

High Performance P³M N-body code: CUBEP3M

Joachim Harnois-Déraps^{1,2} [★], Ue-Li Pen¹, Ilian T. Iliev³, Hugh Merz⁴,
 JD Emberson^{1,5} and Vincent Desjacques^{6,7}

¹Canadian Institute for Theoretical Astrophysics, University of Toronto, M5S 3H8, Ontario, Canada

²Department of Physics, University of Toronto, M5S 1A7, Ontario, Canada

³Astronomy Centre, Department of Physics and Astronomy, Pevenssey II Building, University of Sussex, BN1 9QH, Brighton, United Kingdom

⁴SHARCNET, Laurentian University, P3E 2C6, Ontario, Canada

⁵Department of Astronomy and Astrophysics, University of Toronto, M5S 3H4, Ontario, Canada

⁶Institute for Theoretical Physics, University of Zürich, Zürich, CH 8057, Switzerland

⁷Université de Genève and Center for Astroparticle Physics, 24 Quai Ernest Ansermet, 1211 Genève 4, Switzerland

29 June 2012

ABSTRACT

This paper presents CUBEP3M, an upgraded version of PMFAST that is now one of the fastest highest performance public N-body code. Gravity is solved on a two-level mesh with sub-grid precision due to the particle-particle (pp) interactions inside and across neighboring cells. Among important improvements, CUBEP3M has a volumetric decomposition, allowing for a massive parallelization over more than 20,000 cores, and has achieved close to ideal weak scaling even at this scale. In addition, force kernels have been improved for enhanced matching of the two mesh contributions near the cutoff length, and the code is now equipped with a unique particle identification tag system that allows reconstruction of individual particle trajectories between time steps. In parallel, many utilities and extensions have been added to the standard code package, including a runtime halo finder, a tuneable range for the exact pp force, a non-Gaussian initial conditions generator, etc. We discuss the structure, the accuracy, any known systematic effects, and the scaling performance of the code and its utilities, when applicable (**We should include a density picture somewhere. Ideally a RANGER4000, for which we show the power spectrum and the halo mass function.**).

Key words: N-body simulations — Large scale structure of Universe — Dark matter

1 INTRODUCTION

Many physical and astrophysical systems are subject to non-linear dynamics and rely on N-body simulations to describe the evolution of bodies. One of the main field of application is the modelling of large scale structures, which are driven by the sole force of gravity. Recent observations of the cosmic microwave background (Komatsu et al. 2009, 2011), of galaxy clustering (York et al. 2000; Colless et al. 2003; Schlegel et al. 2009; Drinkwater et al. 2010) of weak gravitational lensing (Heymans & CFHTLenS Collaboration 2012; Sheldon et al. 2009) and of supernovae redshift-distance relations all point towards a standard model of cosmology, in which dark energy and collision-less dark matter occupy more than 95 per cent of the total energy density of the universe. In such a paradigm, pure N-body code are perfectly suited to describe the dynamics, as long as baryonic physics is not very important, or at least we understand how the baryonic fluid feeds back on the dark matter structure. The next generation of measurements aim at constraining the cosmological parameters at the per cent level, and the theoret-

ical understanding of the non-linear dynamics that govern structure formation heavily relies on numerical simulations.

For instance, a measurement of the baryonic acoustic oscillation (BAO) dilation scale can provide tight constraints on the dark energy equation of state (Eisenstein et al. 2005; Tegmark et al. 2006; Percival et al. 2007; Schlegel et al. 2009). The most optimal estimates of the uncertainty requires the knowledge of the matter power spectrum covariance matrix, which is only accurate when measured from a large sample of N-body simulations (Rimes & Hamilton 2005; Takahashi et al. 2009, 2011). For the same reasons, the most accurate estimates of weak gravitational lensing signal is obtained by propagating photons in past light cones that are extracted from simulated density fields (Vale & White 2003; Sato et al. 2009; Hilbert et al. 2009). Another area where large-scale N-body simulations have in recent years been instrumental are in simulations of early cosmic structures and reionization (e.g. Iliev et al. 2006; Zahn et al. 2007; Trac & Cen 2007; Iliev et al. 2011). The reionization process is primarily driven by low-mass galaxies, which for both observational and theoretical reasons, need to be resolved in fairly large volumes, which demands simulations with a very large dynamic range.

[★] E-mail: jharno@cita.utoronto.ca

The basic problem that is addressed with N-body codes is the time evolution of an ensemble of N particles that is subject to gravitational attraction. The brute force calculation requires $O(N^2)$ operations, a cost that exceeds the memory and speed of current machines for large problems. Solving the problem on a mesh (Hockney & Eastwood 1981) reduces to $O(N \log N)$ the number of operations, as it is possible to solve for the particle-mesh (PM) interaction with fast Fourier transforms techniques with high performance libraries such as FFTW (Frigo & Johnson 2005).

With the advent of large computing facilities, parallel computations have now become common practice, and N-body codes have evolved both in performance and complexity. Many have opted for ‘tree’ algorithms, including GADGET (Springel et al. 2001; Springel 2005), PMT (Xu 1995), GOTPM (Dubinski et al. 2004), and Hydra (Couchman et al. 1995), in which the local resolution increases with the density of the matter field. These often have the advantage to balance the work load across the computing units, which enable fast calculations even in high density regions. The drawback is a significant loss in speed, which can be only partly recovered by turning off the tree algorithm. The same reasoning applies to mesh-refined codes (Couchman 1991), which in the end are not designed to perform fast PM calculations on large scales.

PMFAST (Merz et al. (2005), MPT hereafter) is one of the first code designed specifically to optimize the PM algorithm, both in terms of speed and memory usage. It uses a two-level mesh algorithm based on the gravity solver of Trac & Pen (2003). The long range gravitational force is computed on a grid four times coarser, such as to minimize the MPI communication time and to fit in system’s memory. The short range is computed locally on a finer mesh, and only the local sub-volume needs to be stored at a given time, allowing for OPENMP parallel computation. This this setup enable the code to evolve large cosmological systems both rapidly and accurately, on relatively modest clusters. One of the main advantage of PMFAST over other PM codes is that the number of large arrays is minimized, and the global MPI communications are cut down to the minimum: for passing particle at the beginning of each time step, and for computing the long range FFTs. As described in MPT, access to particles is accelerated with the use of linked lists, deletion of ‘ghost’ particles in buffer zones is done at the same time as particles are passed to adjacent nodes, and the global FFTs are performed with a slab decomposition of the volumes via a special file transfer interface, designed specifically to preserve a high processor load.

Since its first published version, PMFAST has evolved in many aspects. The first major improvement was to transform the volume decomposition in multi-node configurations from slabs to cubes. The problem with slabs is that they do not scale well to large runs: as the number of cells per dimension increases, the thickness of each slab shrinks rapidly, until it reaches the hard limit of a single cell layer. With this enhancement, the code name was changed to CUBEPM. Soon after, it incorporated particle-particle (pp) interactions at the sub-grid level, and was finally renamed CUBEP3M. The public package now includes a significant number of new features: the pp force can be extended to an arbitrary range, the size of the redshift jumps can be constrained for improved accuracy during the first time steps, a runtime halo finder has been implemented, the expansion has also been generalized to include a redshift dependent equation of state of dark energy, there is a system of unique particle identification that can be switched on or off, and the initial condition generator has been generalized as to include non-Gaussian features. PMFAST was equipped with a multi-time stepping option that has not been tested on CUBEP3M yet, but which is, in principle

at least, still available. It also contains support for gas cosmological evolution through a TVD MHD module, and a coupling interface with the radiative transfer code C2-Ray (Mellema et al. 2006).

CUBEP3M is one of the most competitive public N-body code and has been involved in a number of scientific applications over the last few years, spanning the field of weak lensing (Vafaei et al. 2010; Lu & Pen 2008; Doré et al. 2009; Lu et al. 2010; Yu et al. 2010; Harnois-Déraps et al. 2012), BAO (Zhang et al. 2010; Ngan et al. 2011; Harnois-Déraps & Pen 2011; Harnois-Déraps et al. 2012), formation of early cosmic structures (Iliev et al. 2008, 2010), observations of dark stars (Zackrisson et al. 2010; Ilie et al. 2012) and reionization (Iliev et al. 2011; Fernandez et al. 2011; Friedrich et al. 2011; Mao et al. 2012; Datta et al. 2011; Friedrich et al. 2012). It is thus important for the community to have access to a paper that describes the methodology, the accuracy and the performance of this public code.

Since CUBEP3M is not new, the existing accuracy and systematic tests were performed by different groups, on different machines, and with different geometric and parallelization configurations. It is not an ideal situation in which to quantify the performance, and each test must be viewed as a separate measurement that quantifies a specific aspect of the code. We have tried to keep to a minimum the number of such different runs, and although the detailed numbers vary with the problem size and the machines, the general trends are rather universal.

Tests on constraints of the redshift step size and on improvements of the force calculations were performed on the Sunnyvale beowulf cluster at the Canadian Institute for Theoretical Astrophysics (CITA). Each node contains 2 Quad Core Intel(R) Xeon(R) E5310 1.60GHz processors, 4GB of RAM, a 40 GB disk and 2 gigE network interfaces. Each of these tests were performed on the same cluster, but with different box sizes, starting redshifts and particle numbers. Hereafter we refer to these simulation sets as the CITA configurations, and specify which parameters were used in each case.

For tests of the code accuracy, of the non-Gaussian initial conditions generator and of the run time halo finder algorithm, we used a third simulation configuration series that was run at the Texas Advanced Computing Centre (TACC) on Ranger, a Sun-Blade x6420 system with AMD x86_64 Opteron Quad Core 2.3 GHz ‘Barcelona’ processors and Infiniband networking. These RANGER4000 simulations evolved 4000^3 particles from $z = 100$ to $z = 0$ with a box side of $3.2h^{-1}\text{Gpc}$ on 4000 cores.

The paper is structured as follow: section 2 reviews the structure and flow of the main code; section 3 describes how Poisson equation is solved on the two-mesh system; we then present in section 4 the scaling of the code to very large problems, including much larger runs that were produced at various high performance computing centers; section 5 discuss the accuracy and systematic effects of the code. We then describe in section 6 the run-time halo finder, in section 7 various extensions to the default configuration, and conclude afterwards.

2 REVIEW OF THE CODE STRUCTURE

An optimal large scale N-body code must address many challenges: minimize the memory footprint to allow larger dynamical range, minimize the passing of information across computing nodes, reduce and accelerate the memory accesses to the large scale arrays, make efficient use of high performance libraries to speed up standard calculations like Fourier transforms, just to name a few. In

the realm of parallel programming, high efficiency can be assessed when a high load is balanced across all processors most of the time. In this section, we present the general strategies adopted to address these challenges¹. We start with a walkthrough the code flow, and briefly discuss some specific sections that depart from standard N-body codes, while referring the reader to future sections for detailed discussions on selected topics.

As mentioned in the Introduction section, CUBEP3M is a FORTRAN90 N-body code that solves Poisson's equation on a two-level mesh, with sub-cell accuracy thanks to particle-particle interactions. The code has extensions that departs from this basic scheme, and we shall come back to these later, but for the moment, we adopt the standard configuration. The long range component of the gravity force is solved on the coarse grid, and is global in the sense that the calculations require knowledge about the full simulated volume. The short range force and the particle-particle interactions are computed in parallel on a second level of cubical decomposition of the local volumes, the *tiles*. To make this possible, the fine grid arrays are constructed such as to support parallel memory access. In practice, this is done by adding an additional dimension to the relevant arrays, such that each CPU accesses a unique memory location. The force matching between the two meshes is performed by introducing a cutoff length, r_c , in the definition of the two force kernels. The value of $r_c = 16$ fine cells was found to balance the communication overhead between processes and the accuracy of the match between the two meshes.

The computation of the short range force requires each tile to store the fine grid density of a region that includes a buffer surface around the physical volume it is assigned. The thickness of this surface must be larger than r_c , and we find that a 24 cells deep buffer is a good compromise between memory usage and accuracy. This is basically to fully compensate for the coarse mesh calculations, whose CIC interpolation scheme reaches two coarse cells deep beyond the cutoff.

When it comes to finding haloes at run time, this buffer can create a problem, because a large object located close to the boundary can have a radius larger than the buffer zone, in which case it would be truncated and be assigned a wrong mass, center of mass, etc. It could then be desirable to increase the buffer zone around each tile, at the cost of a loss of memory dedicated to the actual physical volume, and the code is designed to allow for such a flexibility.

Because the coarse grid arrays require 4^3 times less memory per node, it does not contribute much to the total memory requirement, and the bulk of the foot-print is concentrated in a handful of fine grid arrays. Some of these are only required for intermediate steps of the calculations, hence it is possible to hide therein many coarse grid arrays². We present here the largest arrays used by the code:

- (i) `xv` stores the position and velocity of each particle
- (ii) `ll` stores the linked-list that accelerate the access to particles in each coarse grid cell
- (iii) `rho_f` and `cmplx_rho_f` store the local fine grid density in real and Fourier space respectively
- (iv) `force_f` stores the force of gravity (short range only) along the three Cartesian directions

¹ Many originate directly from MPT and were preserved in CUBEP3M; those will be briefly mentioned, and we shall refer the reader to the original PMFAST paper for greater details.

² This memory recycling is done with ‘equivalence’ statements in F90

```
program cubep3m
  call initialize
  do
    call timestep
    call particle_mesh
    if(checkpoint_step) then
      call checkpoint
    elseif(last_step)
      exit
    endif
  enddo
  call finalize
end program cubep3m
```

Figure 1. Overall structure of the code simplified for readability.

(v) `kern_f` stores the fine grid force kernel in the three directions

(vi) `PID` stores the unique particle identification tags.

The particle ID is a feature that can be switched off by removing a compilation flag, and allows to optimize the code for higher resolution configurations.

The code flow is presented in Fig. 1 and 2. Before entering the main loop, the code starts with an initialization stage, in which many declared variables are assigned default values, the redshift checkpoints are read, the FFTW plans are created, and the MPI communicators are defined. The phase-space array is obtained from the output of the initial conditions generator, and the force kernels on both grids are constructed from the specific geometry of the simulation. For clarity, all these operations are collected under the subroutine `call initialize` in Fig. 1, although they are actually distinct calls in the code.

Each iteration of the main loop starts with the `timestep` subroutine, which proceeds to a determination of the redshift jump by comparing the step size constraints from each force components and from the scale factor. The cosmic expansion is found by Taylor expanding Friedmann's equation up to the third order in the scale factor, and can accommodate constant or running equation of state of dark energy. The force of gravity is then solved in the `particle_mesh` subroutine, which first updates the positions and velocities of the dark matter particles, exchange with neighbouring nodes those that have exited to volume, creates a new linked list, then solve Poisson's equation. This subroutine is conceptually identical to that of PMFAST, with the exception that CUBEP3M decomposes the volume into cubes (as opposed to slabs). The loop over tiles and the particle exchange are thus performed in three dimensions. When the short range and pp forces have been calculated on all tiles, the code exits the parallel OPENMP loop and proceeds to the long range. This section of the code is also parallelized on many occasions, but, unfortunately, the current MPI-FFTW do not allow multi-threading. There is thus an inevitable loss of efficiency during each global Fourier transforms, during which only the single core MPI process is active³. As in PMFAST, the particle position and velocity updates are performed in a leap frog scheme (Hockney & Eastwood 1981).

If the current redshift corresponds to one of the checkpoints, the code advances all particles to their final location and writes

³ Other libraries such as P3DFFT (<http://code.google.com/p/p3dfft/>) currently permit an extra level of parallelization, and it is our plan to migrate to one of these in the near future.

```

subroutine particle_mesh
  call update_position + apply random offset
  call link_list
  call particle_pass
  !$omp parallel do
  do tile = 1, tiles_node
    call rho_f_ngp
    call cmplx_rho_f
    call kernel_multiply_f
    call force_f
    call update_velocity_f
    if(pp = .true.) then
      call link_list_pp
      call force_pp
      call update_velocity_pp
      if(extended_pp = .true.) then
        call link_list_pp_extended
        call force_pp_extended
        call update_velocity_pp_extended
      endif
    endiff
  end do
  !$omp end parallel do
  call rho_c_ngp
  call cmplx_rho_c
  call kernel_multiply_c
  call force_c
  call update_velocity_c
  delete_buffers
end subroutine particle_mesh

```

Figure 2. Overall structure of the two-level mesh algorithm. We have included the section that concerns the standard pp and the extended pp force calculation, to illustrate that they follows similar linked-list logic.

them to file. Similarly, the code can compute two-dimensional projections of the density field, halo catalogues (see section 6 for details), and can compute the power spectrum on the coarse grid at run time. The code exits the loop when it has reached the final redshift, it then wraps up the FFTW plans and clears the MPI communicators. We have collected those operations under the subroutine `finalize` for concision.

Other considerations that need to be considered is that any decomposition geometry somehow limits permissible grid sizes, and that volume evenly decomposed into cubic sub-sections – such as CUBEP3M – requires the number of MPI processes to be a perfect cube. Also, since the decomposition is volumetric, as opposed to density dependent – it suffers from load imbalance when probing deep into the non-linear regime.

3 POISSON SOLVER

This section reviews how Poisson’s equation is solved on a double-mesh configuration. Many parts of the algorithm are identical to PMFAST, hence we refer the reader to section 2 of MPT for more details. In CUBEP3M, the mass default assignment scheme are a ‘cloud-in-cell’ (CIC) interpolation for the coarse grid, and a ‘nearest-grid-point’ (NGP) interpolation for the fine grid (Hockney & Eastwood 1981). This choice is motivated by the fact that the most straightforward way to implement a P³M algorithm on a mesh is to have exactly zero mesh force inside a grid, which is only true for the NGP interpolation. Although CIC generally has a smoother and more accurate force, the pp implementation enhances the code resolution by almost an order of magnitude.

The code units inherit from (Trac & Pen 2004) and are summarized here for completeness. The comoving length of a fine grid cell is set to one, such that the unit length in simulation unit is

$$1\mathcal{L} = a \frac{L}{N} \quad (1)$$

where a is the scale factor, N is the total number of cells along one dimension, and L is the comoving volume in $h^{-1}\text{Mpc}$. The mean comoving mass density is also set to unity in simulation units, which, in physical units, corresponds to

$$1\mathcal{D} = \rho_m(0)a^{-3} = \Omega_m\rho_c a^{-3} = \frac{3\Omega_m H_o^2}{8\pi G a^3} \quad (2)$$

Ω_m is the matter density, H_o is the Hubble’s constant, ρ_c is the critical density today, and G is Newton’s constant. The mass unit is found with $\mathcal{M} = \mathcal{D}\mathcal{L}^3$. Specifying the value of G on the grid fixes the time unit, and with $G_{grid} = 1/(6\pi a)$, we get:

$$1\mathcal{T} = \frac{2a^2}{3} \frac{1}{\sqrt{\Omega_m H_o^2}} \quad (3)$$

These choices completely determine the conversion between physical and simulation units. For instance, the velocity units are given by $1\mathcal{V} = \mathcal{L}/\mathcal{T}$.

The force of gravity on a mesh can be computed either with a gravitational potential kernel $\omega_\phi(\mathbf{x})$ or a force kernel $\omega_F(\mathbf{x})$. Gravity fields are curl-free, which allows us to relate the potential $\phi(\mathbf{x})$ to the source term via Poisson’s equation:

$$\nabla^2 \phi(\mathbf{x}) = 4\pi G \rho(\mathbf{x}) \quad (4)$$

We solve this equation in Fourier space, where we write:

$$\tilde{\phi}(\mathbf{k}) = \frac{-4\pi G \tilde{\rho}(\mathbf{k})}{k^2} \equiv \tilde{\omega}_\phi(\mathbf{k}) \tilde{\rho}(\mathbf{k}) \quad (5)$$

The potential in real space is then obtained with an inverse Fourier transform, and the kernel becomes $\omega_\phi(\mathbf{x}) = -G/r$. Using the convolution theorem, we can write

$$\phi(\mathbf{x}) = \int \rho(\mathbf{x}') \omega_\phi(\mathbf{x}' - \mathbf{x}) d\mathbf{x}' \quad (6)$$

and

$$\mathbf{F}(\mathbf{x}) = -m \nabla \phi(\mathbf{x}) \quad (7)$$

Although this approach is fast, it involves a finite differentiation at the final step, which enhances the numerical noise. We therefore opt for a force kernel, which is more accurate, even though it requires four extra Fourier transforms. In this case, we must solve the convolution in three dimensions and define the force kernel ω_F such as:

$$\mathbf{F}(\mathbf{x}) = \int \rho(\mathbf{x}') \omega_F(\mathbf{x}' - \mathbf{x}) d\mathbf{x}' \quad (8)$$

Because the gradient acting on \mathbf{F} affects only un-prime variables, we can express the force kernel as a gradient of the potential kernel. Namely:

$$\omega_F(\mathbf{x}) \equiv -\nabla \omega_\phi(\mathbf{x}) = -\frac{mG\hat{\mathbf{r}}}{r^2} \quad (9)$$

Following the spherically symmetric matching technique of MPT (section 2.1), we split the force kernel into two components, for the short and long range respectively, and match the overlapping region with a polynomial. Namely, we have:

$$\omega_s(r) = \begin{cases} \omega_F(r) - \beta(r) & \text{if } r \leq r_c \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

and

$$\omega_l(r) = \begin{cases} \beta(r) & \text{if } r \leq r_c \\ \omega_s(r) & \text{otherwise} \end{cases} \quad (11)$$

The vector $\beta(r)$ is related to the fourth order polynomial that is used in the potential case described in MPT by $\beta = -\nabla\alpha(r)$. The coefficients are found by matching the boundary conditions at r_c up to the second derivative, and we get

$$\beta(r) = \left[-\frac{7r}{4r_c^3} + \frac{3r^3}{4r^5} \right] \hat{\mathbf{r}} \quad (12)$$

Since these calculations are performed on two grids of different resolution, a sampling window function must be convoluted both with the density and the kernel (see [Eq. 7-8] of MPT). When matching the two force kernels, the long range force is always on the low side close to the cutoff region, whereas the short range force is uniformly scattered across the theoretical $1/r^2$ value – intrinsic features of the CIC and NGP interpolation schemes respectively. By performing force measurements on two particles randomly placed in the volume, we identified a small region surrounding the cutoff length in which we empirically adjust both kernels such as to improve the match. Namely, for $14 \leq r \leq 16$, $\omega_s(r) \rightarrow 0.985\omega_s(r)$, and for $12 \leq r \leq 16$, $\omega_l(r) \rightarrow 1.2\omega_l(r)$.

As mentioned in section 2 and summarized in Fig. 2, the force kernels are first in the code initialization stage. Eq. 8 is then solved with fast Fourier transform along each directions, and is applied onto particles in the update_velocity subroutine.

The pp force is calculated during the fine mesh velocity update, which avoids loading the particle list twice and allows the operation to be threaded without significant additional work. During this process, the particles within a given fine mesh tile are first read in via the linked list, then their velocity is updated with the fine mesh force component, according to their location within the tile. In order to organize the particle-particle interactions, we proceed in constructing a set of threaded fine-cell linked list chains for each coarse cell. We then calculate the pairwise force between all particles that lie within the same fine mesh cell, excluding pairs whose separation is smaller than a softening length of one tenth of a grid cell; particles separated by less than this distance have their particle-particle force set to zero. As this proceeds, we accumulate the pp force applied on each particle and then determine the maximum force element of the pp contribution, which is also taken into account when constraining the length of the global timestep.

This kind of force softening is generally required by any code to prevent large scattering as $r \rightarrow 0$, which can otherwise slow the calculation down, and to reduce the two-body relaxation, which can affect the numerical convergence. Many force softening scheme can be found in the litterature, including Plummer force, uniform or linear density profiles or the spline-softened model (see ? for a review). In this case, a sharp force cutoff corresponds to a particle density profile that resembles that of a hollow shell. In comparison with other techniques, this force softening is the easiest to code, the fastest to execute, but has a discontinuous profile at the cutoff lenght, which certainly has an impact on the smallest resolved structure. **(Have we tried anything else? Why did we choose this one? How does this cutoff translates in terms of energy and momentum conservation? Unstable?)**

In addition, PMFAST could run with a different set of force kernels, described in MPT as the ‘least square matching’, which basically adjust the kernels on cell-by-cell basis based on minimization of the deviation with respect to the Newtonian predictions. This was originally computed such as to optimize the force calculation

for the case where both grids are obtained from CIC interpolation. Moving to a mix CIC/NGP scheme requires solving the system of equations with the new configuration, a straightforward operation. With the inclusion of the random shifting, it is not clear how much improvement one would recover from this other kernel matching. It is certainly something we will investigate and document in the near future.

Finally, a choice must be done concerning the longest range of the coarse mesh force. Gravity can be either a) an accurate $1/r^2$ force, as far as the volume allows, or b) modified to correctly match the periodicity of the boundary conditions. By default, the code is configured along to the second choice, which accurately models the growth of structures at very large scales. However, detailed studies of gravitational collapse of a single large object would benefit from the first settings, even though the code is not optimal for such settings.

4 SCALING PERFORMANCES

The parallel algorithm of CUBEP3M is designed for ‘weak’ scaling, i.e. if the number of cores and the problem size increase in proportion to each other, then for ideal scaling the wall-clock time should remain the same. This is to be in contrasted with ‘strong’ scaling codes, whereby the same problem solved on more cores should take proportionately less wall-clock time. This weak scaling requirement is dictated by the problems we are typically investigating (very large and computationally-intensive) and our goals, which are to address such large problems in the most efficient way, rather than for the least wall-clock time. Furthermore, we recall that there is no explicit load balancing feature, thus the code is maximally efficient when the sub-domains contain roughly an number of particles. This is true for most cosmological-size volumes that do not resolve too deep in the non-linear regime, but not for e.g. simulations of a single highly-resolved galaxy.

Because of the volumetric decomposition, the total number of MPI processes needs to be a perfect cube. Also, for maximal resource usage, the number of tiles per node should be a multiple of the number of available CPUs per MPI process, such that no core sits idle in the threaded block. Given the available freedom in the parallel configuration, as long as the load is balanced, it is generally good practice to maximize the number of OPENMP threads and minimize the number of MPI processes: the information exchange between cores that are part of the same motherboard is generally much faster. In addition, having fewer MPI processes reduces the total amount of buffer zones, freeing memory that can be used to increase the mesh resolution. As one probes deeper into the non-linear regime however, the formation of dense objects can cause memory problems in such configurations, and increasing the number of MPI processes helps to ensure memory locality, especially in non-uniform memory access (NUMA) environments.

The intermediary version of the code – CUBEPM – was first ported to the IBM Blue Gene/L platform, and achieved weak-scaling up to 4096 processes (over a billion particles), with the N-body calculation only incurring a 10 per cent overhead at runtime (compared to 8 processes) for a balanced workload ⁴. In order to accommodate the limited amount of memory available per processing core on the Blue Gene/L platform machines, it was necessary to perform the long range MPI FFT with a volumetric decomposition

⁴ <http://web.archive.org/web/20060925132146/http://www-03.ibm.com/servers>

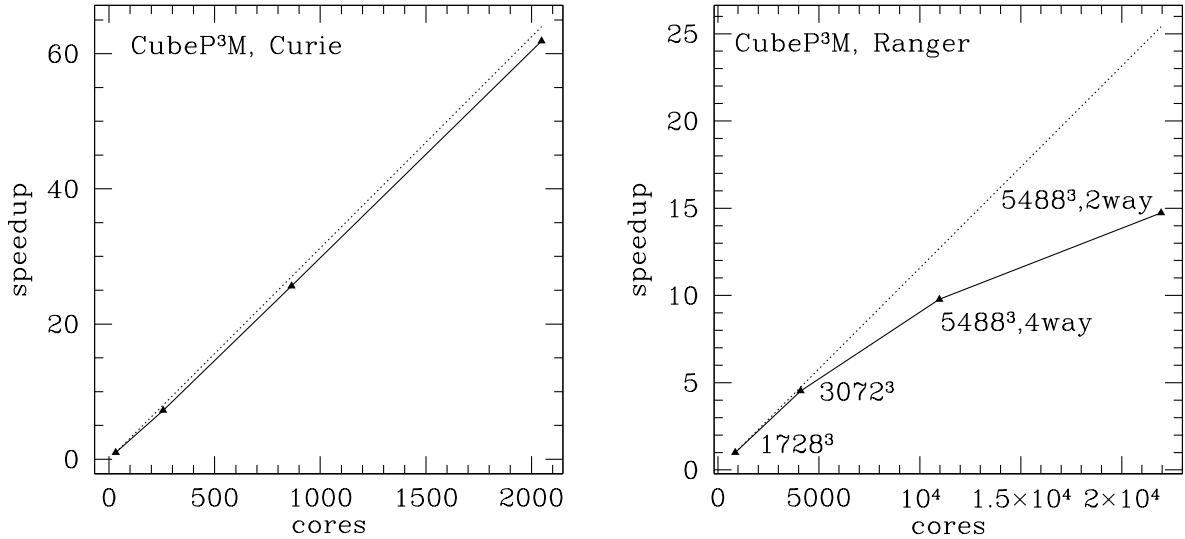


Figure 3. Scaling of CUBEP3M on Curie fat nodes (left) and on Ranger TACC facility for very large number of cores (right). Plotted is the code speedup ($N_{\text{particles}}^3/t_{\text{wallclock}}$) against core count, normalized by the smallest run in each case. Dashed line indicates the ideal weak scaling. The data are listed in Table 1.

Table 1. Scaling of CUBEP3M on Curie. Speedup is scaled to the smallest run.

number of cores	speedup	ideal speedup	absolute timing (min)	$N_{\text{particles}}$	box size ($h^{-1}\text{Mpc}$)
32	1.00	-	3.2	256^3	256
256	7.21	8	3.55	512^3	512
864	25.63	27	4.8	864^3	864
2048	61.87	64	26.48	2048^3	2048

Table 2. Scaling of CUBEP3M on Ranger. Speedup is scaled to the smallest run.

number of cores	speedup	ideal speedup	absolute timing (min)	$N_{\text{particles}}$	box size ($h^{-1}\text{Mpc}$)
864	1.00	-	258	1728^3	6.3
4096	4.53	4.74	320	3072^3	11.4
10976	9.78	12.7	845	5488^3	20
21952	14.73	25.4	561	5488^3	20

(Eleftheriou et al. 2005). Slab decomposition would have required a volume too large to fit in system memory given the constraints in the simulation geometry.

The scaling of CUBEP3M was first tested with a dedicated series of simulations – the CURIE simulation suite– by increasing the size and number of cores on the ‘fat’ (i.e. large-memory) nodes of the Curie supercomputer at the Très Grand Centre de Calcul (TGCC) in France. For appropriate direct comparison, all these simulations were performed using the same particle mass ($M_{\text{particle}} = 1.07 \times 10^{11} M_{\odot}$) and force resolution (softening length $50 h^{-1}\text{kpc}$). The box sizes used range from $256 h^{-1}\text{Mpc}$ to $2048 h^{-1}\text{Mpc}$, and the number of particles from 256^3 to 2048^3 . Simulations were run on 32 up to 2048 computing cores, also starting from redshift $z = 100$, and evolving until $z = 0$. Our results are shown in Fig. 3 and in Table 1, and present excellent scaling, within ~ 3 per cent of ideal, at least for up to 2048 cores.

We have also ran CUBEP3M on a much larger number of cores, from 8000 to up to 21,976, with 5488^3 – 6000^3 (165 to 216 billion) particles on Ranger and on JUROPA at the Jülich Supercomputing

Centre in Germany, which is an Intel Xeon X5570 (Nehalem-EP) quad-core 2.93 GHz system, also interconnected with Infiniband. Since it is not practical to perform dedicated scaling tests on such a large number of computing cores, we instead list in Table 2 the data directly extracted from production runs. We have found the code to scale within 1.5 per cent of ideal up to 4096 cores. For larger sizes ($\geq 10,976$ cores), the scaling is less ideal, due to increased communication costs, I/O overheads (a single timeslice of 5488^3 particles is 3.6 TB) and load balancing issues, but still within ~ 20 per cent of ideal. These first three Ranger runs were performed with 4 MPI processes and 4 threads per Ranger node (‘4way’)⁵.

Furthermore, due to the increasing clustering of structures at

⁵ For these very large runs, we used a NUMA script *tacc_affinity*, specially-provided by the technical staff, that bind the memory usage to local sockets, thus ensuring memory affinity. This becomes important because the memory sockets per node (32 GB RAM/node on Ranger) are actually not equal-access. Generally, the local memory of each processor has much shorter access time.

those small scales, some of the cuboid sub-domains came to contain a number of particles well above the average, thereby requiring more memory per MPI process in order to run until the end. As a consequence, throughout most of their late evolution, the largest two of these simulations were run with 4096 and 21,952 cores and with only 2 MPI processes and 8 threads per Ranger node ('2way'), which on Ranger allows using up to 16 GB of RAM per MPI process⁶. Because each processor accesses memory that is not fully local, this configuration does affect the performance somewhat, as does the imperfect load balancing that arises in such situations. This can be seen in the rightmost point of Fig. 3 (right panel), where the scaling 42 per cent below the ideal. We note that we still get ~ 1.5 speedup from doubling the core count, even given these issues. Overall the code scaling performance is thus satisfactory even at extremely large number of cores. We expect the code to handle even larger problems efficiently, and is thus well suited to run on next generation Petascale systems.

Finally, we note that several special fixes had to be developed by the TACC and JUROPA technical staff in order for our largest runs to work properly. In particular, we encountered unexpected problems from software libraries such as MPICH and FFTW when applied to calculations of such unprecedented size.

(We should also compare the memory usage to other codes like Gadget. Ilian mentioned Gadget uses 90 Bytes per particle. Is cubep3m really using 130 Bytes per particle? Check with mem_usage.)

5 ACCURACY AND SYSTEMATICS

This section describes the systematics effects that are inherent to our P^3M algorithm. We start with a demonstration of the accuracy with a power spectrum measurement of a RANGER4000 simulation. The halo mass function also assess the code capacity to model gravitational collapse, but depends on the particular halo finder used. We thus post-pone the discussion on this aspect until in section 6, and focus for the moment on the particles only. In addition to the power spectrum, we further quantify the accuracy of the force calculation with comparisons to Newton's law of gravitation.

5.1 Power Spectrum

One of the most reliable way to assess the simulation's accuracy at evolving particles is to measure the density power spectrum at late time, and compare to non-linear prediction. For an over-density field $\delta(x)$, the power spectrum is extracted from the two point function in Fourier space as:

$$\langle |\delta(k)\delta(k')| \rangle = (2\pi)^3 P(k)\delta_D(k' - k) \quad (13)$$

where the angle bracket corresponds to an ensemble (or volume) average. We plot in Fig. 4 the dimensionless power spectrum in a RANGER4000 simulation and observe that the agreement with the non-linear prediction of Lewis et al. (2000) is at the per cent level over a large dynamical range. The drop of power at high- k is partly caused by the finite mesh resolution, partly from expected deviations about the predictions. The fluctuations at low- k come from the white noise imposed in the initial conditions. The vertical line corresponds to the scale of the coarse mesh.

⁶ In order to insure local memory affinity, a second special NUMA control script, `tacc_affinity_2way`, was developed for us by the TACC technical staff and allowed to run more efficiently in this mode.

5.2 Mesh force at grid distances

The force test presented in MPT was carried by placing two particles at random locations on the grid, calculate the force between them, then iterate over different locations. This pairwise force test is useful to quantify the accuracy on a cell-by-cell basis, but lacks the statistics that occur in an actual time step calculation. The actual force of gravity in the P^3M algorithm, as felt by a single particle during a single step, is presented in the top left panel of Fig. 5. This force versus distance plot was obtained from a CITA128 realization, and the calculation proceeds in two steps: 1- we compute the force on each particle in a given time step. 2- we remove a selected particle, thus creating a 'hole', compute the force again on all particles, and record on file the force difference (before and after the removal) as a function of the distance to the hole.

Particles in the same fine cell as the hole follow the exact $1/r^2$ curve. The scatter at distances of the order of the fine grid is caused by the NGP interpolation scheme: particles in adjacent fine cells can be actually very close, as seen in the upper left region of this plot, but still feel the same mesh force at grid cell distances. This creates a discrepancy up to an order of magnitude, in loss or gain, depending on the location of the pair with respect to the centre of the cell. As the separation approaches a tenth of the full box size or so, the force on the coarse mesh scatters significantly about Newton's law due to periodic boundary conditions. As mentioned at the end of section 3, the longest range of force kernel can either model accurately Newtonian gravity, or the structure growth of the largest modes, but not both. For the sake of demonstrating the accuracy of the force calculation, we chose the first option in this section.

It is not clear from these scatter plots whether the mean force felt is below or under the predictions, hence we rebin the results in 50 logarithmically spaced bins and compute the mean and standard deviation. We show this plot in the middle left panel of Fig. 5, where we observe that on average, there is a loss of force of the order of unity in the range $0.4 < r < 1$ (in units of fine grid), but the force force is otherwise in excellent agreement with Newton's law.

The top right and middle right panels of Fig. 5 show the fractional error on the force along the radial and tangential directions, also calculated from a single time step. The mean of the radial component is up to 80 per cent away from the exact law at sub-grid distances, but quickly drops under ten per cent for larger distances. The transverse fractional error peaks exactly at the grid distance, and is about two time smaller than the radial counterpart.

At grid scales, the fractional error is larger than PMFAST, largely due to the fact that the fine mesh force is performed with an NGP interpolation scheme – as opposed to CIC. This prescription is responsible for the larger scatter about the theoretical value, but, as mentioned earlier, NGP interpolation is essential to our implementation of the pp part. At the same time, the biggest problem with the straightforward pp force calculation is that the results are anisotropic and depend on the location of the fine mesh with respect to the particles. As an example, consider two particles on either side of a grid cell boundary, experiencing their mutual gravity attraction via the fine mesh force with a discretized one-grid cell separation. If, instead, the mesh was shifted such that they were within the same cell, they would experience the much larger pp force. This is clearly seen in the top left panel of Fig. 5, where particles physically close, but in different grid cells, feel a force up to an order of magnitude too small. This effect is especially pronounced at the early stages of the simulation where the density is more homoge-

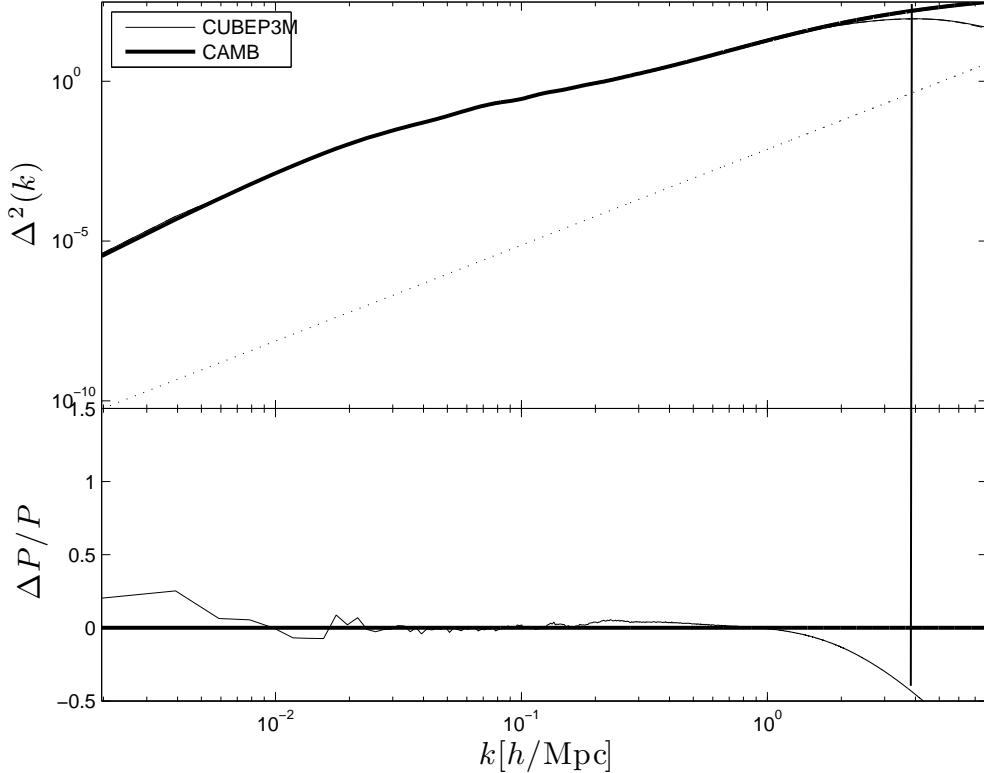


Figure 4. Dark matter power spectrum, measured at $z = 0$ in a RANGER4000 simulation, compared to the non-linear predictions of CAMB. The vertical line corresponds to the scale of the coarse mesh, while the dotted line represents the Poisson noise. **Should I show another redshift instead? Also, the legend box cuts the text: need to fix this. Finally, I could also show linear theory.**

neous, and leads to mesh artefacts appearing in the density field (**Should I show a density plot?**).

In order to minimize these two systematic effect – the large scatter and the anisotropy – we randomly shift the particle distribution relative to the mesh by a small amount – up to 2 fine grid cells in magnitude – in each dimension and at each time step. This adds negligible computational overhead as it is applied during the particle position update, and suppresses the mesh behaviour that otherwise grows over multiple time steps. It is possible to shift back the particles at the end of each time steps, which prevents a random drift of the whole population, a necessary step if one needs to correlate the initial and final positions of the particles for instance, or for hybrid dark matter – MHD simulations.

We ensure that, on average, this solution balances out the mesh feature, by tuning the force kernels such as to provide a force as evenly balanced as possible, both at grid cell distances and at the cutoff length ($r_c = 16$ fine cells). These adjustments are performed from the pairwise force test mentioned above (and described in MPT). The bottom panels of Fig. 5 show the effect of averaging over ten time steps, where the random offset is applied on the particles (and on the hole) at the end of each force calculation. We observe that the agreement extends at the per cent level down to $r \sim 1.5$, beyond which point the discretize effect of the NGP causes an underestimate of the force of up to 60 per cent. (**Can't we beat this by tempering with the fine force kernel? Will the least square fit enhance this?**)

We note that this inevitable loss of force in the NGP scheme

is one of the driving argument to extend the pp force outside the fine mesh cell, since the scattering about the actual $1/r^2$ law drops rapidly as the distance increases. As discussed in section 7.3, this gain in accuracy comes at a computational price, but at least we have the option to run the code with high precision.

We present in Fig. 6 the dramatic impact of removing the random offset in the otherwise default code configuration. This test was performed with a CITA256 simulation of very large box size, the output redshifts are very early (the upper most curve ($z = 10$) was obtained after about 60 time steps), such that the power spectrum should agree with linear theory up to the resolution limit. Instead, we observe that the power spectrum grows completely wrong, due to the large scatter in the force from the fine mesh, and to the anisotropic nature of the P³M calculations mentioned above. When these effects are not averaged over, the errors directly add up at each time step, which explains why later time are worst. We recall that PMFAST did not have this problem since it used CIC interpolation on both meshes. (**Anything else to add here?**)

5.3 Constraining redshift jumps

At early stages of the simulation, the density fields is rather homogenous, causing the force of gravity to be rather weak everywhere. In that case, the size of the redshift jumps is controlled by a limit in the cosmological expansion. If the expansion jump is too large, the size of the residual errors can become significant, and one can observe, for instance, a growth of structure that does not match

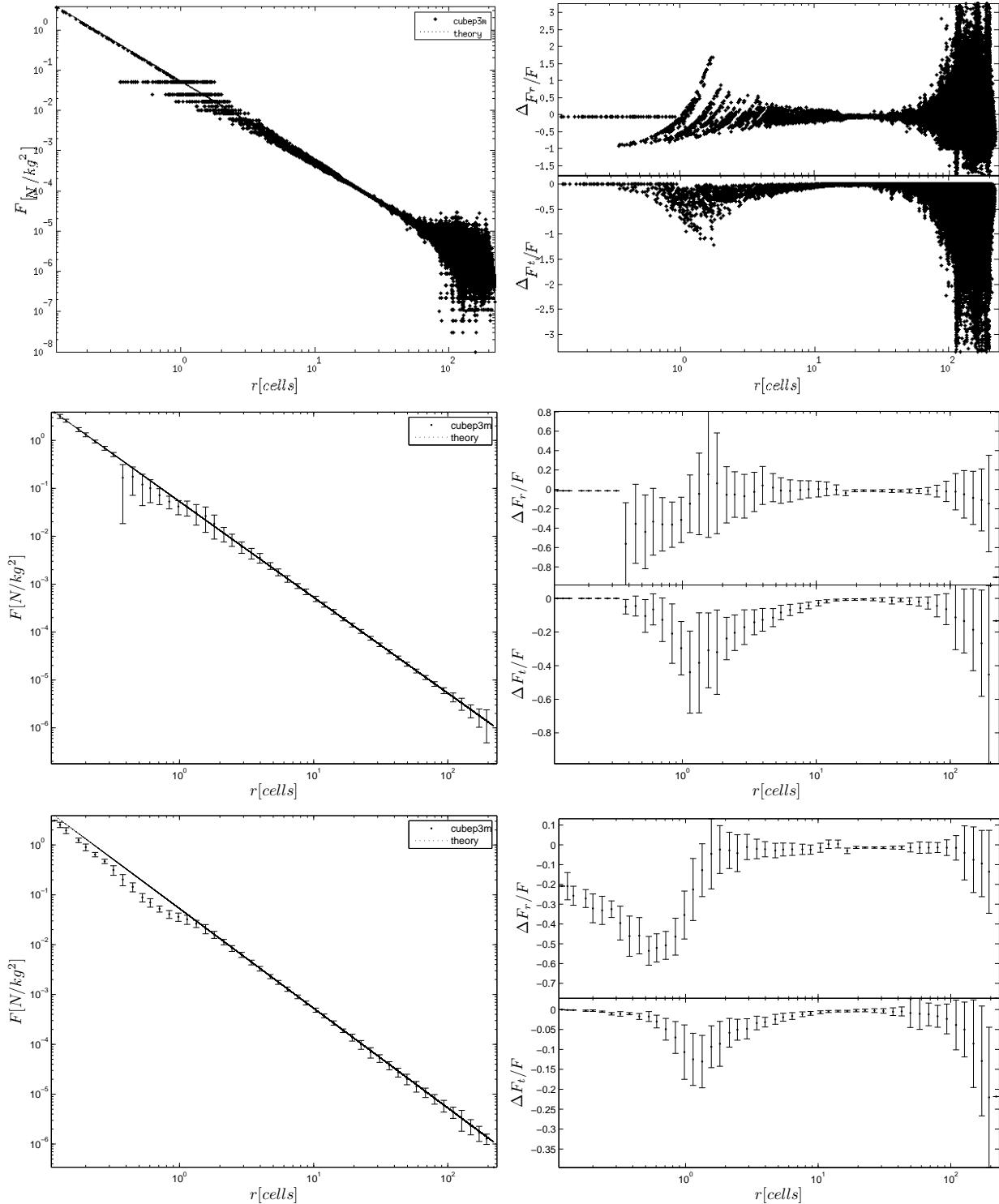


Figure 5. (top left:) Gravity force in of the P^3M algorithm, versus distance in fine mesh cell units, compared with the exact $1/r^2$ law. This particular calculation was obtained in a CITA128 realization with a box size of $100h^{-1}$ Mpc, in a single time step. (top right:) Fractional error on the force in of the P^3M algorithm, in the radial (top) and tangential (bottom) directions. In a full simulation run, the scatter averages over many time steps, thanks to the inclusion of a random offset that is imposed on each particle, as discussed in the main text. (middle row:) Top row organized in 50 logarithmic bins. (bottom row:) Same as middle row, but averaged over ten time steps.

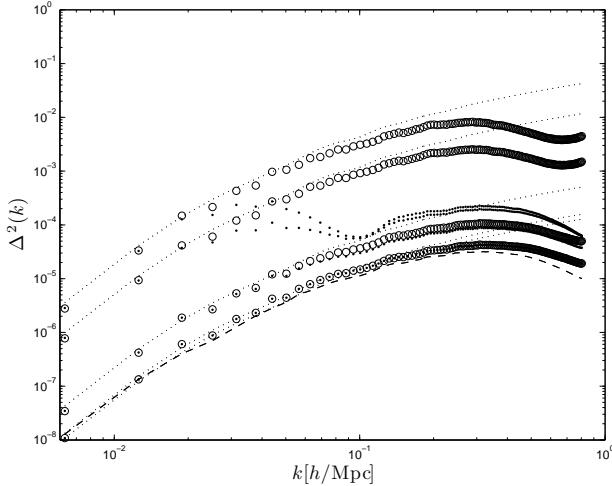


Figure 6. Dark matter power spectrum, measured at $z = 180, 100, 20$ and 10 , in a CITA256 realization that is $1000 \text{ Mpc}/h$ per side. The dashed line represent the initial condition power spectrum, the dotted lines are the linear predictions, and the open circles the standard P³M configuration. The dots were obtained by simply removing the random offset that is normally applied at each time step.

the predictions of linear theory even at the largest scales. One therefore needs to choose a maximum step size. In CUBEP3M, this is controlled by r_{max} , which is the fractional step size, $da/(a+da)$ and is set to 0.05 by default. Generally, a simulation should start at a redshift high enough that the initial dimensionless power spectrum is well under unity at all scales. This ensures that the Zel'dovich approximation holds at the per cent level at least. A drop of accuracy can occur if one starts the simulation too early, where truncation error will be significant at the early time steps.

It is possible to reduce this effect, and thereby improve significantly the accuracy of the code, by modifying the value of r_{max} , at the cost of increasing the total number of time steps. Fig. 7 shows a comparison of late time power spectra of a series of CITA256 realizations that originate from the same initial conditions, and used the same random seeds to control the fine mesh shifts (mentioned above): only the value of r_{max} was modified between each run. We observe that the impact is mostly located in the non-linear regime, where decreasing the time step to 0.006 allows the simulation to recover about 30 per cent of dimensionless power at the turn over scale, in this simulation configuration. This gain is greatly affected by the choice of initial redshift, the resolution, and the box size, and ideally one would make test runs in order to optimize a given configuration. As expected, the CPU resources required to run these simulations increase rapidly as r_{max} decreases, as seen in Table 3. In this test case, reducing further at 0.001 shows only a mild improvement in accuracy, but the increase in time is more than a factor of four. We should mention that with a proper use of the non-Gaussian initial conditions, it is possible to start the simulations at much later redshifts, where the time step sizes are dominated by a stronger mesh force.

(Any other systematics effects we want to discuss here?)

6 RUNTIME HALO FINDER

We have implemented a halo finding procedure, which we have developed based on the spherical overdensity (SO) approach (Lacey

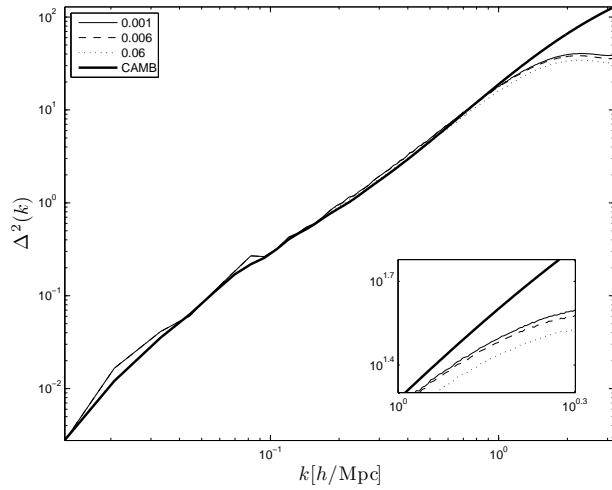


Figure 7. Dark matter power spectrum, measured at $z = 0$ in a series of CITA256 realizations. The starting redshift of was raised to $z = 200$ to enhance the systematic effect. The different curves show different values of r_{max} . The resources required to run these simulations increase rapidly as r_{max} decreases, as seen in Table 3.

Table 3. Scaling in CPU resources as a function of the value of r_{max} . The tests were performed on the CITA Sunnyvale cluster, and general trends could vary slightly on other machines.

r_{max}	time (h)
0.1	1.46
0.06	1.48
0.01	1.67
0.006	1.91
0.003	2.83
0.002	4.13
0.001	8.15

& Cole 1994). In the interest of speed and efficiency the halo catalogues are constructed on-the-fly at a pre-determined list of redshifts. The halo finding is massively-parallel and threaded based on the main CUBEP3M data structures discussed in section 2. The code first builds the fine-mesh density for each sub-domain using CIC or NGP interpolation. It then proceeds to search for all local density maxima above a certain threshold (typically set to a factor of 100 above mean density) within the local tile. It then uses parabolic interpolation on the density field to determine more precisely the location of the maximum within the densest cell, and records the peak position and value.

Once the list of peak positions is generated, they are sorted from the highest to the lowest density value. Then each of the halo candidates is inspected independently, starting with the highest peak. The grid mass is accumulated in spherical shells of fine grid cells surrounding the maximum, until the mean density within the halo drops below a pre-defined overdensity cutoff (usually set to 178 in units of the mean, in accordance with the top-hat collapse model). As we accumulate the mass, we remove it from the mesh, so that no mass element is double-counted. This method is thus inappropriate for finding sub-haloes since within this framework, those are naturally incorporated in their host haloes. Because the haloes are found on a discrete grid, it is possible, especially for those with lower mass, to overshoot the target overdensity. To

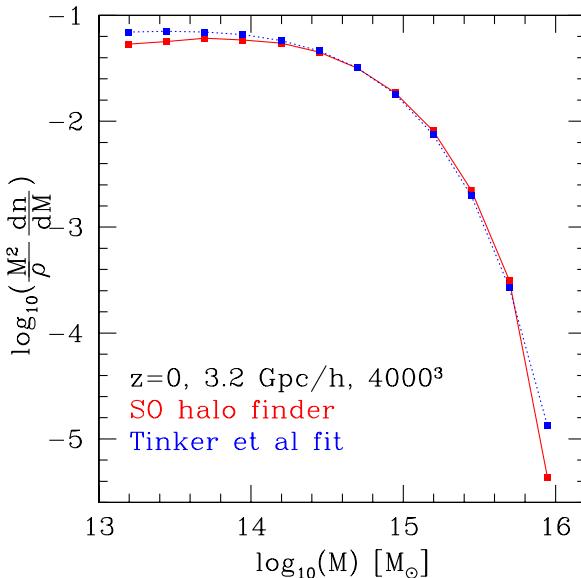


Figure 8. Simulated halo multiplicity function, $\frac{M^2}{\rho} \frac{dn}{dM}$, based on a RANGER4000 simulation with $3.2 h^{-1}$ Gpc box and 4000^3 particles (solid, red in the online version). For reference we also show a widely-used precise fit by Tinker et al. (2008) (blue, dashed).

minimize this effect, we correct the halo mass and radius with an analytical density profile. We use the Truncated Isothermal Sphere (TIS) profile (Shapiro et al. 1999; Iliev & Shapiro 2001) for over-densities below ~ 130 , and a simple $1/r^2$ law for lower over-densities (**lower or higher?**). These yield a similar outer slope to the Navarro, Frenk and White (NFW Navarro et al. 1997) profile, but extends to lower over-densities and matches well the virialization shock position given by the Bertschinger self-similar collapse solution (Bertschinger 1985).

Once the correct halo mass, radius and position are determined, we find all particles that are within the halo radius. Their positions and velocities are used to calculate the halo centre-of-mass, bulk velocity, internal velocity dispersion and the three angular momentum components, all of which are then included in the final halo catalogues. We also calculate the total mass of all particles within the halo radius, also listed in the halo data. This mass is very close, but typically slightly lower, than the halo mass calculated based on the gridded density field. The centre-of-mass found this way closely follows that found from the peak location, which is based on the gridded mass distribution.

A sample halo mass function produced from a RANGER4000 simulation at $z = 0$ is shown in Fig. 8. We compare our result to the precise fit presented recently by Tinker et al. (2008). Unlike most other widely-used fits like the one by Sheth & Tormen (2002), which are based on friends-of-friends (FOF) halo finders, this one is based on the SO search algorithm, whose masses are systematically different from the FOF masses (e.g. Reed et al. 2007; Tinker et al. 2008), making this fit a better base for comparison here. Results show excellent agreement, within ~ 10 per cent for all haloes with masses corresponding to 1000 particles or more. Lower-mass haloes are somewhat under-counted compared to the Tinker et al. (2008) fit, by ~ 20 per cent for 400 particles and by ~ 40 per cent for 50 particles. This is due to the grid-based nature of our SO halo finder, which misses some of the low-mass haloes. It was shown,

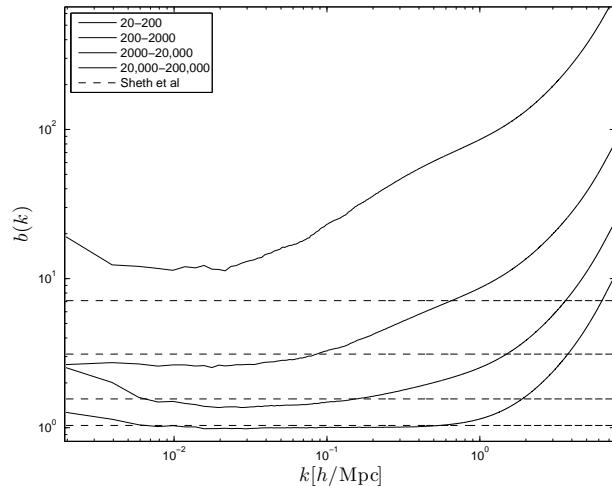


Figure 9. Halo bias, measured at $z = 0$ from a RANGER4000 simulation with a side length of $3.2 h^{-1}$ Gpc. Bottom to top curves correspond to haloes of different masses, from light to massive. The four bins are decades of halo masses; the first contains haloes made of 20-200 particles, the second of 200-2000 and so on. The dashed lines are linear predictions from Sheth et al. (2001).

however, that using more sophisticated halo finders (available only through post-processing calculations due to their heavier memory footprint) it is possible to recover the expected mass function.

A second test of the accuracy of the halo finder algorithm is to extract the halo power spectrum $P_h(k)$ and compare the halo bias with theoretical predictions. The halo bias is defined as $b(k) = \sqrt{P(k)/P_h(k)}$ and is shown in Fig. 9. The results are organized in four mass bins, where the lowest mass consists of 20-200 particles, the second bin contains 200-2000 particles and so on. Comparisons with the linear predictions of Sheth et al. (2001) shows results consistent with the known flaws of the model, namely that low-mass bias is over-predicted and high-mass bias is under-predicted (Tinker et al. 2010).

7 BEYOND THE STANDARD CONFIGURATION

The preceding descriptions and discussions apply to the standard configuration of the code, as described at the beginning of section 2. A few extensions have been recently developed in order to enlarge the range of applications of CUBEP3M, and this section briefly describe the most important improvements.

7.1 Initial conditions

As mentioned in section 2, the code starts off by reading a set of initial conditions. These are organized as a set of $6 \times N$ phase-space arrays – one per MPI process – where N is the number of particles in the local volume. Although many applications are typically based on initial conditions that follow Gaussian statistics, we have developed a non-Gaussian initial conditions generator that we briefly describe in this section.

The original code that provides Gaussian initial conditions for CUBEP3M is extended to include non-Gaussian features of the ‘local’ form, $\Phi(\mathbf{x}) = \phi(\mathbf{x}) + f_{NL}\phi(\mathbf{x})^2 + g_{NL}\phi(\mathbf{x})^3$, where $\phi(\mathbf{x})$ is the Gaussian contribution to the Bardeen potential $\Phi(\mathbf{x})$ (see Bartolo et al. (2004) for a review). We adopted the CMB convention, in

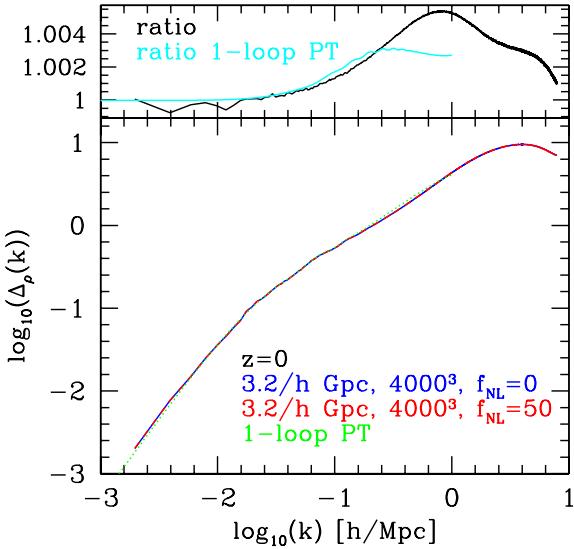


Figure 10. Dark matter power spectrum, measured at $z = 0$ in a volume $3.2 h^{-1} \text{Gpc}$ per side, from 4000^3 particles. The two curves represent two RANGER4000 realizations of the same initial power spectrum, one of which used Gaussian statistics (blue) and the other the non-Gaussian initial condition generator. The two curves differ at the sub-per cent level, as seen in the top panel, and the non-Gaussian realization is in much better agreement with the one-loop perturbation theory calculations.

which Φ is calculated immediately after the matter–radiation equality (and not at redshift $z = 0$ as in the large scale structure convention). For consistency, $\phi(\mathbf{x})$ is normalized to the amplitude of scalar perturbations inferred by CMB measurements ($A_s \approx 2.2 \times 10^{-9}$). The local transformation is performed before the inclusion of the matter transfer function, and the initial particle positions and velocities are finally computed from $\Phi(\mathbf{x})$ according to the Zel'dovich approximation, as in the original Gaussian initial condition generator.

This code was tested by comparing simulations and theoretical predictions for the effect of local primordial non-Gaussianity on the halo mass function and matter power spectrum (Desjacques, Seljak & Iliev 2009). It has also been used to quantify the impact of local non-Gaussian initial conditions on the halo power spectrum (Desjacques et al. 2009; Desjacques & Seljak 2010) and bispectrum (Sefusatti et al. 2010), as well as the matter bispectrum (Sefusatti et al. 2011). Fig. 10 shows the late time power spectrum of two RANGER4000 realizations that started off the same initial power spectrum, but one of which had non-Gaussian initial conditions. We see that the difference between the two power spectra is at the sub-per cent level, but that the non-Gaussian realization agrees much better with one loop perturbation theory (Scoccimarro et al. 2004; Taruya et al. 2008).

7.2 Particle identification tags

A system of particle identification can be turned on, which basically allows to track each particle's trajectory between checkpoints. Such a tool is useful for a number of applications, from reconstruction of halo merging history to tracking individual particle trajectories. The particle tag system has been implemented as an array of double integers, PID, and assigns a unique integer to each particle during

the initialization stage. The location of the tag on the PID array matches the location of the corresponding particle on the xv array, hence it acts as if the latter array had an extra dimension. The location change only when particles exist the local volume, in which case the tag is sent along adjacent nodes with the particle in the `pass_particle` subroutine, and deleted particles result in deleted flags. Similarly to the phase space array, the PID array gets written to file at each particle checkpoint. Since the array takes a significant amount of memory, we opted to store the tags in a separate object – as opposed to adding an extra dimension to the existing xv array, such that it can be turned off when memory becomes an issue.

7.3 Extended range of the pp force

One of the main source of error in the calculation of the force occurs when on the smallest scales of the fine grid. The approximation by which particles in a neighbouring mesh grid can be placed at the centre of the cell is less accurate, which cause a maximal scatter around the exact $1/r^2$ law. A solution to minimize this error consists in extending the pp force calculation outside a single cell, which inevitably reintroduces a N^2 number of operations. Our goal is to add the flexibility to have a code that runs slower, but produces results with a higher precision.

To allow this feature, we have to choose how far outside a cell we want the exact pp force. Since the force kernels on both meshes are organized in terms of grids, the simplest way to implement this feature is to shut down the mesh kernels in a region of specified size, and allow the pp force to extend therein. Concretely, these regions are constructed as cubic layers of fine mesh grids around a central cell; the freedom we have is to choose the number of such layers.

To speed up the access to all particles within the domain of computation, we construct a thread safe linked list to be constructed and accessed in parallel by each core of the system, but this time with a head-of-chain that points to the first particle in the current fine mesh cell. We then loop over all fine grids, accessing the particles contained therein and inside each fine grid cells for which we killed the mesh kernels, we compute the separation and the force between each pairs and update their velocities simultaneously with Newton's third law. To avoid double counting, we loop only over the fine mesh neighbours that produce non-redundant contributions. Namely, for a central cell located at (x_1, y_1, z_1) , we only consider the neighbours (x_2, y_2, z_2) that satisfy the following conditions:

- $z_2 \geq z_1$ always
- if $z_2 = z_1$, then $y_2 \geq y_1$, otherwise we also allow $y_2 < y_1$
- if $z_2 = z_1$ and $y_2 = y_1$, then we enforce $x_2 > x_1$

The case where all three coordinates are equal is already calculated in the standard configuration of the code. To assess the improvement of the force calculation, we present in Fig 11 a force versus distance plot in a CITA128 realization, analogous to Fig. 5, but the pp force has been extended to two layers of fine cells in the force test part of the code. We observe that the scatter about the theoretical curve has reduced significantly, and is still well balanced around the theoretical predictions. The fractional error on the radial and tangential components of the force, as seen in the right panel, are now at least 5 time smaller than in the default P³M algorithm. When averaging over 10 time steps, we observe that the improvement is only mild, as the calculation is already very accurate (**To verify/quantify**).

To quantify the accuracy improvement versus computing time

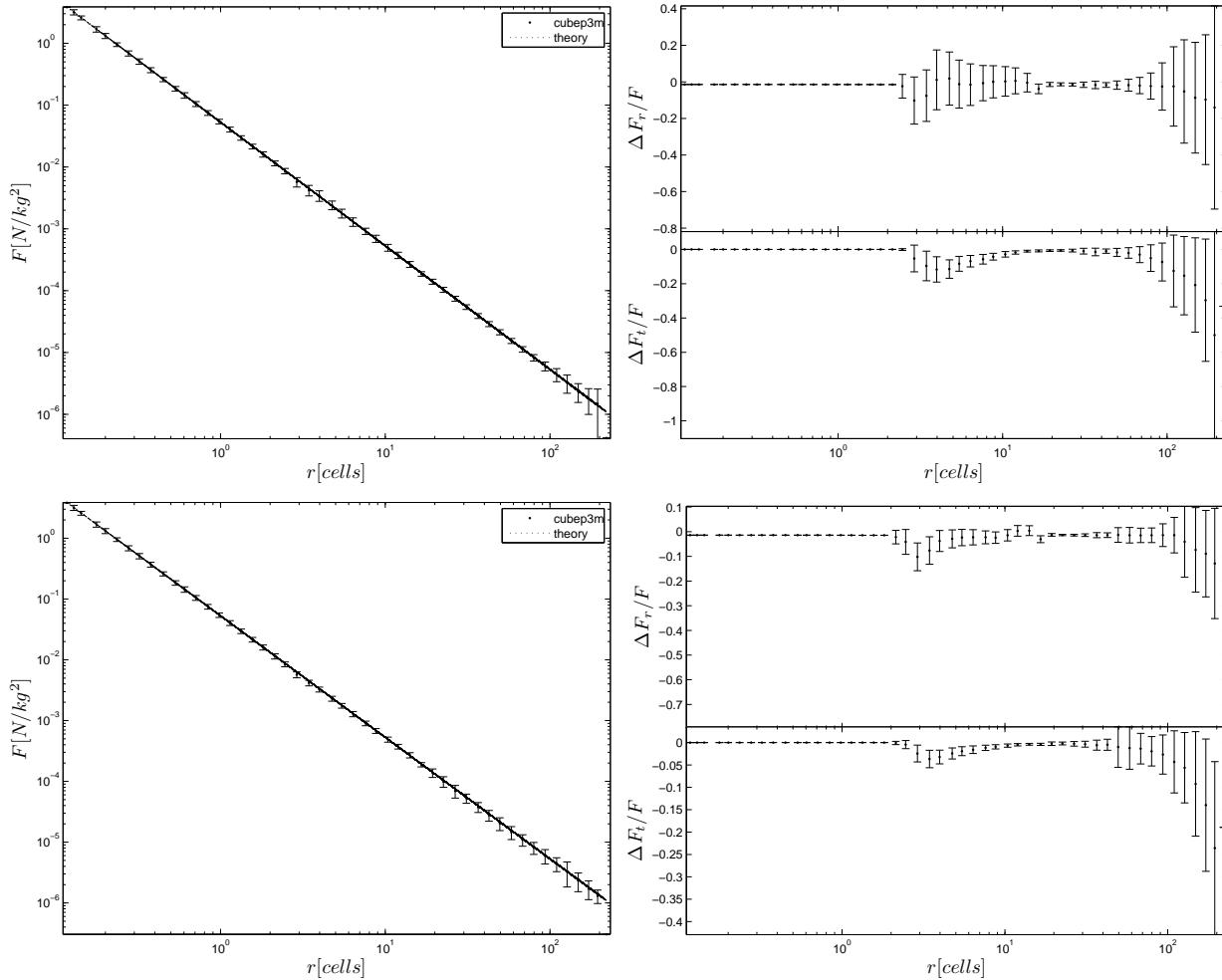


Figure 11. (top left:) Gravity force in the P^3M algorithm, compared with the exact $1/r^2$ law, in the same CITA128 realization as that shown in Fig. 5, except that the exact pp force has been extended to two fine mesh layers around each particle in the force test code. Particles in the that range follow the exact curve, then we observe a much smaller scatter at distance of the order of 2 fine grid, compared to Fig. 5. The NGP interpolation scheme is again responsible for the scatter, but the effect is suppressed with increasing distances, and further suppressed by averaging over many time steps. (top right:) Fractional error on the force in the P^3M algorithm, in the radial (top) and tangential (bottom) directions. This was also obtained over a single time step. (bottom row:) Same as top row, but averaged over ten time steps.

requirements, we performed the following test. We generate a set of initial conditions at a starting redshift of $z = 50$, with a box size equal to $150h^{-1}\text{Mpc}$, and with 512^3 particles. We evolve these CITA512 realizations with different ranges for the pp calculation, and compare the resulting power spectra. For the results to be meaningful, we also need to use the same random seed for the random number generator, such that the only difference between different runs is the range of the pp force. Fig. 12 shows the dimensionless power spectrum of the different runs, where we see a significant gains in resolution when extending PM to P^3M first, and when adding successively one and two layers of fine cell mesh in which the pp force is extended. We have not plotted the results for higher numbers of layers, as the improvement becomes milder there: the mesh calculations become accurate enough as the distance increases. For this reason, it seems that a range of a two layers, suffices to reduce most of the undesired NGP scatter.

Extending the pp calculation comes at a price, since the number of operation scales as N^2 in the sub-domain. This cost is best capture by the increase of real time required by a fixed number of dedicated CPUs to evolved the particles to the final redshift.

For instance, in our CITA512 simulations, the difference between $N_{layer} = 1$ and 2 is about a factor of 2.3 in real time. This number will change depending on the problem at hand and on the machine, and we recommend to perform performance gain vs resource usage tests on smaller runs before performing large scale simulations with the extended force.

The power spectrum does not provide the complete story, and one of the most relevant way to quantify the improvement of the calculation is to measure the impact on the halo mass function from these different CITA512 runs. Fig. 13 presents this comparison at redshift two. About 56,000 haloes were found in the end, yielding a halo number density of about 0.0166 per Mpc/h^3 . We observe that the simulation undershoot the Sheth-Tormen predictions, which is caused by XXXXX. We also observe that the pure PM code yields up 10 per cent less haloes than the P^3M version, whereas the extended pp algorithm generally recovers up to 10 per cent more haloes. The difference in performance between an extended pp range of two vs four cells is rather mild, from where we conclude that $N_{layers} = 2$ is the most optimal choice.

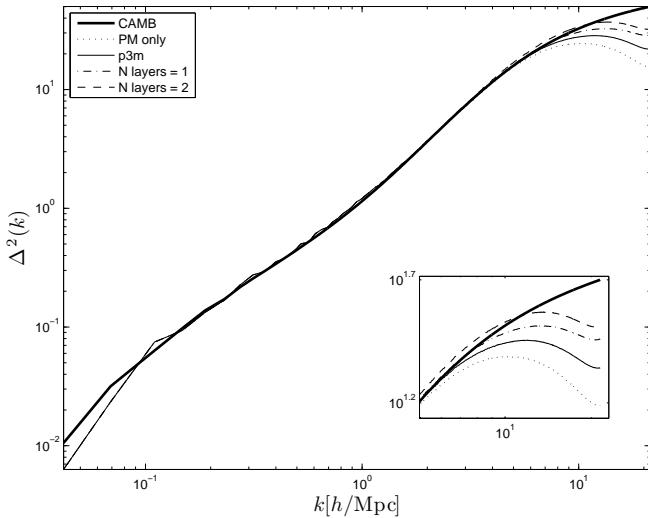


Figure 12. Dimensionless power spectrum for varying range of the exact pp force, compared to CAMB (Lewis et al. 2000). These CITA512 realizations evolved from a unique set of initial conditions at a starting redshift of $z = 50$, with a box size equal to $150h^{-1}\text{Mpc}$ until $z = 2.0$. The only difference between the runs is the ranges of the pp calculation. The inset shows details about the resolution turnaround.

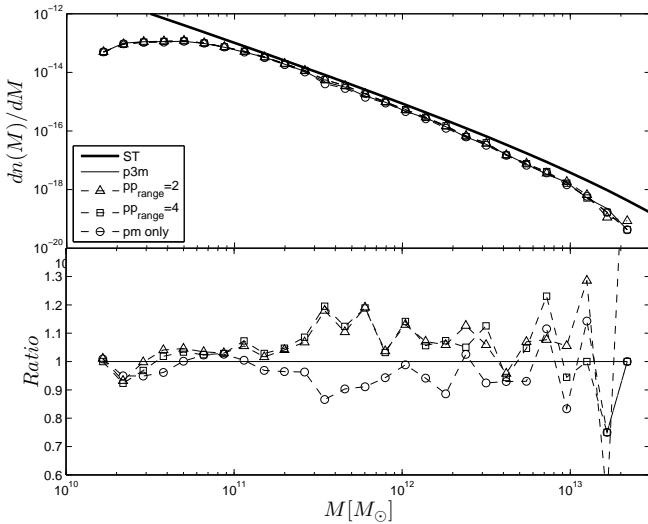


Figure 13. (top:) Halo mass function for different ranges of the pp force calculation, compared to the predictions of Sheth-Tormen. The smallest haloes shown here have a mass equivalent to 100 particles, and fall in the lowest mass bin. (bottom:) Ratio between the different curves and that of the default P³M configuration.

8 CONCLUSION

This paper describes CUBEP3M, a public and massively parallel P³M N-body code that inherits from PMFAST and that now scales well to 20,000 cores, pushing the limits of the cosmological problem size one can handle. We summarize the code structure, review the double-mesh Poisson solver algorithm, and present scaling and systematic tests that have been performed. We also describe various utilities and extensions that come with the public release, including a run time halo finder, an extended pp force calculation and a non-

Gaussian initial condition generator. CUBEP3M is one of the most competitive N-body code that is publicly available for cosmologists and astrophysicists, it has already been used for a large number of scientific applications, and it is our hope that the current documentation will help the community in interpreting its outcome.

The code is publicly available on github.com under `cubep3m`, and extra documentation about the structure, compiling and running strategy is can be found at www.wiki.cita.utoronto.ca/mediawiki/index.php/CubePM.

ACKNOWLEDGEMENTS

The CITA simulations were run on the Sunnyvale cluster at CITA. ITI was supported by The Southeast Physics Network (SEPNet) and the Science and Technology Facilities Council grants ST/F002858/1 and ST/I000976/1. The authors acknowledge the TeraGrid and the Texas Advanced Computing Center (TACC) at The University of Texas at Austin (URL: <http://www.tacc.utexas.edu>) for providing HPC and visualization resources that have contributed to the research results reported within this paper. ITI also acknowledges the Partnership for Advanced Computing in Europe (PRACE) grant 2010PA0442 which supported the code scaling studies. (Ue-Li, Hugh, Vincent, JD, any financial acknowledgements to add here?)

REFERENCES

- Eleftheriou M., Fitch B. G., Rayshubskiy A., Ward T. J. C., Germain R. S., 2005, IBM J. Res. Dev., 49, 457
- Bartolo N., Komatsu E., Matarrese S., Riotto A., 2004, Phys. Rep., 402, 103
- Bertschinger E., 1985, ApJS, 58, 39
- Colless M., et al., 2003, ArXiv e-prints (astro-ph/0306581)
- Couchman H. M. P., 1991, ApJ, 368, L23
- Couchman H. M. P., Thomas P. A., Pearce F. R., 1995, ApJ, 452, 797
- Datta K. K., Mellema G., Mao Y., Iliev I. T., Shapiro P. R., et al., 2011, arXiv:1109.1284
- Desjacques V., Seljak U., 2010, Phys. Rev. D, 81, 023006
- Desjacques V., Seljak U., Iliev I. T., 2009, MNRAS, 396, 85
- Doré O., Lu T., Pen U.-L., 2009, ArXiv e-prints
- Drinkwater M. J., et al., 2010, MNRAS, 401, 1429
- Dubinski J., Kim J., Park C., Humble R., 2004, NewA, 9, 111
- Eisenstein D. J., et al., 2005, ApJ, 633, 560
- Fernandez E. R., Iliev I. T., Komatsu E., Shapiro P. R., 2011, arXiv:1112.2064
- Friedrich M. M., Datta K. K., Mellema G., Iliev I. T., 2012, ArXiv e-prints
- Friedrich M. M., Mellema G., Alvarez M. A., Shapiro P. R., Iliev I. T., 2011, MNRAS, 413, 1353
- Frigo M., Johnson S., 2005, in Proceedings of the IEEE Vol. 93, The Design and Implementation of FFTW3. pp 216–231
- Harnois-Déraps J., Pen U.-L., 2011, ArXiv e-prints
- Harnois-Déraps J., Vafaei S., Van Waerbeke L., 2012, ArXiv e-prints
- Harnois-Déraps J., Yu H.-R., Zhang T.-J., Pen U.-L., 2012, ArXiv e-prints
- Heymans C., CFHTLenS Collaboration 2012, in American Astronomical Society Meeting Abstracts Vol. 219 of American Astro-

- nomical Society Meeting Abstracts, The CFHT Lensing Survey. p. 130.01
- Hilbert S., Hartlap J., White S. D. M., Schneider P., 2009, A&A, 499, 31
- Hockney R. W., Eastwood J. W., 1981, Computer Simulation Using Particles
- Iliev I., Freese K., Valluri M., Iliev I. T., Shapiro P. R., 2012, MNRAS, p. 2794
- Iliev I. T., Ahn K., Koda J., Shapiro P. R., Pen U.-L., 2010, proceedings paper for Moriond 2010 meeting (ArXiv:1005.2502)
- Iliev I. T., Mellema G., Pen U.-L., Merz H., Shapiro P. R., Alvarez M. A., 2006, MNRAS, 369, 1625
- Iliev I. T., Mellema G., Shapiro P. R., Pen U.-L., Mao Y., Koda J., Ahn K., 2011, ArXiv e-prints
- Iliev I. T., Shapiro P. R., 2001, MNRAS, 325, 468
- Iliev I. T., Shapiro P. R., Mellema G., Merz H., Pen U.-L., 2008, in refereed proceedings of TeraGrid08, ArXiv e-prints (0806.2887)
- Komatsu E., Dunkley J., Nolta M. R., Bennett C. L., Gold B., Hinshaw G., Jarosik N., Larson D., Limon M., Page L., Spergel D. N., Halpern M., Hill R. S., Kogut A., Meyer S. S., Tucker G. S., Weiland J. L., Wollack E., Wright E. L., 2009, ApJS, 180, 330
- Komatsu E., et al., 2011, ApJS, 192, 18
- Lacey C., Cole S., 1994, MNRAS, 271, 676
- Lewis A., Challinor A., Lasenby A., 2000, Astrophys. J., 538, 473
- Lu T., Pen U.-L., 2008, MNRAS, 388, 1819
- Lu T., Pen U.-L., Doré O., 2010, Phys. Rev. D, 81, 123015
- Mao Y., Shapiro P. R., Mellema G., Iliev I. T., Koda J., Ahn K., 2012, MNRAS, 422, 926
- Mellema G., Iliev I. T., Alvarez M. A., Shapiro P. R., 2006, New Astronomy, 11, 374
- Merz H., Pen U.-L., Trac H., 2005, New Astronomy, 10, 393
- Navarro J. F., Frenk C. S., White S. D. M., 1997, ApJ, 490, 493
- Ngan W.-H. W., Harnois-Déraps J., Pen U.-L., McDonald P., MacDonald I., 2011, ArXiv e-prints (astro-ph/1106.5548)
- Percival W. J., Cole S., Eisenstein D. J., Nichol R. C., Peacock J. A., Pope A. C., Szalay A. S., 2007, MNRAS, 381, 1053
- Reed D. S., Bower R., Frenk C. S., Jenkins A., Theuns T., 2007, MNRAS, 374, 2
- Rimes C. D., Hamilton A. J. S., 2005, MNRAS, 360, L82
- Sato M., Hamana T., Takahashi R., Takada M., Yoshida N., Matsubara T., Sugiyama N., 2009, ApJ, 701, 945
- Schlegel D., White M., Eisenstein D., 2009, ArXiv e-prints (astro-ph/0902.4680)
- Scoccimarro R., Sefusatti E., Zaldarriaga M., 2004, Phys. Rev. D, 69, 103513
- Sefusatti E., Crocce M., Desjacques V., 2010, MNRAS, 406, 1014
- Sefusatti E., Crocce M., Desjacques V., 2011, ArXiv e-prints
- Shapiro P. R., Iliev I. T., Raga A. C., 1999, MNRAS, 307, 203
- Sheldon E. S., Johnston D. E., Masjedi M., McKay T. A., Blanton M. R., Scranton R., Wechsler R. H., Koester B. P., Hansen S. M., Frieman J. A., Annis J., 2009, ApJ, 703, 2232
- Sheth R. K., Mo H. J., Tormen G., 2001, MNRAS, 323, 1
- Sheth R. K., Tormen G., 2002, MNRAS, 329, 61
- Springel V., 2005, MNRAS, 364, 1105
- Springel V., Yoshida N., White S. D. M., 2001, New Astronomy, 6, 79
- Takahashi R., et al., 2009, ApJ, 700, 479
- Takahashi R., et al., 2011, ApJ, 726, 7
- Taruya A., Koyama K., Matsubara T., 2008, Phys. Rev. D, 78, 123534
- Tegmark M., et al., 2006, Phys. Rev. D, 74, 123507
- Tinker J., Kravtsov A. V., Klypin A., Abazajian K., Warren M., Yepes G., Gottlöber S., Holz D. E., 2008, ApJ, 688, 709
- Tinker J. L., Robertson B. E., Kravtsov A. V., Klypin A., Warren M. S., Yepes G., Gottlöber S., 2010, ApJ, 724, 878
- Trac H., Cen R., 2007, ApJ, 671, 1
- Trac H., Pen U., 2003, in American Astronomical Society Meeting Abstracts Vol. 35 of Bulletin of the American Astronomical Society, Out-of-Core Hydrodynamic Simulations of the IGM. p. 1365
- Trac H., Pen U.-L., 2004, NewA, 9, 443
- Vafaei S., Lu T., van Waerbeke L., Semboloni E., Heymans C., Pen U.-L., 2010, Astroparticle Physics, 32, 340
- Vale C., White M., 2003, ApJ, 592, 699
- Xu G., 1995, ApJS, 98, 355
- York D. G., et al., 2000, AJ, 120, 1579
- Yu H., Zhang T., Harnois-Déraps J., Pen U., 2010, ArXiv e-prints
- Zackrisson E., Scott P., Rydberg C.-E., Iocco F., Sivertsson S., Östlin G., Mellema G., Iliev I. T., Shapiro P. R., 2010, MNRAS, 407, L74
- Zahn O., Lidz A., McQuinn M., Dutta S., Hernquist L., Zaldarriaga M., Furlanetto S. R., 2007, ApJ, 654, 12
- Zhang T., Yu H., Harnois-Déraps J., MacDonald I., Pen U., 2010, ArXiv e-prints

This paper has been typeset from a \TeX / \LaTeX file prepared by the author.