Monthly Notices of the ROYAL ASTRONOMICAL SOCIETY

MNRAS **473,** 1603–1632 (2018) doi:10.1093/mnras/stx2405
Advance Access publication 2017 September 18

GANDALF – Graphical Astrophysics code for *N*-body Dynamics And Lagrangian Fluids

D. A. Hubber,[1,2★] G. P. Rosotti[3] and R. A. Booth[3]

[1]*Universitats-Sternwarte München, Scheinerstraße 1, D-81679 München, Germany*
[2]*Excellence Cluster Universe, Boltzmannstr. 2, D-85748 Garching, Germany*
[3]*Institute of Astronomy, Madingley Rd, Cambridge CB3 0HA, UK*

Accepted 2017 September 14. Received 2017 September 14; in original form 2017 June 29

ABSTRACT

GANDALF is a new hydrodynamics and *N*-body dynamics code designed for investigating planet formation, star formation and star cluster problems. GANDALF is written in C++, parallelized with both OPENMP and MPI and contains a PYTHON library for analysis and visualization. The code has been written with a fully object-oriented approach to easily allow user-defined implementations of physics modules or other algorithms. The code currently contains implementations of smoothed particle hydrodynamics, meshless finite-volume and collisional *N*-body schemes, but can easily be adapted to include additional particle schemes. We present in this paper the details of its implementation, results from the test suite, serial and parallel performance results and discuss the planned future development. The code is freely available as an open source project on the code-hosting website github at https://github.com/gandalfcode/gandalf and is available under the GPLv2 license.

Key words: hydrodynamics – methods: numerical.

1 INTRODUCTION

Numerical simulations are becoming increasingly more important in modern astrophysics research. They allow us to study systems where analytical solutions do not exist and explore the complex (non-linear) interplay due to the multiple physical processes that are normally present in astrophysical problems. In recent years, more attention has been given to exploring which algorithms give the most accurate and reliable results and comparing different algorithms to one another, as well as the development of brand new or hybrid algorithms. While many specialist codes exist with single hard-wired implementations of particular physical processes (e.g. hydrodynamics), the current desire for flexibility in algorithm choice is not always fulfilled with a single code and may often require using multiple codes for a single project.

In this paper, we present GANDALF (Graphical Astrophysics code for N-body Dynamics And Lagrangian Fluids; Hubber & Rosotti 2016), a new multipurpose hydrodynamics, *N*-body and analysis code. GANDALF has been designed with Star and Planet Formation problems in mind, but with the flexibility to be extended with different physics algorithms to simulate other kinds of astrophysical problems.

GANDALF was developed with a heavy object-oriented design philosophy in order to improve code maintainability and simplify the process of implementing new features in the future. C++ was chosen as the main development language as a low-level, high-performance computing (HPC) object-oriented language that is easy to bind with other (often C-based) external libraries and can easily be parallelized with both OPENMP and MPI (either individually or combined with a hybrid OPENMP–MPI approach). GANDALF also contains an optional PYTHON library, which can be used for analysis and visualization of whole simulations or single snapshots. It is also possible to generate initial conditions and set-up and run the simulation from a PYTHON script making it easier for users not accustomed with C++.

GANDALF contains implementations of two particle-based hydrodynamics schemes, smoothed particle hydrodynamics (SPH; e.g. Monaghan 1992) and the meshless finite-volume scheme (MFV; Lanson & Vila 2008; Gaburov & Nitadori 2011; Hopkins 2015). Many algorithms (e.g. gravity, the tree used for neighbour finding) are shared between the two implementations, minimizing the amount of code duplication. GANDALF also includes algorithms for collisional *N*-body dynamics.

This paper is structured as follows. In Section 2, we discuss the Hydrodynamical algorithms that we have implemented into GANDALF, including any differences from traditional implementations. In Section 3, we discuss our implementations of the collisional *N*-body and sink particle algorithms. In Section 4, we discuss other miscellaneous algorithms such as implementing boundary conditions and trees. In Section 5, we discuss the class structure of the code, how to add new classes on top of the existing framework, the PYTHON library and how it can be easily used to perform analysis and run the code. In Section 6, we present results from our test suite comparing all methods against each other and against other

★ E-mail: dhubber@usm.lmu.de

published codes. We also show the serial and parallel performance of the code. In Section 7, we discuss the performance and parallel scaling of the code, both with OPENMP and hybrid OPENMP/MPI. In Section 8, we briefly discuss ongoing work with the code and planned features for the future.

2 HYDRODYNAMICAL METHODS IN GANDALF

GANDALF solves the traditional Euler equations of hydrodynamics with additional physics terms such as gravitational accelerations. In Lagrangian form, these are

$$\frac{\mathrm{d}\rho}{\mathrm{d}t} = -\rho\,\nabla\cdot\boldsymbol{v} \tag{1}$$

$$\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t} = -\frac{\nabla P}{\rho} - \nabla\Phi \tag{2}$$

$$\frac{\mathrm{d}u}{\mathrm{d}t} = -\frac{P}{\rho}\nabla\cdot\boldsymbol{v} \tag{3}$$

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

where ρ is the fluid density, $\boldsymbol{v}$ is the fluid velocity, u is the specific internal, P the thermal pressure and Φ is the gravitational potential and G is Newton's constant.

GANDALF contains implementations of two particle-based hydrodynamical schemes that use the *smoothing kernel* as a fundamental quantity in solving the numerical form of these equations. The fluid properties of all particles are smoothed over a length-scale h, called the *smoothing length*, with a weighting function $W(\boldsymbol{r}, h)$ called the *kernel function*. Each particle occupies/influences a spherical volume called the *smoothing kernel* of total radius $\mathcal{R}\,h$. The fluid particles interact with neighbouring particles, i.e. particles whose smoothing kernels overlap, where the interaction is weighted somewhat by the kernel function. The exact details of how the smoothing kernel influences the hydrodynamical equations are explained in each scheme's implementation.

GANDALF contains two principal kernel functions which have a finite extent of $\mathcal{R}\,h$; (i) the M4 cubic spline kernel (Monaghan & Lattanzio 1985) with $\mathcal{R} = 2$ and (ii) the quintic spline kernel (Morris 1996) with $\mathcal{R} = 3$. The complete mathematical description of all these kernels, plus related derivative and integrated quantities, are given in appendix A of Hubber et al. (2011).

2.1 Smoothed particle hydrodynamics

SPH (Lucy 1977; Gingold & Monaghan 1977) is a popular Lagrangian hydrodynamics scheme that has been implemented in many astrophysical hydrodynamics codes, such as GADGET-2 (Springel 2005), VINE (Wetzstein et al. 2009), SEREN (Hubber et al. 2011) and PHANTOM (Price et al. 2017). The main advantages of SPH are (i) it is simple conceptually and to code, and (ii) its Lagrangian nature which provides various advantages over Eulerian methods, such as having an in-built adaptivity to the wide range of densities found in gravitational collapse problems, (iii) it can be derived from the Euler–Lagrange equations so is naturally conservative, (iv) it can be integrated with symplectic equations such as the Leapfrog resulting in good orbital conservation properties (e.g. angular momentum conservation) and (v) it can be easily coupled to the N-body equations of motion when including point gravitational sources (e.g. stars and planets). SPH has been derived in many mathematical forms, each with different assumptions, different integration variables or different methods of computing hydrodynamical quantities. GANDALF currently uses the standard conservative conservative 'grad-h' SPH following Springel & Hernquist (2002) and Price (2012), with the pressure–entropy scheme of Saitoh & Makino (2013) planned for the future.

2.1.1 Conservative 'grad-h' SPH

Conservative 'grad-h' SPH (Springel & Hernquist 2002) is one of the standard derivations of the SPH equations that is used in astrophysical codes, such as GADGET-2. The fluid equations are derived from Lagrangian mechanics and hence guarantee conservation of mass, momentum, angular momentum and energy to at least integration error. However, it should be noted that the use of the tree in calculating gravitational accelerations and block time-stepping algorithms introduces additional sources of error meaning 'perfect' conservation is not achieved in practice.

The algorithm described here is similar to that implemented in SEREN (Hubber et al. 2011). We first compute the density, ρ, and smoothing length, h of each SPH particle. The smoothed density for particle i is given by

$$\rho_i = \sum_{j=1}^{N} m_j\,W(\boldsymbol{r}_{ij}, h_i), \tag{5}$$

where $\boldsymbol{r}_{ij} = \boldsymbol{r}_i - \boldsymbol{r}_j$, $W(\boldsymbol{r}_{ij}, h_i)$ is the smoothing kernel and m_j is the mass of particle j. The density and smoothing length are related by the simple relation

$$h_i = \eta_{\text{SPH}}\left(\frac{m_i}{\rho_i}\right)^{\frac{1}{D}}, \tag{6}$$

where D is the dimensionality of the simulation and η_{SPH} is a dimensionless parameter that relates the smoothing length to the local interparticle spacing (default value $\eta_{\text{SPH}} = 1.2$). Since h and ρ depend on each other, we must iterate their values until equations (5) and (6) converge to some tolerance, usually to within about $\sim$1 per cent.

The SPH momentum equation is given by

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = -\sum_{j=1}^{N} m_j \left\{ \frac{P_i}{\Omega_i \rho_i^2}\nabla_i W(\boldsymbol{r}_{ij}, h_i) + \frac{P_j}{\Omega_j \rho_j^2}\nabla_i W(\boldsymbol{r}_{ij}, h_j) \right\}, \tag{7}$$

where $P_i = (\gamma - 1)\,\rho_i\,u_i$ is the thermal pressure, u_i is the specific internal energy, γ is the ratio of specific heats for an ideal gas, $\nabla_i W$ is the kernel gradient and

$$\Omega_i = 1 - \frac{\partial h_i}{\partial \rho_i}\sum_{j=1}^{N} m_j \frac{\partial W}{\partial h}(\boldsymbol{r}_{ij}, h_i) \tag{8}$$

is a dimensionless correction term that accounts for the spatial variability of h amongst its neighbouring particles.

If the temperature is not prescribed (e.g. by an isothermal equation of state; hereafter OS), we integrate an energy equation of the form

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = \frac{P_i}{\Omega_i \rho_i^2}\sum_{j=1}^{N} m_j\,\boldsymbol{v}_{ij}\cdot\nabla W_{ij}(\boldsymbol{r}_{ij}, h_i), \tag{9}$$

where $\boldsymbol{v}_{ij} = \boldsymbol{v}_i - \boldsymbol{v}_j$.

The SPH equations presented so far describe a fluid without dissipation, where the fluid quantities are always continuous. However, many astrophysical problems contain shocks, which lead to dissipation and need to be handled properly. We use the Monaghan (1997) formulation of artificial viscosity for shock-capturing,

$$\frac{\mathrm{d}\boldsymbol{v}_i}{\mathrm{d}t} = \sum_{j=1}^{N} \frac{m_j}{\overline{\rho}_{ij}} \left\{\alpha_{\text{AV}}\, v_{\text{SIG}} \mu_{ij}\right\} \overline{\nabla_i W}_{ij}, \tag{10}$$

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -\sum_{j=1}^{N} \frac{m_j}{\overline{\rho}_{ij}} \frac{\alpha_{\text{AV}}\, v_{\text{SIG}} \mu_{ij}^2}{2} \hat{\boldsymbol{r}}_{ij} \cdot \overline{\nabla_i W}_{ij} + \sum_{j=1}^{N} \frac{m_j}{\overline{\rho}_{ij}} \alpha_{\text{AC}}\, v'_{\text{SIG}} (u_i - u_j) \hat{\boldsymbol{r}}_{ij} \cdot \overline{\nabla_i W}_{ij}, \tag{11}$$

where α_{AV} and α_{AC} are constants of order unity that control the dissipation strength, v_{SIG} and v'_{SIG} are the signal speeds for artificial viscosity and conductivity respectively, $\hat{\boldsymbol{r}}_{ij} = \boldsymbol{r}_{ij}/|\boldsymbol{r}_{ij}|$ and $\overline{\nabla_i W}_{ij} = \frac{1}{2}\left(\nabla_i W(\boldsymbol{r}_{ij}, h_i) + \nabla_i W(\boldsymbol{r}_{ij}, h_j)\right)$ and $\mu_{ij} = \text{MIN}(0, \boldsymbol{v}_{ij} \cdot \boldsymbol{r}_{ij})$. For artificial viscosity, we use $v_{\text{SIG}} = c_i + c_j - \beta_{\text{AV}}\, \boldsymbol{v}_{ij} \cdot \hat{\boldsymbol{r}}_{ij}$, where c_i and c_j are the sound speeds of particles i and j, respectively, and $\beta_{\text{AV}} = 2\,\alpha_{\text{AV}}$. The signal speed for artificial conductivity is problem and physics dependent. By default, we chose the Wadsley, Veeravalli & Couchman (2008) prescription, where $v'_{\text{SIG}} = |\boldsymbol{v}_{ij} \cdot \hat{\boldsymbol{r}}_{ij}|$ although the Price (2008) conductivity, $v'_{\text{SIG}} = \sqrt{|P_i - P_j|/\overline{\rho}_{ij}}$, is also available in the code.

Since excessive dissipation is undesirable in hydrodynamical codes, we have implemented two artificial viscosity switches, Morris & Monaghan (1997) and Cullen & Dehnen (2010), in order to reduce the artificial viscosity as much as possible in regions away from shocks.

2.1.2 *Self-gravity*

Computing self-gravity in SPH can be done consistently by considering the continuous density field given by equation (5) in the Poisson equation (equation 4), instead of solving the N-body problem with each particle representing a discrete point mass (Price & Monaghan 2007). Deriving the equations of motion via Lagrangian mechanics leads to a conservative set of equations with self-gravity. The SPH gravitational acceleration is given by

$$\boldsymbol{g}_i = -G \sum_{j=1}^{N} m_j \frac{\phi'(\boldsymbol{r}_{ij}, h_i) + \phi'(\boldsymbol{r}_{ij}, h_j)}{2} \hat{\boldsymbol{r}}_{ij} - \frac{G}{2} \sum_{j=1}^{N} m_j \left\{ \frac{\zeta_{\text{i}}}{\Omega_{\text{i}}} \nabla W_i(\boldsymbol{r}_{ij}, h_i) + \frac{\zeta_{\text{j}}}{\Omega_{\text{j}}} \nabla W_i(\boldsymbol{r}_{ij}, h_j) \right\}, \tag{12}$$

where

$$\phi'(\boldsymbol{r}, h) = \frac{4\pi}{r^2} \int_0^r W(\boldsymbol{r}', h)\, r'^2\, \mathrm{d}r', \tag{13}$$

$$\zeta_{\text{i}} = \frac{\partial h_i}{\partial \rho_i} \sum_{j=1}^{N} m_{\text{j}} \frac{\partial \phi}{\partial h}(\boldsymbol{r}_{ij}, h_i), \tag{14}$$

and Ω_i is given by equation (8). $\phi'(\boldsymbol{r}, h)$ is often called the gravitational force or gravitational acceleration kernel and in effect calculates the gravitational force between SPH particles accounting for the smoothed density distribution. Similarly $\phi(\boldsymbol{r}, h)$ is the gravitational potential kernel which gives the smoothed gravitational potential. The ζ term is an additional term to Ω in accounting for the spatial variation of h for self-gravity.

2.1.3 *Time integration*

The SPH particles can be integrated with two related integration schemes, the Leapfrog kick-drift-kick (KDK) and the Leapfrog drift-kick-drift (DKD) schemes. Leapfrog schemes are symplectic schemes that exhibit accurate but stable integration of gravitational orbits. The KDK and DKD schemes are mathematically equivalent in case of global, constant time-steps with similar integration errors. However, in case of non-constant, individual time-steps (see Section 2.1.4), they can behave differently with different rates of error growth.

The position and velocity of a particle integrated with the KDK scheme is described by:

$$\boldsymbol{r}_i^{n+1} = \boldsymbol{r}_i^n + \boldsymbol{v}_i^n\, \Delta t + \frac{1}{2}\, \boldsymbol{a}_i^n\, \Delta t^2, \tag{15}$$

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n + \frac{1}{2}\left(\boldsymbol{a}_i^n + \boldsymbol{a}_i^{n+1}\right) \Delta t. \tag{16}$$

where Δt is the time-step. Although the acceleration appears twice in equation (16), we only compute it once per step, since the second acceleration term, $\mathbf{a}_i^{n+1}$, then becomes the first acceleration term for the next step.

In the DKD scheme, the updates to the positions and velocities are shifted by half a step:

$$\boldsymbol{r}_i^{n+1/2} = \boldsymbol{r}_i^n + \frac{1}{2}\boldsymbol{v}_i^n\, \Delta t, \tag{17}$$

$$\boldsymbol{v}_i^{n+1/2} = \boldsymbol{v}_i^n + \frac{1}{2}\, \boldsymbol{a}_i^{n-1/2}\, \Delta t, \tag{18}$$

$$\boldsymbol{v}_i^{n+1} = \boldsymbol{v}_i^n + \boldsymbol{a}_i^{n+1/2}\, \Delta t, \tag{19}$$

$$\boldsymbol{r}_i^{n+1} = \boldsymbol{r}_i^n + \frac{1}{2}\left(\boldsymbol{v}_i^n + \boldsymbol{v}_i^{n+1}\right) \Delta t. \tag{20}$$

The acceleration is computed only once, at the mid-point of the step. This requires in practice the DKD scheme to be computed as a two-step scheme, where particles are 'drifted' to the mid-point, the acceleration is computed and then the second half of the step is computed with the updated acceleration.

2.1.4 *Time-stepping*

All SPH schemes use a Courant–Friedrichs–Lewy (CFL)-like condition to compute the time-steps, $\Delta\, t_i$, of the form:

$$\Delta\, t_i = C_{\text{CFL}} \frac{h_i}{|v_{\text{sig,i}}|}, \tag{21}$$

where C_{CFL} is a dimensionless time-step multiplier (typically ~0.2) analogous to the Courant number in grid codes and the signal speed is

$$v_{\text{sig,i}} = \text{MAX}_j \left[c_i + c_j - \beta_{\text{AV}} \text{MIN}\left(0, \boldsymbol{v}_{ij} \cdot \hat{\boldsymbol{r}}_{ij}\right)\right]. \tag{22}$$

The signal velocity, $v_{\text{sig, i}}$, is the speed of propagation of information either through sound waves or translational velocity. In effect, equation (21) prevents information from crossing the smoothing kernel in a single time-step. The β_{AV} term exists to ensure strong shocks are captured adequately. If additional physics (e.g.

self-gravity) are employed, then we use a second criterion called the *acceleration condition*, i.e.

$$\Delta t_i = C_{\rm GRAV} \frac{h_i}{\sqrt{|\boldsymbol{a}_i|}} \qquad (23)$$

where $C_{\rm GRAV}$ is the dimensionless gravitational acceleration time-step multiplier (typically ∼0.5).

GANDALF uses a hierarchical block time-stepping scheme, similar to many other SPH and N-body codes like GADGET (Springel 2005) and NBODY6 (Aarseth 2003). The basic principle is that all time-steps are integer power-of-two multiples of some base time-step. In GANDALF, we fix the maximum time-step, $\Delta t_{\rm MAX}$, based on the time-steps available whenever the block time-steps are recomputed. By default, particles on the maximum time-step occupy level $l = 0$. Particles on higher levels l therefore occupy shorter time-steps, i.e.

$$\Delta t_l = \frac{\Delta t_{\rm MAX}}{2^l} \quad \text{where } l = 0, 1, 2, \ldots, l_{\rm MAX}. \qquad (24)$$

The time-step level for a given particle can to increase to an arbitrarily high number based on the given time-step criterion when required. However, the time-step level can only be reduced (i) by one level at a time, and (ii) when the new time-step level is correctly synchronized within the time-step hierarchy. When we have completed exactly one full time-step (on the lowest level) then all particles are synchronized and we can recompute the full time-step hierarchy again.

2.1.5 *Time-step limiter*

Block time-steps can introduce numerical artefacts in the results of a simulation if particles on very different time-steps are enabled to interact with each other. As an extreme example, particles in a cold, low-density region may have too long time-steps to react to the passage of a shock front. In GANDALF, we solve this problem similarly to Saitoh & Makino (2009), using a dual approach including both predictive and reactive components. In the predictive component, for each particle we keep track of the minimum time-step of its neighbours during the hydrodynamic force calculation. When assigning new time-steps to the particle, we ensure that the particle does not have a time-step more than a fixed factor longer than the minimum of its neighbours.

Additionally, we apply a reactive limiter for two reasons: (1) in the predictive component, we employ the *old* time-step of the neighbours. This does not guarantee that the *current* time-step obeys the level constraint, once the new time-step has been computed; (2) the time-step of the neighbours may reduce rapidly, e.g. due to an approaching shock. The reactive limiter works by checking whether the minimum time-step of its neighbours has reduced below the acceptable level. This is achieved by using a scatter gather operation, i.e. active particles inform their inactive neighbours of their time-step during the hydrodynamic force calculation. If the neighbour time-step criterion is found to be violated, the inactive particle's time-step is reduced and it becomes active as soon as its new time-step is synchronized with the time-step hierarchy.

We note that the predictive tree-based limiter based on Springel (2010) included in the meshless scheme Section 2.2.8 is not currently included in SPH. This is for pragmatic reasons: the primary advantage of the tree-based limiter is in maintaining exact conservation, which is already not maintained in SPH when block time-steps are used. Given that it is more expensive than the Saitoh & Makino (2009) type limiter (which already performs well) and can introduce unnecessarily small time-steps when gravity is included, we see no clear reason to use it in SPH. However, there is no fundamental reason it could not be easily added.

2.2 Meshless finite-volume scheme

The MFV scheme is a hydrodynamical scheme developed originally by Lanson & Vila (2008) and further developed for astrophysical applications by Gaburov & Nitadori (2011) and Hopkins (2015). The MFV scheme combines elements of both SPH and traditional finite-volume schemes (see Toro 1997) where freely moving particles interact and exchange mass, momentum and energy using a second-order Godunov approach but weighted with a smoothing kernel. We provide here a summary derivation presenting the main assumptions and equations as implemented in GANDALF.

2.2.1 *Volume discretization*

Similar to SPH, the MFV scheme uses the smoothing kernel to compute various smoothed quantities. We first compute the smoothing length of all the particles using the number density, n, instead of the mass density, ρ, i.e.

$$n_i = \sum_{j=1}^{N} W(\boldsymbol{r}_{ij}, h_i), \qquad (25)$$

where the n_i and h_i are related by

$$h_i = \eta_{\rm MFV} {n_i}^{-\frac{1}{D}}, \qquad (26)$$

and $\eta_{\rm MFV}$ is a dimensionless parameter analogous to $\eta_{\rm SPH}$ controlling the number of neighbours. For comparison with our results in Section 6, Hopkins (2015) presents results consistent with $\eta_{\rm MFV} = 1$ in 1D and 3D, but with a larger value $\eta_{\rm MFV} \approx 1.13$ in 2D.

In order to discretize the fluid on to a set of N particles, we must chose a method of partitioning the surrounding fluid volume between the different particles. Springel (2010) uses a Voronoi tessellation, which assigns a volume element to its nearest particle. Lanson & Vila (2008) instead use the SPH kernel to calculate the fraction of a volume element $d^\mu \boldsymbol{r}$ that is assigned to particle i, $\psi_i(\boldsymbol{r}) = W(\boldsymbol{r} - \boldsymbol{r}_i, h(\boldsymbol{r}))\, n(\boldsymbol{r})^{-1}$. In effect, the particles 'share' the surrounding volume in a similar way to SPH, resulting in an ensemble of overlapping 'fuzzy' volume elements (see fig. 1 of Hopkins 2015, for a useful visual aid). The partition function should be normalized such that $\sum_{\rm i} \psi_i(\boldsymbol{r}) = 1$ everywhere. The numerical volume of a particle becomes the integral of all the partial volume elements, i.e. $V_i = \int \psi_i(\boldsymbol{r})\, d^\mu \boldsymbol{r}$. Since this integral cannot be computed analytically for arbitrary particle distributions, we follow Hopkins (2015) in using the second-order accurate approximation, $V_i \sim n_i^{-1} = \left(\eta_{\rm MFV}/h_i\right)^D$.

2.2.2 *Gradient operators*

Instead of using an SPH-type gradient operator, Lanson & Vila (2008) use a least-squares matrix operator which is accurate to second order and is relatively inexpensive to calculate. In this form, the gradient of a general function f_i for particle i is given by:

$$(\nabla^\alpha f)_i = \sum_j \left(f_j - f_i\right)\, \tilde{\psi}_{\rm j}^{\alpha}(\boldsymbol{r}_{\rm i}), \qquad (27)$$

where j is the summation over all (overlapping) neighbouring particles,

$$\tilde{\psi}^{\alpha}_{\mathrm{j}}(\boldsymbol{r}_{\mathrm{i}}) = \sum_{\beta=1}^{\beta=\mu} \boldsymbol{B}^{\alpha\,\beta}_{\mathrm{i}} \left(\boldsymbol{r}_{\mathrm{j}} - \boldsymbol{r}_{\mathrm{i}}\right)^{\beta} \psi_{\mathrm{j}}(\boldsymbol{r}_i), \tag{28}$$

where $\boldsymbol{B} \equiv \boldsymbol{E}^{-1}$ and

$$\boldsymbol{E}^{\alpha\,\beta}_i = \sum_j \left(\boldsymbol{r}_{\mathrm{j}} - \boldsymbol{r}_{\mathrm{i}}\right)^{\alpha} \left(\boldsymbol{r}_{\mathrm{j}} - \boldsymbol{r}_{\mathrm{i}}\right)^{\beta} \psi_j(\boldsymbol{r}_{\mathrm{i}}). \tag{29}$$

In rare cases with pathological particle distributions, the gradient matrix can become close to singular resulting in poor gradient estimation. We follow Hopkins (2015) in using the condition number of the matrix $\boldsymbol{E}$ to detect the occurrence of bad gradients. When the condition number exceeds 100, we switch to a direct SPH estimate of the gradient. We use a constant exact linear gradient estimate (equation 72, Price 2012), which is equivalent to making the substitution

$$\tilde{\psi}^{\alpha}_j(r) \to V_{\mathrm{i}} \nabla^{\alpha}_i W_{ij}. \tag{30}$$

This substitution is made in both the gradient computation and the face area ($\boldsymbol{A}_{ij}$ below).

2.2.3 *The Euler equations in conserved form*

In traditional finite-volume schemes, each fluid cell is a discrete volume where mass, momentum and energy is exchanged at well-defined boundaries between adjacent cells. Traditional grid codes often use the vector $\boldsymbol{U} = (\rho_i, \rho_i \boldsymbol{v}_i, \rho_i e_i)$, which are the conserved quantities (mass, momentum and energy) per unit volume. Since the particle volume can change in MFV, the vector $\boldsymbol{Q} \equiv V\,\boldsymbol{U} = (m_i, m_i \boldsymbol{v}_i, E_i)$ is more appropriate. We also use the vector, $\boldsymbol{W} = (\rho_i, \boldsymbol{v}_i, P_i)$, which is the vector of primitive quantities given to the Riemann solver.

The general conservation laws for hydrodynamics in a moving frame $\boldsymbol{v}_{\mathrm{frame}}$ are

$$\frac{\partial \boldsymbol{U}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{U}) = \mathbf{S}, \tag{31}$$

where $\boldsymbol{U}$ is the vector of conserved variables, $\boldsymbol{F} = (\rho\,\boldsymbol{v}, \rho\,\boldsymbol{v} \otimes \boldsymbol{v} + P\,\mathcal{I}, (\rho\,e + P))$ is the flux matrix, $\mathcal{I}$ is the identity matrix and $\boldsymbol{S}$ is the source vector. Following Lanson & Vila (2008), who discretize these equations using Galerkin methods with the least-squares gradient operators (see Lanson & Vila 2008; Gaburov & Nitadori 2011; Hopkins 2015, for a complete derivation), we obtain the discrete Euler equations,

$$\frac{\mathrm{d}\boldsymbol{Q}_i}{\mathrm{d}t} + \sum_j \left[V_{\mathrm{i}} \boldsymbol{F}^{\alpha}_{\mathrm{i}} \tilde{\psi}^{\alpha}_{\mathrm{j}}(\boldsymbol{r}_{\mathrm{i}}) - V_{\mathrm{j}} \boldsymbol{F}^{\alpha}_{\mathrm{j}} \tilde{\psi}^{\alpha}_{\mathrm{i}}(\boldsymbol{r}_{\mathrm{j}}) \right] = \boldsymbol{S}_i\, V_{\mathrm{i}}. \tag{32}$$

By replacing the two individual fluxes, $\boldsymbol{F}_i$ and $\boldsymbol{F}_j$, with a single flux across the interface between the two particles, $\boldsymbol{F}_{ij}$, we obtain an exactly conservative scheme,

$$\frac{\mathrm{d}\boldsymbol{Q}_i}{\mathrm{d}t} + \sum_j \boldsymbol{F}_{\mathrm{ij}} \cdot \mathbf{A}_{\mathrm{ij}} = \boldsymbol{S}_i\, V_{\mathrm{i}}, \tag{33}$$

where the quantity $\boldsymbol{A}^{\alpha}_{ij} \equiv V_i \tilde{\psi}^{\alpha}_j(\boldsymbol{r}_i) - V_j \tilde{\psi}^{\alpha}_i(\boldsymbol{r}_j)$ is the effective area of the face between the particles.

The flux, $\boldsymbol{F}_{ij}$, can be found by solving 1D Riemann problems between pairs of particles, where we assume that the interface is aligned with the face vector, $\boldsymbol{A}_{ij}$. We have implemented two Riemann solvers in GANDALF; (i) the Exact Riemann solver for adiabatic gases (e.g. Toro 1997), and (ii) the HLLC approximate solver (Toro, Spruce & Speares 1994; Toro 1997), using the wave-speed estimate of Batten et al. (1996). For isothermal equations of state, the HLLC solver has been modified to ensure that the density is constant across the contact discontinuity as well as the pressure, while still resolving shear waves (e.g. Mignone 2007).

2.2.4 *Face reconstruction*

Equation (32) alone can be used to construct a first-order Godunov method without specifying any further information about the location of the face (although its velocity is still needed in a Lagrangian scheme, see below); however, such a scheme is quite diffusive. Second-order accuracy in space can be achieved following the standard Monotonic Upwind Scheme for Conservation Laws (MUSCL) approach (Lanson & Vila 2008; Gaburov & Nitadori 2011; Hopkins 2013), in which the primitive variables evaluated at the cell faces are passed to the Riemann solver (instead of using the particle values). We do this using a slope-limited linear reconstruction to avoid oscillations near discontinuities,

$$\boldsymbol{W}_i(\boldsymbol{r}_{\mathrm{face}}) = \boldsymbol{W}_i(\boldsymbol{r}_i) + \left(\boldsymbol{r}_{\mathrm{face}} - \boldsymbol{r}_i\right) \cdot (\chi \nabla \boldsymbol{W}), \tag{34}$$

where $\chi \nabla \boldsymbol{W}$ is the slope-limited gradient and $\nabla \boldsymbol{W}$ is computed using equation (27). The limiters are applied to each primitive variable independently. Both first- and second-order (linear) reconstructions are available, including a wide range of slope limiters such as those suggested by Springel (2010), Gaburov & Nitadori (2011), Heß & Springel (2010) and Hopkins (2015). The Total Variation Diminishing (TVD) limiter of Heß & Springel (2010) is the most diffusive, while the non-TVD limiters of Springel (2010) and Gaburov & Nitadori (2011) are the least diffusive. The limiter suggested by Hopkins (2015) falls in between.

In the second-order scheme, it is necessary to specify the location of the face. Following Lanson & Vila (2008) and Gaburov & Nitadori (2011), we take

$$\boldsymbol{r}_{\mathrm{face}} = \frac{1}{2} \left(\boldsymbol{r}_i + \boldsymbol{r}_j\right). \tag{35}$$

We are free to choose how the particle positions, $\boldsymbol{r}_i$, are updated. By default, we choose to move the particles at the local fluid velocity, $\boldsymbol{v}_i$. Finally, the Riemann problem must be solved in a frame that is consistent with the motion of the effective faces,[1] which moves along with the particles. An obvious choice for this is

$$\boldsymbol{v}_{\mathrm{face}} = \frac{\mathrm{d}\boldsymbol{r}_{\mathrm{face}}}{\mathrm{d}t} = \frac{1}{2} \left(\dot{\mathbf{r}}_i + \dot{\mathbf{r}}_j\right), \tag{36}$$

where $\dot{\mathbf{r}}_i$ are the velocities with which the particles are moved. This results in the MFV scheme as described by Hopkins (2015). Since this choice of face velocity may differ from the fluid velocity that comes from solving the Riemann problem, this results in a small amount of mass transferred between neighbouring particles. To construct a fully Lagrangian scheme, Hopkins (2015) suggests using the speed of the contact discontinuity in place of $\boldsymbol{v}_{\mathrm{face}}$. This approach is similar to that employed by Inutsuka (2002) and ensures that no mass is advected between neighbouring particles. Following Hopkins (2015), we refer to this modified scheme as the meshless finite-mass (MFM) scheme, which is used by default in GANDALF.

2.2.5 *Time integration: second-order MUSCL–Hancock*

To achieve second-order accurate integration in time, we employ an unsplit second-order MUSCL–Hancock scheme (van Leer 1979;

[1] This is done as described in appendix A of Hopkins (2015).

Toro 1997). The conserved quantities are updated according to $\boldsymbol{Q}_i^{n+1} = \boldsymbol{Q}_i^n + \sum_j \mathrm{d}\boldsymbol{Q}_{ij}$, where

$$\mathrm{d}\boldsymbol{Q}_{ij} = -\Delta t\, \boldsymbol{F}_{ij}^{n+1/2} \cdot \boldsymbol{A}_{\mathrm{ij}}, \tag{37}$$

and $\boldsymbol{F}_{ij}^{n+1/2}$ is the time-centred estimate of the flux. This is calculated by predicting the primitive quantities passed to the Riemann solver to the mid-point of the time-step along with reconstructing them to the faces. This is done via the Taylor-series expansion,

$$\boldsymbol{W}_i^{n+1/2} = \boldsymbol{W}_i^n + \left(\boldsymbol{r}_{\mathrm{face}} - \boldsymbol{r}_i\right) \cdot \nabla \boldsymbol{W} + \frac{\Delta t}{2}\frac{\partial \boldsymbol{W}_i}{\partial t}, \tag{38}$$

and the primitive form of the Euler equations,

$$\frac{\partial \boldsymbol{W}}{\partial t} + \boldsymbol{A}(\boldsymbol{W}) \cdot \nabla \boldsymbol{W} = 0. \tag{39}$$

Equation (39) is used with the slope-limited gradients to replace the time derivative, giving

$$\boldsymbol{W}_i^{n+1/2} = \boldsymbol{W}_i^n + \left[(\boldsymbol{r}_{\mathrm{face}} - \boldsymbol{r}_i) - \frac{\Delta t}{2}\boldsymbol{A}(\boldsymbol{W}_i^n)\right] \cdot (\chi \nabla \boldsymbol{W}). \tag{40}$$

See e.g. appendix A of Hopkins (2015) for the form of $\boldsymbol{A}(\boldsymbol{W})$.

In the Lagrangian mode, the particle positions are then updated via

$$\boldsymbol{r}_i^{n+1} = \boldsymbol{r}_i^n + \frac{\Delta t}{2}\left(\boldsymbol{v}_i^n + \boldsymbol{v}_i^*\right), \tag{41}$$

where $\boldsymbol{v}_i^* = (m_i^n \boldsymbol{v}_i^n + \Delta \boldsymbol{p}_i + m_i^n \boldsymbol{g}_i^n \Delta t)/m_i^{n+1}$ and $\Delta \boldsymbol{p}_i$ is the change in momentum due the fluxes and $\boldsymbol{g}^n$ is the gravitational acceleration (see below).

2.2.6 *Self-gravity*

We adopt the approach of Hopkins (2015) in treating self-gravity, which is itself an adaption of those used by Springel (2010) and Price & Monaghan (2007) applied to the MFV schemes. We have only implemented self-gravity for the MFM scheme and present this implementation here. Similar to SPH, the gravitational softening can be calculated self-consistently following Price & Monaghan (2007) but using the MFV definition for the density. The gravitational force, $m_i \boldsymbol{g}_i$, on a particle is then

$$\begin{aligned} m_i \boldsymbol{g}_i = &- G \sum_{j=1}^{N} m_i\, m_j \frac{\phi'(\boldsymbol{r}_{ij}, h_i) + \phi'(\boldsymbol{r}_{ij}, h_j)}{2} \hat{\boldsymbol{r}}_{ij} \\ &- \frac{G}{2} \sum_{j=1}^{N} \left\{ \frac{\zeta_{\mathrm{i}}'}{\Omega_{\mathrm{i}}'} \nabla W_i(\boldsymbol{r}_{ij}, h_i) + \frac{\zeta_{\mathrm{j}}'}{\Omega_{\mathrm{j}}'} \nabla W_i(\boldsymbol{r}_{ij}, h_j) \right\}, \end{aligned} \tag{42}$$

where the definitions of Ω_i' and ζ_i' for the MFV schemes are

$$\Omega_i' = 1 - \frac{\partial h_i}{\partial n_i} \sum_{j=1}^{N} \frac{\partial W}{\partial h}(\boldsymbol{r}_{ij}, h_i), \tag{43}$$

$$\zeta_i' = m_i \frac{\partial h_i}{\partial n_i} \sum_{j=1}^{N} m_j \frac{\partial \phi}{\partial h}(\boldsymbol{r}_{ij}, h_i). \tag{44}$$

We apply the gravitational force in a similar way to Hopkins (2015), updating the new momentum, $\boldsymbol{p}_i$, and energy, E_i, according to

$$\boldsymbol{p}_i^{n+1} = \boldsymbol{p}_i^n + \Delta\, \boldsymbol{p}_i + \frac{\Delta\, t}{2}\left(m_i^n \boldsymbol{g}_i^n + m_i^{n+1} \boldsymbol{g}_i^{n+1}\right), \tag{45}$$

$$\begin{aligned} E_i^{n+1} = &\ E_i^n + \Delta\, E_i \\ &+ \frac{\Delta\, t}{2}\left(m_i^n \boldsymbol{v}_i^n \cdot \boldsymbol{g}_i^n + m_i^{n+1} \boldsymbol{v}^{n+1} \cdot \boldsymbol{g}_i^{n+1}\right) \end{aligned} \tag{46}$$

For the MFM scheme, since there is no mass flux (i.e. $m^n \equiv m^{n+1}$), the gravitational update [along with the update of particle positions, equation (41)] reduces exactly to a Leapfrog scheme when the pressure forces are negligible. In the original MFV derivation (Hopkins 2015), there are extra terms relating to the mass flux between neighbouring particles, $\mathrm{d}m_{ij}/\mathrm{d}t$, but these also reduce to zero for the MFM scheme.

2.2.7 *Physical viscosity*

Since it is possible to achieve numerical viscosities that are smaller than the physical viscosity in real systems such as accretion discs, we have implemented a physical viscosity in the MFV schemes. The source term in equation (31) due to viscosity can be written as,

$$\boldsymbol{S} = \nabla \cdot (0, \boldsymbol{\Pi}, \boldsymbol{\Pi} \cdot \boldsymbol{v}), \tag{47}$$

$$\boldsymbol{\Pi} = \eta \left\{ \left[\nabla \boldsymbol{v} + (\nabla \boldsymbol{v})^{\mathrm{T}}\right] - \tfrac{2}{3}\mathcal{I}(\nabla \cdot \boldsymbol{v}) \right\} + \zeta \mathcal{I}(\nabla \cdot \boldsymbol{v}), \tag{48}$$

where η and ζ are the shear and bulk viscosity coefficients. Since equation (47) takes the form of the divergence of a flux (with $\boldsymbol{F}_{\mathrm{visc}} = -(0, \boldsymbol{\Pi}, \boldsymbol{\Pi} \cdot \boldsymbol{v})$), we follow Muñoz et al. (2013) in discretizing this term using a finite-volume approach, which simply amounts to including the diffusive flux in equation (33). To compute the viscous flux, one needs to specify a ‘viscous Riemann solver’ along with the edge states to pass to the Riemann solver. Muñoz et al. (2013) suggest using a slope-limited reconstruction of both the primitive variables and the velocity gradients, which are also needed to compute the viscous flux. However, Hopkins (2017) found that reconstructing the velocity gradients makes only a very small difference to the solution (typically less than 1 per cent). Thus, we take a pragmatic approach in using the primitive variables reconstructed at the edges and the particle-centred velocity gradients, which are already available (equations 27 and 40). For the Riemann solver, we simply compute the arithmetic average of the face states and use those to compute the flux.

2.2.8 *Time-stepping*

The MFV scheme uses a similar CFL time-stepping condition as used in SPH (ignoring any artificial viscosity terms), i.e.

$$\Delta\, t_i = C_{\mathrm{CFL}} \frac{h_i}{\mathrm{MAX}_j |v_{\mathrm{sig,ij}}|} \tag{49}$$

where

$$v_{\mathrm{sig,j}} = c_i + c_j - \mathrm{MIN}\left(0, \boldsymbol{v}_{ij} \cdot \hat{\boldsymbol{r}}_{ij}\right). \tag{50}$$

Similarly, when viscosity is included the time-step is limited according to

$$\Delta\, t_i = C_{\mathrm{VISC}} \frac{h_i^2}{2\nu_i} \tag{51}$$

where C_{VISC} is the dimensionless viscosity time-step factor and $\nu_i = (\eta_i + \zeta_i)/\rho_i$ is the total kinematic viscosity for the particles.

Finally, when gravity is included the time-step is limited according to the acceleration condition,

$$\Delta t_i = C_{\rm GRAV} \frac{h_i}{\sqrt{\boldsymbol{g}_i}}. \tag{52}$$

Similarly to SPH, block time-stepping can also be used with MFV. In order to ensure exact conservation, the changes to conserved quantities, $\mathrm{d}\boldsymbol{Q}_{ij}$, are computed on the smallest time-step of the particle pair and built-up over the full time-step, following Springel (2010). Since particles may be interacting with neighbours both on larger and smaller time-steps, the contribution to the fluxes from some particles will be computed once while others may contribute multiple substeps. This means that the conserved quantities only take meaningful values at the beginning and end of the time-steps. Since the primitive quantities may be needed at any point during the particle's time-step to compute the fluxes with a neighbour on a shorter time-step, we also record

$$\frac{\mathrm{d}\boldsymbol{Q}_{\rm i}}{\mathrm{d}t} = -\sum_j \boldsymbol{F}_{ij} \cdot \boldsymbol{A}_{ij}, \tag{53}$$

at the start of the time-step and use it to predict the primitive quantities throughout the time-step. Once the particle reaches the end of its time-step, these are then replaced by the conserved fluxes built-up throughout the time-step.

The block time-stepping scheme can suffer from the same problems with the Meshless scheme as in SPH when particles can interact with neighbours on much longer time-steps. We provide two time-step limiters to solve this problem. First, we have implemented a simple limiter similar to the one used by SPH. When a particle detects that a neighbour is on a time-step lower than the accepted ratio, the particle is 'woken up'. At this time, the fluxes built-up during the block time-stepping scheme are likely to be too large as some neighbours may be on the same time-step level as the particle, or longer. For this reason, we use $\Delta t \frac{\mathrm{d}\boldsymbol{Q}_{\rm i}}{\mathrm{d}t}$ to estimate the new conserved quantities when the particle is woken up. We note that while this breaks the exact conservation, we find that it works well in practice.

Secondly, for cases when exact conservation is required, we have also included the more expensive predictive time-step limiter of Springel (2010), in which the CFL condition is evaluated for distant particles using a tree walk. By limiting the time-step based upon $|\boldsymbol{r}_{ij}|/|\boldsymbol{v}_{\rm sig,ij}|$, this ensures that particles 'wake up' from long time-steps before shocks reach them. In simulations dominated by gravity, pathological configurations can occur where the predictive limiter forces the particles to have much lower time-steps than necessary. In this case, the simple limiter will likely work well since the energy conservation errors are likely dominated by the gravitational forces.

3 *N*-BODY METHODS IN GANDALF

N-body dynamics has been implemented into GANDALF as an independent class separate from the Hydrodynamical algorithms. GANDALF can therefore be run for pure *N*-body problems, albeit not as efficiently compared as dedicated and optimized *N*-body codes such as NBODY6 (Aarseth 2003) or STARLAB/KIRA (Portegies Zwart et al. 2001). In most simulations, the *N*-body module will be used in tandem with the hydrodynamics to represent stars in the guise of sink particles (see Section 3.3). Nevertheless, there are situations where one is interested in the outcome of a simulation if there was no gas present, or as a pure *N*-body simulation after the gas has been removed.

In order to make the *N*-body algorithms compatible with the Hydrodynamical algorithms and to prevent unphysical two- or three-body ejections and/or large energy errors, we give each *N*-body particle a (constant) smoothing length. The acceleration of an *N*-body particle due to all other *N*-body particles is simply:

$$\boldsymbol{a}_s = -G \sum_{t=1}^{N} m_t\, \phi'(\boldsymbol{r}_{st}, \overline{h}_s)\, \hat{\boldsymbol{r}}_{st}, \tag{54}$$

where $\overline{h}_{st} \equiv \frac{1}{2}(h_s + h_t)$.

3.1 Integration schemes

GANDALF can use several integration schemes for simulating *N*-body dynamics independent of the choice of hydrodynamics scheme. For simple problems or when using accreting sink particles (see Section 3.3), we can use the Leapfrog KDK and DKD schemes outlined in Section 2.1.3 using the same sets of equations (15)–(20) together with the acceleration time-step condition (equation 23). For pure *N*-body simulations, or hybrid simulations that require higher accuracy, we can use other higher order schemes.

3.1.1 *Fourth-order Hermite scheme*

In the fourth-order Hermite scheme (Makino & Aarseth 1992), we explicitly calculate the first time derivative of the acceleration (often called the *jerk*), $\dot{\boldsymbol{a}}$, in order to achieve higher integration accuracy. At the beginning of the step, we calculate both the acceleration and the jerk, where the jerk is given by:

$$\begin{aligned}\dot{\boldsymbol{a}}_s^n =& -G \sum_{t=1}^{N} \frac{m_t\, \phi'(\boldsymbol{r}_{st}, \overline{h}_{st})}{|\boldsymbol{r}_{st}|} \boldsymbol{v}_{st} \\ &+ 3\,G \sum_{t=1}^{N} \frac{m_t\, (\boldsymbol{r}_{st} \cdot \boldsymbol{v}_{st})\, \phi'(\boldsymbol{r}_{st}, \overline{h}_{st})}{|\boldsymbol{r}_{st}|^3} \boldsymbol{r}_{st} \\ &- 4\pi G \sum_{t=1}^{N} \frac{m_t\, (\boldsymbol{r}_{st} \cdot \boldsymbol{v}_{st})\, W(\boldsymbol{r}_{st}, \overline{h}_{st})}{|\boldsymbol{r}_{st}|^2} \boldsymbol{r}_{st}.\end{aligned} \tag{55}$$

Once calculated for all stars, we predict the star positions and velocities to the end of the step with a Taylor expansion,

$$\boldsymbol{r}_s^{n+1} = \boldsymbol{r}_s^n + \boldsymbol{v}_s^n\, \Delta t + \frac{1}{2} \boldsymbol{a}_s^n\, \Delta t^2 + \frac{1}{6} \dot{\boldsymbol{a}}_s^n\, \Delta t^3, \tag{56}$$

$$\boldsymbol{v}_s^{n+1} = \boldsymbol{v}_s^n + \boldsymbol{a}_s^n\, \Delta t + \frac{1}{2} \dot{\boldsymbol{a}}_s^n\, \Delta t^2. \tag{57}$$

We then calculate the acceleration jerk again using equation (55) using the predicted positions and velocities at the end of the step, i.e. $\boldsymbol{a}_s^{n+1}$ and $\dot{\boldsymbol{a}}_s^{n+1}$. This allows us to construct the higher order time derivatives for the step,

$$\ddot{\boldsymbol{a}}_s^n = \frac{2\left(-3(\boldsymbol{a}_s^n - \boldsymbol{a}_s^{n+1}) - (2\dot{\boldsymbol{a}}_s^n + \dot{\boldsymbol{a}}_s^{n+1})\Delta t\right)}{\Delta t^2}, \tag{58}$$

$$\dddot{\boldsymbol{a}}_s^n = \frac{6\left(2(\boldsymbol{a}_s^n - \boldsymbol{a}_s^{n+1}) + (\dot{\boldsymbol{a}}_s^n + \dot{\boldsymbol{a}}_s^{n+1})\Delta t\right)}{\Delta t^3}. \tag{59}$$

where $\ddot{\boldsymbol{a}}^n$ and $\dddot{\boldsymbol{a}}^n$ are the second and third time derivatives of the acceleration, respectively. Finally, we apply these higher order derivatives as a correction step to calculate the position and velocity to high order,

$$\boldsymbol{r}_s^{n+1} = \boldsymbol{r}_s^{n+1} + \frac{1}{24} \ddot{\boldsymbol{a}}_s^n\, \Delta t^4 + \frac{1}{120}\, \dddot{\boldsymbol{a}}_s^n\, \Delta t^5, \tag{60}$$

$$\boldsymbol{v}_s^{n+1} = \boldsymbol{v}_s^{n+1} + \frac{1}{6}\ddot{\boldsymbol{a}}_s^n\,\Delta t^3 + \frac{1}{24}\,\dddot{\boldsymbol{a}}_s^n\,\Delta t^4. \qquad (61)$$

To compute the time-step for each star, we use the Aarseth criterion as used in the NBODY codes (e.g. Aarseth 2003),

$$\Delta t_s = \gamma_s\sqrt{\frac{|\boldsymbol{a}_s||\ddot{\boldsymbol{a}}_s| + |\dot{\boldsymbol{a}}_s|^2}{|\dot{\boldsymbol{a}}_s||\,\dddot{\boldsymbol{a}}_s\,| + |\ddot{\boldsymbol{a}}_s|^2}}. \qquad (62)$$

3.1.2 Fourth-order time-symmetric integration scheme

For simulations which require higher stability or more accuracy, particularly with long-term orbital integration (e.g. binary or multiple systems), we can use the Hut, Makino & McMillan (1995) time-symmetric fourth-order Hermite scheme. In this variant, we compute the acceleration and jerk at the beginning of the time-step similar to the standard Hermite scheme. We then predict the position and velocities at the end of the time-step. The corrected position and jerk are recomputed using

$$\boldsymbol{r}_s^{n+1} = \boldsymbol{r}_s^n + \frac{1}{2}\left(\boldsymbol{v}_s^{n+1} + \boldsymbol{v}_s^n\right)\Delta t - \frac{1}{12}\left(\boldsymbol{a}_s^{n+1} - \boldsymbol{a}_s^n\right)\Delta t^2, \qquad (63)$$

$$\boldsymbol{v}_s^{n+1} = \boldsymbol{v}_s^n + \frac{1}{2}\left(\boldsymbol{a}_s^{n+1} + \boldsymbol{a}_s^n\right)\Delta t - \frac{1}{12}\left(\dot{\boldsymbol{a}}_s^{n+1} - \dot{\boldsymbol{a}}_s^n\right)\Delta t^2. \qquad (64)$$

A more accurate solution is obtained by iterating the evaluate-correction step until the particle's position and velocity are converged. Such schemes are often called P(EC)^n where n is the number of correction iterations. In practice, even using $n = 2$ gives improved results. We note that despite its name, a truly time-symmetric integration is only possible for constant time-steps whereas most N-body codes use adaptive time-steps.

3.2 Hybrid SPH and N-body dynamics

GANDALF contains an implementation of the Hubber et al. (2013a) hybrid SPH/N-body algorithm. This is designed to simulate small to intermediate size clusters which also have a live gaseous background. One noticeable difference between this and the original Hubber et al. (2013a) implementation is the mode of symmetrizing the particle–particle interactions. In Hubber et al. (2013a), the gravitational interactions between all particle pairs (gas–gas, gas–star and star–star) were smoothed using the average smoothing length, i.e. $W(\boldsymbol{r}, \frac{1}{2}(h_i + h_j))$. In GANDALF, this has been modified so gas–gas interactions use the standard (Price & Monaghan 2007) form in grad-h SPH with the average of the kernels (equation 12), whereas only the gas–star and star–star interactions use the average smoothing length approach. Smoothing the gas–star interactions with the average smoothing length is designed to prevent the situation where the smoothing lengths of gas and star particles are hugely different leading to the unphysical two-body scattering which softening is designed to prevent. The full equation of motion for gas particles becomes

$$\boldsymbol{a}_i = -\sum_{j=1}^{N_g} m_j\left[\frac{P_i}{\rho_i^2\Omega_i}\frac{\partial W_{ij}}{\partial\boldsymbol{r}_i}(h_i) + \frac{P_j}{\rho_j^2\Omega_j}\frac{\partial W_{ij}}{\partial\boldsymbol{r}_i}(h_j)\right] - G\sum_{j=1}^{N_g} m_j\,\frac{\phi'(\boldsymbol{r}_{ij}, h_i) + \phi'(\boldsymbol{r}_{ij}, h_j)}{2}\,\hat{\boldsymbol{r}}_{ij}$$
$$-\frac{G}{2}\sum_{j=1}^{N_g} m_j\left[\frac{\zeta_i' + \bar{\chi}_i}{\Omega_i}\frac{\partial W_{ij}}{\partial\boldsymbol{r}_i}(h_i) + \frac{\zeta_j' + \bar{\chi}_j}{\Omega_j}\frac{\partial W_{ij}}{\partial\boldsymbol{r}_i}(h_j)\right] - G\sum_{s=1}^{N_s} m_s\,\phi_{is}'(\bar{h}_{is})\,\hat{\boldsymbol{r}}_{is}, \qquad (65)$$

where

$$\bar{\chi}_i = \frac{\partial h_i}{\partial\rho_i}\sum_{j=1}^{N} m_i\,\frac{\partial\phi_{ij}}{\partial\bar{h}_{ij}}(\bar{h}_{ij}). \qquad (66)$$

These equations are then numerically integrated using the second-order Leapfrog KDK scheme (Section 2.1.3). The total equation of motion for stars becomes

$$\boldsymbol{a}_s = -G\sum_{t=1}^{N_s} m_t\,\phi_{st}'(\bar{h}_{st})\,\hat{\boldsymbol{r}}_{st} - G\sum_{i=1}^{N_g} m_i\,\phi_{si}'(\bar{h}_{si})\,\hat{\boldsymbol{r}}_{si}. \qquad (67)$$

This modification removes the need for an additional loop over SPH neighbours to calculate the values for ζ_i using averaged smoothing lengths.

We note that this conservative scheme is not formally implemented to work with the MFV/MFM schemes although the basic fourth-order Hermite scheme can still be utilized together in tandem with the MFV/MFM hydrodynamics integration scheme.

3.3 Sink particles

Sink particles (Bate, Bonnell & Price 1995) are used in self-gravitating hydrodynamics codes to relieve the problem of high-density condensations (e.g. protostars) leading to very short time-steps and prohibitively long CPU run times. In their most basic form, sink particles replace the forming protostar (or other accreting object) with a single particle with an accretion radius R_s that accretes any gas particles that enter the accretion radius by adding their mass and momentum to the sink. Hubber, Walch & Whitworth (2013b) introduced an improved sink particle algorithm in SPH which computed the accretion rate based on an internal subgrid model leading to better convergence of results. GANDALF implements both the simpler 'vacuum-cleaner' sink particles and the improved sinks of (Hubber et al. 2013b), both for SPH and for the MFV/MFM schemes.

3.3.1 Sink formation criteria

A new sink particle is created from an existing gas particle that satisfies a number of criteria. These criteria are designed to ensure that sinks are only formed in genuinely self-gravitating entities, such as in collapsing prestellar cores and protostars. When a sink particle is formed, it is given an accretion radius that is some multiple of the original particle's smoothing length,

$$R_s = X_{\text{SINK}}\,h_i \qquad (68)$$

where X_{SINK} is a user-defined factor of order unity and h_i is the smoothing length of the original gas particle. For consistency, X_{SINK} is normally chosen so that the sink accretion volume is the same as the smoothing kernel volume (e.g. for the M4-kernel, $X_{\text{SINK}} = 2$).

The formation criteria are:

(i) The density of a gas particle should exceed the user-defined sink creation density, ρ_{SINK}, i.e.

$$\rho_i > \rho_{\text{SINK}}. \qquad (69)$$

(ii) A new sink particle formed from a hydrodynamical particle does not overlap any existing sinks upon creation, i.e.

$$|\boldsymbol{r}_i - \boldsymbol{r}_s| > X_{\rm SINK}\, h_i + R_s . \tag{70}$$

(iii) The gravitational potential of a hydrodynamical particle is the minimum (as in most negative) of all of its hydrodynamical neighbours, i.e.

$$\phi_i < {\rm MIN}\left\{\phi_j\right\} . \tag{71}$$

(iv) The density is sufficiently large so local condensations do not lie within the Hill sphere (or equivalently the Roche limit) of all existing sinks, i.e.

$$\rho_i > \frac{3\,X_{\rm HILL}\,\Delta\boldsymbol{r}_{is}\cdot\Delta\boldsymbol{a}_{is}}{4\pi G|\Delta\boldsymbol{r}_{is}|^2} . \tag{72}$$

(v) A condensation can undergo free fall collapse before approaching any existing sinks, i.e.

$$t_{\rm FF} < \frac{|\Delta\boldsymbol{r}_{is}|^2}{\Delta\boldsymbol{v}_{is}\cdot\Delta\boldsymbol{r}_{is}} . \tag{73}$$

3.3.2 *Sink accretion*

In the simplest case, accretion of gas particles on to sink particles can be achieved simply by adding the mass, momentum and energy of every gas particle entering the sink radius. Additional criteria may be employed, such as checking if the gas particles are gravitationally bound to the sink particle. Hubber et al. (2013b) introduced a simple two-mode subgrid model of accretion which we have implemented into GANDALF. The first mode treats the case of purely spherical collapse, i.e. inward radial velocities. The (smoothed average) radial infall time-scale in terms of the particle properties is

$$\left\langle t_{\rm RAD}\right\rangle_s = \frac{\sum_j\left\{m_j\right\}\,\mathcal{W}}{4\pi\sum\limits_j\left\{|\Delta\boldsymbol{r}_{js}|\Delta\boldsymbol{r}_{js}\cdot\Delta\boldsymbol{v}_{js}m_j W(|\Delta\boldsymbol{r}_{js}|,H_s)\right\}} , \tag{74}$$

where

$$\mathcal{W} = \sum_j\left\{m_j W(|\Delta\boldsymbol{r}_{js}|,H_s)/\rho_j\right\} . \tag{75}$$

The second mode treats the case of purely rotational collapse, i.e. where all velocities are tangential with speeds for circular motion. For low-mass discs in approximate Keplerian rotation, the accretion time-scale at a radius R is given by the Shakura–Sunyaev prescription, $t_{\rm SS} \sim \alpha_{\rm SS}^{-1}(GM_\star R)^{1/2}a^{-2}$, where $\alpha_{\rm SS}$ is the Shakura–Sunyaev viscosity and a is the local sound speed. A kernel-weighted average of this time-scale over all particles in the sink gives

$$\left\langle t_{\rm DISC}\right\rangle = \frac{(GM_s)^{1/2}}{\alpha_{\rm SS}\mathcal{W}}\sum_j\left\{\frac{|\Delta\boldsymbol{r}_{js}|^{1/2}m_j W(|\Delta\boldsymbol{r}_{js}|,H_s)}{\rho_j a_j^2}\right\} . \tag{76}$$

Since accreting particles will in general fall between these two limits, we use a simple interpolation using a weighted geometric mean to give an overall accretion time-scale of

$$t_{\rm ACC} = \left\langle t_{\rm RAD}\right\rangle_s^{(1-f)}\left\langle t_{\rm DISC}\right\rangle_s^{f} , \tag{77}$$

where

$$f = {\rm MIN}\left\{2E_{\rm ROT}/|E_{\rm GRAV}|,\,1\right\} \tag{78}$$

is a simple measure of the centrifugal support using the rotational and gravitational energies of particles inside the sink, where $f = 1$ is expected for circular rotation.

The total mass of gas particles to be accreted in the current time-step is then

$$\delta M_{\rm ACC} = M_{\rm INT}\left[1 - \exp\left(-\frac{\delta t_s}{t_{\rm ACC}}\right)\right] . \tag{79}$$

4 MISC

4.1 Dust

The dynamics of dust-gas mixtures have been implemented in GANDALF using the 'two-fluid' formalism. An additional set of dust particles can be included, which are coupled to the gas motions via drag forces. The main scheme closely follows Lorén-Aguilar & Bate (2015), who provide expressions for a semi-implicit update for the drag force that avoids the need for small time-steps when the drag forces are very strong. We refer the reader to Lorén-Aguilar & Bate (2015) for details and only briefly outline the scheme. The equations of motion for gas and dust particles are

$$\frac{{\rm d}\boldsymbol{v}_{\rm g}}{{\rm d}t} = -\frac{\rho_{\rm d}}{\rho_{\rm g}}\frac{(\boldsymbol{v}_{\rm g}-\boldsymbol{v}_{\rm d})}{t_{\rm s}} + \boldsymbol{a}_g - \frac{\nabla P}{\rho_{\rm g}} , \tag{80}$$

$$\frac{{\rm d}\boldsymbol{v}_{\rm d}}{{\rm d}t} = -\frac{(\boldsymbol{v}_{\rm d}-\boldsymbol{v}_{\rm g})}{t_{\rm s}} + \boldsymbol{a}_{\rm d} , \tag{81}$$

where $t_{\rm s}$ is the one-particle stopping time, and the backreaction of the dust on the gas has been included to conserve the total momentum.

To solve these equations over a single time-step Δt, the hydrodynamic and gravitational forces are first calculated as normal. The semi-implicit update is computed by making the ansatz that these forces, along with the densities and $t_{\rm s}$, are constant throughout the time-step. The above equations can then be solved to give the new velocities,

$$\boldsymbol{v}_{\rm d}(t+\Delta t) = \tilde{\boldsymbol{v}}_{\rm d}(t+\Delta t) - \frac{\rho_{\rm g}}{\rho_{\rm d}+\rho_{\rm g}}\boldsymbol{S}_{\rm dg} \tag{82}$$

$$\boldsymbol{v}_{\rm g}(t+\Delta t) = \tilde{\boldsymbol{v}}_{\rm g}(t+\Delta t) + \frac{\rho_{\rm d}}{\rho_{\rm d}+\rho_{\rm g}}\boldsymbol{v}_{\rm dg} \tag{83}$$

where $\tilde{\boldsymbol{v}}_{\rm d,g}(t+\Delta t) = \boldsymbol{v}_{\rm d,g}(t) + \boldsymbol{a}_{\rm d,g}(t)\Delta t$. Writing $\Delta\tilde{\boldsymbol{v}} = \tilde{\boldsymbol{v}}_{\rm d} - \tilde{\boldsymbol{v}}_{\rm g}$ and $\Delta\boldsymbol{a} = \boldsymbol{a}_{\rm d} - \boldsymbol{a}_{\rm g} + \nabla P/\rho_{\rm g}$, then $\boldsymbol{S}_{\rm dg}$ is given by

$$\begin{aligned}\boldsymbol{S}_{\rm dg} &= \left(1 - e^{\Delta t/t_{\rm s}}\right)\Delta\tilde{\boldsymbol{v}}(t+\Delta t)\\ &\quad - \left[(\Delta t + t_{\rm s})\left(1 - e^{\Delta t/t_{\rm s}}\right) - \Delta t\right]\Delta\boldsymbol{a}(t).\end{aligned} \tag{84}$$

To convert this update into SPH form, we project the velocity along the line of sight and sum over the neighbours using a double-hump kernel (which we denote by $\tilde{W}$), in order to ensure angular momentum conservation while computing the drag force accurately (Laibe & Price 2012; Lorén-Aguilar & Bate 2015). The resulting equations are:

$$\begin{aligned}\boldsymbol{v}_{\rm d}^i(t+\Delta t,\boldsymbol{r}_i) &= \tilde{\boldsymbol{v}}_{\rm d}^i(t+\Delta t,\boldsymbol{r}_{\rm i})\\ &\quad - D\sum_a^{\rm Gas}\frac{m_{\rm a}}{\rho_{\rm i}+\rho_{\rm a}}(\boldsymbol{S}_{\rm ia}\cdot\hat{\boldsymbol{r}}_{\rm ia})\hat{\boldsymbol{r}}_{\rm ia}\tilde{W}(\boldsymbol{r}_{\rm ia},h_{\rm a})\end{aligned} \tag{85}$$

$$\begin{aligned}\boldsymbol{v}_{\rm g}^a(t+\Delta t,\boldsymbol{r}_{\rm a}) &= \tilde{\boldsymbol{v}}_{\rm g}^a(t+\Delta t,\boldsymbol{r}_{\rm a})\\ &\quad + D\sum_i^{\rm Dust}\frac{m_{\rm i}}{\rho_{\rm i}+\rho_{\rm a}}(\boldsymbol{S}_{\rm ia}\cdot\hat{\boldsymbol{r}}_{\rm ia})\hat{\boldsymbol{r}}_{\rm ia}\tilde{W}(\boldsymbol{r}_{\rm ia},h_{\rm a}).\end{aligned} \tag{86}$$

The drag force dissipates kinetic energy, which may go into heating the gas, dust or be lost from the system depending on the details of the problem. When using a barotropic EOS, which is common in astrophysical applications with dust-gas mixtures (e.g. discs, star formation or molecular clouds), we do not explicitly track the kinetic energy dissipated. However, when using an adiabatic EOS, we assume that the dissipated kinetic energy heats the gas directly.

To ensure exact conservation, we compute the change in kinetic energy due to drag forces directly from the above equations,

$$\Delta \mathrm{KE}_\mathrm{i} = m_\mathrm{i} |\boldsymbol{v}_\mathrm{i}(t+\Delta t) - \tilde{\boldsymbol{v}}_\mathrm{i}(t+\Delta t)|^2. \tag{87}$$

The change in kinetic energy of a gas particle is added directly to the change in its internal energy. For dust particles, we spread its change in kinetic energy amongst its neighbouring gas particles, using the same kernel as for the drag force calculation. The total change in a gas particle's internal energy is thus

$$m_\mathrm{a} \Delta u_\mathrm{a} = \Delta \mathrm{KE}_\mathrm{a} + \frac{m_\mathrm{a}}{\rho_\mathrm{a}} \sum_i^{\mathrm{Dust}} \frac{1}{N_\mathrm{i}} \Delta \mathrm{KE}_\mathrm{i} \tilde{W}(\boldsymbol{r}_\mathrm{ia}, h_\mathrm{a}), \tag{88}$$

where N_i is a normalization factor,

$$N_\mathrm{i} = \sum_a^{\mathrm{Gas}} \frac{m_\mathrm{a}}{\rho_\mathrm{a}} \tilde{W}(\boldsymbol{r}_\mathrm{ia}, h_\mathrm{a}). \tag{89}$$

Summing equation (88) over all gas particles gives $\sum_a^{\mathrm{Gas}} m_\mathrm{a} \Delta u_\mathrm{a} = \sum_a^{\mathrm{Gas}} \Delta \mathrm{KE}_\mathrm{a} + \sum_i^{\mathrm{Dust}} \Delta \mathrm{KE}_\mathrm{i}$, i.e. manifest energy conservation. Finally, we note that this energy update can be implemented simply. We compute N_i during the drag force calculation for the dust. Once the drag force for the single dust particle has been computed, the change in kinetic energy is then 'given back' to its neighbours. In practice, we use equations (85), (86) and (88) to define time-averaged rates of change in the physical quantities which are included in the standard SPH time integration scheme.

The dust scheme has been described above in terms of SPH, but can naturally be extended to the MFM integration algorithm. To do this, we proceed exactly as in SPH, except that change in velocity is multiplied by the particle mass and added to the change in momentum, $\Delta \boldsymbol{p}$. Also, since the MFM method integrates the total rather than the internal energy, only the change in kinetic energy from the dust particles needs to be included. This allows conservation of energy and momentum to machine precision. However, there is one subtlety, in that MFV and MFM use a single hydrodynamical update per time-step, but the gravitational acceleration is treated using the KDK Leapfrog, i.e. two kicks per time-step. Rather than use two drag kicks per time-step (one with the initial and one with the final gravitational acceleration), we instead take the pragmatic approach of using the time average, $m_1 \bar{\boldsymbol{a}} = (m_0 \boldsymbol{a}_0 + m_1 \boldsymbol{a}_1)/2$, where $m_{0,\,1}$ and $\boldsymbol{a}_{0,1}$ are the accelerations and masses computed at the beginning and end of the step. This works well in practice because the drag forces only depend on the difference between the dust and gas accelerations (see Lorén-Aguilar & Bate 2015), which for gravitational forces is typically close to zero (except perhaps in very poorly resolved regions close to sink particles). Finally, in the meshless, the $\nabla P/\rho_\mathrm{g}$ term is taken from the change in momentum computed using the Riemann solver (equation 53).

In addition to full two-fluid scheme above, GANDALF also includes a test-particle scheme. The main advantage of this scheme is that, unlike the full two-fluid scheme, it can naturally handle block time-steps, whereas the full two-fluid scheme becomes inaccurate if not used with global time-steps. While it would be straightforward to create a test-particle scheme by setting $\rho_\mathrm{i} = 0$ in equation (85) and neglecting equations (86) and (88), in cases where the particle distribution is non-uniform the force accuracy can be improved by using a normalized interpolations scheme, as in Booth, Sijacki & Clarke (2015). In this scheme, equation (85) is replaced by equation (82) and $\boldsymbol{S}_\mathrm{dg}$ is computed by interpolating the gas properties to the location of the dust particle and using them directly in equation (84). In formula, any given quantity A_i, defined on the gas particles, it is interpolated using

$$A_\mathrm{d} = \sum_i^{\mathrm{Gas}} \frac{A_\mathrm{i}}{\hat{n}_\mathrm{d}} W(\boldsymbol{r}_\mathrm{id}, \hat{h}_\mathrm{d}) \tag{90}$$

where

$$\hat{n}_\mathrm{d} = \sum_i^{\mathrm{Gas}} W(\boldsymbol{r}_\mathrm{id}, \hat{h}_\mathrm{d}) \tag{91}$$

and $\hat{h}_\mathrm{d} = \eta_{\mathrm{SPH}}(1/\hat{n}_\mathrm{d})^{1/D}$, which is evaluated using the standard Newton–Raphson iteration with the same tolerance as the mass density.

As with pure hydrodynamics problems with the MFM method, we find that using the quintic kernel can significantly improve the accuracy of the results due to more accurate density estimates and smaller interpolation errors (see e.g. Price 2012; Laibe & Price 2012; Price & Laibe 2015). We thus recommend use of the quintic kernel in problems involving dust, and use it in the tests presented here.

4.2 Tree

In GANDALF, we have implemented a KD tree to efficiently determine neighbour list for computing all local quantities (e.g. smoothing lengths) and for computing gravitational forces. Our implementation is loosely based on the one described in Gafton & Rosswog (2011); we refer the interested reader to that paper and highlight the differences from our implementation in the following text. The tree is built in a top-down approach; starting from a root cell that contains all the particles, each cell is divided in two subcells along a chosen direction until one is left only with *leaf* cells, i.e. cells containing a number of particles equal or smaller than a set maximum, N_{LEAF}. The slice direction is always chosen to be the one along the cell's most elongated axis, in order to avoid having cells with large aspect ratios. In contrast to Gafton & Rosswog (2011), we follow a more traditional KD-tree construction and split cells using the median value of the particle's positions (what they describe as MPS method). This guarantees that the tree is balanced; i.e. if there are 2^l particles, the tree will contain l levels (for $N_{\mathrm{LEAF}} = 1$), which simplifies the memory management.

Once the tree has been constructed, a number of properties can be computed for each cell and propagated upwards to the parent cells, such as the position of the centre-of-mass, the gravitational moments (needed for computing the gravitational acceleration) and the extent of the smallest box containing all the smoothing spheres of the particles. This box will be used during the tree walk to decide if a given cell potentially contains hydrodynamical neighbours of a given particle.

When including self-gravity, the tree is also used to reduce the expensive $O(N^2)$ calculation to $O(N \log N)$ by grouping the contribution from distant particles together. The tree is walked from the root cell and each cell is tested to see whether the contribution from the cell is sufficiently accurate; if not the cell is opened and its children are tested. This can be done using the classic geometric opening criterion (e.g. Barnes & Hut 1986),

$$|\boldsymbol{r}_i - \boldsymbol{r}_c|^2 \geq \frac{l_c^2}{\theta_{\mathrm{MAX}}^2} \tag{92}$$

where $\boldsymbol{r}_c$ is the cell position, l_c is the cell 'size' (i.e. the centre-to-corner distance of the cell) and $\theta_{\rm MAX}$ is the maximum allowed opening angle of the cell (typically ∼0.3). The cell approximation can be used if the inequality is satisfied. Otherwise, we must open the cell and test each of its children cells. Optionally, a second criterion can be included whereby cells are opened if the contribution to the force from their quadrupole moment is too large. Either the Springel (2005) Multipole-acceptance criterion (MAC),

$$|\boldsymbol{r}_i - \boldsymbol{r}_c|^2 \geq \left(\frac{G\, M_c l_c^2}{\alpha_c} \right)^{1/2} |\boldsymbol{a}_{\rm GRAV}|^{-1/2} \tag{93}$$

where M_c is the cell mass, $\boldsymbol{a}_{\rm GRAV}$ is the gravitational acceleration from the previous step and α_c is the maximum fractional contribution to the total acceleration from the cell quadrupole term (typically $\alpha_c \sim 10^{-4}$). or the eigenvalue-based criterion of (see Hubber et al. 2011, for details) can be used in GANDALF.

Even with the optimizations provided by using a tree, walking the tree to find neighbours is still an expensive operation that can dominate the total CPU cost of a simulation. We optimize the walk by retrieving the list of neighbours for each leaf cell rather than for each individual particle (Wadsley, Stadel & Quinn 2004). GANDALF caches the list of particles and cells found during the tree walk. When self-gravity is included, the gravitational force contribution from the particles is computed directly for all of the particles in the leaf cell. For the contribution from the distant cells, the gravitational force calculation can be computed in one of two ways: either directly for each particle in the leaf cell or using a Taylor-series expansion about the centre of the leaf cell similar to Gafton & Rosswog (2011). Both the monopole and quadrupole moments can be included in the force contribution for the cells; when using the Taylor-series method, we expand the monopole term to second order (as in Gafton & Rosswog 2011), but only include the first-order term in the expansion of the quadrupole. In practice, because the actual force computation takes only a small fraction of the time spent walking the tree, we find that computing the force directly for each particle and including the quadrupole moments is typically the most efficient (see Section 6.5). The serial performance and parallel scaling of the tree is found to be sensitive to the choice of value for $N_{\rm LEAF}$. This is discussed in detail in Section 7.

Finally, rather than rebuilding the complete tree at every step, we can update the properties of the tree cells bottom-up. This is particularly relevant for time-steps where only a small fraction of all particles are active, in which case the cost of rebuilding the tree can become comparable to the cost of the hydro step itself. In practice, we rebuild the tree after a fixed number of time-steps (specified by the user). In contrast to Gafton & Rosswog (2011), we do not perform an integrity check on the tree since the tree-walking algorithm will always retrieve the correct neighbours even if the particles have moved outside of the initial cell (provided that the extent of the cells is updated accordingly).

4.3 Boundary conditions

Both the SPH and MFV schemes can naturally handle isolated systems with no need for explicit boundary conditions. However, boundaries need to be explicitly handled in cases such as the join between computational domains, when modelling systems with reflection symmetry, or in periodic domains. Periodic and reflecting boundaries in GANDALF are handled using 'ghost particles', which are copies of real particles that fall near the edges of the simulation domain. Depending on the type of boundary, these particles may be direct copies on a different processor (MPI domain boundaries), copies of particles that have been translated to a new position (periodic ghosts) or reflected across a boundary.

The ghost particles are constructed in one of two different ways; they can be computed in advance of time or generated on-the-fly as needed. In GANDALF, both approaches are used. For the density and dust force calculations, both the physical and MPI ghosts are computed ahead of time. This is done because these loops may require the smoothing lengths to be iterated to achieve convergence, resulting in the need to export the particles every time the smoothing lengths are changed. As long as enough ghosts are constructed initially there is no need to iterate the density. However, in the hydrodynamical and gravitational force calculations, which do not require iteration, ghosts at physical boundaries are constructed on-the-fly. This is done to simplify the gravitational force calculation in periodic simulations. Similar to GADGET-2, the contribution to the forces from interactions with particles on external processors is handled by exporting the particles to the other processor before computing the forces and sending back the result.

When employing periodic boundaries with self-gravity, we use the Ewald method (e.g. Hernquist, Bouchet & Suto 1991) for computing periodic gravity forces. This method assumes that the simulation box is infinitely replicated in all Cartesian directions. A table of periodic gravitational correction terms is generated and used when computing forces between all gravitating particles or tree cells. Wünsch et al. (2017) have recently adapted the original Ewald method to allow periodic gravitational forces for either 1D or 2D periodicity, which has been implemented in GANDALF. This could be used for example to model an infinitely wide sheet or an infinitely long filament. Although GANDALF is a multidimensional code, the periodic gravity can only be employed in 3D, whether using 1D, 2D or 3D periodicity.

4.4 Generating initial conditions

Constructing initial conditions for arbitrary density fields is in general more complicated for particle methods than grid methods, which can simply set the density field for each grid cell directly. The simplest approach is to use Monte Carlo rejection sampling of the density field, which gives approximately the correct density field but with a considerable amount of noise. In Fig. 1(a) (1st column), we use Monte Carlo rejection sampling to select particles representing a simple sinusoidal density field, $\rho(x) = 1.0 + \frac{1}{2}\sin\{2\pi x\}$ in 2D. As can be seen, the particle distribution is extremely non-regular (bottom row) leading to considerable scatter in the density field (top row), even when smoothed using equation (5).

Gaburov & Nitadori (2011) mitigate this problem somewhat by regularizing the particle distribution at start-up (i.e. after initial conditions generation) to reduce this noise by making the local particle distribution more glass-like (Fig. 1b). Although successful, too many iterations leads to a completely uniform distribution of particles, effectively washing out the original density structure. After 100 iterations, while generating a more regular distribution with less noise, the amplitude of the sine-wave has been reduced by approximately a half (Fig. 1b, top row) and will continue to 'decay' with successively more iterations. Alternatively, Whitworth et al. (1995) used a similar method to iterate particle positions towards a given density field (Fig. 1c). While giving a good fit to the density field and an improved particle distribution over the original Monte Carlo sampling, this leads to a imperfect (i.e. not glass-like) distribution of particles with noticeable particle–particle 'clumping' at various points in the distribution.

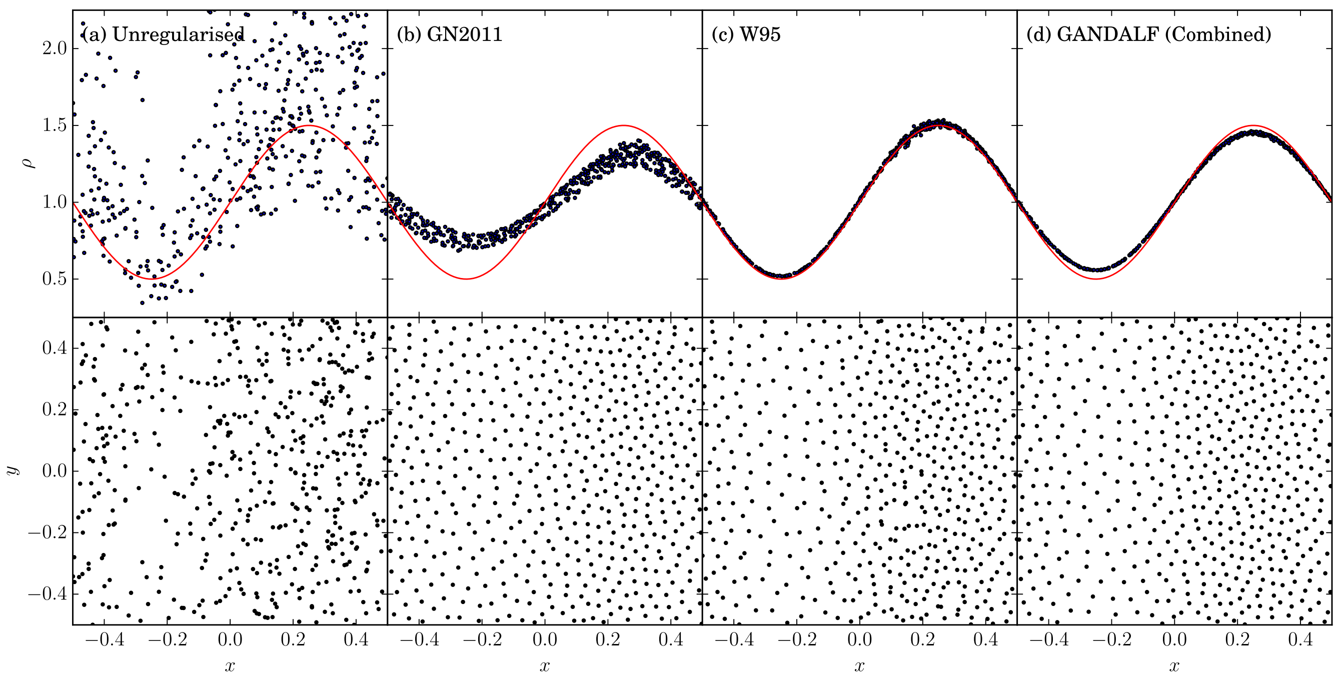


Figure 1. The density profile (top row) and particle distribution (bottom row) resulting from generating a simple sine-wave density field using Monte Carlo rejection sampling (first column), the Gaburov & Nitadori (2011) regularization method (second column), the Whitworth et al. (1995) density method (third column) and the combined approach used in GANDALF (fourth column).

GANDALF contains a general initial conditions (IC) algorithm that effectively combines the two approaches of Whitworth et al. (1995) and Gaburov & Nitadori (2011) by simultaneously iterating towards a given density profile while moving the particles to a more regular distribution. The full procedure for generating ICs is:

(i) Calculate the total mass contained in the computational domain, $M_{\rm TOT}$, either by analytically or by numerical integration of the density field. All particles are assigned an equal mass $m = M_{\rm TOT}/N$.
(ii) Use Monte Carlo rejection sampling to assign the initial positions of all particles. Although our algorithm works in principle from any initial distribution, it converges much faster if the particles are already close to their final positions.
(iii) Iterate the particle positions using

$$\boldsymbol{r}'_i = \boldsymbol{r}_i + h_i \sum_{j=1}^{N} \left\{ \alpha_{\rm IC} - \beta_{\rm IC} \left(\frac{\rho(\boldsymbol{r}_j) - \rho_j}{\rho(\boldsymbol{r}_j)} \right) \right\} W(\boldsymbol{r}_{ij}, h_i)\hat{\boldsymbol{r}}_{ij} \,. \tag{94}$$

where $\rho(\boldsymbol{r}_j)$ is the analytical (or tabulated) density at the position of particle j, ρ_j is the smoothed density of particle j, $\alpha_{\rm IC}$ is the weighting of the particle regularization term and $\beta_{\rm IC}$ is the weighting of the density field term. In practice, we find values of $\alpha_{\rm IC} = 0.1$ and $\beta_{\rm IC} = 0.9$ give a good balance between giving a regular particle distribution and an accurate density field. We note that higher values of $\alpha_{\rm IC}$ gives a more regular distribution but can under-resolve density peaks.
(iv) Once the positions have converged, assign the remaining particle and hydrodynamical properties (e.g. velocity and specific internal energy).

One issue not addressed by this algorithm is creating equilibrium ICs, with the exception of trivial uniform density configurations (such as a uniform glass). Hydrodynamical forces (due to second-order smoothing errors) are not truly represented by any given density gradient, even if the density field is accurate. Gravitational forces also have a similar (although smaller in magnitude) smoothing error. Therefore, exact hydrostatic equilibrium cannot be obtained with this method.

5 IMPLEMENTATION DETAILS

5.1 General design and structure of the code

We have followed many object-oriented principles when designing GANDALF. In this section, we show some examples to demonstrate why an object-oriented approach is useful for a Astrophysics hydrodynamical code; we refer the interested reader to the userguide and the code base for more details on the class structure of GANDALF. The use of object-oriented design has allowed GANDALF to follow a philosophy of 'compile once for all'; all parameters can be selected at run time from the user, without any need for recompiling the code.

GANDALF contains multiple implementations of many important algorithmic features, such as hydrodynamics, the SPH smoothing kernel, *N*-body integration schemes, the spatial decomposition tree and more. If the codes were to inquire about the choice of an algorithm (e.g. how to compute the pressure of a particle) every time it is called, this would require an excessive use of *if–else* statements. Moreover, such a code would be inflexible when adding additional algorithms (e.g. a new EOS); every time a new algorithm is added, every relevant *if* statement called in the code base would need to be modified. To solve this problem, we use the so-called strategy' pattern proposed in the seminal book of Gamma et al. (1995). Different algorithms for performing the same task (e.g. an isothermal or adiabatic EOS; Fig. 2) are coded as different classes inheriting from a common 'parent' class (the EOS class). The parent class declares in its interface a virtual pure function (e.g. *ComputePressure*) that the different strategies implement. 'Users' of the algorithm (e.g. the

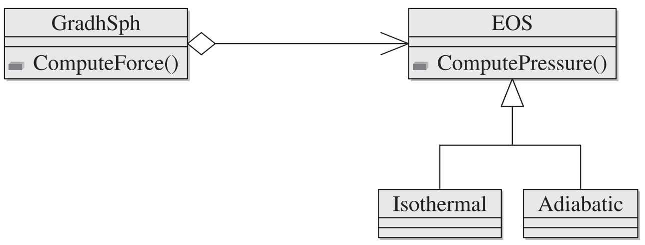


Figure 2. An (idealized) example showing how we use the strategy pattern in GANDALF. In multiple places, the code needs to compute the pressure of a particle; this is accomplished by calling a function defined in an abstract class 'EOS'. At code startup (typically depending on the parameters passed in by the user), it has been decided what the concrete implementation is (e.g. an adiabatic or isothermal EOS); the code that needs the pressure does not need to be aware of how this is computed.

SPH force calculation) only work through a pointer to the parent class, and do not need to behave differently depending on the exact strategy adopted. Using this approach, we can separate the code where we choose the algorithm (typically done at code start-up) from the location where we invoke it, avoiding a long list of *if*s, for the benefit of code clarity and extensibility.

Another example of object orientedness is the use of a well-known feature of C++ called *templates*. This is a way of expressing polymorphism at compile time rather than at run time, and as such incurs less overheads. Therefore, we use this feature in performance critical sections of the code. For example, in a particle-based algorithm the smoothing kernel is a critical part of the code. GANDALF supports several kernels, and we achieve this by templating the functions that use the kernel with the template class. This has the advantage that the kernel can be inlined (early testing has shown that this can lead to a performance improvement up to 30 per cent) and we can retain this performance while still being able to select the kernel at run time (i.e. there is no need to recompile the code if one wishes to change the kernel).

Finally, the last example of best object-oriented practices is the use of composition over inheritance. The top level class present in the code is the simulation class, which governs for example the flow of the main loop (see the flow chart in Fig. 3). While we do use inheritance to distinguish the meshless algorithms from SPH (e.g. we have a SPHSimulation class and a MeshlessSimulation class), there are many other individual algorithms available for use in the code, most of which have different options. This could lead to hundreds of different simulation types. We solve this problem by having multiple classes, each one responsible for one of the main subtasks of the main loop.

Fig. 3 shows some of these subclasses; the main simulation class stores a pointer to each one of them. The main loop starts with integrating the particles in time (a task handled by a dedicated integrator class). We then build/update the structure used to retrieve neighbours (the tree) and proceed to the core of the algorithm: computing smoothing length and hydro forces. These tasks are also handled by the tree; the actual calculations of smoothing lengths and forces are subsequently delegated to an Sph class once the neighbours of a particle have been retrieved. At this point, we compute the acceleration on to the stars and accrete gas on to the sinks. Then, we compute the time-step (this is handled by the simulation class itself), compute the dust forces and finally correct the time integration of the particles with the newly computed accelerations (if necessary, depending on the time integration scheme). The meshless loop closely follows the SPH one, with two important differences.

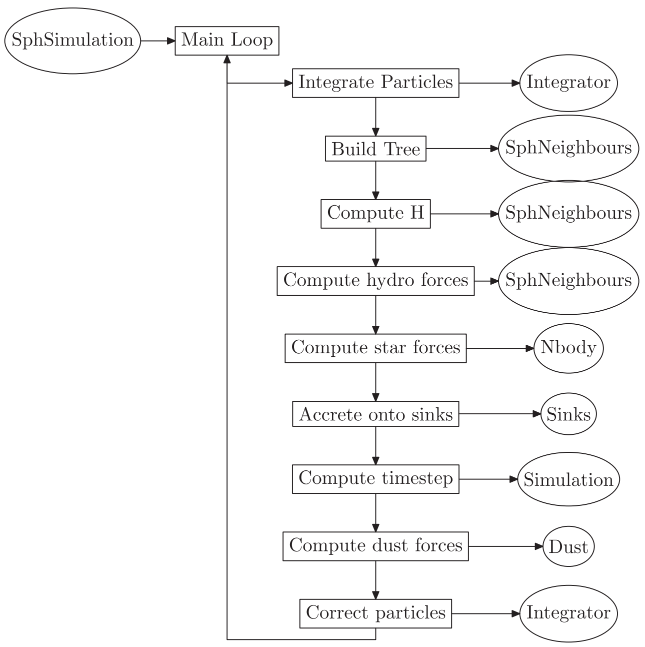


Figure 3. A flow chart showing of the flow of the main integration loop of GANDALF for the SPH case. The circles indicate the class responsible for each action (shown in the rectangle).

The first one is that the force calculation is replaced by two separate loops, one to update the gradient matrices and one to compute the fluxes. The second one is that, while for SPH we compute the gravitational acceleration together with the hydro forces (if both are present), for the meshless we must do it in two independent loops to preserve the second-order accuracy in time of the integration.

5.2 Parallelization

Our approach to parallelization in GANDALF follows recent trends in HPC. We have parallelized the code using both OPENMP and MPI. This hybrid parallelization allows the code to be used flexibly on different architectures. Modern hardware tends to be composed of few machines ('nodes') containing each several cores, interconnected by high-performance, low latency links (such as InfiniBand). An OPENMP only approach has the disadvantage that it is not possible to use more cores than what is available on a single node. Conversely, a pure MPI approach, while capable of running on any arbitrarily large number of nodes, does not take advantage of the fact that the different threads inside the same node are able to share the same memory, and no communication is needed between them. The use of hybrid parallelization allows us to have the best of both approaches.

5.2.1 OPENMP parallelization

The OPENMP parallelization strategy in GANDALF is straightforward in that the majority of the CPU time is spent in simple loops over the active particles, such as the calculation of the smoothing length (common to both SPH and the MFV schemes) and the calculation of the forces (for SPH) or the calculation of gradient matrices and fluxes (for the MFV schemes). In these loops, the computation for each particle is independent, which makes adding OPENMP parallelization trivial. Only in very few places, we need locks or atomics, which can limit the scaling. As we mentioned in Section 4.2, we walk the tree for the particles in a cell rather than for single particles; a single unity of work for OPENMP is thus an active cell rather than each active particle.

The parallelization of the KD tree construction is less straightforward. The tree construction proceeds by bisecting repeatedly the particles on each tree level. The construction of the first level can be performed only by one thread. On the second tree level, there are two sets of particles, each one of which can be processed independently. This allows us to extract parallelism by assigning a thread to each one. We apply this strategy recursively to the each level; we note that, if $N_{\rm threads}$ are available, we need $2^l >= N_{\rm threads}$, where l is the tree level in order to keep all threads busy and obtain reasonable work-sharing. Typically $N_{\rm particles} \gg N_{\rm threads}$, so that eventually all the threads are busy building the tree. However, the bisection is typically an operation $\mathcal{O}(N)$ (GANDALF uses the algorithm included in the C++ standard library, which is usually introselect), which means that the construction of each level takes roughly the same CPU time. Because the constructions of the first levels is done essentially in serial, it will limit the optimal scaling that can be reached during tree build. Further improvements to our strategy are only possible by parallelizing the select algorithm that performs the bisection.

Finally, for completeness we have also parallelized most of the other operations in the code of order $\mathcal{O}(N_{\rm particles})$, such as time integration, the calculation of the time-step and the calculation of the thermal properties, although they do not dominate the wall clock time.

5.2.2 *Hybrid OPENMP/MPI parallelization*

Typically shared-memory HPC machines contain 16 cores which limits the problem sizes that can be investigated with GANDALF. In order to extend this to more processors (a few 10 s, if not ∼100 cores), we have implemented a hybrid OPENMP/MPI parallelization. The typical usage in GANDALF is to use OPENMP inside each shared-memory node and use MPI to communicate between nodes.

We use domain decomposition via a KD tree to assign the particles to each MPI process. This imposes the limitation that the number of MPI processes must be a power of 2. Each MPI node constructs 'pruned' versions of their trees to send to the other processors. These are simplified trees, with a smaller number of levels than the full trees. The pruned trees allow each node to have an large-scale approximation of the mass distribution in the other domains, which is useful for many purposes. We note that the pruned trees in our implementation are not locally essential trees; i.e. they are not necessarily deep enough to allow other processor to compute the gravitational force resulting from the domain.

Some steps of the algorithms in GANDALF (e.g. the density calculation, the gradient estimation in the meshless and the dust forces calculation) need information about the neighbours from other domains. This is accomplished by creating 'ghost' particles on each local domain. Each node uses the pruned tree to establish which of its particles might be ghost particles on other nodes. When using periodic boundaries, we also create MPI ghosts of periodic ghosts. Our algorithm is generic and does not need to treat differently this case.

In other steps, where the ghosts would be modified by the interaction with the local particles (e.g. in the SPH force calculations or the MFV/MFM flux calculations), we have decided to use particle exchange rather than ghosts. This has the advantage that it allows us to treat hydrodynamics and gravity in the same way, and avoids the need to send information about all the ghosts even if only few of them are active. Operationally, when we find that a particle is too close to the boundary or the pruned trees of the other domains are not deep enough for gravity calculations, the particle is sent to the neighbouring domain. The other domains compute the contribution to the force from its local particles and then returns back to the original domain this partial force, which can be added to the total force.

Another significant part of the MPI code deals with transferring particles when they move between domains. The boundaries of the domains need to be updated regularly to maintain load balancing. To estimate the new location of the boundary, we assign each particle a fraction of the total CPU work, which depends on its time-step level; the work on each processor is weighted by the CPU wall-clock time used by the MPI node to ensure a correct interprocessor normalization. We use a bisection iteration method to find the best location of the new boundary, using the pruned trees to compute the new work in the domain. Once the domain boundaries have been updated, particles that are now in different domains are transferred via MPI communication.

5.3 Automated tests

GANDALF contains many different algorithms and types of physics; it is thus important to make sure that any change to the code does not invalidate pre-existing code. To achieve this goal and ensure that no bugs are introduced in GANDALF, we have found invaluable to have a test suite that stresses the different options supported by GANDALF. The experience has shown us that such a test suite needs to be automated: it is impossible to inspect manually every time the results of many simulations. We use the PYTHON library to inspect the results of the simulations run by the test suite, compare them to analytical (or numerical) solutions and check that the overall error is within a given tolerance. Finally, the last requirement is that the test suite must be invoked automatically, or the execution will be procrastinated. We found that the online service TRAVIS-CI,[2] which can be automatically linked to a github repository, perfectly matches this requirement by running the test suite every time a commit is pushed. In this way, during development we receive immediate feedback informing us if a newly added feature has broken any of the existing code.

5.4 PYTHON library

While most of the effort in developing GANDALF has been invested in being able to *run* numerical simulations, this is certainly not enough for making science; being able to *visualize* and *analyse* the outputs is equally important. GANDALF contains a library written in PYTHON dedicated to this task. An excellent software package, called SPLASH (Price 2007) for the visualization of particle-based simulations[3] already exists and it is not the purpose of the library to supersede it. We note that GANDALF snapshot files are fully compatible with SPLASH. While we do provide a very essential subset of the SPLASH functionality in GANDALF (particle and rendered plots), the design principle of the PYTHON library aims to fill a different gap. The goal of the library is to give the user programmatic access (e.g. save in a variable) to the data in the outputs. The library allows us to access the raw data from the simulations (e.g. construct an array containing the smoothing lengths of the particles) and the basic visualizations

[2] https://travis-ci.org/

[3] Although SPLASH is designed for SPH, it can also easily handle outputs from the MFV schemes.

described before (e.g. construct a 2D array containing a rendered plot). Additional functions permit to compare the simulation with analytical solutions (when known) and to repeatedly apply an analysis function to each snapshot in a simulation, making easy to plot a quantity as a function of time. The goal is to simplify writing analysis scripts. As a bonus, having some plotting capabilities built-in the code allows us to inspect the simulation while it is running. We found this feature very convenient, while developing the code. In the same way, we hope that future users wanting to add some physics to GANDALF will find it useful as well. Finally, having interfaced GANDALF with PYTHON makes it possible to set-up the initial conditions directly in PYTHON, in the case the user is not familiar with C++.

As already mentioned, following the general trend in scientific computing, the language of choice for this library is PYTHON. This choice is motivated by the extreme flexibility of the language, its easiness to use, and the existence of libraries devoted to numerical analysis and publication-ready plotting (namely MATPLOTLIB). As GANDALF itself is written in C++, we need a 'bridge' to make the two languages speak. For this purpose, we make use of the SWIG library. With SWIG GANDALF can be compiled as a shared library object and therefore loaded into PYTHON as any standard PYTHON module.

6 TESTS

In order to demonstrate the fidelity and limitations of the various components of GANDALF, we have a performed a wide range of tests of the code. Many of these test cases deliberately overlap with those performed both with AREPO (Springel 2010) and GIZMO (Hopkins 2015) in order to more easily compare them to GANDALF. Since GANDALF is aimed more towards Star and Planet Formation problems (as opposed to Galaxy and Cosmological problems), we have substituted some Cosmology-oriented tests for others that are important for Star and Planet Formation scenarios. In most hydrodynamical test cases, we perform with three different options; (i) grad-h SPH, (ii) MFV and (iii) MFM.

6.1 Soundwave test

The goal of this test is to demonstrate that GANDALF correctly implements the hydrodynamical and time integration algorithms, preserving second-order convergence when dealing with smooth flows. We apply a low-amplitude sinusoidal density and velocity perturbation of the form

$$\rho(x) = \rho_0 \left(1 + A\, \sin\{kx\}\right), \tag{95}$$

$$v(x) = A\, c_{\rm s}\, \sin\{kx\}, \tag{96}$$

where A is the density perturbation amplitude, $c_{\rm s}$ is the sound speed of the unperturbed gas and $k = 2\pi/\lambda$ is the wavenumber. To investigate the scaling of the error with resolution, we calculate the L1-error norms of the density field, i.e.

$$|{\rm L1}| = \frac{1}{N}\sum_{i=1}^{N} |\rho_i - \rho(x)|, \tag{97}$$

where ρ_i is the particle density and $\rho(x)$ is given by equation (95), as a function of particle number, N. The L1-error norm is expected to scale as $\propto N^{-2/D}$, where D is the dimensionality.

The initial conditions are created following Stone et al. (2008). A set of N particles are placed in 1D at equidistant intervals along the x-axis between $x = 0$ and 1. The sinusoidal density perturbation is created by slightly perturbing the positions of the particles along the x-axis to match the correct density profile (see for example Hubber, Goodwin & Whitworth 2006, for a description of creating a sinusoidal density field). We use values $\rho_0 = 1$, $A = 10^{-6}$, $c_{\rm s} = 1$ and $\lambda = 1$ for our perturbation.

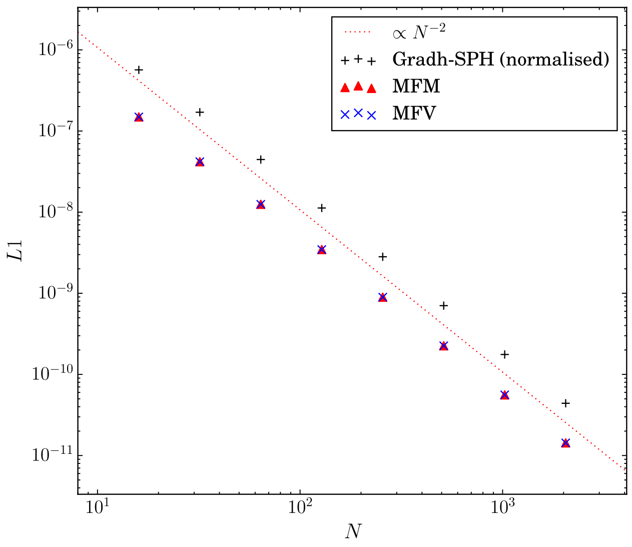


Figure 4. The L1-error norm versus the simulation particle number for the soundwave test using the grad-h SPH (black crosses), MFV (blue crosses) and MFM (red triangles) methods in 1D. For smooth fluid flows, we would expect the errors to be dominated by the spatial and temporal integration errors of the numerical scheme, which in all cases should be second order. Therefore, the L1-error norm should scale as $\propto N^{-2}$ in 1D (red dotted line).

Fig. 4 shows the L1-error norm as a function of particle number for all simulation modes presented here. The MFV (blue crosses) and MFM (red triangles) schemes all scale with the expected $L1 \propto N^{-2}$ error norm (red dotted line) for both low and high resolutions, similar to the results found by Hopkins (2015). For the SPH simulations, one important caveat is that the SPH density sum (equation 5) results in a consistent fractional offset/error from the true uniform density of less than one per cent (for the kernels employed in GANDALF). Normally, this is unimportant in simulations but can affect this test where there is a density perturbation of smaller amplitude. Hopkins (2015) attempts to fix this problem by iterating the particle positions; however at high resolutions this error eventually dominates, breaking the second-order convergence. Since here we are interested in showing second-order convergence in order to test our implementation, we perform our analysis of the SPH simulations by normalizing the average density to ρ_0 (as measured from the simulation itself); this removes the zeroth-order error from the L1 norm. With this normalization applied, we can see that also the SPH results scale with the expected $L1 \propto N^{-2}$ trend, since the spatial error is dominated by the smoothing kernel errors.

6.2 Shocktube tests

Shocktube tests are typically used to test the shock capturing ability of a hydrodynamical code. We use two different equations of state (isothermal and adiabatic) in what follows to test our implementation in both cases (notice that the energy equation is evolved only in the latter case). The initial conditions are set-up in 1D by creating a uniform line of particles in contact to represent the left and right

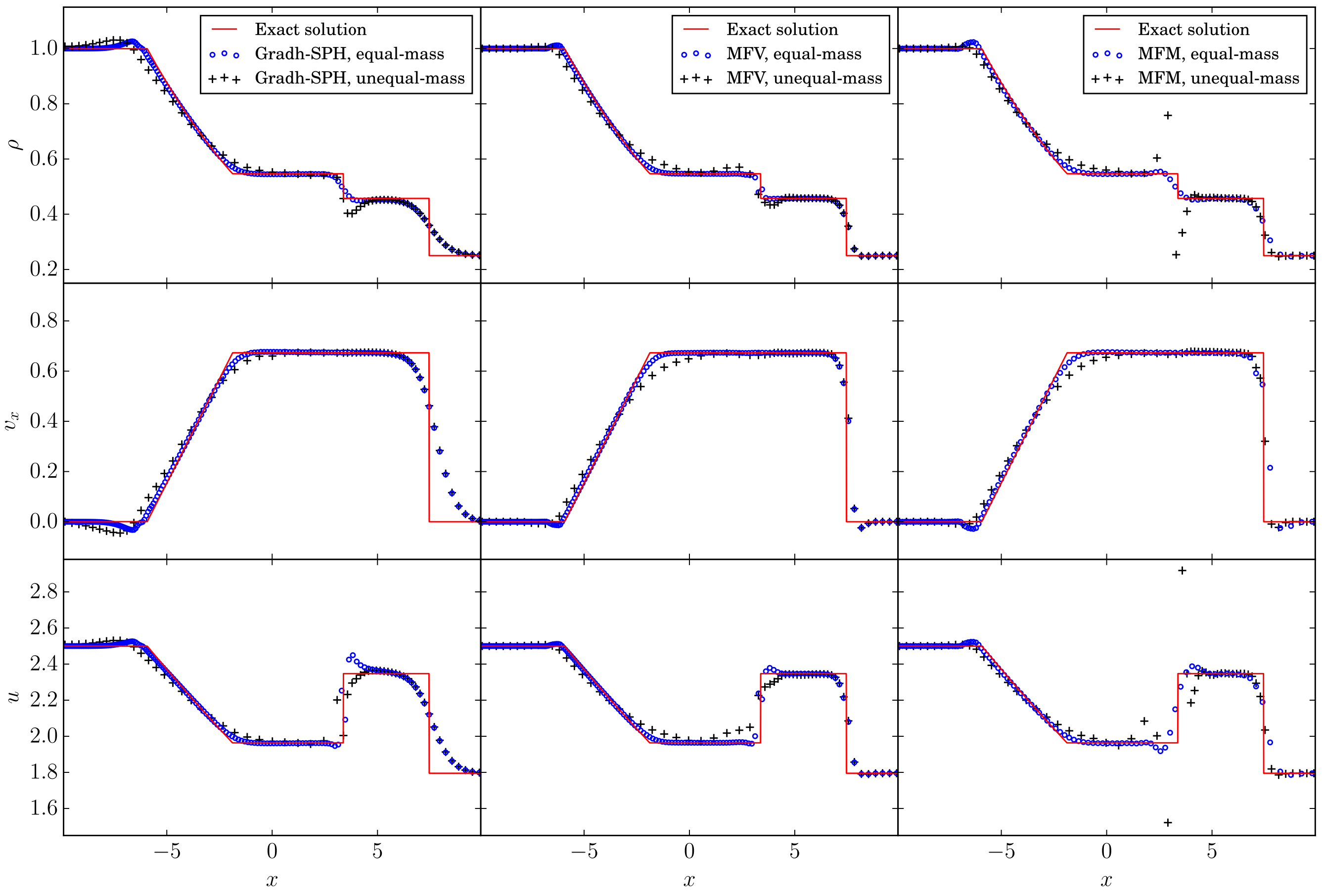


Figure 5. Simulations of the adiabatic Sod test using the grad-h SPH (first column), MFV (second column) and MFM (third column) using initial conditions with equal-mass particles (black plus symbols) and equally spaced particles (blue open circles) at $t = 5.0$. Plotted are the particle density (first row), velocity (second row) and internal energy (third row) for each case including the analytical solution from the Exact Riemann solver (red line).

states. The set-up is similar (albeit with slightly higher resolution) to the same test performed by both Springel (2010) and Hopkins (2015) to allow easy comparison with those two papers. We use the standard Monaghan (1997) prescription for artificial viscosity without limiters for SPH simulations and the Hopkins (2015) limiter for MFV/MFM simulations. The LHS (i.e. $x < 0$) gas state is $P_{\rm L} = 1$, $\rho_{\rm L} = 1$ and $v_{\rm L} = 0$ and the RHS ($x > 0$) is $P_{\rm R} = 0.1795$, $\rho_{\rm R} = 0.25$ and $v_{\rm R} = 0$ in a computational domain of size $-20 < x < 20$. For the adiabatic case, the gas obeys an ideal-gas EOS, $P = (\gamma - 1)\,\rho\,u$, where $\gamma = 1.4$. For the isothermal case, the gas obeys an isothermal equation of state where $c_{\rm s} = 1$ so $P_{\rm L} = 1$ and $P_{\rm R} = 0.25$. We consider two different sets of initial conditions; (i) the LHS contains 240 particles and the RHS contains 60 particles (i.e. equal-mass particles); (ii) both the LHS and RHS contain 60 particles each (i.e. equally spaced particles).

6.2.1 *Adiabatic shocktube*

Fig. 5 shows the results for the adiabatic shocktube for all cases at the final simulation time $t = 5$. For all simulation types, the general form of the density, velocity and pressure profiles are captured correctly, in line with the results of Hopkins (2015), proving the correctness of our implementation of the meshless schemes. We also recover two features noted by Hopkins (2015); SPH in general has larger overshoots and undershoots at the discontinuities for equal-mass initial conditions (blue open circles) and a slightly higher diffusivity (the jumps are not as sharp).

For the equally spaced (non-equal mass) initial conditions (black crosses), we find a more significant dip in the density at the contact discontinuity for SPH and MFV in line with Hopkins (2015); however, they did not show results for MFM. We find that this method has a much stronger ‘blip’ in both the density and energy plots at the discontinuity. We interpret this feature as a wall-heating effect; the lack of mass advection in MFM prevents any (artificial) numerical mixing which can smooth out this blip. SPH and MFV are instead more diffusive due to, respectively, artificial viscosity and mass advection. A slightly more diffusive Riemann solver might allow this blip to be diffused away.

We plot the L1-error norms versus the particle number in Fig. 6. In a shocktube problem, errors near the shock front will dominate the total error in quantities such as the density. In the vicinity of the shock, the numerical schemes should reduce from second (or higher) to first order, since the effect of artificial viscosity, or slope limiters in Godunov codes, is to reduce the scheme to first order to satisfy Godunov’s theorem (e.g. Toro 1997). All the methods broadly follow the expected $L1 \propto N^{-1}$ scaling.

6.2.2 *Isothermal Sod shock*

We perform the same test using an isothermal EOS. The purpose of this test is to test our implementation of the isothermal Riemann

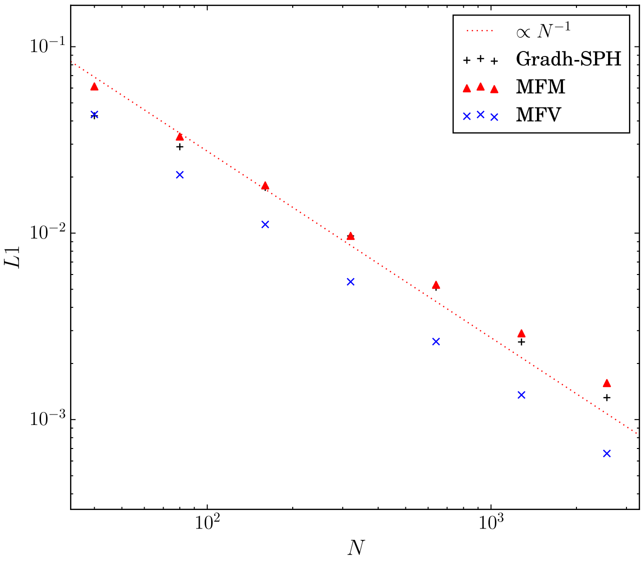


Figure 6. Plots of the L1-error norm versus the simulation particle number for the adiabatic Sod test using the grad-h SPH and MFV methods in 1D. For problems involving shocks, the shock error dominates the total error reducing what are nominally second-order schemes to first order. A line scaling as $L1 \propto N^{-1}$ is shown for comparison.

solver. In Fig. 7, all methods give acceptable results using the equal-mass (blue open circles) initial conditions with similar features to the adiabatic case (but with slightly larger overshoots near the tail of the rarefaction wave). All the methods recover correctly a flat density profile at the original contact discontinuity (although with a small oscillation for the MFV case). For the equally spaced (non-equal mass) case, the methods show instead more prominent numerical artefacts near the contact discontinuity.

6.3 Sedov blast-wave test

The Sedov–Taylor blast wave is a demanding test of the accuracy of energy conservation and of the individual time-stepping algorithm of a particle code; Saitoh & Makino (2009) showed that without a time-step limiter one gets catastrophic results. This is important in many astrophysical applications where a sudden energy input may be triggered by supernovae explosions or high-energy feedback from accreting massive stars. In GANDALF, we provide two different time-step limiters, following Saitoh & Makino (2009) and Springel (2010), and we perform this test to benchmark them.

We set-up a 2D Sedov–Taylor blast-wave simulation by creating a cubic lattice containing 64^2 particles in the region $-1 < x < 1$ and $-1 < y < 1$. The particles are given an equal mass to give a uniform density of $\rho = 1$. We assign the total energy of the explosion ($E = 1$) to the particles within a single smoothing kernel of the origin, where each particle's contribution is weighted by its smoothing kernel value. For both the grad-h SPH and MFV schemes, we perform simulations with (i) global time-steps and (ii) 10 time-step levels using the time-step limiter. For SPH, the only option is the Saitoh & Makino (2009) limiter, while for the meshless we test also the Springel (2010) limiter. We also perform additional simulations with multiple time-step levels with no limiter to check that, confirming the results of Saitoh & Makino (2009) and Hubber et al. (2011), in this case we fail to reproduce the analytical result, getting a noisy density field and wrongly predicting the location of the shock. In this case, we note that the MFV method is less robust than SPH and it is prone to crash when using multiple time-step levels; we cannot run the test to completion without using a time-step limiter.

Fig. 8 plots the density profile at $t = 0.06$ for all cases along with the semi-analytical solution (red line). Both the SPH and MFV schemes follow a similar pattern with the various time-step options. For global time-steps (first column), they both reproduce the semi-analytical solution reasonably well, including most importantly the shock position. All the methods under-resolve the peak shock density due to the finite resolution and the use of smoothing kernels. The MFV scheme resolves the peak slightly better than SPH, with a peak density of just over 3 (compared to just under 3 for the SPH), although the difference is smaller than that found by Hopkins (2015). We note that the kernel weighting at the base of SPH and the meshless methods will always lead to some smoothing of sharp features. Using either of the two implemented time-step limiters, the Saitoh & Makino (2009) limiter (second column) or the Springel (2010) limiter (3rd column), improve the simulation results considerably and are nearly indistinguishable from the single time-step level results, proving the correctness of our implementation. As explained in Section 2.1.5, the Saitoh & Makino (2009) do not enforce energy conservation; for example at the end of the simulation the fractional energy error has gone up to $\sim 10^{-4}$. The Springel (2010) time-step limiter instead is conservative and ensures energy conservation at a level of $\sim 10^{-13}$, which is similar to the result we get with global time-steps. This does not come for free though; the test with the conservative time-step limiter is roughly

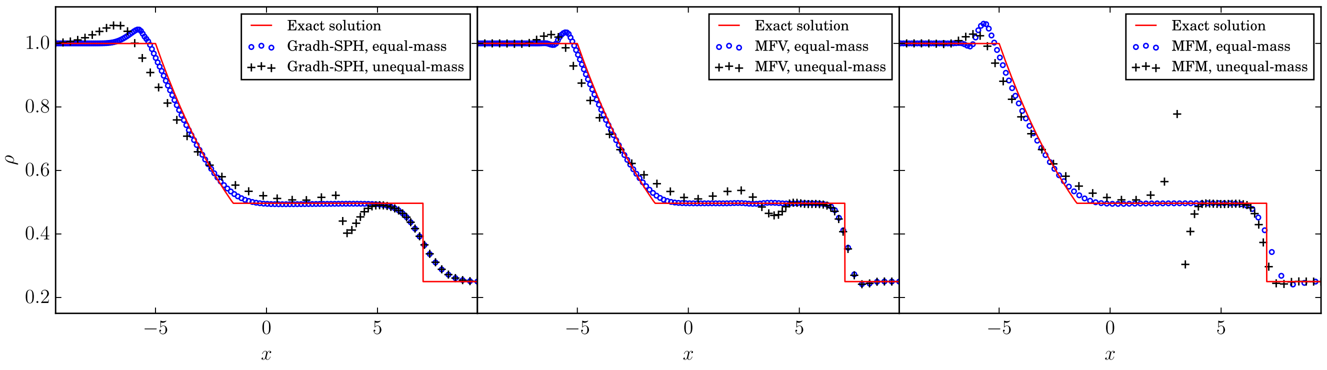


Figure 7. Density profile at $t = 5.0$ resulting from simulations of the isothermal Sod test using the grad-h SPH (first column), MFV (second column) and MFM (third column) using initial conditions with equal-mass particles (black plus symbols) and equally spaced particles (blue open circles). The exact solution is also plotted (red lines).

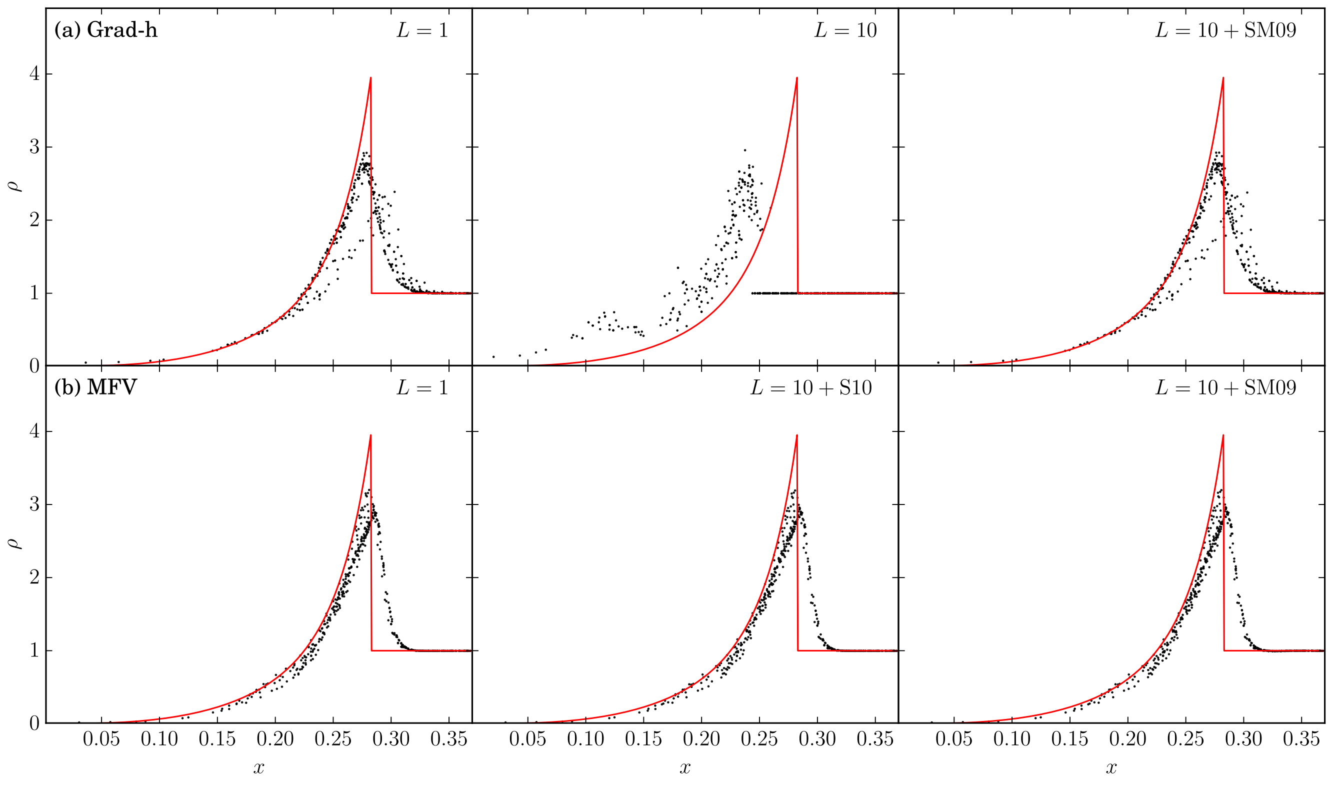


Figure 8. Simulations of the 2D Sedov–Taylor blast-wave test at $t = 0.06$ using grad-h SPH (first row) and the MFV scheme (second row). The first column shows the results with global time-steps. For SPH, we show in the second column the case with 10 time-step levels and no limiter. We do not show this case for the MFV, since it crashes before the end. For MFV, the second column shows instead the results with the Springel (2010) limiter. The third column shows the results with the Saitoh & Makino (2009) time-step limiter. The semi-analytical solution is plotted for comparison (red line). Both time-step limiters perform very well and the results are indistinguishable from the global time-step run, while using individual time-steps without limiter clearly leads to wrong results.

20 per cent more expensive in terms of computational time. Even in this case, the time-step limiter still allows a saving of almost a factor of 3 compared with global time-steps ($\sim$11.4 s compared to $\sim$4.2 s).

6.4 Gresho–Chan vortex

The Gresho & Chan (1990) vortex test involves a steady rotating vortex profile in which the rotation is supported by pressure. We study this problem in 2D, 64×64 particles on a cubic lattice on a periodic domain with $-0.5 < x$ and $y < 0.5$. The initial pressure profile is

$$P(R) = \begin{cases} 5 + \frac{25}{2}R^2 & 0 \le R < 0.2 \\ 9 + \frac{25}{2}R^2 - 20R + 4\ln 5R & 0.2 \le R < 0.4 \\ 3 + 4\ln 2 & R \ge 0.4, \end{cases} \tag{98}$$

and the initial (azimuthal) velocity profile is

$$v_\phi(R) = \begin{cases} 5R & 0 \le R < 0.2 \\ 2 - 5R & 0.2 \le R < 0.4 \\ 0 & R \ge 0.4. \end{cases} \tag{99}$$

The initial density is $\rho = 1$ everywhere and the gas obeys an adiabatic EOS with $\gamma = 5/3$. The initial radial velocity profile is set to zero.

The azimuthal velocity profile at $t = 3$ is shown in Fig. 9 for the both MFM and SPH methods. We do not show the results for the MFV method, which are essentially the same as those as the MFM method. In the SPH simulations both the Cullen & Dehnen (2010) viscosity limiter and the Price (2008) artificial conductivity were used. For the meshless, we show the results for the range of slope limiters included in GANDALF. Finally, we explore both the cubic and quintic spline kernels.

The poor performance of SPH in this test is already well known, with the high artificial dissipation leading to a fast damping of the vortex. The Cullen & Dehnen (2010) switch alleviates this somewhat compared to the behaviour of standard SPH (see Rosswog 2015), but the dissipation remains large. The performance of the MFM method is very sensitive to choice of the slope limiter (note that this was reported by Hopkins 2015, but they did not show the differences in their figures), with the most diffusive limiters (i.e. the first-order Godunov scheme, or Heß & Springel 2010) showing the same poor performance as SPH. The least diffusive limiters (i.e. Gaburov & Nitadori 2011 and Springel 2010) show essentially no dissipation, although we do see some broadening of the vortex peak. The Hopkins (2015) limiter falls between the two extremes, showing a modest level of dissipation.

In addition to running the Gresho & Chan (1990) test with 'standard' cubic spline kernel, we have also run the test using the quintic spline kernel for both the SPH and MFM schemes. This highlights the importance of accurate volume and gradient estimates in the presence of strong shear, which acts to disrupt the ordered particle positions, as shown by Rosswog (2015). In the case of SPH, the dissipation is reduced considerably, to a level that is only slightly greater than the MFM with the Hopkins (2015) limiter.

This test demonstrates that using the quintic spline kernel also significantly improves the performance of the MFM methods. The main effect is a reduced level of noise, which consequently results

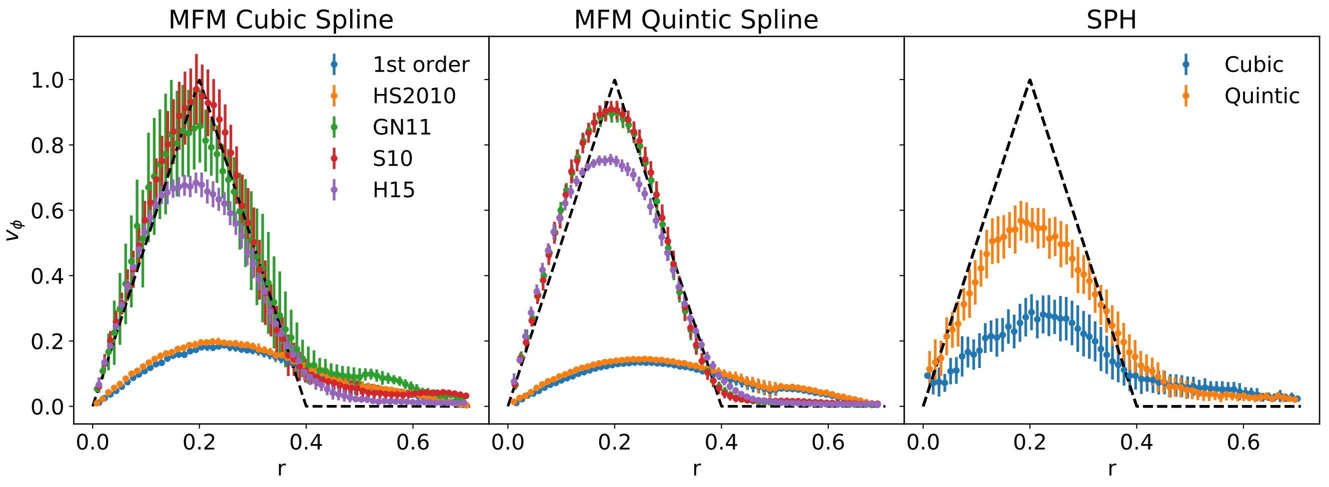


Figure 9. The Gresho vortex test computed with SPH and the MFM schemes, using both the cubic and quintic spline kernels. For the MFM, we also show a range of different slope limiters. For the SPH simulation, the Cullen & Dehnen (2010) switch was used with Price (2008) artificial conductivity. Points and error bars show the mean and standard deviation of particles within each radial bin.

in the slope limiters being triggered less frequently. In practice, this does not much affect the least diffusive methods, where the slope limiters are already triggering very rarely. However, in the case of the Hopkins (2015) limiter, the reduced noise does reduce the level of dissipation. Finally, for the most diffusive cases, the reduced noise does not reduce the triggering of the slope limiter, and thus the predominant effect is one of lower effective resolution (due to the large smoothing volume).

6.5 Gravity tree accuracy

In this test, we set-up a random distribution of particles in a uniform spherical volume of radius 1. We compute the gravitational acceleration on each particle using both the tree and direct sum; the comparison between the two informs us on the accuracy of tree and how it varies with the parameters of the tree. We compute the total net error done in the gravitational acceleration as

$$|\delta \boldsymbol{a}| = \left(\frac{1}{N} \sum_{i=1}^{N} \left\{ \frac{|\boldsymbol{a}_i^{\mathrm{TREE}} - \boldsymbol{a}_i^{\mathrm{DIR}}|^2}{|\boldsymbol{a}_i^{\mathrm{DIR}}|^2} \right\} \right)^{1/2}, \tag{100}$$

where $\boldsymbol{a}_i^{\mathrm{TREE}}$ and $\boldsymbol{a}_i^{\mathrm{DIR}}$ are the accelerations computed via the tree and direct sum, respectively. For this tests, we have employed a resolution of 16 k particles; the number of particles in each leaf cell has been held fixed to 6.

In Fig. 10, we show the mean gravitational acceleration error using different tree opening criteria and different multipole approximations. As expected the errors become smaller in all cases when the tree is required to open more cells. In addition, using higher multipole approximations also improves the accuracy of the tree as expected; we see a clear trend when going from the monopole methods to the cell quadrupole and then to the full quadrupole.

Fig. 11 shows the CPU time to compute the gravitational forces as a function of the accuracy. In our implementation, the quadrupole method calculates the force to a given accuracy with the least amount of CPU time and is therefore the most optimal choice of multipole expansion. The quadrupole method results in a given accuracy by opening less cells during the tree walk, but performing more work per cell in computing the extra quadrupole terms. Whether this is more efficient than opening more cells only using the monopole depends largely on the details of the implementation, and for GANDALF, the tree is faster doing more iterations over distant cells, rather than opening more cells overall. One reason for this behaviour might be that we make local copies of the quadrupole moments of the distant cells, and hence iterating over them is relatively fast since they are already held in the CPU cache. In this problem, there is very little difference between the different opening criteria, as they all reach a given accuracy in roughly the same time. However, this might change with different density fields.

6.6 Jeans instability test

The Jeans instability test (Hubber et al. 2006) is one of the few problems with periodic gravity with known solutions and can be used to validate the Ewald periodic gravity component of the code. This test sets up a simple sinusoidal density perturbation in an otherwise uniform medium and then monitors the evolution of the density and the velocity field compared to that predicted by the simple Jeans theory (e.g. Binney & Tremaine 2008).

The initial conditions are set-up following Hubber et al. (2006). The density field is set-up in a similar fashion to the 1D sound-wave test (equation 95), where the particles positions are adjusted to create the required density field (as opposed to altering the particle's masses). The initial velocity for all particles is zero. These initial conditions lead to solutions which are standing waves rather than travelling waves as in the classical Jeans solution. The time-dependent solution is given in Hubber et al. (2006). For stable ($\lambda \ll \lambda_{\mathrm{J}}$) wavelengths, the perturbations oscillate as sound waves. The oscillation period is

$$T_{\mathrm{OSC}} = \left(\frac{\pi}{G\,\rho_0} \right)^{1/2} \frac{\lambda}{\left(\lambda_{\mathrm{J}}^2 - \lambda^2\right)^{1/2}}. \tag{101}$$

For unstable ($\lambda \gg \lambda_{\mathrm{J}}$) wavelengths, the perturbation growth timescale (defined as the time for the perturbation to grow from an amplitude of A to $A\cosh\{1\} \sim 1.56A$ is

$$T_{\mathrm{COL}} = \left(\frac{1}{4\,\pi\,G\,\rho_0} \right)^{1/2} \frac{\lambda}{\left(\lambda^2 - \lambda_{\mathrm{J}}^2\right)^{1/2}}. \tag{102}$$

Rather than fixing the Jeans length and alter the perturbation wavelength, we fix the perturbation wavelength (so the IC set-up is

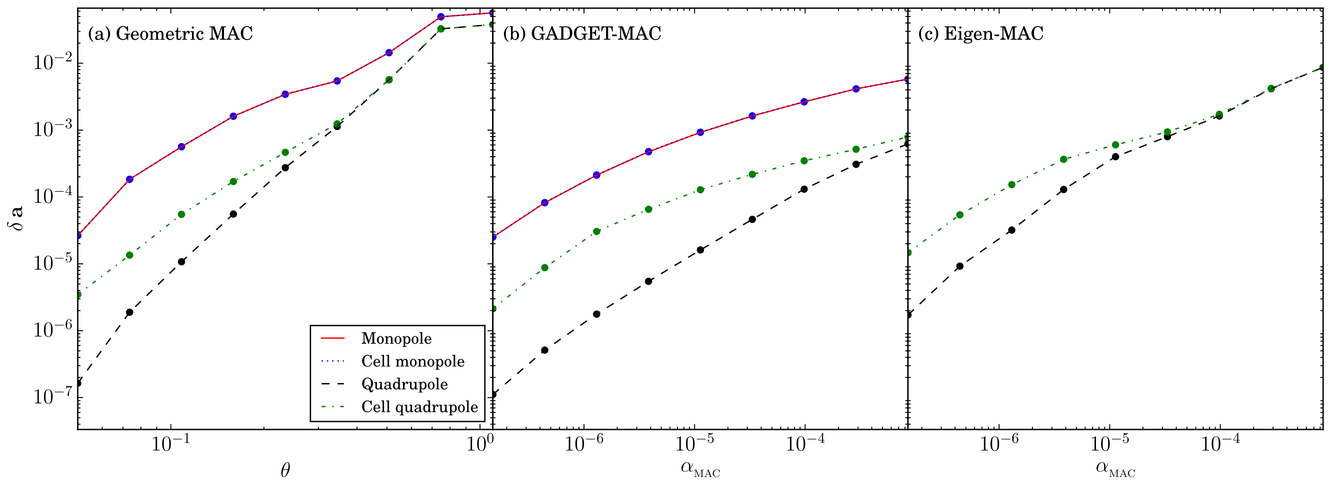


Figure 10. The mean gravitational acceleration error (using equation 100), while computing the initial forces for particles in a uniform density sphere using the KD tree using (a) the geometric opening-angle criterion as a function of the maximum opening angle, $\theta_{\rm MAX}$, and (b) the GADGET-2 (Springel 2005) and (c) eigenvalue-MAC (Hubber et al. 2011) as a function of the error tolerance criterion, $\alpha_{\rm MAC}$, while using the monopole (red solid line), cell-monopole (blue dotted line), quadrupole (black long dashed line) and cell-quadrupole (green dot–dashed line) multipole approximations.

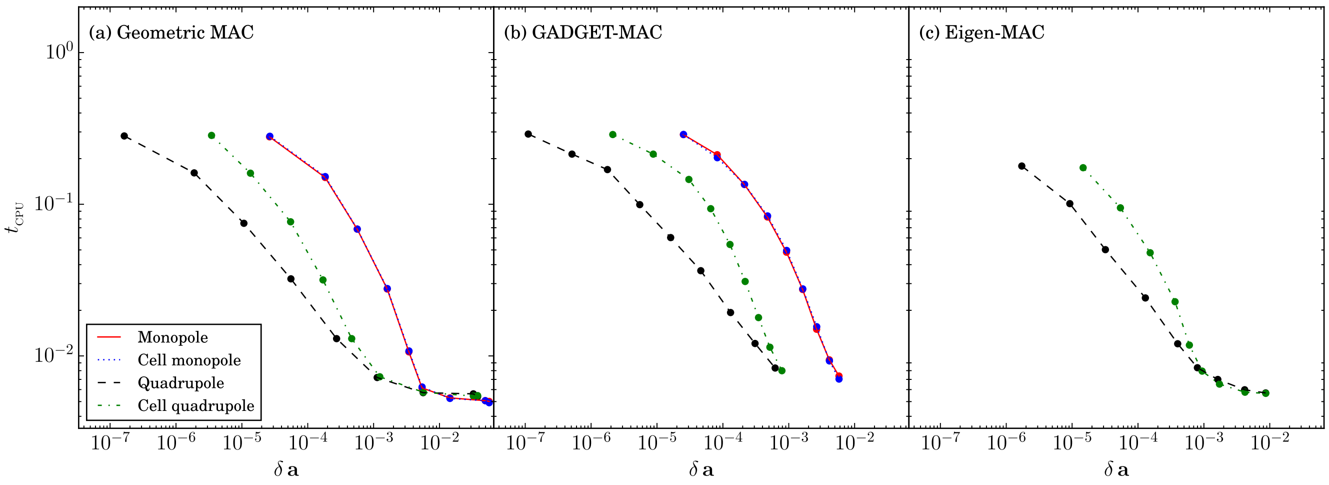


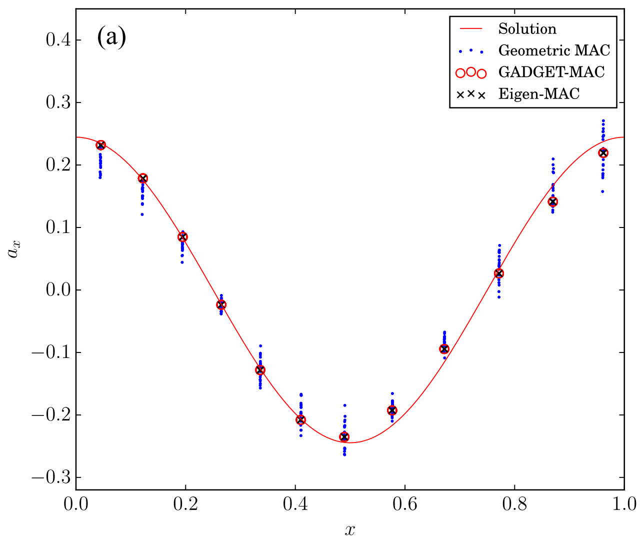
Figure 11. The CPU time (relative to the brute-force $\mathcal{O}(N^2)$ computation) to compute the gravitational forces via the KD tree compared to the mean gravitational acceleration error (using equation 100), while computing the initial forces for particles in a uniform density sphere using (a) the geometric opening-angle criterion (b) the GADGET-2 (Springel 2005) MAC and (c) the eigenvalue-MAC (Hubber et al. 2011). Results are plotted for tree-cell expansions using the monopole (red solid line), cell monopole (blue dotted line), quadrupole (black long dashed line) and cell quadrupole (green dot–dashed line).

always the same) and instead alter the Jeans length via changing the sound speed of the gas. We perform the simulations only for MFM.

We find that this problem is a stringent test of the tree opening criterion, since the contributions to the gravitational accelerations largely cancel out and sum up to exactly zero for no perturbation. We plot in Fig. 12(a), the gravitational acceleration computed with different opening criteria. While the GADGET MAC and the eigenvalue MAC perform quite well in comparison with the analytical solution, the geometric MAC criterion produces a very inaccurate and noisy acceleration. This is not surprising since the criterion does not try to enforce a given error on the acceleration as instead the other two do, which leads to more cells being opened if the acceleration is small. In Fig. 12(b), we plot the oscillation and collapse time-scales for various ratios of the perturbation to Jeans wavelength, $\lambda/\lambda_{\rm J}$. We can see that both evolutionary modes of the perturbation (oscillation and collapse) are correctly realized, i.e. oscillation only for $\lambda < \lambda_{\rm J}$ and collapse only for $\lambda > \lambda_{\rm J}$, similar to the results of Hubber et al. (2006) for so-called 'Vanilla' SPH. As in the previous case, we see that the geometric MAC has a worse agreement with the analytic solution. For the other two criteria, the oscillation period and the collapse time-scale are extremely well matched by the simulations to the theory although all simulations to some degree underestimate the oscillation time-scale and overestimate the collapse time-scale due to smoothing and resolution effects.

6.7 Time integration accuracy

In this section, we investigate how well the different N-body time integration schemes available in GANDALF conserve energy, which we take as a metric of global accuracy. These tests are in an indirect way a test also of the hydrodynamics schemes, since they all employ a variant of the Leapfrog integrator. We will highlight in



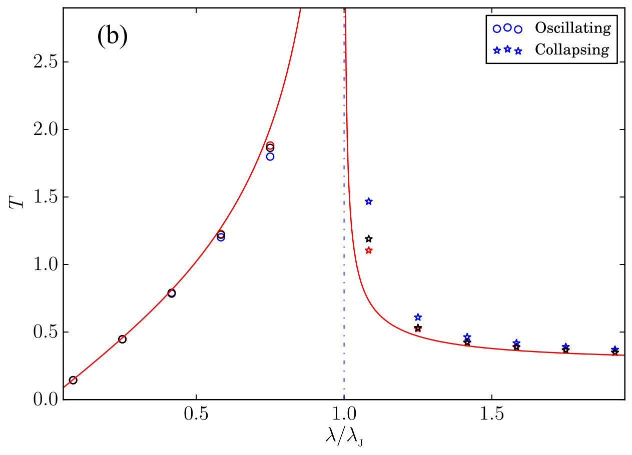


Figure 12. (a) The x component of the gravitational acceleration computed at $t = 0$ for the sinusoidal density perturbation used for the Jeans test using periodic corrections with the KD tree using (i) the geometric MAC (blue dots), (ii) the GADGET MAC (red open circles) and (iii) eigenvalue MAC (black crosses). For reference, we plot also the analytical solution. (b) The characteristic time-scales for the evolution of sinusoidal perturbations in the Jeans instability test. For stable wavelengths (i.e. $\lambda/\lambda_{\rm JEANS} < 1$), the sinusoidal perturbation oscillates with a period given by equation (101). For unstable wavelengths (i.e. $\lambda/\lambda_{\rm JEANS} > 1$), the perturbations grow with a time-scale given by equation (102). The analytical solutions (equations 101 and 102) are plotted in red with the blue-dashed line marking the asymptote where $\lambda = \lambda_{\rm JEANS}$ (and where the oscillation/growth time-scales tend to infinity). Oscillating simulations (open circles) and collapsing simulations are plotted using the (i) geometric MAC (blue), (ii) GADGET MAC (red) and (iii) eigenvalue MAC (black) criteria for walking the KD tree.

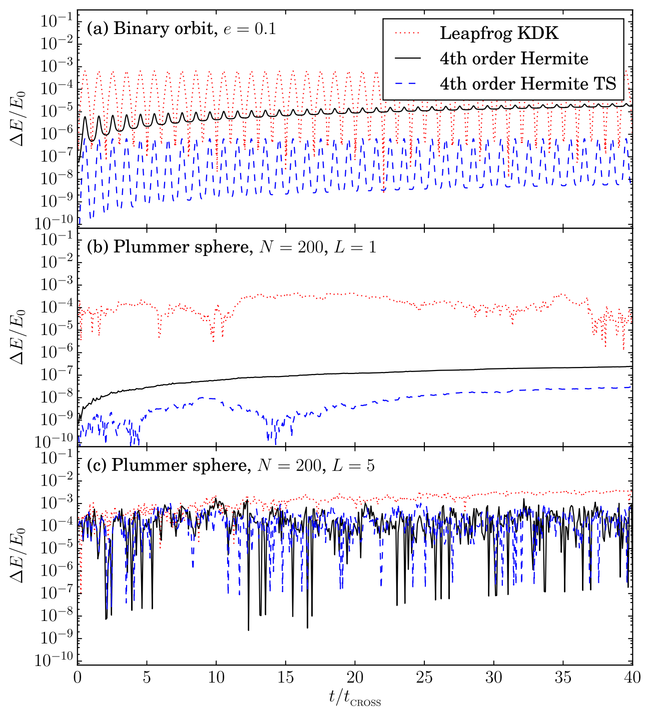


Figure 13. The total cumulative fractional energy error for N-body simulations integrating (a) the orbit of an equal mass binary system with a low eccentricity ($e = 0.1$), (b) the evolution of a Plummer sphere containing $N = 200$ stars with global time-steps, and (c) the Plummer sphere using block time-steps with five time-step levels. For all cases, we perform the integrations using the Leapfrog KDK (red dotted line), the standard fourth-order Hermite (solid black line) and time-symmetric fourth-order Hermite (dashed blue line) schemes.

particular how in N-body dynamics integrators of order higher than the Leapfrog are necessary to guarantee good energy conservation.

6.7.1 *Binary orbits*

A binary star with two masses in a bound orbit is the simplest known N-body test problem with an analytical solution and is useful in demonstrating the fidelity of N-body integration schemes. We have simulated a mildly eccentric ($e = 0.1$) binary orbit for 40 orbits to highlight the differences in the various schemes. In Fig. 13(a), we plot the energy error as a function of time for three integration schemes, the Leapfrog KDK (red dotted line), the standard fourth-order Hermite (solid black line) and the time-symmetric fourth-order Hermite (dashed blue line). There are two trends to highlight; an oscillation in the energy error (with the same period as the binary orbit) and a long-term error growth. The two symplectic schemes are characterized by strong oscillations in the energy error which span 3–4 orders of magnitude, however they do not show a long-term growth in the error. This is expected since these schemes are time reversible. In contrast, the standard fourth-order Hermite scheme shows a much smaller error oscillation, but also a slow long-term increase in the energy error. Initially, the energy error is only slightly higher than the time-symmetric Hermite scheme, but it slowly increases towards the regime occupied by the Leapfrog scheme (cf. Binney & Tremaine 2008, fig. 3.21).

6.7.2 *Plummer sphere*

A Plummer sphere is a popular and simple stellar cluster profile used often in basic N-body cluster simulations and has been modelled extensively in the literature (e.g. Aarseth, Henon & Wielen 1974; Aarseth 2003; Binney & Tremaine 2008). The mass density profile for a Plummer sphere is

$$\rho(r) = \frac{3\,M}{4\,\pi\,a^3}\left(1 + \frac{r^2}{a^2}\right)^{-5/2}, \tag{103}$$

where M is the total mass and a is the Plummer radius. The 1D velocity dispersion of the stars as a function of radius, $\sigma(r)$, is

$$\sigma^2(r) = \frac{G\,M}{6\,a}\left(1 + \frac{r^2}{a^2}\right)^{-1/2}. \tag{104}$$

A detailed explanation of how to generate initial conditions for a Plummer model with stars is given by Aarseth et al. (1974). When including gas, we set-up the Plummer spheres similar to that outlined in Hubber et al. (2013a). The positions of the particles are selected with the same Monte Carlo algorithm, but the gas is given a sound speed equal to the local velocity dispersion. We perform a simulation of a Plummer sphere containing 200 equal-mass stars with total (dimensionless) mass $M = 1$ and Plummer radius $R = 1$. We truncate the Plummer sphere at a radius of $R_{\rm MAX} = 10\,R$. The Plummer sphere is simulated for 40 crossing times.

The energy errors (Fig. 13b) shows markedly different traits to the simple binary orbit. There is no clear oscillatory error although there are some trends for long-term error growth. The Leapfrog scheme (red dotted line) is the most stable scheme in terms of energy growth, although it also has the largest average energy error: about 2–3 orders of magnitude larger than the other schemes. The Hermite scheme (black line) has a clear long-term growth over the full course of the simulation. The time-symmetric Hermite also has long-term error growth, although about an order of magnitude less than the non-symplectic version. The large energy error with the Leapfrog shows why it is important to use higher order, time reversible integrators for the *N*-body dynamics, in contrast to what is done by most contemporary SPH codes.

6.7.3 Plummer sphere with block time-steps

We simulate the same Plummer sphere as Section 6.7.2 using block time-steps (five time-step levels). The total global errors for all schemes (Fig. 13c) are much higher than for the global time-steps simulation. This shows how multiple time-step levels break energy conservation: force calculations are no longer symmetric leading to momentum non-conservation and subsequent energy errors. Overall, the Leapfrog scheme has an energy error starting near 10^{-5} growing quickly to 10^{-4} and finally almost 10^{-2} by the end of the simulation. The Hermite schemes both tend to have on average a significantly smaller error, of order 10^{-4}.

6.8 Hybrid SPH/*N*-body simulations

Following Hubber et al. (2013a), we perform hybrid simulations containing both stars and gas with Plummer profiles. The gas is initially set so the local sound speed matches the local velocity dispersion; the initial internal energy is thus $u(r) = \sigma^2(r)/(\gamma - 1)$ and subsequently evolves according to an adiabatic EOS. Differently from Hubber et al. (2013a), as explained in Section 3.2 in GANDALF, we take a different symmetrization of particle–particle interactions. In this section, we want to show that we still recover the same behaviour in the evolution of a system comprised of gas and stars.

Fig. 14 shows the evolution of the 10 per cent, 50 per cent and 90 per cent Lagrangian radii for both the stellar and gaseous components separately as a function of time. We find the same qualitative evolution as in Hubber et al. (2013a): the stellar components decouple from each other and evolve in separate (and opposite) ways. The stellar Lagrangian radii all contract, most strongly close to the centre. The gaseous Lagrangian radii on the other hand expands at all radii, leading to a general expansion. The reason for this difference is whilst there is still energy exchange in interactions, the energy gained by gas from encounters with stars is converted into heat via shocks leading to a one-way expansion of the gas fed by energy from the stars. After beginning with identical profiles, the two components of several relaxation times eventually decouple.

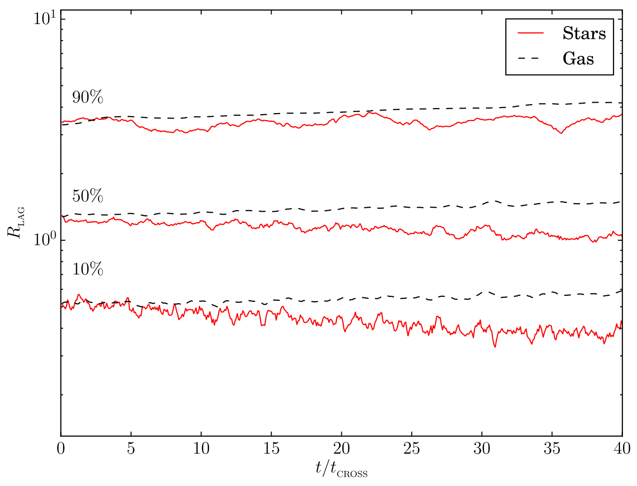


Figure 14. Evolution of the 10 per cent, 50 per cent and 90 per cent Lagrangian radii in a Plummer sphere containing (a) $N = 500$ equal mass stars and (b) $N = 500$ equal mass stars and 5000 SPH gas particles.

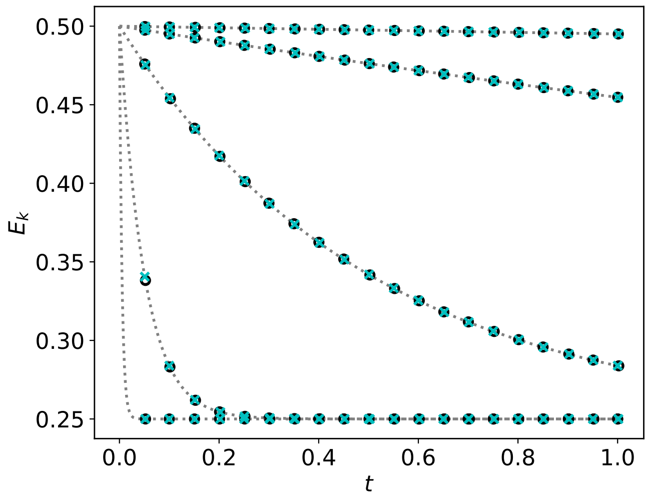


Figure 15. Evolution specific kinetic energy in the DUSTYBOX test using SPH (circles) and MFM (crosses), with feedback included. The evolution is shown for stopping times, t_s, of 0.01, 0.1, 1, 10 and 100.

6.9 Dust tests

The two-fluid dust methods included in GANDALF are essentially identical to the methods presented in Booth et al. (2015) and Lorén-Aguilar & Bate (2015). For this reason, we refer the reader to those papers and references therein for details on the performance of the method. Here, we include a few simple tests to verify the method.

6.9.1 DUSTYBOX

This test consists of two uniform gas and dust fluids which are set-up to initially have a velocity difference. We solve this problem in a 3D periodic box with size $1 \times 0.5 \times 0.5$ using $32 \times 16 \times 16$ particles arranged on a cubic lattice. We set the dust density, gas density, and sound speed to 1, using a fixed stopping time and taking the initial gas velocity to be at rest, while the dust is given a velocity of 1. In Fig. 15, we show the evolution of the kinetic energy for different stopping times computed with the full-scheme including feedback. Both methods produce accurate solutions for all stopping times. We ran this test using an adiabatic EOS to track the conservation of total

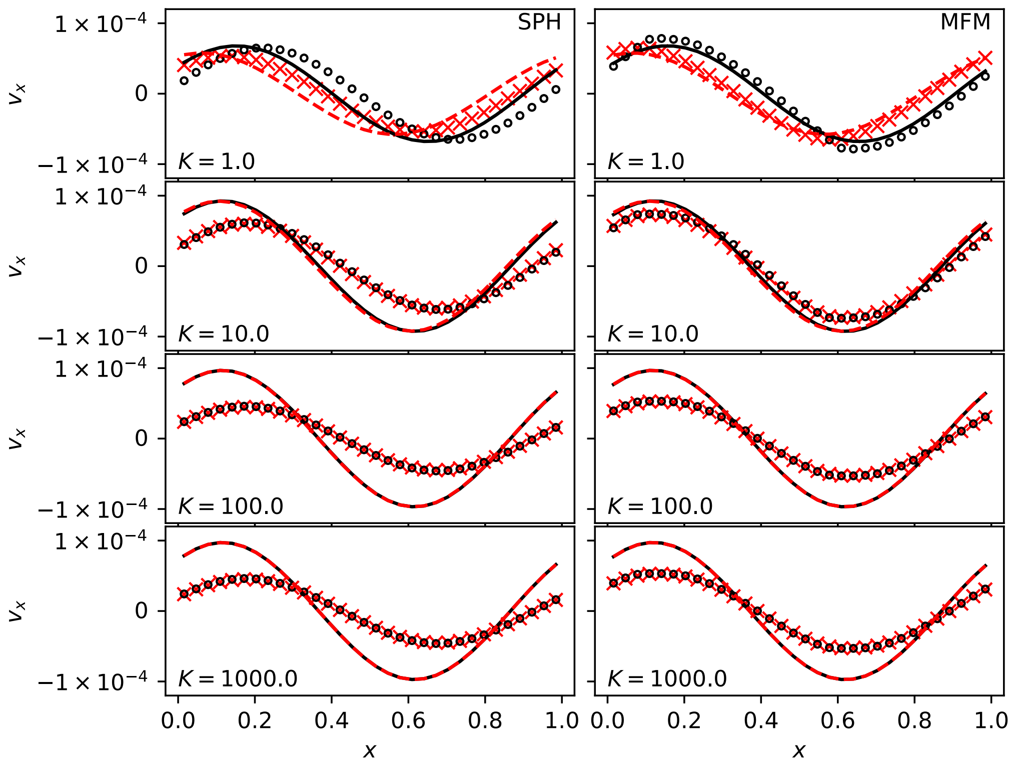


Figure 16. Results of the DUSTYWAVE test at $t = 3.0$. Lines show the analytical solution, while points show the particle values. Gas particles are shown by circles and the dust particles are shown by crosses.

energy: in the MFM method, the energy is conserved to machine precision, while SPH conserves energy up to time-integration errors ($\sim 10^{-9}$).

6.9.2 DUSTYWAVE

This is the DUSTYWAVE test of Laibe & Price (2011), which involves the evolution of two linear sound waves in a dusty fluid. We solve this problem in 1D using 32 particles per phase and dust-to-gas ratio of 0.1. The gas and dust are both given the same initial velocity, a soundwave with initial velocity of 10^{-4}. The gas is isothermal with sound speed, $c_s = 1$, and the simulations are evolved for three sound crossing times. The results for models in which the feedback is included are shown in Fig. 16 for both SPH and MFM, with different values of the drag coefficient, K, as defined by Laibe & Price (2011).

Both methods produce similar results even at this low resolution, but the MFM method reproduces the combined sound speed more closely, which is partly due to the smaller smoothing length ($\eta_{\mathrm{MFV}} = 1$, $\eta_{\mathrm{SPH}} = 1.2$). Both methods exhibit the well known over dissipation of the waves when the stopping time is very small ($c_s t_s \ll h$, Laibe & Price 2012; Lorén-Aguilar & Bate 2015). Here, the MFM method shows marginally lower dissipation, which is again mostly due to higher effective resolution. When run with feedback turned off, both the SPH and MFM implementations show essentially no dissipation, which is expected as the gas velocity is not damped (see e.g. Booth et al. 2015).

6.9.3 Shocks in 2D

Here, we present the 2D shock problem including dust as set-up in Booth et al. (2015), except a dust-to-gas ratio of 0.1 is used. We show this test in both SPH and the MFM method using the test particle dust implementation, and the full two-fluid scheme with feedback in the MFM scheme.

This test is sensitive to level of noise in the gas velocity distribution, which can hide the underlying gas vorticity field and introduce noise into the density fields of both the gas and dust (Sijacki et al. 2012; Booth et al. 2015). Given the much better performance of the quintic spline kernel in the Gresho & Chan (1990) test, we also employ it here. In SPH, the Cullen & Dehnen (2010) switch and Price (2008) artificial conduction are used, while in the MFM, the Hopkins (2015) limiter is employed with the HLLC Riemann solver.

Fig. 17 shows the resulting density and vorticity distributions. The overall features of both SPH and the MFM agree well here, largely due to the improvement of the SPH results that comes from using the quintic kernel. However, the SPH density and vorticity fields are considerably more smoothed than the MFM results. SPH still shows a small level of noise in the dust density. This density noise is nearly absent in the MFM results, which show close agreement with grid-based methods (e.g. Sijacki et al. 2012; Booth et al. 2015).

The MFM simulation with feedback included shows very similar results, demonstrating that the dust particles are not introducing noise into the gas dynamics in this problem. The only significant difference between the test particle and full two-fluid results is that with feedback switched on the peak vorticity is reduced, which is likely due to the physical damping by the feedback.

6.10 Spreading ring

The spreading ring test is a standard test (Flebbe et al. 1994; Artymowicz & Lubow 1994; Murray 1996; Kley 1999) in accretion disc theory to measure the shearing viscosity (either numerical or

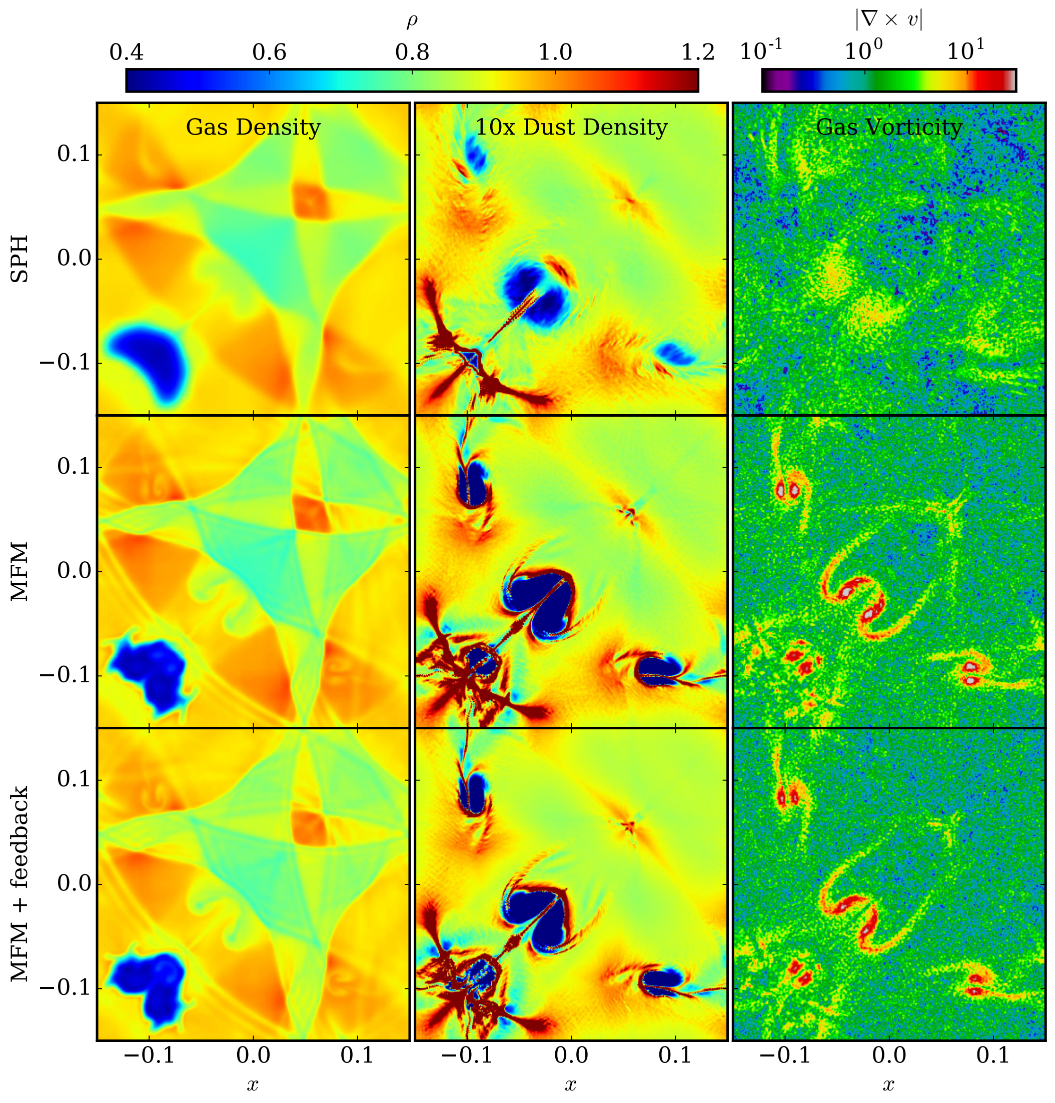


Figure 17. The 2D shock-tube test including dust for both the SPH and the MFM schemes. The top two rows show results computed with the test particle dust implementation, whereas the bottom row shows the results with feedback included.

physical) of a numerical method. The MFM scheme should have a much lower numerical viscosity than SPH and we wish to quantify this effect. We follow Murray (1996) to initialize a ring of particles with a Gaussian density profile $\Sigma \propto \exp\left(-(r - r_{\rm centre})/w\right)$, where w is the width of the ring and $r_{\rm centre}$ its position; the two parameters take the value of 0.033 and 1, respectively. We place the particles in a number of rings (equally spaced by a distance Δr), with a constant interparticle separation in the azimuthal coordinate $\Delta\phi$; the number of rings is set such that $r\Delta\phi \simeq \Delta r$. Therefore, to generate the desired density profile, we employ particles with different mass. To keep the test as clean as possible, we run it in two dimensions.

Previous works (e.g. Murray 1996) have switched off pressure forces to test only the effect of the artificial viscosity term in SPH. This is not possible to do with the meshless schemes, since they do not employ artificial viscosity. Therefore, we run the test with pressure forces. The downside is that pressure forces will contribute to the spreading of the ring. To counteract this problem, we modify the rotation curve of the particles so that the pressure forces are in equilibrium with the gravitational and centrifugal acceleration, preventing spreading due to pressure forces. In addition, we explore different temperatures of the disc (we use a isothermal EOS), sampling both a cold disc ($c_{\rm s} = 10^{-3}$), where the pressure is too little to cause spreading and a hot one ($c_{\rm s} = 0.05$), where it is potentially a significant contribution. Finally, differently from Murray (1996), the particles initially have a vanishing radial velocity, since we do not know a priori the magnitude of viscosity in the meshless schemes.

Fig. 18 shows the evolution of the density. The calculations have employed a resolution of 250 000 particles. To measure the value of the kinematic viscosity ν, we perform a least-squares fit. To define the squared residuals, we compare the average density in each ring of particles after a dimensionless time of 10 to the analytical solution (see e.g. equation 30 in Murray 1996). As common in differential equation theory, the analytical solution is a convolution between the kernel of the equation and the initial conditions; to the best of our knowledge, the convolution cannot be expressed in closed form and therefore we compute the integral numerically. Table 1 shows the results of the fit. The difference between SPH and the meshless is already clear by eye. The fact that the meshless has very little viscosity in the cold case is perhaps not surprising; since

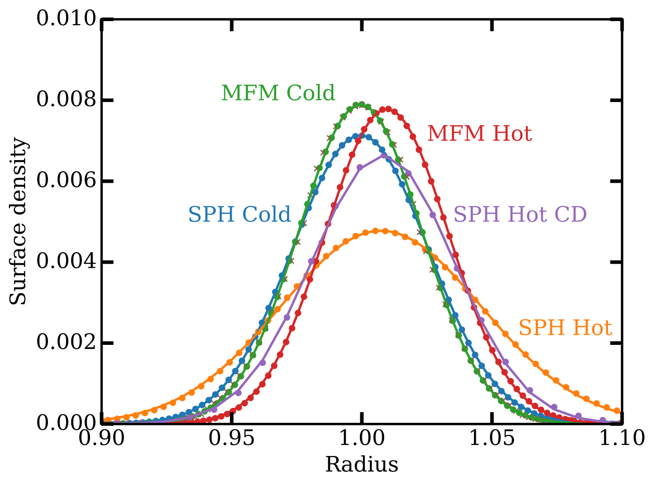


Figure 18. Evolution of the density of a spreading ring for the cases explained in the text at time $t = 10$. The initial conditions are plotted with the red crosses and they are nearly indistinguishable from the MFM cold case. We show the best fit with the numerical solution with a solid line and the results from the simulation (averaged for each ring of particles) with dots.

Table 1. Values of the kinematical viscosity ν derived from fitting the evolution of the spreading ring after $t = 10$. Notice that the values for the meshless should be considered as upper limits rather than measurements.

	Cold	Hot
SPH	2×10^{-6}	1.5×10^{-5}
MFM	7.7×10^{-9}	8.4×10^{-8}

the pressure forces are weak, the code in this case is effectively an N-body integrator. We can see that instead even in this case the artificial viscosity in SPH (here used without any switch), due to the shear, has a significant effect on the evolution of the ring, leading to a relatively high value of ν. In this case, because of the low sound speed, the quadratic β term dominates the artificial viscosity; setting $\beta = 0$ yields a ν of 6×10^{-7}, a factor of 3 smaller but still significantly higher than the meshless. Given that β dominates, an artificial viscosity switch would not change the resulting shear viscosity as the switches only operate on α. The value obtained by our implementation is consistent with the shear viscosity expected from SPH in an accretion disc. According to Artymowicz & Lubow (1994), in 2D the shear viscosity expected is $\nu = \frac{1}{8}\alpha_{\rm SPH} c_s h$. Substituting the value of $\alpha_{\rm SPH} = 1$, $c_s = 10^{-3}$ and $h = 2.7 \times 10^{-3}$, we obtain a value of 3.3×10^{-7}, which is within a factor of 2 from what we measure.

Additionally, we have used this test to verify the physical viscosity implementation in the meshless. Including a fixed shear viscosity, $\nu = 2 \times 10^{-6}$, we find that the spreading is consistent to within 5 per cent. This confirms that physical viscosity implementation is working as intended, and that the spreading ring test is good measure of the effective viscosity.

In the hot case, the meshless still performs very well; even in this case the ring remains almost indistinguishable from the initial one.[4] Notice that the value we report for the meshless is effectively an upper limit rather than a measurement; our numerical solution deteriorates for lower value of ν. For SPH in this case, we get a value of 1.5×10^{-5}. As in the previous case, this compares well to the value expected from the equations in Artymowicz & Lubow (1994) of 1.7×10^{-5}. In this case, the dominant term in the SPH artificial viscosity is the linear α term; setting $\beta = 0$ leads only to a 10 per cent reduction of ν. For this reason, it is worth investigating whether a modern viscosity switch can help reducing the numerical viscosity. We have run this test with both the Morris & Monaghan (1997) switch and the Cullen & Dehnen (2010) one. We find that for this particular test, they perform very similarly, with a small advantage for the latter; they yield a kinematic viscosity of 4×10^{-6} and of 3×10^{-6}, respectively. This is an improvement of a factor of 4–5, clearly visible in the figure (we plot only the Cullen & Dehnen 2010 case for simplicity). We note that this comes though at the cost of increased noise in the particle distribution; when running with either of the two switches, the particles very quickly lose the initial ring structure and rearrange in a more continuous (but noisier) structure. Even when using a viscosity switch in SPH, we conclude that the meshless has a significantly lower numerical viscosity than SPH.

6.11 Disc–planet interaction

Having established in the previous section in an idealized test that the meshless scheme has a lower numerical viscosity than SPH, we now wish to assess how the scheme performs in a more realistic simulation. For this goal, we have decided to run a simulation of a proto-planetary disc with a planet embedded; the set-up is loosely based on de Val-Borro et al. (2006). We have run the simulation both with SPH and the meshless in 3D employing a resolution of 500k particles. Random placement of particles is used to create the initial conditions. The initial surface density scales with radius as $\Sigma \propto r^{-1}$, extending from a radius of 0.4 to a radius of 2.5, while the sound speed scales as $c_s \propto r^{-0.5}$ and the aspect ratio of the disc at the inner boundary is 0.05. We insert a planet with a mass ratio of 10^{-3} with respect to the star (i.e. a Jupiter mass for a solar mass star) in a circular orbit with a semi-major axis of 1 and evolve the simulation for 40 orbits. While in SPH, we consider only artificial viscosity, in the meshless we add a physical viscosity with $\nu = 2 \times 10^{-5}$. Without physical viscosity, a vortex develops outside the orbit of the planet, due to the Rossby Wave Instability arising at the edge of the planetary gap (e.g. Lovelace et al. 1999; de Val-Borro et al. 2007). In SPH instead the much higher numerical viscosity suppresses vortex formation.

Fig. 19 shows the surface density of the disc after 40 orbits. It can be seen how in SPH the disc inside the orbit of the planet has a significantly lower mass compared to the meshless case, since the numerical viscosity caused a much higher accretion rate on to the star. Quantitatively, the calculation run with SPH is left with 300 k particles at this time, while the one with the MFV still has 380 k particles. The depletion of gas close to the star partially masks the opening of a gap by the planet in the SPH case, which is instead clearly visible in the meshless. In addition, due to the slightly higher effective resolution of the meshless (observed already in the shock tubes, see Section 6.2), the spiral arms created by the planet are much better defined in the meshless case.

In Fig. 20, we show the evolution of a disc containing a planet of a lower mass (10^{-4}), a set-up similar to Dipierro et al. (2016). We now use a shallower surface density $\Sigma \propto r^{-0.1}$ and a sound-speed scaling

[4] We have checked in this case that removing the contribution of the pressure forces to the rotation curve leads to a much bigger spread of the ring. Note that in the hot case, the centre of the ring moves slightly further out, but we ignore this effect in the analysis, since it affects both SPH and the meshless.

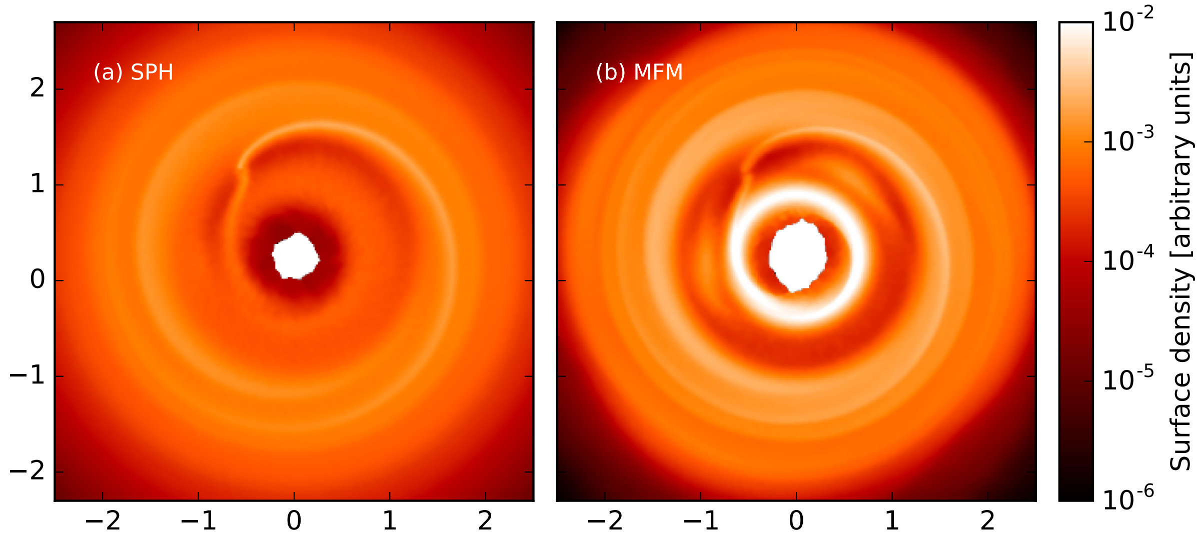


Figure 19. A proto-planetary disc with a Jupiter mass planet embedded after 40 orbits. Left-hand panel: SPH. Right-hand panel: MFM method.

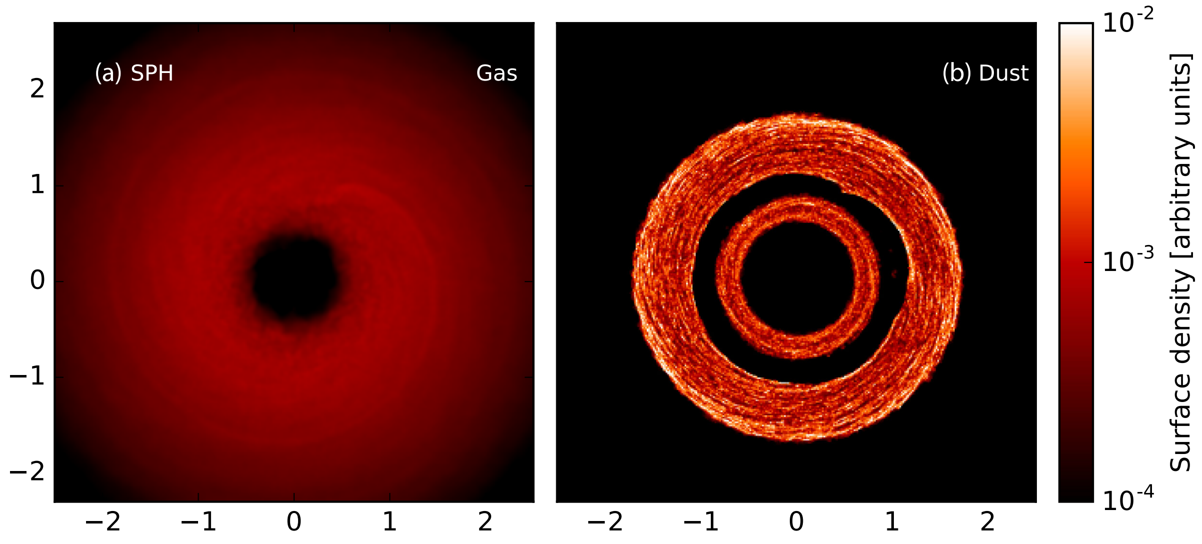


Figure 20. A proto-planetary disc with a planet with a mass of 10^{-4} with respect to the star. We show results for the gas and the dust distributions. The planet opens up a gap in the dust but not in the gas.

as $c_s \propto r^{-0.35}$, with an aspect ratio at the inner boundary of 0.075. To reduce the numerical viscosity, we set $\alpha_{\rm SPH} = 0.1$. We run the simulation both with gas and dust to test our dust implementation. We use 300k particles for the dust, which evolves as test particles. The Stokes number of the dust is 10. We confirm the results of Dipierro et al. (2016) that such a planet open up a gap in the dust, but not in the gas.

6.12 Boss–Bodenheimer test

The Boss–Bodenheimer test (Boss & Bodenheimer 1979) is a standard test of self-gravitating Astrophysical codes that simulates the collapse and fragmentation of a rotating cloud. Originally this test was performed with an isothermal EOS. However, it has also been performed with a barotropic EOS to mimic the optically thick adiabatic collapse phase during Star Formation. It provides a simple test case of combined hydrodynamics with self-gravity in Star Formation and subsequent sink particle formation and evolution.

The initial conditions are set-up similar to that described in Hubber et al. (2011). A spherical cloud of total mass $1\,{\rm M}_\odot$, radius 0.01 pc is created with a density profile

$$\rho = \rho_0\,\left[1 + A\,\sin\left(m\phi\right)\right] \qquad (105)$$

where $\rho_0 = 1.44\times10^{-17}\,{\rm g\,cm^{-3}}$, $A = 0.5$ is the perturbation amplitude, $m = 2$ is the order of the azimuthal perturbation and ϕ is the azimuthal angle about the z-axis. We generate a hexagonal closed-packed array and then cut-out a uniform-density sphere containing the desired number of particles. The total mass and radius of the sphere is scaled to $1\,{\rm M}_\odot$ and 0.01 pc, respectively. We finally alter the azimuthal positions of the particles to reproduce the required density field. The barotropic EOS used in this test gives the temperature as a function of density:

$$T(\rho) = T_0\left\{1 + \left(\frac{\rho}{\rho_{\rm AD}}\right)^{\gamma-1}\right\}, \qquad (106)$$

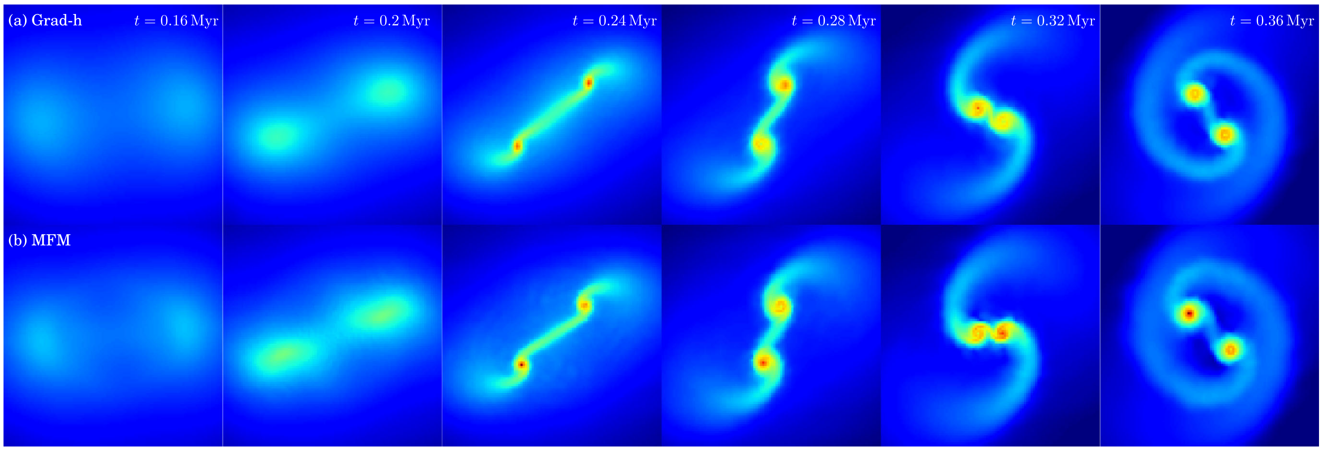


Figure 21. Time evolution of the column density profile (low density: blue and high density: red) for the Boss–Bodenheimer test using (a) grad-h SPH (upper row) and (b) MFM (bottom row). For both methods, the clouds collapse to for a bar-like structure with two denser condensations at either end which collapse to form sink particles. The two sinks form an accreting binary system with an extended circumbinary disc where mass continually infalls on to the two stars leaving a wake of gas behind each star.

where $T_0 = 10$ K, $\rho_{\rm AD} = 10^{-14}$ g cm^{-3} and $\gamma = 5/3$. The gas pressure is given by $P(\rho) = k_{\rm B}\, T(\rho)\rho/(\mu\, m_{\rm H})$ where $k_{\rm B}$ is the Boltzmann constant, $m_{\rm H}$ is the mass of hydrogen and the mean-gas-particle mass, $\mu = 2.35$.

We simulate the evolution until a time of $t_{\rm end} = 0.04$ Myr, by which time the cloud should fragments into two stars (or perhaps more) and the binary should have performed several orbits. The simulations were performed with both SPH and MFM using 32 000 particles.

6.12.1 *Time evolution*

In Fig. 21, we show the time evolution of the Boss–Bodenheimer test for both the SPH (top row) and MFM (bottom row) schemes. The large-scale evolution is the same for both cases as expected with both simulations forming a bar with two density enhancements at either end which gravitationally collapse to form two protostars (i.e. sink particles). The density enhancements are surrounded by disc-like envelope.

Three noticeable differences between the two simulations are apparent. (i) The evolution of the SPH simulation is slightly slower than the MFM scheme (i.e. it seems to lag slightly behind the MFM scheme) and takes slightly longer for the bar to reach the higher densities where it forms two objects at each end. (ii) Once intermediate densities have been reached and the two ends of the bar have reached some state of centrifugal support, the SPH simulations evolves towards higher densities much more quickly than the MFM simulation. In fact, the MFM scheme can never reach the sink density if it is too large compared to the adiabatic density. The main driver of this difference is likely to be the artificial viscosity in the SPH simulations. The artificial viscosity can efficiently (and artificially) transport angular momentum away from the disc-like object allowing it to collapse to higher densities quicker and hence form sinks rapidly. As demonstrated in Section 6.10, the MFM scheme instead has a much lower effective numerical viscosity, leading to less artificial angular momentum transport. In this simulation, we lower the sink density enough to allow comparable sink formation times and to allow a meaningful comparison with other features in the simulation. However this difference highlights that, even though SPH does not artificially cause fragmentation of already unstable regions, other numerical issues can lead to large differences in simulations between SPH and less dissipative methods. This is of particular importance when modelling discs, due to the high shear viscosity of SPH.

Recently Deng, Mayer & Meru (2017) made comparisons between the SPH and MFM by looking at the viscosity-driven angular momentum transport in rotating cores such as the Boss–Bodenheimer test. We confirm that we obtain similar results to Deng et al. (2017) in that SPH tends to lead to more rapid angular momentum transport than MFM, particularly near the edge of the cloud.

7 PERFORMANCE AND SCALING

7.1 Gravity tree scaling

Gravity trees used in particle codes typically scale as $\mathcal{O}(N \log N)$, i.e. N particles each requiring an average of $\mathcal{O}(\log N)$ computations. This is mainly because each particle must walk the tree individually, and then compute all contributions to the force from near (i.e. smoothed) neighbours, distant (i.e. non-smoothed) neighbours and distant cells using the centre-of-mass approximation. Gafton & Rosswog (2011) claimed that, if we walk the tree for groups of particles rather than one at a time, we can compute the contributions from the far cells more efficiently using a multipole expansion around the cell centre and instead approach $\mathcal{O}(N)$ scaling. As we showed in Section 6.5, we do not find a speed benefit in our implementation using the Taylor expansion around the cell centre, implying that our implementation has a different balance of the time spent computing the interaction with near or far particles. Therefore, it is likely that our tree will scale in a different way with the number of particles compared to Gafton & Rosswog (2011).

Fig. 22 shows the performance of GANDALF using direct-sum gravity and the tree. We set-up a uniform sphere of particles with different numbers of particles and compute the time needed to compute the gravitational acceleration. We plot this CPU time as a function of the particle number. The $\mathcal{O}(N^2)$ scaling of the direct-sum gravity is evident. Instead, it can be seen that, as hypothesized, our implementation of the tree scales as $\mathcal{O}(N \log N)$, and not as $\mathcal{O}(N)$. We

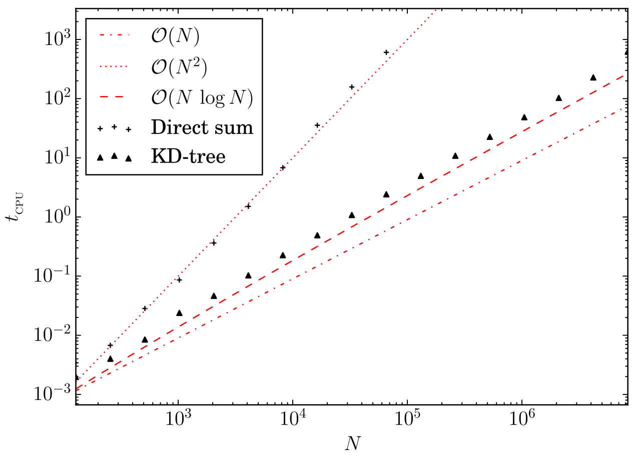


Figure 22. Performance and scaling for computing the gravitational acceleration of all particles using the KD tree in GANDALF as a function of particle number. For comparison, we plot lines showing the $\mathcal{O}(N)$ claimed by Gafton & Rosswog (2011) (red dot–dashed), $\mathcal{O}(N^2)$ expected for direct sum (red dotted) and $\mathcal{O}(N \log N)$ expected for tree gravity (red dashed).

note though that the difference between the two scalings is very small; over the almost 5 orders of magnitude spanned by the plot, the difference in wall clock time is a linear factor of 2–3. It is interesting to note also that Gafton & Rosswog (2011) comment that their scaling is not perfectly $\mathcal{O}(N)$, with an extra factor very similar in value to ours. This means that in practical terms the difference in scaling between our implementation and the one presented by Gafton & Rosswog (2011) is almost negligible.

7.2 OPENMP parallel scaling

GANDALF is parallelized using both OPENMP and MPI to allow the code to be used on much larger problem sizes than are achievable on single-core machines. Here, we investigate the strong scaling of the OPENMP parallelization and experiment with the number of particles at the leaf level of the tree to find the most optimal performance. As discussed in detail by Gafton & Rosswog (2011), the performance of the KD tree can be very sensitive to the chosen value of $N_{\rm LEAF}$, the (maximum) number of particles contained in each leaf cell of the tree. Small values of $N_{\rm LEAF}$ result in more tree walks being required (since there are fewer leaf cells in the tree), whereas large values of $N_{\rm LEAF}$ can result in much larger neighbour lists being generated for each leaf cell. Gafton & Rosswog (2011) empirically determined that the most optimal value of the average number of particles per leaf cell for their tree implementation was $\bar{N}_{\rm LEAF} \sim 12$.

We use the Boss–Bodenheimer test as a benchmark to test the parallel performance, since it is relatively simple to set-up, has a well-known numerical solution and computes both hydrodynamical and gravitational forces, the two most expensive components of the code. We run this test with $\sim 10^6$ particles using $N_{\rm CORE} = 1, 2, 4, 8, 16$ and 32 parallel cores in a shared-memory machine (parallelized with OPENMP) using various values of $N_{\rm LEAF}$ (1, 4, 8, 16 and 32) for 16 steps before terminating the simulation and measuring the time spent in the Main Loop (i.e. ignoring any set-up procedures). We also run with global time-steps, i.e. one time-step level, in order to demonstrate the best-case scaling for the various parameters. In Table 2, we show the total CPU wallclock time, $t(N_{\rm CORE})$ for each combination of $N_{\rm CORE}$ and $N_{\rm LEAF}$ and the scaling, $S(N_{\rm CORE}) \equiv t(1)/t(N_{\rm CORE})$.

We notice some important results from our scaling tests:

(i) For almost all values of $N_{\rm CORE}$, there is a broad minimum in the total CPU wallclock time for the simulation, at $N_{\rm LEAF} = 8$. This represents our most optimal value and default choice for $N_{\rm LEAF}$ in GANDALF.

(ii) The scaling of GANDALF formally increases with increasing values of $N_{\rm LEAF}$ for all values of $N_{\rm CORE}$ (although we note some fluctuations in the timing routines). Although this suggests using as high a value of $N_{\rm LEAF}$ as possible, the raw CPU times are a minimum for $N_{\rm LEAF} = 8$ which should be the most important factor. Although not shown in Table 2, for even larger values of $N_{\rm LEAF}$, achieving good load balancing becomes problematic and the scaling once again drops away.

7.3 Hybrid parallel scaling

As described in Section 5.2, GANDALF is parallelized both via OPENMP and MPI. The left-hand panel of Fig. 23 shows the strong scaling of GANDALF in pure OPENMP mode and in hybrid MPI–OPENMP mode. We tested the code on the Darwin supercomputer, hosted at the University of Cambridge, using version 12 of the Intel compiler. All the tests have been run for the Boss–Bodenheimer test as in Section 7.2. Compared to the previous section, we employ here a resolution of $\sim 4 \times 10^6$ particles, since we test the code up to 128 processors. Up to 8 threads, the speed-up is almost ideal (7.3), and still relatively good with 16 threads (12.7). We note that in both cases most of the time is spent computing the forces (both hydro forces and gravitational forces), with a very good scaling of 14.24 with 16 threads. The bottleneck to the scaling is mostly in the tree building routine and in other serial parts of the code.

With hybrid MPI–OPENMP parallelization, we experiment using different numbers of OPENMP threads. In general, we find that the best performance is achieved by using as many OPENMP threads as possible inside a given node (16 physical cores were available on the supercomputer we used for testing), and using MPI to communicate among the nodes. We interpret this result as a consequence of the fact that the MPI version of the code needs to do more work: pruned trees and ghost particles need to be created and sent to the other processors. This extra work adds to the overhead and limits the parallel scaling. In practice, we find that this particular test problem does not scale well using more than 64 processors, with only minimal improvements on 128 processors.

The real benefit of MPI however is to run simulation at higher resolution than what would be possible otherwise. For this reason, we also conduct tests of the weak scaling of GANDALF (right-hand panel of Fig. 23). The test has been run with a resolution of 2^{18} ($\sim$250 k) particles per processor. When defining a parallel efficiency, we have taken into account the extra $\log N$ factor demonstrated in Section 7.1. We use OPENMP only up to 16 cores, and switch to hybrid MPI–OPENMP mode using more processors. Based on the previous findings, we employ here 16 OPENMP threads, using MPI only to communicate between the nodes. We can see that the code exhibits very good weak scaling: even with 128 processors, the parallel efficiency is around 70 per cent.

8 DISCUSSION, FUTURE DEVELOPMENT AND CONCLUSIONS

In this paper, we have presented the new hydrodynamical code GANDALF with details and tests of all implemented algorithms. The code contains the robust and well-tested SPH method, as well

Table 2. CPU wallclock times in seconds and parallel scaling for 16 steps of the Boss–Bodenheimer test with $\sim 10^6$ particles using different values of $N_{\rm LEAF}$ (=1, 4, 8, 16 and 32) using 1, 2, 4, 8, 16 and 32 cores in parallel with OPENMP. For almost all numbers of parallel cores, $N_{\rm LEAF} = 8$ gives the shortest CPU run times even though the formal parallel scaling for higher values of $N_{\rm LEAF}$ is better. We note the superlinear scaling of $N_{\rm LEAF} = 32$ with 2 cores is due to fluctuations in CPU performance and in the timing routines.

$N_{\rm LEAF}$	Serial	2 cores		4 cores		8 cores		16 cores		32 cores	
	time	Time	Scaling	Time	Scaling	Time	Scaling	Time	Scaling	Time	Scaling
1	1388.7	772.0	1.80	405.7	3.40	210.6	6.60	106	13.1	60.2	23.1
4	791.0	402.8	1.96	204.3	3.88	104.9	7.50	55.2	14.3	31.4	25.2
8	732.6	374.2	1.96	192.0	3.82	100.0	7.30	52.2	14.0	28.2	26.0
16	815.0	416.3	1.96	211.4	3.86	109.6	7.50	56.3	14.6	30.7	26.6
32	1066.1	523.4	2.04	271.2	3.93	137.6	7.75	71.2	15.0	37.7	28.2

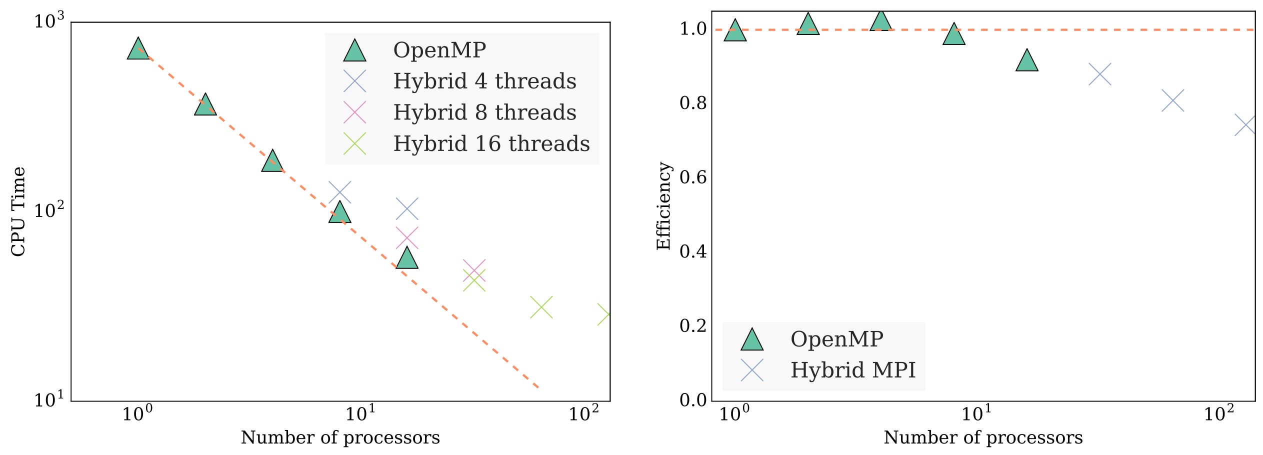


Figure 23. Left-hand panel: strong scaling of GANDALF for 16 steps of the Boss–Bodenheimer test with 4×10^6 particles. The speed-up is relative to the serial version of the code. When using hybrid parallelization, the different colours are for different number of OPENMP threads (as shown in the legend). In all cases, we find that the optimal strategy is to use as many OPENMP threads as possible. Right-hand panel: weak scaling of GANDALF for the same test using 2^{18} ($\sim$250 k) particles per processor. The efficiency has been normalized taking into account the $\log N$ scaling of the algorithms.

as the MFV numerical schemes presented by Gaburov & Nitadori (2011) and Hopkins (2015). In addition, GANDALF can handle N-body dynamics with higher order collisional integrators than what is commonly employed in SPH simulations and implements an energy conserving scheme for integrating the dynamics of stars and gas. Both hydrodynamical schemes can also handle dust dynamics, either in the test particle limit or keeping the back reaction of the dust on to gas into account. The object-oriented design of GANDALF makes the code flexible, easy to adapt with new physics modules and it is relatively easy to add other particle-based schemes.

We have presented an extensive suite of tests to demonstrate the correctness of our implementation, mostly recovering the results of Hopkins (2015) in terms of the benefits of the MFV schemes compared to SPH. In addition, we have conducted a more rigorous test to quantify the numerical viscosity of the method. In the spreading ring test, we have shown that the MFM scheme has a much lower numerical viscosity than SPH and is therefore better suited for accretion disc applications, where the numerical viscosity of SPH is typically too high to perform realistic simulations (unless a very high resolution is used). The same conclusion is reached also looking at the evolution of a proto-planetary disc containing a planet, where the inner part of the disc in SPH is rapidly accreted on to the star due to the high numerical viscosity.

The code is publicly available at this address under the GPLv2 license. The code is parallelized with OPENMP and MPI for running on modern supercomputers. In addition, we provide a PYTHON library to facilitate analysis of the results of the simulations and ease code use and development, since the results of a simulation can be inspected live while it is running.

We plan in the future to implement additional algorithms and physics modules in GANDALF. Examples of developments which are underway include algorithms for radiation transport and coupling with existing chemistry codes (e.g. Grassi et al. 2014). We encourage users of the code to contact us if there are specific algorithms they are interested in.

We hope that the numerical techniques implemented in GANDALF, its ease of use and modularity of design will help future research with this code.

ACKNOWLEDGEMENTS

This research was supported by the DFG cluster of excellence ‘Origin and Structure of the Universe’ including the Munich Institute for Astro- and Particle Physics (MIAPP), DFG Projects 841797-4, 841798-2 (DAH and GPR), the DISCSIM project, grant agreement 341137 funded by the European Research Council under ERC-2013-ADG (GPR and RAB). Some development of the code and simulations have been carried out on the computing facilities of the Computational centre for Particle and Astrophysics (C2PAP) and on the DiRAC Data Analytic system at the University of Cambridge, operated by the University of Cambridge High Performance Computing Service on behalf of the Science and Technology Facilities Council (STFC) DiRAC HPC Facility (www.dirac.ac.uk); the equipment was funded by BIS National E-infrastructure capital grant (ST/K001590/1), STFC capital grants ST/H008861/1 and ST/H00887X/1, and STFC DiRAC Operations grant ST/K00333X/1. We would like to thank the following people for helpful discussions or for contributing code to the public version, including Alexander Arth (discussions on IC generation), Scott Balfour (ionizing radiation algorithms), Seamus Clarke (sink particle algorithm refinements and various bug fixes), James Dale (ionizing radiation algorithms) Franta Dinnbier (periodic gravity), Stefan Heigl (assisting implementing the MFV schemes), Oliver Lomax (assisting implementing the KD tree), Judith Ngoumou (assisting implementing the MFV schemes), Margarita Petkova (parallelization and C2PAP support), Paul Rohde (stellar feedback routines), Steffi Walch (supernova feedback routines) and Anthony Whitworth (discussions on trees and radiation algorithms). We also thank the anonymous referee for helpful and detailed comments which have improved the clarity and readability of this paper.

REFERENCES

Aarseth S. J., 2003, Gravitational N-Body Simulations. Cambridge Univ. Press, Cambridge
Aarseth S. J., Henon M., Wielen R., 1974, A&A, 37, 183
Artymowicz P., Lubow S. H., 1994, ApJ, 421, 651
Barnes J., Hut P., 1986, Nature, 324, 446
Bate M. R., Bonnell I. A., Price N. M., 1995, MNRAS, 277, 362
Batten P., Clarke N., Lambert C., Causon D. M., 1996, SIAM J. Sci. Comput., 18, 1553
Binney J., Tremaine S., 2008, Galactic Dynamics: Second Edition. Princeton University Press, Princeton, NJ
Booth R. A., Sijacki D., Clarke C. J., 2015, MNRAS, 452, 3932
Boss A. P., Bodenheimer P., 1979, ApJ, 234, 289
Cullen L., Dehnen W., 2010, MNRAS, 408, 669
de Val-Borro M. et al., 2006, MNRAS, 370, 529
de Val-Borro M., Artymowicz P., D'Angelo G., Peplinski A., 2007, A&A, 471, 1043
Deng H., Mayer L., Meru F., 2017, ApJ, 847, 43
Dipierro G., Laibe G., Price D. J., Lodato G., 2016, MNRAS, 459, L1
Flebbe O., Muenzel S., Herold H., Riffert H., Ruder H., 1994, ApJ, 431, 754
Gaburov E., Nitadori K., 2011, MNRAS, 414, 129
Gafton E., Rosswog S., 2011, MNRAS, 418, 770
Gamma E., Helm R., Johnson R., Vlissides J., 1995, Addison-Wesley Professional Computing Series. Addison-Wesley, Reading, MA
Gingold R. A., Monaghan J. J., 1977, MNRAS, 181, 375
Grassi T., Bovino S., Schleicher D. R. G., Prieto J., Seifried D., Simoncini E., Gianturco F. A., 2014, MNRAS, 439, 2386
Gresho P. M., Chan S. T., 1990, Int. J. Numer. Methods Fluids, 11, 621
Hernquist L., Bouchet F. R., Suto Y., 1991, ApJS, 75, 231
Heß S., Springel V., 2010, MNRAS, 406, 2289
Hopkins P. F., 2013, MNRAS, 428, 2840
Hopkins P. F., 2015, MNRAS, 450, 53
Hopkins P. F., 2017, MNRAS, 466, 3387
Hubber D. A., Goodwin S. P., Whitworth A. P., 2006, A&A, 450, 881
Hubber D. A., Batty C. P., McLeod A., Whitworth A. P., 2011, A&A, 529, A27
Hubber D. A., Allison R. J., Smith R., Goodwin S. P., 2013a, MNRAS, 430, 1599
Hubber D., Rosotti G., 2016, Astrophysics Source Code Library, record ascl:1602.015
Hubber D. A., Walch S., Whitworth A. P., 2013b, MNRAS, 430, 3261
Hut P., Makino J., McMillan S., 1995, ApJ, 443, L93
Inutsuka S.-I., 2002, J. Comput. Phys., 179, 238
Kley W., 1999, MNRAS, 303, 696
Laibe G., Price D. J., 2011, MNRAS, 418, 1491
Laibe G., Price D. J., 2012, MNRAS, 420, 2345
Lanson N., Vila J.-P., 2008, SIAM J. Numer. Anal., 46, 1912
Lorén-Aguilar P., Bate M. R., 2015, MNRAS, 454, 4114
Lovelace R. V. E., Li H., Colgate S. A., Nelson A. F., 1999, ApJ, 513, 805
Lucy L. B., 1977, AJ, 82, 1013
Makino J., Aarseth S. J., 1992, PASJ, 44, 141
Mignone A., 2007, J. Comput. Phys., 225, 1427
Monaghan J. J., 1992, ARA&A, 30, 543
Monaghan J. J., 1997, J. Comput. Phys., 136, 298
Monaghan J. J., Lattanzio J. C., 1985, A&A, 149, 135
Morris J. P., 1996, PhD thesis, Monash University
Morris J. P., Monaghan J. J., 1997, J. Comput. Phys., 136, 41
Muñoz D. J., Springel V., Marcus R., Vogelsberger M., Hernquist L., 2013, MNRAS, 428, 254
Murray J. R., 1996, MNRAS, 279, 402
Portegies Zwart S. F., McMillan S. L. W., Hut P., Makino J., 2001, MNRAS, 321, 199
Price D. J., 2007, Publ. Astron. Soc. Aust., 24, 159
Price D. J., 2008, J. Comput. Phys., 227, 10040
Price D. J., 2012, J. Comput. Phys., 231, 759
Price D. J., Laibe G., 2015, MNRAS, 451, 813
Price D. J., Monaghan J. J., 2007, MNRAS, 374, 1347
Price D. J. et al., 2017, PASA, preprint (arXiv:1702.03930)
Rosswog S., 2015, MNRAS, 448, 3628
Saitoh T. R., Makino J., 2009, ApJ, 697, L99
Saitoh T. R., Makino J., 2013, ApJ, 768, 44
Sijacki D., Vogelsberger M., Kereš D., Springel V., Hernquist L., 2012, MNRAS, 424, 2999
Springel V., 2005, MNRAS, 364, 1105
Springel V., 2010, MNRAS, 401, 791
Springel V., Hernquist L., 2002, MNRAS, 333, 649
Stone J. M., Gardiner T. A., Teuben P., Hawley J. F., Simon J. B., 2008, ApJS, 178, 137
Toro E. F., 1997, Riemann Solvers and Numerical Methods for Fluid Dynamics: A Practical Introduction. Springer-Verlag, Berlin, New York
Toro E. F., Spruce M., Speares W., 1994, Shock Waves, 4, 25
van Leer B., 1979, J. Comput. Phys., 32, 101
Wadsley J. W., Stadel J., Quinn T., 2004, New Astron., 9, 137
Wadsley J. W., Veeravalli G., Couchman H. M. P., 2008, MNRAS, 387, 427
Wetzstein M., Nelson A. F., Naab T., Burkert A., 2009, ApJS, 184, 298
Whitworth A. P., Bhattal A. S., Turner J. A., Watkins S. J., 1995, A&A, 301, 929
Wünsch R., Walch S., Whitworth A. P., Dinnbier F., 2017, MNRAS, preprint (arXiv:1708.06142)

This paper has been typeset from a TEX/LATEX file prepared by the author.