Tree-based solvers for adaptive mesh refinement code FLASH – I: gravity and optical depths

R. Wünsch,1,⋆ S. Walch,2,3 F. Dinnbier1,3,4 and A. Whitworth5

1 Astronomical Institute, Czech Academy of Sciences, Boční I 1401, CZ-141 00 Prague, Czech Republic
2 Max-Planck-Institute for Astrophysics, Karl-Schwarzschild-Str. 1, D-85741 Garching, Germany
3 1 Physikalisches Institut, Universität zu Köln, Zülpicher Str. 77, D-50937 Köln, Germany
4 Faculty of Mathematics and Physics, Charles University in Prague, V Holešovičkách 2, CZ-180 00 Prague, Czech Republic
5 School of Physics and Astronomy, Cardiff University, The Parade, Cardiff CF24 3AA, Wales, UK

Accepted 2017 December 27. Received 2017 November 28; in original form 2017 August 3

ABSTRACT
We describe an OctTree algorithm for the MPI parallel, adaptive mesh refinement code FLASH, which can be used to calculate the gas self-gravity, and also the angle-averaged local optical depth, for treating ambient diffuse radiation. The algorithm communicates to the different processors only those parts of the tree that are needed to perform the tree-walk locally. The advantage of this approach is a relatively low memory requirement, important in particular for the optical depth calculation, which needs to process information from many different directions. This feature also enables a general tree-based radiation transport algorithm that will be described in a subsequent paper, and delivers excellent scaling up to at least 1500 cores. Boundary conditions for gravity can be either isolated or periodic, and they can be specified in each direction independently, using a newly developed generalization of the Ewald method. The gravity calculation can be accelerated with the adaptive block update technique by partially re-using the solution from the previous time-step. Comparison with the FLASH internal multigrid gravity solver shows that tree-based methods provide a competitive alternative, particularly for problems with isolated or mixed boundary conditions. We evaluate several multipole acceptance criteria (MACs) and identify a relatively simple approximate partial error MAC which provides high accuracy at low computational cost. The optical depth estimates are found to agree very well with those of the RADMC-3D radiation transport code, with the tree-solver being much faster. Our algorithm is available in the standard release of the FLASH code in version 4.0 and later.

Key words: gravitation – hydrodynamics – radiative transfer – ISM: evolution – galaxies: ISM.

1 INTRODUCTION

Solving Poisson’s equation for general mass distributions is a common problem in numerical astrophysics. Grid-based hydrodynamic codes frequently use iterative multigrid or spectral methods for that purpose. On the other hand, particle codes often use tree-based algorithms. The extensive experience with tree gravity solvers in particle codes can be transferred to the domain of grid-based codes. Here, we describe an implementation of the tree-based gravity solver for the adaptive mesh refinement (AMR) code FLASH (Fryxell et al. 2000) and show that its efficiency is comparable to the FLASH intrinsic multigrid solver (Ricker 2008). An advantage of this approach is that the tree code can be used for more general calculations performed in parallel with the gravity; in particular, calculation of the optical depth in every cell of the computational domain with the algorithm developed by Clark, Glover & Klessen (2012) and general radiation transport with the TreeRay algorithm (described in Paper II; Wünsch et al., in preparation).

Hierarchically structured, tree-based algorithms represent a well-established technique for solving the gravitational N-body problem at reduced computational cost (Barnes & Hut 1986, hereafter BH86). Many Lagrangian codes implement trees to compute the self-gravity of both collisionless (stars or dark matter) and collisional (gas) particles, e.g. GADGET-2 (Springel 2005), VINE (Wetzstein et al. 2009; Nelson, Wetzstein & Naab 2009), EVOL (Merlin et al. 2010), SEREN, (Hubber et al. 2011) and GANDALF (Hubber, Rosotti & Booth 2018). The three most important characteristics of the tree algorithm are the tree structure (also called the grouping strategy), the multipole acceptance criterion (MAC) deciding whether to open

* E-mail: richard@wunsch.cz

© 2018 The Author(s)
Published by Oxford University Press on behalf of the Royal Astronomical Society
a child-node or not, and the order of approximation of the integrated quantity within nodes (e.g. mass distribution).

**Tree structure:** each node on the tree represents a part of the computational domain, hereafter a volume, and the child-nodes of a given parent node collectively represent the same volume as the parent node. The most common ‘OctTree’ structure is built by a recursive subdivision of the computational domain, where every parent node is split into eight equal-volume child-nodes, until we reach the last generation. The nodes of the last generation are called leaf-nodes and they cover the whole computational domain.

Tree structures other than the OctTree are also often used. Bentley (1979) constructs a balanced ‘k-d’ binary tree by recursively dividing parent nodes so that each of the resulting child-nodes contains half (± 1) of the particles in the parent node; this tree structure is used in the codes **PKDGRAV** (Stadel 2001) and **GASOLINE** (Wadsley, Stadel & Quinn 2004). In contrast, Press (1986) constructs a binary tree, from the bottom up, by successively amalgamating nearest neighbour particles or nodes into parent nodes. This ‘Press-tree’ has been further improved by Jernigan & Porter (1989), and is used, for instance, by Benz et al. (1990) and Nelson et al. (2009). More complex structures have been suggested. For example, Ahn & Lee (2008) describe the ‘k-means’ algorithm, in which a parent node is adaptively divided into k child-nodes according to the particle distribution in the parent-node.

There seems to be no unequivocally superior tree structure. Waltz et al. (2002) compare OctTrees with binary trees, and find that OctTrees provide slightly better performance with the same accuracy. On the other hand, Anderson (1999) argues, on the basis of an analytical study, that certain types of binary trees should provide better performance than OctTrees. Makino (1990) points out that differences in performance are mainly in the tree construction part, and that the tree-walk takes a comparable amount of time in either type of tree structure. Therefore, the choice of tree structure should be informed by more technical issues, like the architecture of the computer to be used, other software to which the tree will be linked, and so on.

**Multipole acceptance criterion:** another essential part of a tree code is the criterion, or criteria, used to decide whether a given node can be used to calculate the gravitational field, or whether its child-nodes should be considered instead. This is a key factor determining the accuracy and performance of the code. Since this criterion often reduces to deciding whether the multipole expansion representing the contribution from the node in question provides a sufficiently accurate approximation for the calculation of the gravitational potential, it is commonly referred to as the MAC. We retain this terminology even though nodes in the code presented here may possess more general properties than just a multipole expansion.

The original BH86 geometric MAC uses a simple criterion, which is purely based on the ratio of the angular size of a given node and its distance to the cell at which the gravitational potential should be computed. More elaborate methods also take into account the mass distribution within a particular node or even constrain the allowed total acceleration error (Salmon & Warren 1994, SW94; see Section 2.2.1).

**Order of approximation:** Springel, Yoshida & White (2001) suggest that if the gravitational acceleration is computed using multipole moments up to order p, then the maximum error is of the order of the contribution from the (p + 1)th multipole moment. There is no consensus on where to terminate the multipole expansion of the mass distribution in a node. The original BH86 tree code uses moments up to second order (p = 2), i.e. quadrupoles, and many authors follow this choice. Wadsley et al. (2004) find the highest efficiency using p = 4 in the **GASOLINE** code. On the other hand, SW94 find that their code using the SumSquare MAC is most efficient with p = 1, i.e. just monopole moments. This suggests that the optimal choice of p may depend strongly on other properties of the code and its implementation, and possibly also on the architecture of the computer. Springel (2005) advocates using just monopole moments on the basis of memory and cache usage efficiency. We follow this approach and consider only monopole moments, i.e. p = 1 for all implemented MACs.

Further improvements: tree codes have often been extended with new features or modified to improve their behaviour. Barnes (1990) noted that neighbouring particles interact with essentially the same nodes, and introduced interaction lists that save time during a tree-walk. This idea was further extended by Dehnen (2000, 2002) who describes a tree with mutual node–node interactions. This greatly reduces the number of interactions that have to be calculated, leading – in theory – to an O(N) CPU time dependence on the number of particles, N. Dehnen’s implementation also symmetrizes the gravitational interactions to ensure accurate momentum conservation, which is in general not guaranteed with tree codes. Recently, Potter, Stadel & Teyssier (2017) develop this so-called fast multipole method further and implement it into massively parallel cosmological N-body code **PKDGRAV3**.

**Hybrid codes:** tree codes are also sometimes combined with other algorithms into ‘hybrid’ codes. For example, Xu (1995) describes a **TREEMP** code which uses a tree to calculate short-range interactions, and a particle-mesh method (Hockney & Eastwood 1981) to calculate long-range interactions. The **TREEMP** code has been developed further by Bode, Ostriker & Xu (2000), Bagla (2002), Bode & Ostriker (2003), Bagla & Khandai (2009) and Khandai & Bagla (2009). There are also general purpose tree codes available, which can work with both N-body and grid-based codes, e.g. the **MPI** parallel tree gravity solver **FLY** (Becchi et al. 2007).

In this paper, we describe a newly developed, cost-efficient, tree-based solver for self-gravity and diffuse radiation that has been implemented into the **AMR** code **FLASH**. This code has been developed since 2008, and since **FLASH** version 4.0, it is a part of the official release. The GPU accelerated tree gravity solver, based on the early version of the presented code, has been developed by Lukat & Banerjee (2016). The paper is organized as follows: in Section 2, we describe the implemented algorithm, which splits up into the tree-solver (Section 2.1), the gravity module (Section 2.2) and the optical depth module (Section 2.3). Accuracy and performance for several static and dynamic tests are discussed in Section 3, and we conclude in Section 4. In Appendix A, we provide formulae for acceleration in computational domains with periodic and mixed boundary conditions (BCs), and in Appendix B we give runtime parameters of the code.

## 2 THE ALGORITHM

The **FLASH** code (Fryxell et al. 2000) is a complex framework consisting of many inter-operable modules that can be combined to solve a specific problem. The tree code described here can only be used with a subset of the possible **FLASH** configurations. The basic requirement is usage of the **PARAMESH**-based grid unit (see MacNeice et al. 2000 for a description of the **PARAMESH** library); support for other grid units (uniform grid, Chombo) can be added in future. Furthermore, the grid geometry must be 3D Cartesian.

The **PARAMESH** library defines the computational domain as a collection of blocks organized into a tree data structure which we refer
to as the *amr-tree*. Each node on the amr-tree represents a block. The block at the top of the amr-tree, corresponding to the entire computational domain, is called the *root block* and represents refinement level $\ell = 1$. The root block is divided into eight equal-volume blocks having the same shape and orientation as the root block, and these blocks represent refinement level $\ell = 2$. This process of block division is then repeated recursively until the blocks created satisfy an AMR criterion. The blocks at the bottom of the tree, which are not divided, are called *leaf-blocks*, and the refinement level of a leaf-block is labelled $\ell_{lb}$. In regions where the AMR criterion requires higher spatial resolution, the leaf-blocks are smaller and their refinement level, $\ell_{lb}$, is larger (i.e. they are further down the tree).

The number of grid cells in a block (a logically cubicoid collection of cells; see below) must be the same in each direction and equal to $2^n$ where $\ell_{lb}$ is an arbitrary integer number. In practice, it should be $\ell_{lb} \geq 3$, because most hydrodynamic solvers do not allow blocks containing fewer than 8$^3$ cells, in order to avoid overlapping of ghost cells. Note that the above requirements do not exclude non-cubic computational domains, because such domains can be created either by setting up blocks with different physical sizes in each direction or by using more than one root block$^1$ in each direction (Walch et al. 2015).

Within each leaf-block is a local *block-tree* which extends the amr-tree down to the level of individual grid cells. All block-trees have the same number of levels, $\ell_{lb} \geq 3$. The nodes on a block-tree represent refinement levels $\ell_{bt} = \ell_{lb} + 1$ (8 nodes here), $\ell_{bt} + 2 (8^2 = 64$ nodes here), $\ell_{bt} + 3 (8^3 = 512$ nodes here) and so on. The nodes at the bottom of the block-tree are *leaf-nodes*, and represent the grid cells on which the equations of hydrodynamics are solved.

Each node – both the nodes on the amr-tree, and the nodes on the local block-trees – stores collective information about the set of grid cells that it contains, e.g. their total mass, the position of the centre of mass, etc.

Our algorithm consists of a general *tree-solver* implementing the tree construction, communication and tree-walk, and modules which include the calculations of specific physical equations, e.g. gravitational accelerations or optical depths. The tree-solver communicates with the physical modules by means of interface subroutines which allow physical modules, on the one hand to store various quantities on the nodes, and on the other hand to walk the tree accessing the quantities stored on the nodes. When walking the tree, physical modules may use different MACs that reflect the nature of the quantity they are seeking to evaluate. An advantage of this approach is that it makes code maintenance more straightforward and efficient. Moreover, new functionality can be added easily by writing new physical modules or extending existing ones, without needing to change the relatively complex tree-solver algorithm.

The BCs can be either isolated or periodic, and they can be specified in each direction independently, i.e. mixed BCs with one or two directions periodic and the remaining one(s) isolated are allowed (see Section 2.2).

In the following Section 2.1, we describe the tree-solver, and in Sections 2.2 and 2.3, respectively, we give descriptions of the gravity module and the module (called *opticaldepth*) which calculates heating by the interstellar radiation field (ISRF).

---

$^1$ If there is more than one root block, the single tree structure becomes a forest. This decreases the efficiency of the gravity solver, and therefore the number of root blocks should be kept as small as possible.

### 2.1 Tree-solver

The tree-solver creates and utilizes the tree data structure described above. Maintaining a copy of the whole tree on each processor would incur prohibitively large memory requirements. Therefore, only the amr-tree (i.e. the top part of the tree, between the root-block node and the leaf-block nodes) is communicated to all processors. The block-tree within a leaf-block is held on the processor whose domain contains that leaf-block, and communicated wholly or partially to another processor only if it will be needed by that processor during a subsequent tree-walk. The tree-solver itself stores in each tree-node – with the exception of the leaf-nodes – the total mass of the node and the position of its centre of mass, i.e. four floating point numbers. For leaf-nodes (the nodes corresponding to individual grid cells) only their masses are stored, because the positions of their centres of mass are identical to their geometrical centres and are already known. Additionally, each physical module can store any other required quantity on the tree-nodes.

The tree-solver consists of three steps: tree-build, communication and tree-walk. In the tree-build step, the tree is built from bottom up by collecting information from the individual grid cells, summing it, and propagating it to the parent tree-nodes. The initial stages of this step, those that involve the block-trees within individual leaf-blocks, are performed locally. However, as soon as the leaf-block nodes are reached, information has to be exchanged between processors because parent nodes are not necessarily located on the same processor. At the end of this step, each processor possesses a copy of the amr-tree plus all the block-trees corresponding to leaf-blocks that are located on that processor.

The communication step ensures that each processor imports from all other processors all the information that it will need for the tree-walks, which are subsequently called by the physical modules. To this end, the code considers all pairs of processors, and determines what tree information the one processor (say CPU0; see Fig. 1) needs to export to the other processor (say CPU1). To do this, the code walks the block-trees of all the leaf-blocks on CPU0, and applies a suite of MACs (required by the tree-solver itself and the used physical modules) in relation to all the leaf-blocks on CPU1. This suite of MACs determines for each leaf-block on CPU0, the level of its block-tree that delivers sufficient detail to CPU1 to satisfy the resolution requirements of all the physical modules that will be called before the tree is rebuilt. Thus, a leaf-block on CPU0 that has very little physical influence on any of the leaf-blocks on CPU1 (for example by virtue of being very distant or of low mass) may only need to send CPU1 the information stored on its lowest (i.e. coarsest resolution) level, $\ell_{bt}$. Conversely, a leaf-block on CPU0 that has a strong influence on at least one of the leaf-blocks on CPU1 (for example by virtue of being very close or very massive) may only need to send information stored on its highest (finest resolution) level, $\ell_{bt} + \ell_{lb}$. In order to simplify communication, the required nodes of each block-tree on CPU0 are then stored in a 1D array, ordered by level, starting at $\ell = \ell_{bt}$ and proceeding to higher levels (see Fig. 2). Finally, the arrays from all the block-trees on CPU0 are collated into a single message and sent to CPU1. This minimizes the number of messages sent, thereby ensuring efficient communication, even on networks with high latency.

Note that this communication strategy in which tree-nodes are communicated differs from a commonly used one in which particles (equivalents of grid cells) are communicated instead (e.g. GADGET, Springel 2005). In this way, the communication is completed before the tree-walk is executed and the tree-walk runs locally, i.e. separately on each processor. The communication strategy adopted in...
this work provides a significant benefit for the OPTICALDEPTH and the TreeRay modules as they work with a large amount of additional information per grid cell (or particle), which does not have to be stored and communicated (see Section 2.3).

The final step is a tree-walk, in which the whole tree is traversed in a depth-first manner for each grid cell or in general for an arbitrary target point (e.g. the position of a sink particle). During the process, the suite of MACs is evaluated recursively for each node and if it is acceptable for the calculation, subroutines of physical modules that do the calculation are called, otherwise its child-nodes are opened.

The tree-solver itself only implements a simple geometric MAC (Barnes & Hut 1986), which accepts a node if its angular size, as seen from the target point, \( r \), is smaller than a user-set limit, \( \theta_{\text{lim}} \). Specifically, if \( h \) is the linear size of the node and \( r_i \) is the position of the centre of mass of the node, the node is accepted (and so its child-nodes need not be considered) if

\[
\frac{h}{|r - r_i|} < \theta_{\text{lim}}.
\]

It has been shown by Salmon & Warren (1994, hereafter SW94) that the BH86 MAC can lead to unexpectedly large errors when the target point is relatively far from the centre of mass of the node but very close to its edge. Several alternative geometric MACs were suggested to mitigate this problem (Salmon & Warren 1994; Dubinski 1996). Following Springel (2005), we extend the geometric MAC by setting the parameter \( \eta_{\text{SB}} \) such that a node is only accepted if the target point lies outside a cuboid \( \eta_{\text{SB}} \) times larger than the node (with the default value \( \eta_{\text{SB}} = 1.2 \)). Additional MACs specific to the physical modules are implemented by those modules (see Section 2.2).

The tree-walk is the most time-consuming part of the tree-solver. Typically, it takes more than 90 per cent of the computational time spent by the whole tree-solver. We stress that the tree-walk does not include any communication; the tree is traversed in parallel independently on each processor for all the grid cells in the spatial domain of that processor. The tree-solver exhibits very good scaling, with speed-up increasing almost linearly up to at least 1500 CPU cores (see Section 3.5).

### 2.2 GRAVITY module

This module calculates the gravitational potential and/or the gravitational acceleration. We use the same approach as Springel (2005) and store only monopole moments in the tree, because this substantially reduces memory requirements and communication costs. Since masses and centres of mass are already stored on the
tree-nodes by the tree-solver, the gravity module does not contribute any extra quantities to the tree.

In Section 2.2.1, we describe three data-dependent MACs which can be used instead of the geometric MACs of the tree-solver: maximum partial error (MPE), approximate partial error (APE) and the (experimental) implementation of the SumSquare MAC. Furthermore, the code features three different types of gravity BCs. These are isolated (see Section 2.2.2), fully periodic (Section 2.2.3) and mixed BCs (Section 2.2.4). Finally in Section 2.2.6, we describe a technique called the adaptive block update (ABU) to save computational time by re-using the solution from previous time-step when possible.

2.2.1 Data-dependent MACs

A general weakness of the purely geometric MACs is that they do not take into account the amount and internal distribution of mass in a node. This can make the code inefficient if the density is highly non-uniform. For example, if the code calculates the gravitational potential of the multiphase interstellar medium (ISM), the contribution from nodes in the hot rarefied gas is very small, but it is calculated with the same opening angle as the much more important contribution from nodes in dense molecular cores.

**MPE MAC:** To compensate for the above problem, SW94 propose an MAC based on evaluating the maximum possible error in the contribution to the gravitational acceleration at the target point, \( \mathbf{r} \), that could derive from calculating it using the multipole expansion of the node up to order \( p \) (instead of adding directly the contributions from all the constituent grid cells)

\[
\Delta a_{(p)}^{\text{max}} = \frac{G}{d^2} \left( 1 - b_{\text{max}} / d \right)^2 \times \left\{ (p+2) \left( \frac{B_{(p+1)}}{d^{p+1}} \right) - (p+1) \left( \frac{B_{(p+2)}}{d^{p+2}} \right) \right\},
\]

(2)

\[
B_{(p)} = \sum_i \left| m_i \right| \left| \mathbf{r}_i - \mathbf{r}_a \right|^p.
\]

(3)

Here, \( \mathbf{r}_a \) is the mass centre of the node, \( d \equiv |\mathbf{r}_a - \mathbf{r}| \) is the distance from \( \mathbf{r}_a \) to the target point, \( b_{\text{max}} \) is the distance from \( \mathbf{r}_a \) to the furthest point in the node, \( B_{(p)} \) is the \( p \)th-order multipole moment, obtained by summing contributions from all the grid cells \( i \) in the node and \( m_i \) and \( \mathbf{r}_i \) are the masses and positions of these grid cells. The node is then accepted only if \( \Delta a_{(p)}^{\text{max}} \) is smaller than some specified maximum allowable acceleration error. This threshold can either be set by the user as a constant value, \( a_{\text{lim}} \), in the physical units used by the simulation

\[
\Delta a_{(p)}^{\text{max}} < a_{\text{lim}},
\]

(4)

or it can be set as a relative value, \( \epsilon_{\text{lim}} \), with respect to the acceleration from the previous time-step \( a_{\text{old}} \)

\[
\Delta a_{(p)}^{\text{max}} < \epsilon_{\text{lim}} a_{\text{old}}.
\]

(5)

**APE MAC:** An alternative way to estimate the partial error of a node contribution was suggested by Springel et al. (2001). It takes into account the node total mass, but it ignores the internal node mass distribution. It is therefore faster, but less accurate. Using multipole moments up to order \( p \), the error of the gravitational acceleration is of order the contribution from the \( (p+1) \)th multipole moment

\[
\Delta a_{(p)}^{\text{max}} \approx \frac{GM}{d^2} \left( \frac{h}{d} \right)^{p+1},
\]

(6)

where \( M \) is the mass in the node and \( p = 1 \) in our case, since we only store monopole moments. Similar to the MPE MAC, the APE error limit can be either set absolutely as \( a_{\text{lim}} \) (equation 4), or relatively through \( \epsilon_{\text{lim}} \) (equation 5).

**SumSquare MAC:** SW94 argue that it is unsafe to constrain the error using the contribution of a single node only, since it is not known a priori how these contributions combine. They suggest an alternative procedure, which limits the error in the total acceleration at the target point; one variant of this procedure is the SumSquare MAC which sums up squares of \( a_{(p)}^{\text{max}} \) given by equation (2) over all nodes considered for the calculation of the potential/acceleration at a given target point. In this way, the SumSquare MAC controls the total error in acceleration resulting from the contribution of all tree-nodes. This MAC requires a special tree-walk which does not proceed in the depth-first manner. Instead it uses a priority queue, which on-the-fly reorders a list of nodes waiting for evaluation according to the estimated error resulting from their contribution. This feature is still experimental in our implementation, nevertheless we evaluate its accuracy and performance and compare it to other MACs in Section 3.4.

2.2.2 Isolated boundary conditions

In case of isolated BCs, the gravitational potential in a target point given by position vector \( \mathbf{r} \) is

\[
\Phi(\mathbf{r}) = -\sum_{a=1}^{N} \frac{GM_a}{|\mathbf{r} - \mathbf{r}_a|}
\]

(7)

where index \( a \) runs over all nodes accepted by the MAC during the tree-walk, \( M_a \) and \( \mathbf{r}_a \) are the node mass and position. The gravitational acceleration is then obtained either by differentiating the potential numerically, or it is calculated, as

\[
a(\mathbf{r}) = -\sum_{a=1}^{N} \frac{GM_a (\mathbf{r} - \mathbf{r}_a)}{|\mathbf{r} - \mathbf{r}_a|^3}.
\]

(8)

The first approach needs less memory and is slightly faster. The second approach results in less noise, because numerical differentiation is not needed.

2.2.3 Periodic boundary conditions

In case of periodic BCs in all three directions, the gravitational potential is determined by the Ewald method (Ewald 1921; Klessen 1997), which is designed to mitigate the very slow convergence in case one evaluates contributions to the potential, essentially \( 1/d \) where \( d = |\mathbf{r} - \mathbf{r}_a| \), over an infinite number of periodic copies, by brute force. This is achieved by splitting it into two parts

\[
1/d = \frac{\text{erf}(\alpha d)}{d} + \frac{\text{erf}(\alpha d)}{d}
\]

(9)

and summing the term \( \text{erf}(\alpha d)/d \) in Fourier space; \( \alpha \) is an arbitrary constant controlling the number of nearby and distant terms which have to be taken into consideration. In this section, we present formulae only for the potential. The expressions for acceleration are straightforward to derive, and we list them in Appendix A.

The computational domain is assumed to be a rectangular cuboid, with sides \( L_x, L_y, L_z \) or \( bL_x \) and \( cL_x \) where \( b \) and \( c \) are arbitrary real numbers. The gravitational potential \( \Phi \) at the target point, \( \mathbf{r} \), is...
then
\[
\Phi(r) = -G \sum_{a=1}^{N} M_a \left( \sum_{k_{i_1,j_1}} \frac{1}{bcL_x^2} \sum_{k_{i_2,j_2}} \frac{4\pi}{k^2} \exp \left( -\frac{k^2}{4\alpha^2} \right) \cos(k \cdot (r - r_a)) \right) + \frac{1}{bcL_x^2} \sum_{k_{i_2,j_2}} \frac{4\pi}{k^2} \exp \left( -\frac{k^2}{4\alpha^2} \right) \cos(k \cdot (r - r_a)) \right) \right) \right). \tag{11}
\]

Here, the first inner sum corresponds to short-range contributions, \(\phi_3(r - r_a)\), from the nearest domains in physical space, and the second sum constitutes long-range contributions, \(\phi_1(r - r_a)\). The outer sum runs over all accepted nodes in the computational domain \(M_a\), is the mass of a node, and \(r_a\) is its centre of mass\(^2\). Indices \(i_1\), \(i_2\), and \(i_3\) are integer numbers; \(e_x\), \(e_y\), and \(e_z\) are unit vectors in the corresponding directions; and \(k\) is a wavevector with components \(k_1 = 2\pi i_1/L_x\), \(k_2 = 2\pi i_2/L_x\), and \(k_3 = 2\pi i_3/cL_x\), where \(l_1\), \(l_2\), and \(l_3\) are integer numbers. By virtue of the Ewald method, both inner sums converge very fast. We follow Hernquist, Bouchet & Suto (1991) in estimating
\[
i_1^2 + (bi_2)^2 + (ci_3)^2 \leq 15 \tag{12}
\]
\[
i_1^2 + (l_2/b)^2 + (l_3/c)^2 \leq 10 \tag{13}
\]
and \(\alpha = 2/L_x\).

2.2.4 Mixed boundary conditions

We generalize the Ewald method, which was developed for computational domains with periodic BCs in all spatial directions, to computational domains with mixed BCs. In 3D space, mixed BCs can be of two types: periodic BCs in two directions (without loss of generality we choose \(x\)- and \(y\)-directions), and isolated BCs in the third \((z)\) direction; and periodic BCs in one direction (we choose \(z\)), and isolated BCs in the other two directions. We abbreviate the former case of mixed BCs as 2P1I, and the latter case as 1P2I. Configuration 2P1I has planar symmetry with axis \(e_y\), while configuration 1P2I has an axial symmetry along axis \(e_z\). These configurations might be convenient for studying systems with the symmetry (i.e. layers or filaments). We note that directions that can be defined as periodic are given by computational domain boundaries and thus they can only be parallel with one or more of the Cartesian coordinate axes.

We find the expression for \(\Phi(r)\) for mixed BCs of 2P1I type by taking a limit of equation (11). Consider a computational domain with side lengths \(L_x\), \(L_y\), \(bL_x\), \(L_y\) = \(cL_x\) and with periodic BC in all three directions, for which the gravitational potential is given by equation (11). Next we shift periodic copies of this domain in the \(z\)-direction so that the periodicity in the \(z\)-direction is \(n\) times larger, i.e. \(L_z = nL_z\), where \(n\) is an integer number and \(L_z\) is the extent in the \(z\)-direction of the original computational domain (Fig. 3). Since the copies are shifted and not stretched, the mass distribution between \(z = 0\) and \(L_z\) is unaltered, and the density is zero between \(z = L_z\) and \(nL_z\), leaving all mass concentrated in plane-parallel layers of thickness \(z = L_z\) and with normals pointing in direction \(e_z\). As \(n\) increases, the layers move away from one another, but equation (11) still holds. In the limit \(n \rightarrow \infty\), the periodic copies of the computational domain are touching one another in \(x\)- and \(y\)-directions, however, neighbouring layers in the \(z\)-direction are at infinite distance and hence they do not contribute to the gravitational field in the original computational domain.

As \(n\) increases, the short-range contributions are zero for all \(i_3 \neq 0\), because the argument of the complementary error function in equation (11) tends to infinity. The long-range term \(\phi_1(r - r_a)\) in the limit \(n \rightarrow \infty\) becomes
\[
\phi_1(r - r_a) = \frac{1}{\pi L_x b} \sum_{i_1,l_2} \exp \left( -\frac{\pi^2}{\alpha^2 L_z^2} \left( i_1^2 + (l_2/b)^2 \right) \right) \times \lim_{n \to \infty} \frac{1}{n} \sum_{i_1,l_2} \exp \left( -\frac{n^2 \pi^2}{\alpha^2 L_z^2} \left( i_1^2 + (l_2/b)^2 + (l_3/c)^2 \right) \right) \times \frac{2\pi l_3 (x - x_a)}{L_x} + \frac{2\pi l_3 (y - y_a)}{bL_x} + \frac{2\pi d^2 (z - z_a)}{cL_z n} \right). \tag{14}
\]

The condition (13), which is now \(i_1^2 + (l_2/b)^2 + (l_3/c)^2 \leq 10\) requires us to conserve resolution in the \(z\)-direction in Fourier space, i.e. to increase the range of \(L_z\) with \(n\) linearly (see Fig. 3).

Note that \(2\pi (z - z_a)/(cL_z)\) is independent of \(n\), because we restrict all mass in the computational domain to interval \((0, L_z)\), i.e. \(|z - z_a| \leq cL_z\) for any target point at \(r\) and node at \(r_a\). Bearing this in mind, the term after the limit sign in equation (14) corresponds to a Riemann sum over interval \((-\sqrt{10}, \sqrt{10})\) with equally spaced partitions of size 1/\(nc\). Using the identity \(\cos(A + B) = \cos(A)\cos(B) - \sin(A)\sin(B)\) where \(B = 2\pi d (z - z_a)/(cL_z n)\), the limit becomes
\[
\cos \left( \frac{2\pi l_1 (x - x_a)}{L_x} + \frac{2\pi l_3 (y - y_a)}{bL_x} \right) \left( I(l_1, l_2, z - z_a). \tag{15}\right.
\]
where
\[ I(l_1, l_2, z - z_a) = 2 \int_0^\infty \frac{\exp(-\xi u^2) \cos(\gamma u)}{l_1^2 + (l_2/b)^2 + u^2} du. \quad (16) \]

To keep the notation compact, we introduce \( \gamma = \frac{\pi}{(\alpha L_\alpha)^2} \). In order to evaluate the integral analytically, we extend the interval of integration to infinity (this extension means that we evaluate the sum even slightly more accurately than by condition 13) If \( |l_1| + |l_2| \neq 0 \), we have
\[ I(l_1, l_2, z - z_a) = \frac{\pi \exp(-\gamma^2/4\zeta)}{2\sqrt{l_1^2 + (l_2/b)^2}} \times \left\{ \text{erfcx} \left( \frac{\xi \sqrt{l_1^2 + (l_2/b)^2} - \gamma/2}{{\sqrt{\xi}}} \right) + \text{erfcx} \left( \frac{\xi \sqrt{l_1^2 + (l_2/b)^2} + \gamma/2}{{\sqrt{\xi}}} \right) \right\}. \quad (17) \]

where \( \text{erfcx}(A) = \exp(A^2) \text{erfc}(A) \). When \( l_1 = l_2 = 0 \), integral (16) is infinite, but this property can be circumvented. With the help of \( \cos(\gamma u) = 1 - \sin^2(\gamma u/2) \), we get two integrals corresponding to these two terms. The former one is infinite, but independent of the spatial coordinates and we set it to zero. The latter one can easily be integrated
\[ I(0, 0, z - z_a) = -\pi \left\{ \gamma \text{erf} \left( \frac{\gamma}{2\sqrt{\xi}} \right) + 2\sqrt{\pi} \frac{\exp(-\gamma^2/4\zeta)}{\sqrt{\xi}} \right\} + 2\sqrt{\pi} \zeta. \quad (18) \]

Now we can write the potential as
\[ \Phi(r) = -G \sum_{i=1}^N m_i \left\{ \sum_{l_1 l_2 z_1 z_2} \frac{\text{erfc}(\alpha |r - r_a - i\epsilon_x L_x - i\epsilon_y b L_x|)}{|r - r_a - i\epsilon_x L_x - i\epsilon_y b L_x|} + \frac{1}{\pi L_x b} \sum_{l_1 l_2 z_1 z_2} \exp(-\xi (l_1^2 + (l_2/b)^2)) \times \cos \left( \frac{2\pi d_1 (x - x_a)}{L_x} + \frac{2\pi d_2 (y - y_a)}{L_x} \right) \right\} \times I(l_1, l_2, z - z_a). \quad (20) \]

Note that the ratio \( c \) is not contained in \( \Phi(r) \) as we may expect, because it is of no physical significance when the BCs are isolated in this direction.

The modification of the Ewald method for a computational domain with mixed BCs of type 1P2I can be derived in a similar way to the previous case. However, the integration is more demanding here, because the result of the limiting process is a double integral
\[ \phi_L(r - r_a) = \frac{1}{\pi L_x b} \sum_{l_1 l_2 z_1 z_2} \cos \left( \frac{2\pi d_1 (x - x_a)}{L_x} + \frac{2\pi d_2 (y - y_a)}{L_x} \right) \times \tilde{I}(l_1, l_2, z - z_a), \quad (19) \]

where function \( \tilde{I}(l_1, l_2, z - z_a) \) is defined by equation (A12).

Tree-based solvers for AMR code FLASH – I

In the second approach, we avoid the singularity of \( \phi(r - r_a) \) by subtracting the term \( 1/|r - r_a| \) from \( \phi(r - r_a) \). This enables us to use only one interpolating grid with uniform coverage for the whole computational domain. Moreover, for mixed BCs, \( \phi(r - r_a) \) can be approximated at some parts of the computational domain by analytic functions. The function \( \phi(r - r_a) \) converges to \( 2\pi(z - z_a)/bL_x \) with increasing \( (z - z_a)/L_x \) for configuration 2P1I, and it converges to \( 2\ln(\sqrt{(y - y_a)^2 + (z - z_a)^2} + L_x) \) with increasing \( (y - y_a)^2 + (z - z_a)^2) \) for configuration 1P2I. The convergence is exponential and the relative error in acceleration is always smaller than \( 10^{-4} \) if \( (z - z_a) > 2L_x \) and \( (y - y_a)^2 + (z - z_a)^2 > 2L_x \), so the grid covers only a fraction of the computational domain if the computational domain is elongated. In combination with using only one interpolating grid, this results in smaller demands on memory while it retains the same accuracy as in the first approach.

In the second approach, we pre-calculate not only \( \phi(r - r_a) \) but also its gradient. The actual value of \( \phi(r - r_a) \) at a given location is then estimated by a Taylor expansion to the first order. This is faster
than the trilinear interpolation used in the first approach, and leads
to a speed up in the GRAVITY module by a factor of $\approx 1.4$ to $\approx 1.9$
depending on the shape of the computational domain, the adopted
BCs, and whether the potential or acceleration is used. Thus, the
second approach appears to be superior to the first one. In each
approach, if gravitational accelerations rather than the potential are
required, we adopt an analogous procedure for each of its Cartesian
components.

Note that in a very elongated computational domain, the evalua-
tion of $\phi(r - r_b)$ can be accelerated by adjusting the parameter
$\alpha = 2/L_b$. Since $\phi(r - r_b)$ is pre-calculated, the choice of $\alpha$ is of
little importance in our implementation and we do not discuss it
further in this paper.

2.2.6 Adaptive block update

Often, it is not necessary to calculate the gravitational poten-
tial/acceleration at each grid cell in each time-step. Since the FLASH
code uses a global time-step controlled by the Courant–Friedrichs–
Lewy (CFL) condition, there may be large regions of the com-
putational domain where the mass distribution almost does not
change during one time-step. In such regions, the gravitational po-
tential/acceleration from the previous time-step may be accurate
enough to be used also in the current time-step. Therefore, to save
the computational time, we implement a technique called the ABU.
If activated, the tree-walk is modified as follows. For each block,
the tree-walk is at first executed only for the eight corner grid cells
of the current block. Then, the gravitational potential or acceler-
ation (or any other quantity calculated by the tree-solver, e.g. the
optical depth) in those eight grid cells is compared to the values
from the previous time-step. If all the differences are smaller than
the required accuracy (given e.g. by equation 4 or 5), the previous
time-step values are adopted for all grid cells of the block.

For some applications, the eight test cells in the block corners
may not be sufficient. For instance, if the gas changes its configura-
tion in a spherically symmetric way within a block, the gravitational
acceleration at the block corners does not change, even though the
acceleration may change substantially in the block interior. Such
situation is more probable if larger blocks than default 8$^3$
cells are used. Therefore, it is easily possible to add more test cells
by editing array gr_bTestCells in file gr_bhData.F90,
where test cells are listed using cell indices within a block, i.e. in a
form (1,1,1), (1,1,8), (8,8,8).

ABU can save a substantial amount of the computational time,
however, on large numbers of processors it works well only if a
proper load balancing among processors is ensured, i.e. each pro-
cessor should be assigned with a task of approximately the same
computational cost. FLASH is parallelized using a domain decomposi-
tion scheme and individual blocks are distributed among processors
using the space filling Morton curve (see Fryxell et al. 2000, for de-
tails). Each processor receives a number of blocks estimated so
that their total expected computational time measured by a workload
weight is approximately the same as the one for the other pro-
cessors. By default, FLASH assumes that processing each leaf-block
takes approximately the same amount of time to compute, and it
assigns workload weight 2 to each leaf-block (because it includes
active grid cells) and workload weights 1 to all other blocks (they
are used only for interpolations between different AMR levels).

The assumption of the same workload per leaf-block cannot be
used with ABU, because if the full tree-walk is executed for a
given block less often, the average computational time spent on it
is substantially lower in comparison with more frequently updated
blocks. It is generally hard to predict whether a given block will
be fully updated in the next time-step or not without additional
information about the calculated problem. Therefore, we implement
a simple block workload estimate that leads in most cases to better
performance than using the uniform workload, even though it may
not be optimal. It is based on the assumption that the probability that
the block will be updated is proportional to the amount of work done
on the block during several previous time-steps. This assumption
is motivated by considering that a typical simulation includes on
one hand regions where the density and the acceleration change
rapidly (e.g. close to fast moving dense massive objects), and on the
other hand, regions where the acceleration changes slowly (e.g. large
volumes filled with hot rarefied gas). Consequently, the past
workload of a given block provides an approximate estimate of its
current workload. However, this information is valid only until
the density field evolves enough to change the above property of
the region. The time at which this happens can be approximately
estimated as the gas crossing time of a single block. Due to the CFL
condition, the corresponding number of time-steps is approximately
a number of grid cells in a block along one direction. Specifically,
the block workload estimate works as follows. For each leaf-block,
a total number of node contributions during the tree-walk to all its
grid cells, $N_{int}$, is determined. Then, the workload weight, $W^{(n)}_b$, of
that block is calculated as

$$W^{(n)}_b = W^{(n-1)}_b \exp\left(-\frac{1}{\tau_{wl}}\right) + \left[1 - \exp\left(-\frac{1}{\tau_{wl}}\right)\right] \left(2 + \frac{\omega_{wl} N_{int}}{N_{max}}\right)$$

where $W^{(n-1)}_b$ is the workload weight from the previous time-step,
$\tau_{wl}$ is a characteristic number of time-steps on which the workload
changes, $\omega_{wl}$ is a dimensionless number limiting the maximum
workload weight, and $N_{max}$ is the maximum $N_{int}$ taken over all leaf-
blocks in the simulation. In this way, the block workload weight
depends on its tree-solver computational cost during the last several
($\sim \tau_{wl}$) time-steps and is between 2 (zero cost) and $2 + \omega_{wl}$ (maxi-
mum cost). By default, we set two global parameters $\tau_{wl} = 10$ and
$\omega_{wl} = 8$. The workload weight of non-leaf-blocks remains equal to
1.

2.3 Optical depth module

The optical depth module is used to evaluate the simplified solution
to the radiative transfer equation

$$I_v = I_{v,0} e^{-\tau_v}$$

where $I_v$ is the specific intensity at frequency $v$, $I_{v,0}$ is the specific
intensity at the source location, and $\tau_v$ is the optical depth along a
given path through the computational domain at frequency $v$. In this
form, the problem of evaluating what radiation intensity reaches a
given point in the computational domain, i.e. a given target point,
is reduced to computing the optical depth in between a radiation
source and the target point. The optical depth is proportional to the
absorption cross-section and the column density along the path.

Hence, the optical depth module calculates the total and/or spe-
cific column densities (e.g. of molecular hydrogen) for each cell in
the computational domain, and can therefore be used to compute
the local attenuation of an arbitrary external radiation field. The im-
plementation presented here follows the idea of the TREECOL method
(Clark et al. 2012), which has been implemented in the GADGET

Downloaded from https://academic.oup.com/mnras/article-abstract/475/3/3393/4795314 by Cardiff University user on 23 April 2019
code (Springel et al. 2001). It has been established as a fast but accurate enough approximative radiative transfer scheme to treat the (self-)shielding of molecules – on-the-fly – in simulations of molecular cloud formation (e.g. Clark & Glover 2014). Recently, the method has also been applied in larger scale simulations of Milky Way-like galaxies (Smith, Glover & Klessen 2014) with a given number of pixels, to the accumulation of the node masses during the tree-walk. For each pixel and adding this contribution to the pixel map. For that should be mapped) of the node with the corresponding weight of the nodes is highly efficient. Since \( \theta \) is large set of nodes at different angular positions (\( \phi \)). These values are stored in a look-up table, respectively. This temporary map is filled while walking the tree, as only the tree-nodes in the line of sight of a given pixel contribute to it, and are added accordingly. At the end of the tree-walk, one has acquired a column density map of a given quantity, e.g. total mass.

Since the tree-walk in FLASH is executed on a block-by-block basis, the additional memory requirement for the local pixel maps is \( 2^{l_b} \times N_{\text{pix}} \times l_q \), where \( l_q \) is the number of quantities that are mapped and stored. For this paper, we map \( l_q = 3 \) variables: (1) the total mass giving the total hydrogen column density, \( N_{\text{H},\text{pix}} \); (2) the \( \text{H}_2 \) column of molecular hydrogen, which is used to compute its self-shielding and which contributes to the shielding of \( \text{CO} \); and (3) the \( \text{CO} \) column of carbon monoxide, which is necessary to compute the self-shielding of \( \text{CO} \). We store three separate maps because we actually follow the relative mass fractions of multiple species in the simulation using the FLASH MULTISPECIES module. After the tree-walk for a given block has finished, the local maps are erased and the arrays can be re-used for the next block. This approach is only possible because the tree-walk is computed locally on each processor (see Section 2.1).

When using the OPTICALDEPTH module, there are two major modifications with respect to the usual tree-walk (as described above). First, the intersection of a given tree-node with the line of sight of each pixel has to be evaluated during the tree-walk. Second, at the end of the tree-walk for a given block, the acquired column density maps have to be evaluated for each cell.

Node-ray intersection: the mapping of tree-nodes on to the individual pixels represents the core of all additional numerical operations that have to be carried out when running OPTICALDEPTH in addition to the gravity calculation. It has to be computationally efficient in order to minimize additional costs. At this point, we do not follow the implementation of Clark et al. (2012), who make a number of assumptions about the shape of the nodes and their projection on to the pixels, which are necessary to reduce the computational cost. Instead, we pre-compute the number of intersecting HEALPIX rays and their respective, relative weight for a large set of nodes at different angular positions (\( \theta, \phi \)) and different angular sizes \( \psi \). These values are stored in a look-up table, which is accessed during the tree-walk. In this way, the mapping of the nodes is highly efficient. Since \( \theta, \phi, \) and \( \psi \) are known, we can easily compute the contribution of a node to all intersecting pixels by simply multiplying the mass (or any other quantity that should be mapped) of the node with the corresponding weight for each pixel and adding this contribution to the pixel map. For better accuracy, we oversample the HEALPIX tessellation and construct the table for four times more rays than actually used in the simulation.

Radiative heating and molecule formation: the information that is obtained by the OPTICALDEPTH module is necessary to compute the local heating rates and the formation and dissociation rates of \( \text{H}_2 \) and \( \text{CO} \). At the end of the tree-walk for a given block, the mean physical quantities needed by the CHEMISTRY module calculating the interaction of the radiation with the gas are determined. For instance, the mean visual extinction in a given grid cell is

\[
A_V = -\frac{1}{2.5} \ln \left[ \frac{1}{N_{\text{pix}}} \sum_{i=1}^{N_{\text{pix}}} \exp \left( -2.5 \frac{N_{\text{H},\text{pix}}}{1.87 \times 10^{21} \text{ cm}^{-2}} \right) \right]
\]

where the constant \( 1.87 \times 10^{21} \text{ cm}^{-2} \) comes from the standard relation between the hydrogen column density, \( N_{\text{H},\text{pix}} \), and the visual extinction in a given direction (Draine & Bertoldi 1996). The weighted mean is calculated in this fashion, because the photodissociation rates of molecules such as \( \text{CO} \) and the photoelectric heating rate of the gas all depend on exponential functions of the visual extinction (see Clark et al. 2012, for details). Additionally, the shielding coefficients, \( f_{\text{shield},\text{H}_2} \) and \( f_{\text{shield,CO}} \) (Glover & Mac Low 2007; Glover et al. 2010), as well as the dust attenuation, \( \chi_{\text{dust}} \) (Glover & Clark 2012; Clark et al. 2012), are computed by averaging over the HEALPIX maps in a similar way. These quantities are stored as globally accessible variables and can be used by other modules. In particular, we access them in the CHEMISTRY module, which locally (in every cell) evaluates a small chemical network (Glover et al. 2010) on the basis of its current density and internal energy and recomputes the relative mass fractions of the different chemical species. The evaluation of the chemical network is operator split and employs the DVIDGE solver (Brown, Byrne & Hindmarsh 1989) to solve a system of coupled ordinary differential equations (ODEs) that describes the chemically reactive flow for the given species, i.e. their creation and destruction within a given time-step. Here, we explicitly follow the evolution of five species, i.e. the different forms of hydrogen (ionized, \( \text{H}^+ \), atomic, \( \text{H} \), and molecular, \( \text{H}_2 \)) as well as ionized carbon (\( \text{C}^+ \)) and carbon monoxide (\( \text{CO} \)). Details about the chemical network, e.g. the considered reactions and the employed rate coefficients in the current implementation can be found in Glover et al. (2010) and Walch et al. (2015).

Parameters: the main parameters controlling both the accuracy and the speed of the calculation are the number of pixels per map \( N_{\text{pix}} \) and the opening angle, \( \theta_{\text{lim}} \), with which the tree is walked [see equation (1)]. Both should be varied at the same time. A high number of \( N_{\text{pix}} \) used with a relatively large opening angle will not improve the directional information since the nodes that are mapped into each solid angle will not be opened and thus, a spatial resolution that is sufficient for a fine-grained map cannot be achieved. Therefore we vary both \( N_{\text{pix}} \) and \( \theta_{\text{lim}} \) at the same time.

The number of HEALPIX pixels is directly related to the solid angle of each element on the unit sphere

\[
\Omega_{\text{pix}} = \frac{4\pi}{N_{\text{pix}}} [\text{sr}].
\]

Tests in Section 3.3.1 show, in agreement with Clark et al. (2012), that the code efficiency is optimal if \( \theta_{\text{lim}} \) is approximately the same as the angular size HEALPIX elements, i.e.

\[
\theta_{\text{lim}} = \sqrt{\Omega_{\text{pix}}}.
\]

Therefore, for \( N_{\text{pix}} = 12, 48, \) and 192 pixels we recommend to use \( \theta_{\text{lim}} \approx 1.0, 0.5, 0.25 \).
3 ACCURACY AND PERFORMANCE

Since more computational time is needed to reach higher accuracy when solving numerical problems, accuracy and performance are connected and therefore, these two properties should always be evaluated at the same time. However, they are often highly dependent on the specific type of the problem and finding a test that allows one to objectively measure both accuracy and performance is hard. Another complication is that the tree-solver saves time by using the information from the previous time-step (if ABU is switched on), and thus any realistic estimate of the performance must be measured by running a simulation in which the mass moves in a similar way as in real applications and by integrating the computational time over a number of time-steps. Unfortunately, such simulations are unavoidably too complex to have an analytic solution against which the accuracy could be easily evaluated.

Therefore, we perform two types of tests: static tests that measure accuracy using simple problems and dynamic tests that evaluate accuracy and performance together. The static tests need substantially less CPU time and thus allow for a higher number of parameter sets to be tested. Furthermore, analytic or semi-analytic solutions are known and the results can be compared to them. On the other hand, the dynamic tests represent more complex simulations which are more similar to problems that one would actually want to solve with the presented code. They also show how well the tree-solver is coupled with the hydrodynamic evolution (where we use the standard piecewise parabolic method (PPM) Riemann solver of the FLASH code) and how the error accumulates during the evolution. In this section, we describe four static and two dynamic tests of the GRAVITY module and one test of the OPTICALDEPTH module.

When possible, i.e. for fully periodic of fully isolated BCs, we compare the results obtained with the new tree-solver to the results obtained with the default multigrid Poisson solver of FLASH (Ricker 2008). The multigrid solver is an iterative solver and the accuracy is controlled by checking the convergence of the L2 norm of the Poisson equation residual $R(r) \equiv 4\pi G\rho(r) − \nabla\Phi(r)$. The iteration process is stopped when $||R_n||/||R_{n-1}|| < \epsilon_{\text{mg, lim}}$, where $||R_n||$ is the residual norm in the $n$th iteration and $\epsilon_{\text{mg, lim}}$ is the limit set by user. If isolated BCs are used, the gravitational potential at the boundary is calculated by a multipole Poisson solver expanding the density and potential field into a series up to a multipole of order $m_{\text{mp}}$. By default $m_{\text{mp}} = 0$ in FLASH version 4.4. However, using this value we found unexpectedly high errors close the boundaries (see test Section 3.1.1 and Figs 4 and 5), and therefore we use $m_{\text{mp}} = 15$ (the highest value allowed for technical reasons) in most tests because it yields the smallest error.

Figure 4. Error in the gravitational acceleration for the Bonnor–Ebert sphere as a function of radius. At a given radius, $r$, the error $\epsilon_{\text{a, r}}$ is calculated as a maximum over all angular directions $\phi$ and $\theta$. The vertical black line shows the BE sphere edge. The solid black line shows the difference between the acceleration obtained analytically and the reference solution calculated using the $N^2$ summation. Left-hand panel shows tests where the acceleration was calculated directly using equation (8), the green, blue, and red lines show errors of runs (a), (b), and (c), respectively, with parameters given in Table 1. Right-hand panel displays tests where the tree-solver calculates the gravitational potential using equation (7) and the acceleration is obtained by numerical differentiation. The green, blue, and red lines denote models (d), (e), and (f). The magenta lines show tests calculated with the multigrid solver using $m_{\text{mp}} = 0$ (dashed) and $m_{\text{mp}} = 15$ (dotted), respectively.

Figure 5. Error in the gravitational acceleration, $\epsilon_{\text{a, r}}$, displayed in the $z = 0$ plane for the Bonnor–Ebert sphere test. The four panels show four selected runs with parameters given in Table 1: top left corresponds to model (b) using the tree-solver calculating the gravitational acceleration directly; top right shows model (e) where the tree-solver calculated the potential; bottom left is model (g) calculated using the multigrid solver with $m_{\text{mp}} = 0$; and bottom right is model (h) calculated using the multigrid solver with $m_{\text{mp}} = 15$. The grid geometry (borders of $8^3$ blocks) is shown in the top right panel.
In general, the calculated gravitational acceleration deviates from the exact analytical solution due to two effects. The first one is the inherent inaccuracy of the gravity solver (either the tree gravity solver or the multigrid solver), and the second one is caused by an imperfect discretization of the density field on the grid. Since we are mainly interested in evaluating the first effect, we measure the error by comparing the calculated accelerations to the reference solution obtained by direct \( N^2 \) summation of all interactions of each grid cell with all the other grid cells in the computational domain. We additionally give the difference between the analytical and the \( N^2 \)-integrated acceleration when possible.

We define the relative error \( e_a \) of the gravitational acceleration \( a \) at the point \( r \) as

\[
e_a(r) = \frac{|a(r) - a_{\text{ref}}(r)|}{a_{\text{ref, max}}},
\]

where \( a_{\text{ref}} \) is the acceleration of the reference solution and \( a_{\text{ref, max}} \) is its maximum taken over the whole computational domain.

In most of the gravity module tests, we control the error by setting the absolute limit \( a_{\text{lim}} \) on the acceleration, which is calculated from the initial maximum acceleration in the computational domain, \( a_{\text{max}} \), as \( a_{\text{lim}} = e_{\text{lim}} \times a_{\text{max}} \); typically, \( e_{\text{lim}} = 10^{-2} \) or \( 10^{-3} \). The difference between using the absolute or the relative error control is discussed in Section 3.4.

Most of the tests were carried out on cluster Salomon of the Czech National Supercomputing Centre IT4I \(^5\). A few static tests that do not need larger computational power have been run on a workstation equipped with a 4-core Intel Core i7-2600 processor.

### 3.1 Static tests of gravity module

In order to test all combinations of the BCs implemented in the GRAVITY module, we present four static tests. A marginally stable Bonnor–Ebert sphere is used to test the code with isolated BCs (see Section 3.1.1) and a density field perturbed by a sine wave not aligned with any coordinate axis is used to test setups with fully periodic BCs (Section 3.1.2). For mixed BCs, periodic in two directions and isolated in a third one, or periodic in a single direction and isolated in the remaining two, we use an isothermal layer in hydrostatic equilibrium (Section 3.1.3) and an isothermal cylinder in hydrostatic equilibrium, respectively (Section 3.1.4). Finally, in Section 3.1.5, we test how the code accuracy depends on the alignment or non-alignment of the gas structures with the grid axes using a set of parallel cylinders lying in the \( xy \)-plane inclined at various angles with respect to the \( x \)-axis.

#### 3.1.1 Bonnor–Ebert sphere

We calculate the radial gravitational acceleration of a marginally stable Bonnor–Ebert sphere (Ebert 1955; Bonnor 1956, BES) with mass \( M_{\text{BE}} = 1 \, M_\odot \), temperature \( T_{\text{BE}} = 10 \, K \) and dimensionless radius \( \xi = 6 \). The resulting BES radius is \( R_{\text{BE}} = 0.043 \, \text{pc} \) and the central density is \( \rho_0 = 1.0 \times 10^{-18} \, \text{g cm}^{-3} \). The sphere is embedded in a warm raredfied medium with temperature \( T_{\text{amb}} = 10^3 \, K \) and density \( \rho_{\text{amb}} = 8.5 \times 10^{-23} \, \text{g cm}^{-3} \), which ensures that the gas pressure across the BES edge is continuous. We use an AMR grid

\( ^4 \) Note that \( e_{\text{lim}} \) is only a device to set \( a_{\text{lim}} \) and it differs from the code parameter \( e_{\text{lim}} \), which sets the limit on the acceleration error ‘on-the-fly’ with respect to the previous time-step acceleration.

\( ^5 \) http://www.it4i.cz/?lang=en

<table>
<thead>
<tr>
<th>Model</th>
<th>solver</th>
<th>quan.</th>
<th>MAC</th>
<th>( \epsilon_{\text{lim}} )</th>
<th>( \theta_{\text{lim}} )</th>
<th>( m_{\text{mp}} )</th>
<th>( e_{a, \text{max}} )</th>
<th>( t_{\text{grv}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>tree</td>
<td>accel.</td>
<td>APE</td>
<td>( 10^{-3} )</td>
<td>–</td>
<td>–</td>
<td>0.0009</td>
<td>83</td>
</tr>
<tr>
<td>(b)</td>
<td>tree</td>
<td>accel.</td>
<td>APE</td>
<td>( 10^{-2} )</td>
<td>–</td>
<td>–</td>
<td>0.0057</td>
<td>35</td>
</tr>
<tr>
<td>(c)</td>
<td>tree</td>
<td>accel.</td>
<td>BH</td>
<td>–</td>
<td>0.5</td>
<td>–</td>
<td>0.0008</td>
<td>110</td>
</tr>
<tr>
<td>(d)</td>
<td>tree</td>
<td>pot.</td>
<td>APE</td>
<td>( 10^{-3} )</td>
<td>–</td>
<td>–</td>
<td>0.0085</td>
<td>80</td>
</tr>
<tr>
<td>(e)</td>
<td>tree</td>
<td>pot.</td>
<td>APE</td>
<td>( 10^{-2} )</td>
<td>–</td>
<td>–</td>
<td>0.0331</td>
<td>38</td>
</tr>
<tr>
<td>(f)</td>
<td>tree</td>
<td>pot.</td>
<td>BH</td>
<td>–</td>
<td>0.5</td>
<td>–</td>
<td>0.0095</td>
<td>106</td>
</tr>
<tr>
<td>(g)</td>
<td>mg</td>
<td>pot.</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0</td>
<td>0.058</td>
<td>21</td>
</tr>
<tr>
<td>(h)</td>
<td>mg</td>
<td>pot.</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>15</td>
<td>0.077</td>
<td>20</td>
</tr>
</tbody>
</table>

Notes. We give the model name in column 1. The following columns are:

(i) solver: indicates whether the tree-solver or the multigrid solver (mg) is used

(ii) quan.: quantity calculated by the gravity solver (acceleration or potential which is then differentiated)

(iii) MAC: MultiPole Acceptance Criterion (Barnes–Hut or APE)

(iv) \( \epsilon_{\text{lim}} \): requested accuracy of the solver as given by equation (4)

(v) \( \theta_{\text{lim}} \): maximum opening angle when the Barnes–Hut MAC is used

(vi) \( e_{a, \text{max}} \): maximum relative error in the computational domain given by equation (29)

(vii) \( t_{\text{grv}} \): time (in seconds) to calculate a single time-step on eight cores.

controlled by the Jeans criterion – the Jeans length has to be resolved by at least by 64 cells and at most by 128 cells. It results in an effective resolution of \( 512^3 \) in the centre of the BES.

Fig. 4 shows the relative error in the gravitational acceleration, \( e_{a, r} \), as a function of radial coordinate, \( r \), and Table 1 lists all models, their maximum relative error, \( e_{a, \text{max}} \), and the time to calculate one time-step, \( t_{\text{grv}} \). We compare the solutions calculated with the tree gravity solver using the geometric (BH) MAC with \( \theta_{\text{lim}} = 0.5 \) (red curves) to the ones calculated using the APE MAC with \( \epsilon_{\text{lim}} = 10^{-2} \) (green lines) and \( \epsilon_{\text{lim}} = 10^{-3} \) (blue lines), respectively. The APE MAC and \( \epsilon_{\text{lim}} = 10^{-3} \) as well as the geometric MAC with \( \theta_{\text{lim}} = 0.5 \) always give a maximum relative error which is smaller than 0.1 per cent. In case of the APE MAC and \( \epsilon_{\text{lim}} = 10^{-2} \), the maximum relative error reaches \( \sim 1 \) per cent. Note that the error due to the discretization of the density field is also of the order of 1 per cent (black line); the jumps are due to changes in the refinement level in the AMR grid.

With the tree gravity solver, the user may choose to directly compute the gravitational accelerations (left-hand panel of Fig. 4) or to calculate them by numerical differentiation of the gravitational potential (right-hand panel of Fig. 4). Usually, the latter is the standard practice in grid-based 3D simulations, also because only one field variable, the potential, has to be stored instead of three, the accelerations in three spatial directions. However, for the tree-solver we generally find that the error in the gravitational accelerations is significantly smaller (about a factor of 10 in the test presented here) if they are computed directly. This is independent of the used MAC.

For comparison, we also show the results obtained with the multigrid solver (magenta lines) using \( \epsilon_{\text{mg, lim}} = 10^{-6} \) and \( m_{\text{mp}} = 0 \) (solid lines) or \( m_{\text{mp}} = 15 \) (dotted lines), respectively. Although the mass distribution is spherically symmetric, the order of the multipole expansion of the BC affects the accuracy of the multigrid solver relatively far away from boundaries, even inside the BES. The error of the multigrid solver is very low in the central region, it reaches \( \sim 1 \) per cent in regions where the refinement level changes (due to numerical differentiation of the potential), and increases to relatively high values at the border of the computational domain (\( \sim 1 \) per cent.)
for \( m_{\text{mp}} = 15 \) and \( \sim 5 \) per cent for \( m_{\text{mp}} = 0 \), due to inaccuracy of the BCs calculated by the multipole solver. We note that a direct calculation of the gravitational acceleration is not possible with the multigrid solver. The distribution of the relative error \( e_r \), in the \( z = 0 \) plane through the centre of the BES is depicted in Fig. 5. The results show that the acceleration obtained with the tree gravity solver using the APE MAC with \( \epsilon_{\text{lim}} = 10^{-2} \) has a substantially smaller error if it is calculated directly [top left panel; see Table 1, model (b)] instead of by numerical differentiation of the potential [top right panel; model (e)]. The bottom panels show the results for the multigrid solver with \( m_{\text{mp}} = 0 \) [model (g)] and \( m_{\text{mp}} = 15 \) [model (h)], respectively. The default setting of \( m_{\text{mp}} = 0 \) gives errors of \( \sim 5 \) per cent near the domain boundaries due to the low accuracy of the multipole solver. This error propagates into a large fraction of the computational domain.

### 3.1.2 Sine-wave perturbation (Jeans test)

In a computational domain with fully periodic BCs, we calculate the gravitational acceleration of a smooth density field with a harmonic perturbation,

\[
\rho(r) = \rho_0 + \rho_1 \cos(k \cdot r),
\]

where \( \rho_0 = 1.66 \times 10^{-24} \) is the mean density and \( \rho_1 = 0.99 \rho_0 \) is the amplitude of the perturbation. The computational domain is a cube of size 500 pc with 128 grid cells in each direction. The wave vector \( \mathbf{k} = 6\pi n(3, 2, 1)/L \) was chosen such that it is not aligned with any of the coordinate axes. The gravitational acceleration can be obtained analytically with the help of the Jeans swindle (Jeans 1902; Kiessling 1999)

\[
g(r) = -4\pi G \rho_1 k^2 \sin(k \cdot r). \tag{31}
\]

Fig. 6 shows the maximum relative error \( e_a, k \) as a function of the position \( x_0 \) on a line parallel to the perturbation wave vector \( k \). The maximum error is computed from all points projected to a given position on the line. It can be seen that the error of the multigrid solver (magenta curve) is very small, almost the same as the difference between the analytical solution and the reference solution (black line). This is because without the need to calculate the BCs separately, and on a uniform grid, the fast Fourier transform (FFT) accelerated multigrid method is extremely efficient. Again, the results for the tree-solver simulations show that direct calculation of the acceleration (solid curves) leads to a much lower error than the calculation of the potential and subsequent differentiation (dashed lines). In particular, the calculation of the potential with the geometric MAC that does not take into account the different mass density in the tree-nodes leads to a relative error greater than 10 percent. However, a direct calculation of the acceleration gives very accurate results for both, the geometric MAC and the APE MAC with \( \epsilon_{\text{lim}} = 10^{-3} \). In Table 2, we list all models with their respective \( e_{a, \text{max}} \) and \( t_{\text{ggv}} \).

#### Table 2. Results of the second static test: sine-wave perturbation. The meaning of the columns is the same as in Table 1.

<table>
<thead>
<tr>
<th>Model</th>
<th>solver</th>
<th>quan.</th>
<th>MAC</th>
<th>( \epsilon_{\text{lim}} )</th>
<th>( \theta_{\text{lim}} )</th>
<th>( e_{a, \text{max}} )</th>
<th>( t_{\text{ggv}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>tree</td>
<td>accel.</td>
<td>APE</td>
<td>( 10^{-3} )</td>
<td>-</td>
<td>0.0009</td>
<td>480</td>
</tr>
<tr>
<td>(b)</td>
<td>tree</td>
<td>accel.</td>
<td>APE</td>
<td>( 10^{-2} )</td>
<td>-</td>
<td>0.0062</td>
<td>210</td>
</tr>
<tr>
<td>(c)</td>
<td>tree</td>
<td>accel.</td>
<td>BH</td>
<td>-</td>
<td>0.5</td>
<td>0.0029</td>
<td>250</td>
</tr>
<tr>
<td>(d)</td>
<td>tree</td>
<td>pot.</td>
<td>APE</td>
<td>( 10^{-3} )</td>
<td>-</td>
<td>0.0180</td>
<td>330</td>
</tr>
<tr>
<td>(e)</td>
<td>tree</td>
<td>pot.</td>
<td>APE</td>
<td>( 10^{-2} )</td>
<td>-</td>
<td>0.0270</td>
<td>130</td>
</tr>
<tr>
<td>(f)</td>
<td>tree</td>
<td>pot.</td>
<td>BH</td>
<td>-</td>
<td>0.5</td>
<td>0.15</td>
<td>150</td>
</tr>
<tr>
<td>(g)</td>
<td>mg</td>
<td>pot.</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0.0016</td>
<td>9</td>
</tr>
</tbody>
</table>

3.1.3 Isothermal layer in hydrostatic equilibrium

In order to test the accuracy of the tree gravity module with mixed BCs (periodic in two directions and isolated in the third one), we calculate the gravitational acceleration of an isothermal layer in hydrostatic equilibrium. The vertical density distribution of the layer is (Spitzer 1942)

\[
\rho(z) = \rho_0 \operatorname{sech}^2 \left( \frac{2\pi G \rho_0 z}{c_s} \right), \tag{32}
\]

where \( \rho_0 = 1.6 \times 10^{-24} \) g cm\(^{-3} \) is the mid-plane density and \( c_s = 11.7 \) km s\(^{-1} \) is the isothermal sound speed. The corresponding vertical component of the gravitational acceleration is

\[
g_z(z) = \frac{2}{2\pi G \rho_0 c_s^2} \tanh \left( \frac{2\pi G \rho_0 z}{c_s^2} \right). \tag{33}
\]

The computational domain is a cube of side length \( L = 1000 \) pc and a uniform resolution of 128 grid cells in each direction.

Fig. 7 shows the maximum relative error \( e_{a, z} \) in the acceleration as a function of the \( z \)-coordinate, where the maximum is taken over all cells with the same \( z \)-coordinate. It can be seen that the error is almost independent of \( z \) and there is only a small difference between the cases where the gravitational acceleration is calculated directly (solid lines) or where it is obtained by differentiation of the potential (dashed lines). The reason is that the density field in this test has relatively shallow gradients (e.g. compared to the Jeans test discussed in the previous section) and numerical differentiation leads to particularly severe errors for steep gradients. We find the largest error for runs with APE MAC and \( \epsilon_{\text{lim}} = 10^{-2} \). All other runs have small errors, which are comparable to the difference between the analytical and the reference solution, resulting from the discretization of the density field. The results are summarized in Table 3.
and contains 256 × 128 grid cells. If numerical differentiation of the potential is used, it is the dominant source of the error, which is as large as 1 percent in these cases (see dashed lines). The results are summarized in Table 4.

### 3.1.4 Isothermal cylinder in hydrostatic equilibrium

In the next static test, we evaluate the accuracy of the tree gravity module for mixed BCs, which are isolated in two directions and periodic in the third one. We calculate the gravitational acceleration of an isothermal cylinder in hydrostatic equilibrium. The long axis of the cylinder is parallel to the x-coordinate and the radius is given as \( R = \sqrt{y^2 + z^2} \). The density distribution is

\[
\rho(R) = \rho_0 \left( 1 + \frac{\pi G \rho_0 R^2}{2 c_s^2} \right)^{-2}
\]

where \( \rho_0 = 3.69 \times 10^{-23} \text{ g cm}^{-3} \) is the central density and \( c_s = 0.2 \text{ km s}^{-1} \) is the isothermal sound speed. The density distribution is cut-off at radius \( R_{cyl} = 1.62 \text{ pc} \) and embedded in an ambient gas with \( c_{s,\text{amb}} = 10 \text{ km s}^{-1} \) and the same pressure as the pressure at the cylinder boundary. The corresponding gravitational acceleration is

\[
g(R) = 2 \pi G \rho_0 R \left( 1 + \frac{\pi G \rho_0 R^2}{2 c_s^2} \right)^{-1}. \tag{35}
\]

The computational domain has dimensions 3.6pc × 1.8pc × 1.8pc and contains 256 × 128 × 128 grid cells.

Fig. 8 shows the maximum relative error \( \epsilon_{a_{xy}} \) of the gravitational acceleration in radial direction, where the maximum error is calculated for all grid cells at the same distance \( R \) to the cylinder axis. In all runs, the error is a very weak function of \( R \). If numerical differentiation of the potential is used, it is the dominant source of the error, which is as large as 1 percent in these cases (see dashed lines). The results are summarized in Table 4.

#### 3.1.5 Inclined cylinders

In order to test whether the alignment of gas structures with the coordinate axes has an impact on the code accuracy, i.e. whether the algorithm is sensitive to any grid effects, we calculate gravitational field of the set of parallel cylinders in the 2P1I geometry. The axes of all cylinders lie in the xy-plane and they are inclined at angle \( \beta_{\text{incl}} \) with respect to the x-axis. The computational domain has an extent 48 pc in the isolated z-direction and approximately 16 pc in the periodic x- and y-directions. The exact extents in the latter two directions are chosen so that the computational domain composes a periodic cell of the infinite plane of cylinders, i.e. the cylinders connect contiguously to each other at the x and y periodic boundaries. Each cylinder has the same radius and density profile as the cylinder described in Section 3.1.4, the distance between the cylinder axes is 4 pc. We have calculated seven models with \( \beta_{\text{incl}} \) increasing from 0° to 90° with a step 15°. For all models, the gravity tree-solver was running with the BH MAC and maximum opening angle \( \theta_{\text{lim}} = 0.5 \).

Fig. 9 shows the relative error of the gravitational acceleration, \( \epsilon_{a_{xy}} \), in the xy-plane using equation (29). The reference acceleration, \( a_{\text{ref}} \), is either obtained numerically by the \( N^2 \)-integration (four panels on the left for \( \beta_{\text{incl}} = 0°–45° \)), or analytically by summing up potential of 1000 parallel cylinders (four panels on the right). The error with respect to the \( N^2 \)-integration is always smaller than 1 percent. The error with respect to the analytical integration is of order 1 percent and is always slightly higher than the former error, as it includes contribution from the imperfect discretization of the density field reaching the highest values along the cylinder edges where the density field has a discontinuity. The bottom panel show the maximum \( \epsilon_{a_{xy}} \) as a function of \( \beta_{\text{incl}} \).
Figure 9. Relative error of the gravitational acceleration in the $xy$-plane, $e_{a,xy}$, for the set of inclined cylinders. Left-hand panels show the logarithm of the error measured with respect to the direct $N^2$ integration, and right-hand panels show the error with respect to analytically obtained accelerations. Each of the top four rows show the calculation with different inclination angle of the cylinders: $0^\circ$, $15^\circ$, $30^\circ$, and $45^\circ$ from top to bottom. The panel at the very bottom shows the logarithm of the maximum error in the acceleration $e_{a,xy}$ as a function of the cylinder inclination angle, $\beta_{incl}$.

demonstrating that the code accuracy is almost independent of the inclination of the gaseous structures with respect to coordinate axes.

3.2 Dynamic tests of gravity module

We run two dynamic tests of the gravity module. The first one (described in Section 3.2.1) is a collapse of a cold adiabatic sphere suggested by Evrard (1988) and it tests how well the energy is conserved during the gravitational collapse. The second one, describes the evolution of a turbulent sphere (Section 3.2.2). Both test the accuracy of the gravity module and its coupling to the hydrodynamic solver.

3.2.1 Evrard test

The Evrard test (Evrard 1988) describes the gravitational collapse and a subsequent re-bounce of an adiabatic, initially cold sphere. It is often used to verify energy conservation in smoothed particle hydrodynamics (SPH) codes (e.g. Springel et al. 2001; Wetzstein et al. 2009), its application on grid-based codes is unfortunately less common. The initial conditions consist of a gaseous sphere of mass $M$, radius $R$, and density profile

$$\rho(r) = \frac{M}{2\pi R^2 r}.$$  \hspace{1cm} (36)

The initial, spatially constant temperature is set so that the internal energy per unit mass is

$$u = 0.05 \frac{GM}{R}.$$ \hspace{1cm} (37)

where $G$ is the gravitational constant. The standard values of the above parameters, used also in this work, are $M = R = G = 1$.

In Fig. 10, we show the time evolution of the total mass as well as the gravitational, kinetic, internal, and total energy. On the top panel, we compare the results obtained with the tree gravity solver and the multigrid solver, both computed on a uniform grid of size $128^3$ corresponding to a constant refinement level equal to 5. The tree-solver run uses the Barnes–Hut MAC with $\theta_{lim} = 0.5$, the multigrid run was calculated with the default accuracy $\epsilon_{mg, lim} = 10^{-6}$ and $m_{mp} = 0$. The two runs are practically indistinguishable, however, the total energy (that should stay constant) rises by approximately 0.1 during the period of maximum compression. Since the distribution of the error in the gravitational acceleration calculated by the two solvers is very different, the same results indicate that the error in the energy conservation is not caused by the calculation of the gravitational acceleration and that the acceleration errors are below the sensitivity of this test.

The bottom panel of Fig. 10 compares runs calculated with the tree-solver at different resolutions. It includes three runs with uniform grids of sizes $64^3$, $128^3$, and $256^3$ (corresponding to constant refinement levels of 4, 5, and 6) and three runs calculated on adaptive grids, which are refined such that the Jeans length is resolved by at least 2, 4, and 8 grid cells, respectively.

We find that low resolution leads to a higher numerical dissipation and artificial heating of the gas. Furthermore, lower resolution does not allow high compression of the sphere centre leading to less pronounced peaks of the internal and gravitational energies. Consequently, the results of this test show that high resolution is needed only in the centre of the sphere where the highest density is reached.
and the gravitational forces among all sink particles and between the particles and the gas are computed by direct summation. They are evolved using a Leapfrog integrator.

In Fig. 11, we show the evolving column density in the $xy$-plane at times 0.2, 2.0, 4.0, and 6.0 Myr. Although the simulation is interesting in itself, we only focus on the error in the resulting gravitational acceleration. Therefore, we compute the same initial conditions six times with different gravity settings and measure the resulting error of the gravitational acceleration, where the supposedly accurate result compared to which we calculate the error is obtained using $N^2$ integration. The results of our analysis are shown in Fig. 12, which depicts the error in the $xy$-plane at $\tau = 2$ Myr. The maximum and average errors $\epsilon_{a,\text{max}}$ and $\epsilon_{a,\text{avg}}$, respectively, and mean times per gravity and hydrodynamic time-step computations $t_{\text{grv}}$ and $t_{\text{hydro}}$, respectively, are given in Table 5. The runs are also shown in the $t_{\text{grv}}$-$\epsilon_{a,\text{max}}$ plane in Fig. 13.

The top two panels of Fig. 12 show calculations with the tree-solver calculating directly the gravitational acceleration using the geometric MAC with $\theta_{\text{lim}} = 0.5$. The left-hand panel (Fig. 12a) was calculated without the ABU off. The relative error is very small everywhere, with sudden changes at constant distances from massive concentrations of gas, resulting from switching tree-node sizes as prescribed by the geometric MAC criterion. The maximum error is approximately 2 per cent, the average error is even an order of magnitude smaller. One iteration of the tree-solver took approximately 20 s, i.e. it was the slowest run. The right-hand panel (Fig. 12b) shows the same calculation, but the ABU was switched on in this case. The relative error exhibits a rectangular pattern, because some blocks, in particular in the outer regions, were not updated in a given time-step and the error in them is larger. The maximum error is approximately 3 per cent, i.e. 1.5 times more than in the run with ABU off, and at the same time, the ABU makes the calculation approximately two times faster.

Panel in Fig. 12(c) shows a run with the tree-solver using the APE MAC with $\epsilon_{\text{lim}} = 10^{-2}$. The results are very similar to the one in run (Fig. 12b), with a maximum relative error of approximately 4.5 per cent (~1.5 larger) and the mean time per gravity time-step is 7 s (slightly smaller). Panel (Fig. 12d) shows the run with the same tree-solver parameters, but instead of calculating the acceleration
Figure 12. Error in the gravitational acceleration in the xy-plane of the turbulent sphere at \( t = 2 \) Myr. The panels show: (a) tree gravity solver calculating the acceleration with BH MAC, \( \theta_{\text{lim}} = 0.5 \), and ABU switched off, (b) tree gravity solver calculating the acceleration with BH MAC, \( \theta_{\text{lim}} = 0.5 \), and ABU on, (c) tree gravity solver calculating the acceleration with APE MAC, \( \epsilon_{\text{lim}} = 10^{-2} \) and ABU on, (d) tree gravity solver calculating the potential with APE MAC, \( \epsilon_{\text{lim}} = 10^{-2} \) and ABU on, (e) tree gravity solver calculating the potential with APE MAC, \( \epsilon_{\text{lim}} = 10^{-1} \) and ABU on, and (f) multigrid solver calculating the potential with \( \epsilon_{\text{mg, lim}} = 10^{-6} \) and \( m_{\text{mp}} = 10 \).

Figure 13. Results of the turbulent sphere plotted in the plane of the gravity calculation duration in seconds, \( t_{\text{grv}} \) (x-axis) versus the maximum relative error in the gravitational acceleration, \( e_{\text{a, max}} \) (y-axis). The error is determined at \( t = 2 \) Myr and the maximum is taken over the whole computational domain. The thin dashed lines are isolines of constant \( t_{\text{grv}} \times e_{\text{a, max}} \) assessing the code efficiency. Parameters of the displayed runs are given in Table 5.

directly, the tree-solver calculates the potential and differentiates it numerically. The relative error exhibits a similar pattern to run (a), however, instead of sudden changes it includes high peaks of the error resulting from a numerical differentiation. Even though the mean error is comparable to runs (b) and (c), the maximum error is much higher, reaching 80 per cent. The time of the gravity calculation is slightly higher than in run (c), even though calculating the potential is cheaper than the acceleration for a single target cell.

Panel (Fig. 12e) includes the run with a reduced accuracy of \( \epsilon_{\text{lim}} = 10^{-1} \) made to test the limits of the tree-solver usability. The relative error is high, in particular in the outer regions where the blocks are updated less often, reaching a maximum of 20 per cent, however, it is still a factor of 2 smaller than the error at the boundaries of the computational domain found in the run with the multigrid solver (see below). On the other hand, the calculation is very

Table 5. Accuracy and performance of the turbulent sphere test.

<table>
<thead>
<tr>
<th>Model</th>
<th>solver</th>
<th>quan.</th>
<th>MAC</th>
<th>ABU</th>
<th>( \epsilon_{\text{lim}} )</th>
<th>( \theta_{\text{lim}} )</th>
<th>( e_{\text{a, max}} )</th>
<th>( e_{\text{a, avg}} )</th>
<th>( t_{\text{grv}} )</th>
<th>( t_{\text{hydro}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>tree</td>
<td>accel.</td>
<td>BH</td>
<td>off</td>
<td>–</td>
<td>0.5</td>
<td>0.021</td>
<td>0.0020</td>
<td>19.9</td>
<td>7.5</td>
</tr>
<tr>
<td>(b)</td>
<td>tree</td>
<td>accel.</td>
<td>BH</td>
<td>on</td>
<td>–</td>
<td>0.5</td>
<td>0.032</td>
<td>0.0061</td>
<td>9.7</td>
<td>5.3</td>
</tr>
<tr>
<td>(c)</td>
<td>tree</td>
<td>accel.</td>
<td>APE</td>
<td>on</td>
<td>( 10^{-2} )</td>
<td>–</td>
<td>0.045</td>
<td>0.0056</td>
<td>7.0</td>
<td>4.5</td>
</tr>
<tr>
<td>(d)</td>
<td>tree</td>
<td>pot.</td>
<td>APE</td>
<td>on</td>
<td>( 10^{-2} )</td>
<td>–</td>
<td>0.801</td>
<td>0.0062</td>
<td>7.7</td>
<td>4.0</td>
</tr>
<tr>
<td>(e)</td>
<td>tree</td>
<td>accel.</td>
<td>APE</td>
<td>on</td>
<td>( 10^{-1} )</td>
<td>–</td>
<td>0.200</td>
<td>0.0472</td>
<td>2.5</td>
<td>4.8</td>
</tr>
<tr>
<td>(f)</td>
<td>mg</td>
<td>pot.</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>0.416</td>
<td>0.0447</td>
<td>12.2</td>
<td>4.7</td>
<td></td>
</tr>
</tbody>
</table>

Notes. Column 1 gives the model name. The following columns list:

(i) solver: indicates whether the tree-solver or the multigrid solver (mg) is used
(ii) quan.: quantity calculated by the gravity solver (acceleration or potential which is then differentiated)
(iii) MAC: Multipole Acceptance Criterion (Barnes–Hut or APE)
(iv) ABU: Adaptive Block Update (on or off)
(v) \( \epsilon_{\text{lim}} \): requested accuracy of the solver as given by equation (4) \( \epsilon_{\text{lim}} = \epsilon_{\text{lim}} \times a_{\text{max}} \) where \( a_{\text{max}} \) is the maximum gravitational acceleration in the domain
(vi) \( \theta_{\text{lim}} \): maximum opening angle when the Barnes–Hut MAC is used
(vii) \( e_{\text{a, max}} \): maximum relative error in the computational domain given by equation (29) measured at \( t = 2 \) Myr
(viii) \( e_{\text{a, avg}} \): average relative error in the computational domain given by equation (29) measured at \( t = 2 \) Myr
(ix) \( t_{\text{grv}} \): time per time-step (in seconds) to calculate the gravitational acceleration on 96 cores
(x) \( t_{\text{hydro}} \): time per time-step spent in the hydrodynamic solver on 96 cores.
Figure 14. Number of updated blocks, i.e. blocks for which the tree-walk was executed for all grid cells in a given time-step, as a function of time (top x-axis) or time-step number (bottom x-axis). The figure shows first 2 Myr of the evolution of the turbulent sphere test. Individual curves represent models (a)–(e) as given in the legend (see also Table 5 for model parameters). Note that blue and magenta lines [models (b) and (c)] are on the top of each other.

fast with a mean time per gravity time-step of 3.5 s, which is ∼70 per cent of the time needed by the hydro solver.

Panel (Fig. 12f) displays the calculation with the multigrid solver and as in Section 3.1.1 it shows that the largest error (reaching ∼40 per cent) is along the boundaries of the computational domain where the potential is influenced by the boundary values obtained by the multipole expansion. The error in the central region is of the order of several percent, comparable to the runs in panels of Figs 12 (b) and (c). The run with the multigrid solver is 20 per cent–30 per cent slower.

In order to evaluate the efficiency of the ABU, we show in Fig. 14 a number of updated blocks in each time-step as a function of time/time-step number. The red curve corresponds to model (a) where ABU was switched off, i.e. it shows the number of all blocks in the simulation. It grows from 4096 to almost 6000, as the AMR creates more blocks in regions of high density formed by the gravitational collapse. The number of all blocks is the same for all simulations, as they run the identical model. For model (b) (blue curve), the number of updated blocks stays very small for the first ∼10 time-steps, because the initial time-step is very low and the density and gravitational acceleration fields almost do not change. As soon as the time-step reaches a value given by the CFL condition, the number of updated blocks quickly rises up to 1000 and then it increases slowly to almost 2000 at 2 Myr. Throughout the evolution, the number of updated blocks is approximately three times lower than in model (a) with ABU off. As a result, model (b) runs more than twice as fast as model (a) and the maximum error is ∼1.5 larger.

Model (c) with larger error limit updates less than 10 per cent of blocks in each time-step and as a result it runs 8 times faster than model (a) and its maximum error is almost 10× larger. Model (d) calculating the potential instead of the acceleration behaves in a different way. The number of updated blocks exceeds 3000 shortly after the start of the simulation, their fraction stays above 50 per cent and reaches 100 per cent in the last quarter of the time. It is because the numerical differentiation of the potential at the border between updated and not-updated blocks tends to give high error in the acceleration. Therefore, we do not recommend the use of ABU together with calculating the potential.

Note that the efficiency of the ABU test is highly problem dependent. In this regard, the used turbulent sphere setup is a relatively hard one, because the sphere quickly forms dense filaments with large-density gradients and they move supersonically as the whole structure collapses (i.e. the time-step is given mainly by the gas velocity, not the sound speed). On the other hand, there are still regions where the gravitational acceleration changes slowly, e.g. in the computational domain corners, and these regions can be updated less often making the ABU inefficient. If the volume with fast moving dense objects is larger, the ABU can be less efficient and vice versa.

3.3 Test of the optical depth module

In order to evaluate the accuracy of the optical depth module, we perform two tests. For both of them, we repeat the calculation of the turbulent sphere described in Section 3.2.2, using an adiabatic equation of state with γ = 5/3 (instead of the isothermal equation of state used previously). Additionally, we switch on the optical depth and chemistry modules calculating the gas cooling and heating, and the mass fractions of various species. The sphere is heated from the outside assuming a typical ISRF of strength $G_0 = 1.7$ times the Habing field. This causes the low ambient density gas to heat up to a few $10^4$ K, while the interior of the sphere is cold and thus it collapses to form stars as in runs with the isothermal equation of state in Section 3.2.2. A detailed description of the chemical network, the heating and cooling processes it includes, the dust temperature it calculates, and how the optical depth module is coupled to it can be found in Walch et al. (2015). Here, we are only concerned with the workings of the optical depth module and with the column density (or optical depth) it delivers.

In the first test, we evaluate how accurately the optical depth module determines the column density depending on the chosen angular resolution; and in the second test, we compare the resulting optical depth with the optical depth computed using the Monte Carlo radiative transfer code RADMC-3D.6

3.3.1 Column density with increasing $N_{\text{pix}}$

We perform a test similar to the one by Clark et al. (2012, their section 3.2), and calculate the ‘sky map’ of hydrogen column density, $N_H$, as seen from the centre of the computational domain, for the turbulent sphere simulation at time $t = 2$ Myr (see the top right panel of Fig. 11). The hydrogen column density determined by the optical depth module, $N_{\text{Pix}}$, is compared to the ‘actual’ reference hydrogen column density, $N_H(\theta, \phi)$, obtained using a direct integration over individual grid cells of the simulation and very high HEALPIX resolution $N_{\text{Pix}}$ = 3072. The angular resolution of the optical depth module is controlled by two parameters: number of HEALPIX elements $N_{\text{pix}}$ and tree maximum opening angle $\theta_{\text{min}}$ determining the maximum angular size of tree-nodes. We calculate five models with $N_{\text{pix}} = 12, 48, 192$, and two maximum opening angles $\theta_{\text{min}} = 0.5$ and 0.25 (see Table 6). We define a relative error in the hydrogen column density

$$e_{N_{\text{Pix}}} = \frac{|N_{\text{Pix}} - (N_H(\theta, \phi))_{\text{Pix}}|}{(N_H(\theta, \phi))_{\text{Pix}}}$$

(38)

6 See http://www.ita.uni-heidelberg.de/~dullemond/software/radmc-3d/
Table 6. Results of the \textsc{opticaldepth} module test studying the dependency of the column density accuracy on the resolution.

\begin{tabular}{ccccccc}
Model & \(N_{\text{pix}}\) & \(\theta_{\text{lim}}\) & \(\epsilon_{\text{N_{\text{pix}}},\text{max}}\) & \(\epsilon_{\text{A}_V}\) & \(t_{\text{rec}}\) & \(N\text{versus}H\) \\
\hline
(a) & 12 & 0.5 & 0.16 & 0.19 & 41 & \(N<H\)  \\
(b) & 48 & 0.5 & 0.18 & 0.07 & 44 & \(N<H\)  \\
(c) & 48 & 0.25 & 0.14 & 0.08 & 256 & \(N<H\)  \\
(d) & 192 & 0.5 & 0.48 & 0.01 & 48 & \(N>H\)  \\
(e) & 192 & 0.25 & 0.30 & 0.002 & 286 & \(N>H\)  \\
\hline
\end{tabular}

Notes. We give the model name in column 1. The following columns are:

(i) \(N_{\text{pix}}\): number of \textsc{healpix} pixels corresponding to the angular resolution
(ii) \(\theta_{\text{lim}}\): maximum opening angle (Barnes–Hut MAC is used in all tests)
(iii) \(\epsilon_{\text{N_{\text{pix}}},\text{max}}\): maximum relative error in the hydrogen column density (equation 38); maximum is taken over all \textsc{healpix} pixels
(iv) \(\epsilon_{\text{A}_V}\): relative error in the mean visual extinction (equation 39)
(v) \(t_{\text{rec}}\): time per step (in seconds) spent in the tree-solver on 96 cores
(vi) \(N\text{versus}H\): indicates the relative size of tree-nodes (\(N\)) and \textsc{healpix} elements (\(H\)).

where \(\langle N_{\text{H}}(\theta, \phi) \rangle_{\text{pix}}\) is the mean value of reference hydrogen column density, \(N_{\text{H}}\), in element \(i_{\text{pix}}\). In Table 6, we give for each model the maximum error \(\epsilon_{\text{N_{\text{pix}}},\text{max}} = \max(\epsilon_{\text{N_{\text{pix}}},i_{\text{pix}}})\), where the maximum is taken over the whole sphere.

However, directionally dependent \(N_{\text{H}_{\text{pix}}}\) does not directly enter calculations of the gas-radiation interaction. Instead, the CHEMISTRY module uses quantities averaged over all directions, e.g., the mean visual extinction, \(A_V\), given by equation (26). Therefore, we further define a relative error in the mean visual extinction

\[ \epsilon_{A_V} = \frac{|A_V - A_{V,\text{ref}}|}{A_{V,\text{ref}}} \]  

(39)

where \(A_V\) is the mean visual extinction calculated by the \textsc{opticaldepth} module and \(A_{V,\text{ref}}\) is the reference visual extinction value obtained by averaging the high-resolution reference hydrogen column density \(N_{\text{H}}(\theta, \phi)\) using equation (26). Values of \(\epsilon_{A_V}\) for the five calculated runs are also given in Table 6 and in top right corners of right-hand panels in Fig. 15.

The results are summarized in Fig. 15, showing, in the Hammer projection, the reference hydrogen column density \(N_{\text{H}}(\theta, \phi)\) (top panel), values of \(N_{\text{H}_{\text{pix}}}\) calculated by the \textsc{opticaldepth} module (left-hand panels), and relative errors, \(\epsilon_{\text{N_{\text{pix}}},\text{pix}}\) (right-hand panels). Our findings are generally in agreement with those of Clark et al. (2012). Even run (a) with \(N_{\text{pix}} = 12\) recovers approximately the overall structure of the cloud and results in \(\epsilon_{\text{N_{\text{pix}}},\text{max}} = 0.16\) and \(\epsilon_{A_V} = 0.19\). Increasing the \textsc{healpix} angular resolution to \(N_{\text{pix}} = 48\) [run (b)] with the approximately same size of tree-nodes and \textsc{healpix} elements, we see the last column in Table 6) leads to a smaller error in \(A_V\) while keeping \(\epsilon_{\text{N_{\text{pix}}},\text{max}}\) approximately the same. Since runs (a) and (b) take nearly the same time to calculate, their comparison shows that it is not worth to degrade the \textsc{healpix} resolution (by decreasing \(N_{\text{pix}}\)) below the tree-solver resolution (given by \(\theta_{\text{lim}}\)). Similarly, run (c) with better tree-solver resolution (\(\theta_{\text{lim}} = 0.25\)) and the same \textsc{healpix} resolution results in the approximately same \(\epsilon_{\text{N_{\text{pix}}},\text{max}}\) and \(\epsilon_{A_V}\) as run (b), even though the computational costs are much higher. Run (d) with \textsc{healpix} elements smaller than the tree-node size leads to smaller \(\epsilon_{A_V} = 0.01\), however, \(\epsilon_{\text{N_{\text{pix}}},\text{max}} = 0.48\) is very high. It is because the approximations adopted when the mass of relatively large tree-nodes is distributed to \textsc{healpix} elements sometimes result in the assignment of the mass to a different element. This problem is diminished in run (e), which has again the approximately same angular size of tree-nodes and \textsc{healpix} elements, and in which the visual extinction error drops to a very small value, \(\epsilon_{A_V} = 0.002\) and \(\epsilon_{\text{N_{\text{pix}}},\text{max}}\) also decreases in comparison with run (d), even though it is still higher than in runs (a)–(c).

3.3.2 Comparison to \textsc{radmc-3d}

Here, we compare the spatial distribution of the optical depth, \(\tau_{\text{f}}\), calculated by the \textsc{opticaldepth} module to the optical depth, \(\tau_{\text{R}}\), computed using the \textsc{radmc-3d} code\footnote{Note that the index F in \(\tau_{\text{F}}\) refers to \textsc{flash}, i.e. calculation by the \textsc{opticaldepth} module, and R in \(\tau_{\text{R}}\) refers to \textsc{radmc-3d}.}. We use a snapshot at \(t = 2\) Myr from a turbulent sphere simulation similar to the one discussed in Sections 3.2.2 and 3.3.1, but calculated on a uniform grid 128\(^3\) to make the \textsc{radmc-3d} calculation feasible. We use \(N_{\text{pix}} = 48\) pixels and a geometric MAC with \(\theta_{\text{lim}} = 0.5\) (see Section 2.3 for details on the \textsc{opticaldepth} module). Here, we assume a constant dust-to-gas ratio of 0.01. We select an ultraviolet (UV) wavelength, \(\lambda_0 = 9.36 \times 10^{-2} \text{ \mu m}\), because scattering effects in the UV are minimal, and we can easily relate the dust column density to the optical depth using the dust opacity at this wavelength, \(\kappa_{\text{abs}}(\lambda)\). This approach neglects possible variations along the line of sight, e.g., due to temperature variations or changes in the dust properties. Using a typical Milky Way dust opacity provided by Weingartner & Draine (2001) (table for MW\_R\_V\_4.0), we have \(\kappa_{\text{abs}}(\lambda_0) = 6.555 \times 10^4 \text{ cm}^2 \text{ g}^{-1}\). We obtain

\[ \tau_{\text{F}} = \kappa_{\text{abs}}(\lambda_0) \times N_{\text{dust,F}}. \]  

(40)

Using the dust density field and dust temperature provided by the simulation, we compute the optical depth at the same wavelength using the \textsc{radmc-3d} code, \(\tau_{\text{R}}\). With \textsc{radmc-3d}, it is possible to provide an external radiation field, in which case the photon packages are launched from the borders of the computational domain and pass through the grid in random directions. In each cell, they interact with the present dust according to its opacity. We use the same dust opacity table for \textsc{radmc-3d} as described above. For the incoming radiation, we use the intensities of a typical ISRF as provided by Evans et al. (2001). The incoming intensity at wavelength \(\lambda_0\) is \(I_0 = 9.547 \times 10^{-21} \text{ erg s}^{-1} \text{ cm}^{-2} \text{ Hz}^{-1} \text{ sr}^{-1}\). We run \textsc{radmc-3d} in the mode \textsc{emmode} to compute the intensity field at \(\lambda_0\) in every cell of the computational domain. Then, we convert this intensity, \(I_{\text{cell}}\), to \(\tau_{\text{R}}\) using

\[ \tau_{\text{R}} = \ln \left( \frac{I_0}{I_{\text{cell}}} \right). \]  

(41)

We use a very large number of photon packages in order to reduce the noise in the \textsc{radmc-3d} calculation to an acceptable level. Specifically, we use 200 million photon packages and therefore it takes \(\sim 53\) min on one 10-core Intel-Xeon E5-2650 CPU to simulate one wavelength on the given uniform grid with 128\(^3\) resolution, while the calculation with the \textsc{opticaldepth} module took 24 s on 4-core Intel i7-2600, i.e. it was \(\sim 30\) times faster when normalizing both calculations by number of cores.

In Fig. 16, we show a slice at \(z = 0\) of the resulting optical depths (shown in logarithmic scale), \(\tau_{\text{F}}\) from the \textsc{flash} calculation (top left panel) and \(\tau_{\text{R}}\) from the \textsc{radmc-3d} calculation (top right panel), as well as the difference between the two, normalized to the maximum \(\tau_{\text{F,max}}\), of the \textsc{flash} optical depth in the \(xy\)-plane (bottom right panel). The resulting gas temperature calculated by \textsc{flash} is displayed in the bottom left panel. The overall agreement...
Figure 15. Results of the OPTICALDEPTH module test evaluating the accuracy of the hydrogen column density calculation as seen from the centre of the turbulent sphere, as a function of the used angular resolution. Top panel shows the reference hydrogen column density, $N_H(\theta, \phi)$, displayed in the Hammer projection. Left panels below: show $N_{H,\text{PIX}}$ determined by the OPTICALDEPTH module with $N_{\text{PIX}} = 12, 48, 192,$ and $\theta_{\text{lim}} = 0.5$ and 0.25 as denoted in the top left corner of each panel. Right panels below show the relative error in the column density, $e_{N_H/\text{PIX}},$ calculated using equation (38). The relative error in the mean visual extinction, $e_{A_V},$ is given in top right corners of the panels.

is very good and on the level of the remaining noise of the RADMC-3D calculation of a few per cent. Although there is a tendency for the OPTICALDEPTH module to slightly overestimate the optical depth in the densest regions, the difference is always <10 per cent and is ~1 per cent for most cells in the computational domain. The result improves slightly if we use $N_{\text{PIX}} = 192$ and $\theta_{\text{lim}} = 0.25,$ but the additional expense of the calculation is generally not worth the effort.

3.4 Comparison of various MACs

We compare all available MACs with their typical parameters for a simple calculation similar to the static Bonnor–Ebert sphere test described in Section 3.1.1, however, carried out on a uniform 128$^3$ grid. The aim is to provide an approximate measure of the code behaviour. A rigorous analysis of the efficiency of individual MACs, which would need many more tests, since it is highly problem
dependent, is beyond the scope of this paper. The time of the gravity calculation, $t_{\text{grv}}$, was measured on a single processor core, and since it is a single time-step calculation, the time is meaningful only for a mutual comparison between individual MACs (as is also the case for all the static tests in Section 3.1). For each calculation, we determine the relative error in the gravitational acceleration and find its maximum in the computational domain, $e_{a, \text{max}}$. The tested MACs are: the geometric (BH) MAC with three maximum opening angles $\theta_{\text{lim}} = 1.0$ (red plus), $\theta_{\text{lim}} = 0.5$ (olive x), $\theta_{\text{lim}} = 0.2$ (blue star); two APE MACs with absolute error limit $\epsilon_{\text{lim}} = 0.01$ (dark cyan empty square); and MPE MAC with absolute error limit $\epsilon_{\text{lim}} = 0.01$ (dark green empty triangle); and two SumSquare MACs with absolute error limits $\epsilon_{\text{lim}} = 0.01$, and relative error limit $\epsilon_{\text{lim}} = 0.01$ (dark cyan filled square); an MPE MAC with absolute error limit given by $\epsilon = 0.01$ (dark green empty triangle); and two SumSquare MACs with absolute error limits $\epsilon_{\text{lim}} = 0.01$ (black filled circle) and $\epsilon_{\text{lim}} = 0.1$ (orange empty circle). The arrow filled triangle shows the calculation by the multigrid solver. The thin dashed lines are isolines of constant $t_{\text{grv}} \times e_{a, \text{max}}$ assessing the code efficiency and must be made by the user on the basis of the knowledge of the physical configuration that is being treated.

The comparison between the two APE MACs, one using the absolute error limit $\epsilon_{\text{lim}} = 0.01$, and the second one using the relative error limit $\epsilon_{\text{lim}} = 0.01$, shows an interesting, yet not dramatic, difference: the APE with absolute error limit seems to be more efficient by being both faster and more accurate. This result seems to support claims by SW94 that setting the absolute error limit is more appropriate, even though it requires more effort by the user.

The two SumSquare MACs are not among the most efficient, however, they provide an additional advantage of guaranteeing that the error will not exceed the pre-set accuracy limit. It also seems that increasing $\epsilon_{\text{lim}}$ to values as high as 0.1 and above does not result in substantially lower $t_{\text{grv}}$.

The multigrid solver is among the fastest calculations and also among the least accurate. However, the error is high only in the vicinity of the computational domain boundaries caused by an inaccurate multipole solver used to calculate boundary values of the gravitational potential ($M_{\text{lim}} = 10$). In practice, the high accuracy is often not needed close to the boundaries and if the region of size $\sim 20$ per cent around boundaries is excluded from the error calculation, the error $e_{a, \text{max}}$ drops by approximately one order of magnitude. Then, the multigrid solver is comparable to the most efficient and fast APE and MPE MACs.

3.5 Scaling tests

We perform both strong scaling and weak scaling tests. For that we use the setup of the turbulent sphere from Sections 3.2.2 and 3.3.2. The strong scaling tests are done for the Gravity module only, the weak scaling is done for both Gravity and OpticalDepth modules.
where \( t \in \) t \( \epsilon \) \( t \) supercomputer using 48–1536 cores. The speed-up on \( n \) All tests have been run for 10 time-steps on the IT4I/Salomon BH MAC, \( \theta \) running for 10 time-steps. It compares scaling of the tree-solver with the processor cores measured for the turbulent sphere test (see Section 3.2.2) Strong scaling test. Speed-up as a function of the number of tree-solver with BH MAC and ABU off (model (a) from Table 5), S \( mg \) solver (blue crosses). The solid black line shows the (ideal) linear scaling the PPM hydrodynamic solver measured at the test with the BH MAC tree-solver (blue crosses). The solid black line shows the (ideal) linear scaling of leaf-blocks. The run with the APE MAC and ABU exhibits \( \lim \) = 0.5, no ABU (red squares), the tree-solver with the APE MAC, \( \epsilon_{\text{lim}} \) = 0.01 and ABU switched on (green circles), the multigrid solver with \( \epsilon_{\text{ng, lim}} \) = 10^{-6} and \( n_{\text{ng}} \) = 15 (magenta triangles), and the PPM hydrodynamic solver measured at the test with the BH MAC tree-solver (blue crosses). The solid black line shows the (ideal) linear scaling \( S_n \sim n \).

For the strong scaling tests, we use three code configurations: tree-solver with BH MAC and ABU off (model (a) from Table 5), tree-solver with APE MAC and ABU on [model (c)], and multigrid solver with default parameters [model (f)]. We also show the scaling of theFLASH PPM hydrodynamic solver with default parameters. All tests have been run for 10 time-steps on the IT4I/Salomon supercomputer using 48–1536 cores. The speed-up on \( n \) processor cores, \( S_n \) is determined with respect to the run with 48 cores

\[
S_n = \frac{t_{48}}{t_n}
\]

(42)

where \( t_{48} \) is the time spent by the evaluated module on 48 cores and \( t_n \) is the time spent by the same module on \( n \) cores.

We see that the run with the tree-solver and BH MAC gives the best behaviour (speed-up closest to linear). On the other hand, this model is also the slowest one out of the three on 96 cores (see Table 5 and Fig. 18). This can be understood by noting that most of the computational time is spent in the tree-walk, which runs completely in parallel without any communication. Additionally, without ABU there is no problem with load balancing, because the computational time is more or less directly proportional to the number of leaf-blocks, and thus each core receives the same number of leaf-blocks. The run with the APE MAC and ABU exhibits slightly worse scaling, however, the test in Section 3.2.2 shows that on 96 cores, it is almost three times faster than the BH MAC run. This is partially because, due to its more efficient MAC the code spends less time in fully parallel parts and partially because the ABU does not save the time equally on each processor core. The APE MAC scaling is still very good, comparable to the scaling of the hydrodynamic solver, which is highly parallel and needs only to communicate information at the boundaries between domains belonging to different processor cores. The multigrid solver is very fast on 96 cores, comparable to the tree-solver with APE MAC and ABU, however, its efficiency decreases on higher number of cores.

The weak scaling test have been done for two configurations: (i) the tree-solver with the gravity only runs using the APE MAC, \( \epsilon_{\text{lim}} = 0.01 \) and ABU switched on, and (ii) runs calculating the gravitational acceleration and column densities of the three components (total, H\(_2\) and CO; see Section 2.3) using the BH MAC, \( \theta_{\text{lim}} = 0.5 \) with ABU switched off. Each configuration is run for four different grid resolution ranging from 64\(^3\) to 512\(^3\), with the number of cores, \( n \), proportional to the number of grid cells (\( n = 3, 24, 192, 1536 \)).

Results of the weak scaling tests are shown in Fig. 19, where a single time-step runtimes of the two configurations are compared with each other and with runtimes of theFLASH internal hydrodynamic solver. The hydrodynamic solver times (blue curve) follow approximately the \( n^{0.109} \) power law, which is slightly worse than the ‘ideal’ constant scaling. The tree-solver using both theGRAVITY andOPTICALDEPTH modules (green curve) exhibits a similar \( n^{0.15} \) scaling. On the other hand, runs, with only theGRAVITY module (red curve) show the \( n^{0.15} \) scaling only between 3 and 24 cores, and for higher number of cores the scaling gets worse approaching \( n^{0.33} \). This is due to two reasons. First, the gravity only runs are cheaper and the communication making the scaling worse is relatively more important. Note that the communication is negligible for runs on up to 24 cores, since a node on the Solomon computer comprises 24 cores with shared memory. Secondly, the load balancing needed by the ABU becomes worse on a high number of cores. We can also see that the additional calculations of column densities in theOPTICALDEPTH module make the code approximately 10 times slower than the calculation of the gravity on a small number of cores, but it becomes only a factor of \( \sim 3.5 \) for 1536 cores due to better scaling of the more expensive runs with theOPTICALDEPTH module.

### 4 SUMMARY

We have developed an MPI parallel, general purpose tree-solver for the AMR hydrodynamic codeFLASH, that can be used to calculate the
gas gravitational acceleration (or potential), optical depths enabling inclusion of the ambient diffuse radiation, and in future also general radiation transport (Paper II). The code uses an efficient communication strategy predicting which parts of the tree need to be sent to different processors allowing the whole tree-walk to be executed locally. The advantage of this approach is a relatively low memory requirement, important in particular for the optical depth calculation, which needs to process information from different directions. This also makes the implementation of the general tree-based radiation transport possible. In addition to commonly implemented, fully isolated and fully periodic BCs, the code can handle mixed (i.e. isolated in some directions and periodic in others) BCs using a newly developed generalization of the Ewald method. The gravity module implements several MACs that increase the code efficiency by selecting which tree-nodes are acceptable for the calculation on the basis of the mass distribution within them. Using the ABU technique, the code is able to re-use information from the previous time-step and thus further save computational time.

We have run a series of tests evaluating the code accuracy and performance, and compared them to the in-built multigrid gravity solver of FLASH. The simpler ‘static’ tests of the gravity module show that the code provides good accuracy for all combinations of BCs. Comparison with the FLASH default multigrid solver suggests that the tree-solver provides better accuracy for the same computational costs in case of fully isolated BCs, while with fully periodic BCs the multigrid solver seems to be more efficient.

Further, we run two more complex dynamical tests. The Evrard test (gravitational collapse and rebounce of the adiabatic, initially cold gaseous sphere) shows that it is critical to resolve well the dense centre, in order to ensure energy conservation. We found that in order to limit the error in the total energy to less than a few percent, it is necessary to resolve the Jeans length with at least four grid cells, a result similar to that of Truelove et al. (1997) where the same resolution is needed to prevent artificial fragmentation. In general, the Evrard test turns out to be harder for grid codes in comparison with Lagrangian (e.g. SPH) hydrodynamic codes which reach almost perfect energy conservation with very small numbers of particles.

In the second dynamical test, we simulate a turbulent sphere which collapses, fragments and forms sink particles (representing newly formed stars). We find the tree-solver performs well and runs with accuracy of order several percent if it calculates the accelerations directly, and if it is used with the BH or APE MAC with typical parameters. Calculating the gravitational potential instead of particles.

And we find an excellent agreement even for relatively low angular resolution – 48 HEALPIX rays.

Further, using a simplified turbulent sphere test with uniform resolution, we compare the efficiency of all available MACs with their typical parameters. Generally, the BH MAC provides better accuracy for higher computational costs, while APE and MPE MACs result in lower (but often still acceptable) accuracy and are substantially faster. For applications, where an accuracy of order $10^{-2}$ is sufficient, the fastest choice seems to be the APE MAC with the absolute limit on the error.

Finally, we run strong scaling tests and show that the code scales up very well up to at least 1536 processor cores. We conclude that the presented tree-solver is a viable method for calculating self-gravity and other processes in astrophysics and that it is competitive with more commonly used iterative multigrid methods.

ACKNOWLEDGEMENTS

This study has been supported by project 15-06012S of the Czech Science Foundation and by the institutional project RVO:67985815. SW acknowledges the Deutsche Forschungsgemeinschaft for funding through the SPP 1573 ‘The Physics of the Interstellar Medium’, the Bonn–Cologne Graduate School, the SFB 956 ‘The conditions and impact of star formation’, and the funding from the European Research Council under the European Community’s Framework Programme FP8 via the ERC Starting Grant RADFEEDBACK (project number 679852). APW gratefully acknowledges the support of a consolidated grant (ST/K00926/1) from the UK Science and Technology Facilities Council. This work was supported by The Ministry of Education, Youth and Sports from the Large Infrastructures for Research, Experimental Development and Innovations project ‘IT4Innovations National Supercomputing Center - LM2015070’. The software used in this work was in part developed by the DOE NNSA-ASC OASC Flash Center at the University of Chicago.

REFERENCES

Barnes J., Hut P., 1986, Nature, 324, 446 (BH86)
Dubinski J., 1996, New Astron., 1, 133
Ebert R., 1955, Z. Astrophys., 37, 217
Dubinski J., 1996, New Astron., 1, 133
Ebert R., 1955, Z. Astrophys., 37, 217

Downloaded from https://academic.oup.com/mnras/article-abstract/475/3/3393/4795314 by Cardiff University user on 23 April 2019
APPENDIX A: EQUATIONS FOR ACCELERATION IN COMPUTATIONAL DOMAINS WITH PERIODIC AND MIXED BOUNDARY CONDITIONS

In this appendix, we provide formulae for acceleration in computational domains with periodic and mixed BCs. These formulae might be interesting particularly for the reader who intends to implement the Ewald method or its modification to a computational domain with mixed BCs. The orientation of symmetric axes is the same as in Section 2.2.

In analogue to the equation for potential (23), we write acceleration \( \mathbf{a}(r) \) at target point \( r \) as

\[
a(r) = -G \sum_{a=1}^{N} m_a \mathbf{A}(r - r_a),
\]

where

\[
\mathbf{A} = -\nabla \phi.
\]

A1 Periodic boundary conditions

Defining

\[
e_{l_1,l_2,l_3} = \frac{\exp(-\zeta (l_1^2 + (l_2/b)^2 + (l_3/c)^2))}{l_1^2 + (l_2/b)^2 + (l_3/c)^2},
\]

\[
u_{l_1,l_2,l_3} = \frac{2\pi l_1(x-x_a)}{L_x} + \frac{2\pi l_2(y-y_a)}{L_y} + \frac{2\pi l_3(z-z_a)}{L_z},
\]

one obtains by differencing equation (11), the components of function \( \mathbf{A} \) in the form of

\[
A_k = \sum_{l_1,l_2,l_3} \left\{ \frac{2a}{\sqrt{\pi}} \frac{\exp(-a^2 u_{l_1,l_2,l_3})}{u_{l_1,l_2,l_3}} + \frac{\text{erfc}(a \sqrt{u_{l_1,l_2,l_3}}/2)}{u_{l_1,l_2,l_3}^{3/2}} \right\} (x - x_a - l_1 L_x) + \frac{2}{b c L_x^2} \sum_{l_1,l_2,l_3} l_1 e_{l_1,l_2,l_3} \sin(\nu_{l_1,l_2,l_3}),
\]

where

\[
\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-t^2} dt = 1 - \text{erf}(x).
\]

The formulae are organized so as to avoid problems with floating point representation.

---

\[ A_y = \sum_{i_1^2 + (b_2/b)^2 + (c_2/b)^2 \leq 10} \left\{ \frac{2a \exp(-\alpha^2u_{i_1,i_2,i_3})}{\sqrt{\pi}u_{i_1,i_2,i_3}} + \frac{\text{erfc}(\alpha \sqrt{u_{i_1,i_2,i_3}})}{u_{i_1,i_2,i_3}^{3/2}} \right\} (y - y_4 - i_2bL_y) + \frac{2}{b^2cL_z^2} \sum_{i_1^2 + (b_2/b)^2 + (c_2/b)^2 \leq 10} l_2 e_{i_1,i_2,i_3} \sin(e_{i_1,i_2,i_3}), \]  
(A7)

\[ A_x = \sum_{i_1^2 + (b_2/b)^2 + (c_2/b)^2 \leq 10} \left\{ \frac{2a \exp(-\alpha^2u_{i_1,i_2,i_3})}{\sqrt{\pi}u_{i_1,i_2,i_3}} + \frac{\text{erfc}(\alpha \sqrt{u_{i_1,i_2,i_3}})}{u_{i_1,i_2,i_3}^{3/2}} \right\} (z - z_6 - i_3cL_y) + \frac{2}{bc^2L_z^2} \sum_{i_1^2 + (b_2/b)^2 + (c_2/b)^2 \leq 10} l_3 e_{i_1,i_2,i_3} \sin(v_{i_1,i_2,i_3}). \]  
(A8)

### A2 Mixed boundary conditions of type 2P11

To simplify the formulae below, we define

\[ u_{i_1,i_2} = (x - x_a - i_1L_x)^2 + (y - y_a - i_2bL_y)^2 + (z - z_a)^2, \]  
(A9)

\[ v_{i_1,i_2} = \frac{2\pi d_1(x - x_a)}{L_x} + \frac{2\pi d_2(y - y_a)}{bL_x}, \]  
(A10)

and

\[ \tilde{I}(l_1, l_2, z - z_a) \equiv I(l_1, l_2, z - z_a) \exp(-\zeta(l_1^2 + (l_2/b)^2)) \]

\[ = \frac{\pi}{2\sqrt{l_1^2 + (l_2/b)^2}} \left\{ \exp\left(-\frac{\gamma^2}{4\alpha}\right) \exp(-\zeta(l_1^2 + (l_2/b)^2)) \text{erfcx} \left( \frac{\xi \sqrt{l_1^2 + (l_2/b)^2} + \gamma/2}{\sqrt{\xi}} \right) \right\}. \]  
(A11)

\[ I'(l_1, l_2, z - z_a) \equiv d\tilde{I}(l_1, l_2, z - z_a) \]  

\[ = \frac{\pi}{2} \left\{ \exp(-\frac{\gamma^2}{4\alpha}) \exp(-\zeta(l_1^2 + (l_2/b)^2)) \text{erfcx} \left( \frac{\xi \sqrt{l_1^2 + (l_2/b)^2} + \gamma/2}{\sqrt{\xi}} \right) \right\}, \]  
(A12)

where \( I(l_1, l_2, z - z_a) \) is defined by equation (17).

Function \( A \) then takes the form

\[ A_x = \sum_{i_1^2 + (b_2/b)^2 \leq 10} \left\{ \frac{2a \exp(-\alpha^2u_{i_1,i_2})}{\sqrt{\pi}u_{i_1,i_2}} + \frac{\text{erfc}(\alpha \sqrt{u_{i_1,i_2}})}{u_{i_1,i_2}^{3/2}} \right\} (x - x_a - i_1L_x) + \frac{2}{bL_x^2} \sum_{i_1^2 + (b_2/b)^2 \leq 10} l_1 \sin(v_{i_1,i_2}) \tilde{I}(l_1, l_2, z - z_a), \]  
(A13)

\[ A_y = \sum_{i_1^2 + (b_2/b)^2 \leq 10} \left\{ \frac{2a \exp(-\alpha^2u_{i_1,i_2})}{\sqrt{\pi}u_{i_1,i_2}} + \frac{\text{erfc}(\alpha \sqrt{u_{i_1,i_2}})}{u_{i_1,i_2}^{3/2}} \right\} (y - y_a - i_2bL_y) + \frac{2}{b^2L_y^2} \sum_{i_1^2 + (b_2/b)^2 \leq 10} l_2 \sin(v_{i_1,i_2}) \tilde{I}(l_1, l_2, z - z_a), \]  
(A14)

\[ A_z = \sum_{i_1^2 + (b_2/b)^2 \leq 10} \left\{ \frac{2a \exp(-\alpha^2u_{i_1,i_2})}{\sqrt{\pi}u_{i_1,i_2}} + \frac{\text{erfc}(\alpha \sqrt{u_{i_1,i_2}})}{u_{i_1,i_2}^{3/2}} \right\} (z - z_a) - \frac{2}{bL_z^2} \sum_{i_1^2 + (b_2/b)^2 \leq 10} \cos(v_{i_1,i_2}) I'(l_1, l_2, z - z_a). \]  
(A15)

### A3 Mixed boundary conditions of type 1P2I

Here, we introduce

\[ u_{i_1} = (x - x_a - i_1L_x)^2 + (y - y_a)^2 + (z - z_a)^2, \]  
(A16)

\[ v_{i_1} = \frac{2\pi d_1(x - x_a)}{L_x}, \]  
(A17)

which simplifies the formula for function \( A \) to

\[ A_x = \sum_{i_1^2 \leq 10} \left\{ \frac{2a \exp(-\alpha^2u_{i_1})}{\sqrt{\pi}u_{i_1}} + \frac{\text{erfc}(\alpha \sqrt{u_{i_1}})}{u_{i_1}^{3/2}} \right\} (x - x_a - i_1L_x) + \frac{4\pi}{L_x^2} \sum_{i_1^2 \leq 10} l_1 \exp(-\zeta l_1^2) \sin(v_{i_1}) K(l_1, y - y_a, z - z_a), \]  
(A18)
\[ A_y = \sum_{l_i,i_z^2 \leq 10} \left\{ \frac{2\pi}{\sqrt{\pi}} \exp(-\alpha^2 u_{i_z}) + \text{erfc}(\alpha u_{i_z}) \right\} \left( y - y_a \right) + \frac{4\pi}{L^2} \frac{y - y_a}{(y - y_a)^2 + (z - z_a)^2} \sum_{l_i,i_z^2 \leq 10} \exp(-\alpha l_i^2 \cos(v_i)M(l_i, y - y_a, z - z_a)). \]  

(A19)

\[ A_z = \sum_{l_i,i_z^2 \leq 10} \left\{ \frac{2\pi}{\sqrt{\pi}} \exp(-\alpha^2 u_{i_z}) + \text{erfc}(\alpha u_{i_z}) \right\} \left( z - z_a \right) + \frac{4\pi}{L^2} \frac{z - z_a}{(y - y_a)^2 + (z - z_a)^2} \sum_{l_i,i_z^2 \leq 10} \exp(-\alpha l_i^2 \cos(v_i)M(l_i, y - y_a, z - z_a)). \]  

(A20)

where \( K(l_i, y - y_a, z - z_a) \) is given by equation (22), and function \( M(l_i, y - y_a, z - z_a) \equiv -dK(l_i, \eta(y - y_a, z - z_a))/d\eta \) is

\[ M(l_i, y - y_a, z - z_a) = \int_0^\infty J_1(\eta q) \exp\left( -\zeta q^2 \right) q^2 dq, \]  

(A21)

where \( J_1 \) is the Bessel function of the first kind and first order. Note that variables \( \xi, \gamma, \) and \( \eta \) are defined in Section 2.2.4.

**APPENDIX B: CODE RUNTIME PARAMETERS**

Here, we list runtime parameters of the tree-solver and GRAVITY and OPTICALDEPTH modules that can be set in the flash.par configuration file. Apart from parameters discussed in the main body of this work (e.g. MAC selection and accuracy limit), the code should work well with the default parameters. Additional information is provided in the Flash Users Guide and directly in the source code as comments.

**B1 Tree-solver parameters**

- `gr_bhPhysMACTW`– indicates whether MACs of physical modules (e.g. GRAVITY) are used during tree-walks; if false, the geometric BH MAC is used instead (type: logical, default: false)
- `gr_bhPhysMACComm`– indicates whether MACs of physical modules (e.g. GRAVITY) are used for communication of block-trees; if false, the geometric BH MAC is used instead (type: logical, default: false)
- `gr_bhTreeLimAngle`– maximum opening angle, \( \theta_{\text{lim}} \), of the geometric BH MAC (type: real, default: 0.5)
- `gr_bhTreeSafeBox`– relative (with respect to the block size) size of a cube around each block, \( \eta_{\text{SB}} \), in which the target point cannot be located (type: real, default: 1.2)
- `gr_bhUseUnifiedTW`– obsolete, will be deleted in future versions
- `gr_bhTWMaxQueueSize`– maximum number of elements in the priority queue (type: integer, default: 10000)
- `gr_bhAcceptAccurateOld`– indicates whether ABU (see Section 2.2.6) is active; will be renamed to `gr_bhABU` in future versions (type: logical, default: false)
- `gr_bhLoadBalancing`– indicates whether Load Balancing (see Section 2.2.6) is active (type: logical, default: false)
- `gr_bhMaxBlkWeight`– maximum workload weight, \( \omega_{\text{ub}} \) (type: real, default: 10)

**B2 GRAVITY module parameters**

- `grv_bhNewton`– Newton’s constant of gravity; if negative, the value is obtained from the Flash internal data base of physical constants (type: real, default: -1)
- `grv_bhMAC`– type of MAC calculated by the GRAVITY module if `gr_bhPhysMACTW` or `gr_bhPhysMACComm` is set true; currently accepted values are: ‘ApproxPartialErr’, ‘MaxPartialErr’, and ‘SumSquare’ (experimental) (type: string, default: ‘ApproxPartialErr’)
- `grv_bhMPDegree`– degree of multipole expansion used to estimate the error of a single-node contribution with APE and MPE MACs; `grv_bhMPDegree` corresponds to \( p + 1 \) used in equations (2) and (6) (type: integer, default: 2)
- `grv_bhUseRelAccErr`– indicates whether the `grv_bhAccErr` parameter (below) should be interpreted as a relative error limit, \( \epsilon_{\text{lim}} \) (type: logical, default: false)
- `grv_bhAccErr`– maximum allowed error set either relatively with respect to the accuracy from the previous time-step, \( \epsilon_{\text{lim}} \), or absolutely, \( \epsilon_{\text{ub}} \) (type: real, default: 0.1)
- `grav_boundary_type`– type of BCs for gravity for all directions; the accepted values are: ‘isolated’, ‘periodic’, and ‘mixed’; if set to ‘mixed’, BCs in individual directions are set by the parameters below (type: string, default: ‘mixed’)
- `grav_boundary_type_x`– type of gravity BCs in the x-direction; the accepted values are: ‘isolated’ and ‘periodic’ (type: string, default: ‘isolated’)
- `grav_boundary_type_y`– same as `grav_boundary_type_x` but in the y-direction
- `grav_boundary_type_z`– same as `grav_boundary_type_x` but in the z-direction
- `grv_bhEwaldSeriesN`– number of terms used in the expansion given by equation (11) to calculate the Ewald field (type: integer, default: 10)
- `grv_bhEwaldAlwaysGenerate`– indicates whether the Ewald field should be regenerated at the simulation start; if false, it is read from file with name given by parameters `grv_bhEwaldFName` or `grv_bhEwaldFNameAccV42` and `grv_bhEwaldFNamePosV42` (type: logical, default: true)
grv_bhEwaldFieldNxV42 – number of points of the Ewald field lookup table in the x-direction when the first approach described in Section 2.2.5 is used (default in FLASH versions up to 4.2); (type: integer, default: 32)
grv_bhEwaldFieldNyV42 – same as the preceeding parameter but for the y-direction
grv_bhEwaldFieldNzV42 – same as the preceeding parameter but for the z-direction
grv_bhEwaldNRefV42 – number of nested grid levels of the Ewald field when the first approach described in Section 2.2.5 is used; if negative, the number of nested grid levels is calculated automatically from the minimum cell size (type: integer, default: -1)
grv_bhLinearInterpolOnlyV42 – indicates whether the linear interpolation in the Ewald field is used (with the first approach described in Section 2.2.5); if false, then the more expensive and accurate quadratic interpolation is used for some calculations (type: logical, default: true)
grv_bhEwaldFNameAccV42 – name of file to store the Ewald field accelerations when the first approach described in Section 2.2.5 is used (type: string, default: ‘ewald_field_acc’)
grv_bhEwaldFNamePotV42 – name of file to store the Ewald field potential when the first approach described in Section 2.2.5 is used (type: string, default: ‘ewald_field_pot’)
grv_bhEwaldNPer– number of points in each direction of the Ewald field coefficients when the second approach described in Section 2.2.5 is used (type: integer, default: 32)
grv_bhEwaldFName– name of file to store the Ewald field coefficients in the case the second approach described in Section 2.2.5 is used (type: string, default: ‘ewald_coeffs’)
grv_useExternalPotential– indicates whether the external time-independent gravitational potential read from file is used (type: logical, default: false)
grv_usePoissonPotential– indicates whether the potential (or accelerations) computed by the (tree) Poisson solver is used (type: logical, default: true)
grv_bhExtrnPotFile– name of file with the external gravitational potential (type: string, default: ‘external_potential.dat’)
grv_bhExtrnPotType– symmetry of the external gravitational potential; currently, two options are available: ‘spherical’ and ‘planez’ (plane parallel, varying along the z-direction); (type: string, default: ‘planez’)
grv_bhExtrnPotCenterX – centre of the external potential x-coordinate given in the FLASH internal coordinates (type: real, default: 0)
grv_bhExtrnPotCenterY – same as the preceeding parameter but for the y-direction
grv_bhExtrnPotCenterZ – same as the preceeding parameter but for the z-direction

B3 Optical depth module parameters

tr_nSide – level of the HEALPIX grid; number of pixels is \( N_{\text{pix}} = 12 \times 4^{(\text{tr}_\text{nSide} - 1)} \) (type: integer, default: 1)
tr_i1NR – number of points in the radial direction for the calculation of the fraction of node that intersects with a given ray (type: integer, default: 50)
tr_i1NTTheta– number of points in the \( \theta \)-direction of the table recording a fraction of the node that intersects with a ray at a given \( \theta \) (type: integer, default: 25)
tr_i1NPhi– number of points in the \( \phi \)-direction of the node-ray intersection table (type: integer, default: 50)
tr_i1NNS– number of points describing the angular node size in the node-ray intersection table (type: integer, default: 25)
tr_i1FinePix– number of additional pixels in each angular direction used to calculate the node-ray intersection table (type: integer, default: 4)
tr_bhMaxDist – maximum distance from a target point up to which the optical depth is calculated (type: real, default: 10^{33})
tr_odCDT0Index– exponent relating the gas density to the absorption coefficient used during the calculation of the optical depth in a given direction (type: real, default: 1)

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