Journal of Computational Physics **171,** 151–181 (2001)
doi:10.1006/jcph.2001.6780, available online at http://www.idealibrary.com on IDEAL®

A Staggered Grid, Lagrangian–Eulerian Remap Code for 3-D MHD Simulations

T. D. Arber,[1] A. W. Longbottom, C. L. Gerrard, and A. M. Milne

Mathematical Institute, University of St. Andrews, St. Andrews, Fife KY16 9SS, Scotland, United Kingdom
E-mail: tda@astro.warwick.ac.uk

Received August 23, 2000; revised March 7, 2001

In this paper an approach to multidimensional magnetohydrodynamics (MHD) which correctly handles shocks but does not use an approximate Riemann solver is proposed. This approach is simple and is based on control volume averaging with a staggered grid. The method builds on the older and often overlooked technique of on each step taking a fully 3-D Lagrangian step and then conservatively remapping onto the original grid. At the remap step gradient limiters are applied so that the scheme is monotonicity-preserving. For Euler's equations this technique, combined with an appropriately staggered grid and Wilkins artificial viscosity, can give results comparable to those from approximate Riemann solvers. We show how this can be extended to include a magnetic field, maintaining the divergence-free condition and pressure positivity and then present numerical test results. Where possible a comparison with other shock capturing techniques is presented and the advantages and disadvantages of the proposed scheme are clearly explained. © 2001 Academic Press

Key Words: MHD; shocks; Lagrangian.

1. INTRODUCTION

Over the past 20 years approximate-Riemann-solver-based algorithms for solving hyperbolic systems of equations have proven both simple and robust. In MHD studies there is a growing reference list of multidimensional Riemann-solver-based numerical schemes. An early example by Brio and Wu [1] demonstrated the power of such a scheme for 1D problems and later such techniques were shown to extend well to TVD schemes [2]. In more than one dimension approximate Riemann solvers have been devised which maintain the magnetic field divergence-free condition [3, 4]. This list is by no means complete and other examples can be found in [1–3]. With this background it may seem unusual to be developing new schemes which are not based on an approximate Riemann solver and all of the benefits

[1] Current address: Department of Physics, University of Warwick, Coventry CV4 7AL, UK.

that such an approach brings. This introduction explains the background problems which led us to reconsider Lagrangian remap codes and sets the scene for later sections which discuss in detail the choice of algorithm, its testing, its advantages and disadvantages.

The primary motivating factor in writing this code was that it should be easily adaptable to a variety of problems in solar coronal physics. While having broad applicability (see later tests), it is in this niche that the benefits of the current approach are most clear. The corona is dominated by its magnetic field but observed optically. The corona is also highly dynamic with current sheets and MHD shocks. For comparison with observations simulations must therefore give accurate temperature predictions but be capable of handling shocks. This presents a problem which applies to any method which solves the equations in conservative form, including Riemann-solver-based schemes. By solving for the total energy such approaches conserve energy to machine precision. This is of course a good thing on its own. However, this does mean that the pressure must be found by subtracting the magnetic and kinetic energy from the total energy. In the low-beta plasma of the solar corona the magnetic energy density is typically 10^2 to 10^3 times larger than the thermal energy density. For a plasma beta, defined as the ratio of thermal pressure to magnetic pressure, of 10^{-3} this means that a 0.1% error in the magnetic field leads to a 200% error in calculating pressure and temperature. While one could argue that for such low betas the plasma pressure has little effect on the evolution this does not help if, as is often the case, it is an accurate estimate of the temperature which one wants. This becomes particularly important if one needs to find ionization levels or thermal conductivities which are sensitive to the local temperature. There is a pragmatic solution to this problem proposed by Balsara and Spicer [5] but unfortunately this is ignored in most Riemann codes. This involves using an additional equation to update the pressure. The error in the temperature is then determined by the scheme's truncation error and not the difference of two large terms. Balsara and Spicer [5] use flags to determine when to use the pressure from the extra equation or that calculated from the conservative form. The approach adopted in Lare3d can therefore be viewed as not enforcing exact energy conservation, although the error is of course still bounded and convergent with other finite difference errors, in order to get an accurate and physical prediction for the temperature.

Any approach which does not depend on a characteristic decomposition of the equations also benefits from freedom in the choice of equation of state (EOS). While arbitrary equations of state can be accommodated in Riemann-solver-based schemes, the success of such approaches for table lookup EOS (for example, the Los Alamos SESAME database [6]) where the decomposition cannot be achieved analytically is not clear. However, this is a simple procedure to adopt in the Lagrangian remap scheme described in this paper. Non-hyperbolic terms, such as viscosity and resistivity, can be included in most schemes without too much difficulty and in the present scheme such terms are simply added into the Lagrangian step; the remap is purely geometrical and contains no physics. If large viscosity or resistivity are included then a more efficient approach would be to use simple finite differences as in these cases any shocks would be diffuse anyway. In the solar corona and many astrophysical plasmas, the viscosity and resistivity are so low that the aim is to include the minimum possible diffusion. In such circumstances shock-capturing techniques of some sort are always needed as the gradients present will still cause Gibbs's overshoot with finite difference schemes. Indeed, there is a strong case for always including artificial viscosity, i.e., viscosity which is only present at shocks and goes to zero as the resolution is increased in smooth regions, to avoid known numerical problems with upwind-based shock schemes (see [7] for details). Such an approach has been adopted in MHD Riemann codes (see [8], for example) but such schemes will still have difficulty with low-beta simulations if accurate estimates of temperature are required. Some additional terms, such as the Hall term in Ohm's law, change the characteristics of the problem and their inclusion is cumbersome in a Riemann code but straightforward in codes such as presented here or other shock schemes which do not rely on characteristic decomposition, e.g., TVD Lax–Friedrichs as in [9]. A further advantage of the current scheme is the ease with which it can be adapted to include multimaterial interface tracking as described in [10].

The problem we address in this paper is therefore to devise a scheme that can handle shocks in ideal MHD but is not tied to a Riemann solver or conservative form. The hope is then to have a core code which can easily accommodate extra physics without sacrificing shock resolution or energy conservation. In attempting to do this we have returned to the scheme originally devised by van Leer [11]. This was based on taking a Lagrangian step and then conservatively remapping back onto the original Eulerian grid. Limiters were applied at the remap step to ensure monotonicity and the Lagrangian step used a Riemann solver. Following this paper most research was directed at schemes which abandoned the Lagrangian step and concentrated on the approximate Riemann solver. Here we take the opposite approach and keep the Lagrangian remap scheme but abandon the Riemann solver. For Euler solvers this is not a new idea. Indeed the earliest examples, such as used by Noh in the CEL code [12], predate van Leer's paper. A complete history and bibliography of such work can be found in Benson's review article [13]. One other noteworthy paper which discusses Lagrangian remap schemes must also be mentioned. This is the review article by Woodward and Collela [14] which used the BBC Lagrangian remap code in a set of comparison tests. In these tests the BBC code performed rather badly, actually giving worse results than FCT schemes. This has somewhat dented the credibility of Lagrangian remap schemes. We will show later in this paper that the algorithm adopted in this paper does in fact produce results comparable to those of Riemann-based solutions and the poor results in Ref. [14] stem from that particular Lagrangian remap scheme.

Where this paper does introduce new computational algorithms is in the inclusion of the magnetic field **B**, viscosity, and resistivity. While none of these is in itself radical it is the first time that all of these effects have been included in such a code. The magnetic field is defined on a staggered grid and uses Evans and Hawley's constrained transport model [15] to keep the divergence of the **B** zero to within machine precision. A staggered grid is not actually essential for keeping $\nabla \cdot \mathbf{B} = 0$, as was demonstrated by Peterkin *et al.* [16], but the staggering greatly simplifies the spatial centering of the scheme, as will become clear later. We should also point out that the paper by Peterkin *et al.* [16] is also a Lagrangian remap code but did not include viscous or resistive effects and was not tested on MHD shock problems. Artificial viscosity is based on the form of viscosity presented by Wilkins [10]. Resistivity is split into two parts: an artificial resistivity similar to the artificial viscosity and a user-specified resistive term.

The scheme outlined in this paper, i.e., a Lagrangian step followed by a Van Leer limited remap, has great similarity to the arbitrary Lagrangian–Eulerian (ALE) codes. Indeed the present scheme can be viewed as a sort of ALE scheme. It is therefore important to assess the current scheme against the background of published ALE codes. In ALE codes one splits the time update into a Lagrangian step, as here, and then a remap onto a new grid before the next time step. Actually ALE codes do not have to remap every step and can leave remapping until the Lagrangian grid becomes too distorted or some other criterion is satisfied. Note, however, that this remap is not necessarily back to the original grid, as is the case here, but to any arbitrary grid. In this context the current scheme is an ALE scheme in which the remap is not to a general grid but to the original Eulerian grid. Details of ALE codes for fluid dynamics can be found in Hirt [17] and Brackbill and Pracht [18]. The greatest difference between the current scheme and these references is in the nature of the remap. Here this is done using van Leer piecewise linear reconstructed data. This is vital if code is to accurately resolve shocks. For example, the shock problem in Fig. 4a of [17] is a weak shock with 60 zones in total and approximately 10 zones across the shock. This compares with the current scheme which resolves the shock with just three zones (see Fig. 2 of this paper). In 3D, ALE codes which rely only on viscosity to resolve shocks would require around 3^3 times more grid points to achieve the accuracy of a shock-capturing scheme. This remains true for ALE codes extended to MHD such as that in Ref. [19]. The strength of ALE schemes lies in their ability to adapt the grid without the defects of full Lagrangian codes, not in their treatment of shocks. The reverse is true for the current scheme. Although the scheme in this paper is a variant of an ALE code our desire to test the shock capturing properties of our scheme means that comparison with more conventional ALE schemes would be inappropriate as these were not designed with shock treatment as the primary goal. Instead we compare results with codes which also directly attempt to handle shocks, such as approximate Riemann solvers, as this is a more fair comparison of like with like.

Section 2 contains a description of the code. Test results from the code are then presented in Section 3. We have chosen a set of tests which are well established in the literature and in particular have chosen the same initialization used in Ref. [4], as this paper contains the largest set of quantified tests. The conclusions and final comments are in Section 4 with details of the finite difference equations in Appendix A and less common aspects of the code moved to Appendix B and C.

2. THE `Lare3d` CODE

The full version of this code solves the resistive MHD equations in 3D Cartesian geometry. The code is freely available to anyone by contacting the authors by email. In order to simplify the description of each of the steps in the code this section is split into subsections each of which deals with one of the main features. An explicit description of the finite difference equations can be found in Appendix A.

2.1. *The Model Equations*

Here we introduce the normalized MHD equations in the form they are used in the code. For reference we also introduce the terminology used later in the paper when discussing Riemann problems in ideal MHD. Adopting standard normalization the equations of resistive MHD can be written in Lagrangian form as

$$\frac{D\rho}{Dt} = -\rho\nabla\cdot\mathbf{v}, \tag{1}$$

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho}(\nabla\times\mathbf{B})\times\mathbf{B} - \frac{1}{\rho}\nabla P, \tag{2}$$

$$\frac{D\mathbf{B}}{Dt} = (\mathbf{B}\cdot\nabla)\mathbf{v} - \mathbf{B}(\nabla\cdot\mathbf{v}) - \nabla\times(\eta\nabla\times\mathbf{B}), \tag{3}$$

$$\frac{D\varepsilon}{Dt} = -\frac{P}{\rho}\nabla\cdot\mathbf{v} + \frac{\eta}{\rho}\,j^2, \tag{4}$$

where $j = \nabla\times\mathbf{B}$ is the current density, v is the velocity, P is the thermal pressure, $\varepsilon = P/\rho(\gamma - 1)$ is the internal energy density ($\gamma = 5/3$ is the specific heat ratio), ρ is the mass density and η is the resistivity. These equations are also supplemented by the condition $\nabla\cdot\mathbf{B} = 0$ which is an initial constraint for the differential equations but must be enforced explicitly in any numerical scheme. A key dimensionless parameter in MHD studies is the plasma beta defined as the ratio of thermal to magnetic pressure; i.e., $\beta = P/B^2$. Viscous effects will be discussed in a later section.

One of the advantages of the current approach is that the Lagrangian step is fully three-dimensional; i.e., there is no Strang splitting imposed on this step. This is particularly important for updating ρ, as Eq. (1) is not used in practice. Instead, the density change is related directly to volume changes using mass conservation. In particular, if a plasma fluid element is initially at a point $\mathbf{X} = (X_1, X_2, X_3)$ and moves to a point $\mathbf{x} = (x_1, x_2, x_3)$ then this new point $\mathbf{x}$ is a function of the old position $\mathbf{X}$ and of time. This implies that the change in element length is given by

$$dx_i = \frac{\partial x_i}{\partial X_\alpha}\,dX_\alpha, \tag{5}$$

with summation convention on α. We then have that

$$\rho = \frac{\rho_0}{\Delta}, \tag{6}$$

where ρ_0 is the original density and Δ is the determinant of the Jacobian transformation matrix,

$$\Delta = \frac{\partial(x_1, x_2, x_3)}{\partial(X_1, X_2, X_3)} = \begin{vmatrix} \frac{\partial x_1}{\partial X_1} & \frac{\partial x_2}{\partial X_1} & \frac{\partial x_3}{\partial X_1} \\ \frac{\partial x_1}{\partial X_2} & \frac{\partial x_2}{\partial X_2} & \frac{\partial x_3}{\partial X_2} \\ \frac{\partial x_1}{\partial X_3} & \frac{\partial x_2}{\partial X_3} & \frac{\partial x_3}{\partial X_3} \end{vmatrix}. \tag{7}$$

When dealing with control volumes using Eq. (6) Δ is the ratio of the final volume to the initial volume. Δ appears in the control volume averages of most quantities and in this code is evaluated as

$$\Delta = 1 + (\nabla\cdot\mathbf{v})\,dt, \tag{8}$$

where the divergence is evaluated on the Eulerian grid. This is second order accurate since if we consider the change of position,

$$x_1(X, t) = X_1 + \int_0^{dt} v_x(X_1, X_2, X_3, t)\,dt. \tag{9}$$

Taking a Taylor series for v_x about the original position X and then expanding any remaining time dependence about $t = 0$ gives

$$x_1 = X_1 + v_x dt + \frac{\partial v_x}{\partial X_1}v_x dt^2 + \frac{\partial v_x}{\partial X_2}v_y dt^2 + \frac{\partial v_x}{\partial X_3}v_z dt^2 \tag{10}$$

with x_2 and x_3 similarly defined. Taking partial derivatives of these with respect to (X_1, X_2, X_3) allows us to expand the Jacobian matrix with each term accurate to second order,

$$\Delta = \begin{vmatrix} 1 + \frac{\partial v_x}{\partial X_1}\, dt & \frac{\partial v_y}{\partial X_1}\, dt & \frac{\partial v_z}{\partial X_1}\, dt \\ \frac{\partial v_x}{\partial X_2}\, dt & 1 + \frac{\partial v_y}{\partial X_2}\, dt & \frac{\partial v_z}{\partial X_2}\, dt \\ \frac{\partial v_x}{\partial X_3}\, dt & \frac{\partial v_y}{\partial X_3}\, dt & 1 + \frac{\partial v_z}{\partial X_3}\, dt \end{vmatrix}. \tag{11}$$

This gives

$$\Delta = 1 + \left(\frac{\partial v_x}{\partial X_1} + \frac{\partial v_y}{\partial X_2} + \frac{\partial v_z}{\partial X_3} \right) dt + O(dt^2), \tag{12}$$

$$\Delta = 1 + (\nabla \cdot \mathbf{v})\, dt + O(dt^2); \tag{13}$$

hence Eq. (8) is second order accurate and Eq. (6) is used in preference to a finite difference representation of Eq. (1). Relating density changes to volume changes and using those volume changes consistently elsewhere in the code also guarantees exact conservation of mass.

The update procedure outlined later requires the Lagrangian equations for the control volume averaged **B** field and the flux. These are derived from Eq. (3) and are

$$\frac{D}{Dt} \int B_i \, d\tau = \int v_i \, \mathbf{B} \cdot \mathbf{ds} - \int [\nabla \times (\eta \nabla \times \mathbf{B})]_i \, d\tau, \tag{14}$$

$$\frac{D}{Dt} \int \mathbf{B} \cdot \mathbf{ds} = - \int \eta \mathbf{j} \cdot \mathbf{dl}, \tag{15}$$

where integrals in Eq. (14) over τ and **ds** refer to integrals over the volume of a control volume and its surface. Integrals in Eq. (15) over **dl** refer to line integrals around the surface integrated over in the **ds** integral.

While the above equations are the complete set needed to explain the algorithms used in `Lare3d` here we also present some standard equations from approximated Riemann solver theory. This is simply to give a convenient reference point for later discussions. To keep the presentation brief we simply use a 1D version of a non-TVD limited scheme. Details can be found in, for example, Ref. [2]. Representing the ideal MHD equations, i.e., Eqs. (1)–(4), with $\eta = 0$, in conservative form we have

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}(\mathbf{U})}{\partial x} = 0. \tag{16}$$

Then updating the cell-averaged $\mathbf{U}_i$ is achieved by

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \frac{\Delta t}{\Delta x} \left(\mathbf{F}_{i+1/2} - \mathbf{F}_{i-1/2} \right), \tag{17}$$

$$\mathbf{F}_{i+1/2} = \frac{1}{2} [\mathbf{F}(\mathbf{U_L}) + \mathbf{F}(\mathbf{U_R})] - \frac{1}{2} \sum \alpha_k |\lambda_k| \mathbf{r}_k, \tag{18}$$

where $\mathbf{r}_k$ are right eigenvectors of the Jacobian matrix $\frac{\partial \mathbf{F}}{\partial \mathbf{U}}$, λ_k are the eigenvectors, and α_k are the coefficients in the expansion

$$\mathbf{U_R} - \mathbf{U_L} = \sum_k \alpha_k \mathbf{r}_k. \tag{19}$$

Throughout these equations $\mathbf{U_R}$ refers to the state on the right of the interface and $\mathbf{U_L}$ the state on the left.

2.2. *The Grid*

The variables are staggered on a computational cell as shown in Fig. 1. All scalars are defined at the cell volume center. **B**-field components are staggered onto cell faces so that $\nabla \cdot \mathbf{B} = 0$ can be maintained with the Evans and Hawley constrained transport [15]. All velocity components are defined at the cell vertex. The velocities must be staggered with respect to both **B** and the pressure to avoid checkerboard instabilities. This can be satisfied by defining the velocities at cell edges or the cell vertex. Defining all velocities at the same point leads to a single velocity control volume at the remap stage and a more compact code and is therefore the choice adopted here.

2.3. *The Lagrangian Step*

The Lagrangian step is a simple predictor–corrector scheme. Predicted values are found from an Euler step with timestep $dt/2$. Then conservation of mass in Lagrangian control volumes is used to simplify the time-centered Lagrangian source terms by evaluating derivatives on the original Eulerian grid (see Appendix B for details). The end result is a second order scheme, both in time and space, which is fully three-dimensional and does not use conservative form. There are two complications in this step: the update of the magnetic field and artificial viscosity.

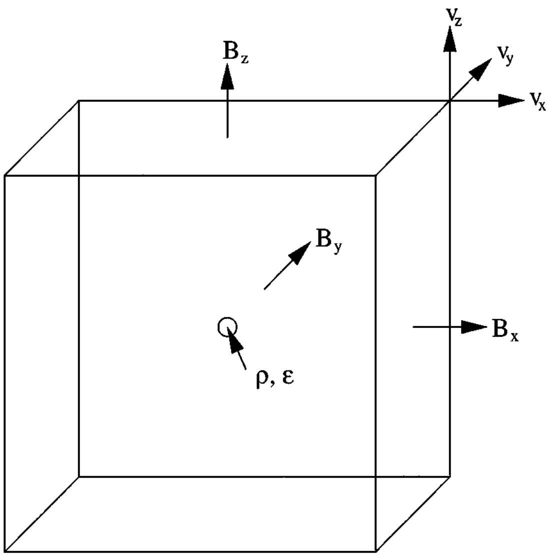

FIG. 1. The position of primary variables defined on a 3D computational cell.

Concentrating first on the **B** field update with $\eta = 0$ we note that Eqs. (14) and (15) are particularly simple in this case. Equation (15) is just Alfvén's theorem for nonresistive MHD; i.e., the magnetic flux through an area moving with the plasma is constant. Also, as the remap step (see Section 2.4) deals only with magnetic flux, all we need at the end of the Lagrangian step is the magnetic flux through the control volume faces. When $\eta = 0$ this is simply the flux at the start of the Lagrangian step and is trivially found by multiplying the **B** field components by the area of the appropriate Eulerian cell face. The corrector step update of Eq. (2) does require the time-centered magnetic field force $(\nabla \times \mathbf{B}) \times \mathbf{B} = -\nabla B^2/2 + (\mathbf{B} \cdot \nabla)\mathbf{B}$. Here it is desirable to have B^2, the magnetic pressure, defined at the same place as the thermal pressure, P. This is achieved easily by using Eq. (14) as an equation to update the control-volume-centered **B** field from the cell-face magnetic fluxes. Thus in the Lagrangian phase $\mathbf{B}^n$ is face centered, $\mathbf{B}^{n+1/2}$ is volume centered, and $\mathbf{B}^{n+1}$ is not needed at all.

The effectiveness of Lagrangian steps with artificial viscosity can be most clearly seen at the theoretical level by returning to Eq. (18). Final proof of the effectiveness will of course be presented later in the form of numerical tests. Equation (18) is capable of treating shocks due to its direct handling of discontinuities as the jumps across the characteristics of the linear problem. It is this feature which is intrinsically difficult to include as artificial viscosity in Eulerian-based finite difference codes. The great advantage of a Lagrangian step is that the fluxes in Eq. (18) are all zero except for the velocity. The terms $\frac{1}{2}\sum \alpha_k|\lambda_k|\mathbf{r}_k$ in Eq. (18) are then precisely of the same form as real viscous forces applied to the velocity update. Put simply, in a Lagrangian step all that one needs to update is the velocities, or equivalently the position of the cell vertices, and all other quantities are found from those using conservation of mass and magnetic flux. This is not true of artificial viscosity applied to an Eulerian code where the care in updating ρ which is implied by Eq. (18) is not guaranteed by the difference scheme. This is the main reason artificial viscosity applied in a Lagrangian remap code is more effective than artificial viscosity applied in an Eulerian scheme.

The actual form of the artificial viscosity used in these tests is taken from Wilkins [10]. This is particularly simple in that it is a scalar viscosity; i.e.,

$$q = c_1 \rho c_f L|s| + c_2 L^2 \rho s^2. \tag{20}$$

In this formula c_f is the local fast mode speed, not the sound speed as would be appropriate for simple gasdynamics. L and s are the grid length and the strain rate, both measured in the direction of the acceleration. This approach preserves shock structure moving obliquely across the grid. More complex forms of artificial viscosity do exist; see e.g., Ref. [20], with tensor viscosities. Our original intention had been to use this simple form just to get the code up and running before moving to more complex approaches. However, we found that for the tests presented here the simple Wilkins form was perfectly adequate and we have not "upgraded." Forms of artificial viscosity simpler than Wilkins, e.g., standard von Neumann viscosity applied in each direction separately, do, however, fail in multidimensional tests. The important features of Wilkins viscosity which must always be correctly included are that it is positive heating and does not apply viscosity tangential to a shock front. Throughout this paper we use $c_1 = 0.1$ and $c_2 = 1.0$ unless otherwise stated.

For MHD shock problems the above viscosity-based approach does handle shocks correctly. However, we have found that $c_1 \simeq 0.8$ is required to guarantee that there is no overshoot. This leads to solutions that are more diffusive and also reduces the timestep through the CFL condition. To avoid this we apply artificial resistivity through Eqs. (14) and (15). The above discussion has demonstrated that there are good theoretical reasons for artificial viscosity to be effective in a Lagrangian step. Furthermore, the specific form of viscosity, Eq. (20), can be justified as the limiting form of the gasdynamical jump conditions (see derivation of Kurapatenko in Ref. [10]). Unfortunately, there seems to be no such derivation for choices of artificial resistivity. Our attempts at such a derivation have also been unsuccessful and we have resorted to a pragmatic approach for the choice of resistivity. The conditions we imposed were that this resistivity must be positive heating and vanish as the resolution increases, except at discontinuities. From the forms we have tried the simplest prescription which satisfies these constraints is setting $\eta = v_A^2 \Delta t$, where v_A is the local Alfvén speed. Furthermore only the perpendicular current is used with this anomalous resistivity in Eqs. (14) and (15). In all of the test results presented later the ohmic heating quoted refers to the heating due to this form of resistivity. Equation (15) is updated in the same way as Eq. (3), which is another simple application of the Evans and Hawley constrained transport method of keeping $\nabla \cdot \mathbf{B} = 0$ [15].

2.4. *The Remap Step*

The remap of variables from the Lagrangian grid back onto the original Eulerian grid is done in one-dimensional sweeps. These are Strang-ordered and are the only part of the code which is not fully three-dimensional. While one would like a fully 3D code this procedure at least has the advantage of all of the physics being handled in a fully multidimensional way in the Lagrangian step. The remap step is a geometrical step designed to maintain monotonicity. At this stage it is possible to build two- or three-dimensional information into the gradient calculations but to date we have only implemented separate 1D sweeps. The remap is performed in each direction by simply following van Leer's original algorithm [11] using Lagrangian variables to remap ρ and then mass coordinates to remap $\mathbf{v}$ and ε. The magnetic flux is remapped using the standard $\nabla \cdot \mathbf{B}$-preserving scheme but is now applied to a remap as opposed to an Eulerian advection. The changes needed to convert flux advection into a remap are trivial. In all remaps we use the third order estimate of gradient suggested by Youngs [21] and limiting procedure Eq. (101) from [11]. The Lagrangian step is only second order accurate, so there is no need to use a third order accurate initial estimate of gradients in the remap step. However, this costs nothing computationally and we found that for simple 1D wave problems it slightly reduces the numerical dissipation. As a result we have retained the third order scheme because it costs nothing, does no harm, and occasionally helps in simple problems.

The scheme for remapping uses mass coordinates wherever possible. This conserves mass, internal energy, and momentum to machine precision. However, in this form it does not conserve kinetic energy. This is only significant at shocks where the limiters flatten gradients in the remap step. The loss of total energy at shocks can weaken them and therefore a procedure for correcting this is needed. Such techniques have been used before, although not in the form used here, and details and references can be found in Section 3.6 of Benson's review article [13]. The precise kinetic energy remap scheme used here is explained in Appendix C . Thus the combined Lagrangian remap step is energy-conserving for Euler's equations. The same technique applied in Appendix C could also be extended to magnetic energy but at present this has not been implemented. This is not a problem for the tests presented here and energy conservation will be reported later with each of the tests. Note that the properties of the scheme mean that these energy errors can always be identified as errors in magnetic field energy.

3. NUMERICAL TESTS

3.1. *1D Euler Equation Riemann Problems*

We begin this catalogue of numerical tests with two tests of the code solving Euler's equations only. These are both of historical interest and are included here to clarify a few points raised in the Introduction. Since the core idea of this code is to take the basic scheme of van Leer's original paper [11] and replace the Riemann solver with an artificial viscosity it is worth returning to the original Sod's problem used by van Leer as a test. Figure 2 shows the result for Sod's problem, having the same initial conditions, etc., as in Ref. [11], both with viscosity (solid line) and with zero viscosity (asterisks). Note that removing the Riemann solver from van Leer's algorithm and replacing it with a predictor-corrector finite difference scheme only introduces a small overshoot behind the shock. This is removed when viscosity is included. This demonstrates two main points: Sod's problem is not particularly demanding and the Lagrangian remap scheme is clearly far more robust at handling shocks than Eulerian finite difference schemes. ALE codes which do not use van Leer remapping, or an equivalent, are also less accurate at treating shocks (see Fig. 4a in Ref. [17]). This also demonstrates, albeit indirectly, the importance of using a limited gradient in the remap phase. It is this, along with the choice of artificial viscosity and staggered grid, which distinguishes `Lare3d` from conventional ALE schemes.

The next Euler equation solver test was for interacting blast waves. This was taken from Woodward and Collela's review article [14]. In this review tests were presented which showed an example of a Lagrangian remap code (BBC) performing rather badly. We repeat this test to show that this is not true of the `Lare3d` code and also to show that such interacting strong shocks present no difficulty. This 1-D problem is initialized in a $0 \le x \le 1$ domain with reflecting ends and $\rho = 1$, $v = 0$. The shocks are initialized by the pressure with $P = 1000$ for $x < 0.1$, $P = 100$ for $x > 0.9$ and $P = 0.01$ elsewhere. Figure 3 shows the

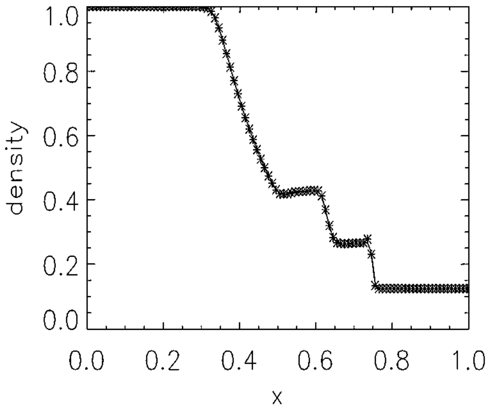

FIG. 2. Numerical solution to Sod's problem for comparison with results in van Leer [11]. The solid line is the result with viscosity the symbols are the result with zero viscosity.

TABLE I
Interacting Strong Shock Convergence Tests

N	Time steps	$\epsilon(4800)$	$\epsilon(9600)$
200	421	0.136	0.141
400	853	0.078	0.082
600	1283	0.053	0.058
1200	2576	0.026	0.029

density at $t = 0.038$ for the region $0.5 \leq x \leq 1$ for two resolutions (200 and 1200 zones) for comparison with the results in Ref. [14]. Assessing these results by eye shows that they are comparable to the results from the MUSCL scheme used in Ref. [14]. To quantify this we have calculated

$$\epsilon = \frac{1}{N}\sum_{i=1}^{N}\left|\rho_i - \rho_i^{acc}\right|, \tag{21}$$

where ρ_i^{acc} is an accurate answer. In Ref. [14] a special adaptive code was used to find the exact solution to use in Eq. (21). We do not have such a code and have therefore used the results from 4800 and 9600 zones as estimates for ρ_i^{acc} and in both cases have calculated ϵ for a range of grid sizes. These are presented in Table I. In this table $\epsilon(4800)$ is the evaluation of Eq. (21) using 4800 zones for ρ_i^{acc} and $\epsilon(9600)$ for the same with 9600 zones. All results were produced with a Courant number of 0.8 and `Lare3d` requires slightly more steps than equivalent codes in Ref. [14] due to the inclusion of the viscosity coefficients in the CFL condition.

Assuming that the `Lare3d` converges to the correct answer these figures show that the current Lagrangian remap scheme, with Wilkins artificial viscosity, gives results as good as those from the MUSCL scheme in Ref. [14] (the estimate of ϵ on a 1200 grid for MUSCL was 0.04 in Ref. [14]). The solid lines on Fig. 3 show overshoots at the end of rarefaction fans (in this case at $x = 0.59$ and $x = 0.76$) which means that the estimates in Table I will be out by a small amount and biased in favor of the current scheme. However, even allowing for this the results shown are better than those for the FCT schemes but not as good those for as the PPM-based algorithms. The better performance of the PPM schemes is not surprising as this is a 1D problem with strong rarefaction waves. We postpone consideration of execution speed compared to other schemes until later.

3.2. *1-D MHD Riemann Problems*

The most commonly tested 1D MHD Riemann problem was originally set up by Brio and Wu [1]. This shock tube test is repeated here with left state $(\rho, B_y, P) = (1, 1, 1)$, right state $(\rho, B_y, P) = (0.125, -1, 0.1)$, and $B_x = 0.75$. All other variables are initially zero and our solution differs from that in Ref. [1] in that we take $\gamma = 5/3$. The numerical solution at $t = 0.1$ with 800 cells for ρ, v_x, B_y and v_y is shown in Fig. 4. This allows direct comparison with the same test in Refs. [1, 2, 9].

What is clear from Fig. 4 is that `Lare3d` handles the MHD shocks as well as the TVD–MHD codes. Since energy is only conserved exactly for Euler problems (see Appendix B) there is an energy error in these results. At $t = 0.1$ energy is conserved to within 0.09% and

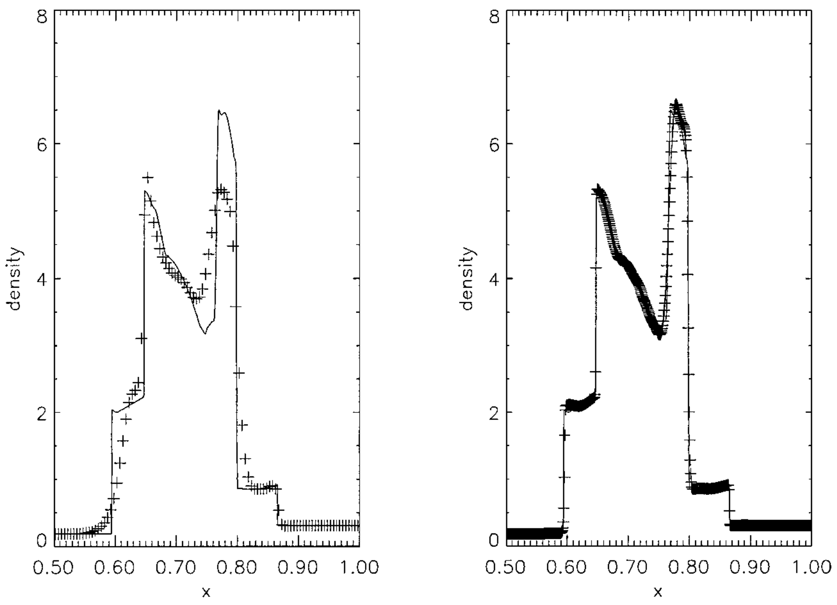

FIG. 3. Results for the interacting blast wave problem at two resolutions. Crosses on the left figure show the result with 200 points while crosses on the right figure are for 1200 points. The solid line on both plots is from a run with 4800 points.

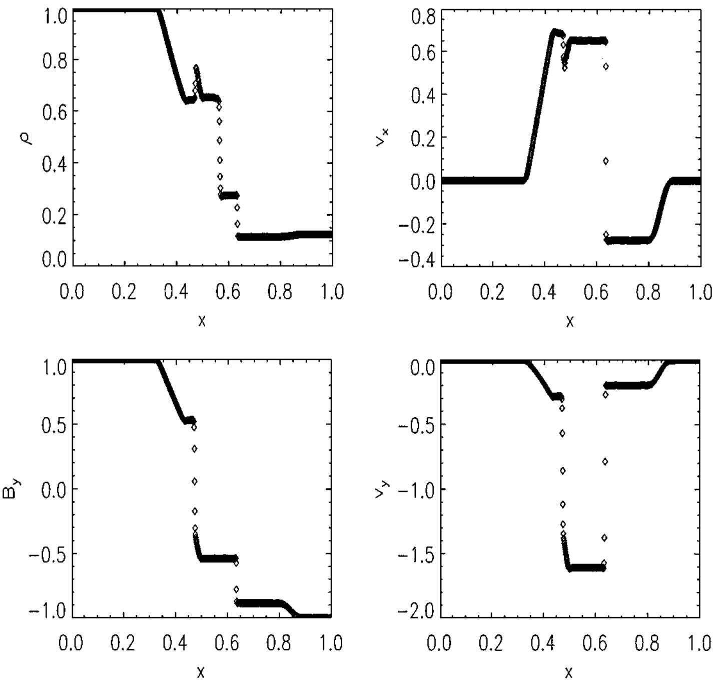

FIG. 4. Solution of the Brio and Wu MHD shock tube problem but with $\gamma = 5/3$ and 800 cells. Snapshots are taken at $t = 0.1$.

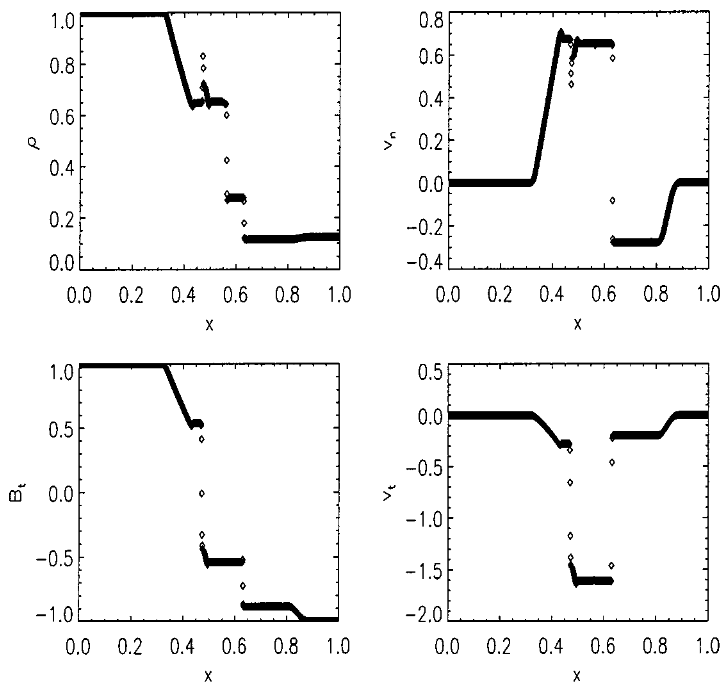

FIG. 5. Repeat of the test from Fig. 4 but with the shock propagating diagonally across an 800 × 800 grid.

the accumulated artificial viscous and ohmic heating contributions are both 0.37% of the total energy. Note that artificial viscosity converts kinetic energy into heat, in total 0.37% of the total energy, but does not contribute to the energy error. It conservatively converts energy from kinetic to thermal. The same is true of the resistive term which converts magnetic field energy into heat conservatively.

Figure 5 is a repeat of the Brio and Wu MHD shock tube problem with the same setup as in the previous paragraph, but now on a 800×800 grid with the shock propagating diagonally across the domain. While this uses a 2D grid it is still essentially a 1D problem and tests the codes' ability to evolve shocks obliquely across the grid. The only discernible differences between these plots and those of Fig. 4 are in the slight change in the treatment of the compound shock and slight overshoots at the end of rarefaction fans at $x = 0.425$ and $x = 0.45$.

3.3. *Orszag–Tang Vortex*

The Orszag–Tang vortex problem has been used in numerous papers [3, 4, 22, 23] as a 2D test for MHD codes. Here we use the same normalization as in Ref. [4]. The advantage of this choice of normalization is that Ref. [4] supplies quantitative estimates of the accuracy of schemes, other papers leaving the success of the Orszag–Tang vortex test to a qualitative or aesthetic judgment. Following Ref. [4] we use three resolutions for the $N \times N$ grid with $N = 50$, 100, and 200. The relative numerical error for a variable u is defined as the L_1 norm of the error relative to an accurate solution. The accurate solution here was taken as that obtained from a run with a 400×400 grid and the accurate solution was obtained on the course grids by control volume averaging as described in Ref. [4]. Using the same

TABLE II
Average Errors in the Orszag–Tang Test

	$N = 50$	$N = 100$	$N = 200$
$\bar{\delta}(t = 1)$	0.0344	0.0157	0.0052
$\bar{\delta}(t = 3.14)$	0.1494	0.0792	0.0308
$\Delta\varepsilon(t = 3.14)$	1.4	0.57	0.23

code to define the accurate solution in the calculation of errors does introduce a bias in favor of the current scheme. This is especially true as the high-resolution reference solution is only double the resolution of the $N = 200$ result. The same procedure is adopted in Ref. [4], with which we are making comparisons, except that in Ref. [4] an average of two $N = 400$ solutions is used to alleviate some of the biasing. Thus while the error estimates presented here for $N = 50$ and 100 should be accurate, that for $N = 200$ is likely to be overly optimistic. However, without an analytic solution it is not possible to estimate this bias, or indeed that in Ref. [4], and we simply proceed with caution when looking at the details of the error estimates for $N = 200$. Table II shows the average errors for this test at $t = 1$ and $t = 3.14$. The averaged error, $\bar{\delta}$, is the average of the L_1 norm of all primitive variables and $\Delta\varepsilon$ is the percentage energy error. For the same test problem in Ref. [4] the best $\bar{\delta}$ on a $N = 200$ grid was 0.0026 at $t = 1$ and the best at $t = 3.14$ was 0.0300.

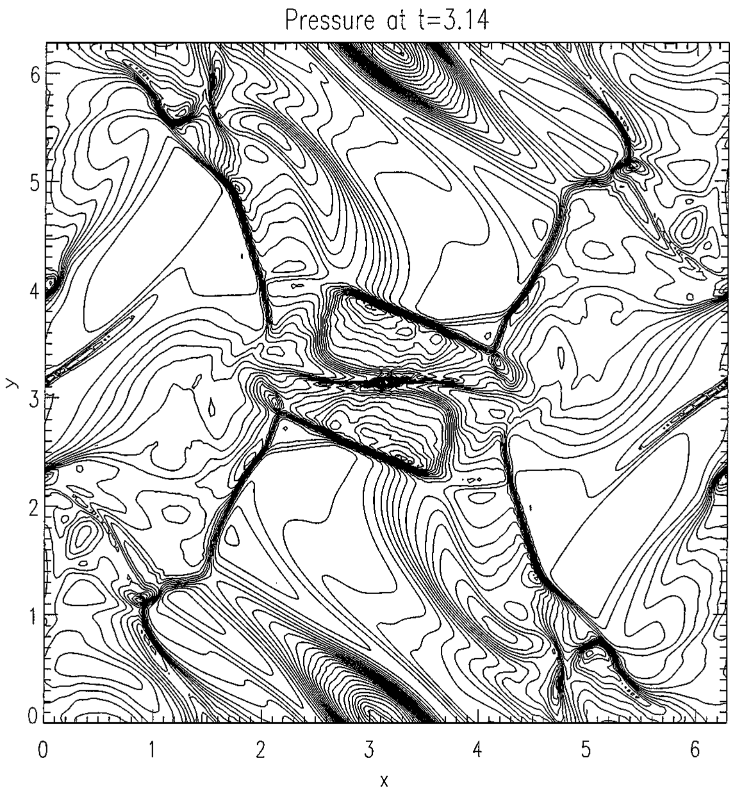

FIG. 6. Pressure distribution for the Orszag–Tang problem at $t = 3.14$. The computational box has 200×200 grid points.

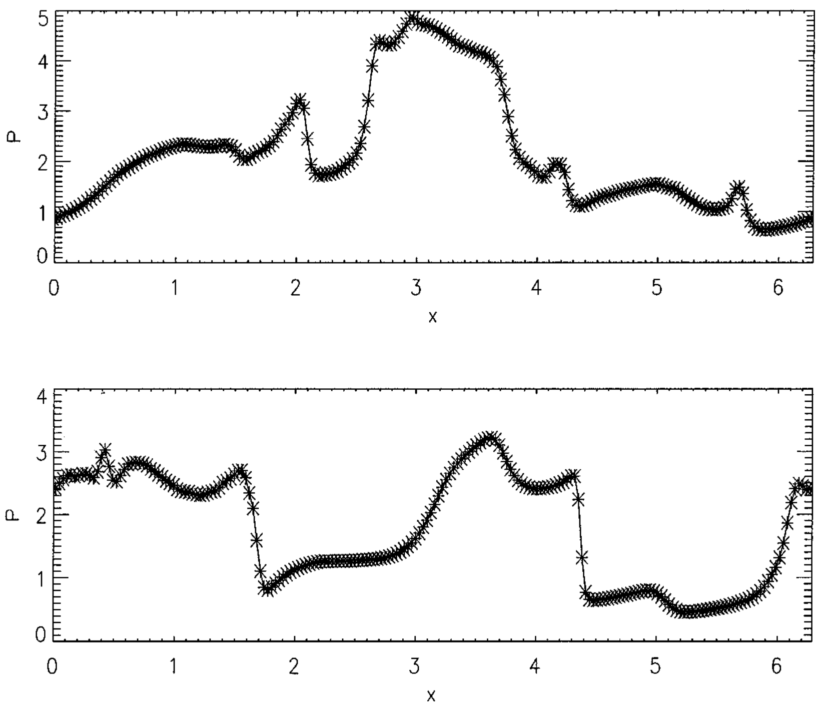

FIG. 7. 1D pressure distribution for the same problem as Fig. 6 along a cut at $y = 2.686$ (upper panel) and $y = 1.963$ (lower panel).

For comparison with results in Ref. [22], Fig. 6 shows a contour plot of the pressure at $t = 3.14$ with 30 contour lines. The time of $t = 3.14$ corresponds to $t = 0.5$ in the normalization used in Ref. [22], so Fig. 6 can be compared directly with Fig. 10 in Ref. [22]. Figure 7 shows cuts through the pressure distribution at points equivalent to those in Fig. 11 of Ref. [22]. All of this quantitative and qualitative evidence shows that the present Lagrangian remap scheme performs as well as Riemann-solver-based schemes once shocks and discontinuities form in the Orszag–Tang vortex test, i.e., at time $t = 3.14$. However, Table II also shows a somewhat surprising feature. While the errors on the 200×200 grid at $t = 3.14$ are as low as those for the best schemes tested in Ref. [4], the results at $t = 1$ are comparable to those of the worst code tested. Thus while `Lare3d` is good at treating shocks it is not as good for smooth flows when compared to approximate Riemann solvers. This can be clarified by finding the convergence of the average errors from `Lare3d` for a simple wave.

3.4. *Circularly Polarized Alfvén Waves*

Circularly polarized Alfvén waves are an exact solution to the full nonlinear equations of ideal MHD in a uniform medium. To allow direct comparison with the results in Ref. [4] we initialize a periodic box with an Alfvén wave propagating at an angle $\alpha = 30^\circ$ to the x-axis. The normalization and initial conditions are the same as in Ref. [4]. For this setup at time $t = 1$ the flow should have returned to its initial state. Table III shows the averaged errors, $\overline{\delta}$, calculated from the $v_\perp$, v_z, $B_\perp$, and B_z variables. These errors are at time $t = 5$ and are based on comparison of the result at $t = 5$ with that at $t = 0$ for the same resolution. As a result

TABLE III
Average Errors for Alfvén Waves

	$N = 8$	$N = 16$	$N = 32$	$N = 64$
$\bar{\delta}$	1.966	0.633	0.142	0.035
Amplitude error	0.014	0.055	0.005	0.0013
Phase error	2.597	0.663	0.138	0.033
$\lvert v_A \rvert$ error	0.072	0.018	0.0028	0.00086

they do not have the biasing discussed in the previous section and can be compared directly with results in Ref. [4]. Also shown are the results from an FFT of the initial and final data which was used to find the amplitude error (defined as $|a(t = 0) - a(t = 5)|/a(t = 0)$, where $a(t)$ is the amplitude), the phase error in radians, and the error in the Alfvén phase speed calculated from that phase error. Note that this is the error in the phase speed in the direction of propagation, not along the x-axis.

The average error, phase error, and $|v_A|$ error all show second order convergence. The odd behavior of the amplitude error on coarse grids is due to these grids not actually representing $a(t = 0)$ accurately. The dominance of phase error over amplitude error can be clearly seen from Fig. 8. The conclusion from these tests is that the present scheme cannot accurately find the phase speeds on coarse grids. The code has to recover this from finite differences

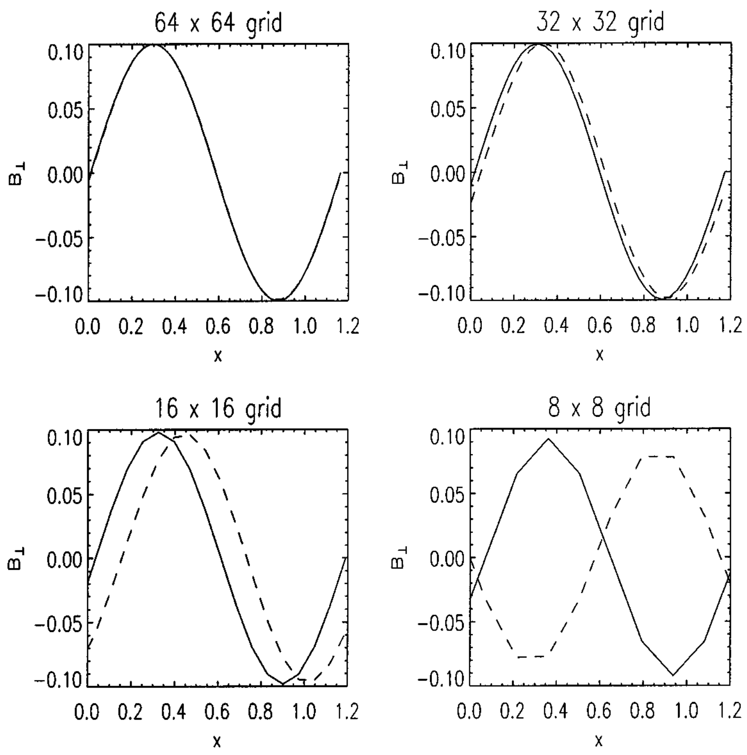

FIG. 8. The orthogonal component of the magnetic field $B_\perp = (\sqrt{3}B_y - B_x)/2$ for the circularly polarized Alfvén problem. The initial condition is shown as a full line and the result at $t = 5$ as dashed line for four resolutions.

TABLE IV
Average Errors in the Rotor Test

	$N = 50$	$N = 100$	$N = 200$
$\bar{\delta}$	0.1421	0.0711	0.0283
$\Delta\varepsilon$	0.51	0.28	0.15

in the Lagrangian step unlike approximate Riemann solvers where the characteristic speeds are included automatically (see Eq. (18)). This may also explain why the errors in the Orszag–Tang test are comparatively poor at $t = 1$. Shock propagation is, however, handled accurately, as witnessed by the accuracy achieved at $t = 3.14$ in the same test. If the need to resolve wave speeds accurately were paramount for a particular problem then the solution would be to replace the Lagrangian step with a higher order version. The scheme would still remain second order due to the remap but the Lagrangian phase would then accurately resolve the phase speeds. Even with the phase speed errors at their current levels it is still not obvious whether this is *physically* worse than the results presented in Ref. [4], where there is no phase error for the best codes but all codes strongly damp the wave on the coarse grid. Indeed, the best codes in Ref. [4] show an amplitude error of approximately 0.7 on an 8×8 grid compared to an amplitude error of 0.014 for the same test using `Lare3d`. Put crudely the question is: in any particular simulation is it better to have wave energy arrive early or on time but with a reduced energy density?

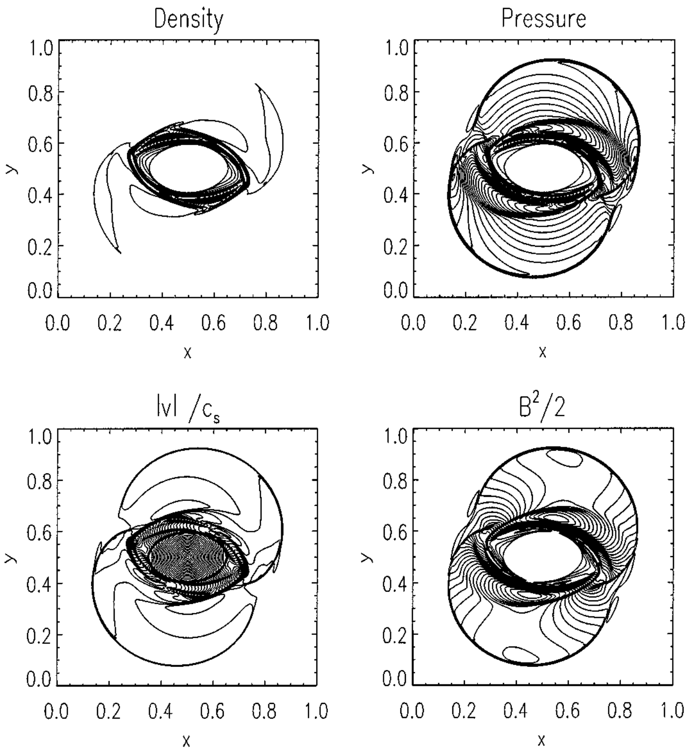

FIG. 9. Density, thermal pressure, Mach number, and magnetic pressure at $t = 0.15$ for the rotor problem. The solution was obtained on a 400×400 grid and the figure shows 30 contourlines.

3.5. *Rotor*

This problem is taken from Ref. [4] and is identical to the first rotor test with $P = 1$ from that paper. Table IV shows the usual table of averaged errors and energy conservation at time $t = 0.15$. Figure 9 shows contour plots of density, pressure, Mach number, and magnetic pressure for a 400×400 grid.

This problem converges with first-order accuracy as expected and once again the results are comparable with those from approximate Riemann-solver-based schemes. The best result obtained for this test on an $N = 200$ grid from Ref. [4] was $\overline{\delta} = 0.0276$ at $t = 0.15$, although once again one needs to be cautious in comparing these numbers, as they were produced with different small (but unquantifiable) biases as discussed in Section 3.3. As in previous sections this is not ideal but without a standard reference solution for 2D MHD shock problems there is little alternative. As a result we have chosen to follow the procedure adopted in Ref. [4], as this was the first paper to attempt a systematic quantitative set of convergence tests for 2D MHD problems.

3.6. *Low Beta MHD Tests*

This section demonstrates the benefits of the current scheme compared to traditional schemes based on the conservative form for low beta plasma tests. Comparisons have been made between results from `Lare3d` and a code which follows exactly the prescription set out for the TVD Riemann scheme in Ref. [23]. Using this TVD scheme reproduces the figures produced by `Lare3d` in the paper thus far. Up until this point the aim has been to demonstrate that a Lagrangian remap scheme, which abandons the conservative form, can correctly handle the standard MHD and Euler shock problems. This last section shows some of the benefits which follow from dropping the conservative form.

The first test is a simple 1D Alfvén pulse. The x domain is defined in $-1.5 < x < 1.5$ and initially $\rho = 1$, $B_x = 1$, $v_y = 0.2 \exp\{-x^2/0.01\}$, and all other components of $\mathbf{B}$ and velocity are zero. The specific energy density is set to 10^{-3} so that the plasma beta is $2 \times 10^{-3}/(\Gamma - 1)$. These tests are run to $t = 1$. Figure 10 shows the results from 300 grid points using `Lare3d` and 30,000 grid points for the TVD scheme.

The reason so many more points were used for the TVD scheme in Fig. 10 is explained in the convergence test for the TVD scheme. Figure 11 shows ε from the TVD scheme with resolutions of 300, 3000, and 30,000 grid points. The poor results from the conservative form of the TVD scheme follow partly from the pressure, and hence specific energy density and temperature, being derived from the difference of two large terms. Repeating the `Lare3d` run with 3000 grid points reproduces the same result as with 300 grid points to within a few percent and it is the `Lare3d` result which is the correct converged answer. The other reason for the TVD schemes overheating the low beta plasma relates to Section 3.4. Here it was shown that `Lare3d` damped the amplitude of simple waves far less than Riemann-based schemes of equivalent order. The energy that the TVD scheme dissipates from the magnetic field of the wave is placed into the thermal energy. Thus for a TVD Riemann scheme to produce accurate temperature estimates in solar coronal simulations it would either have to use around 100 times the resolution of `Lare3d`, or an equivalent approach, or a high-order scheme, such as employed by ENO codes, to prevent wave damping. In practice 3D runs are limited to about 300^3 so Fig. 11 shows the scale of error in the pressure one would expect

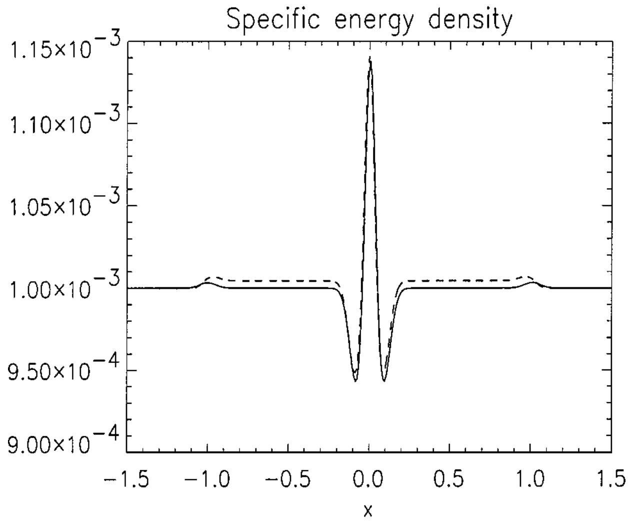

FIG. 10. 1-D distribution of ε for the low beta Alfvén pulse test. The solid line is from `lare3d` on a 300 cell grid and the dashed line is from the TVD scheme on a 30,000 cell grid.

for such a grid; i.e., the error is the same order as the solution itself. However, in 2D and 3D runs it is more usual that conservative schemes (such as the TVD scheme used here) will fail due to negative pressures. This is true for the Orszag–Tang vortex test, which fails with the TVD scheme when initialized with $\beta = 10^{-3}$ but runs without error in `Lare3d`.

As a final test we return to the Brio and Wu shock problem of Section 3.2. Figure 12 presents results from the same setup as in Section 3.2 but now the solid line is from `Lare3d` and the points are from the TVD scheme. This confirms that both schemes produce the same solution for this MHD shock problem.

However, as for the Orszag–Tang vortex test, the TVD scheme fails with negative internal energy if the plasma beta is reduced. Figure 13 shows the result from `Lare3d` of

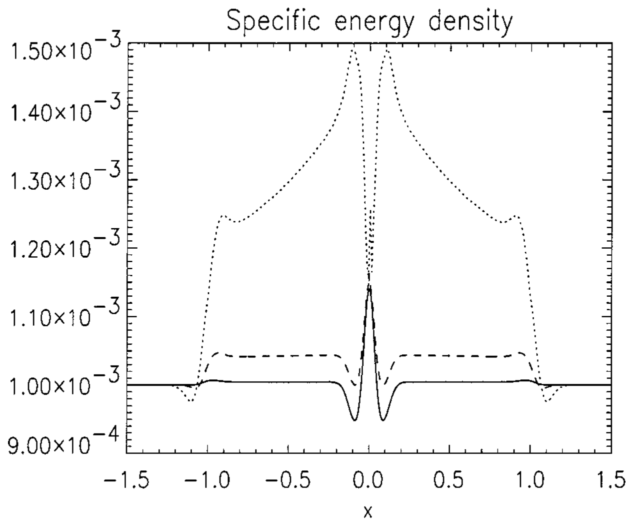

FIG. 11. Convergence of the TVD scheme for the 1D distribution of ε on the low beta Alfén pulse test. The solid line is for 30,000 grid points, the dashed line for 3000 grid points, and the dotted line for 300 grid points.

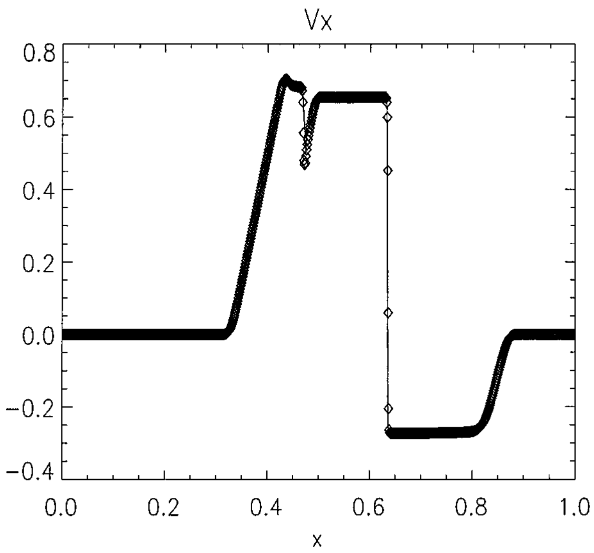

FIG. 12. Solution of the Brio and Wu MHD shock tube problem but with $\gamma = 5/3$ and 800 cells. Snapshots are taken at $t = 0.1$. The solid line is the result from `Lare3d` and the points are from the TVD scheme.

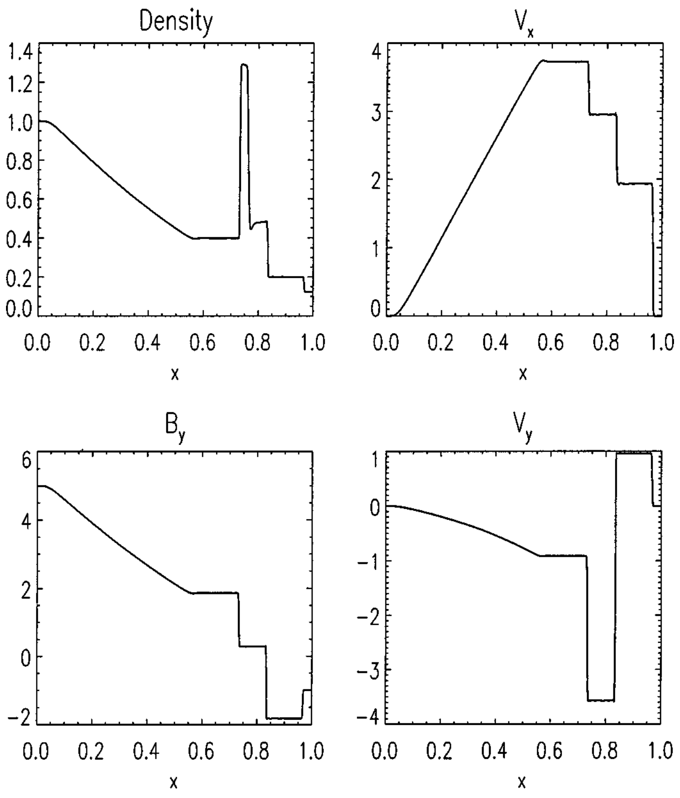

FIG. 13. Solution of the Brio and Wu MHD shock tube problem but with $\gamma = 5/3$, 800 cells, $\varepsilon = 10^{-3}$ throughout and $B_y = 5$ for $x < 0.5$.

setting up the standard Brio and Wu MHD shock test but then setting $\varepsilon = 10^{-3}$ everywhere and increasing the initial perpendicular magnetic field component, B_y, for $x < 0.5$ from 1 to 5.

4. CONCLUSIONS

We have written and tested a Lagrangian remap code (`Lare3d`) for solving multidimensional MHD. One of the main motivating factors in this work was to develop a flexible code suitable for simulating events in the Solar corona. Such a code needed to be capable of capturing shocks, accurately finding the local temperature, and allowing the inclusion of additional nonhyperbolic physics (resistivity, viscosity, radiation, thermal conduction, gravity, etc.). These requirements lead us to a Lagrangian remap scheme for the nonconservative form of the MHD equations. The basic features of the code are

- The Lagrangian step is second order in space and time and uses mass conservation (mass coordinates in 1D) whenever possible.
- The magnetic field is defined on cell faces and is updated with constrained transport to keep $\nabla \cdot \mathbf{B} = 0$ to machine precision.
- The Lagrangian step includes Wilkins viscosity which prevents Gibbs overshoot and maintains shock structure for shocks moving at angles across the grid.
- The density is remapped using Lagrangian coordinates but energy and velocity use mass coordinates.
- Even though the equations are not in conservative form the Euler part of the code conserves energy to machine accuracy. This procedure could be extended to the magnetic field energy but this hasn't shown itself to be a critical issue so far.

We have performed a set of tests with `Lare3d` which demonstrate its ability to handle shocks and have compared the results of these tests with those from other shock capturing schemes. The features of any multidimensional code are always difficult to quantify. In an attempt to highlight these features and compare them with other schemes we include below lists of the successes from the tests and those areas where other codes give better results. As always with such comparisons there are grey areas where different users (depending on the application they have in mind) may see things differently. With those words of caution we begin with a list of the features of the tests where `Lare3d` did not perform as well as other codes.

1. While all of the tests in this paper were performed with the same Wilkins viscosity it was only through experience that we have learnt to use those values. For radically different tests one may find that these values need changing. Thus `Lare3d` is not as robust as approximate Riemann solver-based codes.
2. In the 2-D Brio and Wu test (see Fig. 5) and the high-resolution strongly interacting shock test (see Fig. 3) there is evidence of overshooting at the end of rarefaction fans. This is not Gibbs overshoot, since it is resolved by many points and smooth, but it should not be present in an ideal solution.
3. The Lagrangian step only finds the characteristic speeds to second order accuracy. This causes the circularly polarized Alfvén wave to have a numerical speed which is larger than unity for that test and consequently gets a poor value for averaged L_1 norm errors.

The first of these points is not a major problem. From low-resolution tests one will know if the viscosity needs to be increased so that little time is actually wasted repeating full production runs with an assortment of values for the linear viscosity. The second point is clearly an error in the way the code treats rarefaction fans for such test examples. We have no explanation for why the code should be able to handle shocks well but have trouble with smooth regions of the solution. We do, however, note that `Lare3d` is not alone in having this undesirable feature (see, e.g., Fig. 5 in Ref. [9] and Fig. 13 in Ref. [24]).

The positive features of `Lare3d` are

1. The code does correctly handle shocks. Furthermore, for most shock tests where a quantitative comparison is possible (Orszag–Tang vortex and rotor tests) `Lare3d` performed as well as Riemann-solver-based codes. The exception to this was the 1D interacting strong shock test where the PPM method was more accurate.
2. The scheme preserves $\nabla \cdot \mathbf{B} = 0$ to machine precision at no extra computational cost.
3. By not using the conservative form, but still correctly handling shocks, the scheme is able to accurately find the local temperature even for low-beta plasmas such as the solar corona.
4. For the Alfvén wave test problem `Lare3d` showed very little damping of the wave even on an 8×8 grid.
5. All of the physics is contained in the Lagrangian step which is unsplit and fully three-dimensional.

The results of Ref. [4] have shown that dimensionally split algorithms do give accurate results for the Orszag–Tang and rotor tests. Whether a clear distinction between split and unsplit techniques becomes evident when source terms, radiation, conduction, etc., are included is not clear and will have to be the subject of future investigations. The Lagrangian remap approach adopted in `Lare3d` does easily lend itself to the inclusion of both tensor viscosities and interface tracking through volume fractions (see Benson's review for details [13]).

If you already use, or are about to acquire, a Riemann-based ideal MHD shock capturing code, should you abandon your plans and use `Lare3d` instead? Of course not! These tests have shown that the present scheme is as good as well-written approximate Riemann solvers but not better. Perhaps the most surprising result of this paper is that this is possible without using a characteristic-based method. So provided you want to solve ideal MHD problems and you have a $\nabla \cdot \mathbf{B} = 0$-preserving approximate Riemann solver code the results in this paper are of general interest only. However, if you wish to perform simulations of the solar corona, or any other plasma with a beta below 0.1%, then unless you adopt a procedure similar to that in Ref. [5] any estimates from a conservative-form-based code for the temperature will be very inaccurate. Attempting to then calculate ionization levels or thermal conductivities from these temperatures will inevitably lead to inaccurate results. Balsara and Spicer [5] have shown a way of avoiding these difficulties but this approach is overlooked in the majority of codes. If in addition to a low plasma beta one also wishes to have simulations which are also expected to take into account complex equations of state or interface tracking then the approach of `Lare3d` offers distinct advantages.

Assessment of the performance of codes, especially parallel performance, is always less conclusive than we might like. It depends strongly on architecture, compilers, interconnects, etc. In this case another factor is that the code was written in HPF to run typically on eight CPU's of a cluster of four CPU SMP machines. There is an overhead in using F90 and at times serial speed is compromised to achieve better parallel scaling under high performance Fortran (HPF). With these cautionary remarks noted, using a Compaq ES40 with 500-MHz EV6 processors, 4 MB cache per CPU, and Compaq's F90/HPF compiler, a 192^2 grid test run of the Orszag–Tang problem took 141.6 s on one CPU, 36.1 s on all four CPUs of the ES40, and 20.2 s on eight CPUs, i.e., two ES40s connected through Memory Channel 2 interconnect. For comparison a straight F90 coding of the TVD Riemann scheme in Ref. [23] took 123.5 s for the same problem on one CPU. This code was not written in F90 full array notation; i.e., it avoided slower constructs needed to make HPF effective and is therefore more clearly optimized for serial execution. For a fixed number of timesteps `Lare3d` is actually faster than the TVD scheme but only by about 10%. The longer runtime to a fixed time in the Orszag–Tang problem, even though both codes use a CFL number of 0.8, is because the artificial viscosity appears in the CFL condition in `Lare3d` so it requires more timesteps to reach a given fixed final time. The conclusion that we draw from these tests is that `Lare3d` costs about the same, in CPU time, as an equivalent TVD Riemann scheme. For 3D tests 10 steps with a 128^3 grid took 735 s on 1 CPU, 184 s on four CPUs, and 110 seconds on eight CPUs. This shows signs of stalling on eight CPUs but larger array jobs have scaled linearly on a CRAY T3E up to 64 CPUs.

APPENDIX A

Details of Finite Difference Scheme

In this Appendix we present a summary of the finite difference equations used in the code. It is intended to cover all of the detail necessary to understand the core numerical schemes. Anyone wishing to obtain a copy of the full Fortran90 source code (actually in full parallel HPF) can do so by contacting the authors at tda@astro.warwick.ac.uk. Figure 1 defines the location of the primary variables on the grid. At the start of each step all of these are defined at the same time; i.e., there is no leapfrog component. The Lagrangian step is a simple second order predictor–corrector scheme. To distinguish the different time levels, variables with no superscript, e.g., $\mathbf{v}$, refer to variables on the Eulerian grid at the start of the step; variables with a star superscript are half-timestep Lagrangian predictor values, e.g., $\mathbf{v}^*$, and variables with a superscript 1 represent the values at the end of the Lagrangian step, e.g., $\mathbf{v}^1$, and are defined on the displaced Lagrangian grid. At several points throughout the scheme variables are needed at different locations than those defined in Fig. 1. The averaging used depends on whether the variable is a volume average, e.g., ρ, or a surface average, e.g., Bx. Since the Eulerian grid can be stretched we begin by defining $cvol_{i,j,k}$ as the volume of each cell. For the set of indices (i, j, k) $\rho_{i,j,k}$ and $\varepsilon_{i,j,k}$ are the averages over $cvol_{i,j,k}$ of density and specific energy, and are defined at the cell volume center. $Bx_{i,j,k}$ is the x component of the magnetic field and is defined to be face centered on the face at $xc_{i,j,k} + dxb_{i,j,k}/2$, where $xc_{i,j,k}$ is the x coordinate of the center of the cell and $dxb_{i,j,k}$ is the length of the cell in the x direction. $By_{i,j,k}$ and $Bz_{i,j,k}$ are similarly defined, as are $dyb_{i,j,k}$ and $dzb_{i,j,k}$. The remap stage also uses (dxc, dyc, dzc), where dxc is the distance between the center of the control volume $cvol_{i,j,k}$ and the center of the control volume at $cvol_{i+1,j,k}$, and similar definitions apply to dyc and dzc. All of the components of velocity are defined at the cell vertex so that $vx_{i,j,k}$ is defined at the point $(xc_{i,j,k} + dxb_{i,j,k}/2, yc_{i,j,k} + dyb_{i,j,k}/2, zc_{i,j,k} + dzb_{i,j,k}/2)$. To obtain the density at the cell vertex, $\rho^v_{i,j,k}$, we use control volume averaging; i.e.,

$$\rho^v_{i,j,k} = \frac{1}{8\,cvol^v_{i,j,k}} \sum_{l=i}^{i+1} \sum_{m=j}^{j+1} \sum_{n=k}^{k+1} \rho_{l,m,n}\, cvol_{l,m,n}, \tag{A.1}$$

where

$$cvol^{v}_{i,j,k} = \frac{1}{8}\sum_{l=i}^{i+1}\sum_{m=j}^{j+1}\sum_{n=k}^{k+1} cvol_{l,m,n} \tag{A.2}$$

is the velocity cell control volume. The **B** field components at the cell center are simply the averages of the values on opposing faces. The velocity components defined on cell faces, e.g., $vxb_{i,j,k}$ are defined by averaging over the four vertex values.

Using the above conventions the predictor stage of the Lagrangian step can be broken down into the following finite difference equations. In the remainder of this Appendix omission of a subscript implies that all variables should be subscripted with (i, j, k). Initially only the scheme without the resistive terms will be described and resistive effects introduced separately at the end. This helps to emphasize the core MHD code and shows how additional terms are added to this core solver. In reality, even in running an ideal MHD problem, artificial resistivity must be included at shocks, but purely to make the description of the code as clear as possible we shall define here the core code to be ideal MHD plus only artificial viscosity. First, the thermal pressure P is defined by

$$P = \varepsilon(\Gamma - 1)\rho \tag{A.3}$$

and the total pressure $P_{total} = P + q$ with q defined from Eq. (20) taken directly from [10]. The predictor value of the specific energy density is then given by

$$\varepsilon^{*} = \varepsilon - \frac{\delta t}{2}\frac{P_{total}\nabla\cdot\mathbf{v}}{\rho} \tag{A.4}$$

where δt is the timestep and $\nabla.\mathbf{v}$ is found from

$$\nabla.\mathbf{v} = \frac{vxb_{i,j,k} - vxb_{i-1,j,k}}{dxb_{i,j,k}} + \frac{vyb_{i,j,k} - vyb_{i,j-1,k}}{dyb_{i,j,k}} + \frac{vzb_{i,j,k} - vzb_{i,j,k-1}}{dzb_{i,j,k}}. \tag{A.5}$$

The Jacobian of the predictor Lagrangian step is then defined by

$$\Delta^{*} = 1 + \frac{\delta t}{2}\nabla\cdot\mathbf{v}; \tag{A.6}$$

then

$$\rho^{*} = \frac{\rho}{\Delta^{*}}, \tag{A.7}$$

$$P^{*}_{total} = \varepsilon^{*}(\Gamma - 1)\rho^{*} + q. \tag{A.8}$$

Note that the artificial viscous pressure q is not advanced to the predictor level. To find the predictor value for the force one must now update the **B** field so that the Lorentz force is correctly time-centered. The predictor **B** field is cell-volume-centered and follows from the finite difference version of Eq. (14), e.g., for Bx^{*} this is

$$\begin{aligned} Bx^{*} = \frac{1}{\Delta^{*}}\Bigg\{ & Bx + \frac{\delta t}{dxb}[(vx\,Bx)^{x^{+}} - (vx\,Bx)^{x^{-}}] + \frac{\delta t}{dyb}[(vx\,By)^{y^{+}} - (vx\,By)^{y^{-}}] \\ & + \frac{\delta t}{dzb}[(vx\,Bz)^{z^{+}} - (vx\,Bz)^{z^{-}}]\Bigg\}, \end{aligned} \tag{A.9}$$

where $(vx\,Bx)^{x^+}$ means the product of vx and Bx averaged to the center of the x face at $xc_{i,j,k} + dxb_{i,j,k}/2$ and $(vx\,Bx)^{x^-}$ is averaged to the x face at $xc_{i,j,k} - dxb_{i,j,k}/2$. Similarly $(vx\,By)^{y^+}$ means $(vx\,By)$ averaged onto the y face at $yc_{i,j,k} + dyb_{i,j,k}/2$ etc.

We now have the predictor values of pressure and magnetic field and can therefore calculate the vector force at the cell vertex. This is found from $\mathbf{F}^* = \mathbf{B}^* \cdot \nabla \mathbf{B}^* - \frac{1}{2}\nabla B^{*2} - \nabla P^*$, where now the averaging needed to define magnetic field components at the desired locations must now use control volume averaging, as $\mathbf{B}^*$ is a volume-centered variable. The predictor step is then completed by finding the half timestep predictor velocities defined at the cell vertex from, e.g.,

$$vx^* = vx + \frac{\delta t}{2}\frac{Fx^*}{\rho^v}, \tag{A.10}$$

where Fx^* is the x component of the predictor value of the vertex force.

The corrector step is now straightforward since the $\mathbf{B}$ field does not need to be updated. The $\mathbf{B}$ field components are simply converted into fluxes using $\Phi x = Bx\,dyb\,dzb$, etc. for Φy and Φz. Since the core solver is for ideal MHD, with artificial viscosity, the flux is conserved during the Lagrangian step. The corrector step is therefore an update of the density control volume for each cell using $\Delta^1 = 1 + \delta t \nabla \mathbf{v}^*$; i.e., $cvol^1 = cvol\,\Delta^1$, followed by

$$\varepsilon^1 = \varepsilon - \delta t\, P^*_{total}\, \frac{Fx^*}{\rho}, \tag{A.11}$$

$$\rho^1 = \frac{\rho}{\Delta^1}, \tag{A.12}$$

$$vx^1 = vx + \delta t \frac{Fx^*}{\rho^v}, \tag{A.13}$$

with similar equations for vy^1 and vz^1. Note that in the update of the specific energy density and velocities it is the original Eulerian density that is used. This is based on using control volume mass conservation during the Lagrangian step and is made explicit in Appendix B, where the finer details are discussed. The final set of variables to be updated are (dxb, dyb, dzb), which are simply calculated from $dxb^1_{i,j,k} = dxb_{i,j,k} + (vxb^*_{i,j,k} - vxb^*_{i-1,j,k})\delta t$ etc.

All variables have now been updated a full timestep and are defined on the Lagrangian grid. The next stage is to remap these variables back onto the original Eulerian grid so that the whole process can continue. The remap step is entirely geometrical and includes no physics or time dependence. However, the monoticity-preserving property of 1D hyperbolic equations is built into the remap by the use of van Leer-limited piecewise linear reconstruction. The remap is done in 1D sweeps in the usual Strang-ordered way. To explain the entire remap process it is therefore sufficient to cover just the x remap, as y and z remaps follow exactly the same scheme. In what follows we therefore drop the (j, k) subscripts and deal only with a 1D remap. Note that this simplification assumes that we start with Lagrangian variables such as vx^1 and end with the final fully updated variable on the Eulerian grid, i.e., $vx(t + \delta t)$. In 3D the x remap can actually start with variables which have, for example, been modified by a y remap and for which the output needs to be further remapped in z. In what follows therefore vx' refers to the x component of velocity before the x remap and vx^{n+1} refers to the velocity after the remap. Hence vx' only equals vx^1 if the x remap is the first after the Lagrangian step. If the x remap is not the last direction to be remapped then vx^{n+1} becomes vx' for the next stage of the remap and so on for all other variables.

The density is remapped conservatively so that the total mass in the cell after the remap $\rho^{n+1} dxb$ is equal to the mass before the remap $\rho'\, dxb'$ minus the mass from this Lagrangian cell which overlaps the Eulerian cell at $i+1$ (dM_i) plus the mass from Lagrangian cell $i-1$ which overlaps the Eulerian cell i (dM_{i-1}). Since $\rho'\, dxb' = \rho\, dxb$ this becomes

$$\rho_i^{n+1} = \rho + \frac{1}{dxb_i}\left(dM_{i-1} - dM_i\right), \tag{A.14}$$

where

$$dM_i = \left(\rho_i' + \frac{dxb_i'}{2} D_i\, (1-\psi_i)\right) vx_i^*\, \delta t \tag{A.15}$$

and

$$\psi_i = \frac{|vx_i^*|\delta t}{dxb_i'}. \tag{A.16}$$

Note that in these two equations vx_i^* is the velocity of the boundary but in 3D this needs to be replaced by the face-centered velocity vxb_i^*. The variable D in this equation is the van Leer piecewise linear, limited gradient. In this implementation the gradient is found by initially calculating the third-order upwind gradient from the formula for a general variable f; i.e.,

$$|\bar{D}_i| = \frac{(2-\psi_i)}{3}\frac{|f_{i+1}-f_i|}{dxc_i} + \frac{(1+\psi_i)}{3}\frac{|f_i - f_{i-1}|}{dxc_{i-1}} \quad \text{for } vx_i' > 0, \tag{A.17}$$

$$|\bar{D}_i| = \frac{(2-\psi_i)}{3}\frac{|f_{i+1}-f_i|}{dxc_i} + \frac{(1+\psi_i)}{3}\frac{|f_{i+2} - f_{i+1}|}{dxc_{i+1}} \quad \text{for } vx_i' \le 0. \tag{A.18}$$

The magnitude of the gradient obtained, i.e., $|\bar{D}_i|$, is then limited, if need be, using the procedure given as Eq. (101) in [11]. In the current notation this is

$$D_i = s\ \mathrm{MAX}(|\bar{D}_i| dxb_i, 2|f_{i+1} - f_i|, 2|f_i - f_{i-1}|), \tag{A.19}$$

where

$$\begin{aligned} s &= \mathrm{sign}(f_{i+1}-f_i) \quad \text{if } \mathrm{sign}(f_{i+1}-f_i) = \mathrm{sign}(f_i - f_{i-1}),\\ s &= 0 \qquad\qquad\qquad\qquad \text{otherwise.} \end{aligned} \tag{A.20}$$

This then completes the remap of density and ρ^{n+1} and dM_i are stored.

The specific energy density remap follows the same basic procedure as that of the density but the remap now uses the dM_i values to complete the remap in mass coordinates. This builds the mass conservation into the remap of other variables and is achieved for the specific energy density through

$$\varepsilon_i^{n+1} = (\varepsilon_i'\, dxb_i'\, \rho_i' + d\varepsilon_{i-1} - d\varepsilon_i)\frac{1}{dxb_i\, \rho_i^{n+1}}, \tag{A.21}$$

where

$$d\varepsilon_i = \left(\varepsilon_i' + \frac{dxb_i}{2} D_i\left(1 - \frac{dM_i}{\rho_i'\, dxb_i'}\right)\right) dM_i. \tag{A.22}$$

Now D_i is the van Leer-limited gradient of the specific energy density and $d\varepsilon_i$ is the energy remapped (not specific energy density) from cell i to cell $(i+1)$.

Mass coordinates are also used to remap the velocity components, thus ensuring conservation of momentum in the remap step. The only additional complication introduced for the velocity is that the velocity components are defined at cell vertices. Since the velocity has a different control volume than the density, dM_i and vx_i' must be averaged to the appropriate faces of the velocity control volume before the remap can start. In all other respects the velocity remap is the same as that of the specific energy density.

The calculation of the magnetic flux to be remapped follows the same approach as that of the density. The total flux through the y face at $yc_{i,j,k} + dyb_{i,j,k}/2$ is unchanged during the Lagrangian step and is given by $\Phi y = By\, dxb\, dzb$, and this is remapped using vx^* to find the area of Lagrangian cells overlapping neighboring Eulerian cells in the x pass of the remap. However, since the flux is defined as a face surface averaged quantity the velocity must be defined at the edge center; i.e., in Eq. (A.15) $vx^*_{i,j,k}$ must now be replaced by $vx_{i,j,k} = 0.5(vx^*_{i,j,k} + vx^*_{i,j,k-1})$. In all other respects the calculation of $d\Phi y_{i,j,k}$, the y flux remapped from cell (i, j, k) to cell $(i+1, j, k)$, follows the calculation of $dM_{i,j,k}$. The $\nabla \cdot \mathbf{B} = 0$ scheme then requires that

$$\begin{aligned}
\Phi y^{n+1}_{i,j,k} &= \Phi y_{i,j,k} - d\Phi y_{i,j,k},\\
\Phi y^{n+1}_{i+1,j,k} &= \Phi y_{i+1,j,k} + d\Phi y_{i,j,k},\\
\Phi x^{n+1}_{i,j,k} &= \Phi x_{i,j,k} + d\Phi y_{i,j,k},\\
\Phi x^{n+1}_{i+1,j,k} &= \Phi x_{i+1,j,k} - d\Phi y_{i,j,k},
\end{aligned} \tag{A.23}$$

etc., for the other components. Converting the fluxes back into field components then completes the remap step and all variables are defined on the original Eulerian grid ready for the next Lagrangian step.

The artificial resistivity can then be added in the same manner as the artificial viscosity, i.e., calculated only at the start of the step, and the resulting heating only added into ε based on that value. Alternatively, it can be added in a time-centered way and combined with a real second order accurate resistive term. This is then included in the calculation of the time-centered ε; the time-centered $\mathbf{B}$ field is then used to recalculate the contribution to heating in order to maintain second order accuracy. The time-centered $\mathbf{B}$ field is also used to evaluate the RHS of Eq. (15), which then simply updates the fluxes ready for the start of the remap step.

APPENDIX B

Energy Conservation in the Lagrangian Step

In this Appendix we present the proof of energy conservation in the Lagrangian step update of Euler's equations. The generalization of this to 3D is straightforward but tedious. For simplicity here we also assume that the grid is uniform, at the start of the Lagrangian

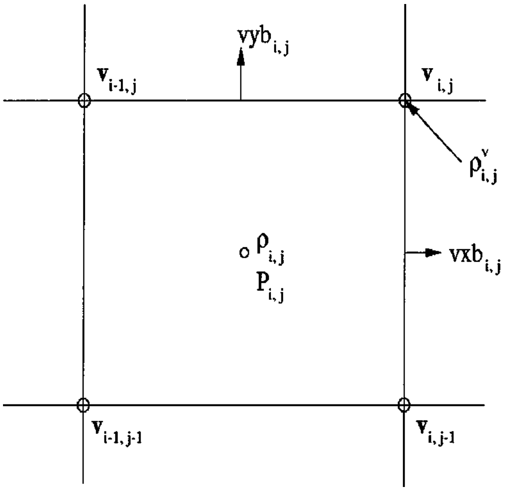

FIG. 14. The position of variables defined on a 2D grid.

step, in x and y but that $dx \neq dy$. This proof is also valuable as it clearly shows how conservation of mass is built into the Lagrangian update step on every possible occasion. In 1D this reduces to the usual mass coordinates of Lagrangian fluid dynamics. In 2D and 3D the scheme is simply a staggered grid control volume approach with exact mass and energy conservation.

We begin by defining a set of new variables. $vxb_{i,j}$ is the x component of the velocity defined on the cell edge (face in 3D); $\rho^v_{i,j}$ is the density defined at the cell vertex; $M^v_{i,j} = \rho^v_{i,j}\, dx\, dy$ is the mass in the velocity control volume.

See Fig. 14 for clarification of the locations on the grid. The energy calculation is performed at the corrector step so superscripts n, $n+1/2$, and $n+1$ refer to time levels. The change in kinetic energy between levels n and $n+1$ over the entire domain is

$$\Delta KE = \sum_{i,j} \frac{1}{2} M^v_{i,j} \left(\left(v^{n+1}_{i,j}\right)^2 - \left(v^n_{i,j}\right)^2 \right), \tag{B.1}$$

$$= \sum_{i,j} M^v_{i,j} \left(v^{n+1}_{i,j} - v^n_{i,j}\right) v^{n+1/2}_{i,j}. \tag{B.2}$$

From Eq. (2) with no magnetic field we get

$$vx^{n+1}_{i,j} - vx^n_{i,j} = \frac{dt}{2\, dx^{n+1/2}\, (\rho^v_{i,j})^{n+1/2}} \left(P^{n+1/2}_{i,j} + P^{n+1/2}_{i,j+1} - P^{n+1/2}_{i+1,j} - P^{n+1/2}_{i+1,j+1}\right). \tag{B.3}$$

Using this equation along with mass conservation in a Lagrangian cell, i.e.,

$$M^v_{i,j} = \left(\rho^v_{i,j}\right)^n dx\, dy, \tag{B.4}$$

$$= \left(\rho^v_{i,j}\right)^{n+1/2} dx^{n+1/2}\, dy^{n+1/2}, \tag{B.5}$$

we can regroup the terms in Eq. (B.2) containing $P_{i,j}$ to get

$$\sum_{i,j} \Delta KE_{i,j} = \sum_{i,j} P^{n+1/2}_{i,j} dV_{i,j}, \tag{B.6}$$

where

$$dV_{i,j} = dt[(vxb_{i,j} - vxb_{i-1,j})\,dy + (vyb_{i,j} - vyb_{i,j-1})\,dx] \tag{B.7}$$

and

$$vxb_{i,j} = \frac{\left(vx_{i,j}^{n+1/2}\,dyv_{i,j}^{n+1/2} + vx_{i,j-1}^{n+1/2}\,dyv_{i,j-1}^{n+1/2}\right)}{2\,dy}. \tag{B.8}$$

In these expressions $dyv_{i,j}^{n+1/2}$ is the predicted value of $dy_{i,j}^{n+1/2}$ through the vertex, i.e., the predicted value of dy for the velocity control volume. Note that Eq. (B.7) is the equivalent to $dV_{i,j} = \nabla \cdot \mathbf{v}\,dt\,dx\,dy$ using Eq. (B.8) for the velocities and taking derivatives on the original Eulerian grid. With these definitions if we insist that

$$\sum_{i,j} \Delta E_{i,j} = -\sum_{i,j} P_{i,j}^{n+1/2}\,dV_{i,j} \tag{B.9}$$

with $E = \int \rho\,\varepsilon\,d\tau$, then the scheme will conserve energy to machine precision. This amounts to using Eq. (B.8) to find average velocities and then when calculating the adiabatic $P\,dV$ heating term in Eq. (4), making sure that we use these same average velocities. This is also sufficient to guarantee that at the Lagrangian corrector step all derivatives are still taken on the original Eulerian grid. This last property also carries over to finding the time-centered magnetic forces in Eq. (2). However, we have no proof that exact energy conservation holds when the magnetic field is included. Indeed, in this code the magnetic field at the end of the Lagrangian step is never actually needed or calculated. Thus when energy errors are quoted in this paper they must entirely be due to errors in the magnetic field update scheme.

APPENDIX C

Kinetic Energy Remap

In this Appendix we present the calculations used to conserve energy in the remap step. We consider the change in the kinetic energy. This is then summed over the cells to find the energy which is lost in the remap. This energy is then added into the internal energy as a heating term, thus conserving the energy (note that magnetic energy can still be lost). We explain the velocity remap energy conservation by describing the first-order donor cell method. The full remap for any order is similar but with the velocities replaced by fluxes. Here we deal only with the x remap. The y and z remaps are carried out in a similar manner. The x remap step uses u_i^0 as the vertex velocity before the remap and u_i' as the vertex velocity after the remap. Conservation of mass can be written as

$$m_i' = m_i^0 - dm_{i+1/2} + dm_{i-1/2}, \tag{C.1}$$

where m_i^0 is the mass in the velocity control volume before the remap and m_i' that after the remap. $dm_{i+1/2}$ is the mass flux through the left boundary of the velocity cell during the remap. The remap also conserves momentum; i.e.,

$$u_i' m_i' = u_i^0 m_i^0 - u_{i+1/2}^{1/2} dm_{i+1/2} + u_{i-1/2}^{1/2} dm_{i-1/2}, \tag{C.2}$$

We consider the change in the kinetic energy,

$$\Delta KE_i = \frac{1}{2}m_i'(u_i')^2 - \frac{1}{2}m_i^0\left(u_i^0\right)^2, \tag{C.3}$$

or

$$\Delta KE_i = u_i^0\left(u_{i-1/2}^{1/2} - \frac{1}{2}u_i^0\right)dm_{i-1/2} - u_i^0\left(u_{i+1/2}^{1/2} - \frac{1}{2}u_i^0\right)dm_{i+1/2} + \frac{1}{2}m_i'a_i^2, \tag{C.4}$$

where

$$a_i = \left(u_i^0 - u_{i+1/2}^{1/2}\right)\frac{dm_{i+1/2}}{m_i'} + \left(u_{i-1/2}^{1/2} - u_i^0\right)\frac{dm_{i-1/2}}{m_i'}. \tag{C.5}$$

We now split $\Delta KE_{i,j}$ in terms of $dm_{i+1/2}$ and $dm_{i-1/2}$ to obtain,

$$\begin{aligned}\Delta KE_i = {} & dm_{i+1/2}\left(-u_i^0\left(u_{i+1/2}^{1/2} - \frac{1}{2}u_i^0\right) + \left(u_{i+1/2}^{1/2} - u_i^0\right)\frac{a_i}{2}\right)\\ & + dm_{i-1/2}\left(u_i^0\left(u_{i-1/2}^{1/2} - \frac{1}{2}u_i^0\right) + \left(u_i^0 - u_{i-1/2}^{1/2}\right)\frac{a_i}{2}\right).\end{aligned} \tag{C.6}$$

For exact kinetic energy conservation we require

$$\sum_i \Delta KE_i = 0. \tag{C.7}$$

However, we have

$$\sum_i \Delta KE_i = \sum_i dk_{i+1/2}dm_{i+1/2}, \tag{C.8}$$

where

$$\begin{aligned}dk_{i+1/2} = {} & \left(u_{i+1}^0 - u_i^0\right)\left(u_{i+1/2}^{1/2} - \frac{1}{2}\left(u_{i+1}^0 + u_i^0\right)\right) + \frac{1}{2}a_i\left(u_{i+1/2}^{1/2} - u_i^0\right)\\ & + \frac{1}{2}a_{i+1}\left(u_{i+1}^0 - u_{i+1/2}^{1/2}\right).\end{aligned} \tag{C.9}$$

This means that we have total energy conservation if we convert the lost kinetic energy into thermal energy by using

$$\sum_i \Delta(\epsilon\rho\tau)_{i+1} = +\sum_i dk_{i+1/2}dm_{i+1/2}. \tag{C.10}$$

The Lagrangian step followed by a remap onto the original Eulerian grid is not an operator-splitting algorithm. There is no time dependence at all in the remap step which is simply a geometrical rezoning of computational variables. It is for this reason only that we are allowed to enforce total energy conservation in this way. Indeed, all conservative schemes, i.e., schemes cast in the form of Eq. (16), which limit the momentum or velocity at shocks by enforcing total energy conservation will ensure that dissipated kinetic energy appears as thermal energy. Since we do not use conservative forms we must ensure that this happens by a different computational mechanism.

ACKNOWLEDGMENTS

This work was supported by a PPARC rolling grant at St. Andrews and the numerical results were produced using the JREI PPARC funded Compaq MHD Cluster in St. Andrews.

REFERENCES

1. M. Brio and C. C. Wu, *J. Comput. Phys.* **75**, 400 (1988).
2. D. Ryu and T. W. Jones, *Astrophys. J.* **442**, 228 (1995).
3. W. Dai and P. Woodward, *Astrophys. J.* **494**, 317 (1998).
4. G. Toth, *J. Comput. Phys.* **161**, 605 (2000).
5. D. S. Balsara and D. Spicer, *J. Comput. Phys.* **148**, 133 (1999).
6. K. S. Holian, *T-4 Handbook of Material Properties Data Bases. Vol. Ic, Equations of State* (Los Alamos National Laboratory, LA-10160-MS-v.1C, 1984).
7. J. J. Quirk, *Int. J. Numer. Meth. Fluids* **18**, 555 (1994).
8. S. A. E. G. Falle, S. S. Komissarov, and P. Joarder, *MNRAS* **297**, 265 (1998).
9. G. Toth and D. Odsrcil, *J. Comput. Phys.* **128**, 82 (1996).
10. M. L. Wilkins, *J. Comput. Phys.* **36**, 281 (1980).
11. B. van Leer, *J. Comput. Phys.* **32**, 101 (1979).
12. W. F. Noh, *Methods in Computational Physics*, *Vol. 3*, *Fundamental Methods in Hydrodynamics* (Academic Press, New York, 1964), pp. 117–179.
13. D. J. Benson, *Comput. Methods Appl. Mech. Eng.* **99**, 235 (1992).
14. P. Woodward and P. Collela, *J. Comput. Phys.* **54**, 115 (1984).
15. C. R. Evans and J. F. Hawley, *Astrophys. J.* **332**, 659 (1988).
16. R. E. Peterkin Jr., M. H. Frese, and C. R. Sovinec, *J. Comput. Phys.* **140**, 148 (1998).
17. C. W. Hirt, A. A. Amsden, and J. L. Cook, *J. Comput. Phys.* **135**, 203 (1997).
18. J. U. Brackbill and W. E. Pracht, *J. Comput. Phys.* **13**, 455 (1973).
19. J. U. Brackbill, *J. Comput. Phys.* **96**, 163 (1991).
20. E. J. Caramana, M. J. Shashkov, and P. P. Whalen, *J. Comput. Phys.* **144**, 70 (1998).
21. D. L. Youngs, in *Numerical Methods in Fluid Dynamics*, edited by K. W. Morton and M. J. Baines (Academic Press, New York, 1982), p. 273.
22. P. Londrillo and L. Del Zanna, *Astrophys. J.* **530**, 508 (2000).
23. D. Ryu, F. Miniati, T. W. Jones, and A. Frank, *Astrophys. J.* **509**, 244 (1998).
24. W. Dai and P. Woodward, *J. Comput. Phys.* **115**, 485 (1994).