A Simple Multigrid Scheme for Solving the Poisson Equation with Arbitrary Domain Boundaries.

Thomas Guillet[a], Romain Teyssier[a,b]

[a] *IRFU/SAp, CEA Saclay, Batiment 709, 91191 Gf-sur-Yvette Cedex, France*
[b] *Institute of Theoretical Physics, University of Zurich, Winterthurerstrasse 190, 8057 Zurich, Switzerland*

Abstract

We present a new multigrid scheme for solving the Poisson equation with Dirichlet boundary conditions on a Cartesian grid with irregular domain boundaries. This scheme was developed in the context of the Adaptive Mesh Refinement (AMR) schemes based on a graded-octree data structure. The Poisson equation is solved on a level-by-level basis, using a "one-way interface" scheme in which boundary conditions are interpolated from the previous coarser level solution. Such a scheme is particularly well suited for self-gravitating astrophysical flows requiring an adaptive time stepping strategy. By constructing a multigrid hierarchy covering the active cells of each AMR level, we have designed a memory-efficient algorithm that can benefit fully from the multigrid acceleration. We present a simple method for capturing the boundary conditions at all levels of the multigrid hierarchy, based on a second-order accurate reconstruction of the boundary. In case of very complex boundary, small scale features become smaller than the discretization cell size of coarse multigrid levels and convergence problems arise. We propose a simple solution to restore desirable convergence properties. The proposed method was successfully implemented on distributed memory archi-

tectures in the RAMSES code, for which we present and discuss convergence properties and timing performances.
Keywords:

---

1. Introduction

Elliptical partial differential equations play a central role in many mathematical and physical problems. The Poisson equation, in particular, arises naturally in the context of electromagnetism, fluid dynamics and gravity. Is is therefore of great significance in astrophysics, of which those fields are essential theoretical aspects. In the classical Newtonian limit, one can relate the gravitational potential Φ to the matter distribution (local density) ρ by:

$$\Delta\Phi = \rho \tag{1}$$

Applications in computational astrophysics therefore often require solving a discretization of Eq. (1) on some discrete grid. The ubiquity of the Poisson equation in science has led to the developement of many suitable numerical solving techniques, for a wide range of specific applications and problems.

In some cases, one can solve for the discretized potential directly from Eq. (1): the convolution theorem allows to solve the Poisson equation directly in Fourier space using the Green function of the Laplacian operator on the grid of interest. In computational applications, this is usually done using the Fast Fourier Transform (FFT), which is particularly suited to periodic boundary conditions problems on rectangular, regular grids. This direct method is very effective: on an grid of linear size N, the computing time in 3 dimensions is of order $\mathcal{O}(N^3 \log N)$, and yields the solution to the discretized

problem within numerical roundoff accuracy. The Green function method can be generalized to non-periodic boundary conditions and adaptive grids on rectangular domains [see e.g. 8].

Another class of Poisson solving schemes are relaxation methods, such as Gauss-Seidel or successive over-relaxation (SOR) [see e.g. 20]. Unlike the Green function method, those schemes are iterative, and rely on an initial estimate ("first guess") of the solution. The estimate is then successively improved by iterative damping of the residual $r = \Delta\Phi - \rho$. Relaxation methods are generally simple to implement, and suitable for a wide range of elliptic problems. Methods such as Gauss-Seidel or SOR only require the forward computation of the differential operator, which is usually inexpensive, and makes it possible to use such relaxation methods on complex grid geometries. Krylov subspace methods [see e.g. 24], such as the Conjugate Gradient (CG) method, are another class of iterative methods which can be used to solve Eq. (1) in the form of the optimization problem:

$$\min_{\Phi} \|\Delta\Phi - \rho\|^2 \tag{2}$$

Classical iterative methods have gained in interest in the last decade with the advent of massively parallel computing. They only require one-cell thick, boundary layer from the neighboring processor, while FFT involves a massive transpose of the whole volume of data. Moreover, one does not need the exact solution of the discretized Poisson equation that the FFT provides: the error in the solution is already dominated by truncation errors of a fraction of a percent for practical applications. Iterative methods can therefore be stopped quite early, when the residual norm has dropped well below the truncation errors. Still, classical iterative methods can be very

time consuming, especially when the grid size in increased. Usually, the large scale modes are the slowest to converge, and the number of iterations required to reach a given level of residual usually increases with the number of grid points per dimension [21].

The convergence rate of iterative relaxation methods, such as Gauss-Seidel, can be dramatically improved by multigrid acceleration [2, 30]. Multigrid methods use a hierarchy of discretizations of decreasing spatial resolution. At the resolution of the initial problem (on the finest grid), the solution is iteratively improved by subtracting corrections obtained from the coarser grids. This ensures that the large-scale modes of the residual are efficiently damped, while a traditional smoothing relaxation scheme takes care of the small-scale modes of the error. In ideal cases (e.g. for smooth problems with simple boundary conditions on rectangular domains), multigrid methods can exhibit remarkable convergence properties: the residual norm decreases at a constant rate during the whole convergence process (down to machine precision). The convergence rate can be very fast (see examples below), and more importantly *it does not depend on the size of the grid*. This last property is perhaps the most important one, since it allows one to consider very large computational problems: for a given level of convergence in the residual norm, multigrid methods are $N^3 \log N$ algorithms. This scaling is similar to the one offered by the FFT, and although multigrid is still slower than FFT for small configurations, it can be quite competitive for large parallel computations and reasonable stopping criteria. In contrast to traditional iterative schemes, the convergence properties of multigrid are unaffected by the choice and quality of the first guess. Another reason to use multigrid is that

it couples nicely to adaptive Cartesian grids, while FFT-based methods in this case are still possible, but quite complicated [see e.g. 4, 8, 22]. Last but not least, while FFT requires a Poisson equation with constant coefficients, multigrid methods, because they are defined in real space, can be adapted to solve Poisson equations with non-uniform coefficients, or even non-linear equations. This property is particularly useful for problems with complex boundaries [19], multiple incompressible fluids [23], or models with modified gravity [12, 28, 15]).

The objective of this paper is to implement a simple and efficient multigrid Poisson solver for a Cartesian grid with *arbitrary domain boundaries*. Complex boundary conditions are usually found in fluid mechanics problems featuring immersed bodies or complex multiphase flows [26, 25, 14]. Here, our motivation is different: we would like to design a Poisson solver for the class of Adaptive Mesh Refinement (AMR) codes for which the mesh is defined on a "graded octree" [13, 27, 19]. The octree mesh structure ensures that the geometry of the grid closely adapts to the properties of the flow, without the traditional overhead associated to the large rectangular patches of block–structured AMR. The consequence is however that the mesh of a given AMR level can have a very complicated shape, with holes and irregular boundaries. If one solves the Poisson equation on the whole AMR hierarchy, this translates into a modification of the Laplacian operator at fine–coarse grid boundaries, but the corresponding multigrid solver remains similar to its Cartesian grid equivalent [see e.g. 10].

In most astrophysical applications, it is however almost always impractical to solve the Poisson equation on the whole grid at once: self-gravity has

the effect of dramatically increasing the dynamical range of the density field, leading to very different characteristic timescales within the same computational domain. Advancing the whole system with one single time step can be very inefficient, since only a very small fraction of the volume actually needs high accuracy in the time coordinate. This is particularly true for cosmological simulations, where most of the computational volume is covered by low density regions that evolve slowly, while a small number of highly resolved cells sample dense regions (such as galaxies), where the dynamical time scale is very short. Most AMR codes address this problem by using *adaptive time stepping*, where a given fine level is updated twice as often as its coarser level, ensuring that the actual timestep remains close to the natural CFL timestep (so that the whole AMR grid is not updated more often than needed). As a consequence, consecutive fine and coarse levels are only synchronized at every other fine timestep.

Whenever all the AMR levels are synchronized, it is possible to solve the Poisson equation on the whole grid at once. Such a Poisson solver is called a "two-way interface" scheme, as the information is propagated both from the coarse grids to the finer grids and back: the coarse grids "feel" the effect of the finer grids. Many multigrid solvers have been successfully implemeted in this case [10, 19, 17, 22]. Popinet [19] has developed an incompressible flow solver with complex boundary conditions, using an octree data structure. The solver features a "half V-cycle" multigrid schedule, which make it possible to perform multigrid iterations within the existing AMR hierarchy, with no additional storage. This pioneering work, however, requires the use of a single global time step for the whole AMR grid. Most of the time, fine

and coarse levels are not synchronized, and it is not possible to solve on the whole grid at once. It is therefore necessary to resort to a "one-way interface" scheme [see 9, 13, 27, 17]. In the "one-way interface" approach, the AMR levels are updated separately: the coarsest level is updated first, then the computed coarse potential is used to impose boundary conditions on the finer levels, which can then be solved separately and more frequently. The gravitational potential on finer AMR levels is solved by imposing Dirichlet boundary conditions at the finer level boundary, enforcing potential continuity. Because we are dealing with arbitrarily complex octree structure, we must be able to solve the Poisson equation on a given refinement level with arbitrarily shaped domains. The problem we would like to address here is therefore to solve the Poisson equation using the multigrid method, with the constraint that Dirichlet boundary conditions are imposed on the outer cell faces of an arbitrary domain (see Fig. 1).

As Cartesian grids are much more practical than unstructured meshes, the accurate description of boundaries immersed in Cartesian grids have been the focus of much effort, especially in computational fluid dynamics [see e.g. 1, 31]. This approach has also led to the developement of sophisticated Poisson solvers, some relying on multigrid acceleration [10, 16].

Our goal is to present a simpler but efficient solver, suitable in particular to the case of tree-based AMR codes with a one-way interface scheme. We will apply our new scheme to the AMR code RAMSES, a self-gravitating fluid dynamics code for astrophysical applications [27]. The octree structure is based on the "Fully Threaded Tree" approach proposed by Khokhlov [11] for which cells are split into "octs" (groups of 8 cells) on a cell-by-cell ba-

sis [13, 27]. The paper is organized as follows: we first describe our basic multigrid algorithm for the Poisson equation with Dirichlet boundary conditions. We then detail how we capture complex boundary conditions within our multigrid framework. We report a very fundamental issue of our approach (namely the "small islands" problems) and we propose a simple fix to overcome it, and we finally discuss the performance of our algorithm.

2. A Multigrid algorithm for complex boundary conditions

Multigrid methods combine a relaxation solver (usually a smoother such as Gauss-Seidel or Jacobi sweeps) and a multi-resolution approach, to ensure efficient damping of both small and large scale components of the residual (see for example [30] for an introduction).

For one-way interface schemes (in the RAMSES code for example), the Poisson equation is solved over the whole AMR hierarchy on a level-by-level basis, the size of the cell involved in each individual Poisson solve being uniform. Each AMR level constitutes the finest level of the multigrid hierarchy. The finest multigrid level is therefore defined by a Cartesian grid with complex irregular boundaries. We then build a hierarchy of coarser grids which cover this reference domain (see Fig. 1 for an illustration). This new hierarchy defines the multigrid structure for the current level. Although it is also based on an octree structure similar to the underlying AMR grid, it is a different structure that does not interfere with the coarse AMR levels (which are, in a way, "orthogonal" to the multigrid levels).

One major advantage of using this secondary grid hierarchy for each Poisson solve is that the computational cost of the multigrid solver at a given

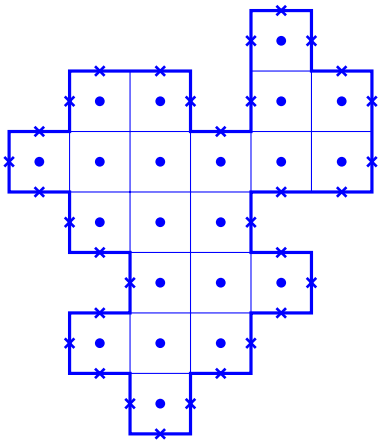

Figure 1: Poisson problem at the fine level for a given AMR level. The solution potential to be determined it defined at the cell centers (dots). The boundary conditions are defined at the border cell faces (crosses). The boundary values are linearly interpolated from the solution at the coarser AMR level (not represented), which was previously computed because of the one-way interface scheme.

level only scales as the number of cells in that level, as we use only a subset of the AMR coarser cells. This is especially crucial for astrophysicall problems, where fine grids can occupy a fraction of the volume much smaller than the corser levels.

Let us consider the Poisson problem, discretized on a fine grid domain Ω_f.

$$\begin{aligned} \Delta_f \Phi_f = \rho_f \quad & \text{on } \Omega_f, \\ \text{Dirichlet conditions} \quad & \text{on } \partial\Omega_f. \end{aligned}$$

where the f subscripts denote the discretized Laplacian operator and fields on the fine grid. For a single multigrid iteration, we follow the traditional recipe:

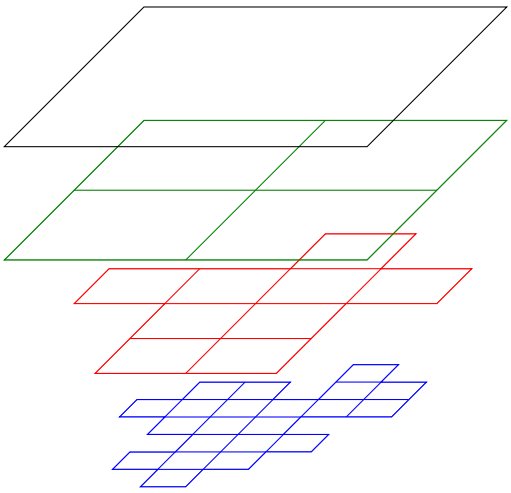

Figure 2: Levels of the multigrid hierarchy for the problem of Fig. 1.

1. Perform N_{pre} Gauss-Seidel smoothing passes on the current estimate Φ_f of the fine solution,
2. Compute the fine residual

$$r_f \longleftarrow \Delta\Phi_f - \rho,$$

3. Compute the fine-to-coarse restrictions of the solution and residual:

$$\Phi_c \longleftarrow R(\Phi_f)$$
$$r_c \longleftarrow R(r_f),$$

4. Define a coarsened domain Ω_c and boundaries $\partial\Omega_c$ from Ω_f. The prescription for deriving $\partial\Omega_c$ from the fine level is the key issue, and will be discussed in section 3.
5. Compute a coarse correction $\delta\Phi_c$, by performing N_{cycles} multigrid iter-

ations for the following coarse problem:

$$\Delta_c(\delta\Phi_c) = -r_c \quad \text{on } \Omega_c$$
$$\delta\Phi_c = 0 \quad \text{on } \partial\Omega_c,$$

6. Prolong $\delta\Phi_c$ into a fine correction

$$\delta\Phi_f \longleftarrow P(\delta\Phi_c),$$

7. Correct the fine level solution

$$\Phi_f \longleftarrow \Phi_f + \delta\Phi_f,$$

8. Perform N_{post} smoothing passes on Φ_f.

This multigrid iteration is repeated on the fine grid until the norm of the fine residual is considered small enough. To fully specify the scheme, one needs to specify integers N_{pre}, N_{post} and N_{cycles}, the restriction and prolongation operators R and P, and the discretization of the Laplacian operator.

In our case, we have found $N_{pre} = N_{post} = 2$ to yield the best performance in terms of solver time for our applications in RAMSES. We will now discuss the prolongation, restriction and discretized Laplacian operators. We postpone the discussion of the parameter N_{cycles} to section 5.2.

We use a finite difference approach to discretize the Poisson equation on the Cartesian grid. Inside the domain, away from the boundaries, we use the standard 7-point discretization of the Laplacian operator:

$$(\Delta\Phi)_i = \frac{1}{h^2} \sum_{\{i,j\}} (\Phi_j - \Phi_i) \tag{3}$$

where the sum extends over all the 6 pairs $\{i, j\}$ of neighbouring cells, and h is the size of the cells on which the Laplacian is evaluated. This Laplacian operator is second-order accurate and our multigrid implementation is currently restricted to this case.

For the relaxation smoother, we use a Gauss-Seidel smoother with red-black ordering, with the Laplacian operator as defined above. With red-black ordering, the cells are updated in two passes, each pass running over a half of the domain following the colors of a checkerboard pattern.

Prolongation and restriction operators define how the problem is down-sampled to a coarser level (restriction) and how the coarser level correction is interpolated back to the finer level (prolongation).

A common rule of thumb for the choice of prolongation and restriction operators for the Poisson equation is the order condition $n_P + n_R > q$, where n_P and n_R are the prolongation and restriction operator orders, respectively, and q is the order of the Laplacian operator [see 30, 20]. Simple standard operators are shown in Figure 3. Straight injection is defined as $P1$, while its transpose, $R1$, boils down to a simple averaging. These operators are first–order in space ($n_R = 1$ and $n_P = 1$). Second–order schemes can be easily designed, like tri-linear interpolation (noted here P2) and its transpose (R2), whose weights are shown in Figure 3.

We have settled for simple averaging for restriction (R1), and linear interpolation (P2) for prolongation (see Fig. 3), as this choice turned out to yield the best convergence rates for the classes of grids which we have studied. Additionally, as will be discussed in Section 4, picking simple averaging as the restriction operator turns out to be particularly convenient in conjunction

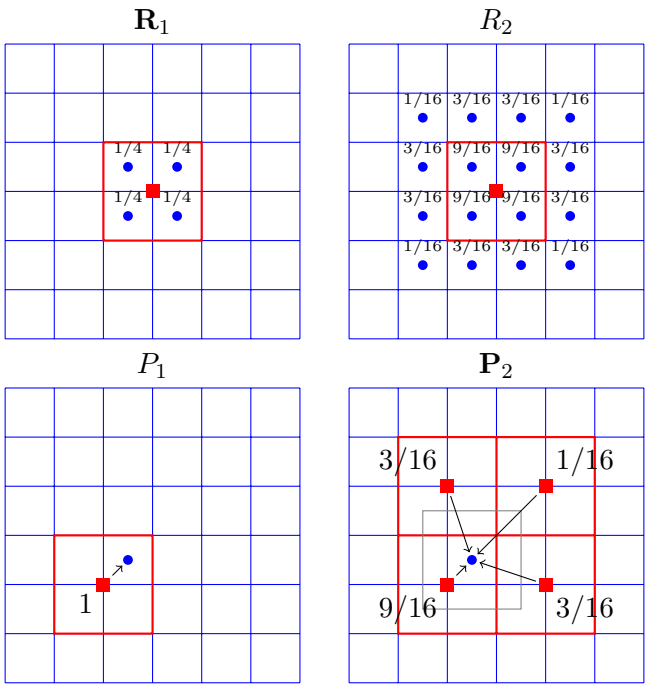

Figure 3: Common first-order and second-order (respectively left and right) restriction and prolongation schemes (respectively top and bottom). R_1 simply averages the values of the subcells to obtain a coarse value. P_1 assigns the same coarse value to the all the fine subcells (straight injection). Our multigrid scheme uses R_1 and P_2.

with our prescription for boundary representation.

3. Second-order boundary reconstruction and the modified Laplacian operator

In our multigrid algorithm, the key ingredient is the mathematical representation of the domain boundary, and, more importantly, of its projection at coarser levels. Because our goal is to take advantage of the Cartesian grid structure of the AMR scheme, we restrict oursleves to Cartesian grid interface capturing techniques. Among the many different solutions proposed in the literature, the cut cell [10] and the level set [26, 18, 7] approaches stand out as simple and efficient techniques for interface reconstruction. The multigrid algorithm for complex boundary we propose here can be easy implemented with any of these techniques as the basic interface reconstruction scheme.

In what follows, we however simplify the problem even further, since we work in the framework of a one-way AMR Poisson solver. At the finest level (i.e., on the actual grid where the Poisson equation is to be solved), the boundaries $\partial\Omega$ are simply positioned at the faces of the outer cells themselves. In the level set approach, the boundary could have been placed at the zero level of some interpolated distance function [see e.g. 7]. Since computing distance function comes with a cost, we have considered here a simpler approach based on a coloring scheme: inner cells are initialized with a mask function with value $m_i = 1$ and outer cells with a mask $m_i = -1$. The domain boundary $\partial\Omega$ is defined at the position where the interpolated value of the cell-centered mask ($m_i \in [-1, 1]$) crosses zero. At the finest multigrid level, these interpolated positions are at the cell faces, as desired.

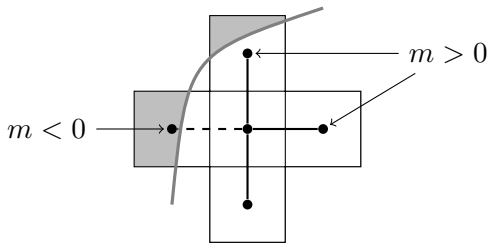

Figure 4: The Laplacian stencil near boundaries: the center of the leftmost cell lies outside the domain, and therefore has a negative mask value $m < 0$. For this cell, a ghost value will be reconstucted as shown on Fig. 5. For all the three other cells, the actual cell center values are used unchanged.

We now compute the mask value at coarser multigrid levels by simply averaging recursively the mask value at the finest level (as for the residual). Here again, if one wants to use a distance function to capture the boundary, one could compute the coarser representation of the distance function also by simple averaging. At each level ℓ in the multigrid hierarchy, the domain Ω^ℓ is defined as the set of cells for which $m_i^\ell > 0$. The use of the R1 operator maintains the correct location of the boundary as much as possible, without spreading the boundary information too much across neighbouring coarse cells.

In order to reconstruct the location of the boundary, and solve for the corresponding Poisson problem, we use a prescription similar to the one described in Gibou et al. [7]. The idea is to redefine the Laplacian operator close to the domain boundary. Whenever one of the neighbouring Φ_j has $m_j \leq 0$ (and therefore lies outside Ω), it is replaced by a ghost value $\tilde{\Phi}_j$ linearly extrapolated from Φ_i and the boundary value (see Fig. 4). This ghost value depends explicitly on Φ_i and the boundary condition. A 1D illustration

is presented on figure 5. As in [7], the boundary is positioned in each direction independently, and the 1D case trivially generalizes to 3D. Since this prescription uses a linear reconstruction for the boundary in each direction, the boundary position is recovered at second order spatial accuracy.

The coarsening of the boundary from $\partial\Omega_f$ to $\partial\Omega_c$ is now fully specified by the restriction operator used for the mask m_i and by the modified Laplacian closed to the boundary. These are the 2 key ingredients in our multigrid scheme: the boundary remains at the same location (up to second order in space) when we go from fine to coarse levels in the multigrid hierarchy. The boundary condition $\Phi = 0$ is therefore imposed at the correct location, so that the coarse solution of the Poisson equation with the coarse boundary corresponds to the solution of the Poisson equation at the finer level with the fine representation of the boundary.

We speculate that the chosen boundary reconstruction scheme should satisfy an order condition similar to the prolongation and restriction operators [30, 20]. Although a mathematical proof is beyond the scope of this paper, our numerical experiments suggest that this is indeed the case. We see in Figure 7 that, in case of a smooth boundary, our second-order boundary reconstruction scheme results in a perfect multigrid convergence[1], while the first-order scheme doesn't. As we will now discuss, in case of very complex boundaries, second-order reconstruction of the boundary is not possible anymore, and we have to degrade our scheme to first-order to ensure convergence.

[1]Multigrid convergence means here that the damping rate of the residual norm does not depend on the grid size.

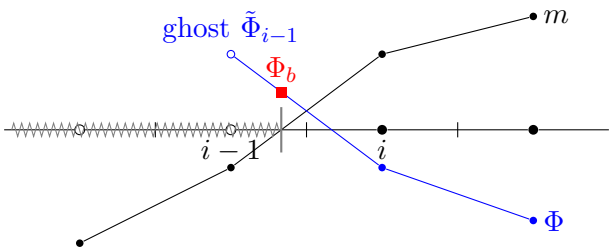

Figure 5: Reconstruction of ghost cell values across boundaries. Cells with negative mask (white boxes at centers) are outside of the domain and do not carry a valid Φ value. The computation of the Laplacian at cell i requires the use of a ghost value $\tilde{\Phi}_{i-1}$ which is obtained from Φ_i by linearly extrapolating the boundary condition. In accordance with the level set idea, the boundary is positioned at the point where m crosses zero, which is found by linearly interpolating the mask values.

4. First-order boundary reconstruction and the "small islands" problem

We see in Figure 7 that a different type of boundary can lead to a catastrophic divergence of our multigrid scheme (the blue line shows the residual norm evolution in case of second-order boundary reconstruction). This rather complex boundary condition is typical of AMR grids in cosmological simulation [27]. It features many small disconnected domains that cluster in a large central region with many "holes". Successive coarsening of the grid resolution lead in this case to a loss of boundary conditions, especially when "holes" or "small islands" are present. Indeed, in this case, the finer boundary small scale features cannot be represented anymore on coarser grids. Figure 6 illustrates such a case, for which the fine grid still has a cell with a negative mask value, so that the $\Phi = 0$ boundary condition still applies, but the coarse grid has cells with only positive values. Because of this topological change

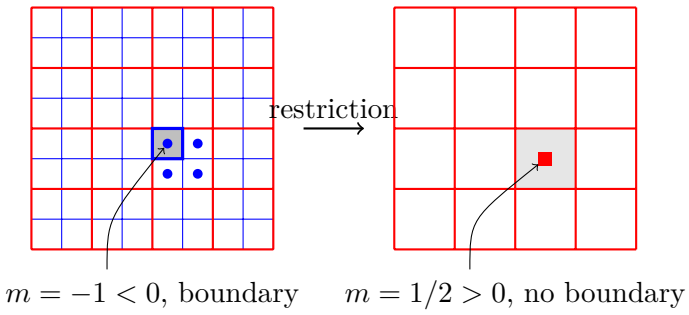

Figure 6: Coarsening (reduction) may cause loss of boundary conditions. The Dirichlet BCs around an isolated cell with $m = -1$ among a sea of $m = 1$ cells are lost after one coarsening step, as the resulting coarse mask is positive everywhere.

in the boundary representation, the coarse and fine solutions of the Poisson equation become significantly different. Spurious eigenmodes associated to this loss of constraints at the boundary are not damped quickly enough by the smoothing operator at the fine level: this leads to slower convergence rates and, in some case, to catastrophic divergence (see Fig. 7).

One previously proposed solution to this problem is the subtraction of the divergent modes introduced by these small islands, for example by recombining the multigrid iterants [3]. This however turns out to be too computationally and memory intensive for us to use in the Poisson solver of the RAMSES code: it requires storing a large number of previous solutions. This recombination method also tends to perform worse when the number of islands increases, which would likely render it useless in the case of the complex grid structure encountered in cosmological simulations.

Another proposed solution is to stop coarsening the residual in the multigrid hierarchy whenever such a problem occurs during multigrid coarsening

[10]. This degrades the performance of the multigrid method, since large scale modes are no longer damped efficiently. In the case of very complex boundaries with small islands, like in Figure 7, the level at which the hierarchy has to be truncated is so close to the finest level that the whole multigrid approach breaks down.

In order to overcome these limitations, McCorquodale et al. [16] proposed to keep track of the boundary representation at the fine level, and modify the Laplacian on coarse grids, whenever the operator stencil crossed this fine boundary. Stencil nodes which cannot be reached from the stencil center by a straight segment without crossing the boundary are excluded, and an asymmetric stencil is used, chosen in function of the local configuration of the interface.

We propose here a simpler approach, based on the observation that these small islands correspond to local minima in the color function m_i (or equivalently in the distance function if needed). The averaging scheme will tend to smooth these extrema, resulting in the disparition of the negative values (see Fig. 6) and in the apparition of spurious boundary conditions. In analogy to traditional high-order numerical schemes for hyperbolic systems of conservation laws [29, 5], we solve the problem by switching *globally* to a first-order boundary reconstruction scheme. We impose the $\Phi = 0$ constraint at the coarse level on the grid point nearest[2] to the interface: in practice, for each cell for which the mask value $m_i < 1$, we impose $\Phi_i = 0$. The simple averaging restriction for the mask ensures that cell i has $m_i < 1$ if and only

[2]The so-called NGP scheme for Nearest Grid Point.

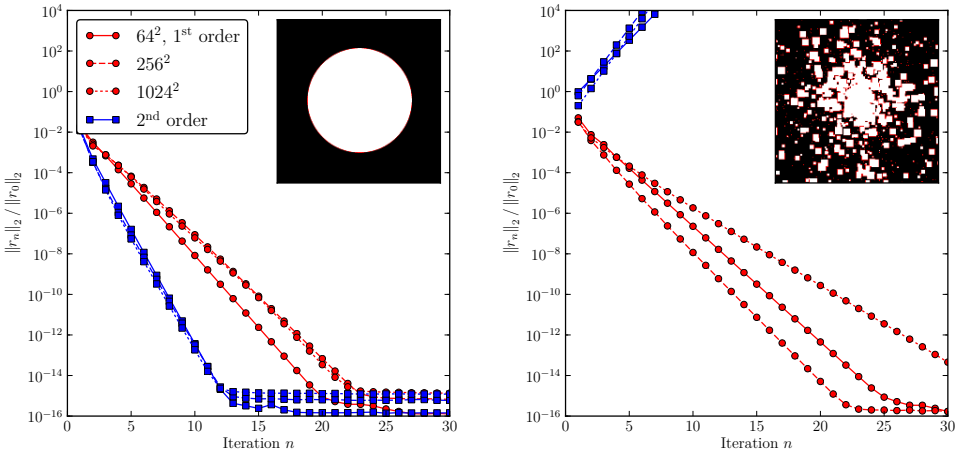

Figure 7: L_2 norm of the residual as a function of iteration number in our multigrid Poisson solver for both presented boundary capturing schemes. For each domain shape shown in the insets, the residual for both 2^{nd} order (in blue) and 1^{st} order (in red) boundary capturing schemes is presented for a 64^2 grid up to a 1024^2 grid.

if it has a non–zero intersection with the boundary.

We have tested our first-order boundary reconstruction scheme on 2 extreme types of boundary conditions : a simple disk with no hole or island, and a complex clustered grid, typical of AMR cosmological simulations. We have compared its convergence properties to the second-order reconstruction scheme in Figure 7. In the disk case, the simple topology of the boundary allows us to take full advantage of the second-order reconstruction of domain boundaries, resulting in fast convergence rates (in blue), totally independent of the problem size, This shows that our overall scheme features a true "textbook" multigrid convergence. Using the first-order boundary reconstruction, we degrade the convergence rate significantly, and it now depends slightly on the problem size. This is expected since the position of the boundary

at two consecutive multigrid levels can differ by half a cell width, and this first-order error in the boundary positioning limits the accuracy of the coarse correction.

In the complex boundary case, with many holes and islands, our second–order boundary reconstruction fails, as we can see from the divergent behavior of the residual norm in Figure 7. We interpret this catastrophic behavior as follows: topological changes in the boundary representation at coarse levels result in spurious long–range mode that are not damped by the smoothing operators at finer levels. If we use first–order boundary reconstruction, we observe that the convergence is restored: although first–order reconstruction of the boundary introduces small-scale errors close to the boundary, these errors are efficiently damped by the smoothing operator at finer levels. Note however that the convergence rate is slower than in the disc case, and that the convergence rate now depends on the grid size. A similar effect was observed in [6, 19].

Note that in our present implementation, we have to decide beforehand to which order we wish to reconstruct the boundary. We have tried to implement an adaptive algorithm, for which the reconstruction scheme is adapted locally to the topological changes of the boundary, but no satisfactory results were found. This then raises the question: how do we choose between first or second order reconstruction, when solving the Poisson equation on an arbitrarily complex grid? It may be possible to decide from a topological analysis of the grid (such as its Genus and the number of connected components), but this proves difficult to achieve in practice.

We have opted for a pragmatic approach, and decide at run time of which

scheme to employ for each AMR level independently. When we start solving the Poisson equation on a given level, we first try using the second-order reconstruction. We monitor the convergence rate during the iterations, and if it becomes slower than a fixed threshold, i.e. if $\|r_{n+1}\| / \|r_n\| > \eta$ with typically $\eta = 0.5$, we switch to the first-order scheme for that level only and for the next 10 solves. With our current implementation, if the AMR grid is really complex (with small islands), the solver only wastes a couple of iterations before deciding on which of first or second–order gives the best convergence rate. This works very well in practice, even in cosmological simulations featuring clusters and filaments, as only a few intermediate AMR levels exhibit very complex topologies.

5. Performance tests

5.1. Cosmological simulations

The new Poisson solver has by now been used in RAMSES in a variety of simulations. We present timings for our new multigrid (MG) solver on Table 1, together with corresponding timings for the conjugate gradient (CG) method, used here as a reference example of traditional iterative solvers. The timings represent the total wall clock time required to solve for the Poisson equation on the whole AMR grid, for a residual L2 norm reduction factor of 10^{-3}. For the reference timings, we have set the initial guess of the solution to zero everywhere on the grid, for both solvers.

The tests were run on the CEA/CCRT Platine computer, consisting of BULL NovaScale 3045 units totaling 932 nodes, networked by an InfiniBand

Type of AMR grid	**PE**	**Resolution**	**Ncell**	**CG (s)**	**MG (s)**
A. Cartesian	32	256^3 + 0 levels	19M	52	4.1
B. Zoom-in	64	128^3 + 3 levels	47M	160	15
C. Cosmology	1024	1024^3 + 5 levels	4.8G	2750	1070

Table 1: Poisson solver timings on three reference simulations, for conjugate gradient (CG) and multigrid (MG)

interconnect. Each node hosts 4 Intel® Itanium® 2 dual-core 1.6 GHz processors sharing 24 Gb of RAM.

Simulation A (a cosmological simulation at very early time) has a base 256^3 grid, with no additional level of refinement, on 32 processors. Simulation B is a "zoom" simulation, with a base resolution of 128^3 and successive forced refinements up to a 1024^3-equivalent zoom-in area. The cosmological computation follows the formation of a dense dark matter halo within the focus area. Simulation C is a clustered, $50h^{-1}$Mpc box cosmological simulation at $z = 1$, with 1024^3 particles. At this level of nonlinearity, the finer AMR levels are extremely clustered, while the intermediate levels exhibit a complex topology as they follow the intermediate-density structures (walls, filaments and clumps).

All timings of Table 1 show a strong performance advantage of the multigrid method over the conjugate gradient. Additionally, during the time evolution of cosmological simulations, we witnessed that the multigrid solver convergence times are much more predictable and consistent across different runs than the conjugate gradient. We attribute this effect to the fact that the multigrid performance only depends on the topology of the grid, which

changes progressively during the simulations, whereas the conjugate gravient is very sensitive to the quality of the first guess.

In the context of one-way interface solvers on AMR grids, we can improve significantly the performance of the conjugate gradient solver by computing a first guess solution based on the coarser level solution. We choose to linearly interpolate the initial guess of Φ at level ℓ from the solution at the coarser level $\ell - 1$, which has just been computed. This "multilevel" approach guarantees an initial guess of reasonable quality at a small extra cost — the cost of interpolating the solution from the coarser AMR level to the finer level. Note that for our CG implementation, iterations only take place at the finest level, and are therefore *not* multigrid-accelerated. We now discuss timings for our 3 test simulations using this improved CG solver. Since new multigrid timings have shown to be practically unchanged down to 2 digits, when using this new first guess, we only give new timings for the conjugate gradient solver.

Simulation A features a CG time between 5.8 and 23 seconds. The conjugate gradient timings are particularly erratic on the first few timesteps, because the first cosmological structures form at very small scale in the middle of a nearly uniform density field; therefore such small scale features are not accurately represented on the first guess obtained by interpolating the coarse solution. The number of iterations necessary to reach a given residual reduction factor is therefore high at the start of the simulation, before decreasing significantly as larger structures grow. In any case, the multigrid method performs significantly better than the conjugate gradient on cartesian grids, even though the conjugate gradient benefits dramatically from an

optimal first guess, and has less overall overhead.

On simulation B, the CG solver with the new first guess takes 140 seconds. The almost tenfold performance gain of the multigrid algorithm over the conjugate gradient can be explained by the fast evolution of the matter density at the coarse levels in the early stages of the simulation. Since coarse levels use a bigger time step than finer levels because of adaptive timestepping, the potential on coarse cells—which is interpolated as a first guess for the finer potentials—is updated less frequently. The finer first guesses thus tend to be out of synchronization with the real solution, resulting in additional conjugate gradient iterations. The multigrid algorithm is much less sensitive to first guess quality, resulting in a significant advantage over the conjugate gradient. This situation shows the real strength of the new solver in the context of adaptive time stepping.

Finally, in the case of simulation C, the CG solver runs for 850 seconds, about 20% less than multigrid. Decomposing the solver time by level shows that the MG solver spends most of its time dealing with very fine and very clustered grids, at the finest end of the AMR structure. This is easily understood, as this type of grid geometries represent a worst-case scenario for the multigrid solver in terms of small islands, forcing intermediate AMR levels into the slightly less efficient 1^{st} order capturing mode. Moreover, at this stage of the simulation, the timestep is usually extremely small, which is beneficial to the conjugate gradient as discussed in the case of simulation B. This result suggests using a hybrid scheme in practice, where one uses the new multigrid method for most levels of the AMR hierarchy except the finest ones, which can be handled by the CG solver. This method has been

implemented in RAMSES.

5.2. Effect of N_{cycles}

The N_{cycles} parameter controls how many multigrid iterations are performed when computing a coarse correction, at any level of the MG hierarchy. More iterations usually yield a better correction (and less multigrid iterations at the finest level before reaching tolerance), but significantly increase the cost of each iteration. One must therefore find an appropriate tradeoff.

Performing more than 2 or 3 cycles is usually not desirable, because the coarse problem is only itself an approximation of the fine correction problem. We have studied $N_{cycles} = 1$ ("V-cycles"), $N_{cycles} = 2$ ("W-cycles") and $N_{cycles} = 3$. We have measured the residual reduction rate and the total solver time to solve to a given accuracy (10^{-10} in our tests) for a simple disk domain as shown on Figure 7.

Typical results are presented on Figure 8. The first conclusion is that V-cycles are very sensitive to the chosen boundary reconstruction scheme. First-order boundary reconstruction is clearly detrimental to the convergence rate, though it does ensure convergence of grids with small islands. Interestingly enough, schedules with $N_{cycles} \geq 2$ (like W-cycles) appear to be insensitive to the order of the boundary reconstruction scheme. This suggests that first-order boundary reconstruction used in conjunction with W-cycles would ensure a convergence of the solver as fast as the second-order scheme. Since W-cycles have more costly iterations, we see in Figure 8 that this additional cost translates however into a longer overall time. For most astrophysical applications we have explored with the RAMSES code, we have found that V-cycles perform generally better than W-cycles. This can however depend

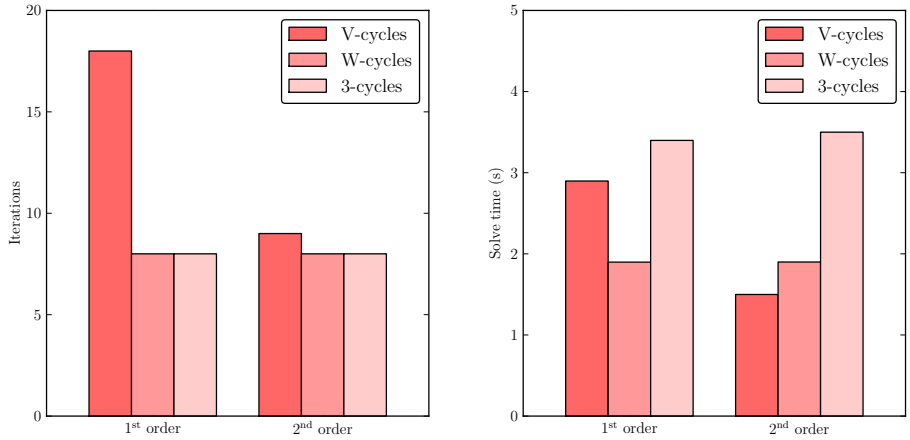

Figure 8: Effect of the multigrid schedule on the solver, for a simple test case where second-order boundary reconstruction does converge.

on the actual implementation of the solver, and the kind of grid geometries arising in other applications.

5.3. Parallel computing

The progress of distributed memory architectures over vector supercomputers has led to a regain of interest in iterative methods, as direct global solvers such as FFT are particularly expensive in terms of inter-process communications. Iterative methods can often be adapted to distributed memory architectures, with little modification, and therefore remain simple to implement, while requiring limited communications.

A broad class of parallelization techniques for physical problems consists of splitting the computational domain into subregions, which are each managed and updated by a dedicated computing core. Such *spatial domain decomposition* techniques rely on the ability to update each CPU independently first, then address the couplings between different domains.

In RAMSES, this last step is implemented using buffer regions (see Fig. 9). Each computing core manages its own cells, but also posesses a local copy of cells from other neighbouring CPUs which are needed for local computation. These buffer cells need to be updated after every iteration of the various iterative solvers used in the code. The update operation is done by communicating the updated values of the buffer cells from the CPUs which own them to the buffer regions in other processors. Therefore, any CPU only communicates with its direct neighbours, and the number of neighbours usually remains small. Moreover, the number of buffer cells scales only as a surface term, limiting the transfer to computation volume ratio. This is in contrast with the FFT, which requires a full transpose of the grid, and global all-to-all communications. In our multigrid scheme, we need to communicate both the solution and the residual for each the buffer cell between every Gauss-Seidel sweep, and also after each restriction or prolongation operation.

We have performed weak and strong scaling timings of our multigrid Poisson solver in RAMSES. The strong scaling test case is a simple 256^3 cosmological simulation without any refinement (Cartesian grid), starting at 4 processes up to 512 processes. The weak scaling test scales from 256^3 with 4 processes to 2048^3 with 2048 processes. The test results are presented on Figure 10. We see that our parallel efficiency degrades down to 50% when we reach 32^3 cells per processor. Beyond this limit, we spend more time communicating data than updating the solution during each Gauss–Seidel sweep. We could improve our scaling by a factor of 2 by hiding surface cells communications by inner cells computations. The weak scaling tests shows that if we keep the computational load above 64^3 cell per processor, the

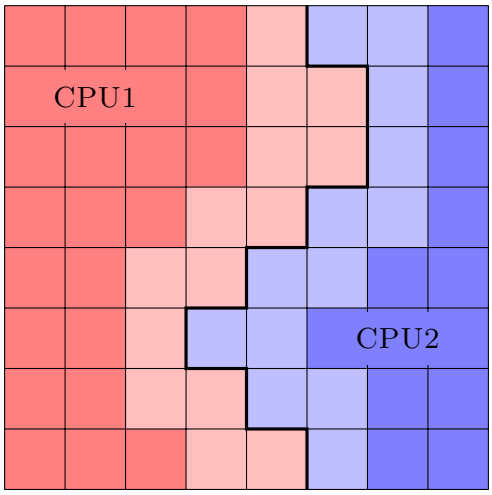

Figure 9: Domain decomposition and buffer regions as used in RAMSES, in particular for the Poisson solver. The thick black line marks the boundary of the spatial domain decomposition between CPU 1 and CPU 2. CPU 1 owns all the red cells, while CPU 2 owns all the blue cells. In order to perform an update, each CPU needs the values of the fields in the immediate exterior vincinity of its domain (light red and light blue for CPUs 2 and 1 respectively). These buffer cells are updated after each iteration of the various solvers using inter-processor communications.

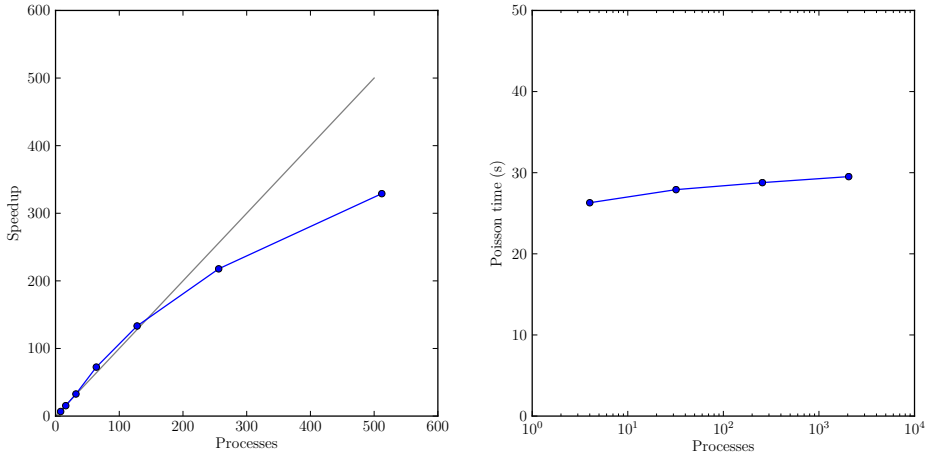

Figure 10: Strong (left panel) and weak (right panel) scaling test of our multigrid Poisson solver for a 256^3 Cartesian grid.

scaling is almost perfect. This rule of thumb applies also for more complex grid geometry, although load balancing can degrade significantly in case of very deep adaptive time step strategies.

6. Conclusion

We have presented a simple and efficient multigrid algorithm for solving the Poisson equation on irregular domains defined on a regular Cartesian grid. This kind of problem frequently arises in AMR codes using a one-way interface strategy, and the method is therefore of particular interest for astrophysical applications with multiple adaptive time steps. Using a second-order boundary reconstruction scheme, we have shown that our multigrid scheme features optimal convergence properties. Since we use a multigrid hierarchy orthogonal to the actual AMR grid, our memory requirements are minimal. In case of particularly complex boundary conditions, we have observed that

our scheme fails to converge, an issue previously identified as the "small island" problem. We have designed a simple fix to this problem, degrading our boundary reconstruction accuracy to first-order. We have implemented this new technique in the RAMSES code, using a simple algorithm to determine "on the fly" which reconstruction technique should be used. We have measured significant performance gains over our standard conjugate gradient solver, especially in the case of cosmological "zoom-in" simulations, where large fully–refined domain are present in the AMR grid. This simple and efficient multigrid solver could also be used for incompressible flow solvers, in presence of multiphase fluids or complex embedded solid boundaries.

Acknowledgments

This work was granted access to the HPC resources of CCRT under the allocation 2009-SAP2191 made by GENCI (Grand Equipement National de Calcul Intensif).

References

[1] Almgren, A., Bell, J., Colella, P., Marthaler, T., 1997. A cartesian grid projection method for the incompressible euler equations in complex geometries. SIAM Journal of Scientific Computing 18, 1289–1309.

[2] Brandt, A., 1977. Multi-level adaptive solutions to boundary-value problems. Mathematics of Computation 31, 333–390.

[3] Brandt, A., Mikulinsky, V., 1995. On recombining iterants in multigrid algorithms and problems with small islands. SIAM Journal of Scientific Computing 16 (1), 20–28.

[4] Cheng, H., Greengard, L., Rokhlin, V., 1999. A fast adaptive multipole algorithm in three dimensions. Journal of Computational Physics 155, 468–498.

[5] Colella, P., Woodward, P. R., Sep. 1984. The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations. Journal of Computational Physics 54, 174–201.

[6] Day, M. S., Colella, P., Lijewski, M. J., Rendleman, C. A., Marcus, D. L., 1998. Embedded boundary algorithms for solving the poisson equation. . ., 41811.

[7] Gibou, F., Fedkiw, R. P., Cheng, L.-T., Kang, M., Feb. 2002. A second-order-accurate symmetric discretization of the Poisson equation on irregular domains. Journal of Computational Physics 176, 205–227.

[8] Huang, J., Greengard, L., 2000. A fast direct solver for elliptic partial differential equations on adaptively refined meshes. SIAM Journal of Scientific Computing 21.

[9] Jessop, C., Duncan, M., Chau, W. Y., Dec. 1994. Multigrid methods for N-body gravitational systems. Journal of Computational Physics 115, 339–351.

[10] Johansen, H., Colella, P., 1998. A cartesian grid embedded boundary method for Poisson's equation on irregular domains. Journal of Computational Physics 147 (1), 60–85.

[11] Khokhlov, A., Jul. 1998. Fully Threaded Tree Algorithms for Adaptive Refinement Fluid Dynamics Simulations. Journal of Computational Physics 143, 519–543.

[12] Knebe, A., Gibson, B. K., Feb. 2004. Galactic haloes in MONDian cosmological simulations. MNRAS347, 1055–1064.

[13] Kravtsov, A. V., Klypin, A. A., Khokhlov, A. M., Jul. 1997. Adaptive Refinement Tree: A new high-resolution N-body code for cosmological simulations. ApJS111, 73–+.

[14] Liu, X., Fedkiw, R. P., Kang, M., May 2000. A Boundary Condition Capturing Method for Poisson's Equation on Irregular Domains. Journal of Computational Physics 160, 151–178.

[15] Llinares, C., Zhao, H. S., Knebe, A., Apr. 2009. Physics of Galactic Colliders: High-Speed Satellites in ΛCDM Versus Mondian Cosmology. ApJ695, L145–L148.

[16] McCorquodale, P., Colella, P., Johansen, H., 2001. A cartesian grid embedded boundary method for the heat equation on irregular domains. Journal of Computational Physics 173 (2), 620–635.

[17] Miniati, F., Colella, P., Nov. 2007. Block structured adaptive mesh and time refinement for hybrid, hyperbolic + N-body systems. Journal of Computational Physics 227, 400–430.

[18] Osher, S., Fedkiw, R. P., May 2001. Level Set Methods: An Overview and Some Recent Results. Journal of Computational Physics 169, 463–502.

[19] Popinet, S., Sep. 2003. Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries. Journal of Computational Physics 190, 572–600.

[20] Press, W. H., Flannery, B. P., Teukolsky, S. A., Vetterling, W. T., 1992. Numerical Recipes in C: The Art of Scientific Computing. Cambridge University Press.

[21] Press, W. H., Teukolsky, S. A., Vetterling, W. T., Flannery, B. P., 1992. Numerical recipes in FORTRAN. The art of scientific computing.

[22] Ricker, P. M., May 2008. A Direct Multigrid Poisson Solver for Oct-Tree Adaptive Meshes. ApJS176, 293–300.

[23] Rudman, M., Aug. 1998. A volume-tracking method for incompressible multifluid flows with large density variations. International Journal for Numerical Methods in Fluids 28, 357–378.

[24] Saad, Y., 2003. Iterative Methods for Sparse Linear Systems. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[25] Sussman, M., Almgren, A. S., Bell, J. B., Colella, P., Howell, L. H., Welcome, M. L., Jan. 1999. An Adaptive Level Set Approach for Incompressible Two-Phase Flows. Journal of Computational Physics 148, 81–124.

[26] Sussman, M., Smereka, P., Osher, S., Sep. 1994. A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow. Journal of Computational Physics 114, 146–159.

[27] Teyssier, R., Apr. 2002. Cosmological hydrodynamics with adaptive mesh refinement. a new high resolution code called RAMSES. A&A385, 337–364.

[28] Tiret, O., Combes, F., Mar. 2007. Evolution of spiral galaxies in modified gravity. A&A464, 517–528.

[29] van Leer, B., Mar. 1984. On the relation between the Upwind-Differencing schemes of godunov, Engquist–Osher and roe. SIAM Journal on Scientific and Statistical Computing 5 (1), 1–20.
URL `http://link.aip.org/link/?SCE/5/1/1`

[30] Wesseling, P., 1992. An introduction to multigrid methods. Wiley.

[31] Ye, T., Mittal, R., Udaykumar, H. S., Shyy, W., 1999. An accurate cartesian grid method for viscous incompressible flows with complex immersed boundaries. Journal of Computational Physics 156, 209–240.