code we present here provides a different method to compute the

[1] There are however ways to compute the lightcone on the fly in the simulation (e.g. Refs. [23, 25]).

[2] See, however, the approach of Ref. [29], in which the lensing quantities are integrated using the three-dimensional distribution of the simulations (without projection onto planes), but which is still only available at a finite number of redshifts.

same quantities which, amongst other things, can be used in important cross-checks of the traditional methods.

We note in passing that the ray-tracing machinery that we installed in RAMSES may also serve as a starting point to develop a code that could be applied in radiative transfer studies (see e.g. Refs. [46–50] and references therein). This will, however, require some modifications to the code presented here, which is primarily oriented for integrated observables along lines of sight.

III. CODE DESCRIPTION

In this section, we describe the main parts of the ray tracing code. We start by presenting a quick review of the default RAMSES code, which is followed by an overview of the ray tracing algorithm and how it is implemented in RAMSES. We then explain with more detail each of the main parts of the ray tracing code.

A. Notation and basics of the RAMSES code

Our ray tracing modules are installed in the publicly available AMR RAMSES N-body code [38]. RAMSES employs a number of simulation particles N_p which act as discrete tracers of the underlying matter field. The simulation box is covered by a three dimensional mesh, on which the density values are calculated using the cloud-in-cell (CIC) interpolation scheme given the particle distribution at any time step. The code then solves for the gravitational potential field on the mesh, which can be finite-differenced to find the corresponding gravitational forces. The force at the particle positions is obtained by interpolating back from the mesh using the same CIC scheme to ensure momentum conservation. This is then used to update the particle's velocity and position at the next time step. The whole process is repeated from some initial time (typically redshift $z = 50 - 100$) to a later time (usually $z = 0$). The cubic cells of the 3D mesh can get refined if the effective number of particles contained in them exceeds some pre-specified threshold, $N_{\rm refine}$. Conversely, the cells are also de-refined if the number of particles drops below that threshold. This AMR nature of the code is useful in cosmological simulations, because it allows for high resolution in regions of high matter density, whilst saving computational resources in regions of lower density where the resolution can be lower.

The term "domain level" refers to the coarsest mesh that regularly covers the whole simulation volume. In RAMSES, the domain level contains $N_p^{1/3}$ cells in each direction[3]. If a cell of the domain level gets refined, then it is called a "father cell" with eight cubic "son cells". If the cell size of the father cell is h, then each of the eight sons has cell size $h/2$. The father cell, together with its son cells, form a so-called "grid" or "oct" of the refined level. If one of these eight sons gets further refined, then it will form a grid of the second refined level, i.e., its son cells will have cell size $h/4$. This series of grids accross refinement levels is organized in a tree structure. RAMSES stores the grid and cell IDs in separate arrays and in a way that (i) given a son cell's relative position inside its grid and the ID of that grid, one can find the ID of the son cell, and vice versa; and (ii) given the parent cell's ID one can find the grid ID. Each level of refinement is labelled by l. The domain level $l = l_{\rm domain}$ is defined by $2^{l_{\rm domain}} = N_p^{1/3}$. For instance, if the simulation has $N_p = 512^3$, then $l_{\rm domain} = 9$. The first refinement level is labelled by $l_{\rm domain}+1$ and so on. Another characteristic of the RAMSES code is that, at refinement boundaries, the coarse and fine sides differ only by one level of refinement. The size of the time steps is determined independently for each refinement level, with higher refinements taking smaller time steps. For example, one of the criteria to determine the size of the time steps is that the particles should move only by a fraction of the cell size they are currently in (cf. Sec. 2.4 of Ref. [38]).

RAMSES is also efficiently parallelised using MPI. When run in parallel, each grid is "owned" by the same CPU that "owns" the parent cell of the grid. "To own" here means that the CPU knows all necessary information of a cell/grid, and is responsible for calculating all the relevant quantities inside the cell (density, potential, etc.), as well as the son cell position inside the grid. As domain decomposition strategies, RAMSES can employ Peano-Hilbert, angular and planar schemes. Of these, the Peano-Hilbert space filling curve is the most optimal for standard N-body runs. However, given the angular geometry of the ray tracing operations that we aim to perform, it is beneficial to consider the angular scheme since it distributes better the rays accross CPUs (specially when the number of rays becomes large).

Our modifications to the RAMSES code are mostly in the form of additional independent numerical modules, which keep the base code unchanged (except only for a few interfaces). We refer the reader to the RAMSES code paper [38] for more details about its operation.

B. Tiled simulation boxes

High-resolution N-body simulations of boxes that are large enough to contain the distance travelled by photons from $z \gtrsim 1$ ($\chi \gtrsim 2$ Gpc$/h$) typically require massive computational resources (see e.g. Refs. [24, 25]). To circumvent this, one can "tile" together a number of simulation boxes in order to fit the whole light bundle [21, 36, 37]. Figure 1 shows an example of a possible tiling scheme. The observer lies in the box that we refer to as the "last box", as opposed to the "first box", which contains the ray sources. The source redshift is $z_s = 1$ in this example. Each simulation box takes as input the position of the observer w.r.t. its origin, $x_{\rm obs}, y_{\rm obs}, z_{\rm obs}$. For example, for the case illustrated in Fig. 1, the observer

[3] In this paper, we always assume three dimensional systems, although some times we shall use two-dimensional diagrams to facilitate the illustrations and explanations.

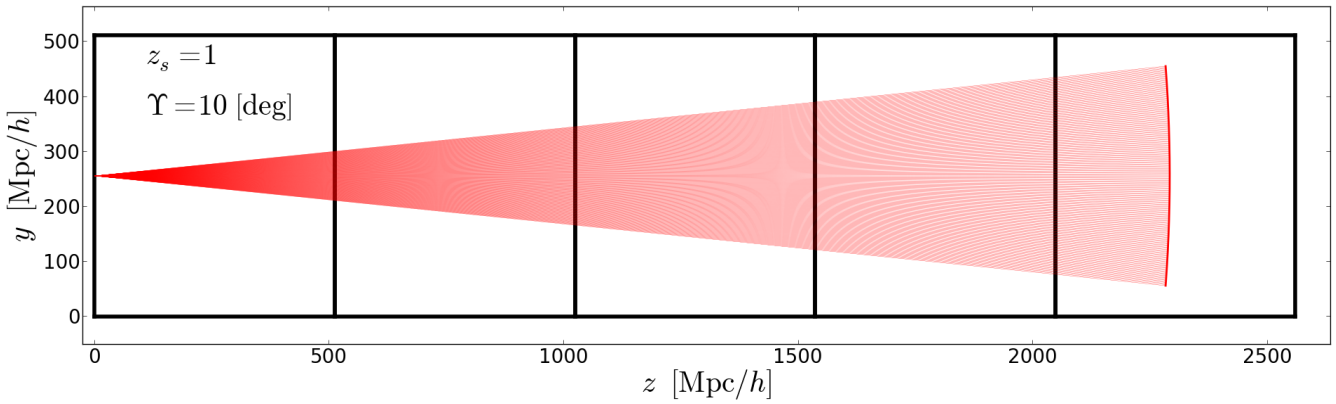

FIG. 1. Example of a tiling scheme for ray tracing. The x-axis points into the plane of the figure. The boxes have size $L = 512\ \mathrm{Mpc}/h$. The thin red lines illustrate the trajectory of the rays in a light bundle with opening angle $\Upsilon = 10$ deg from $z_s = 1$. The different boxes should also simulate different realizations of the initial density field to avoid rays crossing the same structures at different times. If the ray trajectories are straight lines, then the boxes in the tile can be simulated simultaneously. In the case of bending rays, each box can start when the previous box (higher redshift) has finished the calculations. Although it is not the case in the figure, closer to the observer where the bundle covers a smaller volume, the simulation boxes can be made smaller to increase the particle resolution.

is located at the center of x-y face ($z = 0$) of the last box[4], and so we have $x_{\rm obs} = y_{\rm obs} = 256\ \mathrm{Mpc}/h$, $z_{\rm obs} = 0$ for that box. For the first box, on the other hand, these would be $x_{\rm obs} = y_{\rm obs} = 256\ \mathrm{Mpc}/h$, $z_{\rm obs} = -2048\ \mathrm{Mpc}/h$. Given the geometry of the light bundle, the ray positions are more easily described using a spherical coordinate system with the observer at its origin

$$\begin{aligned} x_{\rm ray} &= \chi\ \sin\theta\ \cos\varphi, \\ y_{\rm ray} &= \chi\ \sin\theta\ \sin\varphi, \\ z_{\rm ray} &= \chi\ \cos\theta, \end{aligned} \tag{2}$$

where $\theta \in [0, \pi]$, $\varphi \in [0, 2\pi]$ are the two angular coordinates on the sky and χ is the radial coordinate. If the rays follow straight trajectories, then $\chi(z)$ is equal to the comoving distance $D_{\rm c}(z) = c \int_0^z \mathrm{d}z/H(z)$, with $H(z)$ being the Hubble expansion rate, z the redshift and c the speed of light.

In the tiling scheme, a ray is only traced in a given box in the redshift interval during which the ray position lies within that box. For example, the integration of the rays in the first box would start at $z = 1$ and it will last until $z \approx 0.86$, which is approximately when the rays "touch" the face of the box. Following the same reasoning, the second box would start the integrations at $z \approx 0.86$, which will continue until $z \approx 0.60$; and so on and so forth, until the rays reach the observer at $z = 0$. Naturally, rays located in the outermost regions of the light bundle move from one box to the other before the more central rays. The conditions for the start and end of integration in each of the boxes are explained with more detail in Sec. III F 3. We note also that for straight ray cases, the boxes in the tile can be run simultaneously, since they all "know" *a priori* the position of the rays at all times. On the other hand, if rays bend, then boxes located closer to the observer can start tracing the rays after reading their positions from higher-redshift boxes.

We note that each simulation box should also start from different statistical realizations of the initial density field. This way, one ensures that the rays do not see the same structures throughout their trajectories (due to the periodic conditions of the simulation box). Finally, although not depicted in Fig. 1, it is also worth mentioning that closer to the observer, where the spatial volume covered by the light bundle is smaller, the boxes in the tile can be made smaller to gain resolution without sacrificing computational efficiency (although not too small to still allow large enough structures to form).

C. Outline of the code

Figure 2 shows a sketch of the flow of calculations in the code. The first operation of the ray tracing calculation consists of the initialization of the ray data structure (cf. Sec. III D). The goal is to identify the ID of the grid that a given ray belongs to, i.e., determining the physical location of the ray within the grid structure (cf. Sec. III E). This is performed only when the rays start the integration because, as the rays move through the mesh, it is possible to determine the ID of the next crossed grid, by searching for neighbouring grids.

After the rays have been initialized, they are moved across the mesh on a cell-by-cell basis (cf. Sec. III F), integrating a given quantity along the path inside each cell. As we explain in Sec. III G, the integration can be done analytically by using the values of the desired quantity at each crossed cell centre or at its vertices. The latter have to be obtained by interpolation from the cell centres, which is where `RAMSES` evaluates all fields (density, potential, etc.) by default (cf. Sec. III H).

In a given time step, each CPU moves the rays that are currently within its spatial domain until one of the following pos-

[4] We use the same letter, z, to denote redshift and one of the cartesian coordinates. The meaning of z should be taken by the context.

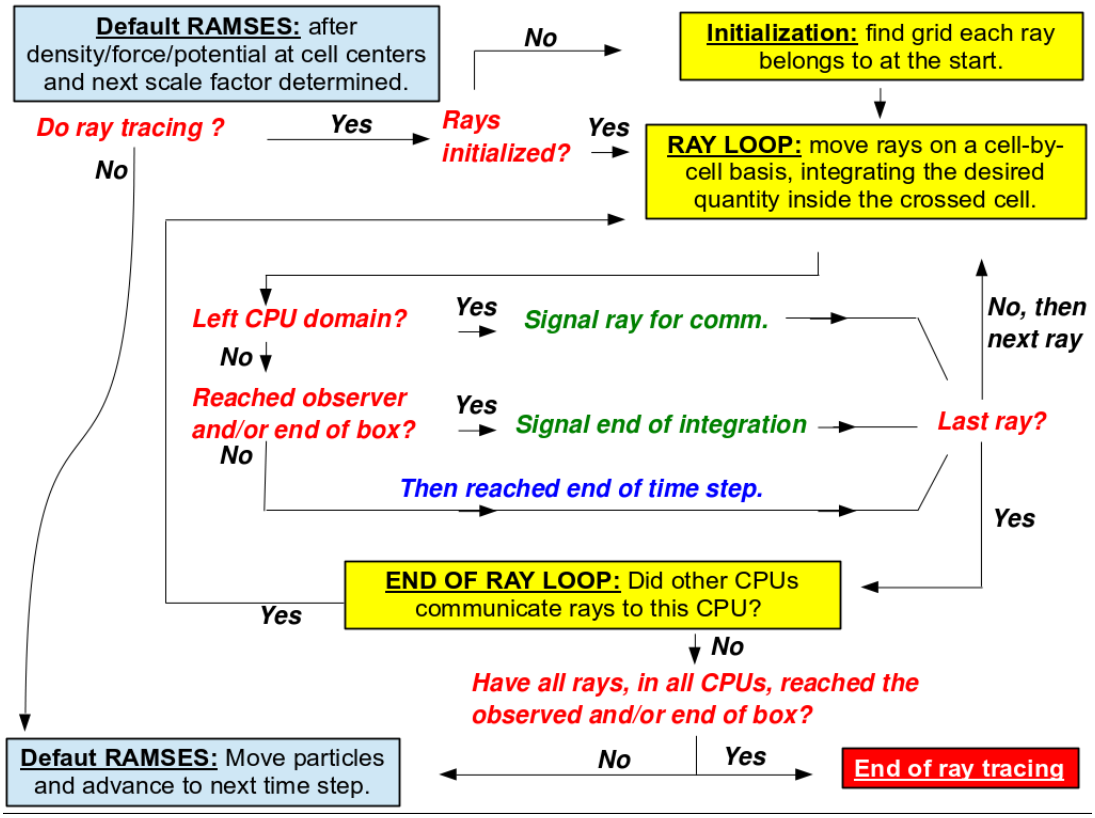

FIG. 2. Sketch of the code flow. The ray tracing routines are called after default RAMSES computes the field values and the next scale factor, but before the particles are moved. The ray tracing routines are initialized once each simulation box in the tile reaches the redshift at which the rays start their propagation there. In each time step, each CPU integrates each ray on a cell-by-cell basis (i) until it travels the maximum allowed distance light can travel, in which case the CPU goes directly to the next ray; (ii) until it reaches a CPU domain boundary, in which case the ray is marked for communication for another CPU to continue its integration; or (iii) until it reaches the observer and/or the end of the box, in which case the ray integration in the box is marked as finished for that ray.

sibilities happens:

i the rays travel the distance that light can travel in that time step;

ii the rays reach the observer/face of the box;

iii the rays reach the end of the CPU's spatial domain.

If (i) is satisfied, then the CPU simply moves on to the next ray. If (ii) happens, then the ray's integration is marked as finished, and the CPU also proceeds to the next ray. Finally, if (iii) happens, then the ray is marked for communication and the CPU still carries on to its next ray. Once each CPU has dealt with its initial number of rays, it checks whether rays from other CPUs have been marked to enter its domain, and whether its own rays have been marked to leave. If there are rays entering and/or leaving the CPU's domain, then the relevant CPUs exchange ray data via MPI communication and repeat the above calculations for the incoming rays. This process is repeated until all rays satisfy (i) or (ii).

Our ray integration routines require as input the field values given by RAMSES at a given time step, and hence, they are called after RAMSES computes these quantities. Once the ray tracing calculations for this time step are finished, the code proceeds with the standard N-body part, until it is time to call the ray tracing routines again at the next time step. Our modifications consisted therefore in the development of independent modules that do not impact in any way the standard N-body part.

In the remainder of this section, we explain in more detail each of the steps and concepts involved in the propagation and integration of the rays across the mesh. The reader who wishes to skip these details can jump to Sec. IV, from whereon we present tests and results from weak lensing ray tracing simulations.

D. Ray data structure

To implement our ray tracing algorithm in RAMSES we need to establish a data structure that links ray and grid IDs

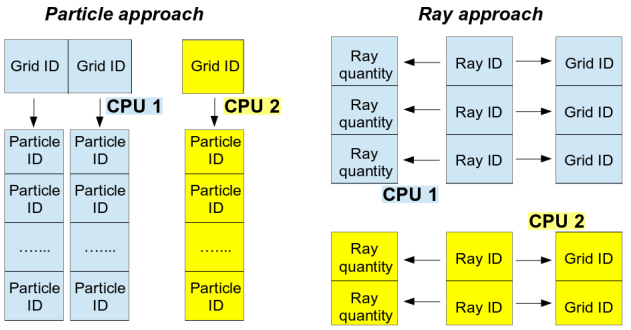

FIG. 3. Sketch of two possible data structure schemes to link global ray IDs with local grid IDs. The *particle approach* treats rays as a different particle type in RAMSES associating each grid with all the rays contained it. This approach, which is based on RAMSES's linked lists, is, however, computationally expensive because ray particles travel at the speed of light, which requires the linked lists to be update too many times. In the *ray approach*, two ordered lists link each ray to the grid it is currently in. If a ray leaves its current grid, then all there is to do is to update the entry of the grid list that corresponds to that ray.

[5]. One can think of at least two ways to do so. We call one the "particle approach" and the other the "ray approach". These two approaches are sketched in Fig. 3.

In crude terms, the particle approach determines *which rays are in each grid* (cf. left-hand side of Fig. 3). The advantage of it is that it enables direct use of the existing RAMSES structure for other types of "particles" (dark matter, stars, sinks, etc.), therefore making the coding easier. In this approach, a *linked list* data structure is used in RAMSES to find the global IDs of the particles that lie within a grid, given the grid ID. The communication of rays between CPUs would also follow the strategy already set up for other types of particles in RAMSES. However, in the code, one criterion for determining the size of the time step ensures that dark matter particles move only by a fraction of the current grid size. During this particle time step, photons, which travel at the speed of light, can cross many grids. This means that one has to either update the linked lists each time rays change grids, which involves a large numbers of operations and memory allocations/deallocations, or drastically reduce the particle time step so that rays do not cross more than one grid in a time step, which would make the code prohibitively slow.

The above drawbacks motivated us to implement the ray approach, in which one determines *which grid a ray is in* (cf. right-hand side part of Fig. 3). In this case, the data structure consists of two ordered lists of global ray IDs and their corresponding local grid IDs. Ordered here is in the sense that the global ray ID that is in the n-th entry of the ray list physically resides inside the grid whose local grid ID is that of the n-th entry of the grid list. Note that, naturally, if a ray is inside a refined grid, then it also physically lies inside its parent grid. In the data structure, rays are always linked to the grid that is at the highest refinement level. With the ray approach, the integrated quantities are associated to each ray ID in the same way as grid IDs. For example, if the ray in the n-th entry of the ray list crosses one cell to enter another, then all the code needs to do is to (i) compute the integral for the crossed cell and accumulate it in the n-th entry of the quantity's list; and (ii) update the n-th entry of the grid list with the grid ID that contains the new cell (which can be the same grid). The ray approach is more efficient than the particle approach, even though it involved developing new strategies to move rays and communicate between CPUs (when rays move outside a CPU's spatial domain or when RAMSES does *load balancing* to reassign domains to CPUs).

E. Ray initialization

The goal of the ray initialization is to set up the data structure described in the previous subsection. Just to guide the discussion in this section, we assume that the central ray of the bundle travels in the negative z direction. The light bundle is specified by (i) its opening angle in the x and y directions, Υ_x, Υ_y, respectively; and (ii) the number of rays in each of these two directions, N_x and N_y. The angular positions of the rays in the field of view are assigned as

$$\theta_a = \frac{\Upsilon_a}{N_a - 1}(i_a - 1) - \frac{\Upsilon_a}{2}, \qquad a = x, y, \tag{3}$$

and we define the global ray ID number as[6]

$$\texttt{rayid} = (i_y - 1)N_x + i_x, \tag{4}$$

with $i_x \in [1, N_x]$ and $i_y \in [1, N_y]$. Once initialized, the global ray ID stays the same throughout the simulation, even when rays change CPUs. At the end of the simulation, given a ray ID, one can reverse the above equations to find the ray's position in the 2D ray tracing map.

In the RAMSES structure, it is straightforward to retrieve the spatial location of a grid given its ID, but not the inverse operation: to find the grid ID given a certain (the ray's) spatial location. To achieve this, one can loop over all grids, and for each grid, loop over all rays to identify those that are inside it. We note that such a "brute force" search may not lead to huge overheads to the overall performance of the code since the initialization is performed only once per box. Nevertheless, we have developed a more efficient algorithm that loops over all grids, but for each, only loops over a smaller targeted range of ray IDs. The details of the algorithm are given in Appendix A, but in short, the idea is to compute the solid angle that subtends a sphere that contains the grid, with which one can determine a range of θ_x and θ_y. This then enables us to

[5] Once the link between grid and ray IDs is set, then the link between the ray and the relevant cell is made by checking in which octant of the grid the ray is in.

[6] There is an abuse of notation with the subscripts $_x$ and $_y$. Here, they denote the two directions perpendicular to the line of sight and should not be confused with the 3D Cartesian coordinates.

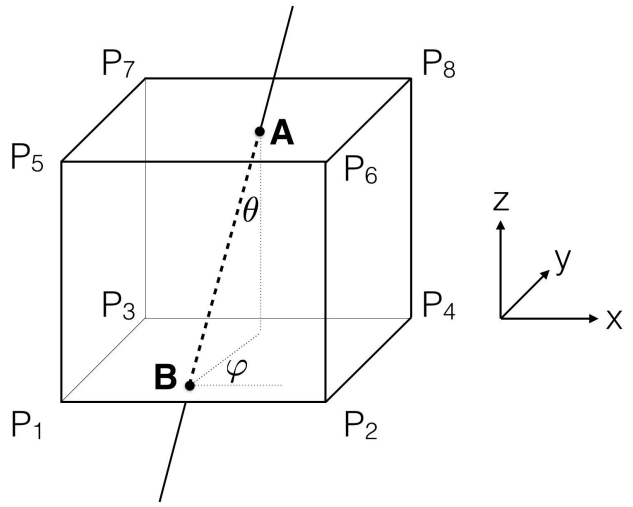

FIG. 4. Example of a ray trajectory in a cell. Points A and B are, respectively, the ending and starting points of the trajectory. In this illustration, these two points are the intersection points of the ray trajectory and the cell faces, but in general points A and B can lie anywhere inside the cell.

significantly narrow down the ray IDs to loop for each grid using Eqs. (3) and (4).

For non-first boxes in the tile, the rays are initialized at the box face, and as a result, if the spatial location of the grid (which can be determined straightforwardly in `RAMSES`) does not lie on the box face, then the loop over rays can be skipped. Note also that since rays are linked always to the finest grid (cf. Sec. III D), one can also skip the loop over grids that are not at the highest refinement level.

As we commented above already, the initialization needs only to be performed once per box. During the course of the ray integrations, when a ray leaves a grid to enter another, there are other ways to efficiently determine the IDs of the new grid.

In this paper, we limit ourselves to analysing the code results for small opening angles, for which the flat-sky approximation holds. For full-sky ray tracing, one may benefit from using techniques such as `HEALPix` [51] to describe the ray distribution across the sphere. We leave the generalization to full-sky cases for future work.

F. Moving rays

In this section, we describe how the algorithm determines the path of a ray in a cell and how the rays are moved through the mesh across particle time steps and CPU domains.

1. *Trajectory inside cells*

Given the direction of a ray inside a cell, the calculation of its trajectory and the determination of the face from which the ray leaves the cell (which determines the next crossed cell) is mostly a geometrical exercise. Denoting by θ and φ the two spherical coordinate angles that specify the direction of the ray, then the position of the ray can be parametrized by λ in the equation [7]

$$\vec{r} = \big(\chi_A + \lambda\big)\big(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta\big), \tag{5}$$

where χ_A is the comoving distance of the ray to the observer at the beginning of the trajectory in the cell (cf. Fig. 4). Hence,

$$\vec{r}_A = \chi_A\big(\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta\big) \tag{6}$$

is the position vector of the ray at the beginning of its trajectory. Of all of the six cell faces the ray can cross, three can be ruled out by the signs of $\cos\theta$, $\sin\varphi$ and $\cos\varphi$. For instance, if $\cos\theta > 0$, then the ray is travelling in the negative z direction ($\theta \in [0, \pi/2]$). This means that the ray cannot enter the neighbouring cell that lies in the positive z direction. Similarly, if $\sin\varphi > 0$ ($\cos\varphi > 0$), then the ray cannot enter the neighbouring cell that lies in the positive y (x) direction. More compactly, the faces from which the ray can leave the cell lie on one of three planes, each characterized by

$$\begin{aligned} x &= x_{\rm target} = x_{\rm cell} - {\rm sign}(\cos\varphi)h/2, \\ y &= y_{\rm target} = y_{\rm cell} - {\rm sign}(\sin\varphi)h/2, \\ z &= z_{\rm target} = z_{\rm cell} - {\rm sign}(\cos\theta)h/2, \end{aligned} \tag{7}$$

where h is the cell size and $x_{\rm cell}$, $y_{\rm cell}$ and $z_{\rm cell}$ are the cell center coordinates. The trajectory of the ray, Eq. (5), intersects each of these three planes, respectively, at

$$\begin{aligned} \vec{r}_{B,x} &= x_{\rm target}\left(1, \frac{\sin\varphi}{\cos\varphi}, \frac{\cos\theta}{\sin\theta\cos\varphi}\right) \\ \vec{r}_{B,y} &= y_{\rm target}\left(\frac{\cos\varphi}{\sin\varphi}, 1, \frac{\cos\theta}{\sin\theta\sin\varphi}\right) \\ \vec{r}_{B,z} &= z_{\rm target}\left(\frac{\sin\theta\cos\varphi}{\cos\theta}, \frac{\sin\theta\sin\varphi}{\cos\theta}, 1\right), \end{aligned} \tag{8}$$

where the subscript $_B$ denotes the end of the ray trajectory inside the cell (cf. Fig. 4). The face from which the ray leaves the cell is that to which the ray needs to travel the least. Therefore, one computes

$$\begin{aligned} D_x &= \|\vec{r}_A - \vec{r}_{B,x}\|^2 = \chi_A - \frac{x_{\rm target}}{\sin\theta\cos\varphi}, \\ D_y &= \|\vec{r}_A - \vec{r}_{B,y}\|^2 = \chi_A - \frac{y_{\rm target}}{\sin\theta\sin\varphi}, \\ D_z &= \|\vec{r}_A - \vec{r}_{B,z}\|^2 = \chi_A - \frac{z_{\rm target}}{\cos\theta}, \end{aligned} \tag{9}$$

and the value of $\min\{D_x, D_y, D_z\}$ specifies the target face. For example, if $\min\{D_x, D_y, D_z\} = D_x$, then the ray leaves from the face $x = x_{\rm target}$ (and analogously for y and z).

Once the exiting face is found, then one can identify the next crossed cell by searching for neighbours using the default

[7] For straight rays, θ and φ coincide with the angles that specify the spherical coordinates of the ray (cf. Eq. (2)). We leave the generalization of our algorithm to follow non-straight trajectories for future work.

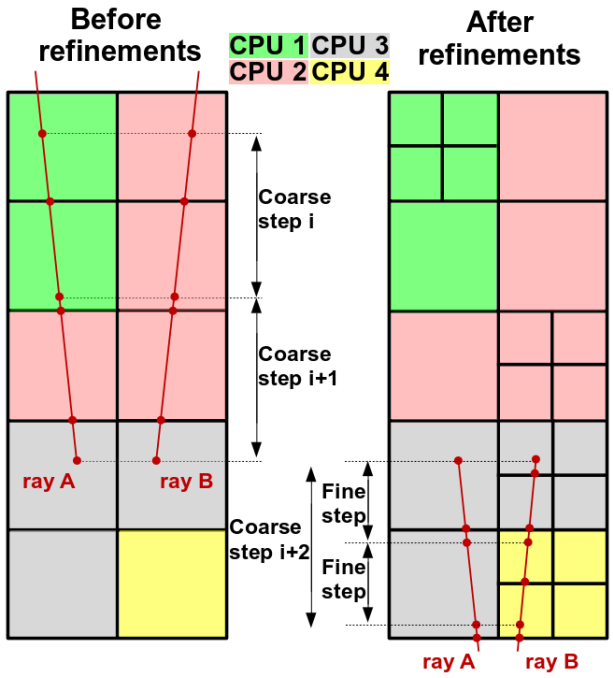

FIG. 5. Two-dimensional sketch of two rays moving through the mesh. The colors indicate different CPU domains and the red dots denote the starting and ending points of a ray inside a cell. The rays move from top to bottom in the figure. There are three coarse time steps shown, and the mesh gets refined from the second to the third. Note that the distance that rays travel in a time step becomes smaller when there are refinements.

data structure in RAMSES. If the new cell belongs to the same grid, then one needs only to update the cell information. If on the other hand the next crossed cell belongs to a different grid, then one updates the grid information as well. Note that the new grid can lie at a different refinement level and/or in a different CPU domain, and this information is also recorded in our ray data structure.

2. *From cell to cell across different time steps and CPU domains*

Figure 5 shows a two-dimensional sketch of rays moving through the mesh. Shown are the trajectories of two rays during three coarse time steps. During the first two there are no refinements (left), and in the last coarse step some cells get refined (right). The different colors indicate different CPU domains. The red dots along the ray trajectories indicate where the rays cross cells or reach the end of a time step.

In the first coarse time step, ray A starts from the middle of a cell and is moved to the face of this cell. During this trajectory, the code integrates the desired quantity using the algorithm that shall be described in Sec. III G. Ray A then continues its trajectory in the next crossed cell, but stops before reaching the new cell's face, at the point where the ray has travelled the maximally allowed distance in that time step. In the meantime, Ray B is moved similarly to ray A, but in the domain of CPU 2.

In the second coarse time step, CPU 1 moves ray A to the face of the cell and marks it for communication to CPU 2, which is found to own the next crossed cell (CPU 1 would then continue to move the other rays in its domain, if any). In the meantime, CPU 2 moves ray B until it is marked for communication to CPU 3. After CPU 2 has dealt with ray B (and all the other rays in its domain), it checks whether other CPUs have marked rays to be sent to it and moves these incoming rays as described above. Similarly, CPU 3 also checks for incoming rays. In the case sketched in Fig.5, CPU 2 receives ray A and integrates it until it is marked for communication to CPU 3, and CPU 3 receives ray B and integrates it until the end of the time step. These series of CPU communications occur until all rays have reached the end of the time step [8]. In our particular example, CPU 3 receives ray A and integrates it until the end of the time step.

The way that rays move in the third coarse step is analogous to that of the other two, except that the distance that the rays travel before RAMSES updates the field values is smaller, because of the smaller particle time step. In the current implementation of the code, all rays travel by the distance determined by the particle time step on the finest level, regardless of which level they belong to. In principle, this is not necessary since, if a ray only crosses unrefined cells, then it can be moved by the (larger) distance allowed in the coarse step, if the field values at the corresponding cell are not updated during the finer steps taken by the code. The implementation of this in RAMSES is, however, slightly more involved and therefore we opted to have all rays moving by the distance determined by the finest cells of the mesh.

3. *From box to box in the tile*

As the bundle approaches the face of non-last boxes, some of the rays will cross the face earlier than other rays. For instance, the outermost rays in the bundle are the first to reach the box face, whereas the center rays are the last. To exemplify how the code deals with this transition, consider the trajectory of the outermost and center rays as they leave a given box (call it Box 2) to enter another (call it Box 1), and denote by $z_{\rm end,2}^{\rm outermost}$ and $z_{\rm end,2}^{\rm center}$ the redshift values at which the outermost and center rays cross the face of the box. In this case, Box 2 propagates the outermost ray until it reaches its face. At this point, the integration for this ray stops, but the calculation for the center ray is still ongoing. The simulation of Box 2 should therefore only be stopped for $z \leq z_{\rm end,2}^{\rm center}$. As for Box 1, it initializes the ray data structure at $z = z_{\rm end,2}^{\rm outermost}$ for all rays, including the center one whose position is known beforehand [9]. However, it starts by integrating only the out-

[8] We note in passing that, in addition to the communications that occur when rays leave CPU domains, there are other situations that require communicating the ray information, namely when the code performs load balancing (redistribution of the spatial domains across CPUs).

[9] As mentioned previously, the geometry is fixed for straight ray simulations and therefore Box 1 "knows" *a priori* the position of all rays at any redshift.

ermost ray. The center ray remains "initialized" at the box face, and it only starts being integrated when Box 1 reaches $z = z_{\rm end,2}^{\rm center} < z_{\rm end,2}^{\rm outermost}$.

G. Integration in cells

One of the key parts of our algorithm is the computation of the integral of a given field along the ray trajectories. Explicitly, we wish to evaluate integrals of the form of Eq. (1), which we repeat here for easy reference,

$$I = \int_0^{\chi_s} K(\chi) Q(x,y,z) \mathrm{d}\chi, \tag{10}$$

where $\chi_s = \chi(z_s)$ (with z_s the source redshift), $K(\chi)$ is some weighting function or kernel that can be a polynomial of χ and $Q(x,y,z)$ is any quantity that can be evaluated in the cells. Compared with Eq. (1), Eq. (10) is more specific as it encodes the information that we wish to follow rays from some distant source to an observer at redshift zero. The integral of Eq. (10) can be split into the contribution from each cell crossed by a ray

$$I = \sum_{\rm cells} I_c, \tag{11}$$

in which

$$I_c = \int_{\chi_B}^{\chi_A} K(\chi) Q(x,y,z) \mathrm{d}\chi, \tag{12}$$

with χ_A and χ_B being, respectively, the radial coordinate of the starting and ending points of the ray trajectory in a cell as the ray travels towards the observer ($\chi_A > \chi_B$) [10] (see Fig. 4). To perform the integral we need to be able to evaluate the quantity Q at any given point inside the cells. The simplest way to do so is to take the field to be constant inside the cell and equal to the value at its center Q_c, as given by default `RAMSES`. In this case, Q can be taken out of the integral in Eq. (12) and the contribution of each cell to the total integral becomes

$$I_c^{\rm NGP} = Q_c \int_{\chi_B}^{\chi_A} K(\chi) \mathrm{d}\chi, \tag{13}$$

where the superscript NGP stands for nearest grid point. We note that, in this way, the fields being integrated do not vary continuously when crossing cell boundaries.

Another, more involved, way to construct the field at an arbitrary point inside the cell is via trilinear interpolation of the values of Q at each cell vertex:

$$\begin{aligned} Q(x,y,z) = {} & \alpha_1 + \alpha_2 \Delta_x + \alpha_3 \Delta_y + \alpha_4 \Delta_z + \alpha_5 \Delta_x \Delta_y \\ & + \alpha_6 \Delta_y \Delta_z + \alpha_7 \Delta_x \Delta_z + \alpha_8 \Delta_x \Delta_y \Delta_z, \end{aligned} \tag{14}$$

where the α_i's are given by

$$\begin{aligned} \alpha_1 &= Q_1, \\ \alpha_2 &= Q_2 - Q_1, \\ \alpha_3 &= Q_3 - Q_1, \\ \alpha_4 &= Q_5 - Q_1, \\ \alpha_5 &= Q_4 - Q_3 - Q_2 + Q_1, \\ \alpha_6 &= Q_7 - Q_5 - Q_3 + Q_1, \\ \alpha_7 &= Q_6 - Q_5 - Q_2 + Q_1, \\ \alpha_8 &= Q_8 - Q_7 - Q_6 - Q_4 + Q_2 + Q_5 + Q_3 - Q_1, \end{aligned} \tag{15}$$

in which Q_i denote the values of $Q(x,y,z)$ at the cell vertices P_i, as indicated in Fig. 4. In Eq. (14), Δ_x, Δ_y and Δ_z are given by

$$\begin{aligned} \Delta_x &= \frac{x_{\rm ray} - x_1}{h} = \frac{a + (\chi - \chi_A)\sin\theta\cos\varphi}{h}, \\ \Delta_y &= \frac{y_{\rm ray} - y_1}{h} = \frac{b + (\chi - \chi_A)\sin\theta\sin\varphi}{h}, \\ \Delta_z &= \frac{z_{\rm ray} - z_1}{h} = \frac{c + (\chi - \chi_A)\cos\theta}{h}, \end{aligned} \tag{16}$$

where h is the cell size, and $(x_{\rm ray}, y_{\rm ray}, z_{\rm ray})$, (x_1, y_1, z_1) and (a,b,c) are, respectively, the coordinates of the given ray (cf. Eq. (2)), the point P_1, and the point A w.r.t. point P_1.

Using Eqs. (14), (15) and (16), it is possible to compactify the expression for Q as

$$Q(\chi,\theta,\varphi) = \sum_{N=1}^{4} d_N (\chi - \chi_A)^{N-1}, \tag{17}$$

where the coefficients d_N depend on $a, b, c, \alpha_i, \theta$ and φ, but not on χ (note also that we are now specifying Q as a function of spherical coordinates). Their expression is given in Appendix B. Since the d_N's do not depend on χ, one can combine Eqs. (12) and (17) to write

$$I_c = \sum_{N=1}^{4} d_N \int_{\chi_B}^{\chi_A} K(\chi) (\chi - \chi_A)^{N-1} \mathrm{d}\chi. \tag{18}$$

The integrand in the above equation is a polynomial in χ, and so the integral can be solved analytically [37]. Therefore, the calculation of the integral in the cell reduces to the geometrical exercise of determining the position of points A and B and the evaluation of the quantity Q at the cell vertices. In Secs. IV B and IV D we shall see a concrete example of the use of these formulae when we apply it to lensing.

Equations (13) and (18) provide two possible ways to compute the final result. The latter has the advantage of allowing for the integration of a continuous field when crossing cell boundaries, by appropriately evaluating the fields at the vertices of the cells (see next section). On the other hand, the use of Eq. (13) does not involve evaluating the fields at the cell vertices (which do not exist in default `RAMSES`), making it therefore more computationally efficient. In Sec. VI D, we compare results based on Eqs. (13) and (18).

In this case Box 1 and Box 2 can be run simultaneously.

[10] The points χ_A and χ_B can lie anywhere inside the cell and not necessarily at the intersection of the line-of-sight with the cell faces. This is, for instance, the general case at the start and end of particle time steps.

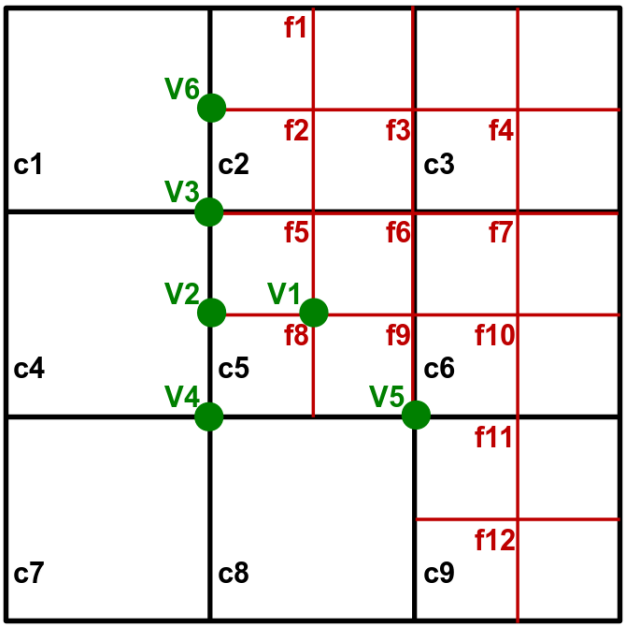

FIG. 6. Two-dimensional illustration of a boundary region. The coarse cells are labelled from c1, c2, etc., and a few fine cells are labelled f1, f2, etc. The vertices V1 to V6 are illustrative vertices where the fields values are to be evaluated from interpolation from the cell centers, which is where RAMSES evaluates the fields by default.

H. Field values at cell vertices

In default RAMSES, the field values are evaluated at cell centers, but the application of Eq. (18) above requires them to be interpolated to cell vertices. When doing so, one can ensure that the fields reconstructed with trilinear interpolation (cf. Eq. (14)) vary continuously from cell to cell and also that the total mass [11] is preserved. This is trivial on a regular mesh without AMR, but requires special care at the boundary of refined regions. We discuss these subtle issues in this subsection.

Figure 6 illustrates a refinement boundary in two dimensions (we shall later generalize the discussions to three dimensions). In the figure, the nine coarse level cells shown are labelled from c1 to c9 and a few fine level cells are labelled from f1 to f12. The vertices V1 to V6 represent different types of vertices at which one wishes the evaluate the fields. To be concrete, we shall take the example of the matter density field, ρ, but the interpolation scheme that we describe here is applied to any type of field.

For vertices of the type of vertex V1, which is shared by four cells of the same level, one can simply set $\rho_{\rm V1} = \big(\rho_{\rm f5}+\rho_{\rm f6}+\rho_{\rm f8}+\rho_{\rm f9}\big)/4$, where ρ_i are the values at the relevant cell centers. This average ensures that ρ varies continuously from one cell to another when computed with Eq. (14) (or its two-dimensional version). In regular mesh simulations, this is the only type of vertex. The vertex V2, which lies at the middle of a coarse cell edge and is shared by two fine cells, can be straightforwardly computed after the values in vertices V3 and V4 have been determined. Explicitly, $\rho_{\rm V2} = \big(\rho_{\rm V3}+\rho_{\rm V4}\big)/2$. This ensures, for instance, that two rays that are infinitesimally close to vertex V2, with one crossing cell c4 and another crossing fine cells f5 or f8, experience the same density field (as reconstructed with trilinear interpolation), i.e., there is no sharp discontinuity between the density experienced by one ray and the other.

The cases of vertices V3, V4 and V5 are more involved as they are shared by both coarse and fine cells. Let us consider vertex V3 as an example, which is shared by cells c1, c4, f2 and f5. It is natural to assume that the density at V3 depends on the density in each of these four cells. The simplest way to write this is as

$$\rho_{\rm V3} = \alpha_{\rm c1}\rho_{\rm c1} + \alpha_{\rm c4}\rho_{\rm c4} + \alpha_{\rm f2}\rho_{\rm f2} + \alpha_{\rm f5}\rho_{\rm f5}, \qquad (19)$$

where the α's are some weights to be determined. Let us focus on $\alpha_{\rm c1}$. The mass associated with cell c1 is $\rho_{\rm c1}h^2$, where h in this subsection is the coarse cell size. After the interpolation, we want the mass of this cell to be the sum of the masses associated with each of its vertices. Hence, the mass associated with vertex V3 due to cell c1 is $m^{\rm c1}_{\rm V3} = \rho_{\rm c1}h^2/4$. Due to mass conservation, the value of $m^{\rm c1}_{\rm V3}$ is *re-distributed* by vertex V3 to other cells at the boundary. In this sense, we can colloquially describe vertex V3 as a *mass reservoir that is collecting mass from c1 and redistributing it to neighbouring cells*. This mass distribution constraint can be written as

$$\frac{\rho_{\rm c1}h^2}{4} = \frac{\alpha_{\rm c1}\rho_{\rm c1}}{4}\left(h^2+h^2+\left(\frac{h}{2}\right)^2+\left(\frac{h}{2}\right)^2 + \frac{1}{2}\left(\frac{h}{2}\right)^2+\frac{1}{2}\left(\frac{h}{2}\right)^2+\frac{1}{2}\left(\frac{h}{2}\right)^2+\frac{1}{2}\left(\frac{h}{2}\right)^2\right). \qquad (20)$$

In this equation, the first two terms on the RHS represent the *mass from V3 due to cell c1 that is redistributed to cells c1 and c4* ($\alpha_{\rm c1}\rho_{\rm c1}/4$ is the density that V3 contributes to c1 and c4 and h^2 is the area of cells c1 and c4). The third and forth terms are the same as the first two, but for cells f2 and f5 (note that $(h/2)^2$ is the area of the fine cells). The last four terms on the RHS of Eq. (20) must be included as vertex V3 also contributes to the masses in cells f1, f2, f5 and f8 via vertices V2 and V6. Since the contribution from V3 to $\rho_{\rm V2}$ and $\rho_{\rm V6}$ is only $\rho_{\rm V3}/2$, each of these four terms gets a factor of $1/2$ compared to the third and forth terms. We can solve for $\alpha_{\rm c1}$ in Eq. (20) and, to facilitate the discussions below, we write the result as

$$\alpha_{c1} = \left(N_c + \frac{1}{4}\sum_i^{N_f} N_{fv,i}\right)^{-1}, \qquad (21)$$

where N_c is the number of coarse cells that share vertex V3, N_f is the number of fine cells that share vertex V3 and $N_{fv,i}$ is

[11] In this subsection, we use the word "mass" to denote the product of a surface or volume density with an area or volume.

the number of vertices that the fine cell i that shares V3 has at the boundary ($i =$ f2, f5). Explicitly, for V3 we have $N_c = 2$, $N_f = 2$ and $N_{fv,\mathrm{f2}} = N_{fv,\mathrm{f5}} = 2$. The coefficient α_{c4} in Eq. (19) is obtained in the same way: Eq. (20) remains the same, just with ρ_{c1} replaced by ρ_{c4} (which appears on both sides of the equation and therefore cancels). Hence, $\alpha_{\mathrm{c1}} = \alpha_{\mathrm{c4}}$.

The determination of the remaining coefficients, α_{f2} and α_{f5} differs from α_{c1} and α_{c4}. To determine α_{f2}, the RHS of Eq. (20) remains the same, just with ρ_{c1} and α_{c1} replaced by ρ_{f2} and α_{f2}, respectively. From the reasoning that led to Eq. 20, one could naively think that the LHS would be simply given by *the mass that vertex V3 collects from f2*, $m_{V3}^{f2} = \rho_{f2}(h/2)^2/4$. However, recall that vertex V3 also contributes to the mass in cells f2 and f1 via V6 (this is why two of the last four terms in Eq. (20) appear). As a result, the mass that *V6 collects from f2*, $m_{V6}^{f2} = \rho_{f2}(h/2)^2/4$ must also be included in the LHS of the mass distribution equation that determines α_{f2}. Explicitly,

$$\frac{\rho_{\mathrm{f2}}}{2}\left(\frac{h}{2}\right)^2 = \frac{\alpha_{\mathrm{f2}}\rho_{\mathrm{f2}}}{4}\left(h^2 + h^2 + \left(\frac{h}{2}\right)^2 + \left(\frac{h}{2}\right)^2 + \frac{1}{2}\left(\frac{h}{2}\right)^2 + \frac{1}{2}\left(\frac{h}{2}\right)^2 + \frac{1}{2}\left(\frac{h}{2}\right)^2 + \frac{1}{2}\left(\frac{h}{2}\right)^2\right). \tag{22}$$

The equation for α_{f5} is the same (apart from $_{\mathrm{f2}} \rightarrow_{\mathrm{f5}}$) and the equation for both can be written as

$$\alpha_{\mathrm{j}} = \frac{N_{fv,j}}{4}\left(N_c + \frac{1}{4}\sum_i^{N_f} N_{fv,i}\right)^{-1}, \tag{23}$$

where $i, j =$ f2, f5. The meaning of N_c, N_f and $N_{fv,i}$ is the same as above and since $N_{fv,\mathrm{f2}} = N_{fv,\mathrm{f5}}$, then $\alpha_{\mathrm{f2}} = \alpha_{\mathrm{f5}}$

Equations (21) and (23) were written in terms of N_c, N_f and $N_{fv,i}$ because this way they also hold for the other vertices. By following the same steps for V4 and V5 we can write:

$$\rho_{\mathrm{V4}} = \alpha_{\mathrm{c4}}\rho_{\mathrm{c4}} + \alpha_{\mathrm{c7}}\rho_{\mathrm{c7}} + \alpha_{\mathrm{c8}}\rho_{\mathrm{c8}} + \alpha_{\mathrm{f8}}\rho_{\mathrm{f8}}, \tag{24}$$

$$\rho_{\mathrm{V5}} = \alpha_{\mathrm{c8}}\rho_{\mathrm{c8}} + \alpha_{\mathrm{f9}}\rho_{\mathrm{f9}} + \alpha_{\mathrm{f10}}\rho_{\mathrm{f10}} + \alpha_{\mathrm{f11}}\rho_{\mathrm{f11}}, \tag{25}$$

where the coefficients associated with coarse and fine cells are obtained as in Eq. (21) and (23), respectively. For V4, $N_c = 3$, N_f = 1 and $N_{fv,\mathrm{f8}} = 3$, whereas for V5 $N_c = 1$, $N_f = 3$, $N_{fv,\mathrm{f9}} = 2$, $N_{fv,\mathrm{f10}} = 1$ and $N_{fv,\mathrm{f11}} = 2$. Note that in Eqs. (19), (24) and (25), the summed value of the α weights adds up to unity, as it should.

1. Generalization to three dimensions

In three dimensions, the above derivation holds with only a few generalizations. When writing the RHS of mass distribution equations, Eqs. (20) and (22), for each vertex, in addition to considering the contribution from vertices that lie at the edge of coarse cell vertices (which get a factor of $1/2$), one must also consider the contribution to vertices that lie at the center of the coarse cell faces, which get a factor of $1/4$. Moreover, in two dimensions, the ratio of the area of a fine to coarse cell is $1/4$, whereas in three dimensions the ratio of their volumes is $1/8$. Bearing these two things in mind, it is possible to show that the weights associated with coarse cells are given by

$$\alpha_{\mathrm{coarse}} = \left(N_c + \frac{1}{8}\sum_i^{N_f} N_{fv,i}\right)^{-1}, \tag{26}$$

and the weights associated with fine cells by

$$\alpha_j = \frac{N_{fv,j}}{8}\left(N_c + \frac{1}{8}\sum_i^{N_f} N_{fv,i}\right)^{-1}. \tag{27}$$

These expressions differ from their two-dimensional counterparts by replacing the factors of $1/4$ by $1/8$. The meaning of N_c, N_f and $N_{fv,i}$ is the same as in two dimensions. Just to give an example, consider a vertex V that is shared by seven coarse cells and one fine cell. In the scheme described above, the density at this vertex is

$$\rho_{\mathrm{V}} = \alpha_{\mathrm{coarse}}\sum_{i=1}^{7}\rho_{i,\mathrm{coarse}} + \alpha_{\mathrm{fine}}\rho_{\mathrm{fine}}, \tag{28}$$

where $\rho_{i,\mathrm{coarse}}$ is the density at the center of the i-th coarse cell and ρ_{fine} is the density at the center of the fine cell. For this case, $N_c = 7$, $N_f = 1$ and $N_{fv,\mathrm{fine}} = 7$ in Eqs. (26) and (27).

Analogously to the two dimensional case, once the density at the vertices that are shared by both coarse and fine cells is determined, then (i) the density at vertices that lie at the middle of a coarse cell edge is given by the average of the densities of the two coarse cell vertices of that edge; and (ii) the densities at vertices that lie at the center of a coarse cell face is given by the average of the densities at the four coarse cell vertices of that face. The density at vertices that are shared by eight cells of the same level (i.e. not in a refinement boundary) is given by the average value of the density at those eight cell centers.

As a test of our interpolation scheme, we have measured the total mass inside simulation boxes by using the values at the cell centers and at the cell vertices. The agreement between the two ways of measuring the total mass was perfect for meshes with and without refinements, which confirms that the design and implementation of our interpolation is correct. We note also that these operations to interpolate the field values from cell centers to cell vertices naturally add some computational costs to the code, and hence, it is desirable to reduce the number of times these operations should be performed. For instance, since a single cell can contain a large number of rays, the interpolation needs to be performed only once to compute the integral for all rays. Morevoer, if the fields at the vertices of a given cell do not change from one time step to the other (e.g., if it is a coarse cell that is not at a refinement boundary and the time step taken was a fine one), then one can

also store the interpolated values from the previous time step, thereby saving some computational time.

Before proceeding, we note that the interpolation scheme, as describe in this section, represents in practice a form of adaptive smoothing of the fields used in the ray integration [27]. This is in the sense that the field values at a given cell vertices (and hence the interpolated field inside that cell) depend on the field values on neighbouring cells. The reason why we dub this an adaptive smoothing is because the size of the *smoothing kernel* (roughly the volume occupied by all the cells that are used in the interpolation) depends on the sizes of the given cell and its neighbours, which in turn depends on the local matter density.

IV. WEAK LENSING SIMULATIONS: METHOD

In this section, we explain how our algorithm can be applied to studies of weak gravitational lensing.

A. Lensing basics

We start with a brief recap of the basics of gravitational lensing (see e.g. Refs. [17–20, 52, 53]). In a perturbed Friedmann-Robertson-Walker (FRW) spacetime, the line element in the absence of anisotropic stress can be written as (considering only scalar perturbations)

$$\mathrm{d}s^2 = \left(1+\frac{2\Phi}{c^2}\right)c^2\mathrm{d}t^2 - a^2\left(1-\frac{2\Phi}{c^2}\right)\mathrm{d}s^2_{\mathrm{space}}, \quad (29)$$

where $a = 1/(1+z)$ is the scale factor. Photons travelling from distant sources towards an observer get their trajectories bent due to the intervening gravitational potential, Φ. The (unobserved) angular position of the source on the source plane, $\vec{\beta}$, is related to the observed one, $\vec{\theta}$, by the lensing deflection angle $\vec{\alpha}$ as

$$\begin{aligned}\beta^i &= \theta^i + \alpha^i \\ &= \theta^i - \frac{2}{c^2}\int_0^{\chi_s}\frac{(\chi_s-\chi)}{\chi_s}\nabla^{x^i}\Phi(\chi,\vec{\beta}(\chi))\mathrm{d}\chi \\ &= \theta^i - \frac{2}{c^2}\int_0^{\chi_s}\frac{(\chi_s-\chi)\,\chi}{\chi_s}\nabla^{\beta^i}\Phi(\chi,\vec{\beta}(\chi))\mathrm{d}\chi, \quad (30)\end{aligned}$$

where $i = 1, 2$ denotes the two perpendicular directions to the line-of-sight. The third line in Eq. (30) is obtained from the second one by defining the derivatives w.r.t. the angular coordinate $\nabla_{\beta^i} = \chi\nabla_{x^i}$, or equivalently, $\nabla^{\beta^i} = \nabla^{x^i}/\chi$. The Jacobian matrix, A^i_j, of this source-to-observer mapping is obtained by differentiating the above equation w.r.t. $\vec{\theta}$ as in

$$A^i_j = \nabla_{\theta^j}\beta^i = \delta^i_j - \frac{2}{c^2}\int_0^{\chi_s} g\left(\chi_s,\chi\right)\nabla^{\beta^i}\nabla_{\theta^j}\Phi(\chi,\vec{\beta}(\chi))\mathrm{d}\chi, \quad (31)$$

where $g\left(\chi_s,\chi\right) = \left(\chi_s-\chi\right)\chi/\chi_s$. Note that the integral is performed along the perturbed path of the photon, as indicated by the $\beta(\chi)$ dependence of the potential inside the integral. Note also that one of the derivatives of Φ is w.r.t. $\vec{\beta}$ and another w.r.t. $\vec{\theta}$. These two aspects add complication to the ray tracing, but they can be neglected to obtain approximate solutions. To first order, we can write

$$A^i_j = \delta^i_j - \frac{2}{c^2}\int_0^{\chi_s} g\left(\chi_s,\chi\right)\nabla^{\theta^i}\nabla_{\theta^j}\Phi(\chi,\vec{\theta}(\chi))\mathrm{d}\chi, \quad (32)$$

in which the integral is now peformed along the unperturbed apparent direction of the photons, which is the so-called *Born approximation*, and the derivatives are now both w.r.t. $\vec{\theta}$, which amounts to neglecting the so-called *lens-lens coupling*[12]). The lensing results that we present in this paper are obtained under these two approximations, which are generally found to be valid, at least in what concerns determinations of the power spectrum of lensing quantities [27]. The generalization of our ray tracing calculations to follow the rays in their perturbed paths, as well as calculations that take lens-lens coupling into account is the subject of ongoing work (see e.g. the Appendix of Ref. [37] for a discussion).

Equation (32) can be written in matrix form

$$\hat{A} = \begin{bmatrix}1-\kappa-\gamma_1 & -\gamma_2 \\ -\gamma_2 & 1-\kappa+\gamma_1\end{bmatrix}, \quad (33)$$

to define the *lensing convergence*, κ,

$$\begin{aligned}\kappa &= 1 - \left(A^1_1 + A^2_2\right)/2 \\ &= \frac{1}{c^2}\int_0^{\chi_s} g\left(\chi_s,\chi\right)\left[\nabla^1\nabla_1\Phi + \nabla^2\nabla_2\Phi\right]\mathrm{d}\chi, \quad (34)\end{aligned}$$

and the two components of the lensing shear $\vec{\gamma} = (\gamma_1, \gamma_2)$,

$$\begin{aligned}\gamma_1 &= -\left(A^1_1 - A^2_2\right)/2 \\ &= \frac{1}{c^2}\int_0^{\chi_s} g\left(\chi_s,\chi\right)\left[\nabla^1\nabla_1\Phi - \nabla^2\nabla_2\Phi\right]\mathrm{d}\chi, \quad (35) \\ \gamma_2 &= -A^1_2 = -A^2_1 \\ &= \frac{2}{c^2}\int_0^{\chi_s} g\left(\chi_s,\chi\right)\nabla^1\nabla_2\Phi\mathrm{d}\chi, \quad (36)\end{aligned}$$

where we have now denoted $\nabla_i \equiv \nabla_{\theta^i}$ for compactness of notation.

B. Lensing integration in the code

The integrals of Eqs. (34), (35) and (36) can be found by using the algorithm outlined in Sec. III G. In the case of lensing, the integration kernel in Eq. (12) is given by (up to factors $\propto 1/c^2$) $K \equiv g(\chi_s,\chi)$, and the quantity Q that one needs to evaluate at cell vertices is $Q = \nabla^1\nabla_1\Phi + \nabla^2\nabla_2\Phi$ for κ, $Q = \nabla^1\nabla_1\Phi - \nabla^2\nabla_2\Phi$ for γ_1, and $Q = \nabla^1\nabla_2\Phi$, for γ_2.

[12] Lens-lens coupling refers to the correlation between the distortions of the sources with the intervening sources that act as lenses, whose images and positions seen by the observer are also distorted.

Consequently, the contribution of each crossed cell to the integral according to method NGP (cf. Eq. (13)) is given by

$$I_c = Q_c \int_{\chi_B}^{\chi_A} \frac{\chi_s - \chi}{\chi_s} \chi \mathrm{d}\chi = \frac{Q_c}{\chi_s} \left[\frac{\chi_s}{2} \left(\chi_A^2 - \chi_B^2 \right) - \frac{1}{3} \left(\chi_A^3 - \chi_B^3 \right) \right]. \quad (37)$$

Alternatively, according to Eq. (18), the contribution is

$$I_c = \frac{1}{\chi_s} \sum_{N=1}^{4} d_N \int_{\chi_B}^{\chi_A} \chi \left(\chi_s - \chi \right) \left(\chi - \chi_A \right)^{N-1} \mathrm{d}\chi = \frac{1}{\chi_s} \sum_{N=1}^{4} d_N \left[\frac{R^N}{N} \left(\chi_A - \chi_s \right) \chi_A + \frac{R^{N+1}}{N+1} \left(2\chi_A - \chi_s \right) + \frac{R^{N+2}}{N+2} \right], \quad (38)$$

where $R = \chi_B - \chi_A$.

What is left to specify is the relation between the quantities $\nabla^1\nabla_1\Phi$, $\nabla^2\nabla_2\Phi$ and $\nabla^1\nabla_2\Phi$, with the values of $\partial_a\partial_b\Phi$ $(a, b = x, y, z)$ that are actually computed on the simulation mesh (see the subsection below). This relation is

$$\nabla^1\nabla_1\Phi = \sin^2\varphi \left[\partial_x\partial_x\Phi\right] + \cos^2\varphi \left[\partial_y\partial_y\Phi\right] - \sin 2\varphi \left[\partial_x\partial_y\Phi\right], \quad (39)$$

$$\nabla^2\nabla_2\Phi = \cos^2\varphi\cos^2\theta \left[\partial_x\partial_x\Phi\right] + \sin^2\varphi\cos^2\theta \left[\partial_y\partial_y\Phi\right] + \sin^2\theta \left[\partial_z\partial_z\Phi\right] + \sin 2\varphi\cos^2\theta \left[\partial_x\partial_y\Phi\right] - \sin\varphi\sin 2\theta \left[\partial_y\partial_z\Phi\right] - \cos\varphi\sin 2\theta \left[\partial_x\partial_z\Phi\right], \quad (40)$$

$$\nabla^1\nabla_2\Phi = \frac{\cos\theta}{\sin\theta} \left[\cos\varphi\sin\varphi \Big(\left[\partial_y\partial_y\Phi\right] - \left[\partial_x\partial_x\Phi\right] \Big) + \Big(\cos^2\varphi - \sin^2\varphi \Big) \left[\partial_x\partial_y\Phi\right] \right] + \sin\varphi \left[\partial_x\partial_z\Phi\right] - \cos\varphi \left[\partial_y\partial_z\Phi\right]. \quad (41)$$

The above equations are derived by associating the two spherical coordinates φ and θ that specify the incoming direction of the rays with θ_1 and θ_2. Then, the expressions follow straightforwardly from applying $\nabla_i\nabla_j\Phi = \partial_i\partial_j\Phi - \Gamma^k_{ij}\partial_k\Phi$, with Γ^k_{ij} being the Christoffel symbols of the line element[13] $\mathrm{d}s^2_{\mathrm{space}} = \mathrm{d}\chi^2 + \chi^2\mathrm{d}\theta^2 + \chi^2\sin^2\theta\mathrm{d}\varphi^2$, and then writing ∂_φ and ∂_θ in terms of ∂_x, ∂_y, ∂_z according to Eq. (2).

C. Calculation of $\partial_a\partial_b\Phi$

The values of $\partial_a\partial_b\Phi$ $(a, b = x, y, z)$ can be computed at the center of a given cell by finite differencing the values of Φ on neighbouring cells. If a cell has all its neighbours at the same refinement level, this calculation is straightforward. For instance, if $\Phi_{i,j,k}$ is the gravitational potential on the cell labelled by $\{i, j, k\}$, then we have

$$\partial_x\partial_x\Phi = \frac{\Phi_{i-1,j,k} - 2\Phi_{i,j,k} + \Phi_{i+1,j,k}}{h^2}$$
$$\partial_x\partial_y\Phi = \frac{\Phi_{i+1,j+1,k} + \Phi_{i-1,j-1,k} - \Phi_{i+1,j-1,k} - \Phi_{i-1,j+1,k}}{4h^2}, \quad (42)$$

as two representative examples. The other components of $\partial_a\partial_b\Phi$ are obtained similarly. However, some complications arise at boundary regions of refinements. As an example, consider that we wish to compute $\partial_x\partial_x\Phi$ on cell f5 in Fig. 6, where x and y are, respectively, the horizontal and vertical directions on the figure. The fine cell f5 is missing the neighbour that would exist if coarse cell c4 had been refined. One can think of two ways to compute the missing values that are needed for the finite difference. One option is to interpolate the values of Φ obtained from the coarse level to the point in cell c4 where the center of the relevant son cell would be if it existed. This value could then be used in a fine-level finite difference to compute $\partial_x\partial_x\Phi$ in cell f5. Another option is interpolating directly the coarse values of $\partial_x\partial_x\Phi$ in cells c1, c2, c4 and c5 to the center of cell f5, without finite differencing.

To test these two approaches, we have set up a grid with more that one refinement level in the code and used the cell centers to define a Gaussian potential on the mesh. We then compared the analytical result of $\partial_a\partial_b\Phi$ with the result given by the code. We have found that the second approach agrees very well with the analytical result, but the first option showed larger discrepancies at the refinement boundaries. This is because by taking the finite difference using interpolated values, one amplifies the interpolation error in Φ by the factor of h^{-2}, which enters in the finite difference. In the results that follow, we have therefore implemented the second approach, which is also more computationally efficient[14]. Once $\partial_a\partial_b\Phi$ is eva-

[13] We note in passing that by using this line element one takes into account the curvature of the sky. However, Eqs. (39), (40) and (41) remain the same if ∇_1 and ∇_2 are interpreted as being derivatives w.r.t. the coordinates $(x_1, x_2) = \chi(\theta_1, \theta_2)$, i.e., by taking the *flat sky* approximation. This coordinate system is essentially a Cartesian system rotated such that the direction of the incoming ray is perpendicular to the $x_1 - x_2$ plane. A simple argument for this equivalence is that the sphere is locally flat, which means that the curvature can in practice be neglected when one takes derivatives.

[14] As a technical point, imagine that there is a CPU domain along the line that contains vertices V6, V2, V3 and V4 in Fig. 6. In RAMSES, there are "communication buffers" at the CPU domain boundaries, i.e., regions in the next

luted at the center of the cells, then its interpolation to cell vertices is as described in Sec. III H.

D. Alternative lensing integration in the code

The lensing methodology described above involves the calculation of $\partial_a\partial_b\Phi$ $(a, b = x, y, z)$ on the mesh, which inevitably adds some computational overheads. However, as described in this section, it is possible to compute κ and γ by integrating only quantities that are computed by default in `RAMSES`.

Equation (34) can be written as

$$c^2\kappa = \int_0^{\chi_s} g\left(\chi_s,\chi\right)\left[\nabla^2\Phi - \nabla_\chi^2\Phi\right]\mathrm{d}\chi$$
$$= \frac{3}{2}\Omega_{m0}H_0^2\int_0^{\chi_s} g\left(\chi_s,\chi\right)\frac{\delta}{a}\mathrm{d}\chi \quad + \quad \int_0^{\chi_s}\nabla_\chi\Phi\,\partial_\chi g\mathrm{d}\chi \quad + \quad \frac{1}{c}\int_0^{\chi_s} g\,\partial_t\left(\nabla_\chi\Phi\right)\mathrm{d}\chi \quad - \quad g\nabla_\chi\Phi\Big|_0^{\chi_s}, \tag{43}$$

where in the second equality we have used the Poisson equation to relate the comoving three-dimensional Laplacian to the matter density contrast, δ, and the term $g\nabla_\chi^2\Phi$ was integrated by parts using $\nabla_\chi = \partial_t/c + \partial_\chi$, where t is the physical time. The integration of the first term on the RHS of Eq. (43) is the same as that of Eqs. (34), (35) and (36), but with δ/a as the quantity Q[15]. The second term is obtained analogously, but with $Q = \nabla_\chi\Phi = \sin\theta\cos\varphi\partial_x\Phi + \sin\theta\sin\varphi\partial_y\Phi + \cos\theta\partial_z\Phi$ (where $\partial_x\Phi$, $\partial_y\Phi$ and $\partial_z\Phi$ are the negative of the three components of the gravitational force) and the kernel $K = \partial_\chi g = 1 - 2\chi/\chi_s$. The third term involves the time derivative of the potential, which is not calculated in `RAMSES` by default. In the results presented in this paper, we neglect the contribution from this term, which is expected to be small anyway (see e.g. Ref. [21]).

The last term on the RHS of Eq. (43), which is a surface term, is exactly zero in theory, since the lensing kernel g vanishes at $\chi = 0$ and $\chi = \chi_s$. However, if one breaks down the calculation of this term into the contributions coming from each time step

$$g\nabla_\chi\Phi\Big|_0^{\chi_s} = \sum_{\text{time steps}} g\nabla_\chi\Phi\Big|_{\chi_f}^{\chi_i}, \tag{44}$$

where χ_i and χ_f are, respectively, the values of χ at the start and end of a particle time step, then the result is not zero due to discontinuities between particle time steps and at the boundary of boxes in the tile. This is because, if `RAMSES` updates the values of the potential from one time step to the next, then its value at the end of the current time step (when the ray is at $\chi = \chi_f^{t_\mathrm{current}}$) is not the same as at the start of the next time step (when the ray is at $\chi = \chi_i^{t_\mathrm{next}}$), where $\chi_f^{t_\mathrm{current}} = \chi_i^{t_\mathrm{next}}$ [16]. As a result,

$$g\nabla_\chi\Phi\Big|_0^{\chi_s} = g\nabla_\chi\Phi\Big|_{\chi=\chi_s} - g\nabla_\chi\Phi\Big|_{\chi=0} + \epsilon_\mathrm{disc}$$
$$= \epsilon_\mathrm{disc}, \tag{45}$$

where ϵ_disc denotes the cumulative error that arises due to the discontinuities at each time step. The latter are unavoidable since they are linked to the intrinsic discreteness of N-body simulations. Note however that the nonzero value of ϵ_disc, which comes from integrating the second term in the bracket of the first line of Eq. (43), means that the same discreteness also affects the integration of the first term there. Therefore, having ϵ_disc included in the calculation for each time step can reduce the error in the density integral of Eq. (43) that comes from the same discreteness. We have checked explicitly that including ϵ_disc brings the κ map obtained using Eq. (43) into closer agreement with the result from Eq. (34).

Note that the discreteness in time also introduces another source of error. For example, when computing the second integral in Eq. (43), the values of $\nabla_\chi\Phi$ are assumed to be constant in a time step, and therefore the integral misses the contribution that comes from the time evolution of fields within the time step. This also affects the integral of Eq. (34). However, we expect this error to be small in cosmological simulations where the fine time step is typically of order $\Delta a \sim 10^{-4}$ or smaller. In principle, our algorithm can be straightforwardly extended to interpolate the fields between neighbouring time steps so that they are continuous in time; however,

CPU's domain that are available to the present CPU. In order to compute the value of $\partial_a\partial_b\Phi$ in cell f5, then its CPU needs to access the value of $\partial_a\partial_b\Phi$ at cell C4, whose calculation involves Φ on cells further left of C4 (not shown in Fig. 6). These latter cells are outside of the communication buffer of standard `RAMSES`, which means that we had to increase its size. This is expected to make some communication

[15] In the current implementation of the code, a is taken to be constant during the time step integration. This should not lead to big errors in high-resolution simulations if the time steps are sufficiently small. There are however ways to go beyond this by, for instance, implementing the relation $a(\chi)$ in the integration.

[16] That is, the ending point at the current time step is the starting point at the next time step integration.

given the small error that this discontinuity causes, we will leave this for future implementations.

In the end, we have two ways to compute the lensing convergence. One is that of Eq. (34), which we call Method B and it involves only one integral term. The other, which we call Method A[17], uses Eq. (43) and in the current implementation of the code involves two integral terms and the inclusion of the surface term at each time step to cancel some of the errors of the integral terms. These two methods therefore respond differently to the discontinuities between time steps. Another numerical difference between these two methods is that Method B involves the calculation of $\partial_a\partial_b\Phi$, which for each cell requires using a different number of field values on neighbouring cells, compared to the evaluation of δ or $\nabla_\chi\Phi$ in Method A. In Sec. VI D, we shall see that the two methods give consistent results, which is telling that these numerical differences are not critical.

As we mentioned above, Method A is more convenient than Method B in the sense that it avoids the computationally expensive calculation of $\partial_a\partial_b\Phi$. On the other hand, it does not yield directly the lensing shear, which needs to be obtained indirectly from the κ maps. This can be done by (i) Fourier transforming $\kappa(\vec{\theta})$ to obtain $\tilde{\kappa}(\vec{\ell})$; (ii) computing the Fourier transform of the shear as [54]

$$(\tilde{\gamma}_1, \tilde{\gamma}_2) = \left(\frac{\ell_1^2 - \ell_2^2}{\ell_1^2 + \ell_2^2}\tilde{\kappa}, \frac{2\ell_1\ell_2}{\ell_1^2 + \ell_2^2}\tilde{\kappa}\right), \tag{46}$$

where $\ell_1 = \pi/\theta_1$, $\ell_2 = \pi/\theta_2$; and finally, (iii) inverse Fourier transforming $(\tilde{\gamma}_1, \tilde{\gamma}_2)$ to get $\left(\gamma_1(\vec{\theta}), \gamma_2(\vec{\theta})\right)$ [18].

V. A CODE TEST: LENSING BY A GAUSSIAN POTENTIAL

As a check of our ray tracing implementation for lensing, we tested the code results in a controlled setup for which we can obtain solutions with other integration methods. Specifically, we compute the lensing signal associated with a fixed Gaussian potential

$$\tilde{\Phi}(r) = A\exp\left[-\frac{r^2}{2\sigma^2}\right], \tag{47}$$

where the tilde denotes that the potential is written in code units. We set the parameters as $A = 10^4$ and $\sigma = 2\ \mathrm{Mpc}/h$ (the exact values are not critical for our tests). The center of the potential is located at the center of a box of size $L = 10\ \mathrm{Mpc}/h$ and the potential is defined on cell centers. The observer lies at the center of one of the box faces and we integrate 128×128 rays covering a $40 \times 40\ \mathrm{deg}^2$ field of view. The source redshift for this test is $z_s = 0.95$, but the rays are only integrated in the box that contains the observer and the Gaussian potential. We consider three domain mesh resolutions with 32, 64 and 256 cells per dimension. For each domain mesh resolution, we hierarchically refine the mesh towards the inner region of the box using two cubic-shaped refined levels. That is, if l_d is the domain level, then a ray moving away from the center of the potential will go through cells on level $l_d + 2$, then on level $l_d + 1$ and finally l_d. We create the refinement levels by appropriately distributing particles inside the box. However, these particles are only used to define the AMR structure and play no other role in this test, e.g., the potential felt by the rays is that of Eq. (47) and not that associated with the particle distribution. We let the code run as if it was a normal N-body simulation, but at each time step the particles are kept from moving to ensure that the AMR structure remains fixed. We ran this test on 8 CPUs using integration method B with Eq. (38) (cf. Sec. IV B), and compared our code results with the integral solutions obtained with an adaptive 1D numerical integrator from the GSL library [55] for the same ray settings.

The outcome of this test is shown in Fig. 7, which displays κ profiles (left), γ_1 profiles (middle) and γ_2 maps (right), for the tested resolutions (the κ and γ_1 profiles correspond to a given radial slice of the corresponding maps). Figure 7 illustrates the very good agreement between the expected result and that from our ray tracing code. In the κ and γ_1 panels, the error (the difference between the expected result and the ray tracing one) for the highest mesh resolution is kept well within 1%. The error is also smaller than 1% for the intermediate resolution, and even for the poorest resolution case it never exceeds $\approx 3\%$. When quoting these figures, we do not consider the radial scales where κ and γ_1 cross zero, since this artificially amplifies the relative error there. We have checked that these small errors are mostly caused by errors in the interpolation of the potential values from cell centers to cell vertices[19], and not due to the integration routines. This is why the agreement with the expected result becomes noticeably better when the cells become finer, and hence the interpolation more accurate [20]. For a spherically symmetric potential, $\gamma_2 = 0$, which makes it harder to quantify the code error. However, the γ_2 panels do show that its absolute value is close to zero (up to some weak noisy pattern) and that the agreement with the expected result improves with increased resolution. Note also that our κ, γ_1 and γ_2 results show no evidence of any inaccuracies caused by the interpolation at refinement boundaries.

This test represents an important validation of not only the integration algorithm, but also of the calculation of the tidal tensor, $\partial_a\partial_b\Phi$ and the interpolation scheme from cell centers to cell vertices. These are all routines that do not ex-

[17] This naming is the same as in Ref. [37].

[18] Equation (46) is valid only in the flat-sky approximation, in which Fourier modes and spherical harmonic multipoles are equivalent.

[19] We checked this by comparing the reconstructed potential values at cell vertices (cf. Sec. III H) with the values determined by Eq. (47).

[20] Although not explicit in Eq. (47), when written in code units, Φ acquires a factor of a^2, which we take to be constant in each time step. Since the time steps get smaller with increased resolution, this also contributes to the better accuracy seen in the higher resolution setting.

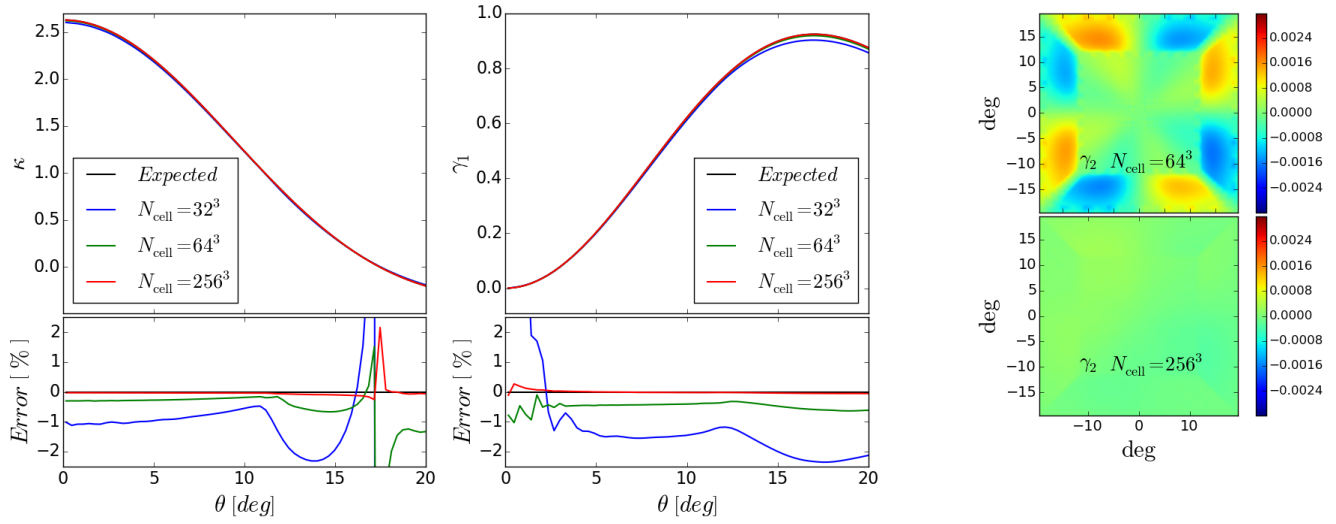

FIG. 7. Code lensing tests with a fixed Gaussian potential. From left to right, the panels show the profiles of κ, γ_1 and the maps of γ_2, for different domain grid resolutions, as labelled. The black curve shows the expected result obtained using the same potential and ray geometry, but with the equations solved with an independent Runge-Kutta numerical integrator. The panels below the κ and γ_1 profiles show the percentage error relative to the expected result. In the γ_2 panels, the expected result is $\gamma_2 = 0$.

ist in default RAMSES and this successful test demonstrates that they have been implemented correctly. Finally, we note that this test serves also as an important check of the code infrastructure, namely the fact that the ray propagation through the AMR structure and the communications between CPUs are done correctly – otherwise there would be errors coming from artifacts at the boundaries of CPU domains or not all rays would reach the observer at redshift $z = 0$.

VI. COSMOLOGICAL WEAK LENSING SIMULATIONS

In this section, we show and discuss some code results for cosmological weak lensing simulations. We start by summarising our simulation settings and ray geometry, and then show our code results for one- and two-point statistics, as well as for the lensing signal around dark matter haloes.

A. Summary of the simulation and tile settings

Our lensing maps were obtained by tracing $N_{\rm ray} = 2048 \times 2048$ rays from $z_s = 1$, covering a field of view with $10\times10\,\deg^2$. This yields an angular resolution of $0.005\,\deg = 0.3'$. To encompass this lightcone geometry, we employed the tiling scheme depicted in Fig. 1 with five $L = 512\,{\rm Mpc}/h$ boxes, in which we simulate a flat ΛCDM cosmology using $N_p = 512^3$ and $N_p = 1024^3$ dark matter tracer particles. We refer to these two resolutions as LowRes and HighRes, respectively. Our ΛCDM parameters are $\Omega_{b0} = 0.049$, $\Omega_{c0} = 0.267$, $\Omega_{\Lambda0} = 1-\Omega_{b0}-\Omega_{c0}$, $h = 0.671$, $n_s = 0.9624$, $\sigma_8 = 0.834$, in accordance with the recent results from the Planck satellite [56, 57] (but assuming that all neutrino species are massless). The simulation in each box in the tile stops after all rays have reached the end of the integration there (cf. Secs. III B, III C) and the observer is located at the center of the face of the last box (that which is furthest away from the sources). In particular, from the first to the last box, the ray integrations are performed in the redshift intervals, $[1, 0.86]$, $[0.86, 0.6]$, $[0.6, 0.38]$, $[0.38, 0.18]$ and $[0.18, 0]$, respectively.

For each particle resolution, we considered five realizations of the particle initial conditions (generated with the MPGRAFIC code [58] at $z = 49$) for each of the five boxes in the tile. This allows to construct $5^5 = 3125$ lensing maps by combining the integration results from each box realization. These maps are not all independent from one another [21], but they are equally likely realizations of the lightcone. This "shuffling" of different realizations of the boxes in the tile allows for a measure of the uncertainties associated with cosmic variance, specially for those boxes in the tile whose volume is only very partially covered by the light bundle. In these lensing map constructions, we do not mix the results from LowRes and HighRes boxes. If one requires all tiles to be completely independent from one another then we can construct 5 tiles for each resolution [22].

All these lensing maps are obtained using integration Method A with Eq. (38) (cf. Sec. IV D). However, for testing purposes, we have selected one combination of initial conditions for a HighRes tile to construct lensing maps with method B, using both Eq. (37) and Eq. (38). We refer to the re-

[21] For instance, consider the lensing map obtained by one realization of the lensing tile. Then, the lensing map constructed from this map by replacing the result of first box with another realization of the first box still gets the same contribution from all the other boxes.

[22] This is in the sense that using the available box realizations, one can only construct 5 tiles using each box realization only once.

sults obtained using Eq. (37) as Method B (NGP). If not specified, when we refer to Method A and Method B, we mean the result obtained using Eq. (38). We compare the different integration methods in Sec. VI D.

B. Convergence probability distribution function

The color maps in Fig. 8 show the κ fields obtained from one particular tile realization of the HighRes and LowRes ray tracing simulations. The upper panels show the maps as computed by our code and the lower panels show the maps smoothed by a Gaussian filter with size $2' \approx 0.03$ deg. The corresponding rightmost panels show the probability distribution function (PDF) of the κ fields shown in the color maps, but also for the other 4 independent tile realizations that we can construct from our simulations (cf. Sec. VI A).

The upper right panel shows that the PDFs of the HighRes and LowRes tiles are in good agreement for $\kappa \gtrsim 0.02$, but show some discrepancies at smaller values. In particular, the distribution of the HighRes realizations is shifted towards lower values of κ, relative to the LowRes ones. This result can be attributed to the differences in resolution. In particular, the particle CIC clouds in the LowRes case are larger than in the HighRes case and the particle mass is distributed to a larger volume. Hence, the LowRes simulations do not develop density troughs that are as deep as in the HighRes case, which pushes the low-κ tail of the distribution in the LowRes tiles to larger values of κ, as seen in the upper right panel of Fig. 8. By the same reasoning, the PDF of the HighRes simulations should be higher for larger values of κ because of the better resolved high density peaks. However, the PDFs for large values of κ becomes noteciably suppressed, which makes that assessment more difficult.

The lower right panel of Fig. 8 shows the same as the upper right panel, but for the smoothed maps. The size of the filter ($2'$) corresponds roughly to cluster size scales at $z \approx 0.5$ and to typical smoothing scales employed on real lensing maps [59, 60]. For the smoothed maps, the two resolutions now agree quite well for all values of κ. Compared to the unsmoothed cases, the smoothing suppresses the PDF for $\kappa \gtrsim 0.02$, which indicates that these values of κ were due to peaks with size smaller than $2'$. The smoothing, however, does not noticeably suppress the amplitude of the PDF for $\kappa \lesssim 0$.

C. Convergence power spectrum

Figure 9 shows our convergence power spectrum results for the tiles constructed with the HighRes (blue) and LowRes (green) simulation boxes. The solid lines indicate the median of the 3125 power spectra that we can construct for each resolution and the errorbars indicate the 25% and 75% percentiles. The black solid line shows the result given by the formula[23]

$$C_\ell^{\kappa\kappa} = \frac{9\Omega_m^2}{4}\left(\frac{H_0}{c}\right)^4 \int_0^{\chi_s} g^2\left(\chi_s,\chi\right)\frac{P_\delta(k=\ell/\chi,\chi)}{a^2}\mathrm{d}\chi, \tag{48}$$

with P_δ being the nonlinear matter power spectrum computed in the Halofit formalism presented in Ref. [61], and later tuned by Ref. [62] (we use the latter). The spectra from the HighRes and LowRes tiles are within each other's errorbars and agree also with the Halofit prediction for $\ell \lesssim 2.0 \times 10^3$. The small differences in between the two resolutions and with the Halofit prediction can be attributed to cosmic variance. The latter can be particularly important in weak lensing studies with small opening angles (compared to full sky), as observed for instance in the full-sky analysis of Ref. [24].

For $\ell \gtrsim 2 \times 10^3$, the Halofit prediction is above our code predictions. This result can be associated with at least two factors. The first one relates to the adaptive smoothing effect that is caused by our interpolation scheme from cell centers to cell vertices (to use Eq. (38)), and which works to suppress the power below a given angular scale (recall the discussion at the end of Sec. III H). The second factor relates to the accuracy with which Halofit describes the three-dimensional clustering in our simulations. Figure 10 compares the Halofit prediction for P_δ with the nonlinear matter power spectrum measured from our simulations at a number of epochs, as labelled. The figure shows that the clustering power in our HighRes and LowRes simulations is lower than what Halofit predicts for $\kappa \gtrsim 4\ h/\mathrm{Mpc}$ and $\kappa \gtrsim 1\ h/\mathrm{Mpc}$, respectively. These differences naturally propagate into the two-dimensional convergence power spectrum on small angular scales, which helps to explain why our simulations underpredict the Halofit result in Fig.9. This also explains why the suppression in power is more pronounced in the LowRes case. Although we do not explicitly test for that, we note that the agreement between our code results and Eq. (48) is expected to improve if instead of using Halofit to compute P_δ, one uses directly the nonlinear matter power spectrum measured from our simulations (see e.g. Refs. [22, 63]).

D. Comparison of different integration methods

Figure 11 compares the results from different ray integration methods for the PDF of κ (left panel) and its power spectrum (right panel). The three methods shown are method A, method B and method B (NGP), as labelled. Recall that method A and method B make use of Eq. (38) and method B (NGP) uses Eq. (37). The three integration methods are

[23] We have used the publicly available `CAMB Sources` software (http://camb.info/sources/) to compute this integral. Note that we are assuming the Limber approximation, which is valid for small fields of view like ours (compared to full sky). Morevoer, for small fields of view, the assumption of a flat sky is a good approximation, which in practice means that Fourier and spherical harmonic transforms give equivalent results.

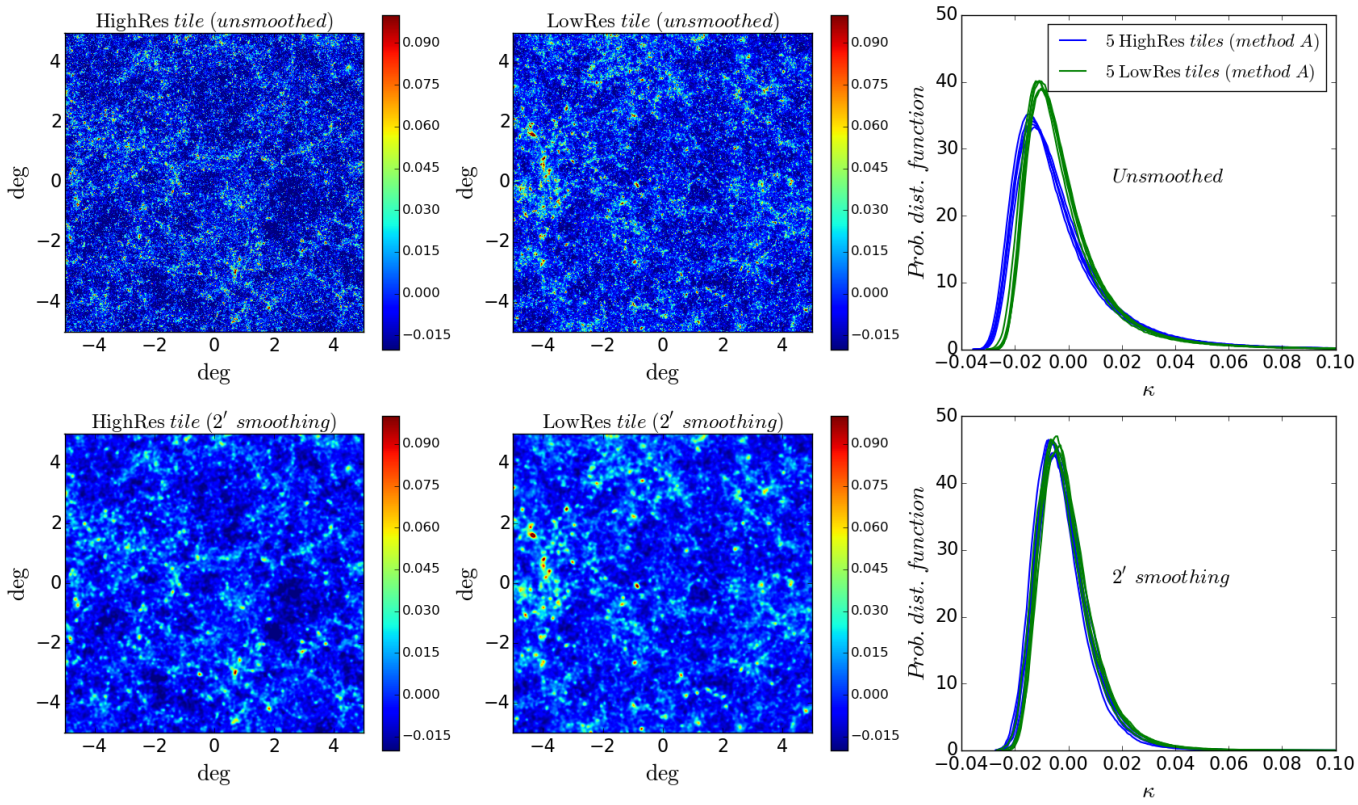

FIG. 8. Lensing convergence maps and κ probability distribution functions (PDFs). The color maps show the κ field obtained for a realization of the HighRes and LowRes tiles without any smoothing and with a Gaussian smoothing with width $2' \approx 0.03$ deg, as indicated in the panel titles. The right panels show the PDFs of the κ field displayed in the color maps and for four other independent tile realizations with and without Gaussian smoothing, as labelled. These results correspond to integration Method A with Eq. (38).

overall in good agreement in their PDF results, for both the unsmoothed and the smoothed cases. One notes that the PDFs of the two maps from method B are slightly shifted to higher values of κ, compared to the method A result. This could be attributed to differences in the detailed implementation of methods A and B (cf. Sec. IV D). Nevertheless, we stress that these are only small differences, which are in fact smaller than spread of the five method A realizations shown in Fig. 8. This agreement between the different integration methods is a reassuring result that can be regarded as a check of the consistency of our ray tracing implementation for lensing.

For the case of the convergence power spectrum, integration methods A and B are also in very good agreement for all the scales shown, which once again demonstrates the robustness of our ray tracing modules (any observed difference is much smaller than the spread due to cosmic variance shown in Fig. 9). The integration method B (NGP) agrees also very well with method A and B for $\ell \lesssim 5 \times 10^3$, but for larger values of ℓ the shape of its power spectrum agrees better with the Halofit prediction. This is due to the adaptive smoothing effects of our interpolation scheme from cell centers to cell vertices that affects methods A and B, but does not affect method B (NGP). In particular, the shape of the curve from method B (NGP) agrees with Halofit up to $\ell \sim 2 \times 10^4$. A more detailed assessment of the behavior of the convergence power spectrum on small scales would benefit from ray tracing simulations with higher resolution than those used for this paper, and hence we defer such investigations for future work.

On scales $\ell \gtrsim 2.0 \times 10^4$, the three curves exhibit an upturn that is caused by ray shot noise. We have checked that decreasing the number of rays traced makes the spurious effects of shot noise more noticeable at lower ℓ (not shown, but see e.g. Ref. [21]).

E. Halo lensing

Figure 12 shows the lensing signal around dark matter haloes found in the middle box of a HighRes tile (we call this Box 3) at $z = 0.5$. The halo catalogue was built using the `Rockstar` code [64]. We applied a mass[24] cut $M_{200} > 10^{14}\ M_\odot/h$ and considered only haloes with shape parameters $c/a > 0.55$ and $b/a > 0.75$, where $a > b > c$

[24] Our mass definition is $M_{200} = (4\pi/3)\,\rho_c(z) 200 R_{200}^3$, where R_{200} is the halo radius defined as the radial distance to the halo center within which the mean density is equal to 200 times the critical density in the Universe $\rho_c(z)$.

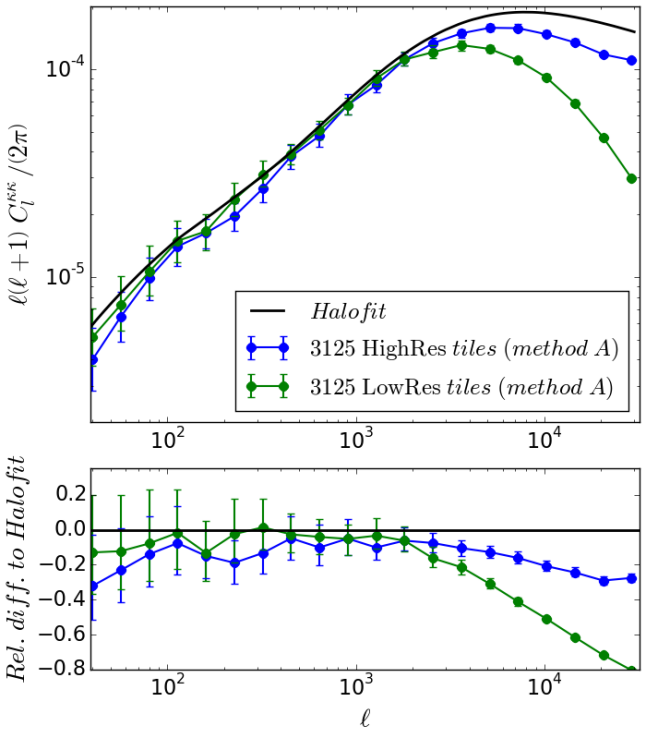

FIG. 9. Lensing convergence power spectrum from the maps constructed with integration method A (with Eq. (38)) for the HighRes (blue) and LowRes (green) tiles. The dots indicate the median and the errorbars indicate the 25% and 75% percentiles of the distribution of the 3125 tile realizations for each particle resolution. The black line indicates the semi-analytical prediction based on the Halofit formalism (Eq. (48)).

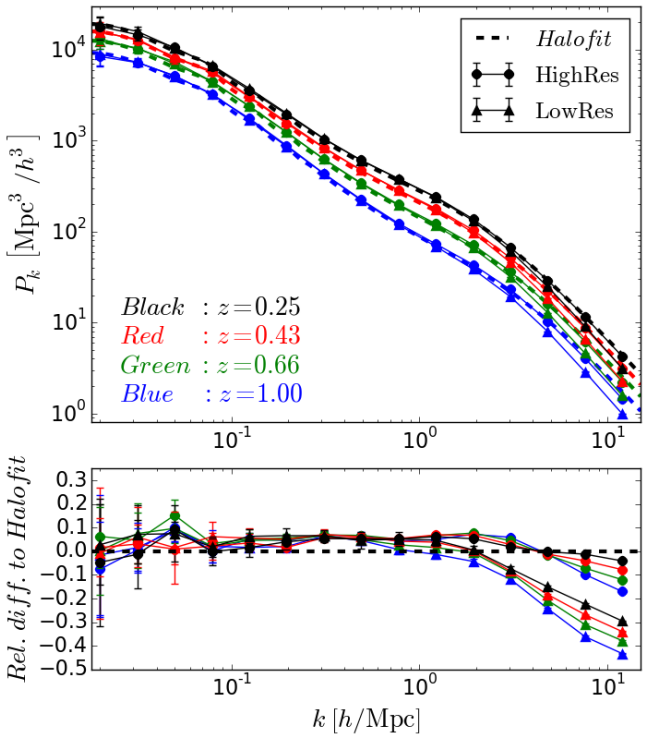

FIG. 10. Matter power spectrum measured from the HighRes (dots) and LowRes (triangles) simulations. For each redshift shown (different colors, as labelled), the symbols indicate the mean and the errorbars the standard deviation of the five realizations of the boxes that will trace the rays right after those redshifts. For example, the result for $z = 0.25$ was measured in the last boxes (those that have the observer), $z = 0.43$ in the second-to-last boxes, and so on. The dashed lines indicate the prediction using the Halofit formalism.

are the three ellipsoidal axis. We have also kept only haloes with $f_{\rm sub} < 0.1$, where $f_{\rm sub}$ is the fraction of halo mass associated with resolved substructure. The ten haloes shown are those which lie within the field of view. The color maps of Fig. 12 show the κ field zoomed into a 0.2×0.2 deg^2 region around the halo center determined by `Rockstar`. The profile panels below each color map show the corresponding spherically averaged convergence profile around the halo center [25]. The blue points correspond to the profiles obtained using the maps from the full tile, whereas the green ones show the profiles obtained from the signal computed by Box 3 alone. This helps to separate the contribution from the halo itself and from foreground and background structures. The solid red lines show the analytical Navarro-Frenk-White (NFW) [65] result (see e.g. Refs. [66–68]) computed with the mass and concentration (M_{200}, c_{200}) values given by `Rockstar`. The angular size associated with R_{200} is depicted by the yellow dashed lines in the color maps and by the vertical dashed lines in the profile panels.

The fact that κ peaks in our maps coincide with the halo angular positions determined by another independent code constitutes another consistency check of our ray tracing implementation for lensing. Furthermore, for all ten haloes shown and despite some expected differences, Fig. 12 shows that there is good agreement between our code results and the NFW analytical prediction for the amplitude and shape of the κ profiles. The observed differences can be caused by a variety of effects. For instance, despite our attempt to select the haloes that are the "most spherical" ($c/a > 0.55$ and $b/a > 0.75$) and devoid from substructure ($f_{\rm sub} < 0.1$), they are naturally still not perfectly spherical nor have perfectly smooth mass distributions. The nonsphericity and irregular shape of haloes leaves room for projection effects that are known to induce a bias between concentration and mass values estimated from lensing and those estimated from the 3D mass distribution in the simulations (see e.g. Refs. [69–71]. Moreover, owing also to projection effects and substructure, there can be offsets between the angular position of the halo center (around which we compute our lensing profiles) and of the κ peaks, which can lead to some differences to the NFW analytical prediction. Some of these differences are particularly noticeable for the second halo from the left in the upper

[25] The points at each angular scale show the mean κ at a number of points that sample a circle around the halo center with that angular size. The errorbars show the standard deviation around this mean. We evaluate the values of κ from the map via bilinear interpolation given the values of κ at each pixel.

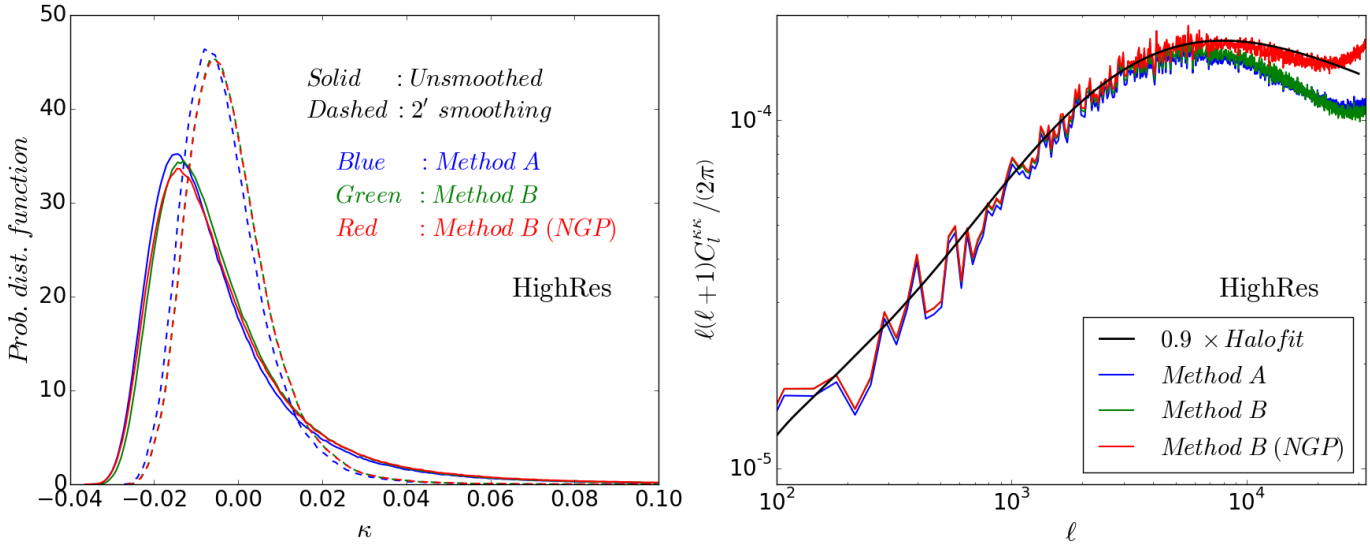

FIG. 11. Comparison of integration methods A, B and B (NGP). Method A and Method B make use of Eq. (38), whereas Method B (NGP) uses Eq. (37). The left panel shows the PDF obtained from unsmoothed and smoothed maps for the three methods, as labelled. The right panel shows the convergence power spectrum of (unsmoothed) maps obtained with the three integrations methods. In this panel, the Halofit curve was scaled down to facilitate comparisons between the shape of the curves at high-ℓ. All these results correspond to the same realization of a HighRes tile.

part of the figure, whose lensing map reveals the presence of several κ peaks inside R_{200}. Similarly, the first halo from the left in the lower part of the figure shows an offset between its concentrated lensing peak and the center of the halo (center of the dashed yellow circle).

Figure 12 shows also that the contribution from foreground and background structures barely modifies the lensing profiles in the inner regions, where the κ values are larger. This is indicative that the rays that crossed the inner regions of the ten haloes did not cross the inner regions of other haloes along their trajectories, which would have otherwise induced a difference between the blue and green lines at small radii [72–74]. At larger angular scales, the amplitude of the convergence decreases, which makes the lensing signal there more sensitive to the influence of intervening matter. Illustrative of this situation is the first halo from the left in the upper part of the figure. For this halo, rays that propagated through its outskirts picked up the lensing signal from matter that surrounds the halo, but seemed to have travelled through mostly underdense regions ($\kappa < 0$) in the rest of their trajectory from the source to the observer (which is why the blue dots are below the green ones).

A more thorough study of halo lensing using our code could involve applying more strict criteria to select haloes (based on their relaxed state, shape, substructure, etc), choosing different points around which to evaluate profiles (halo center, halo density peaks, κ peaks) or studying the average lensing profiles of a stack of haloes [75, 76]. Our several tile realizations can also be used in studies of the contamination to the lensing signal along the line of sight. The latter may not be a critical source of systematics in cluster lensing related work, but that may not necessarily be the case for voids [77–80], which have an intrinsically smaller lensing amplitude. We leave these investigations for future work.

VII. SUMMARY AND OUTLOOK

A. Summary

We have presented a ray tracing code to compute integrated cosmological observables (cf. Eq. 1) that runs on the fly in AMR N-body simulations. Our algorithm is based on the original ideas of Refs. [36, 37], but we implemented it here in the efficiently parallelised `RAMSES` AMR N-body code, which makes it possible to reach the resolution levels that are required by current and future observational surveys. The routines we described in this paper move the rays on a cell-by-cell basis, from some source redshift until an observer at redshift zero. The ray initialization routines (cf. Sec. III E) can self-consistently handle cases where a light bundle is initialized inside the simulation box or at its faces. This ensures that one does not need to simulate a box that is large enough to encompass the whole lightcone but can, instead, "tile" several smaller boxes and let the rays move from one to the other (cf. Sec. III B). The integral along the whole line of sight is obtained by summing up the contribution from each crossed cell (cf. Eq. (11)). The latter can be performed analytically either by treating the field as constant inside each cell (called NGP field, Eq. (13)) or by reconstructing it via trilinear interpolation from the field values at cell vertices (cf. Eq. (18)). The default `RAMSES` code does not evaluate the fields at cell vertices, but at cell centers. For ease of our integration routines (namely those which employ Eq. (18)), we designed an interpolation scheme from cell centers to cell vertices, which

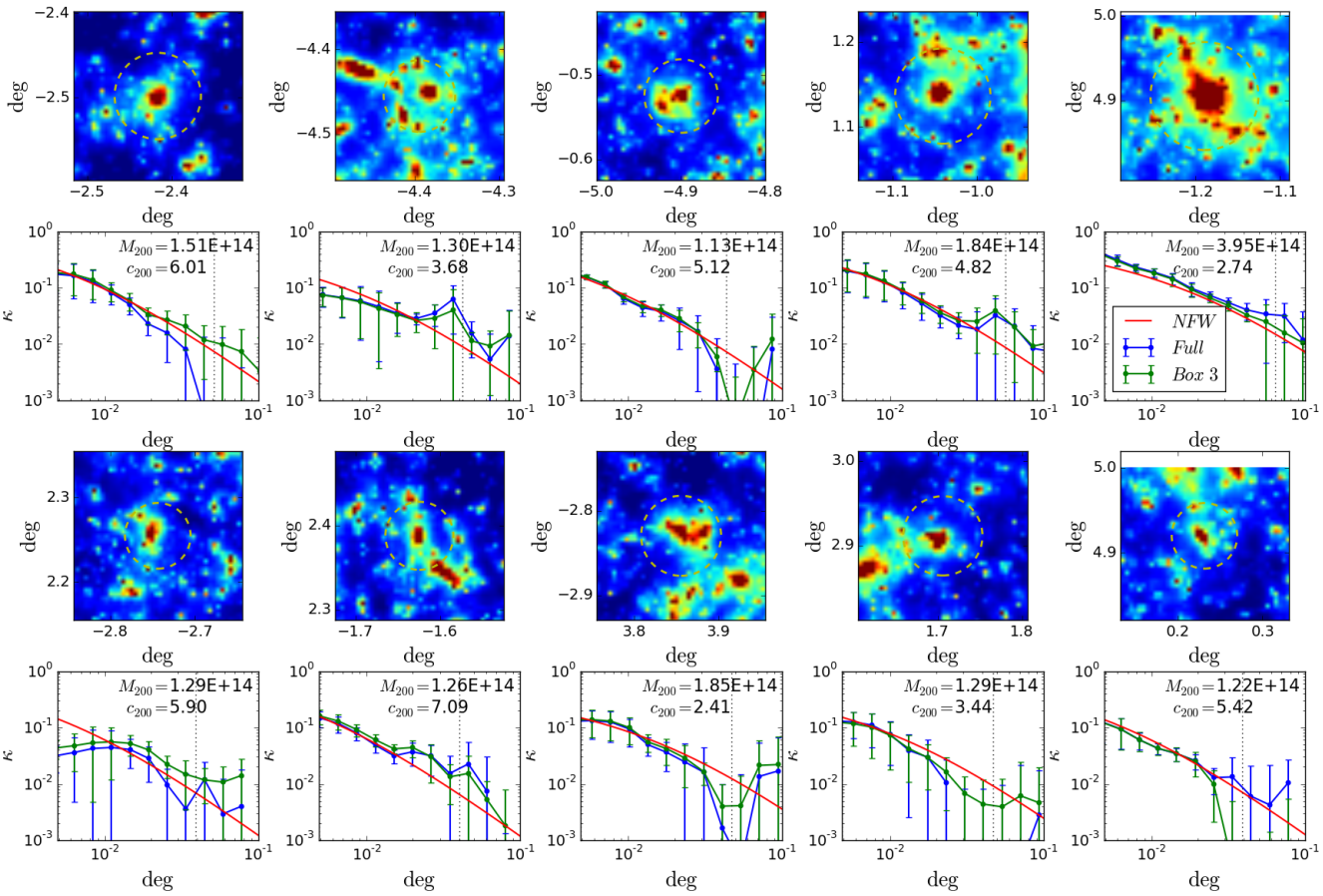

FIG. 12. Spherically averaged lensing signal around individual dark matter haloes. The color maps show the κ field zoomed into a 0.2×0.2 deg^2 region around the center of mass of haloes found at $z = 0.5$ in the middle box of a HighRes tile. This box, called Box 3, is the one in which the redshift interval of the ray integration brackets $z = 0.5$. The dashed yellow lines depict the angular size of the haloes, R_{200}. The ten haloes shown are those which lie inside the lightcone geometry and have $M_{200} > 10^{14}\ M_\odot/h$, $c/a > 0.55$, $b/a > 0.75$ and $f_{\rm sub} < 0.1$. The panels below each color map show the corresponding spherically averaged κ profiles computed around the halo's center of mass taking into account the full lensing signal of the tile (blue) or just the contribution from Box 3 (green), where the haloes were found (the color maps show the full lensing signal, and not just the contribution from Box 3). The points are obtained by taking the mean value of κ at a number of points that sample a circle of a given angular scale around the cluster center. The errorbars show the standard deviation around that mean. The solid red lines show the analytical NFW prediction computed using the halo mass and concentration values (upper right part of each panel) determined by the `Rockstar` halo finder. The vertical dashed lines indicate the angular size of the haloes.

we used to ensure that the reconstructed fields from trilinear interpolation vary continuously when crossing cell boundaries and are mass conserving (cf. Sec. III H).

Since it runs on the fly in the N-body simulation, our code can produce maps of the integrated observables without requiring large amounts of data (or even any) to be stored and further post-processed for ray tracing. Furthermore, our code takes full advantage of the time and spatial resolution available in the N-body run, which is not the case in standard ray tracing numerical studies. These constitute two of the main improvements of our code over more conventional ray tracing methods.

We have tested our ray tracing implementation by applying it to gravitational lensing. We have explained how the lensing convergence, κ, in our code can be computed in two different ways. The first one involves direct integration of the two-dimensional transverse Laplacian of the lensing potential (which we call Method B, Eq. (34)), whereas the second makes use of the three-dimensional Poisson equation to relate the transverse derivatives of the potential to the density field and a series of other integral and surface terms that depend on the radial gradient of the potential (called Method A, Eq. (43)). These two methods make use of Eq. (18) (or Eq. (38)). We have also computed the lensing signal with a method we called Method B (NGP), which is the same as Method B but using Eq. (13) (or Eq. (37)) instead. The lensing shear field can be obtained by integrating directly the corresponding combination of second derivatives of the potential (cf. Eqs. (35) and (36)) or indirecly via the κ result from Method A using Eq. (46). We have tested the numerical solutions of our code by comparing them with the solutions from an independent numerical integrator for lensing through

a fixed Gaussian potential (cf. Sec. V).

As an illustration of the application of our code, we have used it to perform cosmological simulations of weak gravitational lensing. We traced $N_{\rm ray} = 2048^2$ rays in a $10 \times 10\,{\rm deg}^2$ field of view from $z_s = 1$. We simulated a ΛCDM model on five boxes with size $L = 512\,{\rm Mpc}/h$ to encompass that lightcone. We considered two particle resolutions: $N_p = 1024^3$ (HighRes) and $N_p = 512^3$ (LowRes). For each resolution setting, we simulated each of the five boxes in the tile using five realizations of the initial conditions. In total, this allows us to build $5^5 = 3125$ different lensing maps. We have analysed 1- and 2-point statistics of the κ maps as well as the lensing profiles around dark matter haloes. Our weak lensing results can be briefly summarised as follows:

• The sets of κ maps constructed with Method A from both particle resolutions agree very well in their PDF results (Fig. 8). There are some expected differences induced by the difference in resolution. These disappear after the maps are smoothed on a few arcmin scales, which is what is typically done in observational studies (cf. Sec. VI B). The PDF results obtained with method B and method B (NGP) are also in very good agreement with method A (cf. Fig. 11), which constitutes a validity check of the code implementation.

• The κ power spectrum results obtained from the HighRes and LowRes tiles with method A are in good agreement with each other and with the semi-analytical prediction computed in the Halofit formalism for $\ell \lesssim 2 \times 10^3$ (cf. Fig. 9). For larger values of ℓ (small angular scales), our method A results exhibit a suppression of power relative to the Halofit result. This can be attributed to the resolution limit of our simulations and to the adaptive smoothing effects that follow from interpolating the field values from cell centers to cell vertices (cf. the discussion at the end of Sec. III H). The results from method B and method B (NGP) are also in good agreement with power spectrum obtained with method A. The lack of small scale power in method B (NGP) is not as pronounced as in the other two cases, which is related to the fact that method B (NGP) is not affected by the adaptive smoothing caused by the reconstruction of the fields at cell vertices.

• Our κ maps show amplitude peaks at the angular positions of haloes found in the box that is at the middle of the tile. This constitutes a trivial (but successful) validation test of our results (cf. Fig. 12). Furthermore, the spherically averaged κ profiles in our maps exhibit the expected level of agreement with the analytic prediction for NFW haloes computed using the mass and concentration inferred from the three-dimensional particle distribution.

B. Future code developments

We comment below on a list of possible ways to go beyond the current implementation of the code.

Beyond tiling of simulation boxes: The cosmological weak lensing results presented in this paper were obtained by running one N-body simulation for each box that makes up a tile (although not all these simulations have to reach $z = 0$). We note, however, that this is not necessary. For instance, when the light bundle leaves a simulation box, instead of terminating that simulation and starting another box which is closer to the observer, one can keep using the same box, but initializing another bundle of rays for an observer at a different location. This process of changing the position of the observer can continue until all rays reach $z = 0$. The consecutive positions of the observer can be chosen to minimize the repetition of structures, e.g., by having the rays crossing regions of the box that had not been previously crossed or regions that had been crossed but in different directions. Hence, if one needs $N_{\rm box}$ to encompass a given lightcone, then for $N_{\rm sim}$ sets of initial conditions one has $N_{\rm box} \times N_{\rm sim}$ map *portions*[26] that can be combined to build map realizations. If one uses each portion only once, then one can build $N_{\rm sim}$ fully independent maps. This number increases if different map realizations (no longer fully independent) can share some map portions. The shared portions can be appropriately chosen to minimize the effects of structure repetition (e.g. repeat the portions close to the observer or to the sources, where the lensing kernel is small).

Beyond first order approximations for lensing: The ray tracing implementation for lensing that we presented in this paper neglected the effects from second and higher order perturbations, which include the well known Born correction and lens-lens coupling (see Ref. [53] for a thorough account of higher order corrections in lensing). As a first step to go beyond this, the effects from these higher order terms can be added to our code by following the strategy outlined in Appendices A1 and A2 of Ref. [37]. In particular, the effects of lens-lens coupling (and other nonlinear couplings) can be included in a perturbative manner along straight trajectories, and in a way that still allows the relevant integrals to be computed analytically. The high line-of-sight resolution of our code can prove useful in clarifying the importance of first-order approximations in lensing studies [81, 82]. To go beyond the Born approximation one can appropriately deflect the rays every time they cross a cell. The integral that determines the deflection angle $\vec{\alpha}$ inside a cell is evaluated analogously to any other integral in the code (e.g. that of κ). For these higher-order studies (specially beyond the Born approximation), some difficulties may arise as it is no longer certain that the rays all reach the observer at $z = 0$. The significance of these issues and possible ways around it (e.g. construction of lensing maps using only rays that are sufficiently close to the observer or running the simulation backwards in time) is the subject of ongoing work.

Ray-Ramses also as a post-processing tool: The application of our code is not restricted to ray-tracing simulations on the fly. Given a number of particle snapshots from some N-body simulation, `RAMSES` can construct the appropriate AMR grid structure, on which our routines can compute the integrals. Note that this does not involve projections into planes and the calculation will remain a three-dimensional one. The ray-tracing algorithm remains essentially unchanged, except

[26] We call a map portion the contribution coming from each single box crossing.

for the fact that the particle distribution is now static, and hence, the ray trajectories are not constrained by the size of code time steps (cf. Sec. III F 2). This can also facilitate the implementation of non-straight ray trajectories, since rays can be traced away from the observer location. These developments are currently ongoing.

All-sky lensing simulations: In this paper, we have restricted our analysis for fields of view that are small enough for the flat-sky approximation to hold. However, in studies where the light source is the CMB (ISW, SZ, CMB lensing), the field of view is normally the full celestial sphere (up to masked areas), in which case the flat-sky approximation naturally breaks down. Other surveys such as LSST [83] and Euclid [84] are also expected to cover ever larger fractions of the sky for lensing, which motivates extending our code for larger fields of view. This can be done with a few steps. Perhaps the most involved one is related to the initialization of the rays, which would benefit from using the `HEALPix` [51] pixelation on the sphere. The initial directions and positions can be generated elsewhere and stored in a file that is read by our code when it is time to initialize the rays. Other steps should involve a careful determination of the box sizes used in the tiles and the location of the observer. We note that the routines that control the ray trajectories and integration would remain as presented in this paper.

Source redshift distribution for lensing: Although this was not tested in this paper, we note that it is possible to initialize the different rays at different source redshifts. For instance, it is straightforward to sample values of z_s across the rays using the source distribution for some observational survey and only let each ray start the integration once the simulations have reached the value of z_s assigned to it. We note also that it is possible to design ways to compute the lensing signal for different source redshift distributions without having to rerun the code. Consider, for instance, two source redshifts, $z_s^a < z_s^b$, with comoving distance $\chi_s^a < \chi_s^b$. Taking the case of lensing applications as an example, the signal associated with z_s^b is given by the integral

$$\begin{aligned} I &= \frac{1}{c^2}\int_0^{\chi_s^b} \chi Q \mathrm{d}\chi - \frac{1}{c^2\chi_s^b}\int_0^{\chi_s^b} \chi^2 Q \mathrm{d}\chi \\ &= \frac{1}{c^2}\mathcal{A} - \frac{1}{c^2\chi_s^b}\mathcal{B}, \end{aligned} \tag{49}$$

where Q is the desired combination of second angular derivatives of the lensing potential and the second equality serves to define the integrals $\mathcal{A}$ and $\mathcal{B}$. The above equation is the same as the equations solved in this paper, but written in this way it becomes clearer that χ_s^b appears as a multiplicative term and as the limit of integration, but not inside the integrand. The integrals $\mathcal{A}$ and $\mathcal{B}$ are in the form of Eq. (10) and so they can be solved using our routines. Focusing on the case of $\mathcal{A}$, we can decompose its calculation as

$$\mathcal{A} = -\int_{\chi_s^b}^{\chi_s^a} \chi Q \mathrm{d}\chi - \int_{\chi_s^a}^{0} \chi Q \mathrm{d}\chi, \tag{50}$$

where we have flipped the integration ranges, just to emphasize that in our code, the integrations are done from the source towards to observer. Hence, given the value of $\mathcal{A}$ computed for χ_s^b, we can get its value for χ_s^a, by subtracting the first term on the right-hand side of Eq. (50). This term can be obtained by letting the code output the convergence calculation accumulated for each ray from z_s^b to z_s^a. The same considerations hold for the integral $\mathcal{B}$. The final step to get the value of I in Eq. (49) for z_s^a is to replace χ_s^b by χ_s^a in the second term on the right-hand side. This reasoning can be generalized to more source redshift values. This involves having to output the accumulated integrals at a number of redshifts within some range where one expects source galaxies to exist. Note, however, that this is not very demanding from a data storage point of view since the outputted lensing maps are relatively light (compared with particle snapshots from simulations, for instance). A scheme such as this can have interesting applications is assessing fairly quickly the impact of different source redshift distributions on the lensing signal.

C. Code applications

To conclude our discussion, we comment briefly on a number of possible applications of our code.

Baryonic effects: The inclusion of baryon physics is relatively straightforward as it involves simply turning on any hydrodynamical modules that are already existing or that can be added to the `RAMSES` code. The degree of complexity of such baryon physics recipes would depend on the exact application in mind. For instance, Refs. [85–87] investigate the effects of gas cooling, stellar feedback, AGN feedback, and others on weak lensing observables. Still in the context of hydrodynamical simulations, our code can also be used in studies of the SZ effect. In practice, this would amount to associating the quantity Q in Eq. (1) with $n_e T$ for thermal SZ and $n_e v_b$ for kinetic SZ effects, where n_e is the electron gas density, T its temperature and v_b its bulk velocity.

ISW simulations: One of the possible applications of our code is in studies of the ISW effect. This can be done by setting Q in Eq. (1) to the physical time derivative of the lensing potential $\dot{\Phi}_{\rm len}$. The latter can be computed implicitly in each cell via finite-differencing using values of the potential at the current ($\Phi_{\rm curr}$) and previous ($\Phi_{\rm prev}$) time steps: $\dot{\Phi} = (\Phi_{\rm curr} - \Phi_{\rm prev})/\Delta t$, where Δt is the time step interval. Our numerical implementation is particularly suited for ISW studies since it allows to directly compute the time-derivative of the potential on all scales, rather than making use of the continuity equation to relate the velocity field in the simulations to $\dot{\Phi}$. The latter approach is what is done in conventional ISW studies [35, 39]. In a future work, we plan to use our ray tracing code to study the ISW effect, particularly its impact on nonlinear scales (also known as the Rees-Sciama effect [88]).

Modified gravity: An interesting application of our ray tracing code can be in the context of theories of modified gravity (see e.g. [89–91] for reviews). This is straightforward in practice, as it amounts to installing the routines presented in this paper for `RAMSES` into codes such as `ECOSMOG` [92–95] and `ISIS` [96], which are themselves also modified versions of `RAMSES`. For those theories that modify directly the lens-

ing potential (e.g. Galileon [97–104], Nonlocal [105–107], K-mouflage [108, 109] gravity, etc.) the time and spatial resolutions along the line of sight have particular importance because of the time evolution of the modified gravity effects. Moreover, appealing theories of modified gravity usually have screening mechanisms to suppress the modifications to gravity on small scales (like in the Solar System), which follow from nonlinearities in the equations that govern the potential. The nonlinearity implies that the superposition principle does not hold, which means that it is not straightforward to use the multiple lens-plane approximation to study lensing in these theories (in the recent work of Ref. [110], the authors use the multiple lens-plane approximation, but focus on models that do not modify the lensing potential directly). Our code can therefore also be seen as a platform where the lensing effects of these theories of gravity can be studied self-consistently.

Other applications: In the context of lensing, other applications of our code may include the study of cosmic flexions [111–115], which are lensing effects sensitive to third spatial derivatives of the lensing potential, as well as the lensing effects associated with vector and tensor perturbations of the metric [116, 117]. The ray-tracing machinery that we have installed in `RAMSES` may also serve as a backbone to develop a code for radiative transfer and cosmic reionization studies [46–50].

In conclusion, the code we presented in this paper provides a different way to compute integrated observables that is, in general, subject to fewer approximations compared to conventional ray tracing methods. One can also argue that it is more practical to use, in the sense that the calculations are done on the fly in the simulation and not at post-processing stages. We believe that works performed with this code can provide a valuable set of results that could complement those obtained with other methods. This should yield a more robust theoretical understanding of the physical processes that determine a number of integrated cosmological observables, which could help to plan better current and future observational missions. In the future, we plan to widen up the range of applications of `Ray-Ramses` and make the code publicly available to the research community.

ACKNOWLEDGMENTS

We thank Romain Teyssier for useful clarifications about `RAMSES` and Xiangkun Liu for helpful numerical tests. We also thank Carlton Baugh, Martin Feix, Lindsay King, Richard Massey, Fabian Schmidt, Gongbo Zhao and Hongsheng Zhao for encouraging this project and useful comments and discussions, and Lydia Heck for invaluable numerical support. This work was supported by the Science and Technology Facilities Council [grant number ST/L00075X/1]. This work used the DiRAC Data Centric system at Durham University, operated by the Institute for Computational Cosmology on behalf of the STFC DiRAC HPC Facility (www.dirac.ac.uk). This equipment was funded by BIS National E-infrastructure capital grant ST/K00042X/1, STFC capital grant ST/H008519/1, and STFC DiRAC Operations grant ST/K003267/1 and Durham University. DiRAC is part of the National E-Infrastructure. AB thanks the support from FCT-Portugal through grant SFRH/BD/75791/2011 during part of this work. CLL acknowledges support from STFC consolidated grant ST/L00075X/1. SB is supported by STFC through grant ST/K501979/1. BL acknowledges support by the UK STFC Consolidated Grant No. ST/L00075X/1 and No. RF040335.

Appendix A: The Dandelin sphere algorithm in the ray initialization

In this appendix, we describe with more detail the algorithm that is used in the initialization routines to narrow down which ray IDs can be physically inside a given grid. Recall that the initialization can always be performed by looping over all grids, and for each, looping over all rays and checking whether they lie in the volume covered by the grid. The goal of this algorithm is to filter out the number of rays to loop for each grid. This is a nontrivial exercise because the mesh structure is cubic and the ray positions are described in spherical coordinates.

The first step consists of determining the smallest sphere that contains the cubic grid. This is shown in the left panel of Fig. 13, where the sphere is centered at point S, which is also the center of the grid (not shown). Then, one identifies the narrowest cone from the observer O that contains this sphere, as illustrated also in the figure. Naturally, those rays whose angular positions do not fall within the opening angles of the cone cannot be in that grid. The problem is then reduced to finding the rays that are inside the cone. To do so, we choose to work within a plane that is perpendicular to the z-axis.

Consider the elliptical cross section obtained by *cutting* through the cone with a plane perpendicular to the z-axis and tangential to the lowest z-coordinate of the sphere (point F in Fig. 13). In this case, the sphere is called a Dandelin sphere and has the property that point F is one of the two foci of the ellipse, whose center is point E. Then the question reduces to determining which rays lie inside the ellipse, which is parallel to the x-y plane. In the right panel of Fig. 13, we show this ellipse projected onto the x-y plane. Points E, F, and O lie on a line that makes an angle φ with the x-axis. In our algorithm, one finds the smallest rectangle that contains the ellipse and whose sides are parallel to the x- and y-axes. As shown in the figure, the rectangle is tangential to the ellipse at four points, which are, repectively, the maximum and minimum x-coordinates ($x_{\rm max}$ and $x_{\rm min}$) and y-coordinates ($y_{\rm max}$ and $y_{\rm min}$) of the ellipse. The rays that lie inside the sphere must have $x_{\rm ray} \in [x_{\rm min}, x_{\rm max}]$ and $y_{\rm ray} \in [y_{\rm min}, y_{\rm max}]$. Given these ranges, it is then straightforward to use Eqs. (3) and (4) to determine which `rayid` values need to be checked (note that x, y are related to θ_x, θ_y in Sec. III E).

Using straightforward geometry, one can determine the boundaries of the rectangle as

$$x_{\rm min} = x_F - \frac{a(1-e^2)\cos\xi}{1-e\cos(\varphi-\xi)},$$

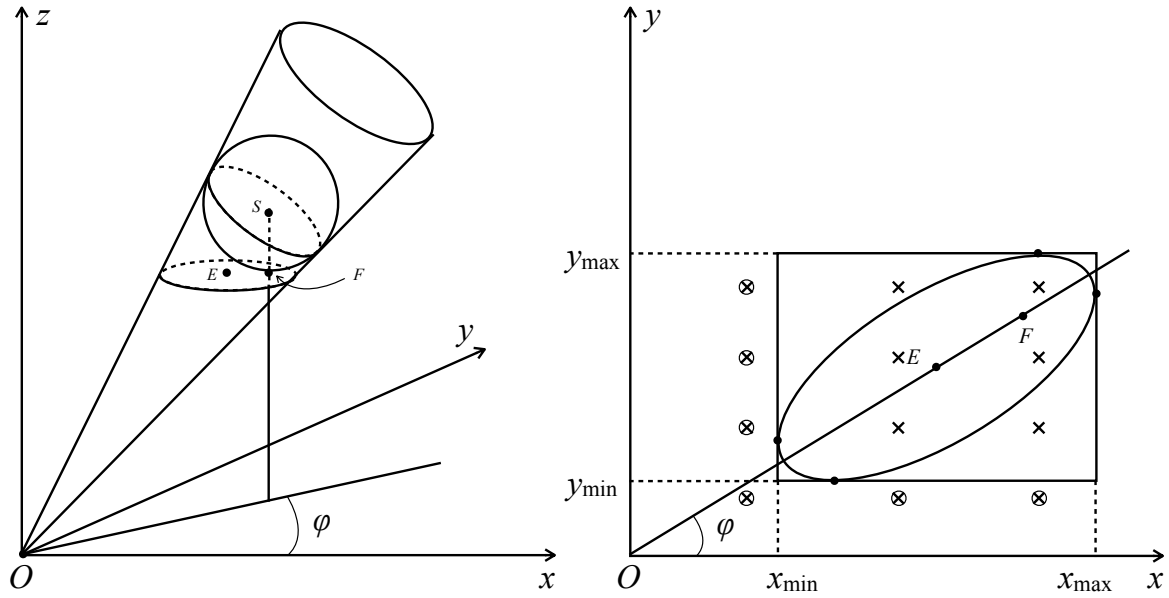

FIG. 13. Dandelin sphere construction used in the ray initialization routines. The left diagram shows the narrowest cone from the observer that contains a sphere, which is the smallest sphere that contains a given cubic grid. The point S is the center of the sphere and of the grid (the grid is not shown to keep the plot less busy). In the left diagram, poins E and F are, respectively, the center and one of the foci of the ellipse that results from *cutting* the cone perpendicularly to the z axis and through the point of lowest z-coordinate of the sphere (which is point F). The right diagram shows this ellipse projected onto the x-y plane. The crosses illustrate the positions of rays crossing the x-y plane. Those that do not fall inside the rectangle that encompasses the ellipse (circled crosses) can be left out from the loops in the initialization routine.

$$
\begin{aligned}
x_{\max} &= x_F + \frac{a(1-e^2)\cos\xi}{1+e\cos(\varphi-\xi)},\\
y_{\min} &= y_F - \frac{a(1-e^2)\sin\zeta}{1-e\cos(\varphi-\zeta)},\\
y_{\max} &= y_F + \frac{a(1-e^2)\sin\zeta}{1+e\cos(\varphi-\zeta)},
\end{aligned}
\tag{A1}
$$

where (x_F, y_F) is the coordinate of the focus of the ellipse in the (projected) x-y plane, a, e are respectively the length of the semi-long axis and the eccentricity of the ellipse, and

$$
\begin{aligned}
\xi &= \sin^{-1}(e\sin\varphi),\\
\zeta &= \cos^{-1}(e\cos\varphi).
\end{aligned}
\tag{A2}
$$

To exemplify, we show in the right panel of Fig. 13 a number of rays (crosses) that cross the plane containing the ellipse (note that all rays are equally spaced in x and y). The rays that fall outside the rectangles are circled, which means that they certainly do not cross the grid, and as a result, do not need to be looped over. For most applications, for each grid, the majority of the rays lies outside of the rectangle, which is why this algorithm speeds up substantially the initialization of the ray data structure.

Appendix B: d_N expressions in Eq. (17)

The d_N coefficients in Eq. (17) that are used in the trilinear interpolation inside cells are given by

$$
d_1 = \alpha_1 + \frac{1}{h}\Big(\alpha_2 a + \alpha_3 b + \alpha_4 c\Big) + \frac{1}{h^2}\Big(\alpha_5 ab + \alpha_6 bc + \alpha_7 ac\Big) + \frac{1}{h^3}\alpha_8 abc \tag{B1}
$$

$$
\begin{aligned}
d_2 = {} & \frac{1}{h}\Big(\alpha_2 \sin\theta\cos\varphi + \alpha_3 \sin\theta\sin\varphi + \alpha_4\cos\theta\Big),\\
& + \frac{1}{h^2}\Big(\,[\alpha_6 c + \alpha_5 a]\sin\theta\sin\varphi + [\alpha_7 c + \alpha_5 b]\sin\theta\cos\varphi + [\alpha_7 a + \alpha_6 b]\cos\theta\Big)\\
& + \frac{1}{h^3}\Big(\alpha_8 bc\sin\theta\cos\varphi + \alpha_8 ab\cos\theta + \alpha_8 ac\sin\theta\sin\varphi\Big),
\end{aligned}
\tag{B2}
$$

$$
d_3 = \frac{1}{h^2}\Big(\alpha_5 \sin^2\theta\cos\varphi\sin\varphi + \alpha_7 \sin\theta\cos\varphi\cos\theta + \alpha_6 \sin\theta\sin\varphi\cos\theta\Big)
$$

$$+\frac{1}{h^3}\Big(\alpha_8 a\sin\theta\sin\varphi\cos\theta+\alpha_8 b\sin\theta\cos\varphi\cos\theta+\alpha_8 c\sin^2\theta\cos\varphi\sin\varphi\Big), \tag{B3}$$

$$d_4=\frac{\alpha_8}{h^3}\Big(\sin^2\theta\cos\theta\cos\varphi\sin\varphi\Big), \tag{B4}$$

where h is the cell size.

[1] D. G. York et al, AJ **120**, 1579 (Sep. 2000), astro-ph/0006396.
[2] M. Colless et al, ArXiv Astrophysics e-prints(Jun. 2003), astro-ph/0306581.
[3] K. S. Dawson et al, AJ **145**, 10 (Jan. 2013), arXiv:1208.0022.
[4] S. Cole et al, MNRAS **362**, 505 (Sep. 2005), astro-ph/0501174.
[5] D. J. Eisenstein et al, APJ **633**, 560 (Nov. 2005), astro-ph/0501171.
[6] W. J. Percival and M. White, MNRAS **393**, 297 (Feb. 2009), arXiv:0808.0003.
[7] L. Guzzo et al, NAT **451**, 541 (Jan. 2008), arXiv:0802.1944.
[8] A. G. Sánchez et al, MNRAS **440**, 2692 (May 2014), arXiv:1312.4854.
[9] R. A. Sunyaev and Y. B. Zeldovich, Comments on Astrophysics and Space Physics **4**, 173 (Nov. 1972).
[10] R. A. Sunyaev and I. B. Zeldovich, MNRAS **190**, 413 (Feb. 1980).
[11] J. E. Carlstrom, G. P. Holder, and E. D. Reese, ARAA **40**, 643 (2002), astro-ph/0208192.
[12] N. Hand et al, Physical Review Letters **109**, 041101 (Jul. 2012), arXiv:1203.4219.
[13] R. K. Sachs and A. M. Wolfe, APJ **147**, 73 (Jan. 1967).
[14] A. Cooray, Phys. Rev. D **65**, 103510 (May 2002), astro-ph/0112408.
[15] S. Ho, C. Hirata, N. Padmanabhan, U. Seljak, and N. Bahcall, PRD **78**, 043519 (Aug. 2008), arXiv:0801.0642.
[16] B. R. Granett, M. C. Neyrinck, and I. Szapudi, APJL **683**, L99 (Aug. 2008), arXiv:0805.3695.
[17] M. Bartelmann and P. Schneider, PhysRep **340**, 291 (Jan. 2001), astro-ph/9912508.
[18] A. Refregier, ARAA **41**, 645 (2003), astro-ph/0307212.
[19] M. Bartelmann, Classical and Quantum Gravity **27**, 233001 (Dec. 2010), arXiv:1010.3829.
[20] M. Kilbinger, Reports on Progress in Physics **78**, 086901 (Jul. 2015), arXiv:1411.0115.
[21] B. Jain, U. Seljak, and S. White, APJ **530**, 547 (Feb. 2000), astro-ph/9901191.
[22] C. Vale and M. White, APJ **592**, 699 (Aug. 2003), astro-ph/0303555.
[23] S. Das and P. Bode, APJ **682**, 1 (Jul. 2008), arXiv:0711.3793.
[24] P. Fosalba, E. Gaztañaga, F. J. Castander, and M. Manera, MNRAS **391**, 435 (Nov. 2008), arXiv:0711.1540.
[25] R. Teyssier, S. Pires, S. Prunet, D. Aubert, C. Pichon, A. Amara, K. Benabed, S. Colombi, A. Refregier, and J.-L. Starck, AAP **497**, 335 (Apr. 2009), arXiv:0807.3651.
[26] C. Carbone, V. Springel, C. Baccigalupi, M. Bartelmann, and S. Matarrese, MNRAS **388**, 1618 (Aug. 2008), arXiv:0711.2655.
[27] S. Hilbert, J. Hartlap, S. D. M. White, and P. Schneider, Astron. Astrophys. **499**, 31 (2009), arXiv:0809.5035 [astro-ph].
[28] M. Sato, T. Hamana, R. Takahashi, M. Takada, N. Yoshida, T. Matsubara, and N. Sugiyama, APJ **701**, 945 (Aug. 2009), arXiv:0906.2237 [astro-ph.CO].
[29] M. Killedar, P. D. Lasky, G. F. Lewis, and C. J. Fluke, MNRAS **420**, 155 (Feb. 2012), arXiv:1110.4894.
[30] M. R. Becker, MNRAS **435**, 115 (Oct. 2013).
[31] R. B. Metcalf and M. Petkova, MNRAS **445**, 1942 (Dec. 2014), arXiv:1312.1128.
[32] M. Petkova, R. B. Metcalf, and C. Giocoli, MNRAS **445**, 1954 (Dec. 2014), arXiv:1312.1536.
[33] J. Liu, A. Petri, Z. Haiman, L. Hui, J. M. Kratochvil, and M. May, PRD **91**, 063507 (Mar. 2015), arXiv:1412.0757.
[34] C. Giocoli, E. Jullo, R. B. Metcalf, S. de la Torre, G. Yepes, F. Prada, J. Comparat, S. Goettlober, A. Kyplin, J.-P. Kneib, M. Petkova, H. Shan, and N. Tessore, ArXiv e-prints(Nov. 2015), arXiv:1511.08211.
[35] Y.-C. Cai, S. Cole, A. Jenkins, and C. Frenk, MNRAS **396**, 772 (Jun. 2009), arXiv:0809.4488.
[36] M. White and W. Hu, APJ **537**, 1 (Jul. 2000), astro-ph/9909165.
[37] B. Li, L. J. King, G.-B. Zhao, and H. Zhao, MNRAS **415**, 881 (Jul. 2011), arXiv:1012.1625.
[38] R. Teyssier, AAP **385**, 337 (Apr. 2002), astro-ph/0111367.
[39] Y.-C. Cai, S. Cole, A. Jenkins, and C. S. Frenk, MNRAS **407**, 201 (Sep. 2010), arXiv:1003.0974.
[40] W. A. Watson, J. M. Diego, S. Gottlöber, I. T. Iliev, A. Knebe, E. Martínez-González, G. Yepes, R. B. Barreiro, J. González-Nuevo, S. Hotchkiss, A. Marcos-Caballero, S. Nadathur, and P. Vielva, MNRAS **438**, 412 (Feb. 2014), arXiv:1307.1712.
[41] A. C. da Silva, D. Barbosa, A. R. Liddle, and P. A. Thomas, MNRAS **317**, 37 (Sep. 2000), astro-ph/9907224.
[42] A. C. da Silva, D. Barbosa, A. R. Liddle, and P. A. Thomas, MNRAS **326**, 155 (Sep. 2001), astro-ph/0011187.
[43] V. Springel, M. White, and L. Hernquist, APJ **549**, 681 (Mar. 2001), astro-ph/0008133.
[44] I. G. McCarthy, A. M. C. Le Brun, J. Schaye, and G. P. Holder, MNRAS **440**, 3645 (Jun. 2014), arXiv:1312.5341.
[45] K. Dolag, E. Komatsu, and R. Sunyaev, ArXiv e-prints(Sep. 2015), arXiv:1509.05134.
[46] I. T. Iliev, B. Ciardi, M. A. Alvarez, A. Maselli, A. Ferrara, N. Y. Gnedin, G. Mellema, T. Nakamoto, M. L. Norman, A. O. Razoumov, E.-J. Rijkhorst, J. Ritzerveld, P. R. Shapiro, H. Susa, M. Umemura, and D. J. Whalen, MNRAS **371**, 1057 (Sep. 2006), astro-ph/0603199.
[47] I. T. Iliev, D. Whalen, G. Mellema, K. Ahn, S. Baek, N. Y. Gnedin, A. V. Kravtsov, M. Norman, M. Raicevic, D. R. Reynolds, D. Sato, P. R. Shapiro, B. Semelin, J. Smidt, H. Susa, T. Theuns, and M. Umemura, MNRAS **400**, 1283 (Dec. 2009), arXiv:0905.2920.
[48] J. H. Wise and T. Abel, MNRAS **414**, 3458 (Jul. 2011), arXiv:1012.2865 [astro-ph.IM].

[49] J. Rosdahl, J. Blaizot, D. Aubert, T. Stranex, and R. Teyssier, MNRAS **436**, 2188 (Dec. 2013), arXiv:1304.7126.

[50] G. Altay and T. Theuns, MNRAS **434**, 748 (Sep. 2013), arXiv:1304.4235 [astro-ph.CO].

[51] K. M. Górski, E. Hivon, A. J. Banday, B. D. Wandelt, F. K. Hansen, M. Reinecke, and M. Bartelmann, APJ **622**, 759 (Apr. 2005), astro-ph/0409513.

[52] P. G. Castro, A. F. Heavens, and T. D. Kitching, PRD **72**, 023516 (Jul. 2005), astro-ph/0503479.

[53] F. Bernardeau, C. Bonvin, and F. Vernizzi, PRD **81**, 083002 (Apr. 2010), arXiv:0911.2244 [astro-ph.CO].

[54] N. Kaiser and G. Squires, APJ **404**, 441 (Feb. 1993).

[55] B. Gough, *GNU Scientific Library Reference Manual - Third Edition*, 3rd ed. (Network Theory Ltd., 2009) ISBN 0954612078, 9780954612078.

[56] Planck Collaboration, P. A. R. Ade, N. Aghanim, C. Armitage-Caplan, M. Arnaud, M. Ashdown, F. Atrio-Barandela, J. Aumont, C. Baccigalupi, A. J. Banday, and et al., AAP **571**, A16 (Nov. 2014), arXiv:1303.5076.

[57] Planck Collaboration, P. A. R. Ade, N. Aghanim, M. Arnaud, M. Ashdown, J. Aumont, C. Baccigalupi, A. J. Banday, R. B. Barreiro, J. G. Bartlett, and et al., ArXiv e-prints(Feb. 2015), arXiv:1502.01589.

[58] S. Prunet, C. Pichon, D. Aubert, D. Pogosyan, R. Teyssier, and S. Gottloeber, APJS **178**, 179 (Oct. 2008), arXiv:0804.3536.

[59] L. Van Waerbeke, J. Benjamin, T. Erben, C. Heymans, H. Hildebrandt, H. Hoekstra, T. D. Kitching, Y. Mellier, L. Miller, J. Coupon, J. Harnois-Déraps, L. Fu, M. Hudson, M. Kilbinger, K. Kuijken, B. Rowe, T. Schrabback, E. Semboloni, S. Vafaei, E. van Uitert, and M. Velander, MNRAS **433**, 3373 (Aug. 2013), arXiv:1303.1806.

[60] V. Vikram et al, PRD **92**, 022006 (Jul. 2015), arXiv:1504.03002.

[61] R. E. Smith, J. A. Peacock, A. Jenkins, S. D. M. White, C. S. Frenk, F. R. Pearce, P. A. Thomas, G. Efstathiou, and H. M. P. Couchman, MNRAS **341**, 1311 (Jun. 2003), astro-ph/0207664.

[62] R. Takahashi, M. Sato, T. Nishimichi, A. Taruya, and M. Oguri, APJ **761**, 152 (Dec. 2012), arXiv:1208.2701.

[63] S. Hilbert, J. Hartlap, S. D. M. White, and P. Schneider, AAP **499**, 31 (May 2009), arXiv:0809.5035.

[64] P. S. Behroozi, R. H. Wechsler, and H.-Y. Wu, APJ **762**, 109 (Jan. 2013), arXiv:1110.4372 [astro-ph.CO].

[65] J. F. Navarro, C. S. Frenk, and S. D. White, Astrophys.J. **490**, 493 (1997), arXiv:astro-ph/9611107 [astro-ph].

[66] C. Oaxaca Wright and T. G. Brainerd, ArXiv Astrophysics e-prints(Aug. 1999), astro-ph/9908213.

[67] M. Bartelmann, AAP **313**, 697 (Sep. 1996), astro-ph/9602053.

[68] K. Umetsu, ArXiv e-prints(Feb. 2010), arXiv:1002.3952 [astro-ph.CO].

[69] Meneghetti et al, APJ **797**, 34 (Dec. 2014), arXiv:1404.1384.

[70] J. Merten et al, APJ **806**, 4 (Jun. 2015), arXiv:1404.1376.

[71] W. Du, Z. Fan, H. Shan, G.-B. Zhao, G. Covone, L. Fu, and J.-P. Kneib, APJ **814**, 120 (Dec. 2015), arXiv:1510.08193.

[72] X. Yang, J. M. Kratochvil, S. Wang, E. A. Lim, Z. Haiman, and M. May, PRD **84**, 043529 (Aug. 2011), arXiv:1109.6333.

[73] T. Hamana, M. Takada, and N. Yoshida, MNRAS **350**, 893 (May 2004), astro-ph/0310607.

[74] Z. Fan, H. Shan, and J. Liu, APJ **719**, 1408 (Aug. 2010), arXiv:1006.5121.

[75] M. Oguri, M. B. Bayliss, H. Dahle, K. Sharon, M. D. Gladders, P. Natarajan, J. F. Hennawi, and B. P. Koester, MNRAS **420**, 3213 (Mar. 2012), arXiv:1109.2594.

[76] M. Oguri and T. Hamana, MNRAS **414**, 1851 (Jul. 2011), arXiv:1101.0650.

[77] P. Melchior, P. M. Sutter, E. S. Sheldon, E. Krause, and B. D. Wandelt, MNRAS **440**, 2922 (Jun. 2014), arXiv:1309.2045.

[78] J. Clampitt and B. Jain, MNRAS **454**, 3357 (Dec. 2015), arXiv:1404.1834.

[79] E. Krause, T.-C. Chang, O. Doré, and K. Umetsu, APJL **762**, L20 (Jan. 2013), arXiv:1210.2446.

[80] Y. Higuchi, M. Oguri, and T. Hamana, MNRAS **432**, 1021 (Jun. 2013), arXiv:1211.5966.

[81] S. Hagstotz, B. M. Schäfer, and P. M. Merkel, ArXiv e-prints(Oct. 2014), arXiv:1410.8452.

[82] M. Calabrese, C. Carbone, G. Fabbian, M. Baldi, and C. Baccigalupi, JCAP **3**, 049 (Mar. 2015), arXiv:1409.7680.

[83] LSST Dark Energy Science Collaboration, ArXiv e-prints(Nov. 2012), arXiv:1211.0310 [astro-ph.CO].

[84] R. Laureijs, J. Amiaux, S. Arduini, J. . Augueres, J. Brinchmann, R. Cole, M. Cropper, C. Dabin, L. Duvet, A. Ealet, and et al., ArXiv e-prints(Oct. 2011), arXiv:1110.3193 [astro-ph.CO].

[85] E. Semboloni, H. Hoekstra, J. Schaye, M. P. van Daalen, and I. G. McCarthy, MNRAS **417**, 2020 (Nov. 2011), arXiv:1105.1075.

[86] I. Mohammed, D. Martizzi, R. Teyssier, and A. Amara, ArXiv e-prints(Oct. 2014), arXiv:1410.6826.

[87] K. Osato, M. Shirasaki, and N. Yoshida, APJ **806**, 186 (Jun. 2015), arXiv:1501.02055.

[88] M. J. Rees and D. W. Sciama, NAT **217**, 511 (Feb. 1968).

[89] T. Clifton, P. G. Ferreira, A. Padilla, and C. Skordis, PHYS-REP **513**, 1 (Mar. 2012), arXiv:1106.2476 [astro-ph.CO].

[90] A. Joyce, B. Jain, J. Khoury, and M. Trodden(2014), arXiv:1407.0059 [astro-ph.CO].

[91] K. Koyama, ArXiv e-prints(Apr. 2015), arXiv:1504.04623.

[92] B. Li, G.-B. Zhao, R. Teyssier, and K. Koyama, JCAP **1**, 051 (Jan. 2012), arXiv:1110.1379 [astro-ph.CO].

[93] B. Li, G.-B. Zhao, and K. Koyama, JCAP **5**, 023 (May 2013), arXiv:1303.0008 [astro-ph.CO].

[94] B. Li, A. Barreira, C. M. Baugh, W. A. Hellwing, K. Koyama, S. Pascoli, and G.-B. Zhao, JCAP **11**, 012 (Nov. 2013), arXiv:1308.3491 [astro-ph.CO].

[95] A. Barreira, S. Bose, and B. Li, ArXiv e-prints(Nov. 2015), arXiv:1511.08200.

[96] C. Llinares, D. F. Mota, and H. A. Winther, AAP **562**, A78 (Feb. 2014), arXiv:1307.6748.

[97] A. Nicolis, R. Rattazzi, and E. Trincherini, PRD **79**, 064036 (Mar. 2009), arXiv:0811.2197 [hep-th].

[98] C. Deffayet, G. Esposito-Farèse, and A. Vikman, PRD **79**, 084003 (Apr. 2009), arXiv:0901.1314 [hep-th].

[99] Y. Park and M. Wyman, PRD **91**, 064012 (Mar. 2015), arXiv:1408.4773.

[100] A. Barreira, B. Li, E. Jennings, J. Merten, L. King, C. M. Baugh, and S. Pascoli, MNRAS **454**, 4085 (Dec. 2015), arXiv:1505.03468.

[101] A. Barreira, M. Cautun, B. Li, C. M. Baugh, and S. Pascoli, JCAP **8**, 028 (Aug. 2015), arXiv:1505.05809.

[102] A. Barreira, B. Li, C. M. Baugh, and S. Pascoli, JCAP **8**, 059 (Aug. 2014), arXiv:1406.0485.

[103] A. Barreira, B. Li, W. A. Hellwing, C. M. Baugh, and S. Pascoli, JCAP **10**, 027 (Oct. 2013), arXiv:1306.3219.

[104] A. Barreira, B. Li, C. M. Baugh, and S. Pascoli, PRD **86**, 124016 (Dec. 2012), arXiv:1208.0600 [astro-ph.CO].

[105] S. Deser and R. P. Woodard, Physical Review Letters **99**, 111301 (Sep. 2007), arXiv:0706.2151.

[106] S. Foffa, M. Maggiore, and E. Mitsou, International Journal of Modern Physics A **29**, 1450116 (Aug. 2014), arXiv:1311.3435 [hep-th].
[107] A. Barreira, B. Li, W. A. Hellwing, C. M. Baugh, and S. Pascoli, JCAP **9**, 031 (Sep. 2014), arXiv:1408.1084.
[108] A. Barreira, P. Brax, S. Clesse, B. Li, and P. Valageas, PRD **91**, 063528 (Mar. 2015), arXiv:1411.5965.
[109] P. Brax and P. Valageas, PRD **90**, 023508 (Jul. 2014), arXiv:1403.5424.
[110] N. Tessore, H. A. Winther, R. B. Metcalf, P. G. Ferreira, and C. Giocoli, JCAP **10**, 036 (Oct. 2015), arXiv:1508.04011.
[111] D. M. Goldberg and D. J. Bacon, APJ **619**, 741 (Feb. 2005), astro-ph/0406376.
[112] D. J. Bacon, D. M. Goldberg, B. T. P. Rowe, and A. N. Taylor, MNRAS **365**, 414 (Jan. 2006), astro-ph/0504478.
[113] Y. Okura, K. Umetsu, and T. Futamase, APJ **660**, 995 (May 2007), astro-ph/0607288.
[114] P. Schneider and X. Er, AAP **485**, 363 (Jul. 2008), arXiv:0709.1003.
[115] B. Rowe, D. Bacon, R. Massey, C. Heymans, B. Häußler, A. Taylor, J. Rhodes, and Y. Mellier, MNRAS **435**, 822 (Oct. 2013), arXiv:1211.0966.
[116] M. Bruni, D. B. Thomas, and D. Wands, PRD **89**, 044010 (Feb. 2014), arXiv:1306.1562 [astro-ph.CO].
[117] D. B. Thomas, M. Bruni, and D. Wands, JCAP **9**, 021 (Sep. 2015), arXiv:1403.4947.