Mon. Not. R. Astron. Soc. **000**, 1–6 (2011) Printed 7 May 2012 (MN LATEX style file v2.2)

CUBEP3M: High Performance P^{3}M N-body code

Joachim Harnois-Déraps[1,2] ⋆, Ue-Li Pen[1] †, Hugh Merz[3] ‡ and Ilian T. Iliev[4] §
[1]*Canadian Institute for Theoretical Astrophysics, University of Toronto, M5S 3H8, Canada*
[2]*Department of Physics, University of Toronto, M5S 1A7, Ontario, Canada*
[3]*Department of XXX*
[4]*Department of XXX*
[5]*Department of XXX*

7 May 2012

ABSTRACT
This paper presents CUBEP3M, an upgraded version of PMFAST, a two-mesh gravity solver that was already among the fastest N-body codes. Among the principal changes, the Poisson solver includes particle-particles interactions at the sub-grid level and across neighbouring cells, plus each level of the volume decomposition is now cubical. Force kernel have been improved for enhanced matching of the two mesh contributions near the cutoff length. We discuss the structure of the code, its accuracy, and its scaling performance. In addition, many utilities have been added, including a runtime halofinder, particle identification tags and non-Gaussian initial conditions generator, and we briefly describe their implementation strategy and accuracy, when applicable .

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 (**?**) of galaxy clustering (York et al. 2000; Colless et al. 2003; Schlegel et al. 2009; Drinkwater et al. 2010) of weak gravitational lensing (**??**) and of supernovae redshift-distance relations all point towards a standard model of cosmology, in which dark energy and collisionless 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 we understands where and 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 theoretical 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 scale (**??**). 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). **(Ilian, could you say something about reionization here? Just general references as to why one needs N-body codes to perform accurate predictions/forecasts, etc.)**

The basic problem that is addressed with N-body codes is a time evolution of an ensemble of N particles that is subject to gravitational attraction. The brute force calculation requires $O(N^2)$ operation, 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, typically using high performance libraries such as FFTW (Frigo & Johnson 2005).

With the advent of large computing facilities, parallel coding has now become common practice, and N-body codes have evolved both in performance and complexity. Many codes have opted for a tree algorithm, including Gadjet (**??**), PMT (Xu 1995), GOTPM (Dubinski et al. 2004), and Hydra (**?**), in which the local resolution increases with the density of the matter field. These have the advantage to balance the load across the computing units, which enable the calculation of 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 the mesh-refine code of Couchman (1991) and the PM code by **?**. these are not designed to perform large scale PM calculations.

(PM code by japanese? Others codes I should mention? That of Hy Trac?)

⋆ E-mail: jharno@cita.utoronto.ca
† E-mail: pen@cita.utoronto.ca
‡ E-mail: merz@sharcnet.ca
§ E-mail: i.t.iliev@sussex.ac.uk

PMFAST (Merz et al. (2005), MPT hereafter) is one of the first code that is designed such as 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 OPENMPparallel computation. This this setup enable the code to evolve very 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. One of 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 came a new name, CUBEPM, which soon after incorporated particle-particle (pp) interactions at the sub-grid level, and the code was renamed CUBEP3M. It now includes a significant number of new features since MPT : the pp force can be extended to an arbitrary range, the size of the redshift jump can be constrained for improved accuracy during the first time steps, a runtime halofinder 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 which can be switched on or off, and the initial condition generator has been generalized as to include non-Gaussian features **(anything else?)**.

This paper aims at presenting and quantifying these new features that make CUBEP3M one of the most competitive public N-body code. It has already 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; **?**), BAO (Zhang et al. 2010; Ngan et al. 2011; Harnois-Deraps & Pen 2011; **?**), re-ionization (**?**), **(anymore cubep3m papers to cite?)**, and it seems relevant for the community to have access to a paper that describes the methodology, the accuracy and the performance of this public code The paper is structured as follow: XXXX

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 challenges[1]. 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 local volumes[2] or *tiles*, and for maximal efficiency, the number of tiles per node should be at least equal to the number of available processors. Given the available freedom in the parallel configuration, it is generally good practice to maximize the number of OPENMP threads and minimize the number of MPI processes, as the information exchange between cores that are part of the same motherboard is generally much faster. As mentioned in MPT, the computation of the short range force requires each tile to store the fine grid density in a buffer surface around the physical volume it is assigned. The thickness of this surface must be larger than the cutoff length, and we find that a 24 cells deep buffer is a good compromise between memory usage and accuracy **(Hugh, is that correct?)**

Because the coarse grid arrays require 4^3 times less memory per node, the bulk of the foot-print is concentrated in a handful of fine grid arrays. In addition, some of these are required for intermediate steps of the calculations only, hence it is possible to hide some of the coarse grid arrays[3]. 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 a local domain
(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
(v) `kern_f` stores the fine grid force kernel in the three directions

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.

[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] To make this possible, the fine grid arrays are constructed such as to support parallel reading and writing. In practice, this is done by adding an additional dimension to the relevant arrays, such that each CPU accesses a unique memory location.
[3] 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.

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. **(Should I mention leapfrog here or anywhere else?)** 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 in 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 head processor is active. Other libraries such as P3DFFT or Intel MKL-α currently permit this extra level of parallelization, and it is our plan to migrate to one of these in the near future.

If the current redshift corresponds to one of the checkpoints, the code advances all particles to their final location and writes 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 **(is this correct?)**. 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.

(Anything else to say in this section? Units by (Trac & Pen 2004)?)

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' interpolation for the fine grid (Hockney & Eastwood 1981). Particle-particle interactions are not interpolated, but a sharp cutoff kills the force for pairs closer to a tenth of a grid cell.

```
subroutine particle_mesh
   call update_position
   call link_list
   call particle_pass
   !$omp parallel do
   do tile = 1, tiles_node
      call rho_f_ngp
      call cmplx_rho_f
      call kernel_multiply_f
      call force_f
      call update_velocity_f
      if(pp = .true.) then
         call link_list_pp
         call force_pp
         call update_velocity_pp
         if(extended_pp = .true.) then
            call link_list_pp_extended
            call force_pp_extended
            call update_velocity_pp_extended
         endif
      endif
   end do
   !$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 extended pp force calculation to show that it follows the same basic logic. We mention here that the three linked list arrays are distinct entities, and their structure differ slightly.

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:

$$\nabla^2\phi(\mathbf{x}) = 4\pi G\rho(\mathbf{x}) \tag{1}$$

G being Newton's constant. We solve this equation in Fourier space, where we write

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

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

$$\phi(\mathbf{x}) = \int \rho(\mathbf{x}')\omega_\phi(\mathbf{x}' - \mathbf{x})d\mathbf{x}' \tag{3}$$

Although this approach is fast, it involves a finite differentiation which enhances the numerical noise. We therefore opt for a force kernel, which is more accurate but has the inconvenient to require four extra Fourier transforms. In this case, we must solve the convolution in three dimensions:

$$F(\mathbf{x}) = -m\nabla\phi(\mathbf{x}) = \int \rho(\mathbf{x}')\omega_F(\mathbf{x}' - \mathbf{x})d\mathbf{x}' \tag{4}$$

The differentiation does not affect the density since it only acts on un-prime variables, and the force kernel is given by

$$\omega_F(\mathbf{x}) \equiv -\nabla\omega_\phi(\mathbf{x}) = -\frac{mG\hat{\mathbf{r}}}{r^2} \tag{5}$$

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:

$$\omega_s(\mathbf{r}) = \begin{cases} \omega_F(r) - \beta(r) & \text{if } r \leqslant r_c \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

and

$$\omega_l(r) = \begin{cases} \beta(r) & \text{if } r \leqslant r_c \\ \omega_F(r) & \text{otherwise} \end{cases} \quad (7)$$

The vector $\beta(r)$ is related to the fourth order polynomial that is used in the potential case 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

$$\beta(r) = \Big[-\frac{7r}{4r_c^3} + \frac{3r^3}{4r^5} \Big]\hat{\mathbf{r}} \quad (8)$$

Because 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, it was realized that close to the cutoff region, the long range force is always on the low side, whereas the short range force is scattered across the theoretical $1/r^2$ value. These behaviors are purely features of the CIC and NGP interpolation scheme respectively. We identified a small range surrounding the cutoff length, in which we empirically adjust both kernels such as to improve the match. Namely, for $14 \leqslant r \leqslant 16$, $\omega_s(\mathbf{r}) \rightarrow \omega_s(\mathbf{r}) * 0.985$, and for $12 \leqslant r \leqslant 16$, $\omega_l(\mathbf{r}) \rightarrow \omega_l(\mathbf{r}) * 1.2$. The two fudge factors were found by performing force measurements on two particles randomly placed in the volume.

Finally, a choice must be done concerning the longest range of the 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 structure at very large scales. However, detailed studies of gravitational collapse would benefit from the first settings.

(Anything else here?)

4 SCALING PERFORMANCES

Real time versus size, bottle neck, description of the largest runs, etc. **(Ilian, I will need your input here as well. Hugh, do you have anything you would like to share concerning the largest runs you did on Blue Gene?)**

5 SYSTEMATICS AND ACCURACY

To discuss :

1) random shift to shuffle systematic clustering at the grid scale,

2) choice of initial redshift, which can lead to large truncation error if too early, but poor inaccurate Zel'dovich if too late **JD, here you could discuss the ra_max tempering...**

3) limits of resolution caused by softening length,

4) Poisson shot noise if too few particles or unrelaxed systems,

5) finite box size effects (with beat coupling?)

Show plots of pairwise force (transverse and radial), density force, power spectrum, mention the work on covariance matrix by Ngan, Harnoisetal.

6 RUNTIME HALO FINDER

We have implemented a halo finding procedure, which we have developed based on the spherical overdensity (SO) approach (Lacey & 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 CubeP^{3}M data structures discussed earlier. The code first builds the fine-mesh density for each sub-domain using CIC or NGP interpolation. It then proceeds to search for and record all local density maxima above certain threshold (typically set to 100 above mean density) within the local sub-domain's physical volume (excluding the sub-domain buffer zones). 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. The halo center determined this way agrees closely with the center-of-mass of the halo particles discussed below.

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 to the top-hat collapse model). As we accumulate mass we remove it from the mesh, so that no mass element is double-counted. This method is thus inappropriate for finding sub-halos as within this framework those are naturally incorporated in their host halos. Because the halos are found on a grid of finite-size cells, it is possible, especially for the low-mass halos, to overshoot the target overdensity. When this occurs we use an analytical halo density profile to correct the halo mass and radius to the values corresponding to the target overdensity. This analytical density profile is given by Truncated Isothermal Sphere (TIS) profile (Shapiro et al. 1999; Iliev & Shapiro 2001) for overdensities below $\sim$ 130, and $1/r^2$ for lower overdensities. The TIS density profile has a similar outer slope (the relevant one here) to the Navarro, Frenk and White (NFW Navarro et al. 1997) profile, but extends to lower overdensities 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 which are within the halo radius. Their positions and velocities are used to calculate the halo center-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 particle center-of-mass corresponds very well to the halo centre found based on the gridded mass distribution.

II: do we need to say anything about having the option to use other halo finders like AHF and FOF?

Show some examples and comparisions to analytical fits and other halo finders.

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 mention in section 2, the code start off by reading a set of initial conditions. These correspond to a $6 \times N$ phase-space array, where N is the number of particles in the local node. Although most applications so far were based on initial conditions that follow Gaussian statistics, we have developed a non-Gaussian initial condition 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, $\Phi(\mathbf{x}) = \phi(\mathbf{x}) + f_{\rm NL}\phi(\mathbf{x})^2 + g_{\rm NL}\phi(\mathbf{x})^3$, where $\phi(\mathbf{x})$ is the Gaussian contribution to the Bardeen potential $\Phi(\mathbf{x})$ (see **?** for a review). We adopted the CMB convention, in 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).

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 to a number of applications, from reconstruction of halo merging history to **(what else?)** 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 with the particle in the `pass_particle` subroutine. Deleted particles results in deleted flags, and the `PID` array gets written to file at each particle checkpoint. **(Anything else to say here?)**

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:

- $z_2 \geqslant z_1$ always
- if $z_2 = z_1$, then $y_2 \geqslant y_1$, 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 quantify the accuracy improvement versus computing time requirements, we performed the following test. We generate a set of initial conditions at a starting redshift of $z = 100$, with a box size equal to $200h^{-1}$Mpc, and with 128^3 particles. We evolve the particles to $z = 0$ 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. 3 shows the dimensionless power spectrum of the different runs, where we see a significant gains in resolution when extending PM to P^{3}M first, and when adding successive layers of fine cell mesh where the pp force is extended. We have not plotted the results for higher numbers of layers, as the improvement becomes milder there: the fine grid calculations are more accurate as the distance increases. For this reason, it seems that a range of a few layers, between 3 and 6, 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. Table 1 presents this usage.

Another way to quantify the improvement of the calculation is to look at the halo mass function for these different runs. **(To do...)**

7.4 Generalization of the dark energy equation of state

The cosmic expansion is obtained from a third order Taylor expansion in the scale factor of Friedmann's equation, and has now the flexibility to accomodate a running equation of state of the form $\omega(a) = \omega_0 + a\omega_1$, which is the standard parameterization adopted by the Dark Energy Task Force (Albrecht et al. 2006). The code

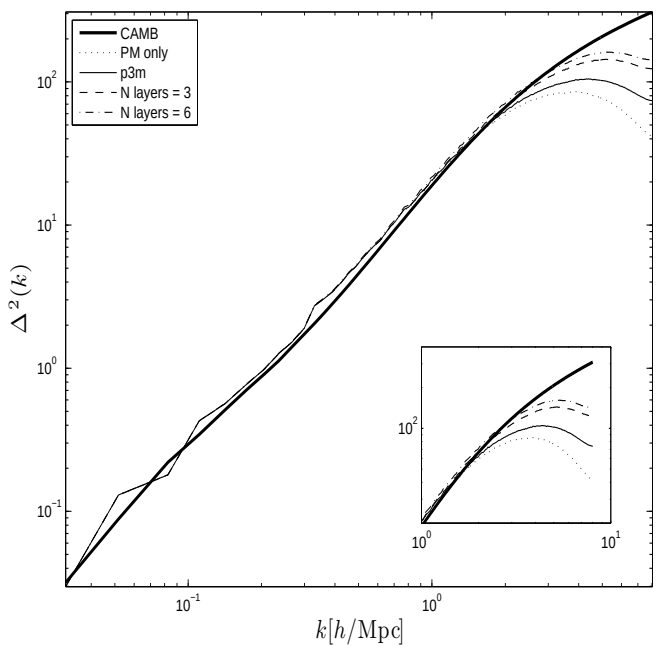

Figure 3. Dimensionless power spectrum for varying range of the exact pp force, compared to CAMB (Lewis et al. 2000).

Table 1. Scaling in CPU resources as a function of the range of the pp interaction. N_{layers} refers to the number of fine mesh layers around a given cell, inside of which the force calculation is purely given by the pp contribution.

Type	time (h)
PM	1.77
P^3M	2.09
$N_{layers} = 1$	8.74
$N_{layers} = 2$	11.47
$N_{layers} = 3$	14.30
$N_{layers} = 4$	18.87
$N_{layers} = 5$	22.52
$N_{layers} = 6$	29.62
$N_{layers} = 7$	34.82
$N_{layers} = 8$	47.00

also currently supports expansion under a modified Chaplygin gas cosmology**?**. **Is this worth mentioning at all?**

8 CONCLUSION

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

REFERENCES

Albrecht A., et al., 2006, ArXiv e-prints (astro-ph/060959)
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
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
Frigo M., Johnson S., 2005, in Proceedings of the IEEE Vol. 93, The Design and Implementation of FFTW3. pp 216–231
Harnois-Deraps J., Pen U.-L., 2011, ArXiv e-prints
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. T., Shapiro P. R., 2001, MNRAS, 325, 468
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
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)
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)
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
Takahashi R., et al., 2009, ApJ, 700, 479
Takahashi R., et al., 2011, ApJ, 726, 7
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
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.