View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Copenhagen University Research Information System

UNIVERSITY OF COPENHAGEN

A simple and efficient solver for self-gravity in the DISPATCH astrophysical simulation framework

Ramsey, J P; Haugbølle, T; Nordlund, Å

Published in:
Journal of Physics: Conference Series

DOI:
10.1088/1742-6596/1031/1/012021

Publication date:
2018

Document license:
CC BY

Citation for published version (APA):
Ramsey, J. P., Haugbølle, T., & Nordlund, Å. (2018). A simple and efficient solver for self-gravity in the DISPATCH astrophysical simulation framework. In *Journal of Physics: Conference Series* (Vol. 1031). Journal of Physics: Conference Series https://doi.org/10.1088/1742-6596/1031/1/012021

Download date: 09. apr.. 2020

Journal of Physics: Conference Series

PAPER • **OPEN ACCESS**

A simple and efficient solver for self-gravity in the DISPATCH astrophysical simulation framework

To cite this article: J P Ramsey *et al* 2018 *J. Phys.: Conf. Ser.* **1031** 012021

View the article online for updates and enhancements.

Related content

- A CFD framework for offshore and onshore wind farm simulation
  Matias Avila, Abel Gargallo-Peiró and Arnau Folch
- DDG4 A Simulation Framework based on the DD4hep Detector Description Toolkit
  M. Frank, F. Gaede, N. Nikiforou et al.
- AN IMPROVED MULTIPOLE APPROXIMATION FOR SELF-GRAVITY AND ITS IMPORTANCE FOR CORE-COLLAPSE SUPERNOVA SIMULATIONS
  Sean M. Couch, Carlo Graziani and Norbert Flocke

IOP ebooks™

Bringing you innovative digital publishing with leading voices to create your essential collection of books in STEM research.

Start exploring the collection - download the first chapter of every title for free.

This content was downloaded from IP address 130.225.178.2 on 15/01/2019 at 15:36

IOP Conf. Series: Journal of Physics: Conf. Series **1031** (2018) 012021 doi:10.1088/1742-6596/1031/1/012021

A simple and efficient solver for self-gravity in the DISPATCH astrophysical simulation framework

J P Ramsey and T Haugbølle and Å Nordlund

Centre for Star and Planet Formation, the Niels Bohr Institute and the Natural History Museum of Denmark, University of Copenhagen, Øster Voldgade 5-7, DK-1350 Copenhagen, Denmark

E-mail: jramsey@nbi.ku.dk; haugboel@nbi.ku.dk; aake@nbi.ku.dk

Abstract. We describe a simple and effective algorithm for solving Poisson's equation in the context of self-gravity within the DISPATCH astrophysical fluid framework. The algorithm leverages the fact that DISPATCH stores multiple time slices and uses asynchronous time-stepping to produce a scheme that does not require any explicit global communication or sub-cycling, only the normal, local communication between patches and the iterative solution to Poisson's equation. We demonstrate that the implementation is suitable for both collections of patches of a single resolution and for hierarchies of adaptively resolved patches. Benchmarks are presented that demonstrate the accuracy, effectiveness and efficiency of the scheme.

1. Introduction

Gravity is at the root of the formation of cosmological structure, galaxies, stars and planets. It also dominates the dynamics of many astrophysical systems, not least planets in orbit around a star or the spiral arms of a galaxy. Determining the strength of gravity at any given location requires solving for the gravitational potential due to a particular mass distribution via Poisson's equation, a second-order partial differential, elliptical equation with Dirichlet boundary conditions. In general, the solution to Poisson's equation for an arbitrary mass distribution is not analytically tractable. As such, given the importance of gravity in the universe, the accurate (and efficient) numerical solution of Poisson's equation has received considerable attention in computational astrophysics.

There exists many different approaches to numerically solve Poisson's equation. The most popular, basic methods include fast Fourier transforms (FFT), direct N-body calculations [1], multi-pole expansions [2], relaxation methods (e.g. Gauss-Seidel and successive over relaxation; SOR) and multi-grid techniques [3]. Many problems in astrophysics, meanwhile, have an intrinsically large dynamic spatial range (e.g. galaxy and star formation), which makes these problems expensive to simulate. In order to keep the cost of simulations down, considerable effort has been spent developing adaptive techniques that focus numerical resolution only when and where it is needed (e.g. adaptive mesh refinement; AMR; [4]). Naturally, the combination of these adaptive techniques with the solution of Poisson's equation is desirable and indeed required in certain contexts. Examples of adaptive resolution techniques for Poisson's equation include variants or combinations of the above listed methods (e.g. FFT + multi-grid; [5]) that have been modified for a specific adaptive framework or data structure (e.g. AMR multi-grid; [6, 7, 8]; tree gravity [9]; sink particles [10]).

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

What all of these methods share in common is the need to globally synchronise information whenever a solution to Poisson's equation is undertaken. Indeed, in a large-scale, domain-decomposed simulation this usually requires substantial communication of information across domains (most commonly, MPI ranks[1]) as the Poisson solver iterates, and a synchronisation point whereby all domains agree on the global solution. Not surprisingly, the details of how the Poisson equation is solved and how updated information is propagated between domains can have a substantial effect on the performance and scalability of a code or simulation framework.

In this contribution, we present a simple and efficient method for solving self-gravity in the DISPATCH framework. DISPATCH is a new framework for grid-based astrophysical fluid simulations [11, 12, 13] which incorporates novel algorithms to produce a framework with potentially unlimited parallel scaling and excellent performance even at the node level. The salient features of DISPATCH can briefly be described as follows: 1) We decompose the computational domain into small, semi-independent patches, which readily fit into cache and can be efficiently vectorised; 2) we store multiple time slices of the conservative variables, which then permits us to use individual, asynchronous time-stepping in each patch; 3) we ensure there are many more patches than hardware threads (e.g. OpenMP) available, thus ensuring threads keep busy; 4) work scheduling is task-based and patches most in need of updating are picked first; 5) each MPI rank only communicates with its nearest neighbours, whether they be geometric neighbours or causally-coupled ones; 6) load and communication imbalances are handled by trading entire patches to MPI neighbours on-demand and continuously. Taken together, these features prevent a very small fraction of the computational domain from dominating the cost of a simulation, they permit continuous and low-cost load balancing and they result in linear OpenMP and MPI scaling to $> 10^5$ cores [11].

The modular design of DISPATCH allows for different solvers to be used depending on the set of partial differential equations being solved and the needs of the user. Currently available solvers include the RAMSES hydrodynamical (HD) and magnetohydrodynamical (MHD) solvers [14, 15], STAGGER solvers of 2nd, 4th and 6th order [16], the ZEUS-3D solver [17, 18] and even the PPcode solver for particle-in-cell applications [19]. In this work, we choose to apply the hydrodynamical variants of the ZEUS-3D and 2nd-order STAGGER solvers. The self-gravity algorithm presented herein is, however, independent of the choice of hydrodynamical solver.

What makes this particular algorithm novel is that it exploits the multiple time slots and asynchronous time-stepping features of DISPATCH, plus the existing infrastructure for prolongation and restriction, leading to a simple but effective means for solving Poisson's equation. Indeed, the algorithm requires no global communication or blocking synchronisation across domains.

2. Solving Poisson's Equation

Poisson's equation for Newtonian gravity is a prototypical example of a second-order partial differential, elliptic equation constrained by boundary conditions:

$$\nabla^2 \Phi = 4\pi G \rho, \tag{1}$$

where ρ is the gas density, G is the universal gravitational constant, Φ is the gravitational potential and $\nabla^2 = \nabla \cdot \nabla$ is the Laplacian operator. The term on the right hand side is commonly referred to as the "source" and, in the case of gravity, this is the distribution of mass.

In periodic cases, one must offset the source term by its global average. Doing this renormalises the problem, effectively cancelling the effect of the mass in the periodic neighbours, which could otherwise lead to a divergent solution.

[1] MPI: "Message Passing Interface"; currently, the most common approach to distributed scientific computing.

To wit, the form of the Poisson equation that must be solved in the periodic case is:

$$\nabla^2\Phi = 4\pi G(\rho - \langle\rho\rangle) = 4\pi G\rho', \tag{2}$$

where $\langle\rho\rangle$ is the average value of the density over the entire domain. It is sufficient to determine this value from the initial conditions and therefore does not require a global summation. In cases with fixed boundary conditions, there is no need for such a renormalisation, and would indeed result in an incorrect solution.

As already discussed, many different methods are available to solve eq. 2. Direct methods (e.g. FFT, N-body) would be preferred because they provide an "exact" solution, but these approaches are often impractical or even intractable in many cases. This is particularly true when the mass is discretised onto a hierarchy of grids, e.g., when using AMR. Instead, iterative relaxation methods have proven to be the most cost-effective in this situation. We choose to solve eq. 2 iteratively via the form:

$$4\pi GR = \nabla^2\Phi - 4\pi G\rho', \tag{3}$$

where R is the residual and is a direct measure of the adjustment in mass that would be required to make Φ the exact solution for a given set of boundary conditions. While the goal is to reduce the residual to zero iteratively, in practice, one instead aims to decrease its magnitude below a given tolerance level. In our case, the quantity used to measure convergence is:

$$\varepsilon = \frac{R}{\rho + \rho_0}, \tag{4}$$

where ρ_0 is a 'floor' value characterising the desired tolerance in regions with very low density. When $\varepsilon < \varepsilon_0$, typically set to 10^{-4}, convergence is signalled and the Poisson solver exits.

For solving Poisson's equation on a single DISPATCH patch, we have implemented both successive over-relaxation (SOR) with Chebyshev acceleration [3] and a conjugate gradient (CG) solver [20] with preconditioner [21]. One could, of course, also apply Gauss-Seidel relaxation, but since SOR is always faster there is no reason to. An important consideration for SOR is that a red-black scheme should be employed, otherwise initial symmetries that exist in a problem will not be preserved. In fact, preserving exact symmetry, in any case, may require the use of an odd number of points in non-periodic problems, and an even number of points in periodic problems.

The cost per point of the SOR and CG (with preconditioner) solvers are relatively similar for a typical number of cells per patch. For example, the average cost per cell update on an Intel Core i5 Haswell-based laptop is ∼0.015 μs/point/iteration. Of course, as the number of cells increases, the CG solver becomes more efficient than SOR (c.f. the tests of Section 4.1).

Once we have obtained the gravitational potential, the gravitational acceleration, $\vec{g} = \nabla\Phi$, can be calculated and applied as a source term within the hydrodynamical solver. The Poisson solver is implemented independently of the hydrodynamical solver; the feedback of gravity on the gas is accomplished via accelerations. For example, when using 2nd-order staggered differences (e.g. ZEUS-3D and STAGGER solvers), the x-component of the gravitational acceleration at the cell face indexed by (i, j, k) may be written as:

$$g_{x,i} = -\partial_x\Phi = -\frac{\Phi_i - \Phi_{i-1}}{\Delta x} + \mathcal{O}(\Delta x^2), \tag{5}$$

where Δx is the grid spacing in the x-direction. Since the gravitational potential is zone-centred, this simple form gives accelerations that are face-centred and co-spatial with staggered velocity/momentum components. In the case of a centred solver, such as RAMSES, we instead apply the following centred difference formula [22]:

$$g_{x,i} = -\partial_x\Phi = -\frac{4}{3}\frac{\Phi_{i+1} - \Phi_{i-1}}{2\Delta x} + \frac{1}{3}\frac{\Phi_{i+2} - \Phi_{i-2}}{4\Delta x} + \mathcal{O}(\Delta x^4). \tag{6}$$

3. Multiple patch and multiple resolution algorithm

When applying adaptive grid techniques to the solution of the (M)HD equations (e.g. [4]), small errors are readily introduced at interfaces between grids of different resolutions. In the case of quantities which are relatively smooth, these errors can be compensated for by applying so-called "flux corrections". In the more extreme case that a discontinuous wave passes through a change in resolution, however, errors are unavoidably generated (see, e.g., Sect. 7.1 of [18]). However, these errors are rarely ever consequential in complex, large-scale numerical simulations and, as such, they are frequently tolerated.

Although self-gravity is clearly a global problem, the gravitational potential is typically smooth and slowly-varying slowly in time. With this in mind, in DISPATCH, we choose to take a different approach to self-gravity: We solve Poisson's equation locally, on a per patch basis, and then export the updated information to neighbouring patches. This potentially admits errors after a single call to the Poisson solver but, independent of the strategy to reduce such errors, their magnitude can always be monitored by computing the residual (eq. 3). If the errors become large, it is typically because the density distribution changes too much per time step, and a simple remedy is then to reduce the local time step. This effectively leads to more Poisson solutions per unit code time, and hence more frequent exchange of updated information between neighbours. Importantly, frequent exchange between neighbours propagates information about changes in the mass distribution across the domain at a rate proportional to the inverse of the (M)HD time step. Furthermore, Poisson's equation is iterated until convergence on all patches within the domain.

The first step to solving for gravity in DISPATCH is restriction and prolongation, which we term here as 'downloading'. Before the Poisson solver is invoked, up-to-date information about the mass distribution and gravitational potential in neighbouring patches is downloaded. Depending on the details of the patch arrangement, this will either take the form of temporal and spatial interpolation of guard zone data from neighbours (i.e. prolongation) or the conservative averaging of finer resolution data from overlying child patches to the current patch (i.e. restriction). In the first regard, we employ linear interpolation in both time and space; higher order interpolation in space is possible, but more costly, and linear interpolation has proven more than sufficient for the current application. In the second regard, we use linear averaging. Via experimentation, we have found that using the finest resolution data available when prolonging values of Φ provides the best precision. The downloading procedure for the gravitational potential is the same as for any of the hydrodynamical variables. Indeed, that is one of advantages of our scheme: no special considerations are needed for the gravitational potential relative to the other variables.

The next step is to actually solve Poisson's equation on a patch (see above). This is accomplished after the download, but before the (M)HD equations are solved. The resulting gravitational potential is then applied as a source term in the (M)HD solver (eqs. 5, 6).

A critical part of the algorithm is to now call the Poisson solver again for the same patch but using the recently time-updated values of the density. This solution typically only takes 1 or 2 iterations to converge and the resulting gravitational potential is placed into the *next* time slot. When using local time-stepping, a neighbour patch will generally not be at exactly the same time as the current patch and will therefore require some interpolation or extrapolation in time to fill its ghost cells before it can take a step forward in time; by providing a forward *prediction* for Φ, the neighbour can fill its ghost cells and proceed with its update.

Since we have access to multiple time slices of Φ for each patch in DISPATCH (see [11]), it is possible to make a (e.g. linear, quadratic, or higher order) extrapolation in time for Φ from these time slices that can then serve as an initial guess for the next invocation of the Poisson solver on a given patch. Experimentation shows that a linear extrapolation in time for a single, isolated patch can reduce the number of iterations in a subsequent Poisson solve by up to a

factor of three. Quadratic temporal extrapolation, meanwhile, typically reduces the iteration count by a further ∼10%. This feature is most effective when the potential changes significantly between time steps (e.g. a quickly moving massive clump). In certain pathological cases, this feature results in non-convergence of the Poisson solver, however, and thus it is implemented as an optional feature.

Finally, in our algorithm, if an initial guess for Φ is not available, a short "burn-in" phase may be desirable to obtain a globally consistent Φ before the dynamics are activated. This is accomplished by running DISPATCH with the dynamics deactivated and only solving for the gravitational potential (e.g. Section 4.1). The procedure followed is the same as when dynamics are activated, and so no special modifications are necessary for this phase. Furthermore, it remains a local process; no explicit global communication of information is required.

4. Verification

4.1. Static gravitational potential

As a first test of our self-gravity algorithm, we investigate a slightly modified version of the static, analytical gravitational potential of [22]. This test employs a 2D, analytic, static and purely radial gravitational potential given by:

$$\Phi_{\mathrm{e}}(r) = \ln\left[\left(\frac{r}{r_0}\right)^2 + 1\right] + 1, \tag{7}$$

where $r^2 = x^2 + y^2$ is the radial coordinate and r_0 is the parameter that determines the radial "concentration" of the potential. We have slightly modified this expression relative to [22] so that, at the centre of the domain ($r = 0$), it has a non-zero value. This has no effect on the resulting density distribution or the quality of the solution, but simplifies the comparison of numerical and analytical solutions near $r = 0$.

The resulting density distribution, obtained by applying Equation (1) to Φ_{e}, is:

$$\rho_{\mathrm{e}}(r) = \frac{4r_0^2}{\left(r^2 + r_0^2\right)^2}. \tag{8}$$

The test is set up by initialising the density to eq. (8) everywhere and setting the gravitational potential to eq. (7) in the boundaries of the computational domain, but zero elsewhere. We set the domain size to $(x, y) \in [-0.5, 0.5]$ and the concentration parameter to $r_0 = 0.1$. We then proceed to solve for Φ iteratively on every patch. A convergence tolerance of 10^{-4} was employed for the Poisson solver. As described above, in between calls to the Poisson solver, updated boundary information is retrieved from neighbouring patches. We alternate between Poisson solves and downloading until the global value of the $L_\infty(\Phi)$-norm converges. As this is a time-independent test, time extrapolation in the gravitational potential was not used.

Furthermore, because this test does not require hydrodynamics or even a time step, we have exploited the modular and hierarchical structure of DISPATCH to implement a shell 'solver' for Poisson's equation that leverages the framework without needing to include any code for a hydrodynamical solver.

We consider three configurations: First, a single patch with varying resolution, from 16^2 to 2048^2 cells. Second, the domain is split into either 3^2 or 9^2 patches of equal resolution and tests run for 4^2 to 256^2 cells per patch. Third, a set of nested, refined grids with a maximum refinement level[2] of 3 or 4 but a constant number of cells per patch. Each nested level is comprised of 3^2 patches centred on the origin and refined relative to the previous level by a

[2] The "root" grid, which encompasses the entire domain, has level = 1.

factor of three. Tests were run with 5^2 up to 129^2 cells per patch. In the nested patches set up, a maximum refinement level of 3 (4) is equivalent to covering the domain with 9^2 (27^2) patches.

Figure 1 shows the resulting L_∞-norm values in both the gravitational potential and acceleration for the three scenarios at different effective resolutions: a single patch of uniform resolution (circles), a domain sub-divided into multiple patches (squares), and nested sets of patches of increasing resolution (triangles). With respect to the gravitational potential, all three scenarios show that the L_∞-norm decreases proportional to the number of cells squared, demonstrating that our algorithm converges at second-order, and that the L_∞-norm does not vary significantly between configurations. As for the gravitational acceleration, $\vec{g}$ our algorithm retains second-order convergence when all patches are of the same resolution. However, once patches of different resolution are introduced, we observe that the convergence in the gravitational acceleration degrades to first-order. As noted in other works (e.g. [22]), this is a result of using linear interpolation for filling guard zones at changes in resolution, and adopting a higher-order interpolation in Φ would improve the convergence rate.

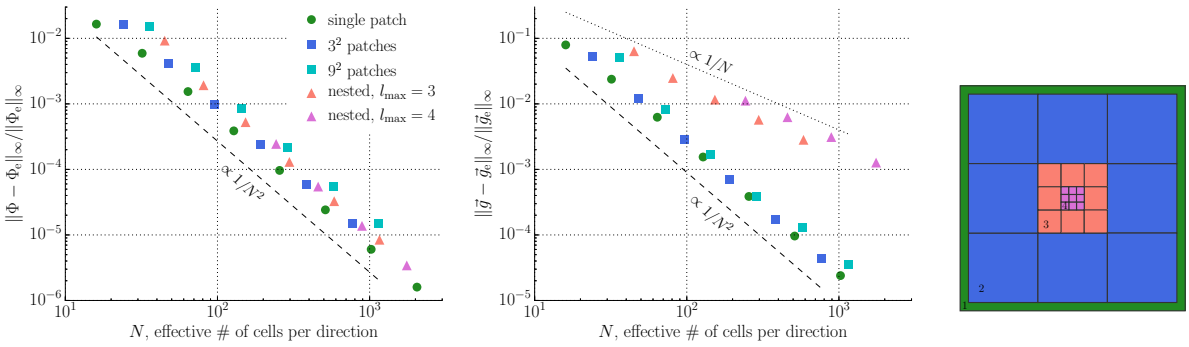

Figure 1. *Left:* The L_∞-norm in the gravitational potential for the static potential problem of Sect. 4.1 as a function of (effective) resolution. The straight line denotes an L_∞-norm which is proportional to $1/N^2$, where N is the effective number of cells. *Middle:* The L_∞-norm in the gravitational acceleration given by eq. 5, as a function of resolution. *Right:* A schematic representation of the nested patch hierarchy used for $l_{\max} = 4$. Numbers denote level; black lines denote patch boundaries.

4.2. The Truelove experiments

The collapse of a gaseous clump under its own gravity is critical to understanding star, disk and planet formation and its dynamics has therefore been well-studied under a wide range of conditions (e.g. [23, 24, 25, 26, 27, 10]).

The criterion for runaway gravitational collapse is originally due to Jeans [28], who found that density perturbations on scales larger than:

$$\lambda_{\rm J} = \sqrt{\frac{\pi c_{\rm s}^2}{G\rho}} \tag{9}$$

are unstable. With respect to numerical simulations of gravitational collapse, [29, 6] further determined that, in order to avoid numerical fragmentation, one must ensure that the so-called Jeans number,

$$J = \frac{\Delta x}{\lambda_{\rm J}}, \tag{10}$$

where Δx is the cell size, is kept below a value of $J_{\max} = 1/4$.

To test the combination of self-gravity and hydrodynamics, we follow [6] and place a massive clump of radius R_0 that is, by definition, gravitationally unstable, at the centre of a 3D Cartesian domain. The density of the clump is given by $\rho_0(1 + A\cos(m\vartheta))$ for $r \leq R_0$ and by ρ_0/χ_ρ, otherwise, where ρ_0 is the uniform density of the clump, A is the amplitude of a possible perturbation, m is the mode of the perturbation, ϑ is the polar angle in the $x - y$ plane and $\chi_\rho = 100$ is the density contrast between clump and ambient medium. In addition to the density, these clumps are characterised by their mass, $M = 1M_\odot$, and their thermal and rotational energies with respect to gravity:

$$\alpha = \frac{E_{\text{therm}}}{E_{\text{grav}}} = \frac{5}{2}\left(\frac{3}{4\pi\rho_0 M^2}\right)^{1/3}\frac{c_{\text{iso}}^2}{G}; \tag{11}$$

$$\beta = \frac{E_{\text{rot}}}{E_{\text{grav}}} = \frac{1}{4\pi}\frac{\Omega^2}{G\rho_0}. \tag{12}$$

For a given initial clump density, the gravitational free-fall time,

$$t_{\text{ff}} = \sqrt{\frac{3\pi}{32G\rho_0}}, \tag{13}$$

can be used to estimate the time scale of collapse.

The criteria for dynamic refinement applied here is that the Jeans number (eq. 10) must remain below 1/4 in every cell. If this criteria is violated, a patch is split into 3^3 child patches, each with the same number of cells as the parent, and the parent data is prolonged onto the new patches.

We attempted three different configurations drawn directly from [6]: i) a non-rotating clump of initially uniform density; ii) a uniformly rotating clump of uniform density; iii) a slightly perturbed (10% amplitude in the $m = 2$ mode) but uniformly rotating clump. The parameters employed can be found in Table 1.

We employed both the ZEUS-3D solver and the 2nd-order STAGGER solver. In the current configuration, both employ the internal energy equation and use a Courant factor of 0.2. The solver-specific artificial viscosity parameters employed for ZEUS-3D are `qlin` = 0.1 and `qcon` = 1 [17], while the relevant hyper-viscosity parameters for STAGGER are $\nu_1 = 0.2$ (viscous stress tensor coefficient), $\nu_2 = 1.0$ (artificial pressure coefficient) [16]. We furthermore used 17^3 cells per patch, a refinement ratio of three and a convergence tolerance of 10^{-4} for the Poisson solver.

As can be seen in Figure 2, our benchmark results compare favourably with those of [6]. There are differences in the rate of collapse, but the density and velocity structures are very similar. In the uniform, non-rotating case, by $t = 1.036t_{\text{ff}}$, the density has increased from its initial value by 4.3 orders of magnitude to $\rho_{\max} = 10^{-10.7}$ g cm^{-3} and we reach a maximum effective resolution of 459 cells ($l_{\max} = 4$). Similarly, for the uniform, rotating case, the density increases by 5.2 orders of magnitude to $\rho_{\max} = 10^{-12.6}$ g cm^{-3} after $1.239t_{\text{ff}}$ and we reach an effective resolution of 1377 cells ($l_{\max} = 5$). Finally, in the 10% perturbation + rotating test, by $1.321t_{\text{ff}}$, the maximum density has reached $\rho_{\max} = 10^{-11.4}$ g cm^{-3}, an increase of 6.04 orders of magnitude, and we attain an effective resolution of 4131 cells ($l_{\max} = 6$). The bottom right panel of Figure 2 shows a slice of the x-component of the gravitational acceleration through the center ($y = z = 0$) of the perturbed Truelove experiment. The profile appears smooth across patch boundaries, indicating that our algorithm is approximating the gravitational force reasonably. However, we do observe changes in the slope of the acceleration (at the <10% level) at jumps in resolution. Therefore, we are currently investigating the use of higher-order interpolations for Φ in the boundaries.

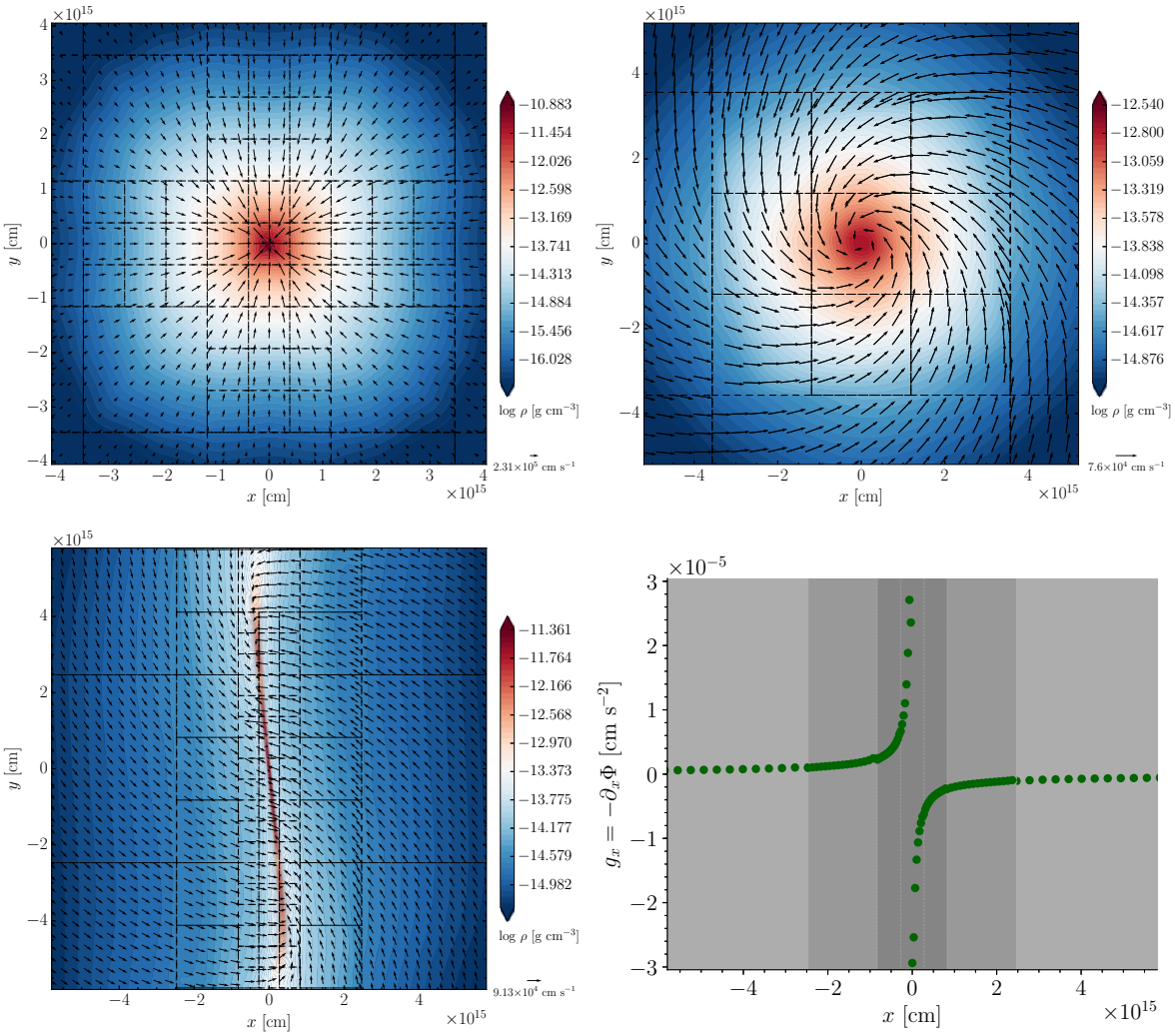

Figure 2. Truelove collapse tests. Plotted is an $x - y$ slice of the logarithm of density at $z = 0.0$. Black arrows denote velocity vectors. Dashed lines indicate patch boundaries. *Top left:* Uniform, non-rotating collapse at $t = 1.036t_{\rm ff}$. *Top right:* Uniform, rotating collapse at $t = 1.239t_{\rm ff}$. *Bottom left:* Perturbed ($m = 2$, 10% amplitude), rotating collapse at $t = 1.321t_{\rm ff}$. *Bottom right:* The x-component of the gravitational acceleration (eq. 5) through $y = z = 0.0$ in the perturbed Truelove experiment. Successively darker shading indicates levels 4–6 while vertical dashed lines denote patch boundaries.

Differences in our results with respect to [6] can be attributed to two factors: i) We are employing fixed rather than periodic boundary conditions and ii) we are using a box size of $8R_0$ versus $4R_0$. The absence of 'mirror' images due to periodic boundaries, combined with a larger box size, implies that our set up should collapse somewhat faster than the simulations presented in [6]. Indeed, with the exception of the uniform, rotating collapse, our simulations attain higher densities than at the times stated in [6].

5. Summary

Herein, we have presented a method for solving Poisson's equation on adaptive resolution grids for self-gravity that does not require any explicit global communication. The approach is

	Non-rotating	Rotating	Rotating + perturbed
ρ_0 [g cm^{-3}]	10^{-15}	$10^{-17.9}$	$10^{-17.4}$
α	0.0475	0.54	0.26
β	0	0.08	0.16
$\chi_p = \frac{P_{\rm clump}}{P_{\rm ambient}}$	1	1	10
R_0 [cm]	7.80×10^{15}	7.23×10^{16}	5.0×10^{16}
$t_{\rm ff}$ [s]	6.65×10^{10}	1.87×10^{12}	1.05×10^{12}

Table 1. Parameters and derived values used in the Truelove tests. In all cases, we employ a box size of $8R_0$, $\gamma = 1.0001$, $\chi_\rho = \rho_{\rm clump}/\rho_{\rm ambient} = 100$ and a cloud mass of $1M_\odot$. See the text for additional details.

built around the concept of multiple individual patches that only exchange information via prolongation and restriction operations; each patch solves gravity locally and then exchanges updated information with its neighbours. The approach exploits the local time-stepping and time slice storage features of DISPATCH to provide time interpolated and extrapolated values of the gravitational potential. There is also the option to increase efficiency by using forward time prediction and by using OpenMP to solve Poisson's equation simultaneously on more than one patch.

Using the static potential benchmarks of [22] and the gravitational collapse benchmarks of [6], we have demonstrated that Poisson's equation can be efficiently solved on a collection of semi-independent patches, and that a global communication or synchronisation step is not required to obtain an accurate solution.

In the Truelove collapse experiments, the DISPATCH self-gravity algorithm accounts for roughly 20% of the total cost during a simulation. Slightly less than half of the remaining cost is due to the hydrodynamical solver, while ∼25-30% is due to downloading. A portion of the downloading cost originates with the self-gravity algorithm because we make an intermediate download after the (M)HD step, but before the predictive solution of Poisson's equation; in experiments without self-gravity, the relative cost of downloading is reduced to ∼15%.

The first scientific application of this self-gravity algorithm is towards models of planet formation in protoplanetary disks. More specifically, we are currently investigating the self-gravity of combined gas+dust fluids and their potential for gravitational instability. We are also currently implementing additional experiments to test this algorithm under a wider range of conditions including, for example, an equal-mass binary with a moving centre-of-mass. The results of these benchmarks will be featured in future work. Finally, DISPATCH, and the self-gravity algorithm presented here, will be made open-source and publicly available in the near future.

Acknowledgments

The work of TH was supported by the Sapere Aude program of the Danish Council for Independent Research (DFF). The work of ÅN was supported by grant 1323-00199B from the DFF. The Centre for Star and Planet Formation is funded by the Danish National Research Foundation (DNRF97). Storage and computing resources at the University of Copenhagen HPC centre, funded in part by Villum Fonden (VKR023406), were used to carry out some of the simulations presented here.

References

[1] Evrard A E 1988 MNRAS **235** 911–934

[2] Couch S M, Graziani C and Flocke N 2013 *The Astrophysical Journal* **778** 181
[3] Press W H, Teukolsky S A, Vetterling W T and Flannery B P 1992 *Numerical recipes in FORTRAN. The art of scientific computing*
[4] Berger M J and Colella P 1989 *Journal of Computational Physics* **82** 64–84
[5] Collins D C, Xu H, Norman M L, Li H and Li S 2010 ApJS **186** 308–333 (*Preprint* 0902.2594)
[6] Truelove J K, Klein R I, McKee C F, Holliman II J H, Howell L H, Greenough J A and Woods D T 1998 ApJ **495** 821–852
[7] Ricker P M 2008 ApJS **176** 293-300 (*Preprint* 0710.4397)
[8] Almgren A S, Beckner V E, Bell J B, Day M S, Howell L H, Joggerst C C, Lijewski M J, Nonaka A, Singer M and Zingale M 2010 ApJ **715** 1221–1238 (*Preprint* 1005.0114)
[9] Wünsch R, Walch S, Whitworth A P and Dinnbier F 2017 *ArXiv e-prints* (*Preprint* 1708.06142)
[10] Federrath C, Banerjee R, Clark P C and Klessen R S 2010 ApJ **713** 269–290 (*Preprint* 1001.4456)
[11] Nordlund Å, Ramsey J P, Popovas A and Küffmeier M 2018 MNRAS (*Preprint* 1705.10774)
[12] Popovas A, Nordlund Å, Ramsey J P and Ormel C W 2018 *ArXiv e-prints* (*Preprint* 1801.07707)
[13] Ramsey J P, Nordlund Å and Popovas A *in preparation*
[14] Teyssier R 2002 A&A **385** 337–364 (*Preprint* astro-ph/0111367)
[15] Fromang S, Hennebelle P and Teyssier R 2006 A&A **457** 371–384 (*Preprint* astro-ph/0607230)
[16] Kritsuk A G, Nordlund Å, Collins D, Padoan P, Norman M L, Abel T, Banerjee R, Federrath C, Flock M, Lee D, Li P S, Müller W C, Teyssier R, Ustyugov S D, Vogel C and Xu H 2011 ApJ **737** 13 (*Preprint* 1103.5525)
[17] Clarke D A 2010 ApJS **187** 119–134
[18] Ramsey J P, Clarke D A and Men'shchikov A B 2012 ApJS **199** 13 (*Preprint* 1112.4390)
[19] Haugbølle T, Frederiksen J T and Nordlund A 2013 *Physics of Plasmas* **20** 062904 (*Preprint* 1211.4575)
[20] Ghysels P and Vanroose W 2014 *Parallel Computing* **40** 224 – 238 ISSN 0167-8191 7th Workshop on Parallel Matrix Algorithms and Applications URL http://www.sciencedirect.com/science/article/pii/S0167819113000719
[21] Ament M, Knittel G, Weiskopf D and Strasser W 2010 *Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on* (IEEE) pp 583–592
[22] Guillet T and Teyssier R 2011 *Journal of Computational Physics* **230** 4756–4771 (*Preprint* 1104.1703)
[23] Hoyle F 1953 ApJ **118** 513
[24] Hunter C 1962 ApJ **136** 594
[25] Nordlund Å, Haugbølle T, Küffmeier M, Padoan P and Vasileiades A 2014 *Exploring the Formation and Evolution of Planetary Systems* (*IAU Symposium* vol 299) ed Booth M, Matthews B C and Graham J R pp 131–135
[26] Burkert A and Bodenheimer P 1993 MNRAS **264** 798
[27] Bate M R and Burkert A 1997 MNRAS **288** 1060–1072
[28] Jeans J H 1902 *Philosophical Transactions of the Royal Society of London Series A* **199** 1–53
[29] Truelove J K, Klein R I, McKee C F, Holliman II J H, Howell L H and Greenough J A 1997 ApJ **489** L179–L183