The Astrophysical Journal Supplement Series, 200:6 (14pp), 2012 May doi:10.1088/0067-0049/200/1/6

PHURBAS: AN ADAPTIVE, LAGRANGIAN, MESHLESS, MAGNETOHYDRODYNAMICS CODE. I. ALGORITHM

Jason L. Maron[1], Colin P. McNally[2], and Mordecai-Mark Mac Low[2]
Department of Astrophysics, American Museum of Natural History, New York, NY, USA; jmaron@amnh.org, cmcnally@amnh.org, mordecai@amnh.org
Received 2011 October 3; accepted 2012 March 28; published 2012 May 3

ABSTRACT

We present an algorithm for simulating the equations of ideal magnetohydrodynamics and other systems of differential equations on an unstructured set of points represented by sample particles. Local, third-order, least-squares, polynomial interpolations (Moving Least Squares interpolations) are calculated from the field values of neighboring particles to obtain field values and spatial derivatives at the particle position. Field values and particle positions are advanced in time with a second-order predictor-corrector scheme. The particles move with the fluid, so the time step is not limited by the Eulerian Courant–Friedrichs–Lewy condition. Full spatial adaptivity is implemented to ensure the particles fill the computational volume, which gives the algorithm substantial flexibility and power. A target resolution is specified for each point in space, with particles being added and deleted as needed to meet this target. Particle addition and deletion is based on a local void and clump detection algorithm. Dynamic artificial viscosity fields provide stability to the integration. The resulting algorithm provides a robust solution for modeling flows that require Lagrangian or adaptive discretizations to resolve. This paper derives and documents the Phurbas algorithm as implemented in Phurbas version 1.1. A following paper presents the implementation and test problem results.

Key words: hydrodynamics – magnetohydrodynamics (MHD) – methods: numerical

Online-only material: color figure

1. INTRODUCTION

1.1. Context

Our understanding of many astrophysical systems relies on the simulation of magnetized plasmas. As a result, much effort has been made to develop tools to efficiently perform high-fidelity simulations of them. Some of these tools have found broad application in other fields of physics and engineering as well.

Many early methods for solving the equations of magnetohydrodynamics (MHD) were based on fixed grids. Discretizing the equations of hydrodynamics or MHD on a fixed grid leads to an Eulerian method, or a method written in terms of Eulerian derivatives. Popular publicly available codes with methods based on point values such as the Pencil Code[3] (Brandenburg & Dobler 2002) and finite volumes, such as ZEUS (Hayes et al. 2006), FLASH (Fryxell et al. 2000), or Athena (Stone et al. 2008), use such methods. Eulerian methods share the common property that the discretized form of the governing equations is not Galilean invariant. Though they still converge to the correct solution, this does lead to two limitations at any finite resolution. First, the explicit integration time step constraint from the Courant–Friedrichs–Lewy (CFL) condition depends on both the signal speed and the flow velocity relative to the grid, not just the signal speed. Second, the numerical diffusion of the scheme, usually highly nonlinear, also depends on the flow velocity relative to the grid.

A fixed grid approach thus has disadvantages particularly where there are high-velocity bulk flows, collapsing flows, or flows that generate localized fine structure. For the latter cases, adaptive mesh refinement (Berger & Oliger 1984) has been a successful approach. This method, while still Eulerian, uses refined meshes to allow the spatial and temporal resolution to vary. However, for problems with significant bulk flows, it is of no help, as the same problems of time step limitation and numerical diffusion apply as with uniform grids. A numerical viscosity dependent on the bulk flow can be significant, because the growth of instabilities from a marginally resolved mode in a method lacking Galilean invariance will depend on the bulk velocity of the flow across this grid. The effects of this can be seen, for example, in Chiang (2008) and Johansen et al. (2009). To circumvent the time step limit in disks treated with cylindrical or spherical coordinates or in a shearing-sheet approximation where the bulk flow is largely Keplerian and aligned with the grid, it is possible to add a separate transport step to the method (Masset 2000). While this extra transport step improves the problems with numerical diffusion, it does not fully cure the issue (Johansen et al. 2009; Stone & Gardiner 2010).

To escape these limits, it is necessary to move to a method formulated in terms of Lagrangian (also known as covariant, co-moving, convective, advective, substantive, or material) derivatives.[4] In contrast to Eulerian formulations, Lagrangian methods have three advantages. Foremost, for problems with significant bulk flows, a purely Lagrangian formulation has a significantly less stringent time step constraint from the signal speed (the CFL condition). This is because the time step in an Eulerian method depends on the maximum of the signal speed and the flow speed, whereas in a purely Lagrangian method the time step depends only on the local signal speed. Relaxing this constraint becomes particularly important in the case of an extended disk

[1] Current address: North Carolina Museum of Natural Sciences, Raleigh, NC, USA.
[2] Also at Department of Astronomy, Columbia University, New York, NY, USA.
[3] See http://www.nordita.org/software/pencil-code/
[4] Methods that solve Eulerian problems in a local frame chosen to be comoving with the fluid in a locally average sense also share in some of the advantages of this formulation.

with supersonic differential rotation, where in an Eulerian formulation the quickly orbiting inner regions constrain the time step severely. A second advantage of a Lagrangian method lies in the Galilean invariance of the inevitable effects of numerical diffusion. Though Galilean invariance itself can formally be achieved in an Eulerian method (Springel 2010; Robertson et al. 2010), a Lagrangian formulation can reduce the diffusivity further because it uses fewer time steps. Finally, Lagrangian methods naturally focus resolution into regions of fluid concentration, which are often, though not always, the regions of greatest interest. (We note that the adaptive, Lagrangian method we describe here can also focus resolution to other, arbitrary regions of interest.)

It is possible to write a comoving discretization in two ways. First, one can discretize the governing equations directly in terms of Lagrangian time derivatives. Second, one can discretize in terms of partial time derivatives around moving interfaces. Historically, the most popular approach has been the first, particularly when used to build a meshless method. Recently, the second has been used, with techniques based on a moving unstructured mesh with mesh reconnection.

One of the earliest and most popular meshless schemes is smoothed particle hydrodynamics (SPH; Lucy 1977; Gingold & Monaghan 1977). SPH quickly gained popularity as the advantages in numerical diffusion, local resolution scales, and local time step advantages were realized (Steinmetz & Mueller 1993). However, the basic SPH algorithm has many shortcomings. The foremost and most fundamental is the lack of discrete, zeroth-order consistency in the SPH representation of a function. SPH interpolation fails to reproduce even a constant function. The importance of this consistency property in general meshless schemes has been pointed out by Liu et al. (1995). This insight has been applied to analysis of SPH by Dilts (1999), Liu et al. (2003), Fries & Matthies (2004), and Quinlan et al. (2006), among others. They find that the lack of zeroth-order consistency can cause substantial gradient and value errors that do not converge with increased particle number alone. The inability of SPH to effectively model subsonic turbulence has been blamed on this lack of consistency by Bauer & Springel (2011), though the behavior in this regime depends strongly on and can be significantly improved by using a more modern formulation of the SPH artificial viscosity (Price 2012a).

Resolution in SPH is further limited by constant particle masses. Some attempts at adaptive particle masses have been made (Kitsionas & Whitworth 2002) but these suffer from difficulties in specifying a well-posed scheme. SPH in general handles differing particle masses poorly, as the pairwise interparticle interactions allow heavy particles to penetrate though the fluid in a nonphysical manner. Similarly, the spatial resolution in SPH is locally isotropic, even when the particle and mass distribution is anisotropic. Attempting to relax this constraint leads to the adaptive SPH scheme of Shapiro et al. (1996) and Owen et al. (1998).

A grid that is both Lagrangian and has a logically Cartesian structure is a simple choice, and a logically Cartesian moving (Lagrangian) mesh has also been used to attempt to minimize numerical diffusion (Norman et al. 1980; Fiedler & Mouschovias 1992). Gnedin (1995) and Pen (1998) used a moving, logically Cartesian mesh to provide adaptivity in collapsing flows. However, this approach falls victim to several limits. In many flows the cells eventually become long and thin, leading to large errors. Also, the grid cannot follow rotation or turbulent flows as it becomes tangled.

Unstructured, moving mesh methods with mesh reconnection have recently been introduced in astrophysics. The methods of Springel (2010), Pakmor et al. (2011), Duffell & MacFadyen (2011), and Gaburov et al. (2012) are finite volume methods based on Voronoi tessellations. The mesh is defined by the Voronoi tessellation of a set of points that move approximately with the mean motion of the fluid in the cell (though formally any motion can be chosen). These methods can be described as Lagrangian though they calculate intercell fluxes with Eulerian Riemann problems stated in a locally comoving frame. The connectivity of the mesh is dictated by the Voronoi neighbor relation. Fluxes between cells are calculated across the moving cell faces. Springel (2010) and Pakmor et al. (2011) describe a Galilean invariant method. The method of Duffell & MacFadyen (2011) is not fully Galilean invariant, but this is due to the formulation chosen for the slightly more complicated relativistic hydrodynamic equations. Both methods use an approximately comoving formulation in a significant sense.

This paper describes an adaptive, Lagrangian, meshless, collocation scheme for MHD or similar sets of equations based on a point (not finite volume or mass) discretization. In what follows, we refer to the discretization points as particles, following the historical usage. However, these discretization points do not in any sense represent identifiable masses or volumes of the fluid. They are simply moving points sampling continuous field variables.

In the next subsection we discuss prior work on related methods to solve the MHD equations. We then describe our algorithm, starting with an overview (Section 2) and then discussing specific numerical aspects, such as the modeling of the function and the time update (Section 3), adaptive addition and deletion of particles (Section 4), explicit time step limits (Section 5), and magnetic divergence correction (Section 6.2). Finally we draw these together with a summary of the algorithm (Section 7). In the next paper of this series (McNally et al. 2012, hereafter Paper II) we present implementation details and present the results of a suite of gas dynamical and MHD tests of the algorithm.

1.2. Prior MHD Methods

Several attempts have been made to design an SPH-type scheme for MHD. The most successful and recent work by Price & Monaghan (2004a, 2004b, 2005) and Price (2010) resulted in an SPH MHD based on a form of the MHD equations that is consistent with $\nabla \cdot \boldsymbol{B} \neq 0$ and a set of artificial dissipation terms. Rosswog & Price (2007) developed a variation based on representing the magnetic fields though Euler angles, which allows a guaranteed $\nabla \cdot \boldsymbol{B} = 0$ at the cost of disallowing tangled field geometries (Brandenburg 2010), severely limiting its applicability. Dolag & Stasyszyn (2009) implement an SPH MHD in GADGET-3, without any constraint on $\nabla \cdot \boldsymbol{B}$, but subtracting the numerical contribution of $\nabla \cdot \boldsymbol{B}$ to the momentum equation. We refer the reader to Price (2012b) for a further overview of the attempts to design an SPH MHD method.

Unfortunately, all these SPH MHD methods suffer from the fundamental drawback of SPH, that the SPH interpolant does not have a zeroth-order consistency property. This zeroth-order inconsistency means that for a disordered set of SPH particles, a constant function cannot be reproduced by the SPH representation of that function. As the SPH representation of even a constant function has significant positive and negative errors, it also has significantly non-zero derivatives. These errors make formulating an SPH MHD difficult. Modifications of SPH to solve or work around the zeroth-order consistency problem have been proposed. Børve et al. (2001, 2006) developed an extension to SPH using a remapping strategy to increase the accuracy of SPH estimates through regularizing the particle distribution, and applied it to MHD shocks. For hydrodynamics, Morris (1996) and Abel (2011) have proposed working around the effects of the zeroth-order consistency problem for pressure forces only, with an alternative derivation of the SPH pressure force. This comes at the price of sacrificing the local momentum conservation enjoyed by the classical formulation. This also only treats the problem of spurious pressure forces arising from the zeroth-order inconsistency, and does not lead to a consistent interpolation of the pressure field or other fields.

It is also possible to construct an SPH MHD scheme using a Godunov approach. Godunov SPH was originally proposed for hydrodynamics, using Riemann problems to solve for the particle interactions. Godunov SPH uses SPH interpolation for density (see Equations (6) and (21) of Inutsuka 2002, and Equation (29) of Iwasaki & Inutsuka 2011).[5] A Godunov SPH MHD implementation using Powell-type source terms and a tensile correction was implemented by Iwasaki & Inutsuka (2011). They point out that all SPH-based MHD schemes that avoid tensile instability do not exactly conserve momentum, energy, or both. Similarly, Gaburov & Nitadori (2011) constructed an SPH-like scheme (a weighted particle method) with a consistent second-order accurate formulation for derivatives, coupling this with a pairwise Riemann-solver based interaction between particles to yield an MHD scheme. A Galilean invariant form of the Dedner et al. (2002) hyperbolic–parabolic cleaning scheme was used to handle $\nabla\cdot\boldsymbol{B}$ errors.

However, these SPH-based methods again suffer from the zeroth-order inconsistency of the SPH interpolant, even though methods with a renormalized first derivative estimate have a consistent first derivative. This means that SPH interpolated fields (such as the density values) have significant noise. To reduce the amplitude of the noise it is necessary to increase the number of neighbors used in the kernel, which greatly increases the computational cost. This means that rigorous convergence studies, even in smooth flow, are not feasible with methods based on SPH-type estimates. In addition, SPH Riemann methods suffer a higher computational cost in comparison to moving unstructured mesh Godunov schemes, because of the requirement of a much higher number of Riemann problem solutions per particle.

Duffell & MacFadyen (2011) implemented an MHD scheme in their Voronoi tessellation method, using a Dedner-type hyperbolic divergence cleaning method, but found it difficult to manage $\nabla\cdot\boldsymbol{B}$ errors when the mesh topology changes. Pakmor et al. (2011) used a very similar approach, with apparently much greater success in managing $\nabla\cdot\boldsymbol{B}$ errors. Gaburov et al. (2012) add a source term to the induction equation to restore Galilean invariance if $\nabla\cdot\boldsymbol{B}\neq 0$ and claim this greatly improves stability.

The method we describe here was inspired by the Gradient Particle Method of Maron & Howes (2003), but removes the underlying instability present in that method (described in Appendix A). A method particularly similar to Maron & Howes (2003), but limited to hydrodynamics using a moving-least-squares fit was proposed by Dilts (1999, 2000). Numerous related methods have been described in the literature on meshfree or meshless methods. The most closely related method is the Finite Pointset Method (FPM) described by Kuhnert (1999, 2002), which is not to be confused with either the similarly named Finite Point Method of Oñate et al. (1996), or the equally similarly named Finite Particle Method of Liu et al. (2005).[6] FPM has limited adaptivity, is first order, and uses an upwinded formulation for hydrodynamics. Similar to the method we describe, it is meshless, Lagrangian, has particle addition and deletion, and uses moving-least-squares interpolation.

[5] An earlier usage of Riemann solvers coupled with SPH is given by Parshikov et al. (2000).

[6] The authors are of the opinion that enough numerical schemes have been named FPM, and as the names are getting confusing the practice should cease.

2. ALGORITHM

For specificity, we focus on using our method to solve the equations of MHD. These can be expressed using Lagrangian time derivatives D_t, as

$$D_t V_j = -\rho^{-1}\partial_j P + \rho^{-1}\varepsilon_{jab}\varepsilon_{acd}(\partial_c B_d)B_b + G_j, \tag{1}$$

$$D_t B_j = B_i \partial_i V_j - B_j \partial_i V_i, \tag{2}$$

$$D_t \sigma = -(\sigma + P)\partial_i V_i, \tag{3}$$

$$D_t \rho = -\rho \partial_i V_i, \tag{4}$$

where V is the velocity, B is the magnetic field, σ is the internal energy volume density, P is the pressure, ρ is the density, G_j is a vector component of a body force, and the Einstein summation rule is assumed. We note that Phurbas is relatively insensitive to the exact form of the equations solved and variables chosen. For example, energy variables other than the internal energy per volume could be used. In Appendix C we give the second time derivatives of these equations for use in the time update. These equations require the addition of an equation of state, such as a gamma-law $P = (\gamma - 1)\sigma$, though the equation of state is arbitrary.

The MHD Equations (1)–(4) are solved on an adaptive set of particles, each particle carrying values for the field variables ρ, $\boldsymbol{V}$, $\boldsymbol{B}$, and σ. Particles move in the frame of the fluid with the local fluid velocity $\boldsymbol{V}$. Field variables evolve in the frame of the particle, so the evolution equations are most naturally expressed using the Lagrangian form for the time derivatives in the MHD equations.

The equations of MHD as stated are ill-suited to the numerical scheme we will use. For the discretization used in Phurbas, we require a system of equations in which short wavelength perturbations decay. Appendix A demonstrates this for a model advection-diffusion equation. To ensure decay of such perturbations, we introduce artificial dissipation terms to the analytic form of the equations before discretizing. These modifications are in the form of a bulk viscosity, and mass and thermal diffusions. Formally, it is this modified version of the MHD equations from which Phurbas computes approximate numerical solutions. The MHD equations, reiterated with the addition of the stabilizing terms, and associated fields are

$$\begin{aligned} D_t V_j = &-\rho^{-1}\partial_j P + \rho^{-1}\varepsilon_{jab}\varepsilon_{acd}(\partial_c B_d)B_b \\ &+ G_j + \partial_j((\zeta_s + \zeta_l)\partial_i V_i), \end{aligned} \tag{5}$$

$$D_t B_j = B_i \partial_i V_j - B_j \partial_i V_i + \xi_j, \tag{6}$$

$$D_t \sigma = -(\sigma + P)\partial_i V_i + (\zeta_s + \zeta_l)(\partial_i V_i)^2 + H_\sigma \rho \partial_i \left(\zeta_s \partial_i \frac{\sigma}{\rho} \right), \tag{7}$$

$$D_t \rho = -\rho \partial_i V_i + H_\rho \partial_i (\zeta_s \partial_i \rho), \tag{8}$$

$$D_t \zeta_l = \partial_i (\kappa_\zeta \partial_i \zeta_l) + \frac{1}{\tau_l} \lambda c_{\max} - \frac{1}{\tau_l} \zeta_l, \tag{9}$$

$$D_t \zeta_s = \partial_i (\kappa_\zeta \partial_i \zeta_s) + \frac{1}{\tau_{s+}} S_s - \frac{1}{\tau_{s-}} \zeta_s, \tag{10}$$

where λ is the Nyquist length, and $c_{\max} = \sqrt{c_s^2 + v_a^2}$ is the maximum signal speed, where c_s is the sound speed and v_a is the Alfvén speed. Bulk viscosity fields ζ_l and ζ_s are introduced to handle the general flow (ζ_l), and shocks and other discontinuities (ζ_s). The action and parameterization of these fields are described in Section 6.1. H_σ and H_ρ are constants that specify the strength of mass and thermal conductivities in continuity and energy equations, while ξ_j is the term representing diffusion of magnetic divergence defined in Section 6.2.

To evolve the field variables in time, we evaluate Equations (1)–(4) for the time derivatives, requiring values for the field variables and their spatial derivatives at the position of each target particle. We obtain this information by fitting a third-order, three-dimensional (3D) polynomial to the set of values carried by the neighboring particles, using the procedure described in Section 3. The resulting polynomial coefficients allow us to compute the field value and its first, second, and third derivatives at the position of the particle, enabling evaluation of the Lagrangian time derivatives. Those in turn are used to update the field variables with a predictor-corrector time step scheme described in Section 3.3.

A particle-based algorithm such as this one has a dynamically evolving spatial resolution. It turns out to be central to the stability and accuracy of the method that the particle distribution not have voids within which the fields cannot be accurately fit. We create and delete particles as necessary to eliminate such voids, while avoiding particle clumps. This further allows us to adaptively satisfy any user-specified physical resolution requirement, as well as to eliminate unnecessary particles (Section 4). We force the resolution to always exceed a spatially and temporally variable target resolution $\lambda(\boldsymbol{x}, t)$. Effectively, the particles can represent the field variables in the same manner as a grid with effective resolution λ at each point. The resolution requirement can be specified depending on the physics requirements of the problem at hand, so long as it remains reasonably smooth.

The Phurbas discretization is based on point values, not finite volumes or finite masses. As such, the discretization used to calculate spatial derivatives and advance the solution in time does not define a value for volume-integrated quantities, including volume-integrated, conserved quantities. To define these quantities, another discretization would need to be added to obtain a multidimensional quadrature from the unstructured set of samples. For example, Voronoi cell volumes could be used to calculate a nearest-grid-point interpolation for a Riemann-sum approximation to a volume integral. Alternatively, using a point density approximated from the number of neighboring points in some small radius as a weighting, a Monte-Carlo-type volume integral approximation could be used.

As the magnetic field evolves, discretization error generates spurious magnetic divergence $\nabla \cdot \boldsymbol{B}$. By dropping the physically vanishing term $-\nabla \cdot \boldsymbol{B}$ when deriving the Lagrangian induction equation from the usual Eulerian form expressed with a partial time derivative, we have made any $\nabla \cdot \boldsymbol{B}$ present in the field into a passively advected scalar. A consequence of this choice of the canonical Lagrangian form is that our MHD equations, by omitting a term that is physically zero, are precisely the same as a form that is claimed in other works to include an extra source term: the same result has been proposed with a source term by Janhunen (2000), and derived from the relativistic form of energy-momentum conservation and relativistic electromagnetic theory by Dellar (2001). In the latter paper, it is shown to be the Galilean invariant momentum and energy conserving form for the MHD equations in the case when $\nabla \cdot \boldsymbol{B}$ is present. We note that $\nabla \cdot \boldsymbol{B}$ of nonphysical origin is easier to numerically handle in the Lagrangian form of the MHD equations, as the presence of $\nabla \cdot \boldsymbol{B}$ errors does not feed back into violations of energy and momentum conservation by itself. Unlike in SPH MHD (Price 2012b), we do not require source terms in the momentum and energy equations, as our discretization does not suffer from the tensile instability as in SPH. The diffusive correction described in Section 6.2 adds a term to the magnetic equation that diffuses $\nabla \cdot \boldsymbol{B}$.

3. TIME EVOLUTION

We evolve the field variables forward in time by evaluating the MHD Equations (1)–(4) at the position of each particle. We do this by constructing a local approximation of the field variables at that position, derived from a spatial fit to the values of the field variables on neighboring particles (Section 3.1). This allows us to compute the values and spatial derivatives of the field variables at the position of the particle. We choose for the form of the continuous approximation a 3D, third-order, polynomial. We further develop a system of particle weights that enhances the accuracy of the fit (Section 3.2). Once we have evaluated the Lagrangian time derivatives from the MHD equations, the field variables and particle positions are updated in time with a predictor-corrector method (Section 3.3).

3.1. Moving Least Squares Procedure

Phurbas uses a moving-least-squares fitting procedure in two versions. First, to approximate derivatives of the dynamical field variables on the right-hand sides of the governing equations, moving-least-squares *interpolants* are used. These are polynomial approximations that are forced though (interpolate) the central particle value. Second, Phurbas uses moving-least-squares *fits* to initialize newly created particles. These are polynomial approximations that are not forced though any particle values, and hence provide a smooth approximation to the field values where no particle currently exists.

In the language of Lancaster & Salkauskas (1981), the first version is an Interpolating Moving Least Squares procedure and the second is a Moving Least Squares procedure. In addition to Lancaster & Salkauskas (1981), discussion at length can be found, for example, in Belytschko et al. (1996), Dilts (1999), or Fries & Matthies (2004).

For the purposes of Phurbas, we can follow the description by Liszka et al. (1996), which leads to what has become known as a Generalized Finite Difference Method (Liszka & Orkisz 1980).

Phurbas uses both the Nodal Approximation described in Liszka et al. (1996, Section 2.1.1) and the Pointwise Approximation in Liszka et al. (1996, Section 2.1.2). We briefly expand on those descriptions here for clarity.

In one dimension, we start with a function that we wish to discretize, defined at some set of points $g = g(\mathbf{x})$. To approximate the function at a point $\mathbf{x}_0$, we select a set of nearby points $\{\mathbf{x}_i\}$, and then write the series approximation using n polynomial terms $p_i(\mathbf{x})$,

$$q(\mathbf{x}) = \sum_{i=0}^{n} a_i p_i(\mathbf{x} - \mathbf{x}_0) \tag{11}$$

so that $q(\mathbf{x}) \approx g(\mathbf{x})$. If the functions p_i are selected as polynomials,

$$p(\mathbf{x}) = [1, x, y, z, x^2, y^2, z^2, xy, xz, x^3, \ldots] \tag{12}$$

then this approximation is a Taylor series about $\mathbf{x}_0$, as in Equation (11) we have shifted the polynomials p to be centered on $\mathbf{x}_0$. The coefficients a_i are then the value and derivatives (multiplied by a Taylor series coefficient) of the approximation $q(x)$.

If the function g is defined at $\mathbf{x}_0$ we can reduce the approximation to a special case, called the Nodal Approximation by Liszka et al. (1996, Section 2.1.1). If we fix the coefficient $a_0 = g(\mathbf{x}_0) = q(\mathbf{x}_0)$, then only the coefficients a_i for $i > 1$ need to be determined, and the approximation becomes

$$q(\mathbf{x}) = g(\mathbf{x}_0) + \sum_{i=1}^{n} a_i p_i(\mathbf{x} - \mathbf{x}_0). \tag{13}$$

We choose to use a number of neighboring points n greater than the number of undetermined polynomial coefficients m. As Equation (13) is then overdetermined, we seek a solution in the least-squares sense. That is, the solution for a_i should minimize the quadratic form

$$J = \sum_{j=1}^{n} W(\mathbf{x}_j - \mathbf{x}_0)(g(\mathbf{x}_j) - q(\mathbf{x}_j))^2, \tag{14}$$

where W is a weight function described below. We can rewrite this set of equations in matrix form by defining

$$\mathbf{g}^{\mathbf{T}} = [g(\mathbf{x}_1) - g(\mathbf{x}_0), g(\mathbf{x}_2) - g(\mathbf{x}_0), g(\mathbf{x}_3) - g(\mathbf{x}_0), ...], \tag{15}$$

$$\mathbf{P} = \begin{bmatrix} p_1(\mathbf{x}_1 - \mathbf{x}_0) & p_2(\mathbf{x}_1 - \mathbf{x}_0) & \ldots & p_m(\mathbf{x}_1 - \mathbf{x}_0) \\ p_1(\mathbf{x}_2 - \mathbf{x}_0) & p_2(\mathbf{x}_2 - \mathbf{x}_0) & \ldots & p_m(\mathbf{x}_2 - \mathbf{x}_0) \\ \vdots & \vdots & \ddots & \vdots \\ p_1(\mathbf{x}_n - \mathbf{x}_0) & p_2(\mathbf{x}_n - \mathbf{x}_0) & \ldots & p_m(\mathbf{x}_n - \mathbf{x}_0) \end{bmatrix}, \tag{16}$$

and

$$\mathbf{W} = \begin{bmatrix} W(\mathbf{x}_1 - \mathbf{x}_0) & 0 & \ldots & 0 \\ 0 & W(\mathbf{x}_2 - \mathbf{x}_0) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & W(\mathbf{x}_n - \mathbf{x}_0) \end{bmatrix}. \tag{17}$$

Then Equation (14) can be written

$$J = (\mathbf{Pa} - \mathbf{g})^{\mathbf{T}}\mathbf{W}(\mathbf{Pa} - \mathbf{g}). \tag{18}$$

If we define

$$\mathbf{A} = \mathbf{P}^{\mathrm{T}}\mathbf{WP} \tag{19}$$

$$\mathbf{B} = \mathbf{P}^{\mathrm{T}}\mathbf{W} \tag{20}$$

then minimizing J as in Belytschko et al. (1996) we obtain

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{Bg} \tag{21}$$

and

$$\mathbf{a}^{\mathbf{T}} = \left[\frac{\partial q}{\partial x}, \frac{\partial q}{\partial y}, \frac{\partial q}{\partial z}, 2\frac{\partial^2 q}{\partial x^2}, \ldots\right] \tag{22}$$

gives the derivatives of the interpolating moving-least-squares approximation.

The second form, the Pointwise Approximation, is defined everywhere, not just where there is a particle. This form is used when adding new particles. The coefficient a_0 is left free, the approximation is Equation (11), and now at an arbitrary point $\mathbf{x}$ the approximation yields

$$\mathbf{g}^{\mathbf{T}} = [g(\mathbf{x}_0), g(\mathbf{x}_1), g(\mathbf{x}_2), g(\mathbf{x}_3), \ldots], \tag{23}$$

$$\mathbf{P} = \begin{bmatrix} p_0(\mathbf{x}_0 - \mathbf{x}) & p_1(\mathbf{x}_0 - \mathbf{x}) & \ldots & p_m(\mathbf{x}_0 - \mathbf{x}) \\ p_0(\mathbf{x}_1 - \mathbf{x}) & p_1(\mathbf{x}_1 - \mathbf{x}) & \ldots & p_m(\mathbf{x}_1 - \mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ p_0(\mathbf{x}_n - \mathbf{x}) & p_1(\mathbf{x}_n - \mathbf{x}) & \ldots & p_m(\mathbf{x}_n - \mathbf{x}) \end{bmatrix}, \tag{24}$$

and

$$\mathbf{W} = \begin{bmatrix} W(\mathbf{x}_0 - \mathbf{x}) & 0 & \ldots & 0 \\ 0 & W(\mathbf{x}_1 - \mathbf{x}) & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & W(\mathbf{x}_n - \mathbf{x}) \end{bmatrix}. \tag{25}$$

The vector of coefficients then yields

$$\mathbf{a}^{\mathbf{T}} = \left[q(\mathbf{x}), \frac{\partial q}{\partial x}, \frac{\partial q}{\partial y}, \frac{\partial q}{\partial z}, 2\frac{\partial^2 q}{\partial x^2}, \ldots\right]. \tag{26}$$

The coefficients vector has either 19 or 20 coefficients, depending on the approximation, so the solution requires inversion of either a 19 × 19 or 20 × 20 matrix $\mathbf{A}$. We use an LU decomposition and back substitution procedure (e.g., Press et al. 1992, p. 32) to solve Equation (21). The derived polynomial coefficients yield the values of the field variables and their derivatives of first, second, and third orders, from which we construct the first- and second-order time derivatives of the field variables.

We have experimentally found that the number of particles included in the evaluation sums for the matrix coefficients should comfortably exceed the number of terms in the polynomial. The choice of how many particles to include is based on a compromise between lack of statistical significance and computational impracticality. The radius r_f of the sphere encompassing the particles included should also be large enough to justify a third-degree interpolation, about twice the characteristic interparticle separation λ. For a uniform particle density of one particle within each volume λ^3, a sphere with $r_f = 2\lambda$ encloses $\sim$34 particles. However, this leaves little room to account for non-uniform particle densities. A sphere with $r_f = 3\lambda$ encloses $\sim$110 particles, which weighs on the cost of calculating the matrix coefficients. A radius as large as this also invites higher-order structure to erode the interpolation. In the end we choose to use

$$r_f = 2.3\lambda, \tag{27}$$

corresponding to $\sim$51 particles. We have not yet derived a rigorous lower bound to the required number of neighbors to use. For computational cost, we find that a third-order fit has computational cost comparable to the other operations that occur in the time step, while a fourth-order fit is substantially more expensive. We term the sphere of radius r_f around a fit center the neighbor sphere.

The target resolution $\lambda(\boldsymbol{x}, t)$ is a property of the location of the fit center, which may or may not be centered at the location of a particle. In practice, as the interpolation is centered at the location of an existing particle, then λ for the interpolation is taken as the λ of that particle. If a fit is being used to generate new values for the addition of a particle, the λ of the particle that triggered the creation of the new particle is used.

3.2. *Weights*

In the moving-least-squares procedure, each neighbor particle j has a weight W_j. There is significant freedom to choose the form of the weights W_j, and no rigorous theoretical framework exists under which an optimal choice can be made. As a first approximation, we choose weights that emphasize particles close to the center of the neighbor sphere, so that the local least-squares approximation varies more smoothly as the target position is changed. The weighting function is a piecewise linear function of the distance of particle j from the fit center r_j, given as:

$$W_j = \begin{cases} 1 & \text{if } 0 \leqslant r_j < r_w \\ 1 - \frac{3}{5}\left(\frac{r - r_w}{r_f - r_w}\right) & \text{if } r_w \leqslant r_j \leqslant r_f, \end{cases} \tag{28}$$

where $r_w = \frac{3}{2}\lambda$. W_j has a nonzero value at the edge of the neighbor sphere so as to not exclude any particles from the fit.

3.3. *Time Update*

The field variables are evolved in time with a Hermite predictor-corrector scheme based on the first- and second-order Lagrangian time derivatives. A derivation of the scheme is presented in Appendix B, as it has not previously been described in the literature. The interpolation procedure in Section 3.1 for time step $i + 1$ is done on the predicted values $q_{p,i}$ computed in time step i, yielding the time derivative values $D_t q_{p,i}$ and $D_{tt} q_{p,i}$ needed for the correction in time step i, as well as for the prediction of time step $i + 1$.

We begin by extrapolating forward from time t_i to time t_{i+1}, over the time interval $\Delta t = t_{i+1} - t_i$, to make a prediction

$$q_{p,i+1} = q_{c,i} + D_t q_{p,i}\Delta t + \frac{1}{2} D_{tt} q_{p,i}\Delta t^2, \tag{29}$$

based on a Taylor series expansion around $q_{p,i}$, using the corrected value from the previous time step $q_{c,i}$. We then evaluate the time derivatives by interpolation on the predicted fields $q_{p,i+1}$ at time t_{i+1}, and correct the prediction to derive the corrected value at t_{i+1},

$$q_{c,i+1} = q_{c,i} + \frac{1}{2}(D_t q_{p,i} + D_t q_{p,i+1})\Delta t + \frac{1}{12}(D_{tt} q_{p,i} - D_{tt} q_{p,i+1})\Delta t^2. \tag{30}$$

The particle positions $\boldsymbol{x}$ are evolved using third-order time information $D_{ttt}\boldsymbol{x} = D_{tt}\boldsymbol{V}$ as well. This allows us to use a third-order predictor of the form

$$x_{p,i+1} = x_{c,i} + V_{c,i}\Delta t + \frac{1}{2} D_t V_{p,i}\Delta t^2 + \frac{1}{6} D_{tt} V_{p,i}\Delta t^3 \tag{31}$$

and to correct it to the final value

$$x_{c,i+1} = x_{c,i} + \frac{1}{2}(V_{c,i} + V_{c,i+1})\Delta t + \frac{1}{10}(D_t V_{p,i} - D_t V_{p,i+1})\Delta t^2 + \frac{1}{120}(D_{tt} V_{p,i} + D_{tt} V_{p,i+1})\Delta t^3. \tag{32}$$

4. REGULARIZING THE PARTICLE DISTRIBUTION

Our algorithm relies on discretization over Lagrangian sample points. These points are not arrayed on a grid, nor are they connected by mesh edges as in the AREPO code (Springel 2010), so this is a meshless method. During evolution, we require that the particles should maintain a distribution such that there are no voids larger than the target local resolution $\lambda(\boldsymbol{x}, t)$ and no excessive point concentrations within the scale λ. The requirement of no voids is introduced to ensure that every fit sphere of radius r_f has enough points to perform the moving-least-squares procedure and that the fit spheres overlap sufficiently so that the set of fit spheres covers the entire simulation volume. The requirement of no point concentrations requires the removal of excess points. We implement these requirements by adding and deleting particles as needed. Satisfying these requirements confers the great benefit of making the code fully adaptive, since the user can dynamically choose the function $\lambda(\boldsymbol{x}, t)$ as required by the physics of the problem, so long as it is reasonably smooth in space and time.

The addition and deletion algorithm begins with the assembly of all neighbors i within the neighbor sphere of radius r_f around a particle j, along with their associated target resolutions λ_i. Voids within the neighbor sphere are identified using the method described in Section 4.1. Any voids identified are reported as candidates for particle creation. Conversely, if a particle j has a mutual nearest neighbor that is too close (see Section 4.2), one of the two particles is deleted. Duplicate voids and clumps are pruned from the global list prior to the particle creation and deletion described in Section 4.3.

4.1. *Voids*

To check for a void at a point in space with position $\boldsymbol{x}$, we identify the nearest particle i, which is located at position $\boldsymbol{x}_i$ and has a resolution scale λ_i. The distance between $\boldsymbol{x}$ and the particle normalized by the resolution scale is then

$$x_{\text{void}} = \frac{|\boldsymbol{x} - \boldsymbol{x}_i|}{\lambda_i}. \tag{33}$$

As a resolution condition, we then choose the condition that if

$$x_{\mathrm{void}} > c_{\mathrm{void}}, \tag{34}$$

for a constant c_{void}, the space around $\boldsymbol{x}$ is indeed too sparsely populated, indicating a need for particle addition.

To heuristically derive c_{void}, we consider an arrangement of particles on the hexagonal lattice representing the tightest possible packing of spheres centered on the particles. If the particle density is one particle per volume λ^3, then the spheres' centers will be separated by a distance $d_p = 2^{1/6}$. This represents the most efficient possible filling of the region with particles. Since any real, fluctuating, particle distribution will require more particles to fully resolve the field, we set

$$c_{\mathrm{void}} = 0.73 d_p \tag{35}$$

so that with a disordered particle set we sample more density than would be required with the ideal ordered particle set.

To identify unique voids, we first identify the most egregious void within the neighbor sphere of each particle, and then check to see if that void violates the condition given by Equation (34). If it does we add a particle as described below in Section 4.3. We begin by examining the space in the vicinity of existing particles. We construct a 3D cubic grid with side length $2r_f$ containing $9 \times 9 \times 9$ grid points, centered on the target particle position $\boldsymbol{x}_j$. This grid covers the volume of the neighbor sphere. For each grid point, the normalized distance x_{void} to all neighboring particles can be calculated using Equation (33) and the minimum value chosen. If the maximum value on the grid of $x_{\mathrm{void}} > c_{\mathrm{void}}$, the position of the grid point with the maximum value is reported as a candidate void for particle creation.

To speed up the calculation, we sieve the grid points lying within the neighbor sphere. We begin the search by initializing a large value on each grid point for the minimum value of x_{void} for that grid point. We then proceed by selecting each particle i in turn, and looping over all grid points. For each grid point, we calculate the normalized distance x_{void} to the particle i. If its value for the grid point is less than the current minimum value on that point, we replace it with the newly calculated value for particle i. If the new value is less than c_{void}, that grid point can be eliminated from the active list of candidates for void identification. We then move to the next particle and calculate its distance to the remaining active grid points, repeating the above procedure. After all particles have been sieved, if any grid points remain as void candidates, we report the one with the maximum x_{void} as a candidate for void creation. To hasten the operation, we first sieve the particles within λ from the target position, then those within $(3/2)\lambda$, and then the remaining particles, where λ is the value for the target position.

4.2. *Clumps*

As the particles move, random fluctuations will move them closer or farther from their neighbors. If two particles approach each other too closely compared to λ, they are essentially sampling the same field variable information, and so are redundant. Because there are no restoring forces in the algorithm to separate nearby particles, we instead remove any particle clumps of this sort, saving the computational cost of evolving the redundant particles. The question then remains of how to determine when a clump has formed.

To do this, we define a scaled distance between two particles:

$$r_{ij}^2 = \frac{(\boldsymbol{x}_i - \boldsymbol{x}_j)^2}{\lambda_i \lambda_j}. \tag{36}$$

The nearest neighbor i to the target particle j is determined. In turn its nearest neighbor is found. If they are mutual nearest neighbors and if

$$r_{ij} < c_{\mathrm{clump}} \tag{37}$$

they are candidates for deletion. We find that a value of

$$c_{\mathrm{clump}} = 0.12 d_p \tag{38}$$

is suitable to prevent over-resolution. Among those two particles, we delete the one which was more recently created, retaining the particles with longer history to minimize the numerical diffusion from adaption.

If particles are to be deleted, this is done so without considering whether that particle triggered the proposed addition of a particle (that is, whether it is in a clump on the edge of a void). However, any proposed addition resulting from the processing of that particle is still considered.

4.3. *Particle Creation and Deletion*

The first examination of all the active particles results in a proposed list of positions requiring particle addition, accompanied by the radius of the void detected. These proposals overlap, as each void may be detected by more than one particle. The list is exchanged by processes handling neighboring spatial domains, so that each process has a list of all the proposed additions within a distance r_f of its boundary. The proposed addition list is then pruned, to select one position in which to add a particle within each void radius. To do this, each particle addition proposal is compared to all other proposals within that spatial domain. If any other proposed location lies within its void radius, the values of the void radii are compared, and the proposal with the smaller void radius is rejected.

We then create particles at the successfully proposed positions. Particles in this algorithm represent sample points, not discrete parcels of gas. When we add particles, we are just sampling the continuous field variables at new positions. Therefore, considerations of conservation do not enter this process, unlike in particle-splitting methods used in SPH (e.g., Kitsionas & Whitworth 2002).

The task of creating new particles needs to be load balanced among processors in order to handle situations where the memory required for new particles represents a large fraction of the total free memory in the particle arrays. The new particles are then initialized in free spaces, on the processors to which they have been assigned by the addition load balance procedure. As we have now deleted some particles, and added others to essentially random processors, a new load balance may be calculated among all particles, and the neighbor search data structure must be updated. Doing this on the entire particle list brings the new particles to optimal positions on the processors and provides neighbor information for the subsequent processing stages.

New particles are initialized using a third-order moving-least-squares fit, as opposed to an interpolation (Section 3.1). This fit is centered on the position of the new particle.

5. TIME STEPS

The time step for each particle is set by taking the minimum of five criteria. These are evaluated at the phase where new time derivatives are computed. The basic limit is the CFL condition for the stability of a forward-time-centered-space discretization. It is used here without explicit derivation as the general principle applies that the maximum stable time step must be short enough that a signal cannot cross a distance exceeding the local resolution λ,

$$\Delta t_{\rm CFL} = C_{\rm CFL} \frac{\lambda}{\sqrt{c_s^2 + v_A^2}}, \tag{39}$$

where $\Delta t_{\rm CFL}$ is the CFL time step, $C_{\rm CFL}$ is the Courant number, which we usually take to be 0.3, c_s is the sound speed, and v_A is the Alfvén speed. Note that, unlike an Eulerian code, the flow velocity does not enter this equation.

The use of the bulk viscosity fields ζ_s and ζ_l require an appropriate time step constraint related to the diffusion term they introduce in the momentum equation,

$$\Delta t_\zeta = \frac{C_{\rm CFL}}{\pi^2(\zeta_s + \zeta_l)}. \tag{40}$$

Another time step constraint of the same form is applied based on the need for a sufficient number of time steps during a compression or expansion to allow for particle addition and deletion. This is the von Neumann time step

$$\Delta t_{\rm VC} = \frac{C_{\rm CFL}}{\pi^2 C_{\rm VC}^2 |\nabla \cdot \boldsymbol{V}|}, \tag{41}$$

where $C_{\rm VC}$ is a constant, which we usually take to be 2. The arbitrary form of the constant term $\pi^2 C_{\rm VC}^2$ comes from an analogy with the form of the von Neumann time step constraint by considering the von Neumann term

$$C_{\rm VC}^2 \lambda^2 (-\nabla \cdot \boldsymbol{V})_+ (-\nabla \cdot \boldsymbol{V}), \tag{42}$$

where $()_+$ denotes that the expression is zero if the term contained is negative, as a diffusion operator and following the time step constraint from Maron & Mac Low (2009, Equation (8)). We also introduce a similar constraint based on the shear of the flow to allow for needed regularization, although the constraint on this vorticity time step much looser than in compression and expansion:

$$\Delta t_{\rm VR} = \frac{C_{\rm CFL}}{10\pi^2 C_{\rm VC}^2 |\nabla \times \boldsymbol{V}|}. \tag{43}$$

The factor $10\pi^2 C_{\rm VC}^2$ is an ad hoc scaling that in practice has been found to be sufficient.

The time step limit assigned for a particle is

$$\Delta t = \min(\Delta t_{\rm CFL}, \Delta t_\zeta, \Delta t_{\rm VC}, \Delta t_{\rm VR}). \tag{44}$$

Each particle has an individually assigned time step. To tailor the algorithm to parallel implementation, a block time step scheme can be adopted. For such a time step scheme, the time steps actually used are rounded down to the nearest block time step interval.

It is also necessary to ensure some degree of spatial coherence to the time steps, so that disturbances propagate from short-time step particles to long-time step particles smoothly. At the end of the time update procedure for a particle, after the assignment of the new time step for a particle, if any of the neighbor particles has an end time greater than the target particle's new end time, a time step limit propagation procedure is triggered for the target particle. The target particle's end time propagates to its neighbors, and if any neighbor's end time is farther from the current time than twice the interval to the target particle's end time, then the neighbor's end time is set to this limit.

6. ARTIFICIAL DIFFUSION

Three types of artificial diffusion terms are used to stabilize the solutions to the equations modeled in Phurbas. These are terms acting as bulk viscosity, mass and internal energy diffusion terms, and a term acting to diffuse magnetic monopoles. We find that these terms are sufficient to damp small scale fluctuations that would otherwise make the scheme unstable.

6.1. Bulk Viscosities

We use two bulk viscosity fields. The first, ζ_l, quenches small scale compressive motion in all areas of the flow, and evolves according to Equation (9). The second, ζ_s, is a shock and discontinuity viscosity that evolves according to Equation (10). Equations (9) and (10) each consist of three terms, a diffusion term, a source term, and a decay term. This configuration ensures that the bulk viscosity fields vary smoothly in time and space.

The diffusion operator on the bulk viscosity fields $\kappa_\zeta = 0.15\lambda c_{\max}$ is chosen to place a time step limit less stringent than the Courant limit from the hyperbolic part of the MHD equations. The bulk viscosity fields have source and decay terms and these terms have associated timescales. For the ζ_l field the source and decay term timescales are the same, $\tau_l = \lambda/c_{\max}$. The ζ_s field is designed to fall off more slowly than it rises, which is particularly advantageous calming post-shock oscillations, so $\tau_{s-} = 20\lambda/c_{\max}$ and $\tau_{s+} = \lambda/c_{\max}$. The use of a diffusion equation with source and decay terms to derive the artificial viscosity field here is analogous to the design in a discontinuous Galerkin method by Barter & Darmofal (2010) and the slow decay of the shock viscosity achieves a similar effect to the bulk viscosity prescription used by Morris & Monaghan (1997).

The ζ_s source term S_s is given by

$$S_s = \max\Big(C_{\rm VN}^2(-\partial_i V_i)_+\lambda^2, \quad C_e|\partial_i\partial_i(\sigma/\rho)|\lambda^2,$$
$$C_\rho \rho^{-1}|\partial_i\partial_i(\rho)|\lambda^3\sqrt{c_s^2+v_a^2}, \quad C_P P^{-1}|\partial_i\partial_i(P)|\lambda^3\sqrt{c_s^2+v_a^2}\Big). \tag{45}$$

The first term has the form of the conventional von Neumann artificial viscosity, with $C_{\rm VN} = 2$. The second term responds to changes in the specific internal energy, in a manner similar to the dissipation introduced by Price (2008). This form is a trigger on the size of the second derivative of the specific internal energy, and $C_e = 0.1$. The third term is constructed analogously to the second, but using the Laplacian of density and $C_\rho = 1.0$. The final term is again constructed analogously to the second, but using the Laplacian of pressure and $C_P = 3.0$. The constants $C_{\rm VN}$, C_e, and C_P can be tuned for a particular problem, with smaller values being preferable, but the values given here have proven to be sufficient for most problems.

The mass and internal energy diffusion terms are coupled to the ζ_s field, with a strength set by the constants $H_\rho = H_\sigma = 5 \times 10^{-4}$. For stability, it would be preferable to have a small scale mass and internal energy diffusion (such as a hyperdiffusion) active everywhere in the flow, but we have not found a formulation of such a term that is sufficiently accurate to yield reasonable mass conservation results.

6.2. Magnetic Divergence Diffusion

As Phurbas solves the equations of MHD, the issue of magnetic monopole errors must be treated. The primary problem caused by monopole errors in schemes of this type is numerical instability. The interpolating moving-least-squares derivative estimates may return derivatives of the magnetic field that do not satisfy $\nabla \cdot \boldsymbol{B} = 0$. Over several time update cycles, these estimates may lead to the local creation of a net magnetic monopole character to the field. In practice, we have found that a diffusive (or parabolic) correction is sufficient to prevent the growth of this monopole character of the magnetic field (see tests in Paper II).

For each particle, a $\nabla \cdot \boldsymbol{B}$ field is defined. The value is simply reset each time step to the value of $\nabla \cdot \boldsymbol{B}$ derived from the fit to the magnetic field. The derivatives of the $\nabla \cdot \boldsymbol{B}$ field derived from the fits are then used to diffuse $\nabla \cdot \boldsymbol{B}$, generally resulting in a reduction of its value. These fitted values and derivatives of $\nabla \cdot \boldsymbol{B}$ are less noisy than values and derivatives of $\nabla \cdot \boldsymbol{B}$ derived directly from fits to the magnetic field.

The diffusion term for particle j is

$$\xi_j = \eta_{\max} \nabla(\nabla \cdot \boldsymbol{B}_j), \tag{46}$$

where $\eta_{\max}$ is the maximum diffusion coefficient possible under the stability criterion

$$\Delta t < \frac{\lambda^2}{\pi^2 \eta} \tag{47}$$

from Maron & Mac Low (2009, Equation (8)). The term given by Equation (46) is added to the right-hand side of the induction equation (Equation (2)) in the first time derivatives used in the second-order predictor-corrector scheme for the evolution of the magnetic field. The effect of this is that the $\nabla \cdot \boldsymbol{B}$ diffusion operator is integrated with a first-order predictor-corrector scheme. The $\nabla \cdot \boldsymbol{B}$ diffusion $\eta_{\max}$ is computed each time the fields are fit, which occurs at times that are the end of one time step and the beginning of the next. The time step used to define $\eta_{\max}$ is the time step that has its end at the instant $\eta_{\max}$ is calculated, i.e., the previous time step. Thus, the $\eta_{\max}$ used in the predictor stage of the time integration of a particular step is different from the $\eta_{\max}$ later used in the corrector stage of the same time step. This diffusion is not conservative, but the Phurbas discretization only preserves the conservation in the MHD equations to truncation error levels, and the $\nabla \cdot \boldsymbol{B}$ operated on by this diffusion is, by definition, only created below truncation error levels. We find that since the canonical form of the Lagrangian MHD equations that we use treats $\nabla \cdot \boldsymbol{B}$ as a passively advected scalar, the presence of small amounts of $\nabla \cdot \boldsymbol{B}$ does not destabilize the solution.

7. SUMMARY AND DISCUSSION

7.1. Summary of the Algorithm

We now summarize the conceptual steps of the algorithm. The operations described here are actually often broken into multiple phases to enable efficient parallelization. Future implementers of the algorithm should consider the specific needs of each operation when designing data structures and communication patterns.

1. Build the neighbor-finding data structure, such as a particle tree.
2. Balance particles among processors.
3. Identify target particles for evolution.
4. Use the tree to assemble all neighbors within a radius r_f of the target particles.
5. Check for voids. Complete particle addition for qualifying voids (Section 4.1).
6. Check if there are mutual nearest neighbor pairs that are too close. Delete one (Section 4.2).
7. Evaluate the polynomial fit weights (Section 3.2).
8. Compute the local polynomial fit to derive values and spatial derivatives at the location of each particle (Section 3.1).
9. Use the polynomial coefficients to evaluate the MHD equations for the Lagrangian time derivatives including diffusivity terms (Section 2).
10. Use the polynomial coefficients to evaluate the governing equations of the bulk viscosity fields ζ_l and ζ_s for the Lagrangian time derivatives (Section 2).
11. Use the time derivatives to correct the previous time step (Section 3.3).
12. Evaluate the resolution scale λ for each particle position (Section 2).
13. Evaluate the size of the next time step (Section 5).
14. Use the time derivatives to predict forward in time to the next time step (Section 3.3).
15. Restrict local time step variations (Section 5).

7.2. Effective Resolution

To understand the effect of varying the effective resolution parameter λ on the numerical resolution, consider a one-dimensional uniform grid with a grid spacing of unity, initialized with a field variable having values given by the Fourier mode $\sin(\pi k x)$. The maximum wave number k of this mode that can be expressed on this grid is $k = 1$, the Nyquist wave number. In order to be able to calculate realistic derivatives with finite differences, k must be less than unity, and the precision increases as k decreases. Maron et al. (2008) and Maron & Mac Low (2009) evaluate the effective precision of finite difference schemes with varying stencil sizes. They find that for a stencil radius of $\{1, 2, 3, 4\}$, finite differences can be calculated with a relative precision of $\sim$1 percent up to a wave number of $k \sim \{1/8, 1/4, 2/5, 1/2\}$. Given that derivatives are more easily calculated on a grid than for irregular particles, we take this as an upper limit for what we can expect from particles. Since a 3D, third-order, polynomial corresponds to a 5-point (or stencil radius 2) 1D finite-difference scheme, we expect $k \sim 1/4$ to be the limit of resolution, corresponding to a wavelength of $\sim 8\lambda$.

8. CONCLUSION

We have described Phurbas, an adaptive, Lagrangian, meshless algorithm for MHD. The algorithm is described for the specific case of the MHD equations, but can be easily generalized to other hyperbolic systems, as the fitting, time integration, and stabilization procedures do not rely on particular properties of the MHD equations. The central principle of the algorithm is that the solution and its spatial derivatives are derived from a high-order, interpolating, polynomial fit to a set of particles that are merely Lagrangian sample points in the flow, not mass elements as in SPH or finite volume methods. This allows for significant flexibility in the design of the algorithm, and the implementation of additional physical processes. Particle addition and deletion is required to prevent the growth of voids or clumps. This naturally allows the numerical resolution to be fully adaptive based on user specified criteria. The Lagrangian nature of the code means that particles can be evolved with time steps dependent only on the nature of the local flow, and that numerical diffusion is Galilean invariant. The version described here is just one subset of the many available options.

Paper II describes a parallel implementation and tests of Phurbas that demonstrate accuracy comparable to that of third-order grid codes on subsonic and supersonic problems.

A few theoretically desirable improvements to the scheme can already be identified. Though the ζ_l field is sufficient to modify the equations to give reasonable results in tests, it would be preferable to only use stabilizing viscosities that scale as λ^2 or a higher power (hyperviscosities) to give faster convergence of the modeled equations to the limit of ideal MHD. Possible avenues though which such an effect could be achieved are: use of a different interpolation scheme than moving least squares, and/or a time or spatially dependent version of ζ_l so that it couples only to the shortest wavelength or shortest timescale motions. Kuhnert (1999) showed methods for introducing upwinding into a scheme similar to Phurbas, a modification that may reduce the need for ζ_l. Additionally, in defining ζ_s we have used a simple formulation where the effects decay on the same timescale and reflect the effects of discontinuities (shocks and otherwise) through the same parameter. In SPH, Price (2012b) and Rosswog (2009) have found that separate parameters for each discontinuity are useful. Analogously in the framework here, ζ_s could be broken into three fields, though this would come at some cost.

The second-order Hermite predictor-corrector scheme is particularly useful as it only requires computations of spatial derivatives at the beginning of each time step, and the predictor half of the integration can be done without knowing the end-time of the time step. It however has the drawback that first and second time derivatives of the field variables must be obtained. In general, these analytic expressions could be very complicated. A time integration scheme, such as a Runge–Kutta method, that uses only first time derivatives may be preferable in this sense.

The anonymous referee made a contribution equivalent to authorship, by pointing out the need for the approximating functions to be interpolating to maintain numerical stability, and by providing a first version of the stability analysis given in Appendix A. The authors also acknowledge useful discussions with A. Brandenburg, V. Springel, and S.C.O. Glover. M.-M.M.L. and C.P.M. acknowledge hospitality from the Max-Planck-Institut für Astronomie, and M.-M.M.L additionally acknowledges hospitality of the Institut für Theoretische Astrophysik der Universität Heidelberg. This work has been supported by National Science Foundation grants AST-0612724 and AST-0835734. Computations were performed under allocation TG-MCA99S024 from the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

APPENDIX A
STABILITY OF A MODEL SCHEME

To determine the stability requirements for our meshless method, we here present a stability analysis for a simple model that captures the essential points of our algorithm. We analyze a numerical scheme for approximating the solution of the diffusion equation

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}. \tag{A1}$$

We discretize $u(x)$ on a grid with spacing Δx. To find derivatives of u we use a moving-least-squares approximation. Shifting the origin to grid point x_0 the polynomial approximation of $u(x)$ is given by

$$U(x) = \sum_{p=0}^{P} a_p x^p. \tag{A2}$$

The coefficients a_p are determined by minimizing the sum of the square of errors E_2 at $2N+1$ neighboring grid points, given by

$$E_2 = \sum_{j=-N}^{N} (u_j - U(x_j))^2. \tag{A3}$$

Combining these two expressions, and using the definition of the grid point positions $x_j = j\Delta x$ gives

$$E_2 = \sum_{j=-N}^{N} \left(u_j - \sum_{p=0}^{P} a_p j^p \Delta x^p \right)^2. \tag{A4}$$

We can find the minimum of E_2 by setting $\partial E_2/\partial a_q = 0$ yielding

$$\sum_{j=-N}^{N} j^q \Delta x^q u_j = \sum_{j=-N}^{N} \sum_{p=0}^{P} a_p j^{p+q} \Delta x^{p+q}, \tag{A5}$$

which is a system of equations for $q = 0..P$ that can be solved for the polynomial coefficients a_p. The second derivative of the polynomial equation (A2) at $x = 0$ is $2a_2$. So, we can write a scheme to update the solution to Equation (A1) as

$$u_0^{n+1} = a_0 + 2Da_2\Delta t, \tag{A6}$$

where $u_0 = u(x_0)$. In each step we replace the value u_0^n with the fit a_0 and use the second derivative of the moving-least-squares fit to construct a forward-Euler-type time update. This is a Maron & Howes (2003)-type scheme for solving the diffusion equation. As an example, for third-order polynomials ($p = 3$) and $N = 4$ this scheme is specifically

$$u_0^{n+1} = u_0 \left(\frac{59}{231} - \frac{10}{231}\frac{D\Delta t}{\Delta x^2} \right) + \frac{59}{231} \sum_{j=1}^{4} (u_j + u_{-j}) - \frac{5}{231} \sum_{j=1}^{4} j^2 (u_j + u_{-j}) \tag{A7}$$

$$+ \frac{D\Delta t}{\Delta x^2} \left[\frac{-1}{231} \sum_{j=1}^{4} (u_j + u_{-j}) + \frac{1}{154} \sum_{j=1}^{4} j^2 (u_j + u_{-j}) \right]. \tag{A8}$$

To perform a von Neumann stability analysis, we substitute $u_\ell^n = \xi^n e^{ik\ell\Delta x}$, and solve for ξ

$$\xi = \frac{1}{231} \left(59 - 10\frac{D\Delta t}{\Delta x^2} + \sum_{j=1}^{4} (59 - 5j^2) \cos(jk\Delta x) \right) + \frac{D\Delta t}{\Delta x^2} \sum_{j=1}^{4} \left(\frac{-1}{231} + \frac{j^2}{154} \right) \cos(jk\Delta x). \tag{A9}$$

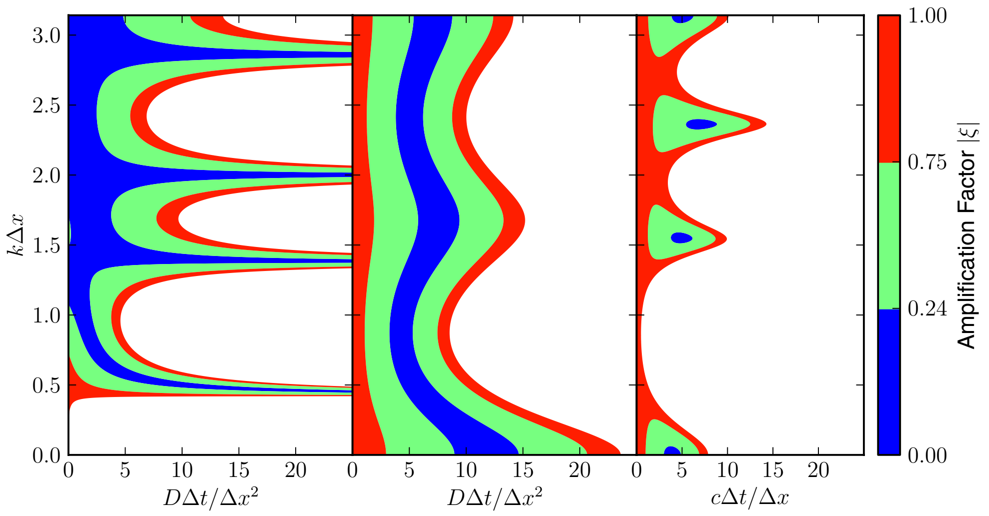

Figure 1. von Neumann stability analysis of model schemes, showing the amplification factor $|\xi|$ as a function of perturbation wavelength k and time step Δt, both appropriately normalized to the grid. The unstable region with values of $|\xi| > 1$ is shown in white. Left: the Maron & Howes (2003)-type scheme for the diffusion equation. Note the instability of low wavenumber perturbations $k\Delta x \lesssim 0.4$. Middle: least-squares interpolant scheme for the diffusion equation. Right: least-squares interpolant scheme for the advection-diffusion equation. Both latter schemes show a region of stability at small enough time step for all wavenumber perturbations.

(A color version of this figure is available in the online journal.)

We can evaluate this expression numerically, plotting the amplification factor $|\xi|$ for each wavenumber $k\Delta x$ and time step $D\Delta t/\Delta x^2$. This is shown in Figure 1. The fundamental trouble with Maron & Howes (2003)-type schemes using a moving-least-squares fit is the region at low wavenumbers and small time steps where the magnitude of the amplification factor is greater than unity. In this region the scheme is unstable.

If instead we use a moving-least-squares interpolant instead of just a fit, the system of equations for the coefficients is given by

$$\sum_{j=-N}^{N} j^q \Delta x^q (u_j - u_0) = \sum_{j=-N}^{N} \sum_{p=1}^{P} a_p j^{p+q} \Delta x^{p+q} \quad \text{(A10)}$$

for $q = 1, 2, 3$. Here we have eliminated the coefficient a_0 by forcing the moving-least-squares approximation to interpolate u_0. Then we can write a forward-Euler-type scheme using the second derivative of this interpolant as

$$u_0^{n+1} = u_0^n + 2Da_2\Delta t. \quad \text{(A11)}$$

For $p = 3$ and $N = 4$ this gives the scheme

$$u_0^{n+1} = u_0 + \frac{D\Delta t}{354\Delta x^2} \sum_{j=0}^{4} j^2 (u_j + u_{-j} - 2u_0) \quad \text{(A12)}$$

$$= u_0 \left(1 - \frac{60D\Delta t}{354\Delta x^2}\right) + \frac{D\Delta t}{354\Delta x^2} \sum_{j=1}^{4} j^2 (u_j + u_{-j}). \quad \text{(A13)}$$

Again, performing a von Neumann stability analysis, we substitute $u_\ell^n = \xi^n e^{ik\ell\Delta x}$ and solve for ξ

$$\xi = \left(1 - \frac{60D\Delta t}{354\Delta x^2}\right) + \frac{D\Delta t}{354\Delta x^2} \sum_{j=1}^{4} j^2 2\cos(jk\Delta x). \quad \text{(A14)}$$

The magnitude of ξ in this case is shown in the center panel of Figure 1 for the case $D = c\Delta x$. In stark contrast to the Maron & Howes (2003)-type scheme constructed with a non-interpolating, moving-least-squares fit, a significant region of stability ($|\xi| < 1$) exists at small time step for all wavenumbers $k\Delta x$.

For the advection equation

$$\frac{\partial u}{\partial t} = c\frac{\partial u}{\partial x} + c\Delta x \frac{\partial^2 u}{\partial x^2}, \quad \text{(A15)}$$

we can write the scheme

$$u_0^{n+1} = u_0^n + c\Delta t a_1 + 2ca_2\Delta x \Delta t. \quad \text{(A16)}$$

Here we have set the diffusion parameter to be scaled by the grid resolution Δx. With the interpolating moving-least-squares coefficients a_1 and a_2 as in the diffusion problem above we have

$$u_0^{n+1} = u_0 \left(1 - \frac{60c\Delta x\Delta t}{354\Delta x^2}\right) + \frac{c\Delta t}{7128\Delta x} \times \left(59 \sum_{j=1}^{4} j^2 (u_j - u_{-j}) - 815 \sum_{j=1}^{4} j (u_j - u_{-j})\right) + \frac{c\Delta x\Delta t}{354\Delta x^2} \sum_{j=1}^{4} j^2 \left(u_j + u_{-j}\right). \quad \text{(A17)}$$

A von Neumann stability analysis yields

$$\xi = \left(1 - \frac{60c\Delta t}{354\Delta x}\right) + \frac{ic\Delta t}{7128\Delta x} \times \left(59 \sum_{j=1}^{4} 2j^2 \sin(jk\Delta x) - 815 \sum_{j=1}^{4} 2j \sin(jk\Delta x)\right) + \frac{c\Delta t}{354\Delta x} \sum_{j=1}^{4} j^2 2\cos(jk\Delta x). \quad \text{(A18)}$$

Note that the second term is imaginary, arising from the advection operator, so if the contributions from the diffusion operator, the part of the first term, and the final term, were dropped, the scheme for the pure advection problem would be unconditionally unstable. We plot the magnitude of the amplification factor $|\xi|$ in Figure 1. The addition of the diffusion operator has stabilized the advection problem for sufficiently small time steps $c\Delta t/\Delta x$.

APPENDIX B

TIME INTEGRATION

Time integration proceeds using a system of Hermite predictor-corrector formulae for field variable and position updates. This scheme is a lower-order version of that presented in Nitadori & Makino (2008). As it is has not previously been described in the literature, we present here a brief derivation of the integration method.

We predict field values $q_{p,i+1}$ at time $t+\Delta t$ using a Taylor series expansion around their values at time t incorporating their time derivatives calculated after the prediction phase of the previous time step, giving

$$q_{p,i+1} = q_{c,i} + D_t q_{p,i}\Delta t + \frac{1}{2} D_{tt} q_{p,i} \Delta t^2, \tag{B1}$$

where $q_{c,i}$ is the corrected field value from the previous time step at time t. After calculating the values of $q_{p,i+1}$, we use them to calculate $D_t q_{p,i+1}$ and $D_{tt} q_{p,i+1}$, as given by Equations (5)–(10), and those in Appendix C. These derivatives will be used in the corrector stage of the current time step and in the predictor stage of the next time step. The stabilizing diffusion terms, linked to the ζ_s and ζ_l fields are proportional to the resolution parameter λ. As asymptotically, the time step varies as λ itself due to the CFL limit, we only integrate these terms to first order in time, so the combined space-time error is second order in λ.

By using higher time derivatives, a Hermite scheme of time integration depending only on field variable values from the previous time step can be constructed. This saves storage, avoids complex start up procedures, and simplifies the use of individual particle time steps in comparison to predictor-corrector schemes based on Newton interpolation (such as Aarseth and Adams–Bashforth–Moulton schemes) that require storage of field values from earlier time steps. The Hermite corrector stage is constructed as

$$q_{c,i+1} = q_{c,i} + \int_0^{\Delta t} f_c(\tau) d\tau. \tag{B2}$$

We choose the function $f_c(\tau)$ to be a Hermite interpolation, that is, a polynomial that interpolates $D_t q$ and $D_{tt} q$ at each end of the time step. We further choose to simplify the formalism by designing the polynomial to be time symmetric about $t + \Delta t/2$, so that

$$f_c(\tau) = f_0 + f_1\left(\tau - \frac{\Delta t}{2}\right) + f_2\left(\tau - \frac{\Delta t}{2}\right)^2 + f_3\left(\tau - \frac{\Delta t}{2}\right)^3. \tag{B3}$$

We determine the coefficients $f_0,\ f_1,\ f_2,\ f_3$ by using four constraints: at $\tau = 0$, $f_c(\tau)$ must have a value of $D_t q_{p,i}$, and a time derivative $D_{tt} q_{p,i}$; while at $\tau = \Delta t$ the value and the derivative must be $D_t q_{p,i+1}$ and $D_{tt} q_{p,i+1}$, respectively. However, evaluating the integral in Equation (B2), the time symmetry we chose yields the simple result that the f_1 and f_3 terms integrate to zero regardless of the values of their coefficients. Performing the integral in Equation (B2) yields a correction stage

$$q_{c,i+1} = q_{c,i} + \frac{1}{2}(D_t q_{p,i} + D_t q_{p,i+1})\Delta t + \frac{1}{12}(D_{tt} q_{p,i} - D_{tt} q_{p,i+1})\Delta t^2. \tag{B4}$$

Velocity V is treated as an independent set of field variables, so for particle positions x there are three time derivatives of information available, as well as corrected values of velocity from the beginning and end of the current time step. Therefore, for the predictor stage for the position we use a third-order Taylor series incorporating the best information available at time t,

$$x_{p,i+1} = x_{c,i} + V_{c,i}\Delta t + \frac{1}{2} D_t V_{p,i} \Delta t^2 + \frac{1}{6} D_{tt} V_{p,i} \Delta t^3. \tag{B5}$$

Similarly to the field variable integration we choose a time-symmetric, Hermite interpolating function, so that the corrector stage is

$$x_{c,i+1} = x_{c,i} + \int_0^{\Delta t} g_c(\tau) d\tau. \tag{B6}$$

The function $g_c(\tau)$ is again a polynomial centered on $t + \Delta t/2$, that now interpolates through $V_{c,i}$, $D_t V_{p,i}$, and $D_{tt} V_{p,i}$ at $\tau = 0$, and through $V_{c,i+1}$, $D_t V_{p,i+1}$, and $D_{tt} V_{p,i+1}$ at $\tau = \Delta t$. (The availability of $V_{c,i+1}$ occurs because we have, at this point, already updated the field variables.) Applying these constraints allows us to evaluate the coefficients in the interpolating polynomial

$$g_c(\tau) = g_0 + g_1\left(\tau - \frac{\Delta t}{2}\right) + g_2\left(\tau - \frac{\Delta t}{2}\right)^2 + g_3\left(\tau - \frac{\Delta t}{2}\right)^3 + g_4\left(\tau - \frac{\Delta t}{2}\right)^4 + g_5\left(\tau - \frac{\Delta t}{2}\right)^5. \tag{B7}$$

Evaluating the integral in Equation (B6), the g_1, g_3, and g_5 terms are zero due to the choice of time symmetry, and so the correction stage for particle positions is

$$x_{c,i+1} = x_{c,i} + \frac{1}{2}(V_{c,i} + V_{c,i+1})\Delta t + \frac{1}{10}(D_t V_{p,i} - D_t V_{p,i+1})\Delta t^2 + \frac{1}{120}(D_{tt} V_{p,i} + D_{tt} V_{p,i+1})\Delta t^3. \tag{B8}$$

It can be useful to split the integration in to a background flow and perturbations, for example in computing the dynamics of a steady-state cylindrical flow. In this case we define the perturbation velocity field as $\boldsymbol{V}' = \boldsymbol{V} - \Omega r\hat{\phi}$ where r is the two-dimensional radius from the center of the cylinder, and $\Omega(r)$ is the angular velocity of the background flow. We denote the components of the circular radius to the point $(x_{c,i}, y_{c,i})$ as $r_{c,i,x}$, $r_{c,i,y}$, and the angular velocity of the background flow at this radius is Ω_p. Then, the predictor step with the background flow separated from the perturbation velocity $\boldsymbol{V}'$ is

$$x_{p,i+1} = r_{c,i,x}\cos(\Omega_p \Delta t) - r_{c,i,y}\sin(\Omega_p \Delta t) + V'_{c,i,x}\Delta t + \frac{1}{2} D_t V'_{p,i,x}\Delta t^2 + \frac{1}{6} D_{tt} V'_{p,i,x}\Delta t^3 \tag{B9}$$

$$y_{p,i+1} = r_{c,i,y}\cos(\Omega_p\Delta t) + r_{c,i,x}\sin(\Omega_p\Delta t) + V'_{c,i,y}\Delta t + \frac{1}{2}D_t V'_{p,i,y}\Delta t^2 + \frac{1}{6}D_{tt}V'_{p,i,y}\Delta t^3. \quad \text{(B10)}$$

We then add the background flow state back into the field variables used in Phurbas to calculate time derivatives for all fields in the usual inertial reference frame with $\boldsymbol{V}$ not $\boldsymbol{V}'$. Then, to transform the time derivatives to time derivatives of the perturbation velocity we use

$$D_t V'_{p,i+1,x} = D_t V_{p,i+1,x} + \Omega_p^2 r_{p,i+1,x} \quad \text{(B11)}$$

$$D_t V'_{p,i+1,y} = D_t V_{p,i+1,y} + \Omega_p^2 r_{p,i+1,y} \quad \text{(B12)}$$

$$D_{tt} V'_{p,i+1,x} = D_{tt} V_{p,i+1,x} - \Omega_p^3 r_{p,i+1,y} \quad \text{(B13)}$$

$$D_{tt} V'_{p,i+1,x} = D_{tt} V_{p,i+1,x} + \Omega_p^3 r_{p,i+1,x}. \quad \text{(B14)}$$

The perturbation velocity $\boldsymbol{V}$ can be integrated directly using these perturbation time derivatives. Using the normal corrector for the field variables and the perturbation velocity, the corrector step for position is then

$$x_{c,i+1} = r_{c,i,x}\cos(\Omega_p\Delta t) - r_{c,i,y}\sin(\Omega_p\Delta t) + \frac{1}{2}(V'_{c,i,x}+V'_{c,i+1,x})\Delta t + \frac{1}{10}(D_t V'_{p,i,x} - D_t V'_{p,i+1,x})\Delta t^2 + \frac{1}{120}(D_{tt}V'_{p,i,x} + D_{tt}V'_{p,i+1,x})\Delta t^3 \quad \text{(B15)}$$

$$y_{c,i+1} = r_{c,i,y}\cos(\Omega_p\Delta t) + r_{c,i,x}\sin(\Omega_p\Delta t) + \frac{1}{2}(V'_{c,i,y}+V'_{c,i+1,y})\Delta t + \frac{1}{10}(D_t V'_{p,i,y} - D_t V'_{p,i+1,y})\Delta t^2 + \frac{1}{120}(D_{tt}V'_{p,i,y} + D_{tt}V'_{p,i+1,y})\Delta t^3. \quad \text{(B16)}$$

To prepare for the next step, it is necessary to shift the perturbation velocity and time derivatives into the correct frame that will be used for the next step. The new frame at the corrected position $(x_{c,i+1}, y_{c,i+1})$ has radius components $(r_{c,i+1,x}, r_{c,i+1,y})$ and the angular velocity of the background flow at this radius is Ω_c. The transformations made to these quantities are

$$V'_{c,i+1,x} \to V'_{c,i+1,x} - r_{p,i+1,y}\Omega_p + r_{c,i+1,y}\Omega_c \quad \text{(B17)}$$

$$V'_{c,i+1,y} \to V'_{c,i+1,y} + r_{p,i+1,x}\Omega_p - r_{c,i+1,x}\Omega_c \quad \text{(B18)}$$

$$D_t V'_{c,i+1,x} \to D_t V'_{c,i+1,x} - r_{p,i+1,x}\Omega_p^2 + r_{c,i+1,x}\Omega_c^2 \quad \text{(B19)}$$

$$D_t V'_{c,i+1,y} \to D_t V'_{c,i+1,y} - r_{p,i+1,y}\Omega_p^2 + r_{c,i+1,y}\Omega_c^2 \quad \text{(B20)}$$

$$D_{tt} V'_{c,i+1,x} \to D_{tt} V'_{c,i+1,x} + r_{p,i+1,y}\Omega_p^3 - r_{c,i+1,y}\Omega_c^3 \quad \text{(B21)}$$

$$D_{tt} V'_{c,i+1,y} \to D_{tt} V'_{c,i+1,y} - r_{p,i+1,x}\Omega_p^3 + r_{c,i+1,x}\Omega_c^3. \quad \text{(B22)}$$

This shifts $\boldsymbol{V}'$ into the accelerating reference frame used for the following predictor step.

APPENDIX C

SECOND TIME DERIVATIVES OF MHD EQUATIONS

For the second-order predictor-corrector scheme Section 3.3 we need both the first and second Lagrangian time derivatives of the MHD Equations (1)–(4). This appendix gives the formulae for the required second time derivatives.

The form for a Lagrangian time derivative D_t in terms of partial derivatives ∂ is

$$D_t\partial q = \partial_t\partial q + V\cdot\nabla q \quad \text{(C1)}$$

$$= \partial_t\partial q + \partial[V\cdot\nabla q] - \partial V_i\partial_i q \quad \text{(C2)}$$

$$= \partial_t\partial_j q + \partial_j[V_i\partial_i q] - (\partial_j V_i)(\partial_i q) \quad \text{(C3)}$$

$$= \partial D_t q - (\partial V_i)(\partial_i q). \quad \text{(C4)}$$

Applying this to the MHD Equations (1)–(4) gives the second time derivatives needed for the second-order predictor-corrector scheme. We start with the velocity equation (Equation (1)). Taking its Lagrangian time derivative, the first term on the right-hand side becomes

$$D_t(-\rho^{-1}\partial_j P) = -\rho^{-1}(\partial_j D_t P - \partial_j V_a\partial_a P) + \rho^{-2}(D_t\rho)\partial_j P. \quad \text{(C5)}$$

Inserting this, the second derivative of velocity is

$$D_{tt}V_j = -\rho^{-1}(\partial_j D_t P - \partial_j V_i\partial_i P) + \rho^{-2}(\partial_j P)D_t P + D_t(\rho^{-1}(\varepsilon_{jab}\varepsilon_{acd}(\partial_c B_d)B_b)). \quad \text{(C6)}$$

This equation can be further reduced. The two pressure-dependent terms depend on the equation of state. For a gamma-law equation of state, $P = (\gamma - 1)\sigma$,

$$D_t P = (\gamma-1)D_t\sigma = (\gamma-1)(-(\sigma+P)\partial_i V_i), \quad \text{(C7)}$$

and so

$$\partial_j D_t P = (\gamma-1)(-(\partial_j\sigma + \partial_j P)\partial_a V_a - (\sigma+P)\partial_{aj}V_a). \quad \text{(C8)}$$

For an isothermal equation of state $P = c_s^2\rho$,

$$D_t P = c_s^2 D_t\rho = c_s^2(-\rho\partial_i V_i), \quad \text{(C9)}$$

and so

$$\partial_j D_t P = c_s^2(-\partial_j\rho\partial_a V_a - \rho\partial_{aj}V_a). \quad \text{(C10)}$$

The magnetic term reduces to

$$D_t(\rho^{-1}(\varepsilon_{jab}\varepsilon_{acd}(\partial_c B_d)B_b)) = (\varepsilon_{jab}\varepsilon_{acd}(\partial_c B_d)B_b)(-\rho^{-2}D_t\rho) + \rho^{-1}(\varepsilon_{jab}\varepsilon_{acd}(((\partial_c B_e)\partial_e V_d + B_e\partial_{ce}V_d - (\partial_c B_d)\partial_e V_e - B_d\partial_{ce}V_e - (\partial_c V_e)\partial_e B_d)B_b + (\partial_c B_d)D_t B_b)), \quad \text{(C11)}$$

while the Lagrangian time derivative of the gravitational force

$$D_t(+G_j) = \partial_t G_j + V_i\partial_i G_j. \quad \text{(C12)}$$

Taking the Lagrangian time derivative of the induction equation (Equation (2)) gives

$$D_{tt}B_j = (D_tB_i)\partial_iV_j + B_i(\partial_iD_tV_j - \partial_iV_k\partial_kV_j) - (D_tB_j)\partial_iV_i - B_j(\partial_iD_tV_i - \partial_iV_k\partial_kV_i). \quad (C13)$$

The Lagrangian time derivative of the internal energy equation is

$$D_{tt}\sigma = -(\sigma + P)(\partial_iD_tV_i - \partial_iV_j\partial_jV_i) - (D_t\sigma + D_tP)\partial_iV_i, \quad (C14)$$

where the required pressure-dependent expressions have already appeared above. If we have a barotropic or isothermal equation of state then this equation is not used, of course. Finally, the Lagrangian derivative of the continuity equation (Equation (4)) is

$$D_{tt}\rho = -(D_t\rho)(\partial_iV_i) - \rho(\partial_iD_tV_i - \partial_iV_j\partial_jV_i). \quad (C15)$$

The second term on the right-hand side can be expanded as

$$\partial_iD_tV_i = \varepsilon_{iab}\varepsilon_{acd}(-\rho^{-2}(\partial_cB_d)B_b\partial_i\rho + \rho^{-1}(\partial_{ic}B_d)B_b + \partial_cB_d\partial_iB_b) + \rho^{-2}\partial_iP\partial_i\rho - \rho^{-1}\partial_{ii}P + \partial_iG_i. \quad (C16)$$

REFERENCES

Abel, T. 2011, MNRAS, 413, 271
Barter, G. E., & Darmofal, D. L. 2010, J. Comput. Phys., 229, 1810
Bauer, A., & Springel, V. 2011, arXiv:1109.4413
Belytschko, T., Krongauz, Y., Organ, D., Fleming, M, & Krysl, P 1996, Comput. Methods Appl. Mech. Eng., 139, 3
Berger, M. J., & Oliger, J. 1984, J. Comput. Phys., 53, 484
Børve, S., Omang, M., & Trulsen, J. 2001, ApJ, 561, 82
Børve, S., Omang, M., & Trulsen, J. 2006, ApJ, 652, 1306
Brandenburg, A. 2010, MNRAS, 401, 347
Brandenburg, A., & Dobler, W. 2002, Comput. Phys. Commun., 147, 471
Chiang, E. 2008, ApJ, 675, 1549
Dedner, A., Kemm, F., Kröner, D., et al. 2002, J. Comput. Phys., 175, 645
Dellar, P. J. 2001, J. Comput. Phys., 172, 392
Dilts, G. 1999, Int. J. Numer. Methods Eng., 44, 1115
Dilts, G. 2000, Int. J. Numer. Methods Eng., 48, 1503
Dolag, K., & Stasyszyn, F. 2009, MNRAS, 398, 1678
Duffell, P. C., & MacFadyen, A. I. 2011, ApJS, 197, 15
Fiedler, R. A., & Mouschovias, T. C. 1992, ApJ, 391, 199
Fries, T. P., & Matthies, H. G. 2004, Classification and Overview of Meshfree Methods, Tech. Rep. Informatikbericht Nr.: 2003-3, Technical University Braunschweig
Fryxell, B., Olson, K., Ricker, P., et al. 2000, ApJS, 131, 273
Gaburov, E., Johansen, A., & Levin, Y. 2012, ApJ, submitted (arXiv:1201.4873)
Gaburov, E., & Nitadori, K. 2011, MNRAS, 414, 129
Gingold, R. A., & Monaghan, J. J. 1977, MNRAS, 181, 375
Gnedin, N. Y. 1995, ApJS, 97, 231
Hayes, J. C., Norman, M. L., Fiedler, R. A., et al. 2006, ApJS, 165, 188
Inutsuka, S.-I. 2002, J. Comput. Phys., 179, 238
Iwasaki, K., & Inutsuka, S.-I. 2011, MNRAS, 418, 1668
Janhunen, P. 2000, J. Comput. Phys., 160, 649
Johansen, A., Youdin, A., & Klahr, H. 2009, ApJ, 697, 1269
Kitsionas, S., & Whitworth, A. P. 2002, MNRAS, 330, 129
Kuhnert, J. 1999, PhD thesis, Fachbereich Mathematik, Univ. Kaiserslautern
Kuhnert, J. 2002, in Meshfree Methods for Partial Differential Equations, ed. M. Griebel & M. A. Schweitzer (Lecture Notes in Computational Science and Engineering, Vol. 26; Berlin: Springer), 239
Lancaster, P., & Salkauskas, K. 1981, Math. Comput., 37, 141
Liszka, T., Duarte, C., & Tworzydlo, W. 1996, Comput. Methods Appl. Mech. Eng., 139, 263
Liszka, T., & Orkisz, J. 1980, Comput. Struct., 11, 83
Liu, M. B., Liu, G. R., & Lam, K. Y. 2003, J. Comput. Appl. Math., 155, 263
Liu, W. K., Jun, S., & Zhang, Y. F. 1995, Int. J. Numer. Methods Fluids, 20, 1081
Liu, M. B., Xie, W. P., & Liu, G. R. 2005, Appl. Math. Modelling, 29, 1252
Lucy, L. B. 1977, AJ, 82, 1013
Maron, J., & Mac Low, M.-M. 2009, ApJS, 182, 468
Maron, J. L., & Howes, G. G. 2003, ApJ, 595, 564
Maron, J. L., Mac Low, M.-M., & Oishi, J. S. 2008, ApJ, 677, 520
Masset, F. 2000, A&AS, 141, 165
McNally, C. P., Maron, J., & Mac Low, M.-M. 2012, ApJS, 200, 7 (Paper II)
Morris, J. P. 1996, PASA, 13, 97
Morris, J. P., & Monaghan, J. J. 1997, J. Comput. Phys., 136, 41
Nitadori, K., & Makino, J. 2008, New Astron., 13, 498
Norman, M. L., Wilson, J. R., & Barton, R. T. 1980, ApJ, 239, 968
Oñate, E., Idelson, S., Zienkiewicz, O. C., & Taylor, R. L. 1996, Int. J. Numer. Methods Eng., 39, 3839
Owen, J. M., Villumsen, J. V., Shapiro, P. R., & Martel, H. 1998, ApJS, 116, 155
Pakmor, R., Bauer, A., & Springel, V. 2011, MNRAS, 418, 1392
Parshikov, A. N., Medin, S. A., Loukashenko, I. L., & Milekin, V. A. 2000, Int. J. Impact Eng., 24, 779
Pen, U.-L. 1998, ApJS, 115, 19
Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. 1992, Numerical Recipes in FORTRAN. The Art of Scientific Computing (2nd ed.; Cambridge: Cambridge Univ. Press)
Price, D. J. 2008, J. Comput. Phys., 227, 10040
Price, D. J. 2010, MNRAS, 401, 1475
Price, D. J. 2012a, MNRAS, 420, L33
Price, D. J. 2012b, J. Comput. Phys., 231, 759
Price, D. J., & Monaghan, J. J. 2004a, MNRAS, 348, 123
Price, D. J., & Monaghan, J. J. 2004b, MNRAS, 348, 139
Price, D. J., & Monaghan, J. J. 2005, MNRAS, 364, 384
Quinlan, N. J., Basa, M., & Lastiwka, M. 2006, Int. J. Numer. Methods Eng., 66, 2064
Robertson, B. E., Kravtsov, A. V., Gnedin, N. Y., Abel, T., & Rudd, D. H. 2010, MNRAS, 401, 2463
Rosswog, S. 2009, New Astron. Rev., 53, 78
Rosswog, S., & Price, D. 2007, MNRAS, 379, 915
Shapiro, P. R., Martel, H., Villumsen, J. V., & Owen, J. M. 1996, ApJS, 103, 269
Springel, V. 2010, MNRAS, 401, 791
Steinmetz, M., & Mueller, E. 1993, A&A, 268, 391
Stone, J. M., & Gardiner, T. A. 2010, ApJS, 189, 142
Stone, J. M., Gardiner, T. A., Teuben, P., Hawley, J. F., & Simon, J. B. 2008, ApJS, 178, 137