Dynamic zoom simulations: A fast, adaptive algorithm for simulating light-cones

Enrico Garaldi [1]⋆ Matteo Nori [2,3,4] and Marco Baldi [2,3,4]

[1]*Max Planck Institute for Astrophysics, Karl-Schwarzschild-Straße 1, D-85741 Garching, Germany*
[2]*Dipartimento di Fisica e Astronomia, Universitá di Bologna, via Piero Gobetti 93/2, D-40129 Bologna, Italy*
[3]*INAF - Osservatorio di Astrofisica e Scienza dello Spazio, via Piero Gobetti 93/3 1, D-40129 Bologna, Italy*
[4]*INFN - Sezione di Bologna, viale Berti Pichat 6/2, I-40127 Bologna, Italy*

ABSTRACT
The advent of a new generation of large-scale galaxy surveys is pushing cosmological numerical simulations in an uncharted territory. The simultaneous requirements of high resolution and very large volume pose serious technical challenges, due to their computational and data storage demand. In this paper, we present a novel approach dubbed dynamic zoom simulations – or DZS – developed to tackle these issues. Our method is tailored to the production of light-cone outputs from *N*-body numerical simulations, which allow for a more efficient storage and post-processing compared to standard comoving snapshots, and more directly mimic the format of survey data. In DZS, the resolution of the simulation is dynamically decreased outside the light-cone surface, reducing the computational work load, while simultaneously preserving the accuracy inside the light-cone and the large-scale gravitational field. We show that our approach can achieve virtually identical results to traditional simulations at half of the computational cost for our largest box. We also forecast this speedup to increase up to a factor of 5 for larger and/or higher resolution simulations. We assess the accuracy of the numerical integration by comparing pairs of identical simulations run with and without DZS. Deviations in the light-cone halo mass function, in the sky-projected light-cone, and in the 3D matter light-cone always remain below 0.1 per cent. In summary, our results indicate that the DZS technique may provide a highly valuable tool to address the technical challenges that will characterize the next generation of large-scale cosmological simulations.

Key words: methods: numerical – dark matter – large-scale structure of Universe – software: simulations.

1 INTRODUCTION

Numerical simulations play a pivotal role in cosmology and astrophysics. Their importance surged in the last few decades, establishing them as one of the primary investigation tools in these disciplines. However, cosmological observations of ever-growing precision require increasingly accurate – and thus expensive – simulations to be interpreted. Forthcoming instruments like *Euclid* (Laureijs et al. 2011), *WFIRST* (Spergel et al. 2015), the Vera C. Rubin Observatory (formerly known as LSST; Ivezić et al. 2019) and DESI (DESI Collaboration 2016a,b) will provide an unprecedented amount of data, outperforming all previous observational campaigns in terms of both quality and covered volume. Their interpretation therefore demands a new generation of numerical simulations comprising volumes of the universe comparable in size with those covered by such surveys and – simultaneously – of high physical fidelity.

This upcoming new era of numerical simulations faces a series of technical challenges. First and foremost, the total time required for their completion increases more than linearly with both resolution and volume. In addition, the information stored per resolution element is hardly reducible, entailing that the memory requirements of such simulations keep increasing, in contrast with the trend in

high-performance computing facilities to decrease the amount of available memory – both on the machine and the CPU level.

Hence, even on the top-ranked supercomputing infrastructures currently available, the most advanced simulations can easily necessitate of large fractions of the entire supercomputer for very long time. For instance, the TNG50 run of the Illustris-TNG simulation project (which follows gravitational, magnetic, and hydrodynamical interactions; Nelson et al. 2019; Pillepich et al. 2019) required approximately 128 TB of memory for over 90 million CPU-hours. Similarly, the *Euclid* flagship simulation (Potter, Stadel & Teyssier 2017) employed a similar amount of memory for more than 4 million CPU-hours to follow the evolution of a Dark Matter-only universe.

Even if the aforementioned issues were taken care of, the community would be faced with the technical challenge of storing the data produced by such large simulations. The typical approach is to output time slices of the whole simulation volume (i.e. saving the properties of all simulated particles at a given cosmic time in a so-called snapshot) and then, in post-processing, inspect them as a time-series. When trying to mimic the survey view of the Universe (where images of structures at different cosmic times coexist), snapshots are combined in order to create light-cones-like data (e.g. Hollowed 2019). This can be done either with a piecewise-constant approximation or interpolating the particle position between adjacent outputs. In both cases, a large number of time slices are necessary in order to reach the degree of fidelity required by modern

galaxy survey data, with only a small fraction of the particles in each snapshot concurring to the light-cone reconstruction. However, such procedure would require a prohibitively large amount of storing space in the case of the most advanced simulations.

This technical challenge prompted some simulators to abandon the traditional snapshot format in favour of light-cone outputs (pioneered by Evrard et al. 2002, and more recently employed e.g. in Fosalba et al. 2008, in the Euclid flagship simulation presented in Potter et al. 2017, and in the $768 \, h^{-1}$ Mpc run of the simulations in Arnold et al. 2019). In this approach, light-cones are produced on the fly by saving to disc only the particles that are, at any given redshift z, in a thin spherical shell with outer radius equal to the light-cone radius[1] $R_{lc}(z)$, centred on the observer. They represent the only output of the simulation (possibly complemented by few traditional snapshots). This enables simulations to reach a much more fine-grained discretization of the light-cone while saving a large amount of storage space, as only particles belonging to the light-cone are stored.

While tackling the issue of storing and post-processing simulation data, the use of light-cone outputs does not ease the challenging run-time memory and computational power requirements of such massive numerical enterprises. Recently Llinares (2017, L17 hereafter) proposed a new simulation method dubbed the Shrinking Domain Framework (SDF) as a possible solution to these problems. Such approach exploits the fact that the light-cone radius R_{lc} decreases with time and, simultaneously, most of the computational effort is invested at late times, as a consequence of increased clustering of matter. In the SDF, particles outside of the light-cone (i.e. more distant than R_{lc} from the observer) are discarded from the numerical integration, thereby speeding up the simulation since the number of resolution elements to be evolved decreases with time as the light-cone volume shrinks on to the observer. However, the performance improvement obtained with such a radical modification of the simulation structure does not come for free: the price to be paid for this faster algorithm is the loss of periodicity of the system, which is a fundamental assumption for many numerical gravity solvers. As a consequence, the nature of the gravitational solver changes, transforming the Poisson equation into an equation that accounts for the finite information propagation speed.[2] Additionally, density perturbations (and therefore gravitational interactions) on scales larger than R_{lc} are effectively ignored. While the latter would not have any impact inside the light-cone in a fully relativistic treatment of gravity, this is not the case in the Newtonian approximation typically enforced in many cosmological simulation codes.

A totally different approach to the problem of reducing the computational cost of large cosmological simulations has been proposed in Tassev, Zaldarriaga & Eisenstein (2013). Their method, dubbed COmoving Lagrangian Acceleration (COLA), solves the large-scale motion using the second-order Lagrangian perturbation theory (LPT), while the small-scale motion is integrated using a *N*-body code. This approach allows simulators to sacrifice accuracy on small scales by forcing large time-steps to gain orders of magnitude in simulation speed. As a result, the COLA method approximates the large-scale matter distribution very well, but fails to match the internal structure of dark matter haloes, which is crucial for

producing realistic galaxies. Additionally, this method does not allow the treatment of baryons (although it has been proposed to use a gradient-based method to mimic the output of hydrodynamical simulations, at the cost of introducing nuisance parameters, see e.g. Dai, Feng & Seljak 2018). In a further development of the COLA algorithm dubbed spatial COLA (or sCOLA; Tassev et al. 2015), small- and large-scales modes are de-coupled by computing the latter via LPT and superimposing them to the former, whose evolution is solved using an *N*-body integrator. This approach can also be used to tile a number of independent sCOLA simulations to produce a larger synthetic universe (Leclercq et al. 2020), effectively removing any overhead due to inter-task communication. However, it does not reduce the CPU time required to integrate small-scale motion, but rather provides an efficient way to distribute it across many computing tasks. Lastly, it can be applied only when an entire non-linear region can be contained in a single computing task, i.e. when particle crossing the tile boundary can be ignored.

In this paper, we present a new method dubbed dynamic zoom simulation (DZS) devised to overcome the technical challenges faced by modern simulations. Similar to the SDF, the algorithm is designed to focus computational effort only in the regions of the simulation domain that are necessary for building a light-cone with given depth. However, in order to retain the large-scale gravity modes and render the implementation simpler and suitable to virtually all simulation codes available, we do not discard particles at distance $d > R_{lc}$ from the light-cone centre. Instead, we progressively decrease their resolution by merging together multiple particles, according to an adjustable set of rules. This approach beats down the computational cost and, simultaneously, does not require any change in the gravity solver.

The paper is organized as follows. In Section 2, we describe the details of the DZS algorithm and its implementation in the popular simulation code PGADGET-3 (a previous version of which has been described in Springel, Yoshida & White 2001; Springel 2005). Then, we describe a suite of numerical tests (Section 3) exploring the accuracy of the DZS algorithm. Its performance is studied in Section 4. We provide concluding remarks in Section 5.

2 DYNAMIC ZOOM SIMULATIONS

In recent years large cosmological simulations shifted from the (historical) snapshot outputs to light-cone ones. This shift has been driven both by the aforementioned technical challenges of storing a large number of snapshots for increasingly large simulations, and by the fact that galaxy surveys data – i.e. the main target of such large simulations – naturally come in the form of a light-cone. However, the internal structure of *N*-body codes has not evolved to reflect this change. In fact, simulation algorithms still commonly integrate the motion of all particles in the whole simulation volume, including those situated at distances $d > R_{lc}$ from the observer. Hence, a substantial amount of computation time is spent on regions of the simulation box that are then discarded in the output process. At low redshift,[3] the time wasted in such a way can be orders of magnitude larger than the one employed to integrate the motion of particles within the light-cone. A way to partially compensate for this is to employ the same simulation to produce multiple light-cones, i.e. placing multiple observers in the simulation box. However, such approach entails that the light-cones produced are not independent,

---

[1] The light-cone radius is defined as the comoving distance $\chi(z)$ along a null geodesic stemming from the observer at $z = 0$ and connecting it to an event situated at redshift z.

[2] The equation used in L17 is not the correct one and should be replaced by the trace of the Einstein's equation. Preliminary tests show no change in results or performance (Llinares, private communication).

[3] The relevant redshift depends chiefly on the box size L_{box}, which determines the redshift $\bar{z}$ at which the light-cone encloses a volume significantly smaller than the simulation box, i.e. $(4/3)\pi R_{lc}(\bar{z})^3 \ll L_{box}^3$.
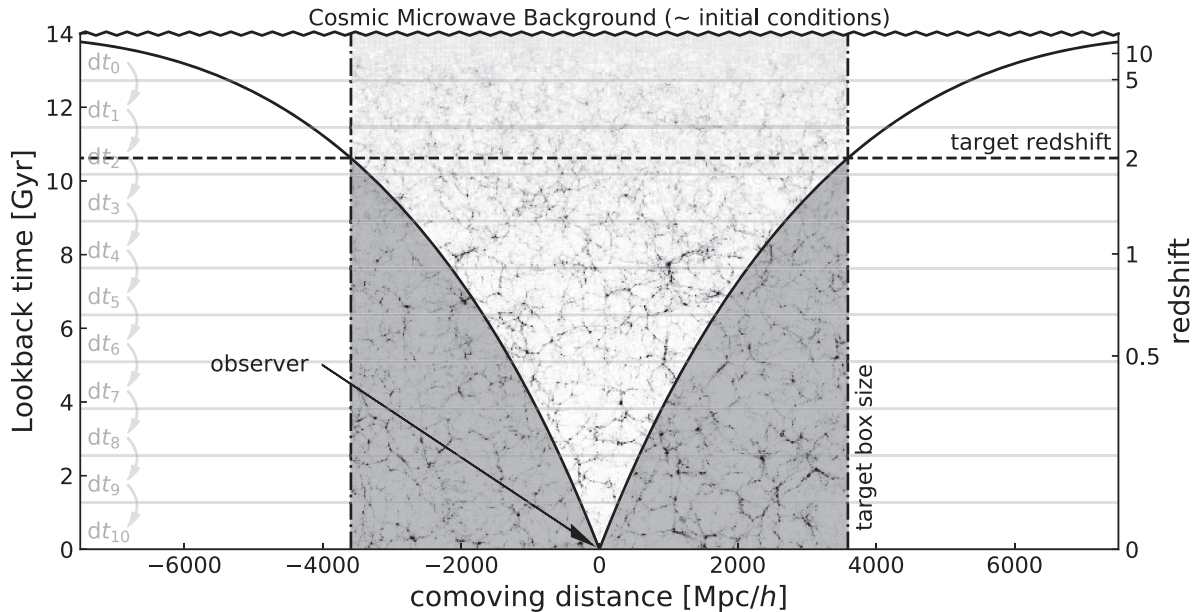
Figure 1. Space–time diagram (in 1+1 dimensions). The black solid lines show the light-cone for an observer placed at the origin assuming a Planck Collaboration XIII (2016) cosmology. The maximum ('target') redshift of the simulated light-cone (horizontal dashed line) corresponds to a minimum ('target') box size (vertical dot-dashed line). In normal simulations aiming at producing a light-cone output, the entire box is simulated from the initial conditions (undulating line on top) down to redshift zero. This is illustrated by the image in the background (representing a simulated cosmological density field in a $512\,h^{-1}$ Mpc box, and used only for displaying purposes). Usually, the initial conditions are evolved until redshift $z \gtrsim 50$ using linear theory, in order to save computational time. For the sake of simplicity, we have disregarded such step in the scheme. The grey-shaded areas highlight regions of space–time that are simulated but discarded, as not part of the light-cone. Finally, the thin light-grey horizontal lines exemplify the integration procedure adopted in simulations, where the entire box is advanced in time by a time-step dt.

as they sample the same region of the universe and thus the same structures (although at different times in their evolution). In the following, we focus on simulations that aim at producing a single full-sky light-cone for an observer placed at the centre of the simulation box. The reason for this choice is that forthcoming galaxy survey will cover a fraction $O(1)$ of the sky.

The concept described above is exemplified in Fig. 1 where we plot a $1 + 1$D projection of the light-cone evolution (solid lines) as a function of look-back time, converging at the observer location at the present time while diverging at the big bang, respectively located at the bottom and the top of the plot. In this example, we highlight a portion of universe with linear size $L_{\rm box} \approx 7000\,h^{-1}$ Mpc by enclosing it in vertical dot-dashed lines. When such a region is simulated (visually represented with an evolving cosmological density field, taken for display purposes only from a simulation with box size $L_{\rm box} = 512\,h^{-1}$ Mpc), the particles' equations of motion are integrated between discrete time-steps dt (horizontal grey lines). Particles are saved in the light-cone output when they encounter the solid lines (i.e. the light-cone boundary) during their time integration (which progresses from top to bottom in the figure).

All particles below the $\pm R_{\rm lc}$ lines (i.e. within the grey-shaded area) are therefore simulated although never used (again) for the light-cone construction. Even assuming the computational time is evenly distributed across cosmic time (while in reality matter clustering increases the computational cost at low redshift), it can be seen that a large fraction of the computing time is spent on particles that are then discarded. The figure also exemplifies how, in order to obtain a light-cone out to some target redshift (horizontal dashed line) without tiling simulation replicas, the box size required (vertical dot-dashed lines) quickly increases.

As we mentioned in the previous section, in a fully relativistic setup, particles outside the light-cone do not affect the evolution of

those inside, and hence their integration can be stopped as soon as they leave it with no consequence on the result (as it is done in the SDF). However, the vast majority of cosmological codes work in a Newtonian approximation, computing the gravitational field assuming an infinite speed of light (i.e. the gravitational field at any given time depends on the full matter distribution at such time). Therefore, simply stopping the integration of (or even removing) particles outside of the light-cone is not a suitable approach.

In order to overcome this limitation, the DZS algorithm combines the relativistic notion of light-cone with the Newtonian nature of typical cosmological simulations, in a way which is fully consistent with the latter and does not require the gravitational solver to be changed. Hence, it can be integrated with minimal modifications in most existing *N*-body codes. Such property of the DZS method is particularly valuable, as an array of auxiliary features, physical and processing modules, and optimizations already implemented in the baseline codes can be used alongside DZS, and benefit from the improved computational performance. In order to test the fidelity and speed-up obtained using the DZS approach, we have implemented it in the popular PGADGET-3 code, which is briefly reviewed in the following section.

2.1 The PGADGET-3 code

In the following, we briefly summarize the main features of the code PGADGET-3, with emphasis on the ones that are most relevant for our implementation of the DZS algorithm. We refer the reader to Springel (2005) for a more-detailed description.

PGADGET-3 follows the evolution of dark matter (DM) and baryons, discretizing them into particles. It employs a smoothed particle hydrodynamic (SPH) approach to solve the hydrodynamics

equation, while an hybrid TREEPM algorithm is used to compute gravitational forces. In this work, we focus on DM-only simulations, where SPH is not relevant, and hence we only describe the features relevant to gravitational interactions. However, the DZS algorithm can be easily extended to any additional implementation employing particles as tracers of some underlying physical quantities.

The hybrid algorithm employed by PGADGET-3 combines the speed, long-range accuracy, and memory efficiency of the particle-mesh algorithm (PM; see e.g. Klypin & Shandarin 1983; White, Frenk & Davis 1983) with the high dynamical range and short-range precision of a TREE algorithm, in the so-called TREEPM algorithm (Xu 1995; Bode, Ostriker & Xu 2000; Bagla 2002; Bagla & Ray 2003). The former uses an auxiliary (uniform) grid to compute the matter density and determines the gravitational force by solving the Poisson equation in Fourier space. In the TREE algorithm, on the other hand, all the particles in the simulation domain populate an OCT-TREE, and the gravitational acceleration exerted on a given particle by all the other ones is computed through the multipole expansion of the gravitational attraction of each tree node. In PGADGET-3 this expansion stops at the monopole order. The size of the TREE nodes employed (i.e. how 'deep' the OCT-TREE is traversed) is determined dynamically employing one of two available TREE-opening criteria: a geometrical one comparing the node linear size with its distance from the particle considered, and a relative one limiting the (estimated) force error introduced by the multipoles expansion. The PM and TREE-based gravitational force estimations are then combined in Fourier space using an exponential kernel, ensuring a smooth transition between the TREE-dominated (inner) region and the PM-dominated (outer) one.

The PGADGET-3 tree structure is implemented as a linked list of nodes. Each node represents a portion of the simulation box and stores both physical properties of the particles (e.g. total mass, centre-of-mass position and velocity) in such region of space and a series of links used to traverse the TREE. The resulting 3D OCT-TREE represents a hierarchical structure of nodes: the top – or root – node coincides with the simulation domain, while the tree level below it is made of eight child nodes, each of which covers one octant of the parent node. This partition is recursively repeated at each level of the TREE for every node containing more than one particle. We adopt a common terminology to indicate the relationship between such nodes, using the terms parent, child, and sibling node to indicate the ones in the upper, lower, and same TREE level of the considered node, respectively. Finally, PGADGET-3 is a massively parallel code. The computation, particle data, and OCT-TREE are distributed over multiple computing tasks using the message passing interface (MPI; Clarke, Glendinning & Hempel 1994). The connection between tree branches residing on different tasks is provided by pseudo-particles, i.e. particle-like structures that store the centre-of-mass properties of each TREE node stored on another task.

In order to cope with the large dynamical range typically encountered in astrophysical systems, PGADGET-3 allows particles to have individual time-steps, determined by their dynamical properties. For instance, high-density environments can have orders-of-magnitude smaller time-scales than the intergalactic medium. In particular, particle time-steps in PGADGET-3 are organized in power-of-two subdivisions of the system time-step dt, i.e. individual particle time-steps are rounded to the closest smaller interval d$t_n \equiv$ d$t/2^n$. Hence, all particle time-steps are commensurable to each other and, therefore, it is ensured that there exist simulation steps – called global – in which all particles are actively evolved.

Finally, in PGADGET-3, particles can belong to six different groups – or types – each one of them potentially having unique properties, most notably their mass and gravitational softening

length h. Typically, different particle types are used to represent either (astro)physically different (e.g. gas, stars, black holes, etc.) or numerically different (e.g. particles with different resolution) objects, as well as a mix of the two. We anticipate here that we will indeed employ different types for the latter.
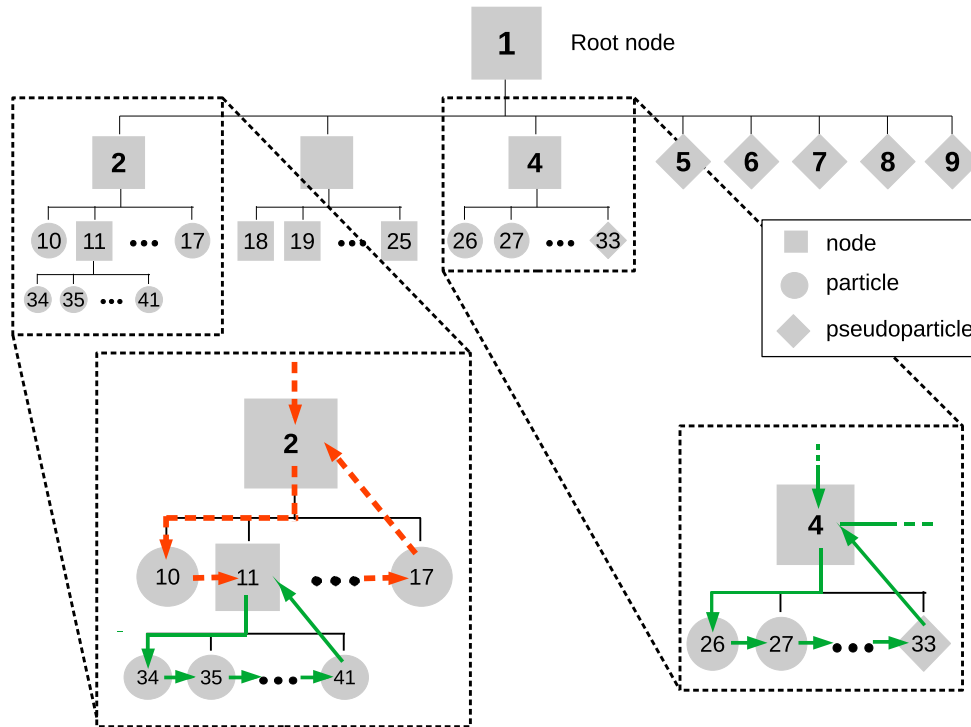
2.2 The DZS algorithm

Our implementation of the DZS algorithm builds on the OCT-TREE structure of PGADGET-3 as an independent module which is called recursively during the simulation. When the DZS module is invoked, the entire simulation OCT-TREE is 'walked' starting from the root node. Whenever a node does fulfil a set of node-merging criteria, its children nodes are replaced by a single particle inheriting all their physical and numerical properties, as detailed in the next paragraphs. If the criteria are not satisfied, the process is repeated separately for each child node. This approach requires only a single tree walk (which scales with the logarithm of the number of particles), making it very efficient. In all our tests, the time required by the execution of the DZS module itself is negligible, never exceeding 0.1 per cent of the entire simulation time. This does not include any additional overhead that may arise from the work-load imbalance due to the (potentially large) difference in resolution across the simulation domain, which will be discussed in detail in Section 4.1.1.

The error introduced by the suppression of the simulation resolution in the nodes that fulfil the merging criteria (corresponding to regions of the simulation domain lying outside the observable light-cone, as will be detailed below) can be arbitrarily reduced by tweaking the node-merging criteria themselves, at the cost of a smaller performance gain. We have implemented the node-merging criteria to resemble the ones used in the force computation (see Section 2.2.1 for more details). Hence, if the parameters used are equal, the error introduced by DZS at any given time-step and for any given particle inside the light-cone is ensured to be at most the one introduced by the force computation. Typically, however, we advise to employ more stringent criteria for DZS, as its effects are permanent and generally non-isotropic for any given particle (unlike, in principle, the error introduced by the force computation).

In Fig. 2 we provide two examples of how the node-merging procedure of the DZS algorithm works in practice. The figure depicts the TREE structure on a single MPI task hierarchically distributed along the vertical direction, where each grey box represents a node, the thin black lines show its connections to other nodes, particles are depicted by grey circles, and diamond symbols are used for pseudo-particles. The two boxes show two details of the node-merging process, using a simple colour coding where red dashed lines signal that particles encountered during the walk should not be merged, while green solid lines flag that particles should be de-refined. In the lower left box, the TREE walk (red dashed line) reaches node 2, the merging condition is checked but not met. The TREE walk continues from the first child node (10) and, since the latter is a particle, the walk continues from its sibling 11. In this case, the merging criterion is met and, therefore, the node is flagged for merging (depicted by the TREE-walk line becoming solid green), and the walk continues from its first child (34). The latter is a particle, and since its parent node satisfies the merging condition, it is flagged for de-refinement before the walk continues from its sibling (35). When the last child node is processed, the walk returns to the parent node (11).

A slightly different configuration is presented in the bottom right-hand box. In this case, one of the child nodes is a pseudo-particle (33). Since the TREE is consistent across different tasks, the de-refinement

Figure 2. Flow chart of the TREE walk in the DZS algorithm. The two boxes in the lower part of the figure show two examples of TREE walk, using a simple colour scheme with green solid lines signalling that particles encountered during the walk should be de-refined, and red dashed lines signifying that they should not be merged.

process produces exactly the same outcome independently of the particle distribution among tasks. Hence, pseudo-particles are safely ignored, as the task containing the data pointed to by such pseudo-particles will process them and is guaranteed to accomplish identical results.

Once the full TREE has been walked, particles flagged to be de-refined are removed and TREE nodes flagged to be merged are turned into particles. This operation is done by transferring the centre-of-mass properties of the flagged node to the newly created particle.[4] If nodes flagged for de-refinement only contained particles local to the task, the new low-resolution particle is simply created in loco; however, whenever a flagged node contains a pseudo-particle – i.e. the TREE branch is partly contained in another task – the newly created particle will reside on the task whose rank is the minimum among all tasks contributing to that TREE branch. This precautionary measure ensures that only a single new particle is created for branches mapped to several tasks and does not require communication, which could be a source of performance loss.

As mentioned, PGADGET-3 allows particles to have individual time-steps. To maintain the time-integration consistency, particles grouped together by DZS need to be synchronized. This is ensured by applying the DZS algorithm only during the so-called global time-steps.

By removing and replacing particles within the simulation, DZS drastically changes the particle distribution across tasks, as well as the TREE structure. For this reason, whenever DZS modifies at least one particle, the TREE needs to be rebuilt, the acceleration and time-step of new particles recalculated, and a new domain decomposition performed. To mitigate (and in fact practically remove) the overhead introduced by these operations, we additionally restrict DZS to be invoked only whenever a new domain decomposition (which is the most time-consuming of them) is performed. By running DZS just before a standard domain decomposition, we ensure that the computational effort invested into it is not duplicated. Finally, we force individual particle time-steps outside of the light-cone to be at most as short as the shortest time-step inside it. While most of the times this condition has no effect, it can prevent some pathological situation where two particles outside of the light-cone get unphysically close as a consequence of the de-refinement procedure, and hence require a very short time-step to precisely integrate their dynamics, despite the latter has a negligible impact on the evolution of particles inside the light-cone.

By grouping particles together, DZS changes the mass of individual resolution elements in the simulation while preserving the total mass in the system, unlike SDF. Although taking place outside of the light-cone – where the detailed dynamics of particles is of no interest – the evolving particle mass can potentially alter the large-scale density field and increase the computational cost of the simulation. The latter is a consequence of the larger gravitational pull of de-refined particles (due to their increased mass) that can significantly decrease the time-step of nearby particles if not compensated by an appropriate scaling of their gravitational softening. Keeping in mind that the latter is related to the ideal volume occupied by the density element represented by each zero-dimensional particle, DZS rescales the

---

[4]This is sufficient in our case since PGADGET-3 stores enough information in the TREE nodes to generate a fully functional new dark matter particle. In general, the same outcome can be achieved by explicitly computing average properties of the de-refined particles, at the price of additional communication in the case a node contains pseudo-particles.

original softening length by a factor $(M/M_{init})^{1/3}$, where M_{init} is the initial (i.e. before de-refinement) mass of particles.

The merging criterion employed by DZS is designed to ensure a precise integration of particles inside the light-cone (see Section 2.2.1). However, the evolution of the large-scale gravitational field needs to be approximately followed as well, since it affects the particles inside the light-cone in the Newtonian approximation. For this reason, the merging criterion is complemented with the possibility to limit the maximum linear size l_{max} of the nodes flagged to be merged. This is similar to setting the background resolution in zoom-in simulations. Notice that even the very conservative choice of $l_{max} = 2l_{min}$, where $l_{min} = L_{box}/N_p^{1/3}$ is the smallest node size if particles were homogeneously distributed, already significantly speeds up the computation approximately by a factor $\gtrsim 16$ outside of the light-cone.[5]

Finally, our implementation of DZS allows de-refined particles to be moved to a different particle type with respect to the full-resolution particles, to allow for an easier handling of the simulation data set for both on-the-fly and post-processing operations.

2.2.1 De-refinement criteria

Since the de-refinement criteria of the DZS scheme are based on the TREE-opening criteria used in the force calculation in PGADGET-3, we first review the latter before introducing their adaptation to DZS. The relevant quantities and geometry are schematically depicted in Fig. 3.

PGADGET-3 allows for two different opening criteria in the force calculation, one geometrical and one dynamical. The former requires that, for each target particle i for which the force is to be computed, a given node n must be opened (i.e. the tree walk should be continued through its child nodes) whenever its linear size l_n is larger than the distance between its centre-of-mass and the particle $d_{in} = |\mathbf{r}_i - \mathbf{r}_n|$ times an (user-defined) 'opening angle' θ_{geom}, i.e.

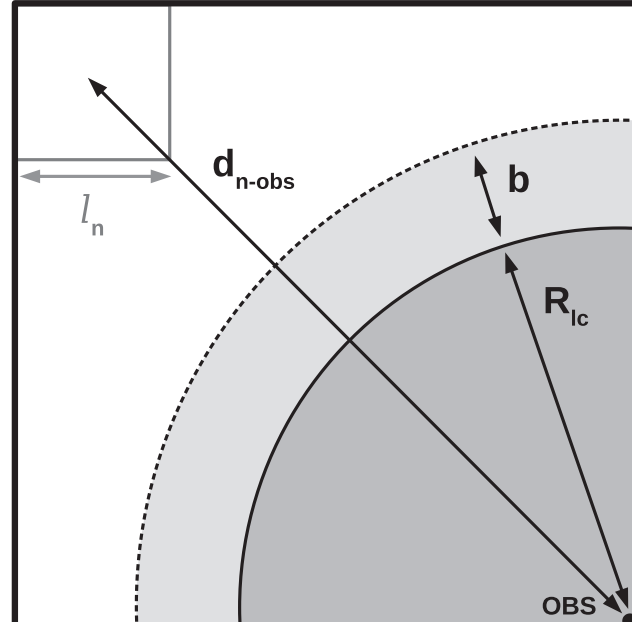$$\frac{l_n}{d_{in}} \geq \theta_{geom}. \qquad (1)$$

The second – dynamical – criterion is devised to strictly enforce a maximum force error introduced by the TREE-opening choice and, at the same time, to allow large nodes to be used whenever possible. It takes the form

$$\frac{GM_n}{d_{in}^2} \left(\frac{l_n}{d_{in}}\right)^2 \geq \alpha_{dyn} |\mathbf{a}_{i,old}|, \qquad (2)$$

where G is the gravitational constant, M_n is the total mass contained in the node, $\mathbf{a}_{i,old}$ is the particle acceleration in the previous time-step, and α_{dyn} is a (user-defined) parameter that sets the force accuracy.

These opening criteria are local, since they are used – and yield different results – for each particle i in the simulation. However, they need to be cast into global criteria to be usable by DZS, since its modifications to the simulation affect all particles. In addition, the force computation is 'particle-centred', while DZS requires 'node-centred' criteria. To achieve this, we have replaced the distance d_{in} with the minimum distance of the node n to any given particle lying inside the light-cone, allowing for an additional spherical shell of

Figure 3. Schematic representation of the relevant lengths involved in the de-refinement process. We are showing here a 2D projection of one octant of the simulation box. The observer is located at the bottom right-hand corner (corresponding to the centre of the box, while the solid circle and dark-grey shaded area represent the light-cone radius and volume, respectively. The light-grey shaded region (delimited on the outside by the dashed circle) denotes the buffer zone. Finally, a single TREE node is depicted in the top left of the plot.

user-defined width b that acts as a buffer zone. In practice, d is replaced by

$$\mathcal{D}_n = \min_{i:\ |\mathbf{r}_i - \mathbf{r}_{obs}| < R_{lc} + b} |\mathbf{r}_n - \mathbf{r}_i| = \max(d_{n-obs} - R_{lc} - b, 0), \qquad (3)$$

where d_{n-obs} denotes the distance between the centre-of-mass of the node and the centre of the light-cone (i.e. the observer's location $\mathbf{r}_{obs}$).

While this replacement is sufficient to make the geometric criterion global, the dynamical one depends on an additional local quantity, namely the last-recorded acceleration of the particles. For this reason, we have replaced $a_{i,old}$ with its minimum value among all particles inside the light-cone.

Finally, we note here that the buffer zone can be used in order to enforce a minimum volume (i.e. a sphere of radius equal to the buffer length b) to be resolved at the original resolution down to $z = 0$. This allows (i) the production of a traditional time-slice output of a desired fraction of the simulation box; and (ii) to produce multiple light-cones (for observers at different locations in the box) within the same run, which are independent from each other out to some redshift z (the exact value depends on the observer locations).

In the remainder of the paper, unless otherwise specified, we will adopt a standard configuration of DZS employing a geometrical opening criterion with angle $\theta_{geom} = 0.1$, a maximum node length of $l_{max} = 4\lambda$, where λ is the mean interparticle distance (e.g. ensuring the low-resolution region is sampled with an equivalent resolution of at least 128^3 for a simulation with 512^3 particles), and a buffer size corresponding to 5λ. These are relatively conservative parameters. The effect of different parameter choices are investigated in Section 3.3.

---

[5]The factor $\gtrsim 16$ is made up by a factor 8 coming from the decreased number of particles, a factor of 2 coming from the increased mean interparticle distance that allows larger time-steps, and an additional gain from the reduced clustering, since the merging criterion is geometrical and, hence, clustered particles are merged into single de-refined particles.

2.2.2 Future developments

Before moving on to a thorough validation of the DZS algorithm in the following section, we want to briefly mention here some foreseeable developments that can – or are going to – be included in DZS. The structure of DZS is such that it can be easily extended to all particle-based implementations of additional physics that may be already available in the code, or that may be included in the future. For instance, PGADGET-3 is already capable of simulating a wide variety of physical processes beyond the (nowadays standard) adiabatic and radiative hydrodynamics, as e.g. Modified Gravity (e.g. Puchwein, Baldi & Springel 2013), non-standard Dark Sector physics (e.g. Baldi et al. 2010; Baldi 2012), massive neutrinos (e.g. Viel, Haehnelt & Springel 2010), Ultra-Light Axions (also known as Fuzzy Dark Matter, see e.g. Nori & Baldi 2018), radiation-transport (e.g. Pawlik & Schaye 2008; Petkova & Springel 2009), magnetic fields (e.g. Dolag & Stasyszyn 2009; Bonafede et al. 2011), and many more. As all these implementations are still resorting on a particle-based discretization of physical processes, such additional modules can be combined with the performance improvement granted by DZS in a relatively easy way. In particular, we are working on extending our current implementation of DZS to include modified gravity (Nori et al., in preparation).

Furthermore, the light-cone output procedure that is currently paired with the DZS algorithm can be easily – and independently from the latter – extended to include a range of additional properties, computed on-the-fly during the simulation run, in order to compensate for the lack of snapshot outputs. In particular, we plan to extend the current light-cone output to include the gravitational potential and its time derivative.

Finally, a possible further optimization (which however is independent from DZS itself) consists in employing initial conditions with coarser resolution outside of the region of space which ends up in the light-cone at the redshift of interest. This allows us to include large-scale density modes with a limited additional burden for the simulation.

3 ALGORITHM VALIDATION

We now present the results of a test suite designed to validate the accuracy of DZS. These simulations are conceived – for comparison purposes – to mimic and extend the ones performed by L17. They cover a factor of 16 in box size and a factor of 8 in number of particles. We complement them with a re-run of the ΛCDM simulation of the DUSTGRAIN suite (a $2\,\mathrm{Gpc}\,h^{-1}$ cosmological box with 2048^3 CDM particles; Baldi et al., in preparation), in order to test our algorithm in a more realistic and computationally challenging setup, and to extend the range covered in particle number N_{part} by another factor 8. The main properties of our test suite of simulations are summarized in Table 1.

In the following, we will use the term 'twin simulations' to indicate pairs of runs that only differ in the usage of DZS, which is enabled in the one labelled as *dzs* and disabled in the one labelled as *std*.

In Fig. 4 we provide a view of the effect of DZS on the simulated particles in the SMALL twin runs. Each top panel shows a slice of thickness $0.5\,h^{-1}$ Mpc at redshift $z = 0.05$ extracted from the *std* run (left-hand panel) and from the *dzs* one (right-hand panel). Each particle in the slice is plotted as a black dot, with size and opacity increasing with the particle mass (linearly and quadratically, respectively, for the sake of visual clarity, as indicated in the bottom left-hand corner of the top right-hand panel). It can be clearly seen that the structures inside the light-cone (dashed orange circle) are unaffected by DZS,

Table 1. Simulation set employed for scaling and accuracy tests, replicating the test suite of L17. Each row corresponds to two twin simulations, one run using the DZS algorithm and one without.

Name	L_{box} (h^{-1} Mpc)	N_p	Modelled on
TINY[a]	128	512^3	
SMALL	512	512^3	Llinares (2017)
MEDIUM	2048	512^3	Llinares (2017)
LARGE	8192	512^3	Llinares (2017)
LARGEHR	8192	1024^3	Llinares (2017)
DG	2000	2048^3	Baldi et al. (in preparation)

Note. [a]Used only in performance estimation (Section 4.2).

while outside of it the number of particles in the right-hand panel rapidly decreases, but the large-scale structures are preserved. We emphasize this feature by showing in the bottom panels the average particle mass as a function of distance from the observer (placed at the centre of the box). The latter does not increase immediately outside of the light-cone (plus buffer) radius since a geometrical de-refinement criterion is used in the SMALL *dzs* run, and hence only nodes with angular size (as viewed from the closest point on the buffer surface) below a given threshold are merged together. Finally, we note that the noise at large radii is due to the spatial nature of the OCT-TREE subdivisions. When a low-resolution particle is produced, its mass will reflect the mass contained in the parent node, and hence the mass in the region of the universe covered by the latter. It follows that nodes encompassing large cosmic structures (e.g. galaxy clusters) will contain a larger mass than those enclosing voids, producing a disparity in the mass of low-resolution particles stemming from them, which ultimately leads to the noisy profile in the figure.

3.1 Light-cone observables

In the following we move to a quantitative evaluation of the accuracy of (our implementation of) DZS. The different simulation paradigm employed by DZS renders typical measures of accuracy (e.g. the matter power spectrum, the halo mass function, etc.) not readily usable, and adapting them to the new framework would produce issues when comparing them to other results. Hence, we decided to base our accuracy analyses only on observable quantities, i.e. measured (using a piecewise-constant approximation) on the 4D-light-cone surface. The latter corresponds to the 3D volume generated as the set of spherical surfaces centred on the observer with radius $R_{lc}(z)$ for any redshift $z \le z_{max}$, with z_{max} being the highest redshift considered for a given light-cone. This choice is also the most conservative in terms of accuracy testing, since by definition the particles on the light-cone spatial boundary are those most affected by the decreased resolution outside of it, being the closest to the low-resolution regions. Additionally, in Section 3.2 we investigate the time-integration fidelity of DZS by comparing time slices in twin simulations. Finally, it is important to stress again here that the DZS algorithm can be made arbitrarily precise by tweaking the node-merging criterion parameter, at the cost of a decreased speedup (see Section 3.3 for details). Hence, we test the accuracy only in our standard set of settings, also in order to be consistent with the performance tests presented in Section 4.

3.1.1 Halo mass function

We start our investigation by comparing the light-cone halo mass function (LCHMF) integrated over the redshift range $0.0 \le z \le 0.68$ in the DG twin simulations. Among our simulation pairs, this
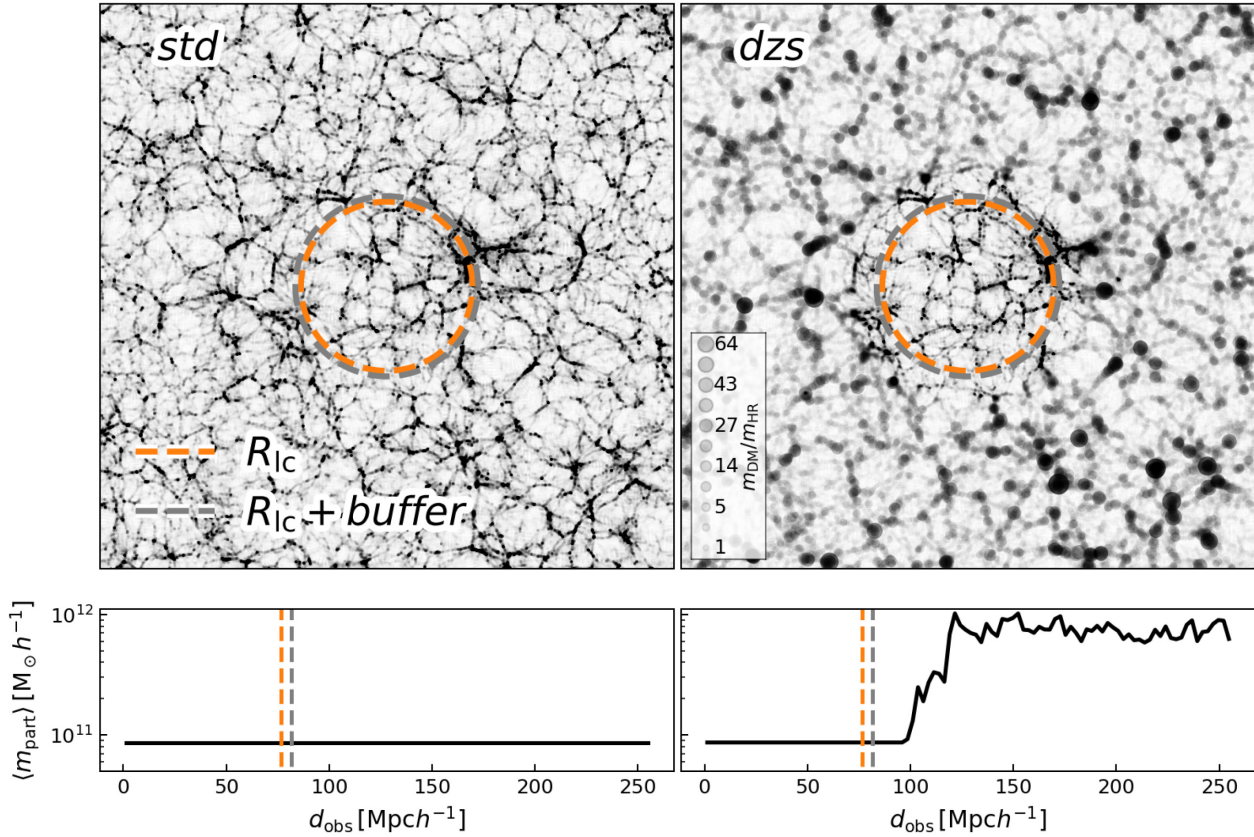
Figure 4. Top panels: Dark matter distribution in a slice of thickness $0.5\,h^{-1}$ Mpc extracted from the SMALL twin simulations, at $z = 0.05$. Each dot represents a particle in the simulation, with size and opacity increasing with their mass (as indicated in the bottom left corner of the right-hand panel). In the *std* run (left-hand panel), each particle has the same mass, while in the *dzs* one, different levels of de-refinement can be seen. The structures inside the light-cone (orange dashed line) are identical in the two runs, and the large-scale matter distribution is preserved also outside of the light-cone in both panels. The grey dashed line shows the effect of adding a buffer zone around R_{lc}. Bottom panels: the mean particle mass as a function of distance d_{obs} from the observer (placed at the centre of the box).

is the one that better resolves structures (i.e. it has the best mass resolution) and, simultaneously, has a box size large enough to (i) allow the construction of a light-cone out to a redshift significantly larger than zero ($z \sim 0.68$) without requiring box replications, and (ii) is affected by DZS for a significant fraction of the simulation time. In comparison, the LARGEHR pair is affected by DZS for a much longer time but has a 550 times worse mass resolution, hindering the identification of many cosmic structures. In order to produce the LCHMF we save with high cadence the halo catalogues produced by the Friend-of-friends (FoF) algorithm integrated in PGADGET-3. These outputs were then combined into a light-cone catalogue by means of a post-processing procedure employing a piecewise-constant approximation, ensuring that this construction only makes use of haloes entirely inside the light-cone at the time of the output.

The results of this procedure are shown in the top panel of Fig. 5, where we show the redshift-integrated mass spectrum of haloes on the light-cone. It is immediately clear that DZS is able to almost-perfectly recover the LCHMF of the standard simulation. More quantitatively, as shown in the bottom panel of the figure where we plot the fractional deviation of the LCHMF computed in the *dzs* run with respect to the *std* one, our algorithm can reproduce the LCHMF with a sub-percent level accuracy.
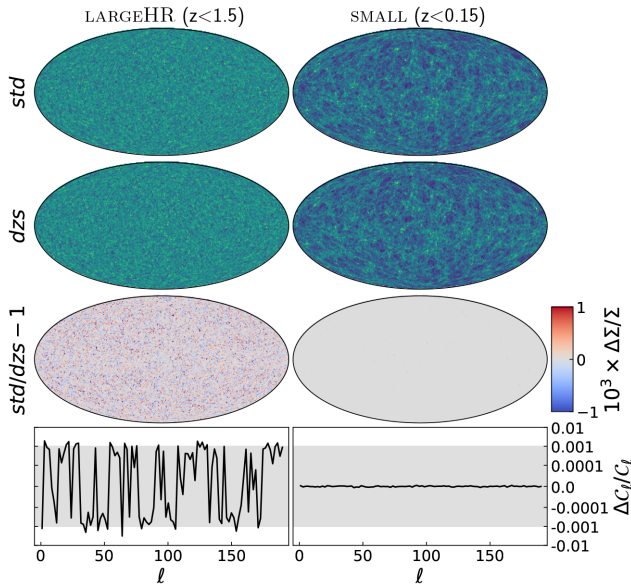
3.1.2 Sky-projected light-cone

While the LCHMF is a measure of the accuracy reached by DZS in simulating large cosmic structures, we can alternatively compare the
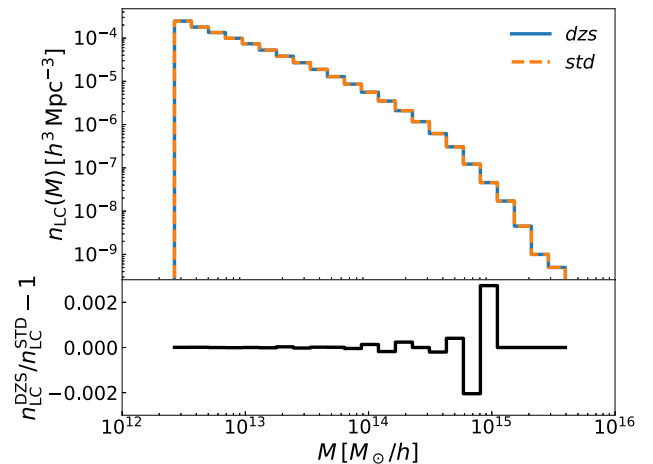
Figure 5. Light-cone halo mass function (top panel) at redshift $0.0 \leq z \leq 0.68$ in the DG simulations with DZS enabled (solid histogram) and disabled (dashed histogram). Their relative deviation is always below 0.2 per cent (bottom panel).

light-cone mass distribution projected on the sky in some redshift interval. In order to obtain such projection, we use HEALPIX (Górski et al. 2005, with parameter $NSIDE = 1024$) to obtain the original pixelization from the simulation, but then we downsample it to $NSIDE = 256$ for display purposes. Although not directly observable,

Figure 6. Sky-projected (using HEALPIX sky pixellization with $NSIDE = 256$) density distribution on the light-cone for the *std* (top row) and *dzs* (second row from the top) runs in the LARGEHR (left-hand column, integrating over $0 < z < 1.5$) and SMALL (right-hand column, integrating over $0 < z < 0.15$) pairs. The pixel-wise relative difference between the top two rows is shown in the third one. Finally, the last row shows the relative difference in the angular power spectrum computed from the sky maps shown in the top two rows. The shaded grey region indicates the 0.1 per cent accuracy level. Notice that the top two sky maps on the right are identical, and hence their difference is always zero. This means that no particle in the two runs is displaced more than the pixel size. Note that the vertical scale in the bottom panels is linear for absolute values smaller than 10^{-4}, otherwise the vertical position for a value v is given by $\text{sign}(v)\log(|v|)$.

this quantity is the one that large-scale surveys aim to reconstruct starting from either biased tracers (first and foremost galaxies) or weak lensing shear measurements.

We present such comparison in Fig. 6. Each column refers to a different simulation pair (reported on the top), and a different redshift range for the light-cone projection (also reported on top). The ranges were chosen in such a way that they encompass most of the time the light-cone has been inside the simulation box. The first two rows show the projection in the *std* and *dzs* runs, while their pixel-wise relative difference is shown in the third row. The latter is very small in most of the pixels, and never exceeds 0.1 per cent. Most importantly, the few pixels where it is close to its maximum are randomly located in the sky, showing that they occur because of numerical stochasticity triggered by the slightly perturbed gravitational potential, rather than by systematic effects. Finally, the last row shows the relative difference in the angular power spectrum between the *std* and *dzs* realizations (i.e. it is not the power spectrum of the difference map). Also in this case, the deviation is almost always below 0.1 per cent, indicated by the grey-shaded horizontal band and its scatter (around 0) appears random.

While Fig. 6 shows the high degree of fidelity achieved employing the DZS approach, it compresses a large redshift range on to the sky. In the left-hand panels in particular, the individual structures are lost and the density field appears random to the eye. For this reason, we show in the bottom right-hand panel of Fig. 7 the sky-projected light-cone in the DG *dzs* run, but now projected over a thin redshift slice $z \lesssim 0.03$, which allows to appreciate even more how cosmic structures

and their renowned web-like topology are unchanged when the DZS approach is employed.

3.1.3 3D light-cone

Finally, we investigate here the ability of DZS to capture the full 3D matter density light-cone constructed from our simulations. Unlike the case of the LCHMF, which has been constructed using a piecewise-constant approximation from the halo catalogue at fixed cosmic time, we produce the particle light-cone on-the-fly. This is done by saving with very high frequency the particles in a thin spherical shell whose outer rim coincides with the light-cone radius at any given redshift.

In Fig. 7 we show the light-cone constructed as described above in the *dzs* (top left-hand panel) and *std* (top right-hand panel) runs of the DG re-simulation in the redshift range $0.0 \leq z \leq 0.68$. A visual inspection reveals that the two appear identical. We quantify this in the bottom left-hand panel of the figure, where we show the relative pixel-wise difference of the two runs (notice that this is exactly zero almost everywhere, making it hard to spot pixel-scale differences). It can be appreciated that the perturbations introduced by the DZS algorithm are minor, amounting to only ~ 0.15 per cent at most (bottom left-hand panel).

3.2 Integration accuracy

In the previous sections, we have shown that DZS is capable of reproducing basic features of the matter distribution, as the LCHMF and the projected matter density across a large redshift range, at a fraction of the cost required by a normal simulation. While these tests show that the DZS algorithm can be faithfully employed, here we test it in the most stringent way possible, i.e. comparing particle-by-particle a pair of twin simulations. Note that DZS permanently affects the simulation structure. Hence, the comparisons presented in the following necessarily show the integrated effect of DZS over time. For this reason, it is expected that differences will be larger than the nominal force computation error introduced by the node-merging criterion, as the latter is a measure of the error introduced at each time-step. By construction, such test makes sense only for particles inside the light-cone. For this comparison, we identify 'twin particles' that started exactly identical in the initial conditions and cross-match them in different runs.[6]

In Fig. 8 we show the cumulative distribution of displacements Δx between particle pairs, normalized by the softening length $h \equiv L_{\text{box}}/N_{\text{p}}^{1/3}/40$, in the LARGEHR simulations pair. The vertical dashed line corresponds to $\Delta x/h = 8$, a value that we consider a reasonable limit for the spatial reliability of a simulation in terms of individual particle positions. In this simulation the first DZS de-refinement occurs at redshift $z \approx 11$. The particle displacement increases with time, as expected, since the small errors introduced by DZS are carried on during the particle integration and therefore

---

[6]In the following, we consider the simulation without DZS (labelled *std*) as the ground truth. While this could be improved upon by, for instance, employing a direct summation method for the computation of gravitational forces, the usefulness of such comparison would be limited since (i) we are interested in studying the impact of the DZS algorithm only, and therefore prefer to remove any additional potential source of differences between the simulation pairs; (ii) de facto all cosmological simulations use approximate methods to compute gravitational interactions (direct summation becomes rapidly prohibitive for large particle numbers and long integration time), and the PGADGET-3 code is one of the most widespread in the community.
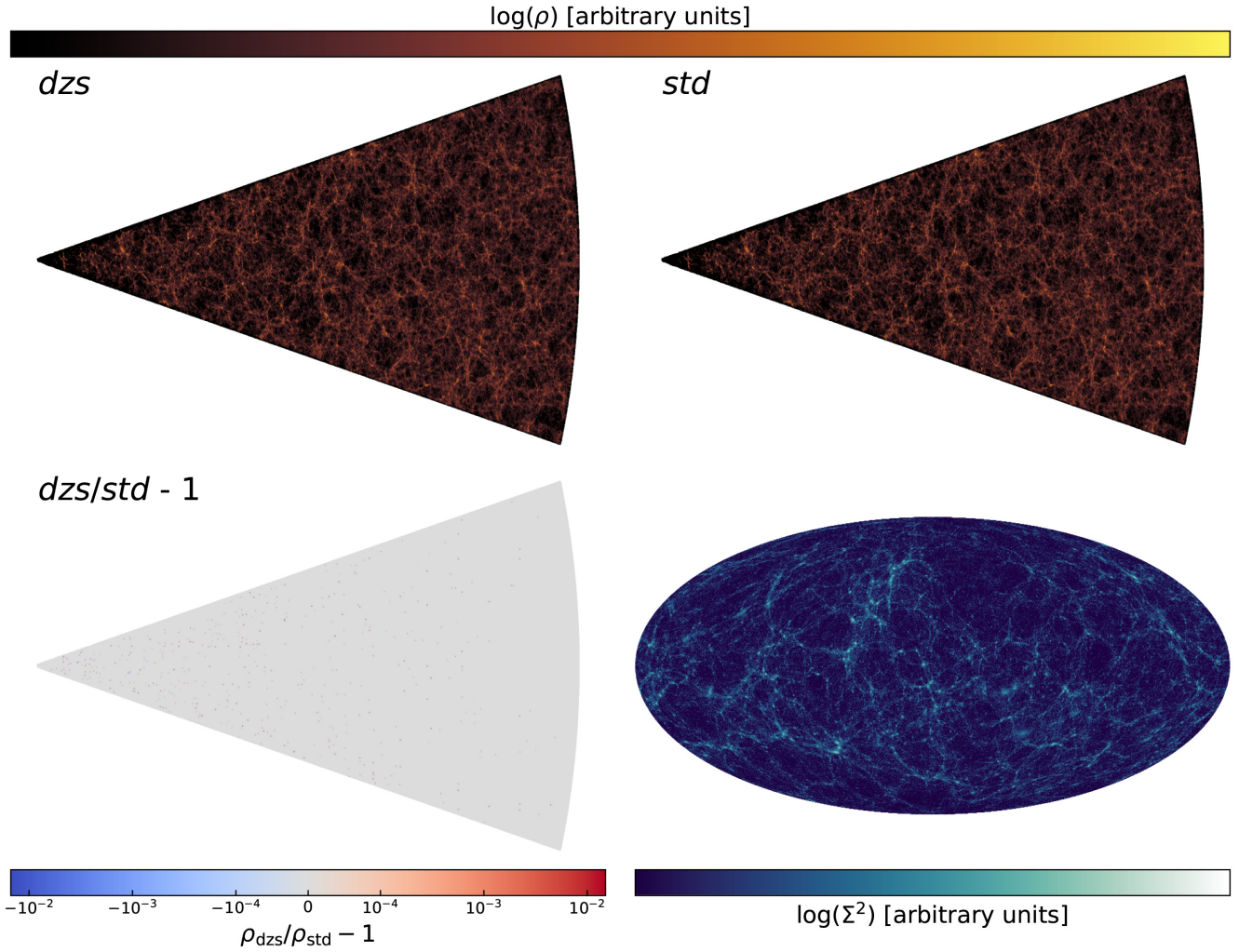
Figure 7. Top panels: Thin slice of the 3D matter density light-cone produced on-the-fly in our *dzs* (left-hand panel) and *std* (right-hand panel) runs of the DG re-simulation. Bottom left-hand panel: Relative difference between the *dzs* and *std* runs. The colour scale is linear for absolute values smaller than 10^{-4}, otherwise the colour for a value v is given by $\text{sign}(v)\log(|v|)$. Bottom right-hand panel: Sky-projected light-cone (using HEALPIX sky pixelization with *NSIDE* = 1024) density distribution in the DG *dzs* run at $z \lesssim 0.03$.
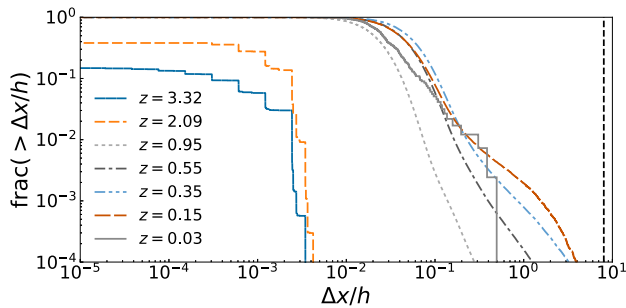
Figure 8. Cumulative distribution of differences in the position (normalized by the softening length) of particles inside the light-cone across the twin simulations LARGEHR. The vertical dashed line on the very far right indicates eight times the softening lengths.

pile up. Nevertheless, the displacements always remain very small, with no light-cone particles being displaced by more than $8h$ at any redshift, and only ≈ 0.3 per cent of them being displaced by h at any redshift. Finally, we note that in the lowest redshift distribution, the

number of particles with large $\Delta x/h$ is in fact reduced with respect to the one at $z = 0.15$. This is a consequence of the small number of particles remaining in the light-cone (which approaches a vanishing volume at $z = 0$) not sampling the tail of the distribution any more.

We have chosen to show the integration accuracy in the LARGEHR simulation since it is the one where DZS has been active for the longest, and therefore errors have accumulated the most. In fact, the displacement distribution is strongly shifted towards smaller values in the MEDIUM and SMALL simulation pairs at any given redshift.

Finally, in Fig. 9 we compare the integration accuracy as a function of resolution. In order to do so, we employ the LARGE and LARGEHR pairs. The increased mass resolution produces two effects: (i) a shift towards larger values of the $\Delta x/h$ distribution, and (ii) the development of a tail containing a small (10^{-2}–10^{-3}) fraction of the particles extending to larger errors. The latter feature (and the distribution itself alike) appears to be very similar to what is found investigating the force errors distribution in PGADGET-3, when a geometrical TREE-opening criterion is employed (see e.g. the top panel of fig. 1 in Springel 2005). In fact, even using a dynamical de-refinement criterion in DZS, we obtain a similar behaviour since, as detailed in Section 2.2.1, the necessity for a node-centric criterion
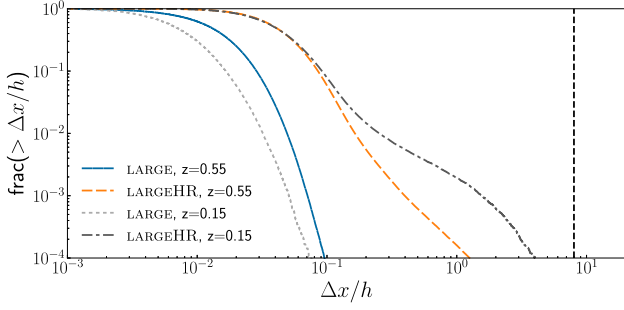
Figure 9. Same as Fig. 8 but now comparing the LARGEHR and LARGE pairs, in order to investigate the effect of increased mass resolution. For visual clarity, we show only the curves corresponding to $z = 0.55$ and $z = 0.15$.

renders the dynamical and geometrical ones somewhat similar to each other.

3.3 Parameter dependence

The comparison presented in the sections above shows a high level of fidelity in the results obtained employing DZS with respect to those obtained without. However, it does so employing somewhat conservative values for the parameters controlling the behaviour of the DZS algorithm. In a realistic setting, such parameters will be chosen to be as 'aggressive' as possible, in order to maximize the performance gain. For this reason, in the following we investigate the dependence of the results on the four different parameters related to the de-refinement process, namely: the geometrical opening angle (θ_{geom}), the dynamical accuracy parameter[7] (α_{dyn}), the buffer size (b), and the maximum size a TREE node can have and still being merged (l_{max}). The former two control the transition between the high-resolution zone and the rest, the third enforces a safety high-resolution region around the actual light-cone, and the last one sets a minimum resolution for the volume outside of the light-cone.

As done in Section 3.2, we match the particles in twin simulations, enabling a particle-by-particle direct comparison of their positions. The results of varying, separately, each one of the four parameters listed above are shown in Fig. 10. In each panel, all parameters but one are kept fixed to the values used in the LARGE runs (i.e. $\theta_{geom} = 0.1$, $b = 5$, $l_{max} = 128 \, h^{-1}$ Mpc). The values investigated are reported in the legend accompanying each panel, ordered with the most conservative value at the top and the most aggressive one at the bottom). The data presented in the figure are complemented by Table 2 where we list for each simulation the ratio between the initial and final particle number N_p^{fin}/N_p^{init}, as well as the reduction in computational time $T_{wc}^{DZS}/T_{wc}^{STD}$. Note that, as discussed in Section 4, the setup of these simulations is such that DZS is somewhat restrained. Hence, the performance boost should be interpreted as a lower limit for what can be achieved in more realistic settings (i.e. employing higher resolution or a more complete set of physical processes simulated). We will address this issue in a future work, where we will apply our DZS algorithm in a realistic simulation setting.

It should be noted that the effects of the DZS parameters investigated here are intertwined with the ones due to the TREE-opening criterion used in the force computation. In particular, whenever the latter introduces errors larger than the one added by DZS, the differences in the integration accuracy are suppressed. Hence, the

---

[7]Note that θ_{geom} and α_{dyn} are mutually exclusive.
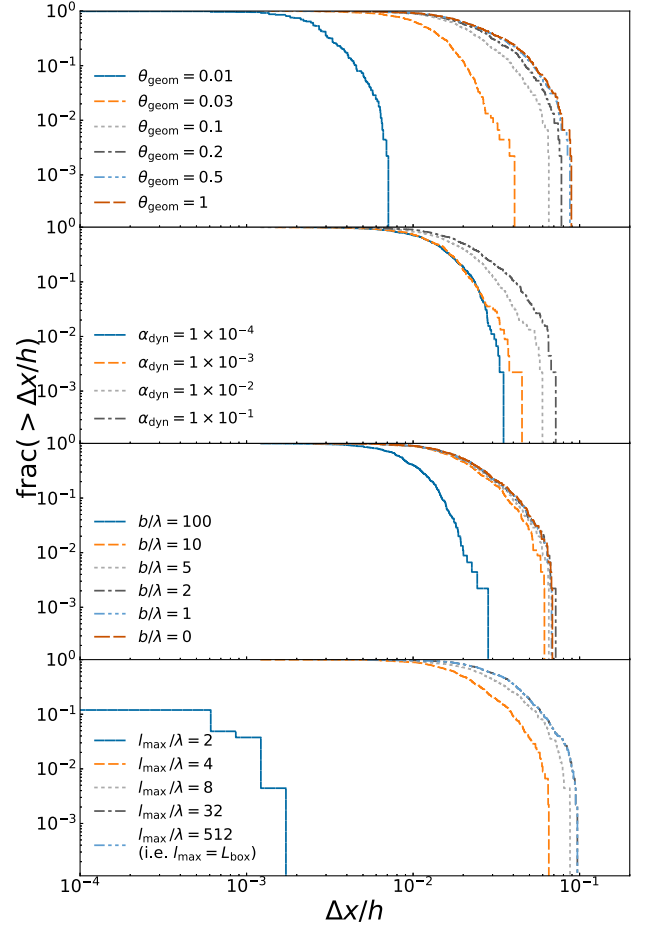
Figure 10. Similar to Fig. 8, but showing only the distribution at $z = 0.03$. Each panel refers to a simulation set where only one parameter has been changed, namely from top to bottom: the opening angle θ_{geom} used in the geometrical merging criterion, the parameter α_{dyn} used in the dynamical de-refinement condition, the size of the buffer zone b (in units of the mean interparticle distance λ), and the maximum length of de-refined tree nodes l_{max} (effectively enforcing a minimum background resolution). Each line corresponds to a different value of the parameter investigated, with the most conservative choice on top and the least one on the bottom of the legend.

results we present in the following should be considered indicative of the trends but not necessarily independent of other simulation parameters. From our tests, some features appear clear:

(i) The accuracy is only weakly dependent on the geometrical opening angle (top panel of Fig. 10), with the exception of the lowest (i.e. most conservative) value $\theta_{geom} = 0.01$. This is the case because for such a low value most of the simulation volume remains at the original resolution for most of the time, in contrast with the situation occurring for larger θ_{geom}. This can be seen also in Table 2, where the run with $\theta_{geom} = 0.01$ at $z = 0$ still contains 32 per cent of the initial particles, in contrast with $\theta_{geom} = 0.1$ (or even $\theta_{geom} = 0.05$, not reported for simplicity) where only ~ 1 per cent of the original particles are left. In fact, an intermediate behaviour is observed for $\theta_{geom} = 0.03$. Finally, we note here that almost all the values of θ_{geom} investigated are smaller than those typically employed in TREE codes (where $\theta \gtrsim 0.7$), but this choice is dictated by the fact that DZS permanently changes the simulations and, for any given particle, such modifications are anisotropic. Both reasons call for a more conservative value of θ to be employed.

Table 2. Particle (N_p^{fin}/N_p^{init}) and time ($T_{wc}^{DZS}/T_{wc}^{STD}$) reduction for different parameters choices in the DZS algorithm. Each table refers to run of the LARGE simulation where a single different parameter has been changed with respect to the fiducial configuration used throughout the paper (highlighted in the Table by †), namely from top to bottom: geometrical opening angle θ_{geom}, parameter α_{dyn} used in the dynamical opening criterion, buffer size b, and maximum length l_{max} allowed for a node to be de-refined (effectively enforcing a minimum background resolution). Note that in our fiducial configuration, the maximum theoretical gain in the particle number is $N_p^{fin}/N_p^{init} = 0.0157$ because of the background resolution enforced.

| θ_{geom}	N_p^{fin}/N_p^{init}	$T_{wc}^{DZS}/T_{wc}^{STD}$
0.01	0.3180	0.9196
0.03	0.0343	0.7042
0.1†	0.0165	0.6372
0.2	0.0158	0.6250
0.5	0.0157	0.6278
1.0	0.0157	0.6043
α_{dyn}	N_p^{fin}/N_p^{init}	$T_{wc}^{DZS}/T_{wc}^{STD}$
1×10^{-4}	0.0214	0.7404
1×10^{-3}	0.0169	0.6859
1×10^{-2}	0.0160	0.6631
1×10^{-1}	0.0157	0.6295
b	N_p^{fin}/N_p^{init}	$T_{wc}^{DZS}/T_{wc}^{STD}$
100	0.0730	0.7982
10	0.0170	0.6443
5†	0.0165	0.6148
2	0.0163	0.6850
1	0.0162	0.6130
0	0.0162	0.6202
l_{max}/λ	N_p^{fin}/N_p^{init}	$T_{wc}^{DZS}/T_{wc}^{STD}$
2	0.7622	0.9863
4†	0.0165	0.6291
8	0.0031	0.6026
42	0.0016	0.6267
512	0.0016	0.6040

Note. †This is the standard value of the parameter in this Paper.

(ii) When the dynamical de-refinement criterion is employed, the integration accuracy depends less, but more steadily, on the α_{dyn} parameter value.

(iii) The buffer size has no effect on the accuracy, with the exception of $b = 100$. As for $\theta_{geom} = 0.01$, the reason is that such a large buffer size (corresponding to 1/5 of the box size) prevents many particles outside of the light-cone from being de-refined (7.3 per cent of the initial particles are left at $z = 0$, in contrast to 1.7 per cent for $b = 10$).

(iv) The background resolution appears to be an important parameter in setting the integration accuracy. In fact, limiting the de-refined node size to be at most $l_{max}/\lambda = 2$ (corresponding to an average mass ratio of 8 between low- and high-resolution particles) yields an excellent accuracy (with a maximum $\Delta x/h \approx 10^{-3}$ at $z = 0$). Allowing for an extra 'level' of de-refinement (i.e. $l_{max}/\lambda = 4$, corresponding to a mass ratio of 64) deteriorates the accuracy by more than one order of magnitude. Further increasing l_{max}/λ only slightly worsens the accuracy (up to a factor of 2 using $l_{max} = L_{box}$, i.e. $l_{max}/\lambda = 512$).
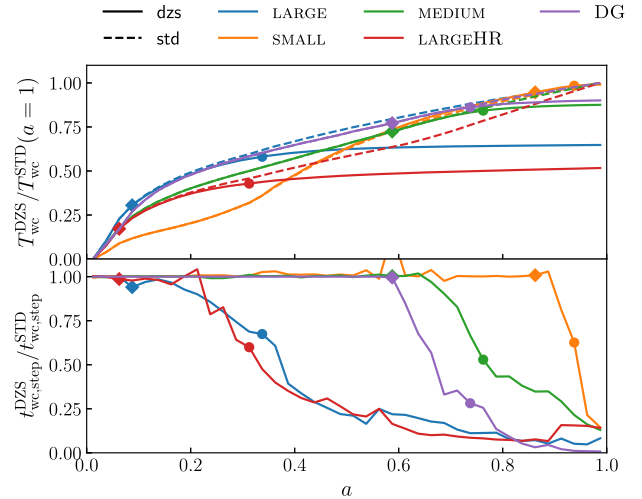
Figure 11. Top: Cumulative simulation wall-clock time – normalized to its final value – as a function of expansion factor for the *dzs* (solid line) and *std* (dashed line) runs of the Llinares suite and the DG re-run. Bottom: Ratio of wall-clock time per time-step in each pair of twin runs. In both panels, diamond symbols mark the time when particles are de-refined for the first time. Similarly, circles indicate when the total number of particles in the *dzs* simulation drops to half its initial value.

4 PERFORMANCE

After assessing the accuracy of the DZS algorithm, we now move to assess its performance. We do so first using our test suite, and then extrapolating its results to realistic next-generation simulations. Before delving into this topic, let us note that throughout the paper we employ a very simple numerical setup. Namely, we only simulate DM particles and ignore baryons. We also reduce to the bare minimum the operations performed on the simulated particles and the output processing (e.g. we do not compute the particle potential, we do not run an on-the-fly halo finder, etc.). All these operations typically become more expensive towards low redshift (either because of structure collapse or because of the time-sampling required is normally finer), where DZS has the largest impact. Hence, all the efficiency boosts presented in the following should be considered as conservative estimates of what can be obtained for more realistic code configurations.

4.1 Performance in our test suite

We start by investigating the performance of our implementation of DZS in the test suite summarized in Table 1. In the top panel of Fig. 11 we show the wall-clock time $T_{wc}(a)$ required by pairs of twin simulations as a function of the cosmological scale factor a, using solid lines for the *dzs* runs and dashed ones for the *std* runs. In order to compare runs with different characteristics (e.g. number of particles, resolution, number of MPI tasks, etc.) we normalize each curve to the total time required by the *std* run, which is always the largest among the two. For all simulations pairs, the time required by the two runs is exactly the same until the DZS algorithm starts merging together particles, reducing the workload. At first, this reduction is marginal, but it quickly becomes prominent as soon as the number of particles is significantly reduced.

This is shown by the diamond symbols (circles) in Fig. 11, which mark the expansion factor when one (half) of the particles has been de-refined. The exact timing of these events strongly depends on the simulation box size and on the parameters used for DZS. Finally,
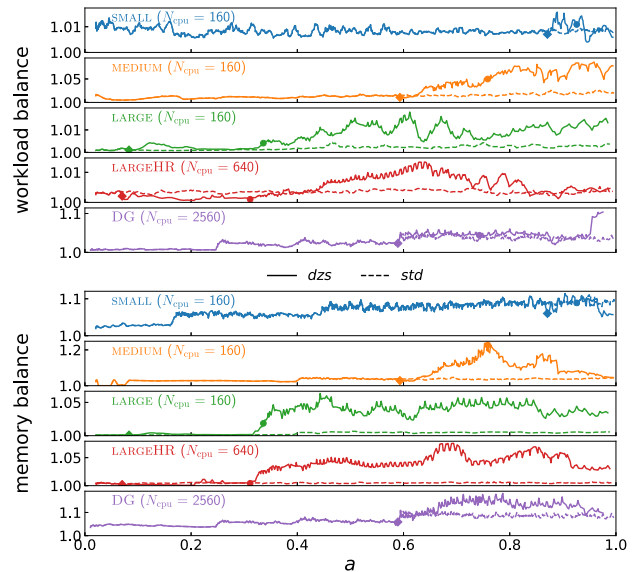
note that in the case of the DG re-run we have started the simulation pair from the most advanced snapshot available before the light-cone enters the simulation box, i.e. $a \approx 0.6$, in order to save computational resources. Before that time, only a single run has been carried out, as DZS could not have any impact.

The impact of DZS on the global simulation time also depends on the evolution of the computational cost. For instance, it is well-known that, at sufficiently-high resolution, matter clustering tends to increase the wall-clock time required per simulated time towards low redshift. However, its impact depends on many algorithmic and physical parameters. Therefore, in order to provide a more general measure of performance, we show in the bottom panel of Fig. 11 the ratio between the time employed by the *dzs* and *std* runs to advance the simulation by a (fixed) interval in expansion factor[8] $\Delta a \approx 0.025$ (labelled as 'step' in the figure for the sake of brevity). As soon as the number of de-refined particles starts to be significant (of the order of 30 per cent of the total), the wall-clock time per step drops dramatically, quickly reaching <10 per cent of its initial value.

We note here that when the simulation resolution increases, low-redshift time-steps typically become increasingly expensive. Hence, the late-time boost in computational efficiency provided by DZS is expected to become more and more effective when the resolution is increased (as we will detail more quantitatively below), rendering DZS a promising tool for the next generation of large cosmological simulations. A first indication of this behaviour can be seen by comparing, in the top panel of Fig. 11, the LARGE and LARGEHR simulation pairs (which have the same box size, i.e. the same onset of the DZS de-refinement procedure, and differ only for their resolution). The latter shows a significantly higher efficiency than the former, despite their per-time-step boost (bottom panel of the same figure) being very similar. We will provide an estimate of the potential efficiency boost for such kind of simulations in Section 4.2. However, increased resolution entails an increased complexity in the numerical treatment (e.g. increased number of computing tasks and communication) which can partially reduce the efficiency, especially when the particle distribution is greatly altered during the simulation, as it is the case for DZS. Hence, in the following we analyse the effect of DZS on the code workload and memory balance.

4.1.1 Memory and workload balance

Fig. 12 shows the workload (top panels) and memory (bottom panels) balance for our twin simulations, defined as the maximum across all tasks of, respectively, the time spent on the TREE walk and the memory allocated, divided by their average value. Hence a value of 1 corresponds to an ideal balance in both cases. (Note that for visualization purposes, the panels cover slightly different ranges along the vertical axis). Each panel shows results concerning one pair of the twin simulations listed in Table 1, using a solid (dashed) line for the *dzs* (*std*) run. We also report in the top left-hand corner of each panel the number of computing tasks $N_{\rm cpu}$ used for the twin runs, since it plays an important role in determining the imbalance. As expected, as soon as DZS begins to systematically de-refine particles the two curves diverge and the *dzs* run has an increased imbalance. This is an unavoidable effect since particles are de-refined based on their position, upon which depends also their location on computing tasks. However, the native domain decomposition algorithm of the PGADGET-3

---

[8]We choose this approach since DZS modifies the particle masses and distribution, and therefore the simulation time-steps differ, preventing a step-to-step comparison.

Figure 12. Workload balance (top panels) and memory balance (bottom panels) evolution with simulated expansion factor for all the runs listed in Table 1 (with the exception of TINY). The solid curves refer to the *dzs* run, while dashed lines to *std* runs. As in Fig. 4, we mark with a diamond symbol the expansion factor when the first de-refinement takes place, and with a circle when the total particle number drops to half its initial value. The curves are smoothed using a running average with window size equal to five time-steps.

code, based on a space-filling Peano–Hilbert curve, appears able to efficiently re-adjust the particle distribution and keep the imbalance under control. In fact, in all the *dzs* runs the workload balance always remains below 1.1, and in most cases below 1.02. Similarly, the memory balance is always below 1.1 in all but two cases. Additionally, the imbalance level reached entirely depends on the details of the algorithm employed to keep it under control, and therefore can largely change among different codes. Hence, the number reported here could be improved upon by different simulation codes.

It is interesting to note how in the SMALL run, the effect of DZS on the workload and memory balance is negligible, as a consequence of the matter clustering (more evident in this simulation pair since it reaches the largest mass resolution) imposing a more substantial toll on the simulation balance than the de-refinement effect of DZS. Indeed, the memory balance starts very close to its ideal value at $a \sim 0$ but increases to its final value well before the DZS algorithm begins de-refining particles ($a \gtrsim 0.8$, marked with a diamond in all panels of Fig. 12).

The simulation performing the worst in terms of both workload and memory balance is MEDIUM, which combines a relatively large box (i.e. allowing DZS to be active for long time) and a decent mass resolution, enabling the formation of a sizeable number cosmic structures. Similarly, the DG re-run, which has a comparable size but better mass resolution, reaches an analogous level of memory imbalance, but its workload balance tracks the one of the *std* run for most of the simulation time, surging only at the very end when few particles are left.

It should be noted that memory imbalance alone is somewhat less of an issue when DZS is employed. Typically, the main concern in having an imbalanced memory load is that it forces the usage of more computing tasks than needed. In fact, each one of them needs to have sufficient memory to store the maximum amount of information any task can be assigned, which is much larger than the average

if the imbalance is high. Beyond the increased time spent by the code in communication, this is particularly problematic whenever the amount of computational resources is limited. However, when such imbalance is triggered by DZS, the amount of information stored in the simulation decreases at the same time. Hence, even the increased imbalance is always associated with a smaller total memory requirement.

Although our tests show that PGADGET-3 is able to limit the imbalance to few per cent, a level not expected to affect the code performance, it is difficult to precisely predict the workload imbalance reached by large-box high-resolution simulation. In any case, even if the workload balance remained always under control during the whole numerical integration, the effect of the DZS technique necessarily implies that the computational requirements of a simulation will change during the run, as DZS progressively reduces the amount of information stored and, hence, diminishes the required total memory. Adjusting the computational setup to the evolving needs of a simulation can prevent resources from being wasted because of almost-empty computing tasks which spend a significant fraction of time communicating or waiting, a feature particularly useful when the amount of resources is limited. For these reasons, we have included in our PGADGET-3 implementation of DZS a series of conditions that force the code to stop and produce an output whenever the imbalance is too large. This will allow users to modify the number of computing tasks the code is run on (and/or other numerical parameters) in order to dynamically optimize the allocation of resources required for continuing the simulation. In particular, we force the code to stop whenever one of the following conditions is met:

(i) the workload balance exceeds a user-defined threshold;

(ii) the ratio between the current and initial number of particles per task $\mathcal{R}(t) \equiv N_{\mathrm{part}}(t)/N_{\mathrm{part}}^{\mathrm{init}}$ is below a user-defined threshold;

(iii) once the code has reached its run-time limit, and $\mathcal{R}(t)$ is predicted (using a linear extrapolation) to fall below the user-defined threshold within a user-defined time from the next simulation re-start.

The latter condition is devised to avoid a simulation from being forced to stop soon after it re-started. In this case, it is more efficient to slightly relax the halt criterion and enforce the special output earlier in order to allow the simulations parameters to be adjusted before requesting a new (and more appropriate) allocation of resources.

4.2 Performance estimation for next-generation simulations

The simulations employed in this work were devised to allow the exploration of different parameter choices for the DZS algorithm, as well as different numerical setups. For this reason, a compromise had to be made between the mass resolution achieved and the size of the simulation. However, the expectation is that DZS will become increasingly more efficient with increasing simulation volume and mass resolution. While we plan to carry out simulations in this regime in a future work, we want to estimate here the performance boost that can be expected from DZS for an arbitrary simulation setup.

In order to have realistic estimations, we need to take into account the effect of small-scale clustering of dark matter particles, which significantly increases the computation time and is particularly prominent at low redshift. We do so by running a simulation with small box but high particle resolution (compared to the other simulations employed in this work), namely $L_{\mathrm{box}} = 128\,h^{-1}$ Mpc and $N_{\mathrm{part}} = 512^3$ (dubbed TINY), and using its normalized wall-clock time evolution $T_{\mathrm{wc}}(a)/T_{\mathrm{wc}}(1)$ as a template for a simulation with larger box size but equal mass resolution. Notice that the small box size implies
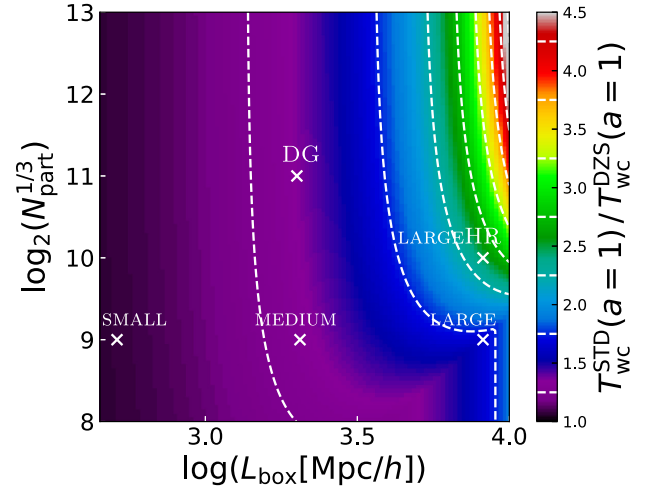
Figure 13. Predicted speedup enabled by DZS as a function of box size (L_{box}) and number of particles(N_{part}). Crosses show where our test simulations lie in this plane (note that TINY is outside of the plotted range). This is a qualitative prediction, see the text for caveats. The dashed lines show contours of equal gain, as indicated in the colourbar.

that large-scale density fluctuations are missing, overall suppressing the densest regions. At the same time, the box is too small to include the most massive structures in the Universe. Both effects imply that such simulation underestimates the late-time matter clustering, and hence the performance boost estimated using it is likely smaller than what can be achieved in a realistic setup.

It can be seen in Fig. 11 that shortly after DZS has de-refined a significant number of particles, the computational cost of the simulation grows only very mildly, contributing only to a small part of the final wall-clock time. To first order, we can approximate this by assuming that a run with DZS enabled has the same cost as one without it until some critical expansion factor a_{DZS}, and 0 afterwards, i.e.:

$$\frac{T_{\mathrm{wc}}^{\mathrm{STD}}(a=1)}{T_{\mathrm{wc}}^{\mathrm{DZS}}(a=1)} = \frac{T_{\mathrm{wc}}^{\mathrm{STD}}(a=1)}{T_{\mathrm{wc}}^{\mathrm{STD}}(a_{\mathrm{DZS}})} \ . \tag{4}$$

Note that this quantity is the reciprocal of the one plotted in the top panel of Fig. 11. In order to be conservative in our estimate, we can use as critical expansion factor the one at which the light-cone volume is half of the simulation one (approximately corresponding to the circles in Fig. 11). Using TINY as a template for $T_{\mathrm{wc}}^{\mathrm{STD}}(a)$ we can estimate the performance boost for a simulation similar to the Euclid flagship one, i.e. $L_{\mathrm{box}} = 4000\,h^{-1}$ Mpc and $N_{\mathrm{part}} = 4 \times 10^{12}$ (Potter et al. 2017) to be *at least*

$$\frac{T_{\mathrm{wc}}^{\mathrm{STD}}(a=1)}{T_{\mathrm{wc}}^{\mathrm{DZS}}(a=1)} = \frac{T_{\mathrm{wc}}^{\mathrm{STD}}(a=1)}{T_{\mathrm{wc}}^{\mathrm{STD}}(\approx 0.12)} \approx 3 \ . \tag{5}$$

We stress again that this is likely to be a severe under-estimation of the actual achievable speedup.

In a similar way, we can estimate the performance boost for a simulation with the same characteristics as the DEUS 'full Universe' run (Alimi et al. 2012) to be $\gtrsim 10$. However, it must be noted that such simulation is very peculiar, as by construction it encloses the light-cone from the very beginning. This represents a configuration that we have not tested, so it may present a slightly different behaviour, hence altering the estimated performance boost.

In order to give a qualitative overview of the speed-up enabled by DZS, we show in Fig. 13 the predicted ratio of wall-clock times for a simulation without and with DZS (the reciprocal of the quantity

plotted on the vertical axis of the top panel of Fig. 11) as a function of box size (L_{box}) and number of particles(N_{part}). In order to compute this quantity, we employ the same approach used above for the *Euclid* flagship simulation. However, we need to account for the different shape of $T_{\text{wc}}^{\text{STD}}(a)$ as a function of mass resolution (due to matter clustering at low redshift having a larger impact at higher mass resolution). We do so in an empirical way, noticing that in our lowest resolution simulation (LARGE) such curve is well approximated by $a^{0.4}$, while in the highest resolution one (TINY) it is approximately a straight line (i.e. $\approx a$). Hence, we approximate the shape of $T_{\text{wc}}^{\text{STD}}(a)$ with a curve a^r, i.e. $T_{\text{wc}}^{\text{STD}}(a)/T_{\text{wc}}^{\text{STD}}(a=1) \approx a^r$, where r is linearly interpolated between the value $r = 0.4$ at our coarsest mass resolution and $r = 1$ at the highest one. For better mass resolutions, the value of r is extrapolated, while for coarser resolutions it is kept fixed at $r = 0.4$, since the clustering is expected to be negligible in such regime. Hence, we end up with the following estimate:

$$\frac{T_{\text{wc}}^{\text{STD}}(a=1)}{T_{\text{wc}}^{\text{DZS}}(a=1)} \approx \frac{T_{\text{wc}}^{\text{STD}}(a=1)}{T_{\text{wc}}^{\text{STD}}(a_{\text{DZS}})}$$
$$\approx a_{\text{DZS}}^{-\max\left(0.4, 0.4 + \frac{1-0.4}{m_{\text{SMALL}} - m_{\text{LARGE}}}(m - m_{\text{LARGE}})\right)} \quad (6)$$

$$a_{\text{DZS}} : \frac{4}{3}\pi R_{\text{lc}}(a_{\text{DZS}})^3 = \frac{1}{2}L_{\text{box}}^3 \,, \quad (7)$$

where m in the first equation is the dark matter mass resolution of the simulation (computed from L_{box} and N_{part} assuming our fiducial cosmology). In Fig. 13 we mark with crosses the locations of our test suite runs in such plane. As predicted, DZS brings the largest benefit for large box size and number of particles. It should be noted that for $\log_2(N_{\text{part}}^{1/3}) \gtrsim 10$ the estimated performance boost is approximately independent of N_{part}. This may be partially due to our extrapolation method. Alternatively, it could simply indicate that the reduction in the number of particles (driven by L_{box}) is dominant over the effect of matter clustering due to increasing resolution.

5 CONCLUSIONS

In this paper we have presented a new method – named Dynamic Zoom Simulations, or DZS in short – devised to speed up cosmological simulations while preserving their accuracy in the production of light-cone outputs. Our approach combines seamlessly the relativistic notion of light-cone with the Newtonian nature of typical cosmological simulations in a way that is fully compatible with the latter, preserves the periodicity properties, and does not require any substantial change in the code structure.

The DZS algorithm de-refines particles outside of the observer's light-cone (see Fig. 4) in order to reduce the total computational time while preserving the large scale density field. While this can, in principle, be done in different ways, we employ a TREE-based approach. This ensures scalability, allows a smooth transition between the high- and low-resolution regions, and integrates effortlessly with TREEPM codes. The latter is of particular importance since: (i) it ensures virtually no overhead in the computational time; (ii) it renders the implementation of DZS into existing TREEPM codes relatively simple (as the necessary infrastructure is already in place), and (iii) enables the exploitation of DZS together with the broad range of features that have been implemented in popular simulation codes over the years.

We have tested the accuracy and performance of this approach by implementing it in the popular PGADGET-3 code, and list in the following our main results:

(i) The DZS algorithm is able to reduce the wall-clock time per time step by more than an order of magnitude once the light-cone radius is a fraction of the box size (Fig. 11). This translates into a reduction of the total computation time of order 50 per cent for the simplest configuration we have tested, and likely much more for higher resolution or more complex (i.e. involving more simulated physics and/or outputs) simulations.

(ii) The de-refinement of particles frees up progressively more memory during the simulation. Hence, individual computing tasks progressively store less information, and spend larger fractions of time in communication. This situation is particularly problematic for large simulations and/or with constrained total resources available. For this reason, we have devised and implemented a mechanism that optimizes the halt and re-start of a simulation in order to fine tune the computational resources assigned to the run. Additionally, this could also help reducing the small imbalances in the memory and workload distribution across different tasks (Fig. 12) that can arise as a consequence of the particle de-refinement.

(iii) The comparison between simulations with and without DZS shows that the light-cone halo mass function (Fig. 5), the sky-projected matter light-cone and its angular power spectrum (Fig. 6), and the 3D matter light-cone (Fig. 7) are unchanged to the ~ 0.1 per cent level.

(iv) The integration accuracy is mostly unchanged by the DZS algorithm, while the simulation time is significantly reduced. In our largest box highest resolution test (i.e. where the effect of DZS is maximal) the particle positions (across the entire simulated time) in twin runs with and without DZS differ by at most few times the mean interparticle distance h. More than 99 per cent of the particles are displaced less than $0.2\,h$ (Fig. 8), while the total runtime is halved.

(v) The integration accuracy can be tweaked thanks to a number of parameters specific to DZS, with a consequent change in the wall-clock time. By investigating a broad region of the parameter space (Fig. 10 and Table 2) we provide some guidance on how to optimize the DZS parameters in order to achieve the optimal trade-off between accuracy and performance boost for the application of choice.

(vi) Combining a high-resolution small-box simulation with coarse-resolution large-box ones, we provide an estimation of the efficiency boost that can be expected when employing DZS for any combination of box size and particle number (i.e. mass resolution, Fig. 13). In the region of the L_{box}–N_{part} space that new-generation simulations are starting to explore, we forecast DZS to reduce the total simulation at least by a factor of a few in the case of DM-only simulation, and likely much more so for runs including baryons (although DZS is currently implemented only for DM-only simulations).

To summarize, we have presented a novel approach to the production of light-cones in cosmological simulations. Our Dynamic Zoom Simulation – or DZS – approach de-refines particles once they are not needed anymore for the light-cone production, in order to reduce the workload of the simulation while simultaneously maintaining its output accuracy and the large-scale gravitational field. This allows us to significantly improve the performance with respect to equal simulations carried out with the traditional approach. A number of parameters control the behaviour of DZS, enabling a customization of the accuracy and speed-up reached through its use. In DZS, traditional time-slice output have reduced resolution outside of the light-cone (i.e. in the entire box at $z = 0$). Our implementation allows the user to enforce a minimum volume to be kept at maximum resolution until the end of the simulation, in order to have a high-resolution time-slice of a portion of the simulation box even at $z = 0$.

Overall, our results show that DZS is a promising approach for the development of large high-resolution simulations, as the ones

required to model and interpret the wealth of high-quality data that upcoming surveys will provide.

DATA AVAILABILITY

The data underlying this article will be shared on reasonable request to the corresponding author.

REFERENCES

Alimi J.-M. et al., 2012, preprint (arXiv:1206.2838)
Arnold C., Fosalba P., Springel V., Puchwein E., Blot L., 2019, MNRAS, 483, 790
Bagla J. S., 2002, J. Astrophys. Astron., 23, 185
Bagla J. S., Ray S., 2003, New A, 8, 665
Baldi M., 2012, MNRAS, 422, 1028
Baldi M., Pettorino V., Robbers G., Springel V., 2010, MNRAS, 403, 1684
Bode P., Ostriker J. P., Xu G., 2000, ApJS, 128, 561
Bonafede A., Dolag K., Stasyszyn F., Murante G., Borgani S., 2011, MNRAS, 418, 2234
Clarke L., Glendinning I., Hempel R., 1994, in Decker K. M., Rehmann R. M., eds, Programming Environments for Massively Parallel Distributed Systems. Birkhäuser Basel, Basel, p. 213
Dai B., Feng Y., Seljak U., 2018, J. Cosmol. Astropart. Phys., 2018, 009
DESI Collaboration 2016a, preprint (arXiv:1611.00036)
DESI Collaboration 2016b, preprint (arXiv:1611.00037)
Dolag K., Stasyszyn F., 2009, MNRAS, 398, 1678
Evrard A. E. et al., 2002, ApJ, 573, 7
Fosalba P., Gaztañaga E., Castand er F. J., Manera M., 2008, MNRAS, 391, 435
Górski K. M., Hivon E., Banday A. J., Wandelt B. D., Hansen F. K., Reinecke M., Bartelmann M., 2005, ApJ, 622, 759
Hollowed J., 2019, preprint (arXiv:1906.08355)
Hunter J. D., 2007, Comput. Sci. Eng., 9, 90
Ivezić Ž. et al., 2019, ApJ, 873, 111
Jones E. et al., 2001, SciPy: Open source scientific tools for Python. Available at: http://www.scipy.org/
Klypin A. A., Shandarin S. F., 1983, MNRAS, 204, 891
Laureijs R. et al., 2011, preprint (arXiv:1110.3193)
Leclercq F., Faure B., Lavaux G., Wandelt B. D., Jaffe A. H., Heavens A. F., Percival W. J., 2020, A&A, 639, A91
Llinares C., 2017, preprint (arXiv:1709.04703)
Nelson D. et al., 2019, MNRAS, 490, 3234
Nori M., Baldi M., 2018, MNRAS, 478, 3935
Pawlik A. H., Schaye J., 2008, MNRAS, 389, 651
Petkova M., Springel V., 2009, MNRAS, 396, 1383
Pillepich A. et al., 2019, MNRAS, 490, 3196
Planck Collaboration XIII, 2016, A&A, 594, A13
Potter D., Stadel J., Teyssier R., 2017, Comput. Astrophys. Cosmol., 4, 2
Puchwein E., Baldi M., Springel V., 2013, MNRAS, 436, 348
Spergel D. et al., 2015, preprint (arXiv:1503.03757)
Springel V., 2005, MNRAS, 364, 1105
Springel V., Yoshida N., White S. D., 2001, New Astron., 6, 79
Tassev S., Zaldarriaga M., Eisenstein D. J., 2013, J. Cosmol. Astropart. Phys., 2013, 036
Tassev S., Eisenstein D. J., Wand elt B. D., Zaldarriaga M., 2015, preprint (arXiv:1502.07751)
van der Velden E., 2020, J. Open Source Softw., 5, 2004
Viel M., Haehnelt M. G., Springel V., 2010, J. Cosmol. Astropart. Phys., 2010, 015
Walt S. v. d., Colbert S. C., Varoquaux G., 2011, Comput. Sci. Eng., 13, 22
White S. D. M., Frenk C. S., Davis M., 1983, ApJ, 274, L1
Xu G., 1995, ApJS, 98, 355
Zonca A., Singer L., Lenz D., Reinecke M., Rosset C., Hivon E., Gorski K., 2019, J. Open Source Softw., 4, 1298

This paper has been typeset from a TEX/LATEX file prepared by the author.