MNRAS **000**, 1–24 (2021) Preprint 15 September 2021 Compiled using MNRAS LATEX style file v3.0

The cosmological simulation code CONCEPT 1.0

Jeppe Dakin[1]*, Steen Hannestad[1]†, Thomas Tram[1]‡

[1] *Department of Physics and Astronomy, Aarhus University, Ny Munkegade 120, DK-8000 Aarhus C, Denmark*

15 September 2021

ABSTRACT

We present version 1.0 of the cosmological simulation code CONCEPT, designed for simulations of large-scale structure formation. CONCEPT contains a P^3M gravity solver, with the short-range part implemented using a novel (sub)tiling strategy, coupled with individual and adaptive particle time-stepping. In addition to this, CONCEPT contains a fully non-linear fluid solver to treat non-baryonic components which are not easily treatable using the particle implementation. This allows e.g. for the inclusion of non-linear massive neutrinos (which can be relativistic) and for simulations that are consistent with general relativistic perturbation theory. Decaying dark matter scenarios are also fully supported. A primary objective of CONCEPT is ease of use. To this end, it has built-in initial condition generation and can produce output in the form of snapshots, power spectra and direct visualisations. It also comes with a robust installer and thorough documentation. CONCEPT is the first massively parallel cosmological simulation code written in Python. Despite of this, excellent performance is obtained, even comparing favourably to other codes such as GADGET at similar precision. The CONCEPT code itself along with documentation is openly released at https://github.com/jmd-dk/concept .

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

1 INTRODUCTION

Measurements of inhomogeneities in our Universe have been performed over a vast range of scales, spanning sub-galactic scales all the way to the current horizon. On large scales and at early times the amplitude of density fluctuations is small enough that it can be treated accurately using perturbation theory. However, on smaller scales and at later times this is no longer the case, and structure formation must be evolved through simulation.

The dominant clustering component is cold dark matter which is well described by a collisionless fluid with negligible thermal velocity dispersion. This means that the full 6D phase space distribution, $f(\boldsymbol{r}, \boldsymbol{p})$, can be collapsed to a 3D sheet which can be followed in time. By far the most efficient way of doing this is to use N-body simulations in which the underlying fluid is described by a large number N of discrete particles, each following the appropriate equations of motion. This method has the advantage of being inherently Lagrangian — regions of high density will automatically correspond to regions of high N-body particle count, unlike e.g. solving the fluid equations using a static Eulerian grid.

* E-mail: dakin@phys.au.dk
† E-mail: sth@phys.au.dk
‡ E-mail: thomas.tram@phys.au.dk

Such simulations of cosmic structure formation have a long history, going back to the pioneering work of Hoerner (1960) who proposed to study stellar clusters using N-body methods. The first papers on N-body methods used direct summation to find the individual forces on particles. However, this approach quickly becomes prohibitively expensive for large N, given that it is an $\mathcal{O}(N^2)$ problem.

In order to make the problem tractable a number of numerical schemes have been developed over the years, including tree codes (Barnes & Hut 1986) and particle-mesh (PM) codes (Hockney & Eastwood 1988). Tree methods work by first grouping the particles into nodes in a hierarchical tree structure, which is then 'walked' to some sufficient depth relative to a given particle in order to provide an approximate but cheap estimate of the gravitational force from several other particles at once. In PM codes a density field on a grid is constructed through interpolation of the particles, which is then transformed to the gravitational potential, typically using fast Fourier techniques. The PM method is much faster than direct summation for large N, scaling as $\mathcal{O}(N \log N)$. Though tree codes have a similar scaling $\mathcal{O}(N \log N)$, they are not as fast as PM codes. However, as pure PM codes are restricted by the finite size of the grid cells, this limits their resolution to scales a few times larger than this size.

The shortcomings of the PM method can be mended by augmenting it with direct summation of particle forces

over short distances. This method was first described in Hockney et al. (1974) and applied to a cosmological setting by Efstathiou & Eastwood (1981). It is known as PP-PM, or P^3M (see Hockney & Eastwood (1988); Bertschinger (1998) for reviews).

While P^3M codes work extremely well for large-scale cosmological simulations in which clustering is moderate, the non-hierarchical nature of the short-range force becomes an issue when the matter distribution becomes very uneven, e.g. for a close-up simulation of a single galaxy formation. This serious problem can be circumvented by using either a tree decomposition of the short-range force (as in the TreePM method of GADGET (Springel 2005b)), or by applying adaptive mesh refinement to the PM grid (Couchman 1991).

This paper is about release 1.0 of the CONCEPT code, a massively parallel simulation code for cosmological structure formation. The main goal of any such code is to track the non-linear evolution of matter, which CONCEPT achieves via N-body techniques, i.e. by describing matter as a set of Lagrangian particles. Additionally, CONCEPT allows for any species to be modelled as a fluid, with quantities like energy density, momentum density and pressure being evolved on a spatially fixed, Eulerian grid. This allows for non-standard simulations, such as ones including non-linearly evolved massive neutrinos (Dakin et al. 2019a) and ones fully consistent with general relativistic perturbation theory (Tram et al. 2019; Dakin et al. 2019c). These more exotic aspects of CONCEPT dates back to previous releases and will not be described in detail in this paper.

The main feature new to the 1.0 release of CONCEPT is that of explicit short-range gravitational forces. Previously, the only feasible[1] gravitational method available was that of PM, leaving gravity badly resolved at small scales. In CONCEPT 1.0 the extremely fast PM method is retained, though the default gravitational solver is now that of P^3M, i.e. long-ranged PM augmented with short-ranged direct summation. This newly added short-range force is implemented using an efficient and novel scheme, based on what we call tiles and subtiles. The increased spatial resolution resulting from the added short-range forces calls for a corresponding increase in the temporal resolution, though needed only in regions of high clustering. Thus, CONCEPT 1.0 further comes with a new individual and adaptive particle time-stepping scheme.

The main goal of this paper is threefold: 1) To describe the numerical methods employed by CONCEPT 1.0, 2) to demonstrate the validity of the code by comparing its results to those of other simulation codes, 3) to measure the code performance in terms of both scaling behaviours and absolute comparison to other codes. For the code comparisons, we use the well-known GADGET-2 code (Springel 2005b) as well as its newer incarnation GADGET-4 (Springel et al. 2020).

This paper is structured as follows: In section 2 we describe the numerical methods built into CONCEPT 1.0, with a focus on gravity and time-stepping. Section 3 then goes on to validate the code results, while code performance is explored in section 4. Finally, section 5 provides a summary and a discussion about the usefulness of the code as it currently stands, as well as what might be implemented in the future in order to enhance both its capabilities and performance. In addition, other features and non-standard software aspects of CONCEPT 1.0 are briefly discussed in appendix A.

[1] An inefficient implementation of P^3M has in fact been available for years. The basic PP method was (and still is) available as well, though due to its $\mathcal{O}(N^2)$ scaling this is intended only for internal testing.

2 NUMERICAL METHODS

This section describes the main numerical methods and implementations used in CONCEPT 1.0, responsible for the gravitational interaction between matter particles and their resulting temporal evolution.

The basic setup of CONCEPT is that of a cubic, toroidal periodic box of constant comoving side length L_{box}, containing N matter particles of equal mass m, each having a comoving position $\boldsymbol{x}_i(t)$ and canonical momentum $\boldsymbol{q}_i(t)$, evolving under self-gravity in an expanding background, captured by the cosmological scale factor $a(t)$, with t being cosmic time. The code is parallelised using the Message Passing Interface (MPI), with the box divided into equally shaped cuboidal domains — one per process — which in turn are mapped one-to-one to physical CPU cores.

The equations of motion for the particles are fully written out in section 2.2. Before that, section 2.1 sets out to find the comoving gravitational force $\boldsymbol{f}_i$, the only force considered; $\partial_t \boldsymbol{q}_i \equiv \boldsymbol{f}_i/a$.

2.1 Gravity

This subsection develops the gravitational solvers available in CONCEPT 1.0, starting with the PP and PM method and culminating in the P^3M method. While the CONCEPT 1.0 implementations of PP and PM does not deviate much from standard procedures, the P^3M implementation is novel.

2.1.1 PP gravity

The particle-particle (PP) method solves gravity via direct summation over pairwise interactions. This direct approach makes the PP method essentially exact, but comes at the cost of $\mathcal{O}(N^2)$ complexity, drastically limiting its usability. Regardless, the PP method is worth studying in detail as it introduces many aspects used for the superior P^3M method.

From particles to fields For a set of N point particles in infinite space one could simply use Newton's law of universal gravitation. As we seek more flexibility we shall instead think in terms of the peculiar potential φ, defined through the Poisson equation (Peebles 1980)

$$\nabla^2\varphi(\boldsymbol{x}) = 4\pi G a^2 \delta\rho(\boldsymbol{x})\,, \tag{1}$$

where G is the gravitational constant and the Laplacian is to be taken with respect to comoving space $\boldsymbol{x} \equiv \boldsymbol{r}/a(t)$, $\boldsymbol{r}$ being physical space. The physical density contrast field $\delta\rho(\boldsymbol{x})$ is constructed from the particles by assigning them a localised shape $S(\boldsymbol{x})$, so that

$$\delta\rho(\boldsymbol{x}) = \frac{m}{a^3}\sum_{\boldsymbol{n}\in\mathbb{Z}^3}\left\{-\frac{N}{L_{\text{box}}^3} + \sum_{j=1}^{N} S(\boldsymbol{x}-\boldsymbol{x}_{jn})\right\}, \tag{2}$$

with $\boldsymbol{x}_{jn} \equiv \boldsymbol{x}_j + L_{\text{box}}\boldsymbol{n}$ and the periodicity of the box implemented by the sum over all integer triplets $\boldsymbol{n}$. The subtraction of N times the reciprocal box volume ensures that $\delta\rho(\boldsymbol{x})$ averages to zero, assuming the shape S to be normalised to unity.

For point particles, $S(\boldsymbol{x}) \to S_\delta(\boldsymbol{x}) \equiv \delta^3(\boldsymbol{x})$, δ^3 being the three-dimensional Dirac delta function. Given a shape, (1) and (2) can be solved for the potential;

$$\varphi(\boldsymbol{x}) = -\frac{Gm}{a}\sum_{j=1}^{N}\sum_{\boldsymbol{n}\in\mathbb{Z}^3} |\boldsymbol{x} - \boldsymbol{x}_{jn}|_{\bullet}^{-1}\,, \tag{3}$$

where we have introduced the generalised reciprocal distance $|\boldsymbol{x}|_{\bullet}^{-1}$, the subscript denoting an arbitrary shape. For the choice of point particles S_δ, we simply have $|\boldsymbol{x}|_{\bullet}^{-1} \to |\boldsymbol{x}|_{\delta}^{-1} = |\boldsymbol{x}|^{-1}$. The comoving force on particle i, $\boldsymbol{f}_i = -am\,\nabla\varphi(\boldsymbol{x})|_{\boldsymbol{x}=\boldsymbol{x}_i}$, is then

$$\boldsymbol{f}_i = -Gm^2\sum_{\substack{j=1\\ j\neq i}}^{N}\sum_{\boldsymbol{n}\in\mathbb{Z}^3} |\boldsymbol{x}_{ijn}|_{\bullet}^{-3}\boldsymbol{x}_{ijn}\,, \tag{4}$$

where $\boldsymbol{x}_{ijn} \equiv \boldsymbol{x}_i - (\boldsymbol{x}_j + L_{\text{box}}\boldsymbol{n})$ and the divergence at $j = i$ has been removed. For point particles $|\boldsymbol{x}|_{\delta}^{-3} = (|\boldsymbol{x}|_{\delta}^{-1})^3 = |\boldsymbol{x}|^{-3}$.

Softening As the N tracer particles are meant to represent an underlying continuous density field, it is desirable to soften the force by choosing a particle shape that is more spread out, dampening the effects of two-body interactions. A simple choice is that of a Plummer sphere (Plummer 1911);

$$S_{\text{P}}(\boldsymbol{x}) \equiv \frac{3}{4\pi\epsilon^3}\left(1 + \frac{\boldsymbol{x}^2}{\epsilon^2}\right)^{-5/2} \tag{5}$$

$$\Rightarrow |\boldsymbol{x}|_{\text{P}}^{-1} = \left(\boldsymbol{x}^2 + \epsilon^2\right)^{-1/2}\,, \tag{6}$$

where $\epsilon \geq 0$ is the softening length, typically chosen to be a few percent of the mean inter-particle distance $L_{\text{box}}/\sqrt[3]{N}$. Substituting $|\boldsymbol{x} - \boldsymbol{x}_{jn}|_{\bullet}^{-1}$ for $|\boldsymbol{x} - \boldsymbol{x}_{jn}|_{\text{P}}^{-1}$ into (3) and $|\boldsymbol{x}_{ijn}|_{\bullet}^{-3}$ for $|\boldsymbol{x}_{ijn}|_{\text{P}}^{-3} = (|\boldsymbol{x}_{ijn}|_{\text{P}}^{-1})^3$ into (4) then results in the Plummer softened potential and force, respectively.

Ideally we would like the softening to vanish for large particle separation, i.e. we seek a shape with compact support. Though CONCEPT 1.0 implements both the point particle S_δ and the Plummer sphere S_{P}, the default softening shape is the B-spline of Monaghan & Lattanzio (1985), as also used by GADGET:

$$S_{\text{B}}(\boldsymbol{x}) \equiv \frac{8}{\pi\epsilon_{\text{B}}^3}\begin{cases} 1 - 6x_{\text{B}}^2(1 - x_{\text{B}}) & x_{\text{B}} < \frac{1}{2} \\ 2(1 - x_{\text{B}})^3 & \frac{1}{2} \leq x_{\text{B}} < 1 \\ 0 & 1 \leq x_{\text{B}} \end{cases} \tag{7}$$

$$\Rightarrow |\boldsymbol{x}|_{\text{B}}^{-1} = \begin{cases} \frac{32}{\epsilon_{\text{B}}}\Big(-\frac{1}{5}x_{\text{B}}^5 + \frac{3}{10}x_{\text{B}}^4 - \frac{1}{6}x_{\text{B}}^2 + \frac{7}{80}\Big) & x_{\text{B}} < \frac{1}{2} \\ \frac{32}{\epsilon_{\text{B}}}\Big(\frac{1}{15}x_{\text{B}}^5 - \frac{3}{10}x_{\text{B}}^4 + \frac{1}{2}x_{\text{B}}^3 - \frac{1}{3}x_{\text{B}}^2 + \frac{1}{10} - \frac{1}{480}x_{\text{B}}^{-1}\Big) & \frac{1}{2} \leq x_{\text{B}} < 1 \\ |\boldsymbol{x}|^{-1} & 1 \leq x_{\text{B}}\,, \end{cases} \tag{8}$$

$$|\boldsymbol{x}|_{\text{B}}^{-3} = \begin{cases} \frac{32}{\epsilon_{\text{B}}^3}\Big(x_{\text{B}}^3 - \frac{6}{5}x_{\text{B}}^2 + \frac{1}{3}\Big) & x_{\text{B}} < \frac{1}{2} \\ \frac{32}{\epsilon_{\text{B}}^3}\Big(-\frac{1}{3}x_{\text{B}}^3 + \frac{6}{5}x_{\text{B}}^2 - \frac{3}{2}x_{\text{B}} + \frac{2}{3} - \frac{1}{480}x_{\text{B}}^{-3}\Big) & \frac{1}{2} \leq x_{\text{B}} < 1 \\ |\boldsymbol{x}|^{-3} & 1 \leq x_{\text{B}}\,, \end{cases} \tag{9}$$

where $x_{\text{B}} \equiv |\boldsymbol{x}|/\epsilon_{\text{B}}$ and ϵ_{B} is the B-spline softening length. Note that the symbol $|\boldsymbol{x}|_{\text{B}}^{-3} \neq (|\boldsymbol{x}|_{\text{B}}^{-1})^3$. Equations (8) and (9) then define the B-spline softened potential and force via (3) and (4), respectively. As in Springel (2005b) we set $\epsilon_{\text{B}} = 2.8\epsilon$, keeping the Plummer softening length ϵ as the canonical softening parameter.

Ewald summation The triply infinite sums of (3) and (4) can be evaluated using the technique of Ewald (1921) (see also Hernquist et al. (1991)). This amounts to writing the functional part of the potential (3) — i.e. the reciprocal distance — as a sum of a short-range and a long-range part; $|\boldsymbol{x}|^{-1} = \mathcal{G}_{\text{sr}}(\boldsymbol{x}) + \mathcal{G}_{\text{lr}}(\boldsymbol{x})$. We employ the common choice $\mathcal{G}_{\text{sr}}(\boldsymbol{x}) = \text{erfc}(|\boldsymbol{x}|/[2x_{\text{s}}])|\boldsymbol{x}|^{-1}$, $\mathcal{G}_{\text{lr}}(\boldsymbol{x}) = \text{erf}(|\boldsymbol{x}|/[2x_{\text{s}}])|\boldsymbol{x}|^{-1}$, where $x_{\text{s}} \geq 0$ is the short-/long-range force split scale. Transforming the long-range part to Fourier space[2], $\mathcal{G}_{\text{lr}}(\boldsymbol{k}) = 4\pi\exp(-x_{\text{s}}^2\boldsymbol{k}^2)/\boldsymbol{k}^2$, the potential may be written

$$\varphi(\boldsymbol{x}) = -\frac{Gm}{a}\sum_{j=1}^{N}\left\{\begin{array}{l}\displaystyle\sum_{\boldsymbol{n}\in\mathbb{Z}^3}\Big[\mathcal{G}_{\text{sr}}(\boldsymbol{x} - \boldsymbol{x}_{jn}) \\ \quad + \left(|\boldsymbol{x} - \boldsymbol{x}_{jn}|_{\bullet}^{-1} - |\boldsymbol{x} - \boldsymbol{x}_{jn}|^{-1}\right)\Big] \\ \quad - L_{\text{box}}^{-3}\displaystyle\sum_{\boldsymbol{h}\in\mathbb{Z}^3\setminus\boldsymbol{0}}\mathcal{G}_{\text{lr}}(\boldsymbol{k}_{\boldsymbol{h}})\cos(\boldsymbol{k}_{\boldsymbol{h}}[\boldsymbol{x} - \boldsymbol{x}_j])\end{array}\right\}, \tag{10}$$

with $\boldsymbol{k}_{\boldsymbol{h}} \equiv 2\pi L_{\text{box}}^{-1}\boldsymbol{h}$. In (10) the softening is implemented by the parenthesis in the real-space sum over $\boldsymbol{n}$, ensuring that only the Newtonian part of the potential is softened, which

[2] In an attempt to minimise notational clutter, Fourier-space quantities are distinguished from their real-space counterparts through their argument only.

decouples the choice of softening from the choice of how the potential has been split. For the B-spline softening (8) with compact support, this parenthesis vanishes for all images $\boldsymbol{n}$ of particle j but that closest to $\boldsymbol{x}$, meaning that softening is only applied to the nearest image.

Figure 1 depicts a simulation box with particles, along with various numerical aspects. For the top left particle, three single-particle potentials are shown: The unsoftened Newtonian potential $\propto |\boldsymbol{x}|^{-1}$, the softened Newtonian potential $\propto |\boldsymbol{x}|_{\mathrm{B}}^{-1}$ and the softened short-range potential[3] $\propto \mathcal{G}_{\mathrm{sr}}(\boldsymbol{x}) + (|\boldsymbol{x}|_{\mathrm{B}}^{-1} - |\boldsymbol{x}|^{-1})$. It is clearly seen how the softening removes the divergence near the particle — without changing the potential further out for this case of B-spline softening — and that the short-range potential tends to zero much more rapidly than the Newtonian potentials. We shall come back to Figure 1 several more times, referring to different aspects.

Given the Ewald prescription of the potential (10), the comoving force on particle i becomes

$$\boldsymbol{f}_i = -Gm^2 \sum_{\substack{j=1 \\ j\neq i}}^{N} \left\{ \begin{aligned} &\sum_{\boldsymbol{n}\in\mathbb{Z}^3} \left[\begin{aligned} &|\boldsymbol{x}_{ijn}|^{-3} \operatorname{erfc}\left(\frac{|\boldsymbol{x}_{ijn}|}{2x_{\mathrm{s}}}\right) \\ &+ \frac{|\boldsymbol{x}_{ijn}|^{-2}}{\sqrt{\pi}x_{\mathrm{s}}} \exp\left(-\frac{\boldsymbol{x}_{ijn}^2}{4x_{\mathrm{s}}^2}\right) \\ &+ \left(|\boldsymbol{x}_{ijn}|_{\bullet}^{-3} - |\boldsymbol{x}_{ijn}|^{-3}\right) \end{aligned} \right] \boldsymbol{x}_{ijn} \\ &+ \frac{4\pi}{L_{\mathrm{box}}^3} \sum_{\boldsymbol{h}\in\mathbb{Z}^3\setminus\boldsymbol{0}} \frac{\exp(-x_{\mathrm{s}}^2\boldsymbol{k}_{\boldsymbol{h}}^2)}{\boldsymbol{k}_{\boldsymbol{h}}^2} \sin(\boldsymbol{k}_{\boldsymbol{h}}[\boldsymbol{x}_i - \boldsymbol{x}_j])\boldsymbol{k}_{\boldsymbol{h}} \end{aligned} \right\}, \tag{11}$$

where again the softening term $(|\boldsymbol{x}_{ijn}|_{\bullet}^{-3} - |\boldsymbol{x}_{ijn}|^{-3})$ vanishes for all images $\boldsymbol{n}$ of particle j but the one closest to particle i, in the case of B-spline softening.

The crux of the Ewald technique is that the infinite sums of (10) and (11) converge exponentially, whereas the original infinite sums of (3) and (4) converge much more slowly and in fact only conditionally (de Leeuw et al. 1980). For some chosen x_{s} the Ewald sums can then safely be truncated at some finite maximum $|\boldsymbol{n}|$ and $|\boldsymbol{h}|$. For the PP method CONCEPT uses the values suggested by Hernquist et al. (1991);

$$\begin{cases} x_{\mathrm{s}} = \dfrac{L_{\mathrm{box}}}{4}\,, \\ |\boldsymbol{x}_{ijn}| < 3.6L_{\mathrm{box}}\,, \\ \boldsymbol{h}^2 < 10\,, \end{cases} \tag{12}$$

as do GADGET-2.

Despite having limited the infinite Ewald sums to a doable number of terms (12), the force computation for each particle pair $\{i, j\}$ — corresponding to the large brace of (11) — is still substantial. In practice, CONCEPT precomputes this force for a cubic grid of particle separations between 0 and $L_{\mathrm{box}}/2$ in all three dimensions, with the softened contribution $|\boldsymbol{x}_{ijn}|_{\bullet}^{-3}$ excluded. During simulation, forces are then obtained using CIC interpolation (covered in section 2.1.2) in this grid, with particle separations outside the tabulated region handled using symmetry conditions. The softened $|\boldsymbol{x}_{ijn}|_{\bullet}^{-3}\boldsymbol{x}_{ijn}$ from the nearest image is then added. By default a grid size[4] of 64 is used for the Ewald grid.

2.1.2 PM gravity

Though the path towards the softened, Ewald-assisted periodic force (11) went through the potential φ, this potential itself is never actually computed by CONCEPT when using the PP method. The particle-mesh (PM) method takes a different approach, establishing φ as a cubic grid of size n_φ, from which particle forces are obtained via numerical differentiation and interpolation. The most expensive step of this method is the creation of φ, which in CONCEPT is based on fast Fourier transforms (FFTs). Assuming (very reasonably) that the number of grid elements $n_\varphi^3 \propto N$, the PM method then inherits the $\mathcal{O}(N \log N)$ complexity of the FFT (Cooley & Tukey 1965), vastly outperforming the $\mathcal{O}(N^2)$ PP method. The price to pay is that of a limited resolution of gravity imposed by the finite size of the grid cells $L_\varphi = L_{\mathrm{box}}/n_\varphi$, which in practice is much larger than the particle softening length ϵ of the PP method.

We can explicitly solve the Poisson equation (1) for the potential,

$$\varphi(\boldsymbol{x}) = -Ga^2 |\boldsymbol{x}|^{-1} * \delta\rho(\boldsymbol{x}) \tag{13}$$

$$\Rightarrow \varphi(\boldsymbol{k}) = -\frac{4\pi G a^2}{\boldsymbol{k}^2} \delta\rho(\boldsymbol{k})\,, \tag{14}$$

where the convolution transforms to multiplication in Fourier space. The strategy of the PM method is to first interpolate the particle masses onto a grid, obtaining $\delta\rho(\boldsymbol{x})$, then Fourier transforming this grid to obtain $\delta\rho(\boldsymbol{k})$, converting to potential values $\varphi(\boldsymbol{k})$ through (14), then Fourier transforming back to real space, obtaining $\varphi(\boldsymbol{x})$. The same grid in memory is used to store all of these different quantities.

Mesh interpolation As for the PP method, we wish to construct a density field $\rho(\boldsymbol{x})$ given the particle distribution by assigning a shape S to the particles. Unlike direct summation, computing forces via the potential does not allow us to explicitly remove particle self-interactions, corresponding to the skipped $j = i$ terms of (4) and (11). Instead, the particle shapes must be chosen such that self-interactions do not occur. For a cubic grid, this limits the possible shapes to the hierarchy (Hockney & Eastwood 1988)

$$S_{p_{\mathrm{i}}}(\boldsymbol{x}) = L_\varphi^{-3p_{\mathrm{i}}} \underbrace{\ast\!\ast\!\ast}_{p_{\mathrm{i}}\text{ times}} \Pi\left(\frac{\boldsymbol{x}}{L_\varphi}\right), \tag{15}$$

with the interpolation order $p_{\mathrm{i}} \in \mathbb{N}_0$ and the big $\ast$ operator representing repeated convolution. With the empty convolution understood to be the Dirac delta function, we obtain $S_0(\boldsymbol{x}) = \delta^3(\boldsymbol{x})$ as the lowest-order shape in the hierarchy. Higher-order shapes are then constructed through convolution with the cubic top-hat $\Pi(\boldsymbol{x}/L_\varphi)$ spanning exactly one

[3] The value of x_{s} used for the short-range potential in Figure 1 is one fitting for the P^{3}M method (see section 2.1.3), not for Ewald summation.

[4] Whenever the size of a (cubic) grid is given, it refers to the number of elements along each dimension. In case of the Ewald grid, this then consists of $64 \times 64 \times 64$ elements (each containing a force vector).

grid cell, with the top-hat function given by

$$\Pi(\boldsymbol{x}) = \prod_{d=1}^{3} \Pi\big(\boldsymbol{x}^{[d]}\big)\,, \qquad \Pi(x) = \begin{cases} 1 & |x| < \frac{1}{2} \\ 0 & \frac{1}{2} \leq |x|\,, \end{cases} \tag{16}$$

where Π of vector input is defined by multiplying results obtained from individual scalar inputs, $\boldsymbol{x}^{[d]}$ representing the d'th Cartesian scalar component of vector $\boldsymbol{x}$.

Given some interpolation order $p_\mathrm{i} \geq 1$ we let the continuous density contrast field $\delta\rho(\boldsymbol{x})$ be defined through (2) with $S \to S_{p_\mathrm{i}-1}$, with $S_{p_\mathrm{i}-1}$ in turn given by (15). We then define the discretised grid version of the density contrast $\delta\rho_{\boldsymbol{m}}$ — with $\boldsymbol{m} \in \mathbb{Z}^3$ labelling the mesh points at[5] $\boldsymbol{x}_{\boldsymbol{m}} = L_\varphi(\boldsymbol{m} + ½)$ — via interpolation of the continuous $\delta\rho(\boldsymbol{x})$ as follows:

$$\delta\rho^{(1)}_{\boldsymbol{m}} \equiv L_\varphi^{-3}\,\Pi\left(\frac{\boldsymbol{x}}{L_\varphi}\right) * \delta\rho(\boldsymbol{x})\bigg|_{\boldsymbol{x}=\boldsymbol{x}_{\boldsymbol{m}}} \tag{17}$$

$$= \frac{m}{a^3} \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \left\{ -\frac{N}{L_\mathrm{box}^3} + L_\varphi^{-3} \sum_{j=1}^{N} W_{p_\mathrm{i}}\left(\frac{\boldsymbol{x}_{\boldsymbol{m}jn}}{L_\varphi}\right) \right\}, \tag{18}$$

$$= \frac{m}{a^3} \sum_{\boldsymbol{n}\in\mathbb{Z}^3} \left\{ \begin{aligned} &-\frac{N}{L_\mathrm{box}^3} \\ &+ L_\varphi^{-3} W_{p_\mathrm{i}}\left(\frac{\boldsymbol{x}}{L_\varphi}\right) * \sum_{j=1}^{N} \delta^3(\boldsymbol{x} - \boldsymbol{x}_{jn}) \end{aligned} \right\}\Bigg|_{\boldsymbol{x}=\boldsymbol{x}_{\boldsymbol{m}}}, \tag{19}$$

where we have introduced the dimensionless weight functions $W_{p_\mathrm{i}}(\boldsymbol{x}/L_\varphi) \equiv L_\varphi^3 S_{p_\mathrm{i}}(\boldsymbol{x})$ and used $\boldsymbol{x}_{\boldsymbol{m}jn} \equiv \boldsymbol{x}_{\boldsymbol{m}} - (\boldsymbol{x}_j + L_\mathrm{box}\boldsymbol{n})$. Equality (18) is the one used for code implementation. The parenthesised superscript counts the number of particle $\leftrightarrow$ mesh interpolations carried out, which we shall want to keep track of.

Deconvolved potential With the PM grid holding $\delta\rho^{(1)}_{\boldsymbol{m}}$ values, an in-place FFT converts the values to $\delta\rho^{(1)}_{\boldsymbol{h}}$, the grid version of $\delta\rho(\boldsymbol{k})$ with $\boldsymbol{h} \in \mathbb{Z}^3$ labelling the grid points at $\boldsymbol{k}_{\boldsymbol{h}} = 2\pi L_\mathrm{box}^{-1}\boldsymbol{h}$. This FFT treats the finite numerical representation of $\delta\rho^{(1)}_{\boldsymbol{m}}$ as periodic, implementing the sum over images $\boldsymbol{n}$ of (18) and (19) automatically.

The density values are then converted to potential values using (14), resulting in grid values

$$\varphi^{(1)}_{\boldsymbol{h}} = -\frac{4\pi G a^2}{\boldsymbol{k}_{\boldsymbol{h}}^2}\delta\rho^{(1)}_{\boldsymbol{h}}\,, \qquad \varphi^{(1)}_{\boldsymbol{0}} = 0\,, \tag{20}$$

where the $\boldsymbol{k} = \boldsymbol{0}$ 'DC' mode is explicitly zeroed, corresponding to removing the background density. This enables us to work with density values ρ rather than density contrast values $\delta\rho$ in the implementation, meaning we can ignore the subtraction of N/L_box^3 in (18) and (19).

From (19) it is then clear that we can correct for the interpolation by dividing out the Fourier transformed weight function, allowing us to obtain deconvolved versions of the grid:

$$\varphi^{(c)}_{\boldsymbol{h}} = \left[\frac{W_{p_\mathrm{i}}(L_\varphi \boldsymbol{k}_{\boldsymbol{h}})}{L_\varphi^3}\right]^{c-1} \varphi^{(1)}_{\boldsymbol{h}}\,. \tag{21}$$

[5] Here vector-scalar addition is defined as adding the scalar to each element of the vector. Unlike e.g. GADGET, CONCEPT 1.0 uses cell-centred grid values (by default), hence the offset by half a grid cell.

The properly deconvolved potential grid is then given by $\varphi^{(0)}_{\boldsymbol{h}}$. Applying such deconvolution removes much of the spurious Fourier aliasing, improving the accuracy of the grid representation at small scales (Sefusatti et al. 2016).

Obtaining forces We now transform back to real space using an in-place inverse FFT, obtaining $\varphi^{(c)}_{\boldsymbol{m}}$. We can then construct a force grid as

$$\boldsymbol{f}^{(c)}_{\boldsymbol{m}} = -amL_\varphi^{-1}\boldsymbol{D}_{p_\mathrm{d}}\varphi^{(c)}_{\boldsymbol{m}}\,, \tag{22}$$

where $\boldsymbol{D}_{p_\mathrm{d}}$ is some finite difference operator of order p_d. The resulting force grid $\boldsymbol{f}^{(c)}_{\boldsymbol{m}}$ must then be interpolated back to the particle positions and applied. Ignoring the sum over images $\boldsymbol{n}$ and subtraction of the background N/L_box^3 as previously mentioned, this interpolation is implemented by (18), except that now the sum runs over mesh points instead of particle indices, as this time the interpolation is from the mesh onto the particles:

$$\boldsymbol{f}_i = \sum_{\boldsymbol{m}\in\mathbb{Z}^3} W_{p_\mathrm{i}}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_{\boldsymbol{m}}}{L_\varphi}\right)\boldsymbol{f}^{(-1)}_{\boldsymbol{m}} \tag{23}$$

$$= \frac{4\pi G m^2}{L_\varphi^4} \overbrace{\sum_{\left\{\boldsymbol{m}\,\middle|\,\max_d\left|\boldsymbol{x}_i^{[d]} - \boldsymbol{x}_{\boldsymbol{m}}^{[d]}\right| < \frac{p_\mathrm{i} L_\varphi}{2}\right\}} W_{p_\mathrm{i}}\left(\frac{\boldsymbol{x}_i - \boldsymbol{x}_{\boldsymbol{m}}}{L_\varphi}\right)}^{\text{particle} \leftarrow \text{mesh}} \boldsymbol{D}_{p_\mathrm{d}}\, \mathcal{F}^{-1}_{\boldsymbol{\emptyset}\,\boldsymbol{m}\leftarrow\boldsymbol{h}} \Bigg\{ \underbrace{\left[\frac{W_{p_\mathrm{i}}(L_\varphi\boldsymbol{k}_{\boldsymbol{h}})}{L_\varphi^3}\right]^{-2}}_{\text{2 deconvolutions}} \overbrace{\frac{1}{\boldsymbol{k}_{\boldsymbol{h}}^2}}^{\text{Poisson kernel}} \mathcal{F}_{\boldsymbol{m}\to\boldsymbol{h}} \underbrace{\sum_{j=1}^{N} W_{p_\mathrm{i}}\left(\frac{\boldsymbol{x}_{\boldsymbol{m}} - \boldsymbol{x}_j}{L_\varphi}\right)}_{\text{particles} \to \text{mesh}} \Bigg\}, \tag{24}$$

where we specifically use $\boldsymbol{f}^{(-1)}_{\boldsymbol{m}}$ to take into account the additional particle $\leftarrow$ mesh interpolation, resulting in a total of 2 deconvolutions. The annotated equality (24) provides a complete overview of the PM method by gathering up the different steps, with $\mathcal{F}$ representing the forward FFT and $\mathcal{F}^{-1}_{\boldsymbol{\emptyset}}$ the inverse FFT — normalised so that $\mathcal{F}\mathcal{F}^{-1}\varphi_{\boldsymbol{m}} = \varphi_{\boldsymbol{m}}$ — and the subscript indicating nullification of the $\boldsymbol{k} = \boldsymbol{0}$ mode prior to performing the inverse transform. The $\boldsymbol{m} \leftrightarrow \boldsymbol{h}$ below the FFT operators are just to indicate the change to the grid index caused by the transforms. Read this large expression backwards for it to follow the flow of the algorithm. For localised weight functions W_{p_i} the infinite sum over $\boldsymbol{m}$ in (23) only need to be over mesh points in the vicinity of $\boldsymbol{x}_i$, as indicated for the sum over $\boldsymbol{m}$ in (24). We shall look at W_{p_i} in detail shortly, including how this particular definition of "the vicinity" arise.

In practice, the values stored in the PM grid goes through the transformations $\rho^{(1)}_{\boldsymbol{m}} \to \rho^{(1)}_{\boldsymbol{h}} \to \varphi^{(-1)}_{\boldsymbol{h}} \to \varphi^{(-1)}_{\boldsymbol{m}}$. A separate scalar grid is used to store the forces obtained from $\varphi^{(-1)}_{\boldsymbol{m}}$, along each dimension d in turn. This scalar force grid is then interpolated onto all particles using (23); $\{\boldsymbol{f}_i^{[d]}\} \leftarrow \boldsymbol{f}^{(-1)[d]}_{\boldsymbol{m}} = -amL_\varphi^{-1}\boldsymbol{D}^{[d]}_{p_\mathrm{d}}\varphi^{(-1)}_{\boldsymbol{m}}$.

Order of interpolation and differentiation Though the entire PM method is summarised by (24), we have yet to explicitly write out the weight functions $W_{p_\mathrm{i}}(\boldsymbol{x})$ and their Fourier transforms $W_{p_\mathrm{i}}(\boldsymbol{k}_{\boldsymbol{h}})$ for different orders p_i. Similarly we have not yet specified the difference operators $\boldsymbol{D}_{p_\mathrm{d}}$ for different orders p_d. We shall do so now.

From the definition $W_{p_\mathrm{i}}(\boldsymbol{x}/L_\varphi) \equiv L_\varphi^3 S_{p_\mathrm{i}}(\boldsymbol{x})$ along with

(15), the first weight functions are given by

$$W_{\mathrm{NGP}}(x) = \begin{cases} 1 & |x| < \frac{1}{2} \\ 0 & \frac{1}{2} \leq |x|\,, \end{cases} \tag{25}$$

$$W_{\mathrm{CIC}}(x) = \begin{cases} 1 - |x| & |x| < 1 \\ 0 & 1 \leq |x|\,, \end{cases} \tag{26}$$

$$W_{\mathrm{TSC}}(x) = \begin{cases} \frac{3}{4} - x^2 & |x| < \frac{1}{2} \\ \frac{1}{8}(2|x| - 3)^2 & \frac{1}{2} \leq |x| < \frac{3}{2} \\ 0 & \frac{3}{2} \leq |x|\,, \end{cases} \tag{27}$$

$$W_{\mathrm{PCS}}(x) = \begin{cases} \frac{1}{6}(3|x|^3 - 6x^2 + 4) & |x| < 1 \\ \frac{1}{6}(2 - |x|)^3 & 1 \leq |x| < 2 \\ 0 & 2 \leq |x|\,, \end{cases} \tag{28}$$

where common names — 'nearest grid point' (NGP) for $p_{\mathrm{i}} = 1$, 'cloud in cell' (CIC) for $p_{\mathrm{i}} = 2$, 'triangular shaped cloud' (TSC) for $p_{\mathrm{i}} = 3$, 'piecewise cubic spline' (PCS) for $p_{\mathrm{i}} = 4$ — have been used as labels. The behaviour regarding vector input is inherited from the top-hat (16), i.e. $W_{p_{\mathrm{i}}}(\boldsymbol{x}) = W_{p_{\mathrm{i}}}(\boldsymbol{x}^{[1]})W_{p_{\mathrm{i}}}(\boldsymbol{x}^{[2]})W_{p_{\mathrm{i}}}(\boldsymbol{x}^{[3]})$. All four weight functions are available in CONCEPT 1.0. From (25)–(28) it is clear that grid points further away than $p_{\mathrm{i}}/2$ grid units — along any dimension — from a particle's position do not take part in its interpolation; hence the set of grid points $\boldsymbol{m}$ included in (24). In Figure 1 the PM grid is drawn as thin grey lines, and the mass of the particle in the lower middle has been assigned to nearby grid points using PCS interpolation. The mass fractions are shown as assigned to the centres of the cells, reflecting the choice of cell-centred grid values in CONCEPT 1.0.

As the hierarchy of real-space weight functions are generated through repeated convolution (15), their Fourier transforms are generated through exponentiation (i.e. repeated multiplication). Given that $W_1(\boldsymbol{x})$ is just the top-hat (16), we obtain

$$\frac{W_{p_{\mathrm{i}}}(L_\varphi \boldsymbol{k})}{L_\varphi^3} = \prod_{d=1}^{3} \mathrm{sinc}^{p_{\mathrm{i}}}\left(\frac{L_\varphi \boldsymbol{k}^{[d]}}{2}\right), \tag{29}$$

with the cardinal sine function $\mathrm{sinc}(x) \equiv \sin(x)/x$ and we once more retain the same behaviour regarding vector input.

Now let us turn to the finite difference operator $\boldsymbol{D}_{p_{\mathrm{d}}}$. This vector operator can be separated into three copies of the same scalar operator $\boldsymbol{D}_{p_{\mathrm{d}}} = (D^{[1]}_{p_{\mathrm{d}}}, D^{[2]}_{p_{\mathrm{d}}}, D^{[3]}_{p_{\mathrm{d}}})$, each acting along a separate dimension. The most natural choice is to use the optimally accurate symmetric difference approximation given the order p_{d}. If by p_{d} we mean the number of grid points used for this approximation — imposing $p_{\mathrm{d}} \in 2\mathbb{N}$ due to the operation being symmetric — these operators can be constructed as (see e.g. Fornberg (1988))

$$D_{p_{\mathrm{d}}}\varphi_m = \sum_{\Delta m=-p_{\mathrm{d}}/2}^{p_{\mathrm{d}}/2} \partial_\xi \prod_{\substack{\Delta m'=-p_{\mathrm{d}}/2 \\ \Delta m' \neq \Delta m}}^{p_{\mathrm{d}}/2} \frac{\xi - \Delta m'}{\Delta m - \Delta m'}\bigg|_{\xi=0} \varphi_{m+\Delta m}\,, \tag{30}$$

where the vector element superscript has been omitted and φ_m is to be understood as a one-dimensional grid (or slice of the 3D grid $\varphi_{\boldsymbol{m}}$) with points labelled by $m \in \mathbb{Z}$ at $L_\varphi(m+½)$. CONCEPT 1.0 implements $p_{\mathrm{d}} \in \{2, 4, 6, 8\}$, which from (30) is

$$D_2\varphi_m = \frac{1}{2}\begin{pmatrix} -\varphi_{m-1} \\ +\varphi_{m+1} \end{pmatrix}, \tag{31}$$

$$D_4\varphi_m = \frac{1}{12}\begin{pmatrix} +\varphi_{m-2} - 8\varphi_{m-1} \\ -\varphi_{m+2} + 8\varphi_{m+1} \end{pmatrix}, \tag{32}$$

$$D_6\varphi_m = \frac{1}{60}\begin{pmatrix} -\varphi_{m-3} + 9\varphi_{m-2} - 45\varphi_{m-1} \\ +\varphi_{m+3} - 9\varphi_{m+2} + 45\varphi_{m+1} \end{pmatrix}, \tag{33}$$

$$D_8\varphi_m = \frac{1}{840}\begin{pmatrix} +\ \ 3\varphi_{m-4} - \ 32\varphi_{m-3} \\ -\ \ 3\varphi_{m+4} + \ 32\varphi_{m+3} \\ +\ 168\varphi_{m-2} - 672\varphi_{m-1} \\ -\ 168\varphi_{m+2} + 672\varphi_{m+1} \end{pmatrix}, \tag{34}$$

with the symmetric property clearly manifest.

The interpolation order p_{i} and difference order p_{d} may be chosen independently, leading to many possible PM schemes available in CONCEPT 1.0. By default, CONCEPT 1.0 uses $p_{\mathrm{i}} = 2$ (CIC) interpolation and $p_{\mathrm{d}} = 2$ differentiation for computing gravity via the PM method.

Parallelisation We have yet to discuss the details of the MPI parallelisation of CONCEPT 1.0, which necessarily must be integrated into the gravitational schemes. Given n_{p} MPI processes, CONCEPT divides the box into n_{p} equally shaped cuboidal domains and assigns one such domain to each process. The exact domain decomposition chosen is uniquely[6] the one with the least elongated domains, minimizing the surface to volume ratio, in turn minimising communication efforts between processes. The domain decomposition shown in Figure 1 — with a thick black outline around each domain — is for a simulation with $n_{\mathrm{p}} = 6$ processes, resulting in the decomposition $3 \times 2 \times 1$.

For the PP method, particles in one domain must explicitly be paired up with particles in all other domains. After having carried out the interactions of particles within their local domain, each process sends a copy of its particle data to another process — the 'receiver process' — while simultaneously receiving particle data from a third process — the 'supplier process'. The interactions between local and received non-local particles are then carried out, with the momentum updates to the non-local particles sent back to the supplier process, while at the same time receiving and applying corresponding local momentum updates from the receiver process. This carries on for all such 'dual' process/domain pairings, of which there are $\lfloor n_{\mathrm{p}}/2 \rfloor$ from the point of view of any given local process, not counting the pairing between the local process and itself.

For the PM method the parallelisation efforts are more involved. The PM grid is distributed in real space according to the domains. Each grid cell must be entirely contained within a single domain, imposing the restriction that the number of domain subdivisions of the box along each dimension must divide n_φ. For the PM grid in Figure 1, $n_\varphi = 54$ is chosen, which indeed is divisible by 3, 2 and 1.

To carry out the required FFTs on the distributed grid,

[6] Up to permutation of the dimensions.

CONCEPT employs the FFTW library (Frigo & Johnson 2005), specifically its MPI-parallelised, real, 3D, in-place transformations. FFTW imposes a 'slab'[7] decomposition of the global grid, in conflict with the cuboidal domain decomposition. Before performing a forward FFT, CONCEPT then constructs a slab-decomposed copy of the domain-decomposed PM grid. Similarly, once the slab-decomposed grid is transformed back to real space, its values are copied over to the domain-decomposed grid. Furthermore, while in Fourier space, grids are transposed along the first two dimensions, as the last step in the distributed FFT routines is a global transposition, which is skipped. Similarly skipping this transposition step when transforming back to real space brings the dimensions back in order.

When it comes to particle interpolation using $W_{p_\mathrm{i}}(\boldsymbol{x})$ (25)–(28) and grid differentiation using $\boldsymbol{D}_{p_\mathrm{d}}$ (31)–(34), data from a few (depending on the orders p_i and p_d) grid cells away are required. Near a domain boundary, some of this required data belongs to a neighbouring domain and thus reside on a non-local process. To solve this, local domain grids are equipped with additional 'ghost layers' of grid points surrounding the primary, local part of the grid. These ghost points must then be kept up-to-date with the corresponding non-local data, and vice versa. The required thickness n_ghost of the ghost layers — i.e. the number of ghost points sticking out perpendicular to a domain surface — depends upon the orders p_i and p_d. As already mentioned, (25)–(28) demonstrate that interpolation through $W_{p_\mathrm{i}}(\boldsymbol{x})$ touches at most $p_\mathrm{i}/2$ grid points to either side of a particle (along each dimension), thus requiring $n_\mathrm{ghost} \geq \lfloor p_\mathrm{i}/2 \rfloor$. For $\boldsymbol{D}_{p_\mathrm{d}}$, the number of required ghost points can readily be read off of (31)–(34) as[8] $n_\mathrm{ghost} \geq \lceil p_\mathrm{d}/2 \rceil$. In total then,

$$n_\mathrm{ghost} = \max(\lfloor p_\mathrm{i}/2 \rfloor, \lceil p_\mathrm{d}/2 \rceil) \tag{35}$$

ghost points are needed around local real-space domain grids.

Figure 1 shows the ghost layers around the lower middle domain as "ghostly shaded" PM cells, using $n_\mathrm{ghost} = 2$. As seen, the periodicity of the box is handled very naturally, which is really a secondary job almost automatically fulfilled by the ghost layers. Even in cases where the box is not subdivided along a given dimension, ghost layers are then still needed to implement the periodicity of the PM grid.

2.1.3 P³M gravity

While the PM method is unrivalled in its performance, it comes with a severe limitation in resolution due to the finite grid cell size L_φ. One approach to overcome this is to only use PM for gravity at scales sufficiently large compared to L_φ, and then supply the missing short-range gravity using direct summation (PP) techniques. This hybrid PP-PM (P³M) method is the default gravitational solver of CONCEPT 1.0. It comes with a free parameter x_r which trades the accuracy of the PP method for the efficiency of the PM method, with practical values yielding a good balance.

[7] Meaning distributed only along a single dimension.

[8] Though rounding up $p_\mathrm{d}/2$ is redundant for the symmetric difference operations of even order p_d, it becomes important for non-symmetric odd orders. CONCEPT does in fact additionally implement $\boldsymbol{D}_1$, in both a 'forward' and a 'backward' version.

Combining PP and PM For the long-range part, the P³M method goes trough all of the same steps as the PM method of section 2.1.2, with the Poisson kernel $4\pi/\boldsymbol{k}^2$ (14) replaced with the long-range kernel $\mathcal{G}_\mathrm{lr}(\boldsymbol{k}) = 4\pi\exp(-x_\mathrm{s}^2\boldsymbol{k}^2)/\boldsymbol{k}^2$ (10) introduced earlier for the Ewald summation. Next, the missing short-range forces — corresponding to the potential $\mathcal{G}_\mathrm{sr}(\boldsymbol{x}) = |\boldsymbol{x}|^{-1}\,\mathrm{erfc}(|\boldsymbol{x}|/[2x_\mathrm{s}])$ or the real-space sum over $\boldsymbol{n}$ of the force (11) — are added in using direct summation. Below, both the long-range and short-range sub-methods of the P³M method are spelled out:

$$\begin{aligned}\boldsymbol{f}_i = {}& \underbrace{\frac{4\pi G m^2}{L_\varphi^4}\overbrace{\sum_{\substack{\{\,\boldsymbol{m}\;|\;\max_d\left|\boldsymbol{x}_i^{[d]}-\boldsymbol{x}_{\boldsymbol{m}}^{[d]}\right|<\frac{p_\mathrm{i}L_\varphi}{2}\,\}}} W_{p_\mathrm{i}}\left(\frac{\boldsymbol{x}_i-\boldsymbol{x}_{\boldsymbol{m}}}{L_\varphi}\right)}^{\text{particle}\,\leftarrow\,\text{mesh}}\boldsymbol{D}_{p_\mathrm{d}}\,\mathcal{F}^{-1}_{\substack{\boldsymbol{\emptyset}\\ \boldsymbol{m}\leftarrow\boldsymbol{h}}}\Bigg\{}_{\text{long-range}} \\ & \underbrace{\left[\frac{W_{p_\mathrm{i}}(L_\varphi\boldsymbol{k}_{\boldsymbol{h}})}{L_\varphi^3}\right]^{-2}}_{\text{2 deconvolutions}}\overbrace{\frac{\exp(-x_\mathrm{s}^2\boldsymbol{k}_{\boldsymbol{h}}^2)}{\boldsymbol{k}_{\boldsymbol{h}}^2}}^{\text{long-range kernel}}\mathcal{F}_{\boldsymbol{m}\rightarrow\boldsymbol{h}}\underbrace{\sum_{j=1}^{N}W_{p_\mathrm{i}}\left(\frac{\boldsymbol{x}_{\boldsymbol{m}}-\boldsymbol{x}_j}{L_\varphi}\right)}_{\text{particles}\,\rightarrow\,\text{mesh}}\Bigg\} \\ & - Gm^2\sum_{\substack{\{\,j\;|\;|\boldsymbol{x}_{ijn'}|<x_\mathrm{r}\,\}\\ j\neq i}}\left[\begin{array}{l}|\boldsymbol{x}_{ijn'}|^{-3}\,\mathrm{erfc}\left(\dfrac{|\boldsymbol{x}_{ijn'}|}{2x_\mathrm{s}}\right)\\ +\dfrac{|\boldsymbol{x}_{ijn'}|^{-2}}{\sqrt{\pi}x_\mathrm{s}}\exp\left(-\dfrac{\boldsymbol{x}_{ijn'}^2}{4x_\mathrm{s}^2}\right)\\ +\underbrace{\left(|\boldsymbol{x}_{ijn'}|^{-3}-|\boldsymbol{x}_{ijn'}|^{-3}\right)}_{\text{softening}}\end{array}\right]\boldsymbol{x}_{ijn'}\Bigg\}_{\text{short-range}}\end{aligned} \tag{36}$$

The exponential decay of the short-range force of (36) allows us to only consider particle pairs within a distance x_r a few times larger than x_s. In particular, choosing x_s small compared to the box ensures that only the single image $\boldsymbol{n}'$ of particle j nearest to particle i has a non-negligible influence, ridding us of the sum over images $\boldsymbol{n}$. In (36) then, $\boldsymbol{x}_{ijn'} \equiv \boldsymbol{x}_i - (\boldsymbol{x}_j + L_\mathrm{box}\boldsymbol{n}')$ with $\boldsymbol{n}'$ chosen such that $|\boldsymbol{x}_{ijn'}| = \min_{\boldsymbol{n}\in\mathbb{Z}^3}|\boldsymbol{x}_{ijn}|$.

We seek to minimize x_s in order to delegate as large of a fraction of the total work load as possible to the efficient PM part. Make x_s too small however and the discrete nature of the grid will start to show up as spurious defects in the long-range force. The default values employed by CONCEPT for P³M is the same as those used by GADGET-2:

$$\begin{cases} x_\mathrm{s} = 1.25\,L_\varphi\,, \\ x_\mathrm{r} = 4.5\,x_\mathrm{s}\,, \end{cases} \tag{37}$$

which is also what is depicted in Figure 1. Here $2x_\mathrm{s}$ is shown for the upper left particle as dictating the width of the short-range potential, and a circle of radius x_r is shown around every particle, illustrating their gravitational region of influence.

Using (37), the performance of the P³M method in CONCEPT 1.0 then depends on the grid size n_φ through $L_\varphi = L_\mathrm{box}/n_\varphi$. We prefer to run with

$$n_\varphi = 2\sqrt[3]{N}\,, \tag{38}$$

i.e. having 8 times as many PM cells as particles. While requiring quite a bit more memory than say $n_\varphi = 1\sqrt[3]{N}$, this large cells to particles ratio lowers x_s, shifting a larger fraction of the computational burden onto the efficient long-range force, speeding up simulations significantly. Even so, for typical simulations the majority of the computation time is

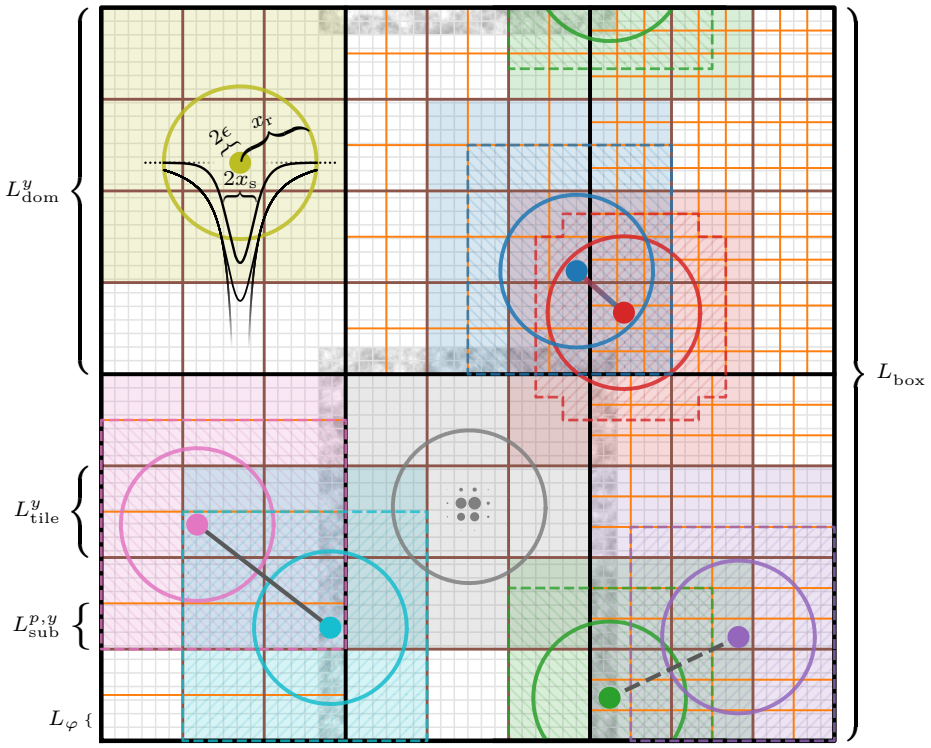

Figure 1. The full geometric picture of the simulation box, demonstrating various aspects. In the example shown we imagine running a simulation with $n_p = 6$ processes, resulting in a domain decomposition of $3 \times 2 \times 1$. For clarity we shall ignore the last dimension. The entire cubic box (outer black square) of length L_{box} is then subdivided into 3×2 domains (black rectangles) all of size $L^x_{\text{dom}} \times L^y_{\text{dom}}$. The global, cubic PM grid is shown in grey, with a grid size of $n_\varphi = 54$. A low number $N = 8$ of particles is shown as small, different coloured solid circles with radii given by the softening length ϵ chosen as $0.03\, L_{\text{box}}/\sqrt[3]{N}$. We note that the actual number of particles in a standard simulation of $n_\varphi = 54$ would be much greater. The mass of the grey particle in the lower central domain is shown as being assigned to the PM grid cells using interpolation, specifically PCS. Also, ghost layers of the local PM grid using $n_{\text{ghost}} = 2$ are shown around this domain as "ghostly" shaded cells. The raw Newtonian, softened Newtonian and softened short-range single-particle potentials are shown for the yellow particle at the upper left, with x_s dictating the width of the localised short-range potential. As seen, the short-range potential is vanishingly small a distance x_r away from the particle. The values (37) are used for x_s and x_r. For two particles to interact under short-range gravity, they must be within x_r of each other, i.e. each coloured hollow circle must contain the other particle. Thus here, only the blue and red particle pair interact. The domains are subdivided into tiles of size $L^x_{\text{tile}} \times L^y_{\text{tile}}$, shown in brown. Each tile has to be at least x_r along each dimension, here leading to a tile decomposition of 3×4 of each domain. The 3×3 tiles within reach of a given particle has been shaded with the colour of that particle, indicating neighbouring tiles needed to be checked for possible interacting partners of the given particle. Given this information alone, one can see that the upper right blue and red, the lower left magenta and cyan, as well as the lower right green and purple particles all have a change of pairwise interacting. Each tile is further subdivided into subtiles — shown in orange — independently within each domain. Their size are given by $L^{p,x}_{\text{sub}} \times L^{p,y}_{\text{sub}}$ with p labelling the domain/process. Coloured hatched regions around particles show which subtiles are within reach x_r of the subtile containing each particle. From this information, it is now clear that the green and purple particle are too far separated to interact. The subtile decomposition employed within the lower left domain is insufficient to tell us that the magenta and cyan particle do not interact.

spent on the short-range forces, and so it is vital to implement these efficiently, to which we shall attend shortly.

As for the PM method of section 2.1.2, the P^{3}M method in CONCEPT 1.0 employs $p_i = 2$ (CIC) interpolation by default. As the long-range mesh of P^{3}M is generally much smoother than the mesh of PM, it makes sense to increase the order of differentiation, and so $p_d = 4$ is chosen as the default for P^{3}M gravity in CONCEPT 1.0. The default P^{3}M settings of CONCEPT 1.0 thus coincide with the (hard-coded) TreePM settings of GADGET-2.

Tiles What remains to be discussed is exactly how to efficiently implement the short-range[9] sum of (36), where each particle should be paired only with neighbouring particles within a distance x_r. What we need is to sort the particles in 3D space using some data structure, which then allows for efficient querying of nearby particles given some location.

The data structure employed for the particle sorting in CONCEPT 1.0 is one we refer to as a *tiling*. Here each domain is subdivided into as many equally sized cuboidal volumes — called *tiles* — as possible, with the constraint that the tiles must have a size of at least x_r along each dimension. This guarantees that a particle within a given tile only interacts

[9] The perhaps equally complicated-looking *long*-range sum over $\boldsymbol{m}$ of (36) is in fact trivial to implement for our regular grid.

with other particles in the surrounding $3 \times 3 \times 3$ block of tiles, i.e. with particles within its own tile or within the 26 neighbouring tiles. The tiling is shown in brown on Figure 1, where the shape and size of the domains give rise to a tile decomposition $3 \times 4 \times 9$ (with the last dimension suppressed on the figure) of each domain. Note that since the domains are not cubic, the tiles will generally not be so either, as is the case on the figure. As all domains are equally shaped, all domain tilings will be similar, giving rise to a global (box) tiling. Further note that this global tiling generally do not align with the global PM mesh.

With the geometry of the tiling fixed, the particles are sorted into tiles in $\mathcal{O}(N)$ time. The interactions between particles within tiles are now carried out in a manner somewhat similar to the parallelisation strategy of the PP method described towards the end of section 2.1.2, though now both at the domain and at the tile level, below described separately for the two cases of tile interaction purely within the local domain and tile interaction across a domain boundary:

Local tile interaction: Every process iterates over its tiles, in turn considering them as the 'receiver tile'. After dealing with the interactions of particles within a given receiver tile itself, a neighbouring tile is selected as the 'supplier tile'. Interactions between particles of the receiver and supplier tile are then carried out. A different neighbouring supplier tile within the local domain is then continually selected, until exhaustion. Once all neighbouring, local tiles (up to 26) have been dealt with, a different tile is considered as the receiver, and so on. Importantly, when selecting the next supplier tile, the one chosen must not have already been paired with the current receiver tile using opposite receiver/supplier roles.

Non-local tile interaction: The local process/domain is 'dual-paired' with a non-local receiver and supplier process/domain, as in the PP method. Unlike the PP method, only the 26 neighbouring domains are considered, resulting in 13 pairings. The particles within local tiles neighbouring the receiver domain are sent to the receiver process, while corresponding particles are received from the supplier process. The local tiles neighbouring the supplier domain is then iterated over, in turn given the role as the receiver tile. Each such local receiver tile is then sequentially paired with non-local supplier tiles from the subset of the tiles (up to 9) received from the supplier process which neighbour the local receiver tile in question. Having directly updated the momenta of local particles due to the interactions, the non-local momentum updates are additionally sent back to the supplier process, while corresponding momentum updates are received from the receiver process, which are then applied as well.

Having at least 3 tiles across the box along each dimension ensures that the above scheme does not double count any tile pairs, even in extreme cases such as $n_\mathrm{p} = 1$ where all 26 "non-local neighbour domains" are really all just the local domain itself. This constraint[10] is thus imposed by CONCEPT 1.0.

[10] In fact CONCEPT 1.0 requires the global tiling to consist of at least 4 tiles across each dimension, as this simplifies some logic regarding the periodicity. For the standard values (37), this restricts $n_\varphi \geq 23$ — really $n_\varphi \geq 24$ as CONCEPT further needs grid sizes to be even — which is not much of a restriction at all.

With n_tile the total number of tiles in the box, the average number of particles in a tile is N/n_tile, resulting in a time complexity for the tiled short-range force computation of $\mathcal{O}(N^2/n_\mathrm{tile})$. As $n_\mathrm{tile} \lesssim n_\varphi^3$ this again shows how using a finer PM grid shifts the computational burden from the short-range computation over to the long-range computation. Furthermore, using $n_\varphi^3 \propto N$, we see that the tiles formally reduce the full short-range interaction to linear time $\mathcal{O}(N)$, beating the rivalling $\mathcal{O}(N \log N)$ tree methods. In practice, having large inhomogeneities will make different tiles require different computational effort, degrading the performance. If the inhomogeneities extend to the domain scale, further degrading arise due to load imbalance between the processes, which CONCEPT currently does not attempt to mend.

Subtiles The basics of the tile-based short-range particle pairing has now been established, but it has room for optimisations. One such optimisation is that of *sub*tiles, i.e. finer tiles within the main tiles.

In Figure 1, the circle of radius x_r shown around every particle demonstrates the reach of the short-range force. In addition, the 3×3 block of tiles surrounding a particle is shaded with a colour matching that particle, showing the possible tiles in which interacting partner particles might reside. Though in fact only the blue and red particle in the upper right are close enough to interact, the magenta and cyan pair in the lower left as well as the green and purple pair in the lower right seem like equally good candidates for possible interaction, from the point of view of the tiling.

Once two particles i and j have finally been paired up by the tiling mechanism, their separation $|\boldsymbol{x}_i - \boldsymbol{x}_j|$ is measured, upon which the interaction is aborted if $|\boldsymbol{x}_i - \boldsymbol{x}_j| > x_\mathrm{r}$. With perfectly small tiles of volume x_r^3, this happens for 9 % of interactions with both particles within the same tile, for 66 % of interactions between tiles sharing a face, for 91 % of interactions between tiles sharing only an edge, and for 98 % of interactions between tiles sharing only a corner. The reason for adding subtiles is to exclude many of these non-interactions early, accelerating the short-range computation. This is done by extending the tile pairing mechanism with another, deeper layer, pairing up subtiles. Crucially, only subtiles close enough so that they could potentially contain interacting particles are paired, leading each receiver subtile to be paired up with subtiles within a surrounding, blocky sphere, approaching a smooth sphere of radius x_r in the limit of infinite subtiles.

Unlike the main tiles, subtiles are local to each domain, meaning that each process is free to choose its own subtile decomposition, though with the same employed throughout the domain. Figure 1 shows a variety of subtile decompositions in orange, e.g. 2×3 for the lower right domain. Here we also find the green and purple particle, which need to be paired according to the tiling, as the green particle is within the purple shaded region, and vice versa. The green and purple hatching shows which subtiles are reachable from the particular subtile containing each particle. As the hatched regions do not contain the subtile of the other particle, it means that adding in this subtiling indeed saves us from having to consider this irrelevant particle pair. Turning to the lower left domain of Figure 1, we see that the applied subtile decomposition of 1×2 is insufficient to rid us of the irrelevant pairing of the magenta and cyan particle, even

though their separation is more than twice the critical distance x_r. Increasing the number of subdivisions by just 1 in either dimension would have made the difference.

Finally let us consider the blue and red particle pair at the upper right of Figure 1, where no amount of subtiling will reject the pairing since these particles are close enough for interaction to take place. For the domain containing the red particle, a subtile decomposition of 3×4 is used, which is substantial enough for the red hatched region to become slightly blocky. As the two particles reside on different processes, the interaction cannot take place before one of the processes sends its particle to the other process, as described earlier. The received particle(s) are then re-sorted according to the local subtiling, which is then traversed in order to locate particle pairs. This means that the subtile decomposition used for the blue $\leftrightarrow$ red interaction depends upon which process ends up as being considered the receiver and which the supplier. This is why Figure 1 shows the blue and red hatched regions extending into the other domain, disregarding the different subtiling used here. Though the details of the inter-process communication may then affect the number of paired particles, which particle pairs end up interacting in the end remain unaffected.

Though subdividing space further could always lead to a still lower number of mistakenly paired particles, the overhead associated with the increased number of subtiles means that there exists a sweet spot. Generally, higher particle number densities calls for finer subtile decompositions. By default, CONCEPT 1.0 automatically estimates the optimal subtiling within each domain. Over time, each process periodically checks whether it is worth subdividing further due to the increased inhomogeneity. It does so by temporarily applying a slightly more refined subtile decomposition and comparing the measured time for a short-range force computation with a record of previous such computation times. If superior, the refined subtiling is kept, otherwise the old one is switched back in. The subtiles are thus both spatially and temporally adaptive.

Other optimisations CONCEPT 1.0 goes to great lengths in order to arrive at a performant short-range computation, as evident from the implementation of subtiles, including automatic refinement. Here we briefly want to discuss further such short-range optimisations employed.

The two-level tile + subtile structure is reminiscent of a shallow tree. While the geometry of a full tree reflects the underlying particle structure, the geometry of our (sub)tilings is determined solely by the simple Cartesian subdivisions. This allows us to pre-compute which of the (sub)tiles to pair with each other, eliminating a lot of decision making from within the actual 'walk' (the iteration over tiles $\rightarrow$ subtiles $\rightarrow$ particles), which in turn saves on clock cycles and lowers the pressure on the branch predictor. Having a static, non-hierarchical data structure further results in simple access patterns with minimal pointer chasing, allowing for proper exploitation of CPU cache prefetching.

Once two particles i and j have been selected for interaction, the first thing to do is to compute their mutual squared[11] distance $|\boldsymbol{x}_{ijn'}|^2$, after which the interaction is rejected if $|\boldsymbol{x}_{ijn'}|^2 \geq x_\mathrm{r}^2$, in accordance with the short-range sum of (36). Here we need to effectively shift $\boldsymbol{x}_i - \boldsymbol{x}_j$ by $L_\mathrm{box}\boldsymbol{n}'$ as to minimise $|\boldsymbol{x}_{ijn'}|^2 = |\boldsymbol{x}_i - \boldsymbol{x}_j - L_\mathrm{box}\boldsymbol{n}'|^2$, corresponding to finding the image of particles j nearest to particle i. The solution $\boldsymbol{n}'$ can in fact be determined just from knowing the tiles of particle i and j, and so we pre-compute this already at the tile pairing level. In the typical case of many tiles across the box, the vast majority of tile pairs will have $\boldsymbol{n}' = \boldsymbol{0}$. To take advantage of this, explicit loop unswitching[12] is utilised to completely eliminate the redundant zero-shift in these cases.

With particles i and j finally selected and deemed close enough for interaction to occur, we now need to compute their mutual short-range force, given by the large bracket of (36). Given that it is needed within the tightest loop of the program, this large expression is quite expensive. We thus have it (including the softening term) tabulated in a 1D table, indexed by $|\boldsymbol{x}_{ijn'}|^2$ between 0 and x_r^2. Here we use the cheapest possible (1D) NGP lookup, with the table being rather large[13] in order to ensure accurate results nonetheless. This strategy works well for modern hardware with large CPU caches.

To further enable good utilisation of the CPU caches, the particles are ordered in memory in accordance with the visiting order resulting from the tile $\rightarrow$ subtile $\rightarrow$ particle walk. The drifting of the particles will gradually degrade this previously optimal sorting, and so the in-memory reordering of the particles is periodically reapplied.

Recap of subvolumes The simulations of CONCEPT 1.0 make use of several different, nested subvolumes, in particular when using $\mathrm{P^3M}$. It may not be clear why we need this many levels of nested subvolumes, or indeed why we do not opt for even more. In fact, each such level exists for a very particular reason, which is briefly outlined below.

Box: Though usually not thought of as a *sub*volume, the simulation box itself exists in order to reduce an infinite universe to a finite volume with a finite number of degrees of freedom. The infinity of space is then imitated by the imposed periodicity.

Domains: The box is subdivided into domains in order to reduce the N-body problem into parallelisable chunks, to be distributed over many CPUs. CONCEPT uses a one-to-one mapping between domains, CPU cores and MPI processes.

Tiles: The domains are subdivided into tiles in order to take advantage of the finite range of the short-range force, divvying up the particles into subvolumes with the guaranteed property that particles within one such subvolume does not interact with particles further away than the nearest neighbour subvolumes. In particular, this lends itself to easy, near-minimal communication between processes at the domain level.

[11] We keep working with *squared* distances in order not to perform an expensive square root operation.
[12] This is achieved through custom transpiler directives and code transformation, briefly described in section A2.
[13] By default this table has 2^{12} elements, exploiting the large CPU caches of modern machines. A far smaller table and e.g. linear interpolation would work as well, but at the cost of performance.

Subtiles: Subtiles exist purely as an optimisation layer, accelerating the short-range computation through effective early rejection of particle pairs, by corresponding elimination of subtile pairs. Unlike all other subvolumes, the number of subtiles are free to change over time, adapting to increased inhomogeneity. In addition, since subtiles are never shared between processes, the number of subtiles is free to vary from domain to domain, introducing spatial adaptivity as well.

One can imagine introducing a still deeper level of subvolumes, i.e. 'subsubtiles', with the hope of further speeding up the computation. For this to not be equivalent to simply increase the number of subtiles, the coarseness of the subsubtilings would have to vary across the domain, e.g. within each tile or subtile. This would in turn imply that the subvolume geometry considered by a given process varies from place to place, which will decrease CPU cache performance. On top, there of course comes a point where sorting particles into still finer subvolumes and indexing into them outweigh the benefits from slightly increased early particle pair rejection. Given a large enough number of processes n_p, the spatial adaptiveness of the subtilings ensures that this in fact is the optimal level at which to stop subdividing space. We conjecture that this is the case also for typical values of n_p.

2.2 Time-stepping

This subsection describes the time-stepping mechanism implemented in CONCEPT 1.0, including how the global simulation time step is chosen throughout cosmic history, and how this global time step is subdivided into finer steps, generating adaptive particle time-stepping.

As eluded to in section 2.1, CONCEPT employs cosmic time t as its choice of time integration variable, and makes use of comoving coordinates $\boldsymbol{x} \equiv \boldsymbol{r}/a$ — with $\boldsymbol{r}$ being physical coordinates — and associated canonical momenta $\boldsymbol{q} \equiv a^2 m\dot{\boldsymbol{x}}$ with $\dot{} \equiv \partial_t$. The Hamiltonian equations of motion for the particles are then (Peebles 1980)

$$\begin{cases} \dot{\boldsymbol{x}}_i(t) = \dfrac{\boldsymbol{q}_i(t)}{a^2(t)m}\,, \\ \dot{\boldsymbol{q}}_i(t) = \dfrac{\boldsymbol{f}_i(t)}{a(t)}\,, \end{cases} \tag{39}$$

with the comoving force $\boldsymbol{f}_i$ being the primary subject of section 2.1.

Given the state of the N-body system $(\{\boldsymbol{x}_i(t)\}, \{\boldsymbol{q}_i(t)\})$ at some time t, the coupled[14] equations (39) can be solved numerically by alternatingly evolving $\{\boldsymbol{x}_i(t)\} \to \{\boldsymbol{x}_i(t+\Delta t)\}$ (keeping $\{\boldsymbol{q}_i\}$ fixed) and $\{\boldsymbol{q}_i(t)\} \to \{\boldsymbol{q}_i(t+\Delta t)\}$ (keeping $\{\boldsymbol{x}_i\}$ fixed) over discrete time steps of size Δt.

2.2.1 *Global time step size*

Typical cosmological N-body simulations start from initial conditions at early, linear times (say $t \approx 10\,\mathrm{Myr}$) and evolve the system forward to the present, non-linear time (say $t \approx 14\,\mathrm{Gyr}$). During this evolution, physical phenomena — related to the particles themselves as well as the background — and numerical aspects introduce various time scales, above which the discrete time-stepping cannot operate if we are to hope for a converged solution. This leads to the concept of a time step limiter; a condition imposing a maximum allowed value for Δt, given by a small fraction of a corresponding time scale. Below we list the main such limiters (time scales) implemented in CONCEPT 1.0, shown together in Figure 2.

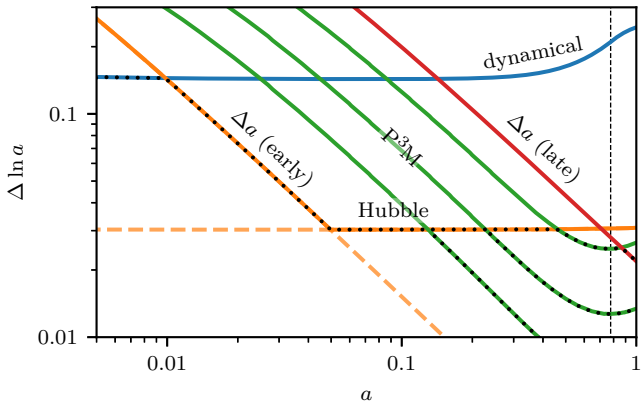

Figure 2. Primary time step limiters in CONCEPT 1.0 and resulting time step size, expressed as $\Delta \ln a = \Delta a(t)/a(t) = \ln a(t+\Delta t) - \ln a(t)$, as function of scale factor a. Different limiters dominate at different times, as indicated by the dotted line, showing the evolution of the time step size itself. Note that 'Δa (early)' and 'Hubble' are part of the same limiter, with the limiter value chosen as the maximum of the two sub-limiter values. Most limiters depend solely on the background cosmology, the exception being the $\mathrm{P^3M}$ limiter which depends on the particle dynamics and thus the simulation resolution. The $\mathrm{P^3M}$ limiter is shown for the cases $L_\mathrm{box} \in \{2, 1, ½\}\sqrt[3]{N}\,\mathrm{Mpc}/h$, with smaller box sizes (higher resolution) leading to lower allowed time step sizes. The dotted line is shown going through all three cases, though of course only a single $\mathrm{P^3M}$ limiter exists for a given simulation. The qualitative change in behaviour of some of the limiters at late times is caused by the transition to Λ domination, wit matter-Λ equality indicated by the vertical dashed line. A standard ΛCDM cosmology (see Table 1 in section 3) was used to produce the figure, with all not-too-exotic simulations resulting in similar limiter values. Though $\Delta \ln a$ decreases over time, the maximum allowed time step size Δt is generally increasing since $\Delta t \propto a^{3/2}\Delta \ln a$ (matter domination).

Dynamical: The gravitational dynamical time scale $(G\bar{\rho})^{-1/2}$, with $\bar{\rho}$ the background density of all non-linear components in the simulation.

Fixed Δa (late): The time step Δt corresponding to a fixed Δa.

Fixed Δa (early) and the Hubble time: This limiter is constructed as the maximum of two sub-limiters; the value Δt which corresponds to a fixed Δa, and the instantaneous Hubble time $H^{-1}(t)$.

P^3M: The time it takes to traverse a distance equalling the short-/long-range force split scale x_s given the root mean square velocity of the particle distribution, $x_\mathrm{s}/\sqrt{\langle \dot{\boldsymbol{x}}^2 \rangle}$.

As seen from Figure 2, which limiter dominates is subject to change during typical simulations. Of the above, only the $\mathrm{P^3M}$ limiter is non-linear — meaning it depends on the particle system — with higher particle resolutions leading to a smaller maximal allowed Δt. All other limiters listed are obtained solely from the background.

Studying the linear growth of matter perturbations in a

[14] Remember that $\boldsymbol{f}_i$ depends explicitly on all positions $\{\boldsymbol{x}_{j\neq i}\}$.

matter-dominated universe, we have $D \propto a$, $D = D(a)$ being the growth factor. A fixed relative tolerance on the discrete evolution of D is then ensured if we keep $\Delta D/D \propto \Delta a/a = \Delta \ln a$ constant. As evident from Figure 2 this is equivalent to having $\Delta t \propto H^{-1}$, i.e. the Hubble limiter. This limiter is employed by GADGET-2 all the way from early times until non-linear limiters take over. We have found that this leads to unnecessarily fine time steps early on, probably due to the very simple initial conditions with each particle coasting along a nearly straight path. While GADGET-2 includes the horizontal dashed line of of Figure 2 as part of its Hubble limiter, CONCEPT 1.0 effectively changes the 'fixed' value of $\Delta \ln a$ by instead using the dynamical limiter at early times, employing the constant Δa (early) as a bridge between the two.

CONCEPT 1.0 implements a few extra limiters, which only come into play for non-standard simulations. These include a non-linear PM limiter and a non-linear Courant limiter for fluid components, as well as component-wise background limiters for the relativistic transition time for components with changing equation of state (relevant for e.g. non-linear massive neutrinos, see Dakin et al. (2019a)) and for the life time of decaying components (relevant for decaying matter, see Dakin et al. (2019b)).

For minimal loss of symplecticity during time-stepping (described in section 2.2.2), the time step size Δt should be kept constant over many steps. On the other hand, keeping Δt at a lower value than necessary introduces further steps than required given the target accuracy. In CONCEPT we use a period of 8 steps[15], after which the particle system is synchronised (see section 2.2.2) and Δt allowed up increase in accordance with the limiters.

2.2.2 Adaptive particle time-stepping

With the size of the time step Δt determined, CONCEPT integrates the particle system forward in time using a symplectic second-order accurate leapfrog scheme (Quinn et al. 1997), as is typical for N-body simulations. This is implemented using drift and kick operators D and K, which advance the canonical variables as $\{\boldsymbol{x}_i(t)\} \xrightarrow{D(\Delta t)} \{\boldsymbol{x}_i(t+\Delta t)\}$, $\{\boldsymbol{q}_i(t)\} \xrightarrow{K(\Delta t)} \{\boldsymbol{q}_i(t+\Delta t)\}$. Discretising (39), their implementations become[16]

$$\begin{pmatrix} \{\boldsymbol{x}_i(t)\} \\ \{\boldsymbol{q}_i(t)\} \end{pmatrix} \xrightarrow{D(\Delta t)} \begin{pmatrix} \left\{ \boldsymbol{x}_i(t) + \dfrac{\boldsymbol{q}_i(t)}{m} \displaystyle\int_t^{t+\Delta t} \frac{\mathrm{d}t'}{a^2(t')} \right\} \\ \{\boldsymbol{q}_i(t)\} \end{pmatrix}, \tag{40}$$

$$\begin{pmatrix} \{\boldsymbol{x}_i(t)\} \\ \{\boldsymbol{q}_i(t)\} \end{pmatrix} \xrightarrow{K(\Delta t)} \begin{pmatrix} \{\boldsymbol{x}_i(t)\} \\ \left\{ \boldsymbol{q}_i(t) + \boldsymbol{f}_i(t) \displaystyle\int_t^{t+\Delta t} \frac{\mathrm{d}t'}{a(t')} \right\} \end{pmatrix}. \tag{41}$$

To evolve the synchronised system $(\{\boldsymbol{x}_i(t)\}, \{\boldsymbol{q}_i(t)\})$ it is first desynchronised by applying $K(\Delta t/2)$. The system is then evolved through repeated application of $D(\Delta t)$ followed by $K(\Delta t)$, under which $\{\boldsymbol{x}_i\}$ and $\{\boldsymbol{q}_i\}$ take turns leapfrogging past each other in time. Re-synchronisation of the canonical variables is achieved by some final drift and kick of appropriate size less than or equal to Δt.

Individual time steps As the non-linear P^{3}M time step limiter of Figure 2 is set through the root mean square velocity of the particle distribution, the resulting limit on Δt will be appropriate for typical particles, but not all. In particular, particles in high-density regions will tend to have much larger velocities, in turn requiring smaller time steps. One could lower the proportionality factor of the P^{3}M limiter accordingly, but at the cost of having unnecessarily fine time steps for the majority of the particles, wasting computational resources. Inspired by the approach of Springel (2005b), CONCEPT 1.0 instead allows each individual particle i to be updated on a time scale $\Delta t/2^{\ell_i}$, where $\ell_i \in \mathbb{N}_0$ is called the *rung*. Particles on rung 0 follows the global time-stepping, while particles on higher rungs receive short-range forces at a higher rate. The slowly varying and collectively computed long-range force remains as is, i.e. it follows the rhythm of rung 0.

With each particle assigned a rung, the system is evolved using a hierarchical scheme demonstrated by Figure 3, here shown for $n_{\text{rung}} = 3$ rungs. In practice, this number dynamically adapts as needed, though with a default maximum value of 8. Though particles act as 'receivers' only during kicks of the given rung in which they are assigned, they act as 'suppliers' for kicks within every rung. This asymmetry breaks strict symplecticity and momentum conservation, though the errors introduced are so small that this is of no concern[17].

To determine which rung ℓ_i a given particle i belongs to, we impose that it must not accelerate across a certain fraction[18] η of its softening length ϵ within the time $\Delta t/2^{\ell_i}$, disregarding its initial velocity. That is,

$$\ell_i(t) = \max\left(0, \left\lceil \log_2 \Delta t \sqrt{\frac{|\boldsymbol{a}_i(t)|}{2\eta\epsilon}} \right\rceil\right) \tag{42}$$

where $\boldsymbol{a}_i$ is the 'comoving acceleration subject to $\boldsymbol{q}_i = \boldsymbol{0}$', which from (39) is $\dot{\boldsymbol{q}}_i/(a^2 m)$. This is implemented as

$$\boldsymbol{a}_i(t) = [\boldsymbol{q}_i(t) - \boldsymbol{q}_i(t_{\text{prev}})] \left[m \int_{t-t_{\text{prev}}}^{t} \mathrm{d}t' a^2(t') \right]^{-1}, \tag{43}$$

where $t_{\text{prev}} < t$ refers to the time of the previous short-range kick undertaken by the particle. At the beginning of the simulation no such previous time exists, and so a "fake" kick is computed without applying the resulting momentum updates.

[15] Beyond striking a good balance, a period of 8 steps plays well with the non-linear fluid implementation as described in Dakin et al. (2019a). Should the maximum allowed value of Δt *de*crease below its current value, the current period is terminated early.

[16] Importantly, $\boldsymbol{f}_i$ itself has no explicit dependence on a, as seen from e.g. (36).

[17] GADGET-4 (Springel et al. 2020) implements a time-stepping scheme similar to the one used in CONCEPT 1.0 as well as one with manifest momentum conservation. This other scheme does not deliver significant improvements to the accuracy, but does come at the cost of additional force computations.

[18] In GADGET-2 the corresponding parameter is called `ErrTolIntAccuracy` and typically has a value of $\eta = 0.025$, which is also chosen as the default value used by CONCEPT 1.0.

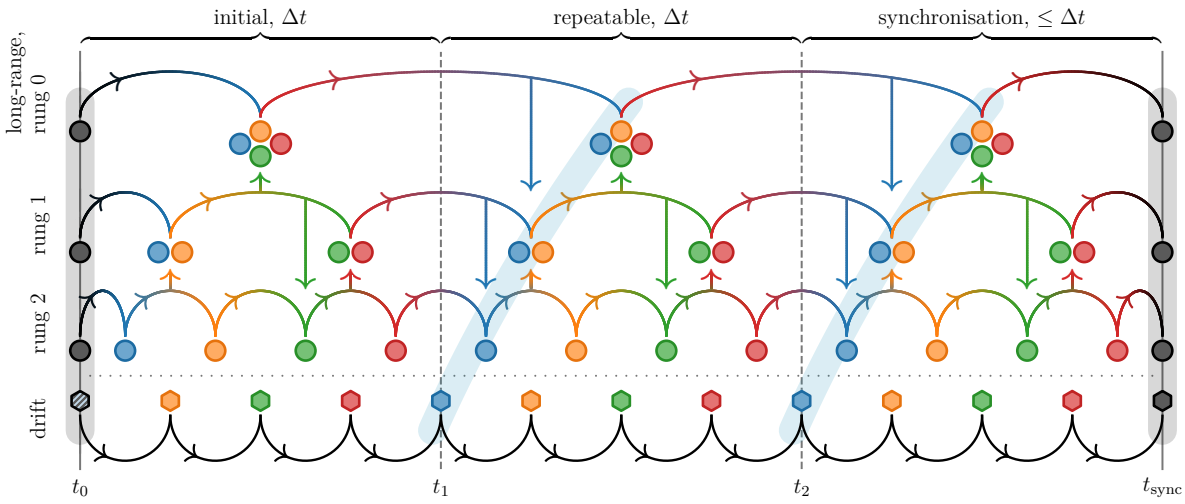

Figure 3. Time-stepping scheme of CONCEPT 1.0 using rung-based leapfrogging. The series of connected hexagons indicates the discrete timeline followed by the particle positions $\{\boldsymbol{x}_i\}$ as they are evolved through drift operations. Note that all positions remain mutually synchronised throughout time. The momenta $\{\boldsymbol{q}_i\}$ are distributed among the different rungs, each with its own discrete timeline indicated by horizontally connected circles. For clarity, we consider the case of only $n_{\text{rung}} = 3$ rungs. At the initial time t_0, all rungs are synchronised mutually and with the positions, as indicated by the vertical band covering the dark circles and the dark (and blue) hexagon. An initial 'half' kick of size $\Delta t/2^{\ell+1}$ is now applied within each rung ℓ, evolving them forward in time to the blue circles. As kicks of different rungs commute (Newtonian gravitation has no dependency on momentum), the order in which these kicks are applied do not matter. Considering the two-coloured hexagon now as blue, the whole system is now in a blue state. Through rung-based kicks and collective drifts, the system now changes state from blue → yellow → green → red → blue, each cycle amounting to a full global time step Δt. We have $2^{n_{\text{rung}}-1} = 4$ types (colours) of states due to the choice of $n_{\text{rung}} = 3$. The changes of state respect the leapfrogging scheme for all rungs and are self-similar from one rung to the next: From blue to yellow, one drift of size $\Delta t/2^{\ell \geq n_{\text{rung}}-1}$ is followed by a similar sized kick, only applicable to the highest rung $\ell = n_{\text{rung}} - 1 = 2$. All lower rungs evolve trivially from blue to yellow. Yellow → green consists of a similar drift followed by a kick of size $\Delta t/2^{\ell \geq n_{\text{rung}}-2}$ in the highest two rungs $\ell \in \{2, 1\}$. Green → red is similar to blue → yellow. Red → blue consists of the usual drift followed by a kick of size $\Delta t/2^{\ell \geq n_{\text{rung}}-3}$, i.e. a kick to all rungs $\ell \in \{2, 1, 0\}$. Once back at a blue state, a full time step Δt has been completed, as indicated by the tilted blue band. Note though that only the positions are truly at $t_1 = t_0 + \Delta t$, while the momenta are all "half a step ahead". Synchronisation at some arbitrary time t_{sync} is achieved simply by restricting the drifts and kicks to not evolve past this time, while otherwise keeping the scheme as is. Once synchronised, all rungs are recomputed and assigned. After a kick within a rung, some particles may accelerate enough so that they no longer belong within their given rung, in accordance with (42). Such particles jump to a more appropriate neighbouring rung by making their next kick either ½ or ¾ as large as usual, as indicated by the vertical arrows. Jumping to a lower rung is only possible at every other kick. In the above, 'kicks' really refer to momentum updates due to short-range forces only, i.e. the lower half of (36). The long-range forces are applied to all particles whenever rung 0 is kicked.

3 CODE VALIDATION AND COMPARISON

This section seeks to demonstrate the correctness of the results obtained with CONCEPT 1.0. This is done by comparing the power spectra of CONCEPT 1.0 simulations to those of similar GADGET simulations. This strategy thus presupposes the correctness of GADGET itself, which is well motivated by its wide usage and thorough testing over the past two decades.

3.1 Simulation setup

GADGET-like CONCEPT simulations A large effort has gone into making CONCEPT consistent with general relativistic perturbation theory. Thus, the large-scale power spectrum obtained from CONCEPT simulations should agree with that of linear Einstein-Boltzmann codes such as CLASS (Blas et al. 2011), which is successfully demonstrated in Tram et al. (2019); Dakin et al. (2019c,b). To this end, CONCEPT makes use of the full CLASS background and employs the N-body gauge framework. Initial conditions generated by CONCEPT are thus in N-body gauge. During simulation, this gauge is preserved by continually applying linear gravitational effects from the non-matter species[19] to the particles, implemented using PM techniques.

Table 1. Cosmological parameters used for all simulations.

Parameter	Value
H_0	$67\,\text{km}\,\text{s}^{-1}\,\text{Mpc}^{-1}$
Ω_{b}	0.049
Ω_{cdm}	0.27
A_{s}	2.1×10^{-9}
n_{s}	0.96

Besides CONCEPT, this strategy for making simulations consistent with general relativistic perturbation theory is further adopted by the PKDGRAV3 code (Euclid Collaboration et al. 2021), though GADGET-4 remains purely Newtonian. For a proper comparison between CONCEPT and GADGET, we then need to run CONCEPT in a 'GADGET-like' mode. We still generate all simulation initial conditions using CONCEPT in its 'standard' mode, and so the simulations start off in

[19] Here photons and neutrinos, both of which are necessarily part of the CLASS cosmology.

N-body gauge. This is contrasted with typical Newtonian setups, where the initial conditions are in no well-defined gauge at all, but has been back-scaled from the full, linear $a = 1$ solution in order to ensure agreement with relativistic perturbation theory on large scales at the present day. As we do not apply radiation perturbations during the simulations nor make use of back-scaled initial conditions, our simulations are not consistent with either approach. We stress that this does not affect the results in any appreciable way. What is important for the comparisons is that CONCEPT and GADGET make use of exactly the same initial conditions and simulation approach.

Leaving out the general relativistic correction kicks during CONCEPT evolution is easy, as these are only applied once explicitly specified in the parameter file. For the background evolution, CONCEPT inherits the tabulated solution from CLASS (incorporating radiation), whereas GADGET solves the matter + Λ Friedmann equation internally. This simplified background can be used within CONCEPT as well, in which case it is likewise solved internally by the N-body code. Lastly, the two codes differ in how they place the PM grid, CONCEPT 1.0 using cell-centred grid values and GADGET using cell-vertex grid values. In effect, the PM grids of the codes are relatively displaced by half a grid cell, $L_\varphi/2$, in all three directions. This makes a difference as the positions of the particles in the initial conditions are specified with respect to absolute space, not the PM grid. Though any effect on results from a purely numerical aspect such as this goes to demonstrate non-convergence of the solution, it is preferable to use identical PM setups when the comparison is between codes, as opposed to the absolute result. Thus for these tests, all grids within CONCEPT (including that used for initial condition generation) has been switched to cell-vertex mode. With these changes to the standard CONCEPT setup, we are ready to perform GADGET-like CONCEPT simulations[20].

Parameters All CONCEPT and GADGET simulations in this section use the cosmology as specified in Table 1 and other simulation parameters as specified in Table 2, with non-listed parameters taking on default[21] CONCEPT 1.0 values. For GADGET parameters that do not have a CONCEPT equivalent, we likewise seek to employ default values. However, GADGET-2 does not have a proper notion of default parameter values, and so we specify our chosen parameter values specific to GADGET-2 in Table 3, with parameters not listed there (nor in Tables 1 or 2) taking on values as suggested by the GADGET-2 user guide (Springel 2005a).

We settle for $N = 1024^3$ particles and thus a PM grid of size $n_\varphi = 2048$, and run simulations for box sizes $L_{\text{box}} \in \{2048, 1024, 512, 256\}\,\text{Mpc}/h$. All power spectra are computed with CONCEPT using a grid similarly of size 2048, employing PCS interpolation and interlacing (Hockney & Eastwood 1988).

[20] The documentation includes a section on how to perform GADGET-like simulations in practice.
[21] CONCEPT inherits non-specified cosmological parameters from CLASS.

Table 2. Simulation parameters used for all simulations unless explicitly stated otherwise, with the number of particles N and the box size L_{box} as free parameters.

Parameter	Symbol	Value
Softening length	ϵ	$0.03\,L_{\text{box}}/\sqrt[3]{N}$
PM grid size	n_φ	$2\sqrt[3]{N}$
Short-/long-range force split scale	x_{s}	$1.25\,L_{\text{box}}/n_\varphi$
Short-range cut-off	x_{r}	$4.5\,x_{\text{s}}$
Initial scale factor	a_{begin}	0.01

Table 3. Simulation parameters specific to GADGET.

Parameter	Value
`MaxSizeTimestep`	0.03
`TypeOfOpeningCriterion`	1
`ErrTolForceAcc`	0.005
`TreeDomainUpdateFrequency`	0.1

Table 4. Parameters for high-precision GADGET-2 simulations.

Parameter	Value
`ErrTolForceAcc`	0.001
`TreeDomainUpdateFrequency`	0.05

3.2 Comparison to GADGET

In Figure 4 we show absolute power spectra from the CONCEPT and GADGET-2 simulation in the $512\,\text{Mpc}/h$ box. Very good agreement between the codes is evident for all scales and times. This is impressive given that the non-linear power grows by more than a factor of 2×10^5 during the course of the simulations, and that the non-linear small-scale power at $z = 0$ is more than 30 times greater than its linear counterpart, demonstrating high non-linearity.

For a more precise comparison between the CONCEPT and GADGET results, their relative power spectra are shown in Figure 5, this time for all four box sizes. Here we see extraordinary good agreement between the two codes, for all scales and times irrespective of the box size. In all cases, the power spectra agree almost perfectly at large scales. Below some particular scale the results begin to diverge, with CONCEPT predicting slightly less power than GADGET for large box sizes and early times (low clustering) and slightly more power than GADGET for small box sizes and late times (high clustering), culminating in $\sim 1\,\%$ difference at the Nyquist scale.

Choosing a $1\,‰$ relative difference as a proxy for the scale at which the results begin to diverge from each other, we find this scale to be $k_{\text{div}} \approx 24 \times 2\pi/L_{\text{box}}$, meaning it is relative to the resolution of the simulation(s) and does not depend on some absolute scale. That is, the difference between the codes is roughly independent on the box size / particle resolution. This $1\,‰$ relative difference is shown in Figure 5 as the innermost grey bands.

We do see some difference as we vary the box size. In particular, CONCEPT predicts slightly less power than GADGET for large boxes and slightly more power than GADGET for small boxes. As the main difference between the codes is that GADGET approximates the short-range force using a tree while CONCEPT does not, we might hope that this difference is the main source of their disagreement. To test this we additionally run GADGET-2 using higher-precision tree settings

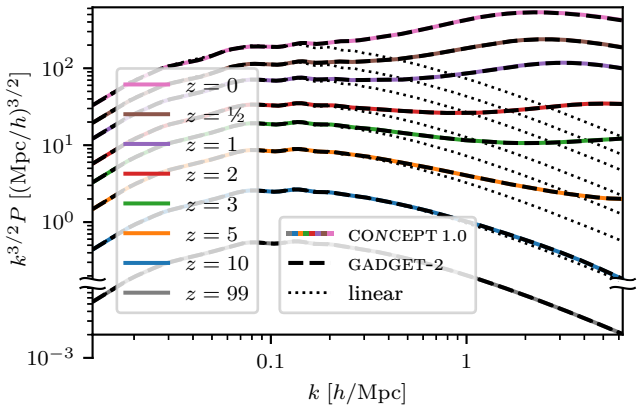

Figure 4. Evolution of the power spectrum in CONCEPT 1.0 and GADGET-2 simulations with $N = 1024^3$ particles in a 512 Mpc/h box. For reference, the linear power spectrum is shown as well. Due to the large gap between the initial power at $z = 99$ and the power at the first output time $z = 10$, the vertical axis has been broken in two. We plot $k^{3/2}P$ rather than P or k^3P as this results in less steep spectra, allowing for a more detailed view.

as listed in Table 4 (all other parameters stay the same), rebuilding the tree anew more frequent and traversing it more deeply. The results of such high-precision GADGET-2 simulations are also shown in Figure 5, compared against the same CONCEPT result as before. For all boxes, increasing the tree precision of GADGET has the effect of lowering the power, leading to better agreement with CONCEPT for the smaller box sizes. Interestingly, improving the tree approximation worsens the agreement for the larger box sizes. This is most likely related to the "fuzzy short-range interaction boundary" of GADGET-2 discussed further down.

Increasing the precision of the tree force as in Table 4 has another effect. Looking carefully at all but the smallest box size of Figure 5, we see that the large-scale power spectra very slightly disagree (at a few tens of a per mille) for the 'standard'-precision GADGET simulations, whereas this constant offset drops by a factor ~ 10 with the high-precision GADGET runs. We stress that even with the 'standard'-precision GADGET runs, this constant offset is very tiny. Indeed, in order to obtain this good of an agreement we have had to update the values of various physical constants used in GADGET-2 to match the exact values used in CONCEPT 1.0. Here the most important one is probably the gravitational constant, which GADGET-2 sets to $G = 6.672 \times 10^{-11}\,\mathrm{m^3\,kg^{-1}\,s^{-2}}$ whereas CONCEPT 1.0 uses the latest value from the Particle Data Group (2020) $G = 6.674\,30 \times 10^{-11}\,\mathrm{m^3\,kg^{-1}\,s^{-2}}$. Without this matching of the values of physical constants, the constant offset between GADGET-2 and CONCEPT 1.0 grows by a factor ~ 2.5.

The relative spectra at the largest scales for the largest box size $L_\mathrm{box} = 2048\,\mathrm{Mpc}/h$ develops a slight but persistent wiggle early on. This effect not only remains but worsens for still larger boxes, and so it is associated with large physical scales, irrespective of the simulation resolution. The feature is robust against increased temporal precision of either code, and also against lowering the tree opening angle of GADGET. However, the wiggle can be made to completely disappear by running GADGET with a slightly increased short-range cut-off scale, $x_\mathrm{r} \gtrsim 5.0\,x_\mathrm{s}$. This is surprising, as the short-range force should have no effect on the largest scales. Indeed, running CONCEPT with a similarly increased x_r only perturbs its spectrum at small scales, leaving the larger scales invariant.

While Figure 5 does not show the case of increased x_r for the largest box size $L_\mathrm{box} = 2048\,\mathrm{Mpc}/h$, it does show $x_\mathrm{r} = 5.5x_\mathrm{s}$ for $L_\mathrm{box} = 512\,\mathrm{Mpc}/h$ at early times, which we see reduces the discrepancy between CONCEPT 1.0 and GADGET-2 to the point where they now agree at the per mille level at all relevant scales. We believe this to be explained by the cut-off x_r being strictly enforced at the particle-particle level in CONCEPT 1.0, whereas the tree in GADGET-2 makes this cut-off somewhat fuzzy due to the physical extent of its nodes. At low clustering this difference ought to be especially pronounced as a lot of precise force-cancellation takes place for the near-homogeneous particle distribution. Including even slightly different sets of particles in any given force computation then start to make a significant difference. In Figure 5 we indeed only find a deficit of power in GADGET-2 relative to CONCEPT 1.0 at low clustering (large boxes and/or early times). Increasing x_r pushes the fuzzy interaction boundary in GADGET-2 to greater distances, with the short-range force exponentially decaying, decreasing its significance. We note that no drastic change to the late-time relative spectrum comes about due to the increase in x_r, hence why Figure 5 only shows the $x_\mathrm{r} = 5.5x_\mathrm{s}$ case at early times. Also noteworthy is the fact that the $x_\mathrm{r} = 5.5x_\mathrm{s}$ lines of Figure 5 use the high-precision settings for GADGET-2, without which the increase of the cut-off scale from $x_\mathrm{r} = 4.5x_\mathrm{s}$ to $x_\mathrm{r} = 5.5x_\mathrm{s}$ actually degrades the agreement with CONCEPT 1.0 at all times.

4 CODE PERFORMANCE

With the correctness of CONCEPT 1.0 established by the previous section, we now set out to demonstrate various performance aspects of the code, both internally and by comparison to GADGET-2/4. All simulations employed in this section use the cosmology as specified in Table 1 along with other simulation parameters as specified in Table 2, just as in the previous section.

All simulations (of this and the previous section) are carried out on the Grendel compute cluster at Centre for Scientific Computing Aarhus (CSCAA), using compute nodes consisting of two 24-core Intel Xeon Gold 6248R CPUs at 3.0 GHz, interconnected with Mellanox EDR Infiniband at 100 Gbit/s. Both CONCEPT 1.0 and GADGET-2/4 are built using GCC 10.1.0 with optimisations `-O3 -funroll-loops -ffast-math -flto` and linked against FFTW 3.3.9 (CONCEPT 1.0 and GADGET-4) or 2.1.5 (GADGET-2), itself built similarly though without link-time optimisations `-flto`. All of CONCEPT 1.0, GADGET-2/4 and FFTW 2/3 are run in double-precision. All is linked against and run with OpenMPI 4.0.3.

4.1 Weak scaling

Here we study the 'weak' scaling of CONCEPT 1.0, i.e. how the computation time is affected for increased problem size while keeping the computational load per process fixed. That is, we hold $L_\mathrm{box} \propto \sqrt[3]{N}$ and $n_\mathrm{p} \propto N$ for varying N, n_p

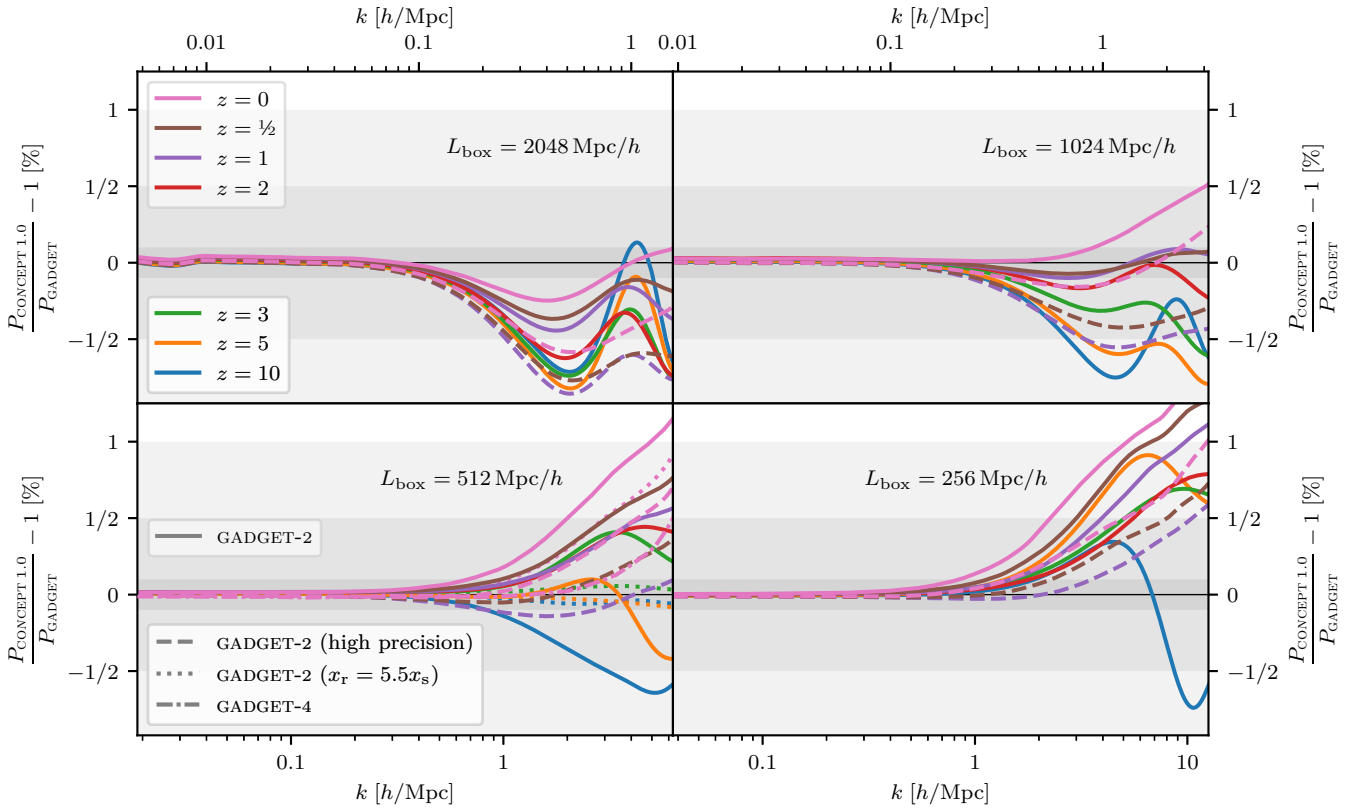

Figure 5. Relative power spectra between CONCEPT 1.0 and GADGET-2, for simulations with $N = 1024^3$ particles and four different box sizes. The relative spectra are shown at various times, with the initial time $z = 99$ left out as here the CONCEPT and GADGET spectra match exactly by construction. The full lines correspond to GADGET-2 simulations using the 'standard' GADGET-2 precision settings of Table 3. For late times, the same CONCEPT spectra are additionally shown relative to GADGET-2 spectra from simulations using the high-precision settings of Table 4, using dashed lines. For the $512\,\text{Mpc}/h$ box, we additionally show the case of increased $x_\text{r} = 5.5x_\text{s}$ (in both CONCEPT 1.0 and high-precision GADGET-2) at early times. Grey bands mark relative errors of 1 %, ½ % and 1 ‰. For each panel, the k axis extends to the Nyquist scale of the particle grid, $k_\text{Nyquist} = \sqrt[3]{N}/2 \times 2\pi/L_\text{box} = 1024\pi/L_\text{box}$.

being the total number of MPI processes, each running on a dedicated CPU core. For perfect weak scaling, increasing the problem size together with the number of CPU cores in lockstep should not incur any increase to the computation time.

Choosing $L_\text{box} = 2\sqrt[3]{N}\,\text{Mpc}/h$ and $N/n_\text{p} \sim 204^3$, the weak scaling of CONCEPT 1.0 is shown in Figure 6. From the top panels, we see that the short-range computation exhibits almost perfect weak scaling at early times and still reasonably good weak scaling at late times. The long-range computation has a less optimal scaling, even overtaking the short-range computation at early times when having many processes. This suboptimal scaling of the long-range computation is owed mostly to the FFTs, as evident from the dashed orange lines having similar shape to the full orange lines but with steeper slope. At late times the computation time is completely dominated by the short-range computation, rendering the bad scaling of the long-range part ignorable. In all, this leads to reasonably good overall weak scaling of CONCEPT 1.0.

Looking at the lower panel of Figure 6, the suboptimal weak scaling of the long-range computation is again evident, here as a clear separations between (most of) the dashed lines. The long-range computation time is however close to constant throughout the simulation. For the short-range times, the different simulations follow each other closely, though still with larger simulations being somewhat slower. The cost of the short-range computation increases as the universe becomes more clustered. Here this effect kicks in at $z \sim 10$ and continues to the present day. This increase is caused by the particle-particle interaction count going up with the amount of clustering. As the load imbalance remains small even at late times and high core count, this is not a significant factor in the slowdown of the short-range computation over the course of the simulation time span.

4.2 Strong scaling

Here we study the 'strong' scaling of CONCEPT 1.0, i.e. how the computation time is affected when increasing the number of CPU cores used within the simulation, keeping everything else fixed. That is, for some chosen L_box and N we vary n_p. For perfect strong scaling, the computation time is required to drop linearly with the number of cores, i.e. the computation time should be inversely proportional to the computational firepower thrown at the problem.

Figure 7 shows the strong scaling of CONCEPT 1.0 for $L_\text{box} = 1024\,\text{Mpc}/h$, $N = 512^3$. The short-range computation scales very well, especially at early times, as evident from

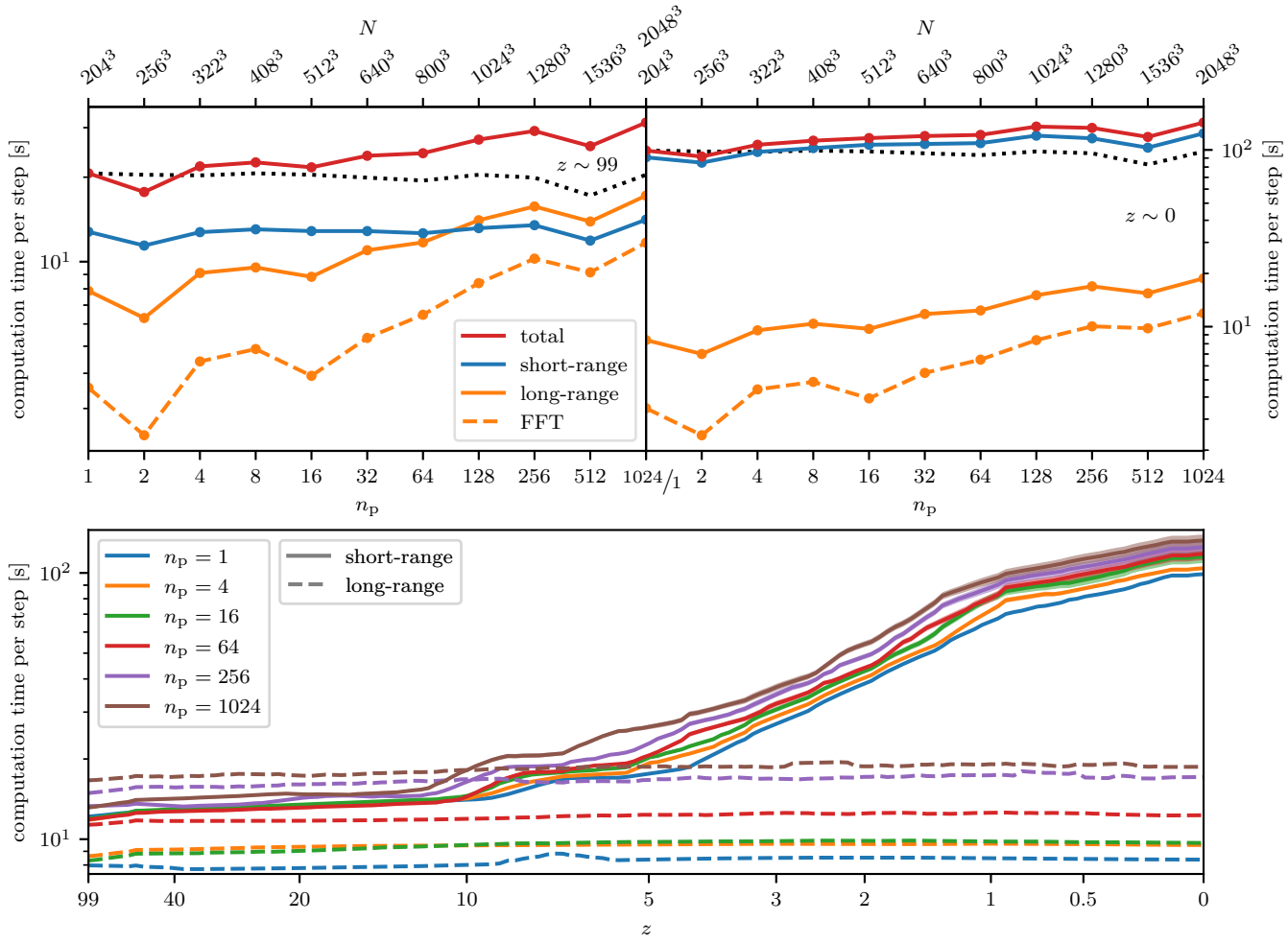

Figure 6. Weak scalability for CONCEPT 1.0 simulations in boxes of size $L_\mathrm{box} = 2\sqrt[3]{N}\,\mathrm{Mpc}/h$, keeping the particle load per process N/n_p roughly fixed at 204^3 particles. The simulations range from serial all the way to 1024 cores and 2048^3 particles. The top panels show the wall-clock computation time per time step near the beginning $z \sim 99$ and end $z \sim 0$ of the simulations, averaged over 8 time steps. Full lines show the total computation time and its dominant components, namely the short-range and long-range gravitational computations. The FFT part of the long-range computation is further separated out and shown using dashed lines. Finally, perfect weak scaling of the total computation time is shown with dotted lines. We cannot explain the dip at $n_\mathrm{p} = 2$, which is seen in both the short- and long-range computation time. The dip at $n_\mathrm{p} = 512$ is caused by having slightly smaller particle load per process than usual (note that this dip appears even in the perfect scaling). Here we ought to use $N \approx 1632^3$, but $n_\varphi = 2\sqrt[3]{N}$ must be divisible by $n_\mathrm{p} = 512$ due to restrictions in CONCEPT (the FFTW slabs must be evenly divisible amongst the processes).
The lower panel shows the evolution of the computation time over the simulation time span for every other simulation, averaged over 8 time steps. Here only the short-range (full) and long-range (dashed) computation times are shown. Towards $z = 0$ the load imbalance can be seen as a widening of the short-range lines, with the widths given by twice the standard deviation of the individual short-range computation times among the processes within a given simulation. The redshift z axis is shown as scaling linearly with the simulation time steps.

the upper panels of the figure. The long-range computation shows a somewhat worse strong scaling behaviour than the short-range computation, even overtaking as the dominant computation for high core counts at early times. As evident from the similar shape of the full and dashed orange curves, this behaviour is caused to the FFTs.

The sudden jump in the trend line of the long-range computation time at $n_\mathrm{p} \geq 32$ is probably explained by the $n_\mathrm{p} < 32$ simulations all running entirely within a single CPU, whereas the $n_\mathrm{p} \geq 32$ simulations all utilise several CPUs, even distributed over several compute nodes for $n_\mathrm{p} \geq 64$. As the short-range computation vastly dominates at later times, this suboptimal strong scaling of the long-range force is not an issue in practice. In total, this makes the overall strong scaling of CONCEPT 1.0 reasonably good.

The top panels of Figure 7 includes the odd case of $n_\mathrm{p} = 37$, a prime. This is to demonstrate that CONCEPT 1.0 may be run with any number of processes and that the nature of this number does not significantly affect its performance. The computation time of the FFTs does increase noticeably, but as usual this effect is dwarfed by the dominance of the short-range computation at late times. Over the course of a whole simulation then, the nature of n_p is of little importance.

For the lower panel of Figure 7, decent strong scalings

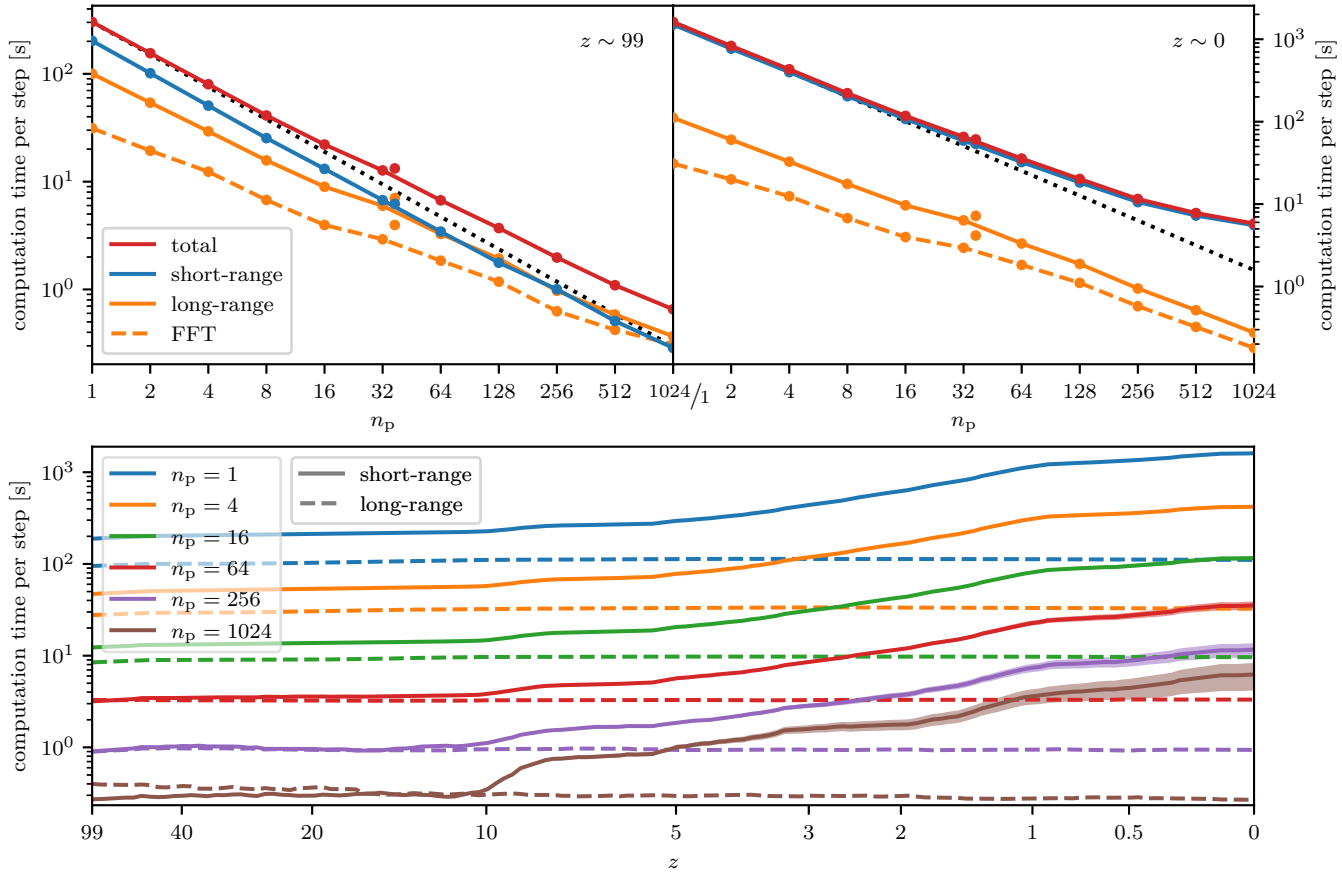

Figure 7. Strong scalability for CONCEPT 1.0 simulations with $N = 512^3$ particles in a box of size $L_\mathrm{box} = 1024\,\mathrm{Mpc}/h$. The simulations range from serial all the way to 1024 cores, corresponding to a load ranging from 512^3 to $\sim 50^3$ particles per core. The top panels show the wall-clock computation time per time step near the beginning $z \sim 99$ and end $z \sim 0$ of the simulations, averaged over 8 time steps. Full lines show the total computation time and its dominant components, namely the short-range and long-range gravitational computations. The FFT part of the long-range computation is further separated out and shown using dashed lines. Finally, perfect strong scaling of the total computation time is shown with dotted lines. In addition to having the number of processes being powers of two, we further show the case of $n_\mathrm{p} = 37$ as disconnected dots. As CONCEPT requires n_φ to be divisible by n_p, this one simulation has been run with slightly increased grid size $n_\varphi = 1036$ instead of the usual $n_\varphi = 2\sqrt[3]{N} = 1024$ used for the other simulations.
The lower panel shows the evolution of the computation time over the simulation time span for every other simulation, averaged over 8 time steps. Here only the short-range (full) and long-range (dashed) computation times are shown. Towards $z = 0$ the load imbalance can be seen as a widening of the short-range lines, with the widths given by twice the standard deviation of the individual short-range computation times among the processes within a given simulation. The redshift z axis is shown as scaling linearly with the simulation time steps.

of the short- and long-range computations are evident from the nearly equidistant separations between the lines. As for the weak scaling results of Figure 6, we again find the computation time of the long-range force to be mostly invariant over the simulation time span, and that the cost of the short-range computation increases as the universe becomes more clustered. At late times, load imbalance starts to become significant for the simulations with large core counts, degrading the strong scaling.

4.3 Absolute performance

The above explorations of the weak and strong scaling of CONCEPT 1.0 demonstrate excellent scaling behaviour when increasing the problem size N and/or the core count n_p. Keeping both of these fixed, the computation time required for a given simulation depends on the level of clustering, which in turn depends on the particle resolution through the box size L_box. We thus now want to investigate the absolute performance of CONCEPT 1.0 as a function of the particle resolution, which we do by comparing the total computation time of CONCEPT 1.0 simulations to equivalent GADGET-2 simulations.

Even though Figure 5 generally demonstrates improved agreement between CONCEPT 1.0 and GADGET-2 for the high-precision GADGET settings of Table 4, we here exclusively run GADGET with the 'standard' settings of Table 3. We choose to do so as it would be unfair not to allow GADGET to make good use of its tree approximation when comparing performance, given that the observed improvements brought about by the high-precision settings are relatively minor. For the two larger boxes of Figure 5, running GADGET-2 with the high precision settings only incurs a performance hit of a

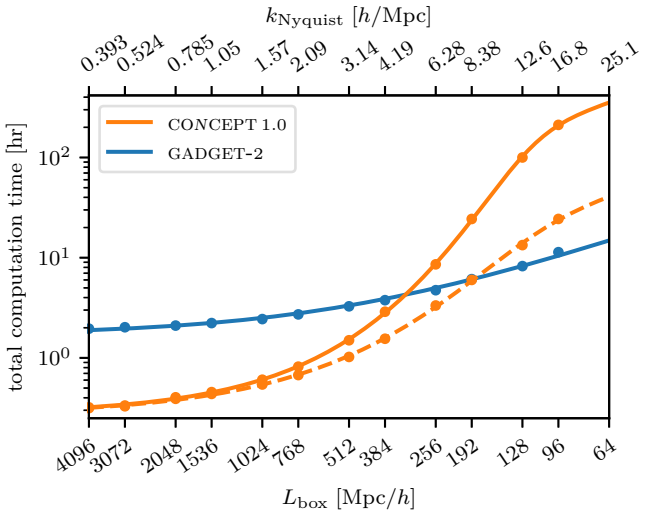

Figure 8. Scalability across particle resolutions for CONCEPT 1.0 and GADGET-2 simulations with $N = 512^3$ particles, all run using 64 processes evenly distributed across two dedicated compute nodes. The wall-clock computation times of entire simulations are plotted against the box size L_{box}, from a very large box $L_{\text{box}} = 4096\,\text{Mpc}/h$ down to a very small box $L_{\text{box}} = 96\,\text{Mpc}/h$. Alternatively, the horizontal axis may be viewed in terms of the Nyquist scale of the particle grid, $k_{\text{Nyquist}} = \sqrt[3]{N}/2 \times 2\pi/L_{\text{box}} = 512\pi/L_{\text{box}}$. The blue line is a linear (in k_{Nyquist}) fit to the GADGET-2 points, whereas the full orange line is a fit to the CONCEPT 1.0 points using separate linear behaviour at either end and a sigmoid transition between the two. The dashed orange line is constructed from the full orange line by subtracting wasted short-range computation time due to load imbalance, and so represent the absolute performance of CONCEPT 1.0 had it contained a perfect load balancing scheme.

few percent, though this grows to $\sim 30\,\%$ for the two smaller boxes.

In Figure 8 we plot the total computation times of CONCEPT 1.0 and GADGET-2 simulations for various box sizes, corresponding to Nyquist scales of the particle grid ranging from $k_{\text{Nyquist}} \sim 0.4\,h/\text{Mpc}$ to $k_{\text{Nyquist}} \sim 17\,h/\text{Mpc}$. For $k_{\text{Nyquist}} \lesssim 5\,\text{Mpc}/h$ CONCEPT 1.0 is much faster than GADGET-2, whereas GADGET-2 is much faster than CONCEPT 1.0 for $k_{\text{Nyquist}} \gtrsim 5\,\text{Mpc}/h$.

We believe that the superior performance of CONCEPT 1.0 at low to moderate clustering has two primary causes. First, the non-hierarchical tile + subtile data structure of CONCEPT 1.0 is much faster to traverse than the tree structure of GADGET, due to simple, precomputed access patterns and minimal pointer chasing. At low clustering, all particles have a similar number of short-range interaction partner particles, and so the benefits of the grouping carried out by the tree is minimal. At stronger clustering, the number of particle-particle short-range interactions increases drastically, which is then efficiently approximated by much fewer particle-node interactions using the tree, outweighing the more expensive tree walk. Second, CONCEPT 1.0 employs a much coarser time-stepping at high redshift than GADGET, as discussed in section 2.2. As evident from Figure 5 this does not induce noticeable artefacts in the solution.

The slowness of CONCEPT 1.0 at very high resolution means that it is currently impractical to use the code for simulations in this regime. Though a tree implementation in CONCEPT would undoubtedly speed up the expensive short-range computation at these resolutions, Figure 5 reveals a more important possible optimisation; load balancing. Currently CONCEPT does no attempt at balancing the computational load across the CPU cores, as discussed in section 2.1.3. At large clustering, this leads to correspondingly large load imbalance of the short-range computation, as visible in e.g. the lower panel of Figure 7. The dashed line in Figure 8 shows the theoretical computation time of CONCEPT 1.0 runs with the load perfectly balanced (assuming the balancing itself is cost free), and as so represents the best performance improvement we can hope to obtain were we to build load balancing into CONCEPT. Though still slower than GADGET-2 for runs with very high resolution, this alone would be enough to make it feasible to perform such simulations with CONCEPT.

The data points of Figure 8 are fitted to trend-lines. In the case of GADGET-2, a simple linear fit match the data nicely. In the case of CONCEPT 1.0, the scaling behaviour is less trivial. In the low-resolution regime CONCEPT 1.0 exhibits linear scaling as well. The other extreme is more tricky to gauge due to scarcity of data, but the fits suggests that here too it moves towards (a different) linear scaling, both in the actual case and with perfect load balancing. The different scaling behaviours at the two ends reflect the fact that at high resolution the short-range force completely dominate the computational budget, whereas at low resolution the long-range computation is comparably (if not more) expensive. In the case of GADGET, but the short- and long-range computation scales as $\mathcal{O}(N \log N)$, and so shifting the computational burden from one to the other does not significantly change the scaling behaviour.

4.4 Memory consumption

With the previous subsections having thoroughly investigated the time complexity of CONCEPT 1.0, let us now turn to its space complexity (consumption of memory).

To understand the memory usage of CONCEPT 1.0 we simply tally up the memory consumed by its major data structures, most important of which are the particle data arrays and the $\text{P}^{(3)}\text{M}$ mesh. The canonical vector variables of each particle contribute to the memory budget with 3 triplets of 8-byte (i.e. double precision) floats (position $\boldsymbol{x}_i$, momentum $\boldsymbol{q}_i$, momentum update $\Delta\boldsymbol{q}_i$), as well as 3 1-byte integers for keeping track of the rung ℓ_i. The tiling brings in another 8-byte integer per particle. At late times the number of allocated particles somewhat exceeds N due to particle exchange between the processes. The memory spent on the particles thus slightly increases during the simulation, why the above memory consumption should be scaled up by some small factor, say ~ 1.25.

Each of the n_φ^3 $\text{P}^{(3)}\text{M}$ grid cells store an 8-byte float, with 3 such global grids present in memory (domain-decomposed potential, slab-decomposed potential, force). All together, this yields a memory consumption of $M \approx (104\,N + 24\,n_\varphi^3)\,\text{B}$, where B is a byte. The tiles and their pre-computed pairings further contribute noticeably to the total memory, as do various buffers. Aided by measurements, we find the true memory consumption to be closer to

$$M \approx (120\,N + 28.3\,n_\varphi^3)\,\text{B}\,, \tag{44}$$

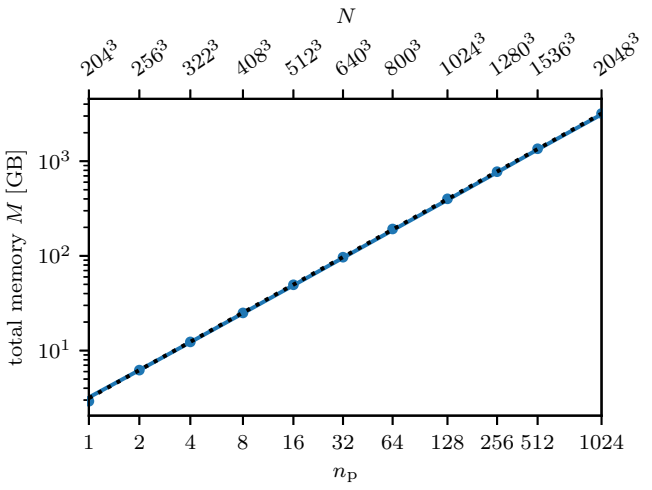

Figure 9. Memory scalability for CONCEPT 1.0 simulations in boxes of size $L_\mathrm{box} = 2\sqrt[3]{N}\,\mathrm{Mpc}/h$ at $z = 0$. While the blue points are data, the blue line is the estimate (45). The dotted black line is perfect scaling $M \propto N$, with outset in the $n_\mathrm{p} = 2$ data (the serial case $n_\mathrm{p} = 1$ is not representative due to a lack of communication buffers).

given the P^{3}M parameters (37).

Factoring in communication buffers and ghost layers, the memory consumption further depends on the number of processes n_p. Finally, libraries loaded by the code provide a constant minimum memory usage. In total, a good memory estimate for CONCEPT 1.0 comes out as

$$M \approx (0.119 + 0.144\,n_\mathrm{p} + 3.46 \times 10^{-7} N)\,\mathrm{GB}\,, \tag{45}$$

where our standard choice (38) has been used to eliminate n_φ^3 in favour of N.

Figure 9 demonstrates the validity of the memory estimate (45) for fixed particle resolution. We see that CONCEPT 1.0 follows this estimate nicely, and that the term proportional to N dominates for typical setups, leading to perfect scaling $M \propto N$. Higher particle resolutions will come at a somewhat higher proportionality factor, though with this scaling retained.

The memory usage (44) is very similar to what is reported for GADGET-2 in Springel (2005a), namely $\sim$ 110 bytes per particle and 24–32 bytes per P^{3}M grid cell, though this can be halved if using single-precision. While not feasible with CONCEPT 1.0 due to vastly increased computation time, GADGET may be run with a smaller P^{3}M grid than our standard choice (38), significantly reducing the memory requirement. As lowering n_φ shifts more of the computational burden onto the short-range computation, Figure 8 demonstrates this difference between the two codes nicely, with higher resolution corresponding to more expensive short-range computations and thus smaller n_φ. We note that decreasing n_φ from $2\sqrt[3]{N}$ to e.g. $1\sqrt[3]{N}$ does make GADGET significantly slower as well, but by an acceptable amount in the case of limited memory resources.

In practice, the availability of memory resources is rarely a limiting factor for most N-body simulations, with modern HPC CPUs each having access to hundreds of GB of RAM. With the total memory of simulations scaling as $M \propto N$ and the total computation time as (at best) $\propto N \log N$, the availability of memory will only become less of a problem in the future, assuming similar advances in computational throughput and memory technology.

4.5 Internal data structures

The P^{3}M method of CONCEPT 1.0 employs both temporal and spatial adaptiveness in the form of rung-based particle time-stepping as described in section 2.2.2 and dynamic domain-specific subtiling as described in section 2.1.3. With the overall code performance showcased in the previous subsections, let us now take a closer look at these dynamic data structures as a function of time and particle resolution.

4.5.1 Rung population

The left panel of Figure 10 show the rung population at $z = 0$ in simulations of different particle resolution. For very large boxes, only the few — here 4 — lowest rungs are populated. Increasing the particle resolution (lowering the box size) leads to migration of particles to higher rungs, slowly draining rung 0 and now populating rungs 4 and 5 as well. This is expected from the larger particle accelerations (see (42)) induced by the increased amount of clustering.

For $k_\mathrm{Nyquist} \gtrsim 2\,h/\mathrm{Mpc}$ however, the trend reverses and particles jump back down to the lower rungs. We can understand this perhaps surprising find by considering the interplay between rungs ℓ_i (42) and the global time step size Δt. From Figure 2 we see that we require $L_\mathrm{box} \gtrsim 2\sqrt[3]{N}\,\mathrm{Mpc}/h$ in order for the P^{3}M limiter not to dictate a lowering of Δt near $z = 0$. That is, $L_\mathrm{box} \sim 2\sqrt[3]{N}\,\mathrm{Mpc}/h$ is the smallest box one can choose before the global time step size is decreased as a result, and so this box size has the largest Δt in relation to the amount of clustering. As $L_\mathrm{box} \sim 2\sqrt[3]{N}\,\mathrm{Mpc}/h$ corresponds to $k_\mathrm{Nyquist} = 1.57\,h/\mathrm{Mpc}$, this exactly matches the observed behaviour of the left panel of Figure 10. We note that rungs $\ell > 5$ are obtainable through increased clustering, e.g. as the result of increasing the amplitude A_s of the primordial perturbations.

The right panel of Figure 10 shows the time evolution of the simulation with $L_\mathrm{box} = 2\sqrt[3]{N}\,\mathrm{Mpc}/h$ or equivalently $k_\mathrm{Nyquist} = 1.57\,h/\mathrm{Mpc}$. All particles start at rung 0 and stays there until a little after $z = 10$, after which rungs 1–3 are quickly populated, followed by rung 4 at $z \sim 4$ and finally rung 5 at $z \sim 1.5$, though with each higher rung occupying much fewer particles than the ones below. That non-linearity commence at around $z \sim 10$ is consistent with the sudden increase in short-range computation time seen in e.g. the lower panel of Figure 7, which we now understand as arising from an increase in kick operations due to additional rungs being populated.

The bulby look of the evolution of each rung count on the right panel of Figure 10 reflects the time step cycles of 8 steps, as described in section 2.2.1. At the end of each cycle, the global time step Δt is allowed to increase, prompting higher rungs as specified by (42). With all particles moving to their newly assigned rung before the next cycle begins, this results in steep increases to the count of rungs $\ell > 0$.

At $z \sim 1$ a qualitative change in behaviour is seen for the rung population of the right panel of Figure 10, where instead of migrating to higher rungs with time, the particles

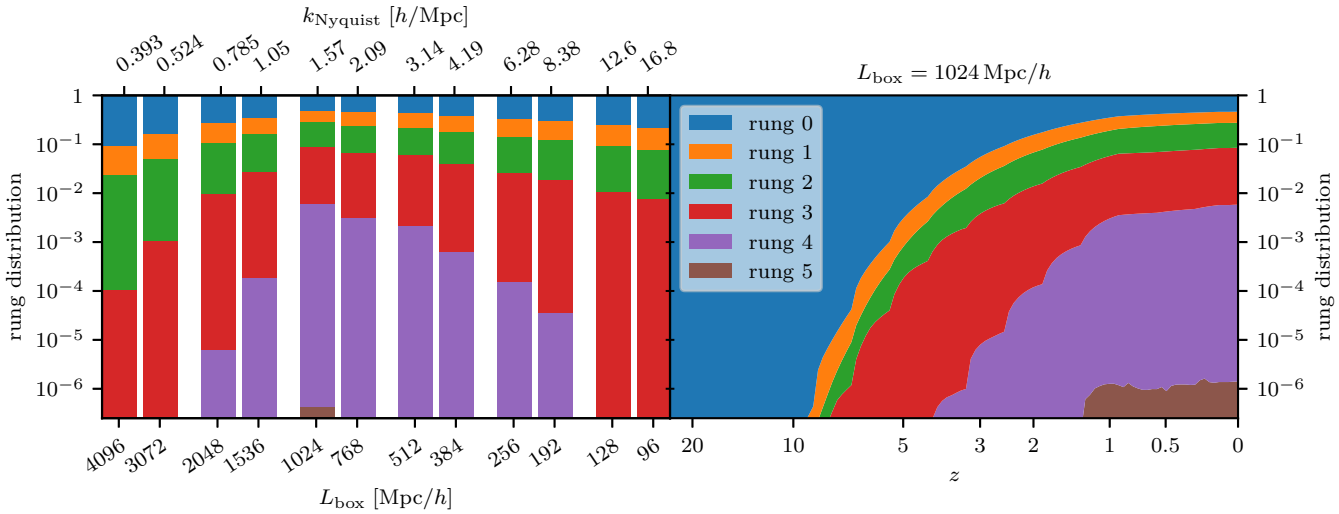

Figure 10. Distribution of particles across rungs in CONCEPT 1.0 simulations with $N = 512^3$ particles. The left panel shows stacked bar charts of the rung distribution at $z = 0$ for simulations with different box sizes L_{box} or equivalently different particle resolutions $k_{\text{Nyquist}} = \sqrt[3]{N}/2 \times 2\pi/L_{\text{box}} = 512\pi/L_{\text{box}}$. In all cases rung 0 is the most populous one, with the particle count within each rung rapidly declining for the higher rungs. Rungs 4 and 5 are only populated at intermediate box sizes, with particles within simulations in very large or very small boxes only occupying rungs 0–3. The right panel shows the temporal evolution of the rung population for the $L_{\text{box}} = 1024\,\text{Mpc}/h$ simulation, averaged over 8 time steps. All particles start on rung 0 and only begins to jump to higher rungs after $z = 10$. The redshift z axis is shown as scaling linearly with the simulation time steps.

all more or less stay on their given rung throughout the rest of the simulation. Once more we can understand this from Figure 2, where $z \sim 1 \Rightarrow a \sim 0.5$ is where the P^3M limiter begins to dominate, no longer allowing the global time step size Δt to drastically increase with each finished time step cycle. As the value of the P^3M limiter is determined by the root mean square velocity of the particle distribution itself (see section 2.2.1), the global time step Δt is now evolved in sync with the velocity distribution of the particles, hence why the particles now remain satisfied occupying the same rung for the rest of the simulation. Looking again at the lower panel of Figure 7 or Figure 6, this change in behaviour is once again seen in the short-range computation times, as the slopes suddenly decrease at $z \sim 1$. As the z axes are all shown as scaling linearly with time steps (as opposed to e.g. z itself, a or t), this slope is proportional to the increase in computation time from one step to the next

4.5.2 Subtile decomposition

5 DISCUSSION AND CONCLUSIONS

Summary. State where CONCEPT shines and where it does not. Discuss possible future upgrades regarding features and performance improvements.

ACKNOWLEDGEMENTS

We wish to thank Volker Springel for valuable discussions, in particular on the code comparison between CONCEPT 1.0 and GADGET-2/4. We are thankful to Tiago Castro for pointing out several bugs and shortcomings of CONCEPT prior to the 1.0 release. We acknowledge computing resources from the Centre for Scientific Computing Aarhus (CSCAA). This work was supported by the Villum Foundation.

REFERENCES

Angulo R. E., Pontzen A., 2016, Monthly Notices of the Royal Astronomical Society: Letters, 462, L1
Barnes J., Hut P., 1986, Nature, 324, 446
Behnel S., Bradshaw R., Citro C., Dalcin L., Seljebotn D. S., Smith K., 2011, Computing in Science & Engineering, 13, 31
Bertschinger E., 1998, Ann.Rev.Astron.Astrophys., p. 599
Blas D., Lesgourgues J., Tram T., 2011, JCAP, 1107, 034
Cooley J. W., Tukey J. W., 1965, Math. Comput., 19, 297
Couchman H., 1991, Astrophys.J.Lett., 368, L23
Dakin J., Brandbyge J., Hannestad S., Haugbølle T., Tram T., 2019a, Journal of Cosmology and Astroparticle Physics, 2019, 052
Dakin J., Hannestad S., Tram T., 2019b, Journal of Cosmology and Astroparticle Physics, 2019, 032
Dakin J., Hannestad S., Tram T., Knabenhans M., Stadel J., 2019c, Journal of Cosmology and Astroparticle Physics, 2019, 013
Efstathiou G., Eastwood J. W., 1981, Mon. Not. Roy. Astron. Soc., 194, 503
Euclid Collaboration et al., 2021, Monthly Notices of the Royal Astronomical Society, 505, 2840
Ewald P. P., 1921, Annalen der physik, 369, 253
Fornberg B., 1988, Mathematics of computation, 51, 699
Frigo M., Johnson S. G., 2005, Proceedings of the IEEE, 93, 216
Harris C. R., et al., 2020, Nature, 585, 357
Hernquist L., Bouchet F. R., Suto Y., 1991, The Astrophysical Journal Supplement Series, 75, 231
Hockney R. W., Eastwood J. W., 1988, Computer simulation using particles
Hockney R. W., Goel S., Eastwood J., 1974, Journal of Computational Physics, 14, 148
Hoerner S., 1960, Z. Astrophys.., 50, 180

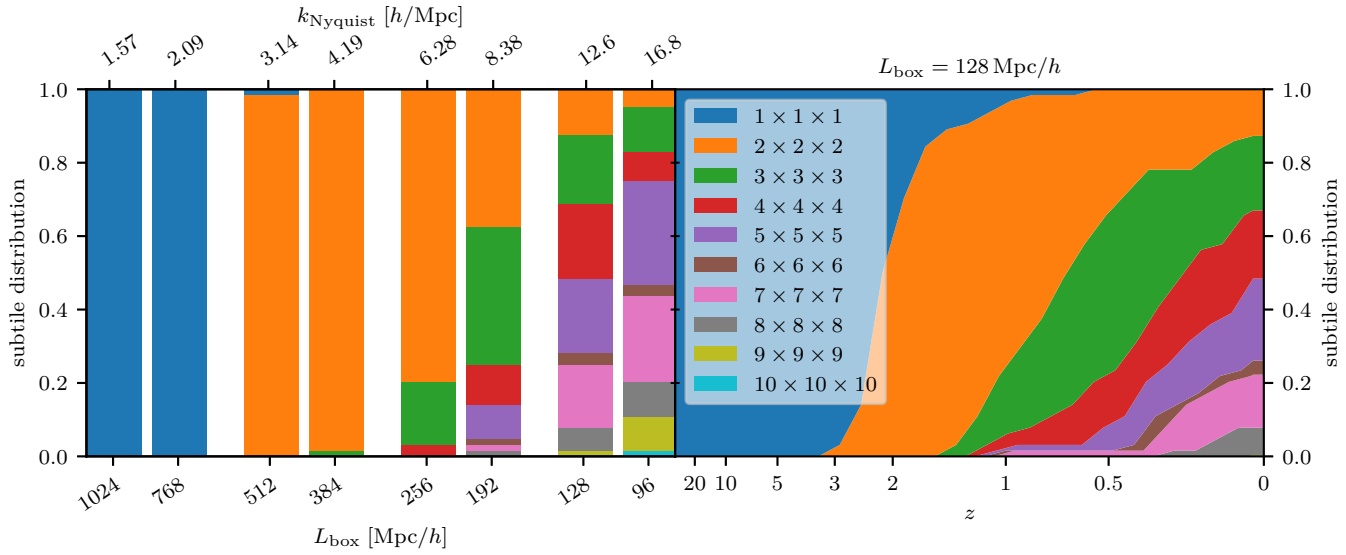

Figure 11. Distribution of subtile decompositions across the domains for simulations with $N = 512^3$ particles run on 64 processes.

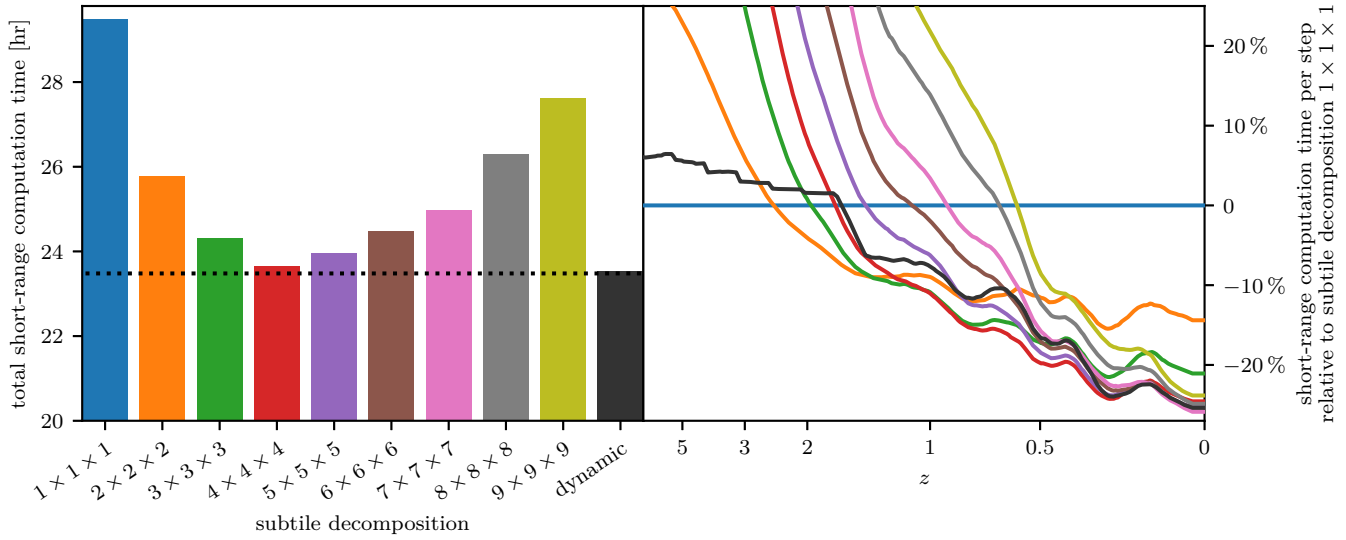

Figure 12. Computation times for the short-range force in simulations with $N = 512^3$ particles in a box of size $L_{\text{box}} = 192\,\text{Mpc}/h$, run on 64 processes.

Monaghan J. J., Lattanzio J. C., 1985, Astronomy and astrophysics, 149, 135
Particle Data Group 2020, Progress of Theoretical and Experimental Physics, 2020, 083C01
Peebles P. J. E., 1980, The large-scale structure of the universe
Plummer H. C., 1911, Monthly notices of the royal astronomical society, 71, 460
Quinn T., Katz N., Stadel J., Lake G., 1997, arXiv preprint astro-ph/9710043
Sefusatti E., Crocce M., Scoccimarro R., Couchman H. M. P., 2016, Monthly Notices of the Royal Astronomical Society, 460, 3624
Springel V., 2005a, Max-Plank-Institute for Astrophysics, Garching, Germany
Springel V., 2005b, Mon. Not. Roy. Astron. Soc., 364, 1105
Springel V., Pakmor R., Zier O., Reinecke M., 2020, arXiv preprint arXiv:2010.03567
Tram T., Brandbyge J., Dakin J., Hannestad S., 2019, Journal of Cosmology and Astroparticle Physics, 2019, 022
de Leeuw S. W., Perram J. W., Smith E. R., 1980, Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences, 373, 27

APPENDIX A: OTHER CODE ASPECTS

The purpose of this section is to provide brief overviews of secondary aspects of CONCEPT 1.0, specifically the many built-in features besides particle dynamics and the unusual software framework in which CONCEPT 1.0 is written.

A1 Additional features

Though this paper focuses on the core N-body functionality of $\mathrm{P^3M}$ and adaptive time-stepping, CONCEPT 1.0 in fact contains a lot of additional features. Here some of these are briefly listed.

Multiple possible non-linear components using either a particle or fluid representation; cold dark matter, decaying cold dark matter (Dakin et al. 2019b), massive neutrinos (Dakin et al. 2019a). Various parameters are tunable at a per component basis.

Linear components allowing for simulations consistent with general relativistic perturbation theory; photons, massive/massless neutrinos (Tram et al. 2019), dynamical dark energy (Dakin et al. 2019c), dark radiation (Dakin et al. 2019b).

On-the-fly initial condition generation of all implemented species, either using standard Gaussian noise or the 'paired and fixed' technique of Angulo & Pontzen (2016).

Complete integration of CLASS (Blas et al. 2011), used to obtain background values such as $a(t)$ and linear perturbations for initial conditions and linear components. All perturbations are transformed to N-body gauge so that simulation results can be interpreted in a relativistic setting (Tram et al. 2019).

Output: Power spectra (data, image), 2D renders (data, image, terminal visualisation), 3D renders (image), snapshots. CONCEPT 1.0 implements its own snapshot format capable of storing particle and fluid components, as well as the full specification of the well-known GADGET format (Springel 2005b).

Grids used for $\mathrm{P^{(3)}M}$ (36), power spectra and 2D renders may use any of the implemented interpolations (25)–(28) with optional deconvolution as well as optional interlacing[22] (Hockney & Eastwood 1988). The grid size of each component is independent, with collective grids computed by adding up (properly shifted) Fourier values, used when e.g. several components contribute to the PM grid or when computing combined auto-spectra of multiple components.

Various auxiliary *utilities* are included alongside the main code, which provide functionality outside of running simulations, such as computing power spectra directly from snapshots.

All user interaction happens through a script with discoverable command-line options, which handles building (on modification) of the code and job execution or even submission via Slurm/TORQUE/PBS.

Complete and flexible installation script for automated installation of CONCEPT 1.0 — along with all of its dependencies — with no special permissions required. Successfully tested on dozens of Linux clusters and laptops.

Docker images of CONCEPT 1.0 are freely available on Docker Hub, convenient for quickly trying out the code.

Large suite of integration tests for continuous code validation. As the installation depends on online resources, the installation along with the entire test suite is automatically tested periodically on GitHub, with the latest result publicly visible.

Thorough documentation — including an expansive tutorial — of how to use the code is publicly released together with the source.

A2 Code language and build process

Though no knowledge of the internals of CONCEPT is needed in order to make use the code, we here want to give a brief overview as the technology employed is rather unique.

Today most scientific code gets written using higher-level languages, probably mainly due to the rapid development these languages and their ecosystems allow for. These languages are typically dynamical and interpreted, which comes at a performance penalty. High-performance simulation codes are thus still primarily written in low-level languages such as Fortran, C and C++. While allowing for performant code where needed, this also forces the lower level aspects upon the rest of the code base, with no performance benefits. This generally makes the code harder to read and extend, especially for the many scientists not fluent in such languages.

The most prevalent high-level language used for scientific computing in the current era is arguably Python, which is also the language chosen for CONCEPT. While performance to some extent is obtainable through the use of libraries such as NumPy (Harris et al. 2020) and FFTW (Frigo & Johnson 2005), this is not enough to compete with high-performance low-level codes such as GADGET. To this end CONCEPT makes heavy use of Cython (Behnel et al. 2011), which translates Python code to equivalent C code, which must then be compiled as any other C program. By further specifying the types of key variables, the translated result can be made as good as hand-written C.

While Cython does allow for seamless mixing of dynamic Python code and typed "C-like" Python code, some of its low-level features (e.g. access to raw pointers) require syntax which breaks Python compatibility (meaning the code now *only* runs after transpilation to C). As rapid development and debugging relies heavily on the code being executable as a pure Python script, CONCEPT effectively implements its own language on top of Cython, with new Python-compatible syntax for these missing functionalities.

While the raw CONCEPT source code may then be executed directly in Python, it can alternatively (and preferably) be built by first transpiling it to valid Cython code[23] using a custom built-in transpiler, after which the code is further transpiled to C using the Cython transpiler, and then finally compiled to machine code using a C compiler.

Besides serving as a bridge between Python and low-level Cython, the built-in transpiler further enables quite a few performance enhancements through direct source code transformations. These include early run-time or even compile-time expression evaluation, loop unswitching and iterator inlining. Oftentimes these are optimisations which cannot be applied by the C compiler itself and which are not easily manually expressible in C.

[22] By default deconvolutions are always on, while interlacing is enabled for power spectra but not for $\mathrm{P^{(3)}M}$.

[23] While with standard Cython one has to further write a header file per code file (as in C), we have automated this task as part of the built-in transpiler. Thus the source code consists solely of the bare Python files, with everything else generated from this.

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