Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational Physics 217 (2006) 722–758

JOURNAL OF COMPUTATIONAL PHYSICS

www.elsevier.com/locate/jcp

A parallel explicit/implicit time stepping scheme on block-adaptive grids

Gábor Tóth [*], Darren L. De Zeeuw, Tamas I. Gombosi, Kenneth G. Powell

Center for Space Environment Modeling, University of Michigan, 2455 Hayward Street, Ann Arbor, MI 48109-2143, USA

Received 9 May 2005; received in revised form 23 December 2005; accepted 18 January 2006
Available online 7 March 2006

Abstract

We present a parallel explicit/implicit time integration scheme well suited for block-adaptive grids. The basic idea of the algorithm is that the time stepping scheme can differ in the blocks of the grid for a given time step: an explicit scheme is used in the blocks where the local stability requirement is not violated and an implicit scheme is used in the blocks where the explicit scheme would be unstable. The implicit scheme is second order in time. The non-linear system of equations is linearized with Newton linearization. The linear system is solved with a preconditioned Krylov subspace iterative scheme. The Schwarz type preconditioning is also based on the block structure of the grid. We discuss load balancing for parallel execution and the optimal choice of the time step for speed and robustness. The parallel efficiency of the scheme is demonstrated for the equations of magnetohydrodynamics with a geophysics application in three dimensions. The control of the numerical divergence of the magnetic field in combination with the explicit/implicit time stepping scheme is also discussed.
© 2006 Elsevier Inc. All rights reserved.

Keywords: Models; Numerical methods; Numerical approximation; Time dependent initial-boundary value problems; Parallel computation; Compressible fluids and gas dynamics; Magnetohydrodynamics

1. Introduction

Numerical modeling of disparate spatial and temporal scales presents a formidable challenge to computational physics. There has been a rapid development in numerical algorithms that can efficiently cope with such problems. Adaptive mesh refinement (AMR) is becoming a standard tool for resolving many orders of spatial scales. There are several variants of AMR grids: cell-based mesh refinement (e.g. [7]), block-based mesh refinement (e.g. [22]), and hierarchical patches (e.g. [5]).

In the cell-based AMR technique each cell can be refined individually. This technique is the most flexible in terms of adapting the grid to the features of interest in the computational domain. On the other hand the resulting unstructured grid is not optimal for loop optimization, it is difficult to use numerical schemes with wide stencils, and achieving good load balancing is complicated.

[*] Corresponding author. Tel.: +1 734 615 5362; fax: +1 734 647 3083.
E-mail address: gtoth@umich.edu (G. Tóth).

0021-9991/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2006.01.029

The hierarchical patch algorithm combines cells of the same refinement level into regular subgrids – patches. The equations are solved at all levels of the grid and the time step is usually taken to be proportional to the grid resolution. The patches interact via ghost cells and possibly via prolongation and restriction operators (e.g. to obtain error estimates, or in multi-grid solvers). Due to the regularity of the patches, this algorithm can use almost all schemes developed for uniform grids. On the other hand the varying size of the patches and the interaction of overlapping coarse and fine patches can make load balancing somewhat complicated.

The block-adaptive grid approach is very similar to the cell-based AMR, except that the smallest unit that can be refined is a block with a fixed number of cells (for example $4 \times 4 \times 4$ cells in 3D). When a 3D block is refined, it is split into eight octants (see Fig. 1). Each octant forms a block with the same number of cells as the original block, but the resolution is increased by a factor of two. The resulting grid structure is an octree of blocks, and the equations are solved at the finest level only, i.e. on the *leaves* of the tree. The time step is either proportional to the grid resolution, or it can be the same for all the blocks. The blocks are surrounded with ghost cells to simplify inter-block communication. Any scheme developed for a uniform grid can be used, although algorithms requiring a very wide stencil will need many ghost cell layers, which can be expensive. Since the blocks have the same number of cells, load balancing the parallel execution is quite simple, especially if the time step is uniform. Note that the number of blocks is typically much larger than the number of processors. The typical number of blocks is many thousands with sizes ranging from $4 \times 4 \times 4$ to $8 \times 8 \times 8$ cells depending on the application.

In this paper we concentrate on time integration algorithms designed for block AMR grids, although the schemes should carry over to AMR grids using hierarchical patches without any difficulty. Explicit time integration on a block AMR grid is relatively simple, especially when all the grid cells are advanced with the same time step. In many applications the number of cells (or blocks) at the finest spatial resolution dominates, thus using the same time step in all the cells is only moderately less efficient than advancing the blocks of different grid resolution with proportional time steps. Using uniform time steps reduces the complexity of the code and allows very simple load balancing: the blocks should be evenly distributed among the processors. Given the large number of blocks this can be done efficiently.

In many applications explicit time stepping can become very inefficient, because the time step can be limited by numerical stability to orders of magnitude smaller than the time step required by the dynamics of the system. Such stiff problems occur for hyperbolic as well as parabolic equations. If the ratio of the time steps required by accuracy and stability is large (say greater than one hundred) it may pay off switching to an implicit time integration scheme. Designing an efficient parallel implicit scheme for a block-adaptive grid is not a simple problem. There are many choices to be made that can affect the performance of the code.

In this paper we describe a fully implicit time integration scheme designed for block-adaptive grids. The algorithm is built up from well known ingredients (see e.g. the review [18]): three-level second-order

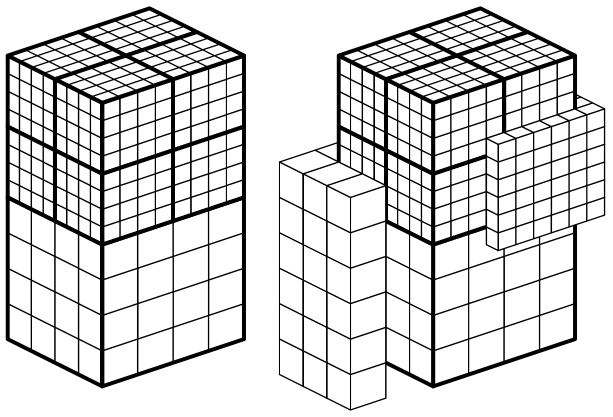

Fig. 1. The block-adaptive grid structure. Blocks are refined into eight octants (left panel). The blocks communicate via layers of ghost cells (right panel).

implicit time discretization, Newton linearization and Krylov subspace iterative solver using Schwarz type preconditioning. Although the building blocks of the algorithm are well known, there are dozens of options to choose from. We think it is valuable to describe which options work well together and result in an efficient scheme as well as to document what options were less successful for our applications. The choices are motivated by theoretical arguments as well as numerical experiments. In the design of the implicit time integration scheme we exploit the simplicity of the grid structure in the blocks as well as the natural decomposition of the grid into these blocks: our Schwarz type preconditioner is block-based, which provides a natural, simple and efficient preconditioning. Although implicit schemes are unconditionally stable according to the linear stability analysis, in reality large time steps can induce non-linear instabilities that can crash the code. We describe an adaptive time-step adjusting algorithm that can significantly improve the robustness of the implicit scheme.

We introduce an *explicit/implicit* time stepping algorithm especially suited for block-adaptive grids. We exploit the block structure of the grid, and advance blocks either with the explicit or the implicit time stepping scheme depending on the local numerical stability condition for a given block. With only a modest investment in coding, the explicit/implicit scheme can be significantly more efficient than the fully implicit scheme. By using the implicit scheme on a subset of the blocks, both the CPU and memory costs are significantly reduced. In addition, the upwind weighted explicit scheme can produce a more accurate solution for fast (close to one cell per time step) moving discontinuities and sharp gradients than the implicit scheme, thus accuracy is significantly improved if the explicit scheme is applied in the regions where fast moving discontinuities and sharp gradients occur. We describe how good load balancing can be achieved for the explicit/implicit scheme. Similar decomposition of the computational domain into explicitly and implicitly integrated regions has already been proposed (e.g. [26]), but not in the context of parallel block-AMR grids.

Our work has been motivated by space physics applications where we solve the equations of magnetohydrodynamics (MHD) to model the magnetosphere of the Earth (and other magnetized planets). The adaptive grid is well suited to resolve the computational domain near the Earth with cells sized to a fraction of the radius of the planet, while covering the computational domain that extends to hundreds of Earth radii. Due to the strong magnetic field and low plasma density near the Earth, the fast magnetosonic speed can approach the speed of light, which means that the explicit time step is limited to a small fraction (about a hundredth) of a second. On the other hand the dynamic time scales of the magnetosphere are typically longer than 5 s. Such a problem is ideally suited for the implicit and explicit/implicit time stepping algorithm. However, the algorithms described in this paper are very general, and there are certainly many other applications where they can be used successfully.

The implicit and explicit/implicit schemes have been implemented in the BATSRUS code [22] which solves the MHD equations with second-order shock-capturing total variation diminishing (TVD) schemes on a block-AMR grid. The explicit schemes in the BATSRUS code achieve excellent scaling on distributed memory parallel super-computers and it has been a high priority to get a good parallel scaling for the implicit scheme as well. To the best of our knowledge this is the first publication about a fully implicit parallel MHD code that works on an AMR grid. On the other hand there are several MHD AMR codes using explicit time stepping (e.g. [3,10,13,15–17,23,25,36,37]). Therefore it is of interest to investigate the parallel performance of the Jacobian-free Newton–Krylov–Schwarz (NKS) methods for 3D MHD problems on AMR grids. Among our applications the implicit time stepping is found to be the most valuable for time-accurate integrations of the MHD equations. This is different from the more typical applications of implicit schemes, where time-marching towards a steady state or solution of elliptic problems is considered. Our choices among the dozens of options in the NKS algorithm are geared towards the time-accurate application, and are different from the choices optimal for steady-state problems. We hope that this paper will provide useful information for others implementing the NKS approach into their AMR MHD codes. The explicit/implicit scheme in BATSRUS has already been successfully used to model the magnetospheres of Earth, Saturn and Uranus [33,12,32].

In Section 2 we describe the explicit, implicit and explicit/implicit time stepping algorithms. The control of the numerical errors in the divergence of the magnetic field is also discussed. Simple numerical tests as well as our typical geophysics applications are presented in Section 3. The tests show substantial speed up of the implicit and explicit/implicit schemes relative to the explicit time integration. Good parallel scaling is demonstrated up to hundreds of processors. We conclude in Section 4.

2. Algorithms

In this section we are mostly concerned with the temporal discretization. For this reason let us take the spatially discretized form of the original partial differential equations. This semi-discretized form can be written as the system of differential equations

$$\frac{\partial \mathbf{U}}{\partial t} = \mathbf{R}(\mathbf{U}), \tag{1}$$

where t is time, $\mathbf{U}$ is the vector of the spatially discretized dependent variables for all the grid cells, and $\mathbf{R}$ is a non-linear function of $\mathbf{U}$. The right-hand side may also depend explicitly on time (not shown). Note that in this semi-discretized form the spatial derivatives of $\mathbf{U}$ are approximated with algebraic functions of $\mathbf{U}$.

2.1. MHD equations

Most of the discussion in this section does not depend on the actual system of equations, and the algorithm should be applicable to a wide class of systems of partial differential equations. Since our applications involve the solution of the MHD equations, certain aspects of the discussion are specific to the MHD equations, which are briefly described here. In general the MHD equations can be written in many equivalent forms, but for solutions involving shock waves the conservative form is preferred. The vector of conservative variables in each grid cell is $\mathbf{U}_i = (\rho, \rho\mathbf{v}, e, \mathbf{B})$, where ρ is mass density, $\rho\mathbf{v}$ is momentum density, e is total energy density and $\mathbf{B}$ is the magnetic field vector. The units of $\mathbf{B}$ can be chosen such that the vacuum magnetic permeability is unity. Then the MHD equations can be written in a conservative form as

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho\mathbf{v}), \tag{2}$$

$$\frac{\partial \rho\mathbf{v}}{\partial t} = -\nabla \cdot (\mathbf{v}\rho\mathbf{v} - \mathbf{B}\mathbf{B}) - \nabla p_{\text{tot}}, \tag{3}$$

$$\frac{\partial e}{\partial t} = -\nabla \cdot (\mathbf{v}e + \mathbf{v}p_{\text{tot}} - \mathbf{B}\mathbf{B}\cdot\mathbf{v} - \mathbf{B} \times \eta\mathbf{J}), \tag{4}$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times (-\mathbf{v} \times \mathbf{B} + \eta\mathbf{J}). \tag{5}$$

The spatially discretized right-hand sides of the above equations correspond to $\mathbf{R}(\mathbf{U})$ in (1). The velocity $\mathbf{v}$, the current density $\mathbf{J} = \nabla \times \mathbf{B}$, and the total pressure $p_{\text{tot}} = p + \mathbf{B}^2/2$ together with the thermal pressure

$$p = (\gamma - 1)\left(e - \frac{1}{2}\rho\mathbf{v}^2 - \frac{1}{2}\mathbf{B}^2\right) \tag{6}$$

are derived quantities. The equation parameters are the adiabatic index γ and the resistivity η. In ideal MHD the resistivity is taken to be $\eta = 0$. An additional constraint on the MHD solution is that the magnetic field is divergence free:

$$\nabla \cdot \mathbf{B} = 0. \tag{7}$$

This equation can be regarded as a constraint on the initial condition, since the induction equation (5) guarantees that $\partial(\nabla \cdot \mathbf{B})/\partial t = 0$. The numerical solution of the induction equation, however, may not conserve $\nabla \cdot \mathbf{B}$ exactly.

2.2. Explicit time stepping

We use a simple two-stage predictor–corrector scheme for the explicit time stepping

$$\mathbf{U}^{n+1/2} = \mathbf{U}^n + \frac{1}{2}\Delta t_n \mathbf{R}(\mathbf{U}^n), \tag{8}$$

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t_n \mathbf{R}(\mathbf{U}^{n+1/2}), \tag{9}$$

where the superscript n shows the time step, the nth time step is of length Δt_n and the index $n + 1/2$ refers to the time half way between t_n and $t_{n+1} = t_n + \Delta t_n$.

We could use almost any other explicit time integration scheme, but there are some differences in terms of efficiency. The larger the time step allowed by the explicit scheme, the faster the combined explicit–implicit scheme becomes. To see this, we compare the efficiency of the above predictor–corrector scheme, which is stable up to CFL number 1.0, with the Adams scheme

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t\left[\frac{3}{2}\mathbf{R}(\mathbf{U}^n) - \frac{1}{2}\mathbf{R}(U^{n-1})\right] \tag{10}$$

which is stable up to CFL number 0.5 only. For a fully explicit time stepping the two schemes have about the same efficiency, because the Adams scheme requires a single evaluation of the right-hand side, while the predictor–corrector scheme requires two evaluations.

For the explicit/implicit time stepping, however, the predictor–corrector scheme allows more blocks to be treated with the explicit time stepping than the Adams scheme. Since the implicit time stepping is much more expensive than the explicit scheme, the more blocks that can be advanced with the explicit time stepping, the more efficient the overall scheme is. Consequently, the explicit scheme with the larger stability region is preferred.

2.3. Implicit time stepping

The implicit time discretization uses a three-level scheme

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t_n\left[\beta\mathbf{R}(\mathbf{U}^{n+1}) + (1-\beta)\frac{\mathbf{U}^n - \mathbf{U}^{n-1}}{\Delta t_{n-1}}\right], \tag{11}$$

where $\beta = (\Delta t_n + \Delta t_{n-1})/(2\Delta t_n + \Delta t_{n-1})$. For constant time steps $\beta = 2/3$ and the equation above simplifies to the second-order backward differentiation formula (BDF2) [11]. Unfortunately, for large time steps (CFL number much larger than 1) the second and higher order implicit Runge–Kutta and multistep schemes do not preserve the TVD property of the one stage scheme [9], but for time-accurate simulations a second-order implicit scheme is still more accurate than the first order and unconditionally TVD backward-Euler method, which corresponds to $\beta = 1$.

The state vector $\mathbf{U}^{n-1}$ from the previous time step is not always available, for example in the first time step. When the grid is refined or coarsened, one could prolongate or restrict $\mathbf{U}^{n-1}$ to the newly created grid, but this may not be an accurate approximation. When the previous time step information is not available or reliable, a two-level scheme can be used:

$$\mathbf{U}^{n+1} = \mathbf{U}^n + \Delta t_n\left[\beta\mathbf{R}(\mathbf{U}^{n+1}) + (1-\beta)\mathbf{R}(\mathbf{U}^n)\right] \tag{12}$$

with $\beta \geqslant 1/2$ for stability. Since this scheme is only applied in a few time steps, second-order accuracy (which corresponds to $\beta = 1/2$) is not crucial, thus we typically use the most stable backward-Euler scheme with $\beta = 1$.

2.3.1. Newton linearization

Eq. (11) is a large system of non-linear equations for the unknowns $\mathbf{U}^{n+1}$. Instead of solving this non-linear system, we linearize it by approximating $\mathbf{R}(\mathbf{U}^{n+1})$ as

$$\mathbf{R}(\mathbf{U}^{n+1}) = \mathbf{R}(\mathbf{U}^n) + \frac{\partial\mathbf{R}}{\partial\mathbf{U}}\cdot(\mathbf{U}^{n+1} - \mathbf{U}^n) + \mathcal{O}(\Delta t^2) \tag{13}$$

which can be substituted into (11) to arrive at

$$\left[I - \Delta t_n\beta\frac{\partial\mathbf{R}}{\partial\mathbf{U}}\right]\cdot(\mathbf{U}^{n+1} - \mathbf{U}^n) = \Delta t_n\left[\beta\mathbf{R}(\mathbf{U}^n) + (1-\beta)\frac{\mathbf{U}^n - \mathbf{U}^{n-1}}{\Delta t_{n-1}}\right], \tag{14}$$

where I is the identity matrix and the $\mathcal{O}(\Delta t^3)$ terms are dropped. Note that if $\mathbf{R}$ depends explicitly on time (e.g. at the boundaries), then $\partial\mathbf{R}/\partial\mathbf{U}$ and $\mathbf{R}(\mathbf{U}^n)$ should be taken with arguments $\mathbf{U}^n$ and t_{n+1}. This linearization can also be regarded as the first Newton iteration towards the solution of the non-linear system. Although we have implemented the full Newton iteration, in our unsteady applications it is found to be more efficient to solve the linearized equation (14), or in other words, only a single Newton iteration is used. If the system of equations and the spatial discretization **R** are written in a conservative form, then the linearized equations are also conservative as long as the Taylor expansion is properly applied on the numerical flux functions [35].

The linearized system is a second-order accurate approximation of the non-linear system, which itself is a second-order approximation of the analytic solution at time level $n+1$. The solutions of the non-linear and the linearized systems have both $\mathcal{O}(\Delta t^2)$ errors (in addition to the truncation errors of the spatial discretization), therefore there is no reason to assume that the solution of the non-linear system is more accurate.

2.3.2. Krylov solvers with preconditioning

The linear system (14) can be written in a compact form as

$$A \cdot x = b, \tag{15}$$

where the matrix $A = (I - \Delta t_n \beta \partial \mathbf{R}/\partial \mathbf{U})$, the unknown $x = (\mathbf{U}^{n+1} - \mathbf{U}^n)$ and b is the right-hand side of (14). This linear system is solved with a preconditioned Krylov subspace solver such as GMRES [24] or BiCGSTAB [34], which are well suited for parallelization. We found that while the BiCGSTAB iterative scheme requires much less memory than GMRES, in our most challenging applications the GMRES scheme is more robust and efficient.

Allowing for both left-hand and right-hand side preconditioning, the matrix A and the right-hand side b are converted to A' and b', respectively, while the solution x' of the preconditioned equation

$$A' \cdot x' = b' \tag{16}$$

is converted back to x. The relation between the original and primed variables is the following:

$$A' = P_\mathrm{L} \cdot A \cdot P_\mathrm{R}, \tag{17}$$

$$b' = P_\mathrm{L} \cdot b, \tag{18}$$

$$x = P_\mathrm{R} \cdot x', \tag{19}$$

where P_L and P_R are the left and right preconditioning matrices, respectively.

In the Krylov subspace schemes the matrix A' is not needed explicitly, only its action on a vector is required. Due to the complexity of the block-adaptive grid structure, calculating, storing and multiplying with the non-zero matrix elements of A' would be quite complicated. Therefore the matrix–vector multiplication with $A' = P_\mathrm{L} \cdot A \cdot P_\mathrm{R}$ is done in three stages. First the vector is multiplied by P_R, which can be done efficiently provided that the preconditioner matrix is simple. Next we use a Jacobian-free evaluation for the multiplication with A, which exploits the fact that A is composed of an identity matrix and a Jacobian:

$$\left[I - \Delta t_n \beta \frac{\partial \mathbf{R}}{\partial \mathbf{U}}\right] \Delta \mathbf{U} = \Delta \mathbf{U} - \Delta t_n \beta \frac{\mathbf{R}(\mathbf{U}^n + \epsilon \Delta \mathbf{U}) - \mathbf{R}(\mathbf{U}^n)}{\epsilon} + \mathcal{O}(\epsilon), \tag{20}$$

where $\Delta \mathbf{U}$ is an arbitrary vector and ϵ is a small parameter. Finally, we multiply with the matrix P_L, which can be done efficiently if P_L is a well chosen preconditioner.

The ϵ parameter is typically taken to be around the square root of the round off errors. For double precision arithmetic the round off error in a simple arithmetic formula is around 10^{-15}, but the evaluation of **R** involves a lot of real arithmetic, so we take the square root of 10^{-12}, i.e. $\epsilon = 10^{-6}$. We note that it is also possible to use a second-order (in ϵ) symmetric difference formula to evaluate the matrix vector product. The second-order formula is twice as expensive as the above first-order formula, because in the symmetric difference **R** needs to be evaluated at $\mathbf{U}^n - \epsilon \Delta \mathbf{U}$ and $\mathbf{U}^n + \epsilon \Delta \mathbf{U}$ for every matrix vector product, while in the first-order formula $\mathbf{R}(U^n)$ does not depend on ΔU and can be reused. A second-order formula may be useful if single precision arithmetic is used, but for double precision arithmetic $\mathcal{O}(\epsilon)$ is much smaller than the other truncation errors.

The Jacobian-free evaluation offers a lot of flexibility in terms of the discretization of A. We find that in our applications it is not necessary or beneficial to use a spatially second-order discretization for A. It can be shown (see e.g. [14]) that the Jacobian matrix $\partial \mathbf{R}/\partial \mathbf{U}$ may be evaluated from a first-order spatial discretization, still the whole scheme remains second-order accurate in space and time. As observed by many authors (see [18] and references therein) it is advantageous to use a first-order upwind type spatial discretization for the Jacobian matrix, because it results in a more efficient and more robust scheme than a second-order evaluation of the Jacobian. This behavior is expected, since a first-order upwind weighted scheme has a stronger diagonal dominance than the second-order schemes (which tend to result in an anti-symmetric matrix for hyperbolic systems), and the Krylov subspace methods converge better for a diagonally dominant matrix. Since the preconditioner is also based on a first-order discretization, it preconditions more effectively for a first-order discretization of A. Finally, the first-order upwind schemes have a smoother $\mathbf{R}(\mathbf{U})$ function than second-order schemes applying non-smooth limiters. Both the first-order local Lax–Friedrichs and Harten–Lax–van Leer (HLL) schemes have differentiable $\mathbf{R}(\mathbf{U})$ functions. The first-order Roe scheme without an entropy fix is non-differentiable where one or more of the wave speeds are zero, but this can be avoided by applying an entropy fix on all waves. The entropy fix replaces the absolute value of the wave speed with a smooth parabola around zero.

On the other hand, we also found that using the first-order variant of the second-order scheme that is used to evaluate $\mathbf{R}(\mathbf{U}^n)$ in (14) is more accurate than using a different scheme. This is again expected, since at sharp gradients the second-order TVD type scheme automatically reduces to the first-order variant, and thus using the same scheme (although different order) in $\mathbf{R}$ and A provides a better match than using different schemes.

The Krylov solver starts from an arbitrary initial guess for x. One may hope that using a good initial guess will reduce the number of iterations. For example one can use the explicit scheme to estimate the solution, or use the solution obtained in the previous time step. We found that starting with the initial guess $x = 0$ is just as good if not better than the other alternatives.

For time-accurate calculations it is neither necessary nor efficient to solve the linearized equations to machine accuracy. In our typical applications it is sufficient to reduce the L2 norm of the residual by a factor of 1000. We have experimented with varying the tolerance (which is an adjustable parameter), and found that requiring a smaller residual increases the number of Krylov iterations without significantly changing the solution as compared to the truncation errors at a practically achievable grid resolution. Using a more relaxed tolerance led to accuracy and robustness problems. The optimal choice of the tolerance is problem dependent. With good preconditioning a factor of 1000 reduction requires typically 10–30 Krylov iterations.

Note that the errors in the different variables can only be added up after normalizing the components of U individually. We use a normalization where each variable has the L2 norm equal to unity. For example, the density variable ρ is normalized to $\rho/\bar{\rho}$ where

$$\bar{\rho}^2 = \frac{1}{N}\sum_{i=1}^{N}\rho_i^2 \tag{21}$$

and the summation is done for all the grid cells where the implicit scheme is applied. This normalization is essential for making the linear solver work.

2.3.3. Schwarz preconditioner

Most preconditioners require the calculation of the matrix elements of some approximations of A. For sake of parallel efficiency we use a Schwarz type preconditioner, using only local data that is available on the processor without communication. The block structure of the grid offers a natural choice: each implicitly integrated block is preconditioned individually. This choice has two advantages:

- The preconditioner does not depend on the parallel distribution of data, since the data in a grid block are guaranteed to be on the same processor.
- The simple structure of the grid block results in a simple matrix, which can be easily and efficiently preconditioned.

While including relatively limited information in the preconditioner may reduce the convergence rate of the Krylov solver, using a simple and fully parallel preconditioner reduces the time spent on calculating and applying the preconditioner matrices. We can investigate these competing effects by changing the block size of the block-adaptive grid, which results in more/less accurate/expensive preconditioner.

The preconditioner matrices are based on the Jacobian matrix A restricted to an individual grid block, which is denoted by $\bar{A}$. Using a first-order discretization for $\bar{A}$ results in a block-heptadiagonal matrix. A higher order discretization would require the calculation of many more matrix elements, and it would also take more computation to calculate the preconditioner from it. Moreover, if A is evaluated as a first-order discretization in (20), it is best to base the preconditioner on the same first-order scheme. In the current implementation the Jacobian matrix is always based on the first-order local Lax–Friedrichs scheme. This is optimal if $\mathbf{R}$ is evaluated with the second-order TVD Lax–Friedrichs scheme, but it may not be optimal for other second-order schemes.

The matrix elements of A can be obtained in various ways. One can take analytic derivatives of the discretized equations (e.g. [35]) or use perturbations of the stencil and calculate the elements with numerical derivatives (e.g. [14]). We use an intermediate approach (also shown in [14]), in which the matrix elements are calculated by analytic differentiation of the discretized equations at the level of fluxes and source terms, but the derivatives of the fluxes and source terms are calculated by numerical differentiation.

The semi-discrete equation (1) is assumed to be in the following form:

$$\frac{\partial \mathbf{U}}{\partial t} = -\nabla \cdot \mathscr{F}(\mathbf{U}) + \mathscr{S}(\mathbf{U}), \tag{22}$$

where $\mathscr{F}$ and $\mathscr{S}$ are the discretized fluxes and source terms, respectively. The source term may be local ($\mathscr{S}_i$ only depends on $\mathbf{U}_i$) or it may contain spatial derivatives of $\mathbf{U}$. For the first-order local Lax–Friedrichs discretization the above equation in 1D is

$$\frac{\partial \mathbf{U}}{\partial t} = -\frac{\mathbf{F}_{i+1} - \mathbf{F}_{i-1}}{2\Delta x} + \frac{c^{\max}_{i+1/2}(\mathbf{U}_{i+1} - \mathbf{U}_i) - c^{\max}_{i-1/2}(\mathbf{U}_i - \mathbf{U}_{i-1})}{2\Delta x} + \mathbf{S}_i, \tag{23}$$

where $\mathbf{F}_i$ and $\mathbf{S}_i$ are the analytic flux and source functions of $\mathbf{U}_i$, respectively, and $c^{\max}_{i+1/2}$ is the maximum propagation speed estimated for the cell interface. For the MHD equations $c^{\max} = |v_x| + c_{\text{fast}}$, where v_x is the flow speed and c_{fast} is the fast magnetosonic speed. The Jacobian matrix elements are obtained by taking the partial derivatives of the right-hand side of the above equation with respect to the flow variables. Assuming that the source terms are local, the main diagonal and off-diagonal block elements are:

$$\begin{aligned}
\frac{\partial R^u_i}{\partial U^w_i} &= \frac{\partial S^u_i}{\partial U^w_i} - \frac{1}{2\Delta x}(c^{\max}_{i+1/2} + c^{\max}_{i-1/2})\delta_{u,w}, \\
\frac{\partial R^u_i}{\partial U^w_{i+1}} &= -\frac{1}{2\Delta x}\frac{\partial F^u_{i+1}}{\partial U^w_{i+1}} + \frac{1}{2\Delta x}c^{\max}_{i+1/2}\delta_{u,w}, \\
\frac{\partial R^u_i}{\partial U^w_{i-1}} &= +\frac{1}{2\Delta x}\frac{\partial F^u_{i-1}}{\partial U^w_{i-1}} + \frac{1}{2\Delta x}c^{\max}_{i-1/2}\delta_{u,w},
\end{aligned} \tag{24}$$

where the u and w indexes refer to specific components of $\mathbf{R}$ and $\mathbf{U}$, and $\delta_{u,w}$ is the Kronecker delta. Note that the derivatives of $c^{\max}$ were ignored, i.e. the time level is dropped from $n+1$ to n in the implicit discretization for this quantity [35]. The Jacobian for the first-order HLL scheme could be obtained in a very similar fashion: only the coefficients of the flux terms and the Kronecker deltas would change, and these again could be evaluated at time level n. The other off-diagonal elements can be obtained by replacing i with j (or k), x with y (or z) and the x component of the maximum propagation speed and the flux vector with the y (or z) component, respectively.

Numerical differentiation is used to take the partial derivatives of F^u_i and S^u_i, for example

$$\frac{\partial F^u_i}{\partial U^w_i} = \frac{F^u(\mathbf{U}_i + \epsilon\delta^w) - F^u(\mathbf{U}_i)}{\epsilon} + \mathscr{O}(\epsilon), \tag{25}$$

where δ^w is a unit vector with the only non-zero element in the wth element, and ϵ is a small parameter relative to the w component of $\mathbf{U}$. The arguments for and against using a second-order (in ϵ) central difference formula are the same as for Eq. (20). Another possibility is using analytic derivatives of the flux and source function. If the flux and source functions are complicated, the analytic derivatives can be quite complicated and more expensive to evaluate than calculating the derivatives with the numerical approximation shown above.

In the above equations it was assumed that the source term S_i only depends on U_i, i.e. it does not contain any spatial derivatives. In case a source term contains spatial derivatives, its contribution to the preconditioner can either be neglected, or the contributions to the main and off-diagonal elements should be properly calculated. In particular, the 8-wave solver [21] of the MHD equations involves source terms proportional to the discretized value of the divergence of the magnetic field. We found that the preconditioner works better if the contribution of these source terms to the Jacobian matrix is taken into account.

To construct the preconditioner matrices P_L and P_R, we use a relaxed modified block incomplete lower upper (MBILU) [1] factorization of $\bar{A}$. Here the *block* refers to the block-diagonal structure of the matrix, where the block size is determined by the number of flow variables. The *incomplete* means that the lower and upper triangular matrices have the same structure as the original matrix $\bar{A}$, and all other matrix elements of the lower and upper triangular matrices $\mathscr{L}$ and $\mathscr{U}$ are taken to be zero. Based on numerical experiments the value 0.5 for the relaxation parameter seems to be approximately optimal in our applications [20].

There are three possible ways to assign the incomplete lower-upper decomposition ($A \approx \mathscr{L} \cdot \mathscr{U}$) to the left and right preconditioning matrices

- pure left hand preconditioning: $P_L = \mathscr{U}^{-1}\mathscr{L}^{-1}$, $P_R = I$,
- pure right hand preconditioning: $P_L = I$, $P_R = \mathscr{U}^{-1}\mathscr{L}^{-1}$,
- symmetric preconditioning: $P_L = \mathscr{L}^{-1}$, $P_R = \mathscr{U}^{-1}$.

We have implemented all three possibilities and found that there is no significant difference between their cost or performance. We typically use the symmetric preconditioning.

2.3.4. Time-step adjustment

While the linear stability analysis suggests that the implicit time stepping scheme is unconditionally stable, the non-linearity of the equations can result in non-linear instability if the time step is increased too much. We have designed a time-step adjustment algorithm to maintain the time step as large as possible without crashing the code.

For the MHD equations the non-linear instability usually manifests itself as negative thermal pressure. The thermal pressure is calculated from the total energy density by subtracting the kinetic and magnetic energies according to Eq. (6). If thermal energy is a small fraction of the total energy, numerical errors can easily result in negative pressure. Occasionally the density may also become negative due to truncation errors. We use the minimum of the relative change in pressure p and density ρ as an indicator for stability:

$$Q = \min_i (p_i^{n+1}/p_i^n, \rho_i^{n+1}/\rho_i^n), \tag{26}$$

where the minimum is taken for all the grid cells and for the density and pressure variables. We use the following algorithm to adjust the time step

1. If $Q < 0.3$ then redo time step with $\Delta t'_n = \Delta t_n/2$.
2. If $0.3 \leqslant Q < 0.6$ then reduce the next time step to $\Delta\ t_{n+1} = 0.9\Delta t_n$.
3. If $Q > 0.8$ then increase the next time step to $\Delta\ t_{n+1} = \min(\Delta t_{\max}, 1.05\Delta t)$.
4. Otherwise use $\Delta t_{n+1} = \Delta t_n$.

If the first case is invoked, the algorithm is applied recursively, however the next time step is only allowed to be increased if there was no prior reduction in the current time step.

This algorithm is obviously not unique but it may be easily modified and adapted. The key features are the following: if the implicit time step results in a huge drop in pressure or density it is redone with a smaller time step. If the drop is moderate, the time step is kept, but the next time step is reduced. If the pressure and density drops are very small, then the next time step is slightly increased, but it should not exceed the accuracy time step limit set by $\Delta t_{\max}$. In our most challenging applications this algorithm has greatly improved the robustness of the implicit time integration scheme.

2.4. Explicit/implicit scheme

The basic concept of the explicit/implicit scheme is very simple. For a given time step, which is limited by accuracy requirements, some parts of the computational domain may allow the inexpensive explicit time integration, while other parts require the more expensive but stable implicit time integration scheme. The blocks of the AMR grid provide a natural way of decomposing the grid into explicit and implicit regions. Based on the stability condition appropriate for a block it is either advanced explicitly or implicitly. The algorithm proceeds as follows:

1. Set the time step based on accuracy, efficiency and robustness requirements.
2. Assign blocks to be explicit or implicit based on the local stability condition.
3. Load balance explicit and implicit blocks.
4. Advance explicit blocks in time.
5. Update ghost cells for implicit blocks.
6. Advance implicit blocks in time.
7. Update ghost cells for all blocks.

Load balancing is done by evenly distributing the explicit as well as the implicit blocks among the processors. We use a Peano–Hilbert space filling curve to order the explicit and implicit blocks into two separate ordered lists. Both lists are split into as many equal pieces as the number of processors. For any two processors the number of explicit (and implicit) blocks can differ by at most one. Using the Peano–Hilbert ordering ensures good data locality, which means that the inter-processor communication is minimized. Although doing load balancing every time step looks expensive, in practice it turns out to require a relatively small fraction of the total execution time. In a typical application the blocks do not change from explicit to implicit or implicit to explicit too often. Even when they change, only a few blocks need to be moved around.

2.4.1. Optimal time step

An interesting aspect of the scheme is the optimal choice of the time step. For an explicit scheme the optimal time step is the largest Δt allowed by the numerical stability. For a fully implicit scheme the optimal time step is the largest Δt allowed by the accuracy requirements and non-linear stability conditions. In case of the explicit/implicit scheme the time step is limited by accuracy and non-linear stability requirements, but the largest allowed time step may not be the optimal one for speed. This is quite easy to see: the larger the time step is, the more blocks require implicit time stepping; this requires more computation (one needs to solve a larger linear system), which can reduce the performance. The optimal time step is application dependent, and usually there is a relatively flat minimum in the time step/performance curve, which can be determined from a few runs, or it may be determined adaptively as the code runs.

To make the explicit/implicit scheme robust, we apply the time-step adjustment algorithm described in Section 2.3.4. The maximum time step parameter $t_{\max}$ should be set to the optimal time step for the explicit/implicit time integration scheme, which is either fixed or dynamically calculated during the run. In case the time step needs to be redone with a reduced time step, the block assignment to explicit and implicit time integration is also redone together with the load balancing. Since a reduced time step usually results in fewer implicit blocks, the penalty in terms of execution time is less severe than for the fully implicit scheme.

2.4.2. Order of accuracy

The explicit/implicit time stepping scheme is second-order accurate in time as long as the explicit and implicit algorithms are both second-order accurate and the boundary conditions between the explicit and implicit blocks are properly handled. When the explicit blocks are advanced forward in time, they need ghost cell information at time level n, and at time level $n + 1/2$. To provide the latter one needs to perform the explicit predictor step (8) in the implicit blocks next to the explicit blocks, provide ghost cell information to the explicit blocks and then reset the variables in the implicit blocks to time level n. This is a slightly complicated but not an expensive procedure. The alternative is to use a single step explicit scheme or to have sufficient ghost cell data in the explicit blocks for the predictor corrector scheme. The latter solution is not really practical for a block AMR grid, because increasing the number of ghost cell layers can drastically increase the memory usage.

In our original implementation the explicit cells at the explicit/implicit boundary used time level n ghost cell information only, so formally those cells are advanced by a temporally first-order algorithm. We have not found any adverse effects due to this simplification, probably because the errors are dominated by the spatial discretization errors in our magnetospheric applications. More recently we have implemented the first algorithm described in the previous paragraph, which results in a fully second order algorithm. The first- and second-order explicit/implicit boundary handlings will be compared in the numerical tests.

When the implicit blocks are advanced, their ghost cells adjacent with the explicit domain are already set to time level $n+1$, which is exactly what is needed for stability and second-order time accuracy.

2.4.3. Conservative properties

If both the explicit and implicit schemes are conservative, it is possible to make the explicit/implicit scheme fully conservative by applying a flux correction step at the interfaces between the explicit and implicit blocks. For example the explicit face fluxes can be replaced by the implicit face fluxes. In order to apply the flux correction, the numerical flux of the implicit scheme has to be calculated. The implicit update can be written in a conservative form as

$$\mathbf{U}_i^{n+1} = \mathbf{U}_i^n - \Delta t \frac{\mathscr{F}_{i+1/2}^{n+1} - \mathscr{F}_{i-1/2}^{n+1}}{\Delta x} + \Delta t \mathscr{S}_i^{n+1}, \tag{27}$$

where the numerical flux for the linearized three-level implicit scheme is

$$\mathscr{F}_{i+1/2}^{n+1} = \beta \left[\mathscr{F}_{i+1/2}^{\text{high}}(\mathbf{U}^n) + \frac{\partial \mathscr{F}_{i+1/2}^{\text{low}}}{\partial \mathbf{U}} \cdot (\mathbf{U}^{n+1} - \mathbf{U}^n) \right] + (1-\beta)\mathscr{F}_{i+1/2}^n. \tag{28}$$

Here $\mathscr{F}^{\text{high}}$ is the spatially second-order numerical flux function, while $\mathscr{F}^{\text{low}}$ is the typically first-order numerical flux function. In the Jacobian-free approach the dot product of the Jacobian with $\Delta\mathbf{U} = \mathbf{U}^{n+1} - \mathbf{U}^n$ is obtained as

$$\frac{\partial \mathscr{F}_{i+1/2}^{\text{low}}}{\partial \mathbf{U}} \cdot \Delta\mathbf{U} = \frac{\mathscr{F}_{i+1/2}^{\text{low}}(\mathbf{U}^n + \epsilon\Delta\mathbf{U}) - \mathscr{F}_{i+1/2}^{\text{low}}(\mathbf{U}^n)}{\epsilon} + \mathscr{O}(\epsilon). \tag{29}$$

Once the explicit and implicit fluxes are calculated at the explicit/implicit interface, the flux correction step can be applied by correcting the state on the explicit side of the interface as

$$\mathbf{U}_i^{n+1'} = \mathbf{U}_i^{n+1} - \Delta t \frac{\mathscr{F}_{i+1/2}^{n+1} - \mathscr{F}_{i+1/2}^{\text{high}}(\mathbf{U}^{n+1/2})}{\Delta x}. \tag{30}$$

If there is a grid resolution change at the explicit/implicit interface, the fluxes on the fine side should be added up, and the correction should be applied on the coarse side of the interface. The above algorithm is quite complicated, therefore in our current implementation the flux correction step is not applied at the explicit/implicit interface. We do apply, however, the conservative flux correction at the resolution changes inside the explicit and implicit domains.

In our magnetospheric applications the maximum time step is selected such that the implicit scheme is restricted to the inner magnetosphere, which has very high Alfvén speeds, but there is no compressive discontinuity, while the bow shock in the outer magnetosphere is handled by the explicit scheme. This way the proper jump conditions are guaranteed. The proper jump conditions are also guaranteed if the bow shock is fully inside the implicitly handled domain. In general a fast (close to one cell per time step) moving discontinuity or sharp gradient is much better modeled by the upwind explicit method than by the implicit time stepping, which takes the fluxes from the next time step, in effect from the downwind direction. Thus the explicit/implicit scheme is not only more efficient in terms of memory and CPU usage than the implicit scheme, but it can also provide a more accurate solution. This will be demonstrated by the numerical tests in Section 3.2.

2.4.4. Memory requirements

The implicit algorithm requires much more memory than the explicit algorithm. The largest arrays to store are the preconditioner matrices and the set of Krylov vectors (if the GMRES solver is used). The preconditioner matrix consists of $N_{\rm impl}N_{\rm U}^2(2D+1)$ reals, where $N_{\rm impl}$ is the number of implicitly treated grid cells, $N_{\rm U}$ is the number of state variables for a single cell (8 for 3D MHD), and D is the number of spatial dimensions (3 for 3D MHD). The storage for the Krylov vectors is $N_{\rm impl}N_{\rm U}N_{\rm K}$ reals, where $N_{\rm K}$ is the maximum number of Krylov vectors stored. We typically use $N_{\rm K} = 100$ for the GMRES solver. Since the preconditioner matrices are applied during the Krylov iteration, the storage cannot be overlapped. Note, however, that neither the preconditioner nor the Krylov vectors are extended to the ghost cells, which is a factor of 8 savings for $4 \times 4 \times 4$ blocks. Still the implicit scheme needs much more memory than the explicit scheme.

The explicit/implicit scheme allows to reduce the number of implicit blocks to be lower than the total number of blocks, which can be important if the application is limited by memory constraints.

2.5. Divergence **B** control

When the multidimensional MHD equations are solved numerically, it is necessary to reduce or eliminate the error in the numerical value of $\nabla \cdot \mathbf{B}$. The following methods have been used in the literature to control the numerical error in $\nabla \cdot \mathbf{B}$:

- The *8-wave scheme* [21] adds source terms to the MHD equations so that magnetic monopoles are advected with the velocity of the plasma.
- The *diffusive control* [6] adds the gradient of $\nabla \cdot \mathbf{B}$ to the induction equations, so that the error in $\nabla \cdot \mathbf{B}$ is diffused away.
- The *hyperbolic correction* [6] solves an extra scalar equation to propagate the error in $\nabla \cdot \mathbf{B}$ with some preset speed, which is independent of the flow speed.
- The *projection scheme* [4] eliminates the $\nabla \cdot \mathbf{B}$ error after each time step by solving a Poisson equation $\nabla^2\phi = \nabla \cdot \mathbf{B}$ and projecting to a divergence free solution $\mathbf{B}' = \mathbf{B} - \nabla\phi$.
- The *constrained transport* (CT) scheme [8] and its numerous variants use a special discretization that conserves $\nabla \cdot \mathbf{B}$ to round off errors on a particular stencil. Note that the CT scheme can always be rewritten into an equivalent scheme based on the *vector potential variable* [19].

There is a huge literature on the advantages and disadvantages of these methods and here we refrain from making any comparisons. Only the possible extension of these schemes to the implicit and explicit/implicit time stepping on block AMR grids is discussed here.

The 8-wave scheme can be used without any difficulty for the implicit time integration. In fact Tóth et al. [29] found that the implicit scheme converges better if the Jacobian matrix contains the 8-wave source terms, even if another method, e.g. the projection scheme is used to eliminate the error in $\nabla \cdot \mathbf{B}$ after every implicit time step. The numerical experiments in this paper confirm this observation.

The diffusive control scheme can be implemented in two different ways: either $\nabla(\mu\nabla \cdot \mathbf{B}^n)$ is added to the induction equation at the same time as the curl of the electric field (unsplit implementation), or $\nabla(\mu\nabla \cdot \mathbf{B}^*)$ is added after the the update (split implementation), so that $\mathbf{B}^*$ already contains the contribution from the curl of the electric field. Here μ is a spatially varying coefficient that should be chosen to maximize the diffusion without introducing a numerical instability. For the explicit time stepping the split implementation allows a larger coefficient μ than the unsplit scheme. We note that the split implementation can be regarded as a single Jacobi iteration towards the solution of the Poisson problem in the projection scheme. In the implicit time stepping scheme we encountered a numerical instability for the unsplit implementation, although we have not analyzed the reason for this. The split implementation, on the other hand, works as expected.

The hyperbolic correction requires the solution of an extra equation with a new variable. Since this involves a substantial change in the data structure of the code, we have not implemented this scheme, and we cannot evaluate its applicability to implicit time stepping.

The projection scheme can be applied after every implicit time step the same way as after the explicit time steps. For the explicit time stepping the projection scheme can be rather expensive on a block-AMR grid.

While a conjugate-gradient based iterative Poisson solver requires about 20–30% of the total CPU time on a structured grid (e.g. [30]), in our experience it may use 80% or even more on a block-AMR grid due to the expensive communication via the ghost cells. More efficient Poisson solvers designed for AMR grids may result in a more efficient projection scheme.

For implicit time stepping the projection scheme is relatively more efficient, because the implicit update is more expensive than the explicit scheme while the Poisson solve costs the same. Both for the explicit and the implicit schemes there is a potential robustness problem: the projection scheme changes the magnetic field, therefore it changes the magnetic energy, thus the pressure derived from the total energy can become negative. Since the implicit scheme allows larger time steps and larger changes, the robustness problem may be more severe. Of course, this is very application dependent. The projection scheme, together with the 8-wave scheme in the Jacobian calculation, was used successfully in combination with implicit time stepping on structured grids [29].

The implementation of the constrained transport scheme for block-adaptive grids is fairly complicated [31,2]. The normal magnetic field components must be staggered to the cell faces to get consistent magnetic flux across resolution changes, and the electric fields must be evaluated at the edges of the 3D cells (or at the corners of the 2D cells). It is likely that a consistent generalization to the implicit time stepping scheme would present further complications.

One way to proceed, however, is to rewrite the CT scheme in terms of the vector potential **A**, which is staggered the same way as the electric field **E**. In fact, in the appropriate gauge, **A** is nothing else but the point-wise time integral of **E**. Once this change is done, the implicit time integration schemes presented in this paper can be directly applied, except for some extra complications in forming the approximate Jacobian for the preconditioner due to the staggering and the higher derivatives of **A**. Since the magnetic field is calculated as $\mathbf{B} = \nabla \times \mathbf{A}$, it remains divergence free. For the explicit/implicit time stepping scheme a consistent solution for the vector potential can ensure that the magnetic field remains divergence free across the boundary between the explicit and implicit domains. The BATSRUS code contains the usual face centered magnetic field implementation of the CT scheme [31], which works for explicit time stepping.

The vector potential formulation of the CT scheme is not implemented either for explicit or for implicit time stepping. For our magnetospheric applications the explicit CT scheme does not produce significant improvement in accuracy relative to the other divergence control schemes. Implementing an implicit version of the CT scheme would require very significant development effort and based on our experience with the explicit version, it would not work better than the implicit scheme with the other divergence control techniques.

3. Numerical tests

We present three sets of test results in this section. In all tests the equations of ideal MHD are solved. Although we have used the implicit and explicit/implicit schemes for resistive MHD problems too, the time step limitation due to resistivity was always less restrictive than the time step limit due to the fast magnetosonic waves. We have not studied any problems dominated by resistive effects.

The first test deals with the propagation of smooth sound and Alfvén waves. The problem is one-dimensional, but it is solved on a 3D AMR grid with one level of refinement. We demonstrate the second order of accuracy of the various time integration schemes by doing a grid convergence study. The boundary conditions are periodic, so that the conservation properties can be checked quantitatively. This test involves waves that propagate about 1 cell per explicit time step, so there is no performance gain with the implicit and explicit/implicit schemes relative to the explicit time integration. This simple test merely demonstrates the basic properties of the various algorithms.

The second set of tests involves the reflection of a sound wave on a magnetic discontinuity. The left half of the computational domain is dominated by magnetic pressure, and the Alfvén speed is 100, while the right half is dominated by thermal pressure, but the sound speed is only 1. Two sound waves are generated in the right half of the domain and one of the waves interacts with the magnetic discontinuity. This test problem represents an extreme idealization of magnetospheric simulations: the left half corresponds to the inner magnetosphere and the right half to the solar wind. In the real magnetosphere the solar wind flows around the magnetosphere and a bow shock is formed, where the flow becomes subsonic. Behind the bow shock there is another discontinuity, the magnetopause, which separates the solar wind from the inner magnetosphere dominated by the magnetic field of the planet. The magnetopause contains a sudden jump in the transverse magnetic field, thermal pressure and density, and it can be characterized as a tangential plus contact discontinuity. The bow shock cannot be realized in 1D, but the magnetopause can be represented by a discontinuity of the transverse field, thermal pressure and density.

This essentially 1D test is very easy to reproduce, since it has simple initial and boundary conditions. Still this simple test is suitable to demonstrate the accuracy, efficiency and robustness of the implicit and explicit/implicit time integration schemes. The main feature of this problem is the large (factor of 100) difference in the wave speeds in the two regions of the simulation domain. The evolution of the system is dominated by the slowly moving sound waves and contact discontinuities, thus the problem can be efficiently modeled with the implicit time integration scheme. Since a significant part of the computational domain does not contain the high speed wave at all, the explicit/implicit time integration can produce further improvement in the efficiency. Although the problem is simple, getting an accurate solution is not at all easy. The discontinuity in the magnetic field spreads out due to numerical diffusion, while the sound waves can produce oscillations or may diffuse away due to numerical errors. The details of the reflection depend sensitively on the numerical errors. Finally, the robustness of the scheme is well tested by the huge (factor of 8000) jump in the thermal pressure and density and the small (1.25×10^{-4}) ratio of the thermal and magnetic pressures in the left half of the domain.

The sound wave reflection problem will be solved with various schemes, with 1D and 3D codes on uniform and block AMR meshes. We will also rotate the problem with respect to the grid to demonstrate the use of the divergence **B** control schemes. All these simulations (except for the highest resolution runs) were performed on a workstation with a 2.1 GHz AMD CPU and the NAG f95 compiler using level 3 optimization.

The third part of this section presents some geophysical applications of the explicit/implicit time stepping scheme for magnetospheric simulations. This example is a 3D time dependent problem, which makes full use of the block-AMR grid as well as of the explicit/implicit time stepping scheme. We have been using the explicit/implicit scheme successfully in the BATSRUS code for several years [33,12,32], and achieved very significant speed-ups relative to the simple explicit time stepping scheme. These applications are typically run on 64–256 CPUs of a super computer. The scaling results reported here were obtained on the SGI 3800 and SGI Altix machines using the Intel ifort compiler with level 2 optimization.

Unless indicated otherwise we use the same options for the implicit solver in all the tests: the linearized second-order implicit discretization (14) is solved with the GMRES solver; the initial guess is zero; a symmetric Schwarz type MBILU preconditioning is used; the Jacobian-free matrix–vector multiplications employ a spatially first-order scheme; and the tolerance of the linear solver is set to $\tau = 0.001$.

3.1. Propagation of smooth waves

The one dimensional simulation domain is $-50 < x < +50$. When the simulation is run with a 3D code, the simulation domain is extended in the perpendicular directions to $-2 < y, z < 2$. The unperturbed initial state is uniform with $\rho = 1$, $v_x = 1$, $v_y = v_z = 0.01$, $B_x = 1.5$, $B_y = B_z = 0.01$ and $p = 0.625$. The adiabatic index is set to $\gamma = 1.6$ so that the sound speed is unity. The units of the magnetic field are chosen such that the magnetic permeability of vacuum is unity, so the Alfvén speed is $B_x/\sqrt{\rho} = 1.5$ in the x direction.

The uniform initial state is perturbed in the region $-30 = x_1 < x < x_2 = -10$. We use a smooth mask function

$$f(x) = \sin^2 \pi \frac{x - x_1}{x_2 - x_1} \tag{31}$$

so that the perturbed initial state and its gradient remain continuous. In the perturbed region $B_y = B_z = 0.01 + 0.001f$ and $p = 0.625 + 0.1f$. Furthermore, the density is modified according to an adiabatic compression to $\rho = (p/0.625)^{1/\gamma}$. The relative amplitudes of the magnetic and pressure perturbations are about the same, 10% and 16% respectively. On the other hand the energy density of the transverse magnetic field is four orders of magnitude weaker than the thermal energy density so that the sound wave is not affected by the transverse field. The initial state is shown in Fig. 2.

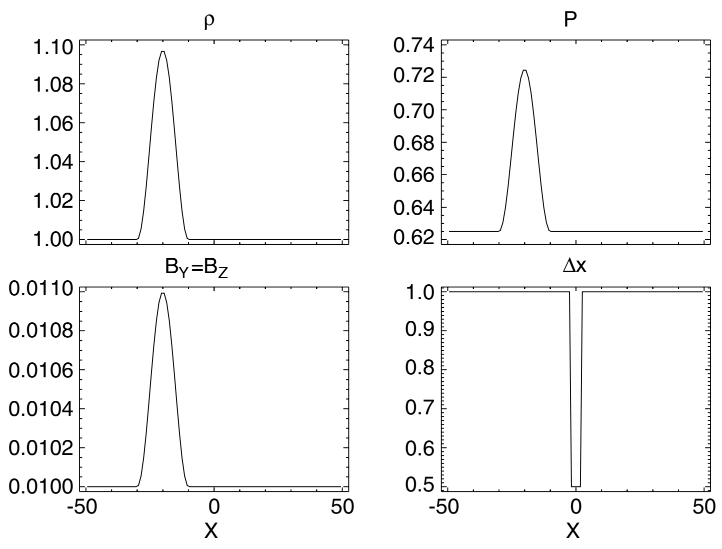

Fig. 2. The density, pressure, the Y and Z components of the magnetic field and the non-uniform grid resolution for the initial state of the smooth wave propagation problem.

The pressure and density perturbations result in two sound waves going left and right with the sound speed relative to the fluid. Since the bulk and sound speeds are both unity, one wave stays at rest, while the other moves towards the right at speed 2. The perturbations in B_y and B_z produce two Alfvén waves propagating to the left and right with speeds 0.5 and 2.5, respectively. The boundary conditions are periodic. The simulation is stopped at $t = 20$.

The reference solution is obtained with the Versatile Advection Code [28] configured to 1D with 1000 uniform grid cells and the TVD MUSCL scheme with a Roe-type approximate Riemann solver. The explicit predictor–corrector time integration scheme is used with the Hancock predictor step. In the reconstruction step the slopes of the primitive variables are limited with the monotonized central (MC) limiter. No entropy fix is applied in these runs. Fig. 3 shows the two sound waves and two Alfvén waves at the final time.

The rest of the test runs are done with the BATSRUS code using the same TVD-MUSCL Roe solver as for the reference solution. There are a few minor differences: (1) the predictor and corrector steps are identical in the two-stage Runge–Kutta scheme (8), (9); (2) an entropy fix is applied for all wave speeds; (3) the wave speed v of the divergence wave is replaced with the maximum wave speed $c^{\max}$ in the 8-wave solver [21]; and (4) the MC limiter

$$\mathrm{MC}(a,b) = 0.5(\mathrm{sgn}\,a + \mathrm{sgn}\,b)\min(\beta|a|,\beta|b|,0.5|a+b|) \tag{32}$$

is used with the $\beta = 1.2$ parameter instead of the maximum value 2 that leads to robustness problems on the AMR grid. Here sgn denotes the sign function. We note that implementing the MC limiter in the above form is quite efficient. The 3D scheme is dimensionally unsplit, i.e. fluxes in the three spatial dimensions are added at the same time. The BATSRUS code uses limited second order restriction and prolongation operators to fill in the ghost cells at the resolution changes.

The 3D AMR grid consists of adaptive blocks with $4 \times 4 \times 4$ cells. The middle part of the grid at $-2 < x < 2$ is resolved with twice the resolution as the rest of the grid. In the fully explicit and fully implicit simulations the fluxes through the resolution change are corrected by replacing the coarse level flux with the sum of the fine level fluxes. In the explicit/implicit runs the finer level is solved implicitly, while the coarse level is solved explicitly, thus no flux correction is applied. We will examine the consequences for conservation.

For the coarse grid the base resolution is $\Delta x = 1$. The block in the middle is refined into eight smaller blocks with $\Delta x = 1/2$, so the total number of blocks is 32. The medium and fine resolution grids have a base resolution $\Delta x = 1/2$ and $\Delta x = 1/4$, and the number of blocks increase to 256 and 2048, respectively. For the explicit run the time step is determined by the CFL condition. Using a CFL number of 0.9, the time step is $\Delta t = 0.0735$

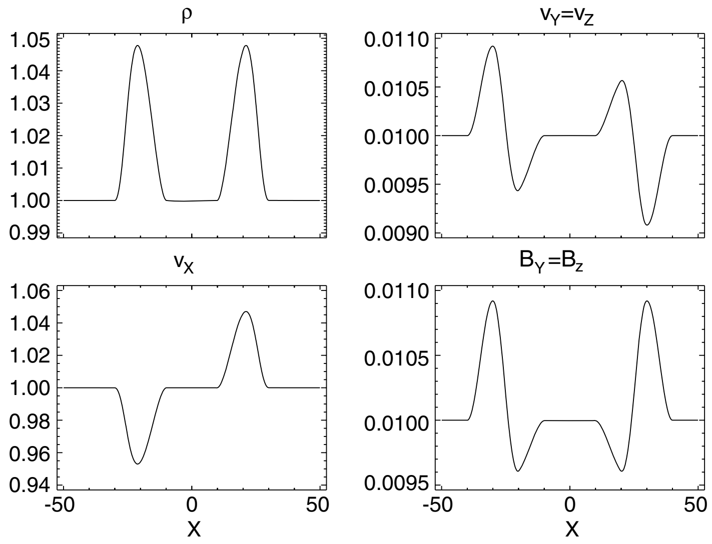

Fig. 3. The reference solution of the smooth wave propagation problem at the final time $t = 20$. The pressure is proportional to ρ^γ and $B_x = 1.5$ is constant.

for the coarse grid, half of that for the medium and quarter of that for the high resolution. The time step is set to $\Delta t = 0.12$ for the explicit/implicit runs on the coarse grid, so that the blocks outside the $-2 < x < 2$ region can be solved explicitly, while the blocks inside this region require the implicit time stepping. For the medium and high resolution runs the time step is reduced to 0.06 and 0.03, respectively. For sake of meaningful comparisons the fully implicit runs are done with the same time steps as the explicit/implicit runs.

For each run we calculate the average relative error. The simulation results are interpolated onto the grid of the reference solution (obtained with 1000 grid points) and the error is defined as

$$\delta = \frac{1}{7}\sum_{w=1}^{7}\frac{(1/n)\sum_i |W_i^w - \bar{W}_i^w|}{\max\limits_i(\bar{W}_i^w) - \min\limits_i(\bar{W}_i^w)}, \tag{33}$$

where i indexes the $n = 1000$ grid points of the reference solution, w is the index for the seven primitive variables ρ, v_x, v_y, v_z, B_y, B_z and p, and $\mathbf{W}$ and $\bar{\mathbf{W}}$ are the primitive state vectors for the interpolated and reference solutions, respectively. The denominator provides the amplitude of the propagating wave for the given variable. Note that B_x remains constant due to the divergence free condition in 1D, so it is omitted from the error calculation.

The results of the convergence study are summarized in Table 1. The explicit scheme converges at a nearly second-order rate as expected. The fully implicit time stepping also converges at a second-order rate, independent whether the first or the second-order Roe scheme is used in the Jacobian-free evaluation (20), or whether the linearized equation (14) are solved with a single Newton iteration or the non-linear equation (11) are solved with a full Newton scheme. The errors and convergence rates are shown by the last three rows of the table. Using the second-order scheme gives a slightly more accurate solution, in fact it is even more

Table 1
Average relative errors and convergence rates for the smooth wave test

Scheme	$\Delta x = 1$ (%)	$\Delta x = 1/2$ (%)	$\Delta x = 1/4$ (%)	Rate
Explicit	1.150	0.345	0.091	1.92
Expl./Impl./Roe1/first-order BC	1.220	0.493	0.221	1.15
Expl./Impl./Roe1/second-order BC	1.135	0.336	0.085	1.98
Implicit/Roe1	1.492	0.403	0.098	2.04
Implicit/Roe2	1.137	0.320	0.080	2.00
Implicit/Newton	1.134	0.318	0.079	2.01

accurate than the explicit scheme, which is somewhat surprising. Comparison of the last two rows of the table shows that solving the non-linear equation to a high accuracy instead of the linearized equation with the 0.001 tolerance hardly improves the results. In both runs the second order Roe scheme is used in the Jacobian-free evaluation. For the non-linear solution each Newton iteration decreases the residual by at least a factor of 10 (tolerance is set to 0.1), and the convergence criterion for the Newton solver is 10^{-7}, i.e. the L2 norm of the change should be less than 10^{-7} in the normalized variables defined by (21). Solving the non-linear equation accurately on the high resolution grid required three times more CPU time than solving the linearized problem with a less stringent tolerance. Most of the extra work is spent on calculating the preconditioner matrix in every Newton iteration (typically 4–5 were needed). The total number of Krylov iterations only differs by a factor of 1.7.

If the explicit/implicit scheme is used with a temporally first-order boundary condition at the explicit/implicit interface (i.e. the implicit blocks at the old time level are used as a boundary for the explicit domain) the convergence rate is 1.15 only. On the coarse grid the errors are only slightly larger than for the fully explicit or fully implicit schemes, but on the finest grid the difference is a factor of 2.5 or even more. On the other hand, if the two-stage explicit scheme is applied in the whole domain to provide second-order accurate fluxes at the explicit/implicit interface, the explicit/implicit time integration scheme reaches a second-order convergence rate and the errors are even slightly lower than for the fully explicit scheme.

For the fully explicit and fully implicit schemes the volume integral of all eight conserved variables are conserved within round-off error. In the explicit/implicit scheme the conservation is not perfect, but the relative error remains less than 10^{-4} for all the conserved quantities. As the resolution is increased, the error in the conservation is reduced. Note that accuracy of the solution, in terms of the primitive variables, is not affected by the lack of exact conservation.

3.2. Reflection of a sound wave on a magnetic wall

The computational domain is the same as used in the previous subsection. The initial velocity $\mathbf{v} = (-0.1, 0.1, 0.1)$ and the magnetic field components $B_x = B_z = 0.1$ are uniform. There is a discontinuity in B_y, ρ and p at $x = 0$. For $x < 0$ the magnetic pressure dominates with $B_y = 100$ and $p = 0.625$, while for $x > 0$ the thermal pressure dominates with $B_y = 0$, $p = 5000.625$. The density is set to $\rho = 1$ for $x < 0$, while $\rho = 8001$ for $x > 0$ so that the sound speed is unity everywhere ($\gamma = 1.6$). In the magnetic field dominated region the fast magnetosonic speed is approximately $B_y/\sqrt{\rho} = 100$. In addition to the discontinuity at $x = 0$, the pressure is increased by 10% to $p = 5500.6875$ in the region $25 < x < 30$. The initial state is shown in Fig. 4. In principle the boundary conditions should be based on the number of characteristics entering at the left and right boundaries, which is 3 and 6, respectively. Since not much is going through the boundaries during the test run, simple zero gradient boundary conditions are applied, and they work satisfactorily.

3.2.1. Reference solution

The initial state is in pressure equilibrium at $x = 0$, but the numerical diffusion will spread the initial discontinuity. The discontinuity is also advected by the flow speed $v_x = -0.1$. The pressure perturbation at $25 < x < 30$ generates two sound waves that propagate left and right and leave behind a reduced density but increased temperature region. The right propagating sound wave exits through the boundary. The left propagating sound wave reaches the region dominated by the magnetic field and interacts with it. A reference solution is obtained with the Versatile Advection Code on a 1D grid with 10,000 uniform grid cells using the same TVD MUSCL scheme as in Section 3.1.

Fig. 5 shows the reference solution at time $t = 10$. The initial perturbation in the pressure has split into two sound waves and a region of depleted density, which is separated by two contact discontinuities from the rest of the flow. The fronts of the sound waves are discontinuous jumps, but the back sides are slowly spreading rarefaction waves. The two fronts move with velocities -1.1 and 0.9, respectively. The small pressure jumps at $x = 24$ and $x = 29$ are physical and move at the same speed as the fronts.

The left moving sound wave reaches the discontinuity of the magnetic field at $t = 25$. In the numerical simulation the interaction starts a little bit earlier due to the numerical spreading of the discontinuities. Fig. 6 shows the X component of the velocity at four different times. At the interface the sound wave generates a

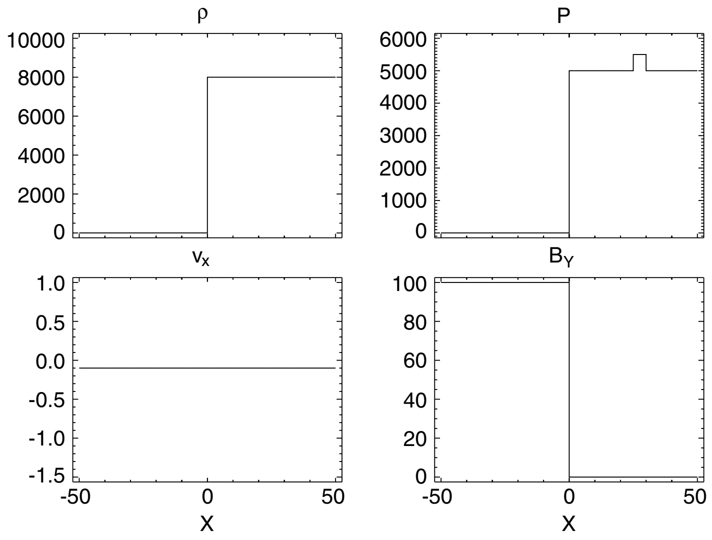

Fig. 4. The density, pressure, the X component of the velocity and the Y component of the magnetic field for the initial state of the sound wave reflection problem.

fast wave, which propagates at a velocity ≈100 and leaves the simulation domain through the left boundary. By $t = 35$ the velocity recovers to its initial value $v_x = -0.1$.

The simulation is stopped at $t = 35$. The final state is shown in Fig. 7. At this time the initial discontinuity in the magnetic field has been advected to $x = -3.5$ and the contact discontinuities are at $x = 21.5$ and $x = 26.5$, respectively. The right moving sound wave has left the simulation box through the right boundary. The originally left moving sound wave has been reflected by the discontinuity in the magnetic field and it is now traveling to the right with an inverted pressure profile.

3.2.2. Tests on a uniform 3D grid

In this subsection the test problem is solved on a uniform 3D grid with the BATSRUS code using the explicit, implicit and explicit/implicit time integration schemes. Doing the problem on a 3D grid allows us to make realistic efficiency comparisons between the schemes for real multi-dimensional applications. We will also use a coarse grid so that the truncation errors of the different schemes are obvious and easily comparable. The $-50 < x < 50$, $-2 < y, z < 2$ domain is resolved by a $100 \times 4 \times 4$ uniform grid, so the grid resolution is $\Delta x = 1$. The problem is solved with the same 3D TVD-MUSCL Roe scheme as in Section 3.1 except that the limiter is the β-limiter [27] with $\beta = 1.2$, which is a typical value used in BATSRUS.

Fig. 8 shows the solution at $t = 10$ obtained by the explicit time integration scheme. The discontinuity at $x = -1$ is quite well captured with three grid cells. The two sound waves are rather dissipated by the numerical diffusion, their amplitude in pressure is 217 instead of the correct 250. The difference in the slopes of the front and back edge of the waves is completely washed out. The velocity clearly shows the two sound waves, but in the $x < -1$ domain $v_x \approx -0.084$ instead of the correct -0.1 value. The velocity fluctuations are due to the waves produced by the initial widening of the discontinuity at $t = 0$. By the end of the simulation the reflected sound wave is very much damped but still visible in the density, pressure and velocity as shown in Fig. 9.

The time steps of the explicit simulation are limited by the CFL condition. With a CFL number of 0.9, the explicit time step is $\Delta t \approx 0.9\Delta x/(3c^{\max}) \approx 3 \times 10^{-3}$. The factor 3 in the denominator is due to the three spatial dimensions and the approximately isotropic distribution of the maximum propagation speed, which is $c^{\max} \approx 100$ in the magnetic field dominated region. The explicit simulation takes 458 CPU seconds when the grid consists of a single block, and only 445 s when the grid is split into 25 blocks of size $4 \times 4 \times 4$ cells. The improvement is due to the better compiler optimization and cache performance despite the extra work required for updating the ghost cells.

Next we solve the problem with the fully implicit time stepping scheme using time steps $\Delta t = 1/10$, 1/4, 1/2, 1 and 2. The obtained solutions are compared with the reference and the explicit solutions in Fig. 10.

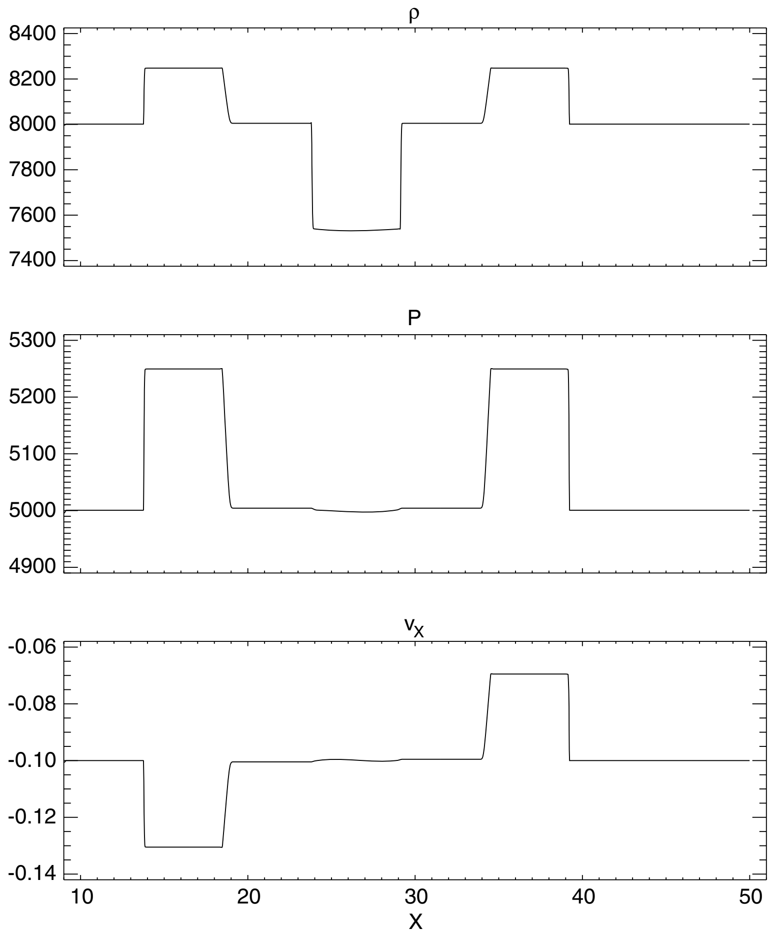

Fig. 5. The density, pressure and the X component of the velocity at time $t = 10$. Only the $10 < x < 50$ region is shown, which contains the two traveling sound waves at $x \approx 15$ and $x \approx 37$ and the two contact discontinuities in the middle.

The pressure is shown in the $10 < x < 50$ region. As the time step is increased the quality of the solution is degraded. For $\Delta t \geqslant 1$ the phase error and the undershoot error become rather significant. The solutions at the final time $t = 35$ are compared in Fig. 11. Note that the explicit (continuous line) and the implicit solution with $\Delta\ t = 1/4$ (dashed line) are practically identical.

It is interesting to compare the efficiency of two implicit runs with the same time step $\Delta t = 1/4$ and grid resolution but with different block sizes. The two runs use a single $100 \times 4 \times 4$ block vs. twenty five $4 \times 4 \times 4$ blocks, respectively. The run with the single large block has a better preconditioner matrix, so it requires fewer (2962) iterations in the Krylov solver than the run with the 25 smaller blocks (3272 iterations), but the difference is relatively small. In terms of execution times, the single block and the 25 block runs took the same time, because the smaller blocks result in a better compiler optimization. For the rest of this section we will use the grid consisting of $4 \times 4 \times 4$ blocks.

For $\Delta t = 1/4$ the implicit scheme finishes in 133s which is about 3.3 times faster than the explicit scheme. Further speed up can be achieved by increasing Δt (e.g. more than a factor of 11 for $\Delta t = 2$), but the accuracy of the solution becomes rather poor. Another way of speeding up the implicit scheme is to replace the first-order Roe scheme with the less expensive first-order Lax–Friedrichs scheme in the Jacobian-free evaluation of the matrix vector products. This has the additional advantage of perfectly matching the preconditioner, which is based on the same scheme. With this scheme the number of Krylov iterations drops to 2864 (a 14% saving) and the execution time becomes 1.4 times less, so this implicit scheme is 4.7 times faster than

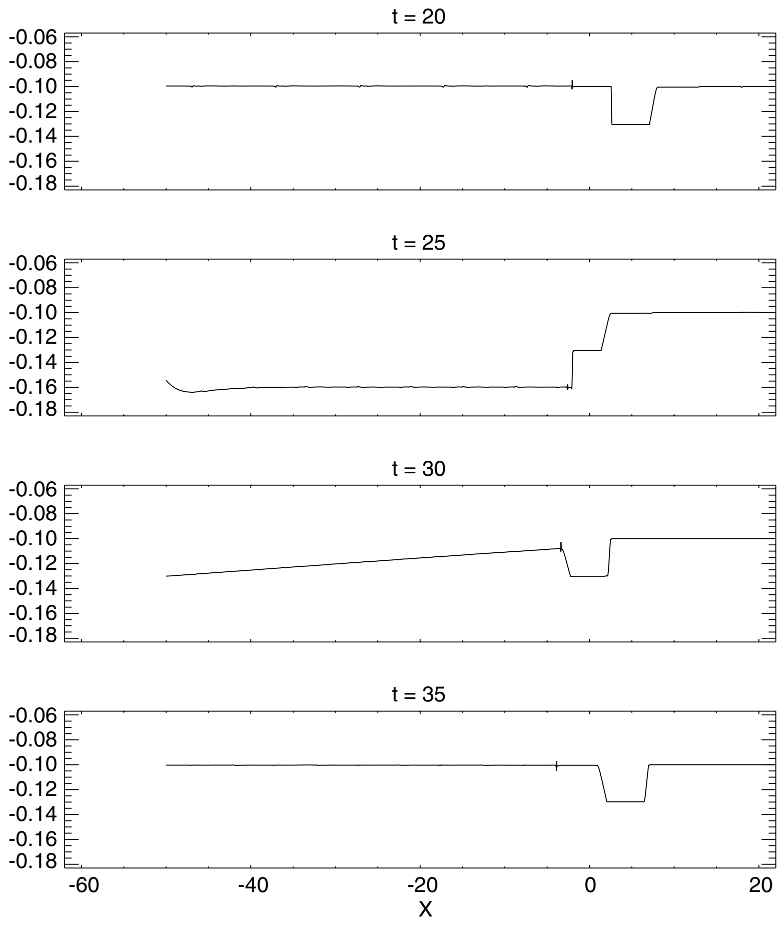

Fig. 6. The *X* component of the velocity before (top), during (middle) and after (bottom) the reflection of the sound wave in the reference solution. The small spikes are at the discontinuity in the magnetic field and are due to numerical errors.

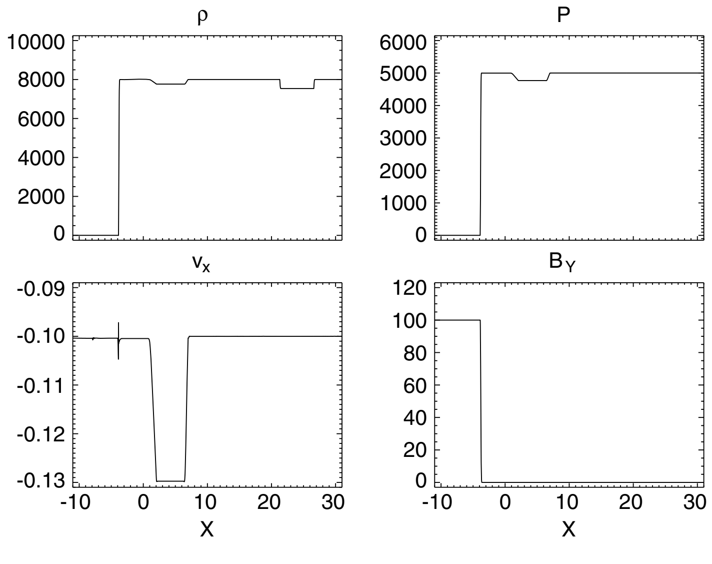

Fig. 7. The reference solution at $t = 35$ in the $-10 < x < 30$ region. The spike in the velocity at $x = -3.5$ is due to numerical error.

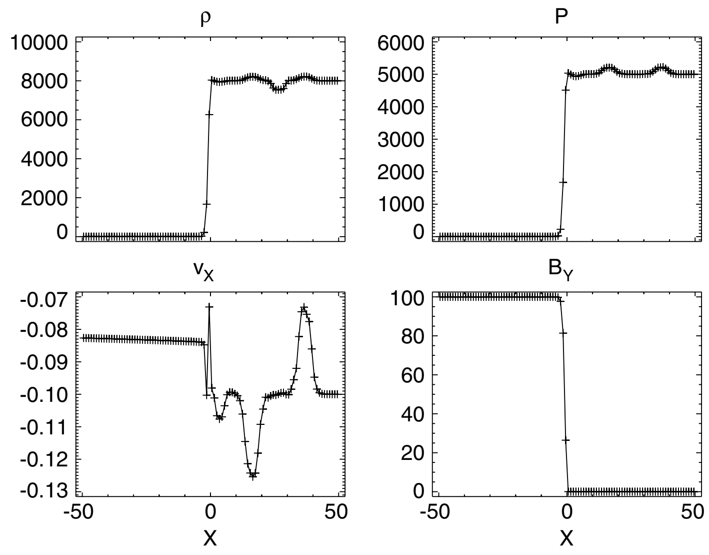

Fig. 8. The solution at $t = 10$ obtained with the explicit scheme on a uniform grid with $\Delta x = 1$ resolution.

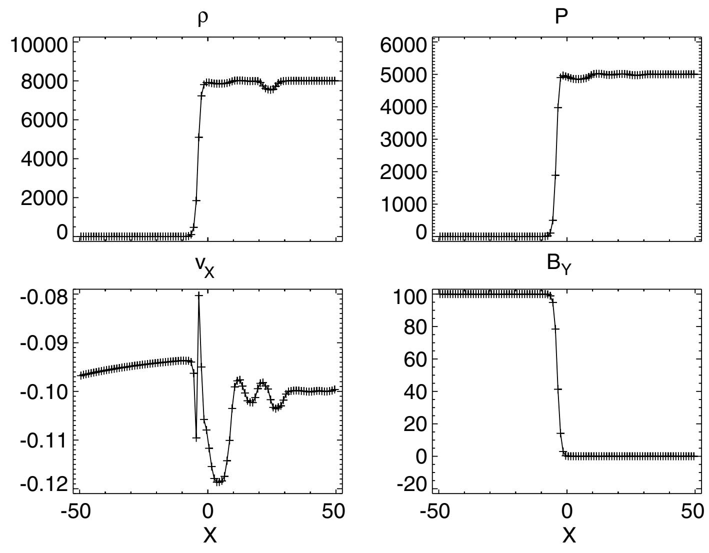

Fig. 9. The solution at $t = 35$ obtained with the explicit scheme on a uniform grid with $\Delta x = 1$ resolution.

the explicit scheme. Unfortunately the discontinuity at $x \approx 0$ becomes more smeared out and there is an undershoot in the B_y component of the magnetic field as shown in Fig. 11.

Using the explicit/implicit time stepping scheme can improve the efficiency of the simulation without sacrificing the accuracy of the solution. With $\Delta t = 1/4$ the blocks on the right half of the computational domain can be solved with explicit time stepping, so only half of the blocks require an implicit scheme. Due to the smaller number of unknowns, the number of Krylov iteration drops to 2929 (from 3272), and the execution time drops to 69 s (from 133s). The explicit/implicit scheme is therefore $445/69 \approx 6.4$ times faster than the explicit scheme, and the solution is practically the same (i.e. the difference between the two solutions are much smaller than the error relative to the reference solution).

Table 2 summarizes the results. The errors are taken in the $10 < x < 50$ interval at $t = 10$ to show how accurately the two sound waves are represented and for the whole grid at the final time $t = 35$ to show the accuracy

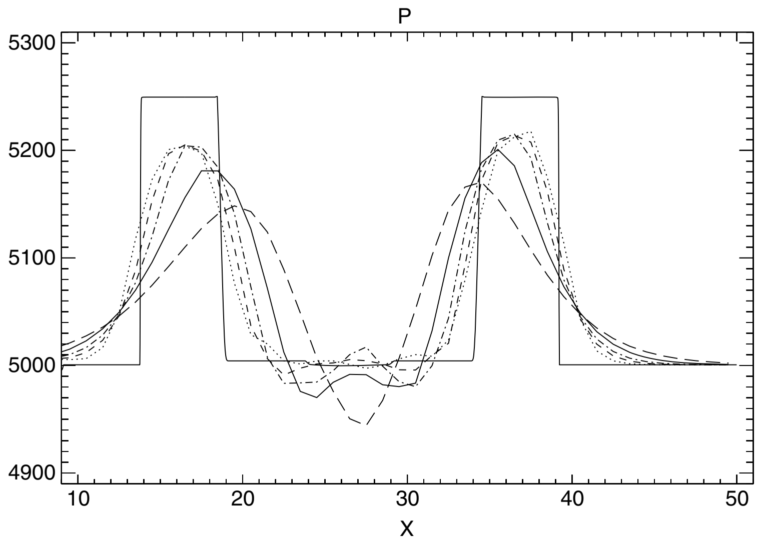

Fig. 10. The pressure at time $t = 10$ obtained by the reference solution (continuous line) on 10,000 grid points, the explicit scheme (dotted line) on 100 grid cells, and the implicit scheme with time steps $\Delta t = 1/4$, 1/2, 1 and 2 (short dashed, dot-dashed, triple-dot-dashed and long dashed lines, respectively).

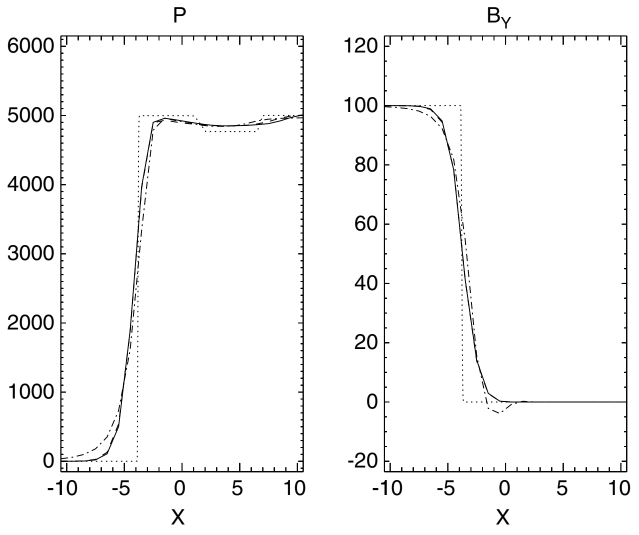

Fig. 11. The pressure and the Y component of the magnetic field at the final time $t = 35$ obtained by the reference solution (dotted line), the explicit scheme (continuous line) on 100 grid cells, and the implicit scheme with time step $\Delta t = 1/4$ using the Roe (dashed line) or Lax–Friedrichs (dot-dashed line) flux functions in the Jacobian-free evaluation. Note that the continuous and dashed lines are very close.

of the contact discontinuity at the magnetic wall and the velocity on the left half of the domain. The error for some variable W is calculated by interpolating the simulation results onto the grid of the reference solution (obtained with 10,000 grid points) and taking

$$\delta(W) = \frac{(1/n)\sum_i |W_i - \bar{W}_i|}{\max_i(\bar{W}_i) - \min_i(\bar{W}_i)}, \tag{34}$$

where i indexes the n grid points of the reference solution inside the domain of comparison, W_i and $\bar{W}_i$ are the interpolated and reference solutions at cell i, respectively. The denominator is a good practical estimate for the amplitude of the wave/discontinuity in the variable W.

Table 2
CPU timings and relative errors for the grid aligned wave reflection test

Scheme	Δt	CPU (s)	$t = 10$, $10 < x < 50$			$t = 35$, all x	
			$\delta(\rho)$ (%)	$\delta(v_x)$ (%)	$\delta(p)$ (%)	$\delta(v_x)$ (%)	$\delta(p)$ (%)
Explicit	0.003	445	6.1	6.0	12.0	12.9	1.1
Expl./Impl./Roe1	1/4	69	6.1	6.0	12.1	9.9	1.1
Impl./Roe1	1/10	228	6.2	6.0	12.0	11.6	1.1
Impl./LF1	1/4	95	6.6	6.5	12.8	8.8	1.3
Impl./Roe1/$\tau = 10^{-1}$	1/4	56	6.7	6.8	13.4	25.1	1.0
Impl./Roe1/$\tau = 10^{-2}$	1/4	82	6.4	6.4	12.7	10.3	1.1
Impl./Roe1	1/4	133	6.4	6.4	12.7	8.6	1.1
Impl./Roe1/$\tau = 10^{-7}$	1/4	413	6.4	6.4	12.7	8.5	1.1
Impl./Roe2	1/4	278	6.1	6.0	12.0	9.9	1.1
Impl./Newton	1/4	681	6.4	6.4	12.7	8.6	1.1
Impl./Roe1	1/2	77	7.3	7.7	15.8	8.2	1.2
Impl./Roe1	1	52	8.9	11.0	21.4	37.9	1.2
Impl./Roe1	2	39	10.9	14.1	28.7	35.8	1.2
Impl./Newton	2	288	10.8	13.5	28.2	35.1	1.3

The explicit, explicit/implicit, and fully implicit schemes are compared with various settings and various time steps. The '/Roe1' description in the first column of the table indicates that the second order implicit Roe solver is combined with the first-order Roe scheme in the Jacobian-free evaluation of the matrix vector products, while the '/LF1' means that the first-order local Lax–Friedrichs solver is used in the Jacobian-free evaluation.

The table contains results from several runs with 'non-standard' settings. The $\tau = 10^{-1} \cdots 10^{-7}$ rows show the effects of varying the tolerance for the Krylov solver. The default tolerance $\tau = 10^{-3}$ gives essentially the same errors as the $\tau = 10^{-7}$ tolerance but the CPU cost is a factor of 3 less, because the average number of Krylov iterations is 23 instead of 76. If the tolerance is relaxed to $\tau = 10^{-2}$ or 10^{-1} the errors become larger, especially in the v_x variable at the final time. For this particular test the $\tau = 10^{-2}$ tolerance still gives acceptable results, and the CPU time is 60% of the time used by the standard run with $\tau = 10^{-3}$.

The 'Impl./Roe2' row shows the results obtained when the second-order Roe scheme is used in the Jacobian-free evaluation. The errors are only slightly (about 6%) smaller at $t = 10$ compared to our standard 'Impl./Roe1' scheme with the same $\Delta t = 1/4$ and at the final time the errors are actually larger. On the other hand the CPU time is more than doubled. The main reason is that the number of Krylov iterations is doubled due to the less effective preconditioning. The preconditioner is based on the first-order local Lax–Friedrichs scheme, which is more optimal for first-order Roe scheme than the second-order Roe scheme.

The 'Impl./Newton' rows of the table correspond to two runs employing the non-linear Newton solver. The first-order Roe scheme is used in the Jacobian-free evaluation (with the second-order Roe scheme the iteration does not converge), but that does not affect the solution of the non-linear equation (11). The tolerance for the Newton iteration is set to 10^{-7}. For the $\Delta t = 1/4$ time step the errors for the linearized 'Impl./Roe1' and non-linear 'Impl./Newton' schemes happen to be identical, but the Newton scheme is more than 5 times more expensive. For $\Delta t = 2$ the non-linear solver is slightly (1% to 4%) more accurate, but the CPU cost is more than 7 times more than for our standard run with a single Newton iteration and $\tau = 10^{-3}$ tolerance.

These results demonstrate that solving the linearized equation (14) with $\tau = 10^{-3}$ tolerance and using a first order Jacobian-free evaluation can improve the efficiency dramatically without any significant loss of accuracy. These results are specific to the actual test problem, but the overall trends agree well with our experience in solving magnetospheric problems.

3.2.3. Rotated test problem

To make the problem truly multidimensional, the initial state is rotated around the Z axis by $\alpha = \tan^{-1} 0.5 \approx 26.57°$. This choice for the angle allows us to apply *sheared* boundary conditions which preserve the planar symmetry of the problem. For example the ghost cells indexed by $(i, 5, k)$ at the $Y = +2$ boundary of the $4 \times 4 \times 4$ block are filled in from the cells indexed by $(i + 1, 3, k)$ for $i, k = 1, \ldots, 4$ (the ghost cells between the blocks are already set when the outer boundary ghost cells are updated). This way the rotated problem can be done in the elongated computational domain without any artifacts entering from the boundary. The selected rotation angle (unlike $\alpha = 45°$) is not special with respect to the divergence **B** control schemes.

The rotation involves the X and Y components of the vector variables, and the initial discontinuities are also rotated to become parallel with the rotated direction of planar symmetry. The orthogonal distance between the magnetic wall and the sound waves are preserved as much as the finite grid resolution allows it. The distance along the X axis is increased from 25 to 28 cells, which is approximately $25/\cos\alpha = 27.95$. The rotated problem is physically identical with the original grid aligned one.

If the magnetic field components are rotated in a simple manner, the usual discretizations of $\nabla \cdot \mathbf{B}$ will not be zero for the initial field. Since the 8-wave, diffusion control and projection schemes are all using the centered difference discretization for $\nabla \cdot \mathbf{B}$, we define the initial magnetic field to be divergence free in this discretization. To achieve this the Z component of the vector potential is calculated in the rotated frame as

$$A_z = 0.1(y\cos\alpha - x\sin\alpha) - 100\min(0, x\cos\alpha + y\sin\alpha), \tag{35}$$

where the 0.1 and 100 coefficients correspond to the non-rotated values of B_x and B_y, and the minimum function restricts B_y to the left half of the rotated domain. The magnetic field components are now obtained by a centered difference approximation of $\mathbf{B} = \nabla \times \mathbf{A}$. The thermal pressure is set to $p = 5000.63 - (B_x^2 + B_y^2)/2$ so as to maintain the pressure equilibrium, and the density is set to $\rho = \gamma p$ to keep the sound speed unity everywhere.

Since the initial discontinuity is not aligned with the grid, the numerical scheme has a hard time to relax to the numerically stable discontinuity. Even the explicit time integration scheme has to reduce Δt during the first 276 time steps. The smallest Δt is around 10^{-6} instead of the CFL limited 3×10^{-3}. Still, for the whole run, the average Δt is 0.002894, and the achieved average CFL number is 0.869 which is quite close to the prescribed 0.9 value.

The rotated solution (shown in Fig. 12) obtained with the explicit scheme is less accurate than in the grid aligned case (see Fig. 9). The discontinuity at $x \approx 0$ is captured by about seven cells along the X axis (instead of the four cells in the grid aligned solution). The expected widening due to the angle between the normal direction and the X direction is only 12% ($1/\cos\alpha \approx 1.12$). One reason for this smearing is that the Roe solver is designed for a 1D grid aligned Riemann problem, not the rotated problem. The other reason is that the grid is much less suitable to represent a rotated discontinuity than a grid aligned jump. Even more disturbing are

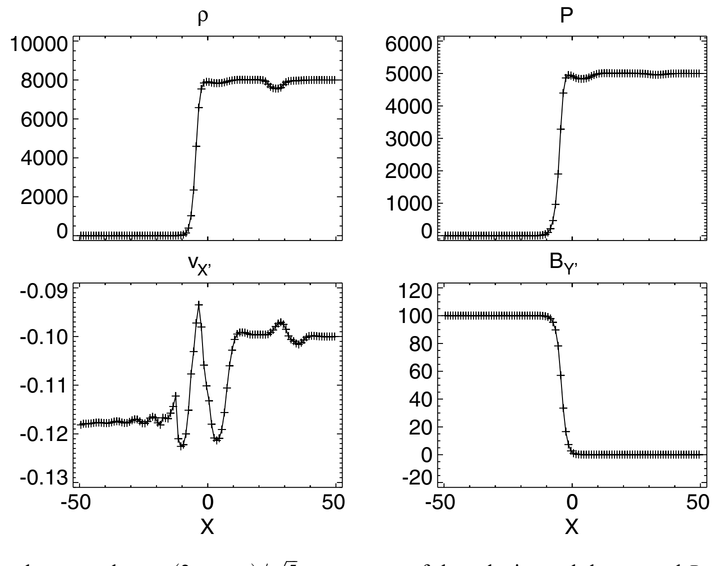

Fig. 12. The density, pressure, the rotated $v_{x'} = (2v_x + v_y)/\sqrt{5}$ component of the velocity and the rotated $B_{y'} = (2B_y - B_x)/\sqrt{5}$ component of the magnetic field at the final time $t = 35$. This solution was obtained with the explicit scheme on a grid rotated by $\alpha = \tan^{-1} 0.5 = 26.6°$. The grid aligned solution is shown in Fig. 9.

the errors in the velocity component orthogonal to the magnetic wall. The velocity is -0.118 instead of the correct -0.1 value at the left boundary. The error is comparable with the velocity amplitude of the reflected sound wave. The error in the tangential Y' velocity component is also large across the boundary of the magnetic wall. The explicit scheme solves the rotated problem in 528 s.

The implicit scheme also has to reduce the time step to get through the initial transients. The time-step adjustment algorithm (see Section 2.3.4) reduces Δt by a factor of 256 in the first eight attempts of executing the implicit time step. After this the time step is gradually increased to the originally set value $\Delta t = 1/4$ and stays there. Since this simulation is relatively short, the average time step is reduced to 0.145. Still the implicit scheme completes the run in 149 CPU seconds, which is about 3.5 faster than the explicit scheme. The initial relaxation can be sped up if the simulation is started with the explicit time stepping and the implicit scheme is switched on at $t = 0.1$ only. This ad hoc trick avoids the initial and expensive attempts to solve the first time steps with large Δt, and reduces the overall execution time to 101 s. The solution obtained by the implicit scheme is very similar to the explicit solution shown in Fig. 9. In fact the error in the velocity is somewhat smaller.

In the explicit/implicit run, the time-step adjustment algorithm also drastically reduces the time step at the beginning of the run, but when Δt becomes less than 0.003, all the blocks become explicit, and the code runs fully explicitly until the time step is gradually increased back above 0.003 and then all the way up to the original $\Delta t = 1/4$ value. The overall run time is 94 s, which is 1.6 times faster than the implicit scheme and 5.6 times faster than the explicit scheme. If we start the run with the explicit scheme and switch to the explicit/implicit scheme at $t = 0.1$, then the overall execution time reduces to 65 s, which is 8 times faster than the explicit method. The explicit/implicit solution is essentially the same as the fully implicit solution.

In the above mentioned runs the numerical error in $\nabla \cdot \mathbf{B}$ was controlled by the 8-wave scheme. If no $\nabla \cdot \mathbf{B}$ control scheme is applied the explicit and the explicit/implicit methods crash after 400 time steps, while the fully implicit method fails after 70 time steps even with the time-step adjustment algorithm. If the diffusive control scheme is used by itself, the explicit time stepping works, but the time step has to be reduced occasionally during the run, the explicit/implicit scheme fails after 270 steps at $t = 2.8$, while the fully implicit method runs with a time step reduced below 0.01 without ever recovering. These experiments suggest that the split diffusive control by itself fails for the implicit and explicit/implicit schemes, and it only works marginally for the explicit scheme.

The combined application of the 8-wave and the diffusive control schemes works well. Fig. 13 compares the magnitude of $\nabla \cdot \mathbf{B}$ for four different runs. The $\nabla \cdot \mathbf{B}$ error is about the same for the explicit and explicit/implicit 8-wave schemes. The error is advected to the left with the flow, and it is slowly diminishing with time. On the other hand, the combined diffusive control/8-wave scheme reduces the error in $\nabla \cdot \mathbf{B}$ by a factor of about 3–4 in the implicit run and by more than a factor of 300 in the explicit run. This difference is not unexpected, since the diffusion is applied about 80 times more often in the explicit run then in the implicit run, so the $\nabla \cdot \mathbf{B}$ error diffuses away more efficiently in the explicit run. Note that the $\nabla \cdot \mathbf{B}$ error is due to truncation error in all the divergence control schemes.

To further reduce the $\nabla \cdot \mathbf{B}$ error in the explicit/implicit run, an approximate projection scheme can be applied after every time step. The Poisson problem is approximately solved by 10 iterations of the conjugate gradient method. To make the implicit solver work well, the 8-wave source terms must also be applied. The source term in the overall scheme is essentially zero, because $\nabla \cdot \mathbf{B}$ is projected to approximately zero, but in the Jacobian-free evaluation of the matrix vector products, the source terms are essential in keeping the matrix well conditioned. The approximate projection scheme reduces the error in $|\nabla \cdot \mathbf{B}|$ by about a factor of 80 relative to the 8-wave scheme. Further reduction is possible by increasing the number of iterations in the conjugate gradient solver.

The effect of the $\nabla \cdot \mathbf{B}$ error on the overall solution can only be judged by comparing the flow variables directly. Fig. 14 compares the parallel components of the velocity and magnetic field and the transverse component of the magnetic field. The reduced error in $\nabla \cdot \mathbf{B}$ corresponds to a more accurate solution for the $B_{x'}$ component, which should be uniformly constant with a value 0.1 in the whole computational domain. The error is largest for the 8-wave scheme, it is smaller but more spread out for the 8-wave plus diffusion scheme and it is smallest and best localized by the 8-wave plus projection scheme. The differences between the solutions are very small in the rotated $B_{y'}$ component, although the diffusion and projection schemes produce a

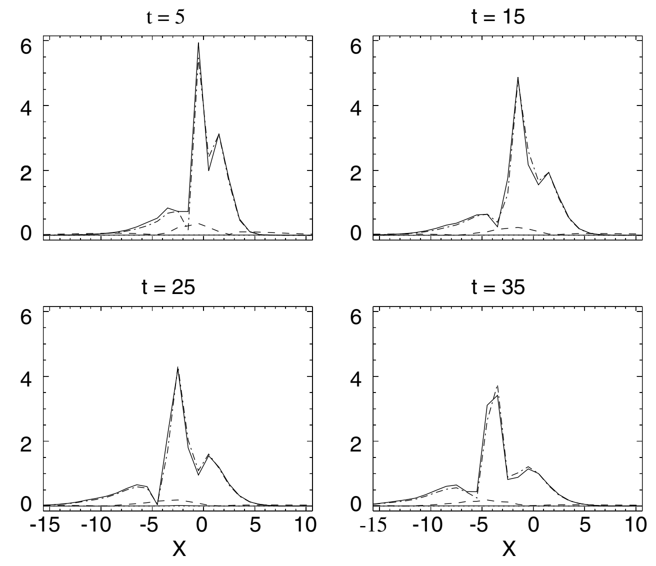

Fig. 13. Time series of the numerical value of $|\nabla \cdot \mathbf{B}|$ for the explicit 8-wave scheme (line), explicit/implicit 8-wave scheme (dot-dashed line), explicit 8-wave scheme with diffusion (dotted line), explicit/implicit 8-wave scheme with diffusion (dashed line), and explicit/implicit 8-wave scheme with projection (triple-dot-dashed line). Note that the dotted and triple-dot-dashed lines almost completely overlap. For comparison, the maximum of the curl of the magnetic field is around 26.

slightly sharper discontinuity. On the other hand the error in $v_{x'}$ is quite random and it remains significant independent of the error in $\nabla \cdot \mathbf{B}$.

The CPU time spent on the diffusive control is about 4.5% of the total execution time in the explicit run, while the same fraction is only about 1% in the explicit/implicit run. When the projection scheme is applied with 10 iterations in the explicit/implicit run, the cost increases to 5.5%. The overall execution times vary between 65 s (8-wave scheme) and 77 s (projection scheme), but most of difference is related to the number of time steps taken due to the time step reductions, rather than the time spent on the $\nabla \cdot \mathbf{B}$ control.

3.2.4. Rotated tests with adaptive grids

The accuracy of the solution can be significantly improved only by increasing the resolution. This can be done efficiently with an adaptive grid, since the sharp features that require higher resolution occupy a small fraction of the simulation domain. We increase the effective resolution from $\Delta x = 1$ to $\Delta x = 1/4$ by applying two levels of refinement. Initially the fine blocks are placed around the initial discontinuities. During the simulation the refinement and coarsening criteria are set such that the finest resolution covers the leading and trailing edges of the sound waves and the slowly advecting tangential discontinuity within a distance of unity. The rest of the grid is coarsened back to the base resolution of $\Delta x = 1$. The resolution of adjacent blocks can differ by at most a factor of 2 only.

The average number of the $4 \times 4 \times 4$ blocks is approximately 300, which is a factor of 12 increase relative to the 25 blocks needed by the uniform grid. This is not at all optimal, since a cell based AMR grid could resolve the discontinuity and the two sound waves with about 2–3 times fewer cells. On the other hand, the number of blocks in the refined grid is a factor of 5.3 less than the 1600 blocks required by a uniform $\Delta x = 1/4$ grid. About two thirds of the blocks are at the finest level, therefore using time steps proportional to the cell size could improve the speed by at most 25% relative to our explicit scheme that uses the same time step in every cell. The test problem is still rotated so it is fully multidimensional.

The explicit scheme has to use reduced time steps at the beginning, but overall it runs close to the prescribed 0.9 CFL number. The average time step is $\Delta t \approx 0.00095$, which is slightly larger than expected from a simple estimate of $0.9 \times 0.25/300$. The reason is that as the tangential discontinuity is spreading out, the fast speed is slightly decreasing in the region covered by the $\Delta x = 1/4$ cells. The adaptive mesh refinement and coarsening is done at every 100 steps, so the sound waves can propagate only a third of the finest cell size $\Delta x = 1/4$ between

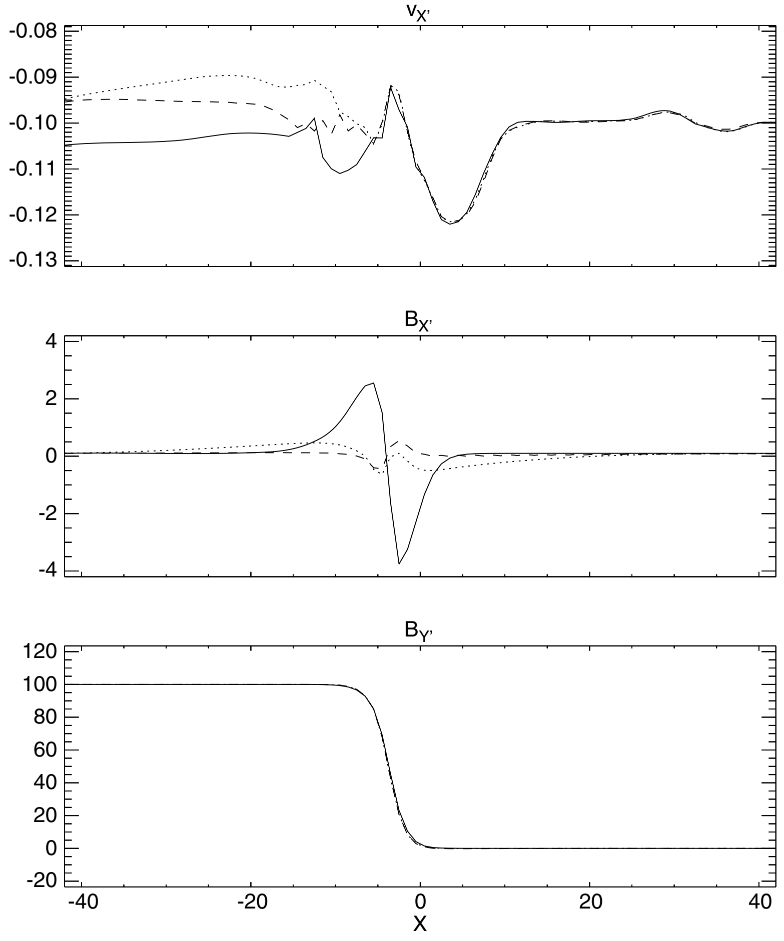

Fig. 14. The rotated $v_{x'} = (2v_x + v_y)/\sqrt{5}$ component of the velocity and the rotated $B_{x'} = (2B_x + B_y)/\sqrt{5}$ and $B_{y'} = (2B_y - B_x)/\sqrt{5}$ components of the magnetic field at the final time $t = 35$ for the explicit/implicit scheme with various $\nabla \cdot \mathbf{B}$ control schemes: 8-wave (line), 8-wave plus diffusion (dotted), and 8-wave plus projection (dashed). Note that the dotted and dashed lines overlap in the bottom panel.

two mesh refinements. The explicit run is completed in 20,868 CPU seconds, from which only 56 s are spent on the AMR.

To reduce the startup time, both the fully implicit and the explicit/implicit runs are started with the explicit method, and the implicit scheme is switched on at $t = 0.1$, after which the code does not need any time step reduction. The grid is adapted every five time steps so that the sound waves cannot travel outside the refined region. The fully implicit scheme with $\Delta t = 1/16$ (which is one quarter of the time step used for the uniform $\Delta x = 1$ resolution) completes the run in 3540 s, so it is about 5.9 times faster than the explicit code. If the time step is increased to $\Delta t = 1/10$, the run is finished in 2785 s (7.5 times faster than explicit), but the sound wave profiles get somewhat distorted as shown in Fig. 15.

The optimal time step for the explicit/implicit scheme is $\Delta t = 1/16$, because this way all the blocks on the right half of the domain can be integrated explicitly. Only about one fifth of the blocks require the implicit time stepping. The explicit/implicit run finishes in 1224 s, which is about 3 times faster than the fully implicit run with the same time step, 2.3 times faster than the fully implicit run with $\Delta t = 0.1$ and 17 times faster than the explicit method. The explicit/implicit scheme is also more accurate in handling the propagating sound waves than the implicit scheme with the same or larger time steps (see Fig. 15). Table 3 shows the CPU timings and the relative errors quantitatively.

While the AMR solutions for the sound waves are more accurate than the coarse uniform grid solution (as expected), the contact discontinuity is not resolved any better because the refinement criterion does not

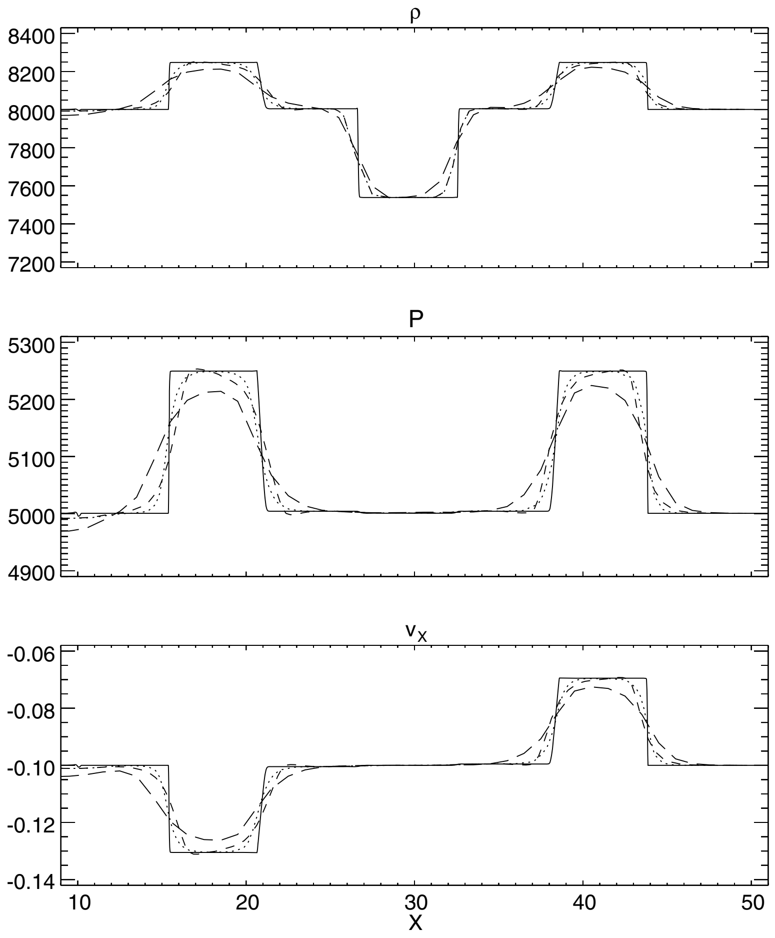

Fig. 15. The density, pressure and the (rotated) X component of the velocity at time $t = 10$. The X coordinate of the grid aligned reference solution (solid line) is stretched by $\sqrt{5}/2$ to match the rotated simulations. The explicit/implicit solution with $\Delta t = 1/4$ on a uniform grid with $\Delta x = 1$ (long dashed), the explicit/implicit solution with $\Delta t = 1/16$ on an AMR grid with $\Delta x_{\min} = 1/4$ (dotted), and a fully implicit solution with $\Delta t = 1/10$ on the same AMR grid (short dashed) are shown.

increase the resolution across the contact discontinuities. This explains why the error for the density is reduced by about a factor of 2.2, while the errors for velocity and pressure decrease by a factor of 3 as the resolution is increased by a factor of 4 by the grid adaptation. The improvement is smaller than expected from a first-order convergence rate. This is not caused by the AMR, because the same level of error was found with a uniformly finer grid. Rather, this suggests that the higher order error terms are not yet negligible at these resolutions.

Although the final solution shown in Fig. 17 is much more accurate with the $\Delta x_{\min} = 1/4$ AMR grid than with the uniform grid, the velocity during the interaction of the sound wave and the magnetic wall is still inaccurate. Fig. 16 shows the velocity at four different times for the reference solution, and the explicit/implicit scheme on the uniform and AMR grids. The reflected sound wave is much better captured by the AMR grid with $\Delta x_{\min} = 1/4$, but the velocity in the magnetized region shows moderate improvement only. We checked that the solutions obtained by the explicit scheme on the same grids are not better either. The last column of Table 3 shows this quantitatively for the final time. To get these details right, an even finer spatial and temporal resolution is required.

The last three rows of Table 3 show AMR runs with an extra level of refinement: the smallest cell size is $\Delta x = 1/8$ and there are 1750 blocks on average. With this grid the fully explicit scheme keeps reducing the time step and finally fails with negative pressure after about 800 time steps. Two changes are needed to make the

Table 3
CPU timings and relative errors for the rotated wave reflection test

Scheme	$\Delta x_{\min}$	Δt	CPU (s)	$t = 10$, $10 < x < 50$			$t = 35$, whole grid		
				$\delta(\rho)$ (%)	$\delta(v_x)$ (%)	$\delta(p)$ (%)	$\delta(B_y)$ (%)	$\delta(v_x)$ (%)	$\delta(p)$ (%)
Explicit	1	0.003	528	6.6	6.0	11.6	1.5	27.4	1.7
Ex./Im./Roe1	1	1/4	65	6.6	6.1	11.9	1.4	10.3	1.7
Impl./Roe1	1	1/4	101	6.7	6.4	12.1	1.4	11.3	1.7
Explicit	1/4	0.001	20,868	3.1	2.0	3.9	0.8	8.3	0.7
Ex./Im./Roe1	1/4	1/16	1224	2.9	2.1	4.1	0.8	8.4	0.7
Impl./Roe1	1/4	1/16	3540	3.2	2.5	4.9	0.8	10.2	0.7
Impl./Roe1	1/4	1/10	2785	3.6	3.1	6.0	0.8	9.6	0.7
Expl./$\beta = 1$	1/8	0.0003	350,668	2.2	1.6	3.1	0.9	14.1	0.7
Ex./Im./LF1	1/8	1/32	21,755	1.8	1.1	2.1	0.4	5.9	0.4
Impl./LF1	1/8	1/32	32,786	2.1	1.5	3.0	0.4	7.7	0.4

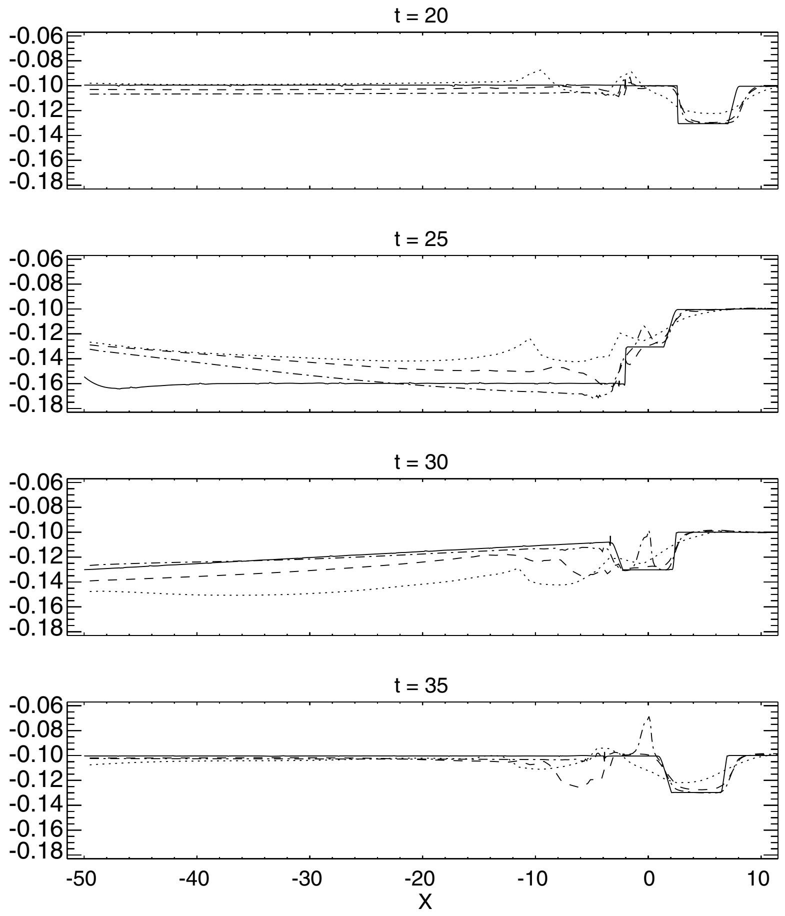

Fig. 16. The (rotated) X component of the velocity before (top), during (middle) and after (bottom) the reflection of the sound wave in the reference solution. The X coordinate of the reference solution (solid line) is stretched to match the rotated solutions obtained with the explicit/implicit scheme on the uniform grid with $\Delta x = 1$, $\Delta t = 1/4$ (dotted line) and on the AMR grids with $\Delta x_{\min} = 1/4$, $\Delta t = 1/16$ (dashed line) and $\Delta\ x_{\min} = 1/8$, $\Delta t = 1/32$ (dot-dash line).

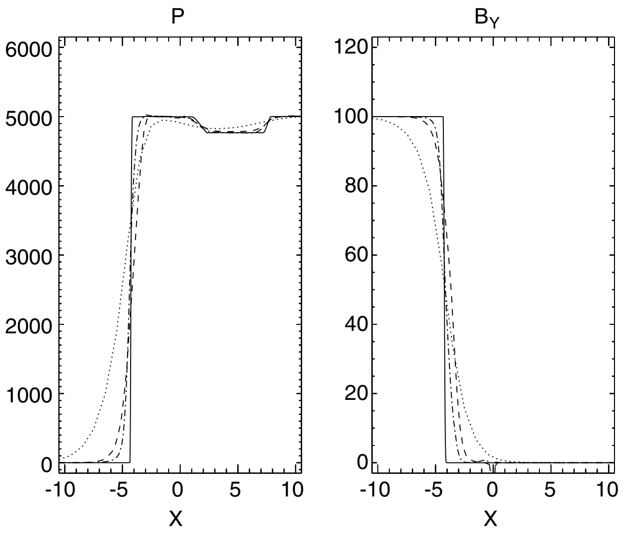

Fig. 17. The pressure and the (rotated) Y component of the magnetic field at the final time $t = 35$. The X coordinate of the reference solution (solid line) is stretched to match the rotated solutions obtained with the explicit/implicit scheme on the uniform grid with $\Delta x = 1$, $\Delta t = 1/4$ (dotted line), and on the AMR grids with $\Delta x_{\min} = 1/4$, $\Delta t = 1/16$ (dashed line) and $\Delta x_{\min} = 1/8$, $\Delta t = 1/32$ (dot-dashed line).

explicit scheme work: (1) the non-conservative pressure equation is solved instead of the conservative energy equation, and (2) in the β-limiter $\beta = 1$ had to be used (which is the same as the minmod limiter). With these changes the time step remains stable and the average $\Delta t = 4.3 \times 10^{-4}$. The explicit simulation took 6.1 hours to complete on 16 processors of a Macintosh G5 cluster (so the timings are not comparable with other runs done on a single AMD processor).

For the explicit/implicit and fully implicit runs the time step is set to $\Delta t = 1/32$, which makes about one fifth of the blocks implicit for the explicit/implicit time integration. The numerical flux used in the Jacobian-free evaluation had to be changed from the first order Roe scheme to the first-order local Lax–Friedrichs scheme to avoid stability problems similar to those found with the explicit scheme. However, the conservative energy equation and the β-limiter could be retained unlike for the explicit scheme. The fully implicit run finished about 10 times faster than the explicit run, while the explicit/implicit run was more than 16 times faster than the explicit one.

The numerical errors for the explicit scheme are larger than expected from first-order convergence, because the minmod limiter is more diffusive than the β-limiter used in the lower resolution runs. On the other hand, both the explicit/implicit and the fully implicit schemes have errors reduced by about a factor of 2 relative to the $\Delta x = 1/4$ resolution runs, which corresponds to a first-order convergence rate. The only exception is the error of the velocity at the final time, which is reduced by about a factor of 1.4 only. Nevertheless, Fig. 16 shows significant improvement for the velocity. We note that the reference solution was obtained on a 1D grid consisting of 10,000 grid cells, while the finest rotated AMR solution is on a 3D grid with an effective resolution of 800 cells in the X direction.

Finally, we provide some more detailed timing information for the explicit/implicit run with $\Delta x = 1/4$ resolution, so that the cost of the various parts of the algorithm can be compared:

- The explicit/implicit scheme: 100%
- Filling up ghost cells: 5%
- Advancing the explicit blocks: 14%
- Advancing the implicit blocks: 78%
  - Creating the preconditioner matrices: 17%
  - Applying the preconditioner matrices: 26%
  - Jacobian-free matrix vector products: 31%

All per cents are taken with respect to the total execution time. The last three items are the dominant tasks performed while advancing the implicit blocks.

It is also interesting to compare the average number of Krylov iterations for the various grid sizes. For the explicit/implicit runs the average number of iterations decreases from 21 through 11 down to 8 as the number of implicit blocks increases from 12 through 60 up to 342. The improved efficiency of the linear solver is due to the time step being decreased from 1/4 to 1/32. In the fully implicit runs the average number of iterations decreases from 23, through 10 to 7 as the number of blocks vary from 25 through 300 to 1750.

3.3. Magnetospheric applications

We present here the performance of the explicit/implicit method in typical magnetospheric simulations done with the BATSRUS code. The 3D ideal MHD equations are solved with time varying boundary conditions, which are based on satellite measurements of the solar wind conditions. The first problem uses a medium fine grid resolution, which is suitable for obtaining results relatively fast, possibly faster than the physical time (which is important if we want to use the code to make predictions). The second problem employs a higher resolution grid, which provides more accurate results. This type of simulation is used to study geophysical events and compare the simulation with observations and theoretical predictions. The third part shows a study of the same problem with various grid sizes to better understand the effects of numerical errors. All these tests are actual geophysical applications and they provide information on the performance of the explicit/implicit time stepping schemes on massively parallel super computers.

3.3.1. Scaling of a medium resolution problem

The global magnetosphere is modeled in a domain $\pm 64R_E$ in the Z and Y directions, while X ranges from $-224R_E$ in the magnetotail to 32 R_E upstream where the solar wind enters. The distance is measured in units of R_E that is the radius of the Earth. The inner boundary is at the surface of a sphere of radius $2.5R_E$ centered around the origin. The grid spacing in the inner magnetosphere is $0.25R_E$, while the grid spacing near the outer boundaries is $4R_E$, as is shown in Fig. 18. The code starts the run with 2557 blocks which are 8 cells cubed, for 512 equally sized cells per block. There is one grid adaptation during this short run, which reduces the number of blocks to 2494, so on average the total number of cells in the simulation is about 1.3 million. The spatial discretization is based on the robust second order TVD Lax–Friedrichs scheme using the minmod limiter. The $\nabla \cdot \mathbf{B}$ error is controlled by the 8-wave scheme.

Due to the large Alfvén speed near the poles of the Earth, the explicit time step is limited to $\Delta t = 0.0156$ s. On the other hand the dynamics of the system changes at a much slower scale, and the implicit and explicit/implicit runs can safely use $\Delta t = 4$ s time steps for typical solar wind conditions. As shown in Fig. 18, a large part of the simulation domain is stable for a 4 second time step, so it can be advanced explicitly by the explicit/

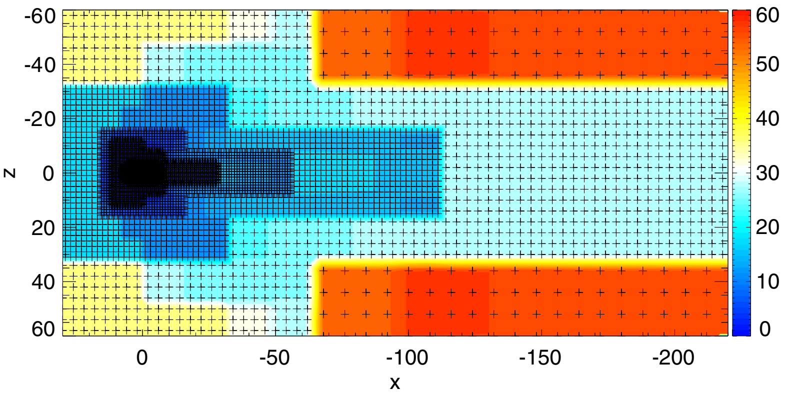

Fig. 18. The $Y = 0$ cut of the 3D magnetosphere simulation with medium grid resolution. The colors show the CFL limited time step in seconds.

implicit scheme. During the 5-min simulation time, which is used for the timing runs, a shock reaches the magnetosphere, which results in a few time step reductions. The average time step still remains 3.41 s.

Table 4 shows the wall clock times, the speed up relative to the explicit scheme and the parallel scaling factors relative to the 64 CPUs on an SGI 3800 machine. The scaling factor is the ratio of the actual wall clock time to the wall clock time which would be obtained for an ideal speed up. Although the explicit scheme scales very well all the way up to 256 CPUs, it runs 7–5 times slower than the implicit scheme and 28–15 times slower than the explicit/implicit algorithm.

For this problem size the implicit and explicit/implicit schemes scale reasonably up to 128 CPUs, but for more CPUs the scaling becomes worse, especially for the explicit/implicit scheme. This is due to several reasons. The number of implicit blocks is only about 400, which means that the calculation/communication ratio during the implicit time stepping is not too large. On more than 128 processors the load balancing of the roughly 400 implicit blocks also becomes uneven: some processors will have one, others two implicit blocks. Finally, the improved speed of the code means that the speed up is more and more subject to Amdahl's law as the serial parts of the code become more and more dominant.

We finally note that the explicit/implicit run is faster than real time even on 64 processors, since the 300 second physical time is simulated in 268 CPU seconds. Neither the explicit nor the fully implicit schemes can reach faster than real time performance even on 256 CPUs. Fig. 19 compares the solutions obtained by the explicit, implicit ($\Delta t = 2$ s) and explicit/implicit ($\Delta t = 2$ s) schemes after 36 min of simulation time. The solutions are practically identical.

3.3.2. Scaling of a high resolution problem

The computational domain is a box with $-224R_E < x < 32R_E$ and $-128R_E < y, z < 128R_E$. The inner boundary is at the surface of a sphere of radius $2.5R_E$ centered around the origin. The block adaptive grid consists of 4572 blocks with $8 \times 8 \times 8$ cells each, altogether 2.3 million cells. The cell size varies from $1/8R_E$ to $8R_E$. The smallest cells are clustered around the inner boundary, which requires high resolution, and we also cover the near-Earth parts of the bow shock, magnetopause and magnetotail with relatively fine $1/4R_E$ resolution.

The spatial discretization is based on the robust second order TVD Lax–Friedrichs scheme using the β-limiter with the $\beta = 1.2$ setting. The $\nabla \cdot \mathbf{B}$ error is controlled by the 8-wave scheme. For the explicit/implicit time integration, the time step is set to $\Delta t = 2.5$ s, which was found to be optimal for this problem. During the short timing runs, this time step results in 3856 implicit blocks out of the 4572 blocks.

Fig. 20 shows the scaling of the explicit/implicit time integration scheme for a fixed problem size on an SGI Altix super computer. The code scales almost perfectly from 32 to 256 CPUs, and the speed keeps improving up to 508 CPUs that was the maximum number available on this machine.

3.3.3. Grid size dependence

This section provides information about the problem-size dependence of the efficiency of the explicit/implicit time integration scheme. The physical problem is a magnetosphere simulation for a certain time period (October 22–24, 2003) and the simulation results were compared with satellite observations. The computational domain is the same as in the problem discussed in the previous subsection. We repeat the simulation with various grid sizes to better understand the effects of numerical errors. The overall conclusion is that the higher resolution runs agree better with the satellite observations than the lower resolution runs, which

Table 4
Wall clock times, speed relative to the explicit scheme, and parallel scaling efficiency for the medium resolution magnetospheric simulation

CPUs	Wall clock time (s)			Relative speed		Scaling factor		
	Expl.	Impl.	Ex./Im.	Impl.	Ex./Im.	Expl.	Impl.	Ex./Im.
64	7721	1057	268	7.3	28.8	1.00	1.00	1.00
96	5218	776	198	6.7	26.4	1.01	1.10	1.11
128	3953	632	164	6.3	24.1	1.02	1.20	1.22
192	2973	509	153	5.8	19.4	1.16	1.44	1.71
256	2263	444	144	5.1	15.7	1.17	1.68	2.15

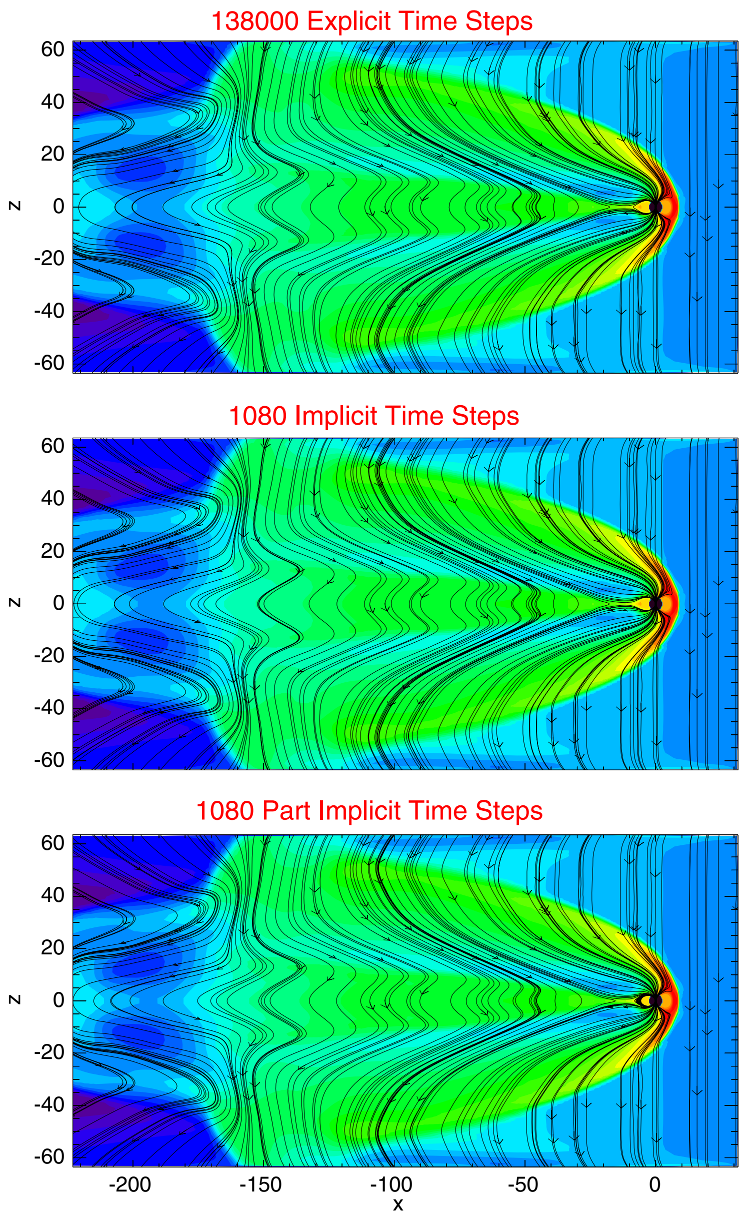

Fig. 19. The $Y = 0$ cuts of the 3D simulation domain for the explicit (top), implicit (middle) and explicit/implicit (bottom) runs. The logarithm of the pressure (colors) and the magnetic field (lines) are shown for simulation time $t = 2160$ s.

is very encouraging. The physics results of this study will be reported in a future paper. Here we are only concerned with the numerical aspects, but we stress that the choice of these grids is motivated by the physics problem, and they are fully representative of our magnetospheric applications.

Table 5 shows the number of blocks and the block size distribution for the various grids. We will refer to these grids as 'coarse', 'low', 'medium' and 'high' resolution. The number of blocks with the smallest cells is the largest for all grids, except for the coarse one. Based on the block size distribution, one can estimate how much speed up can be expected from subcycling, i.e. if the time steps are proportional to the cell size as opposed to constant. The last row shows the expected reduction in CPU time relative to the use of a uniform time step. The reduction ranges from 41% to 64% only, a relatively modest gain that is likely to be reduced by the overhead and the parallel scaling of the subcycling algorithm.

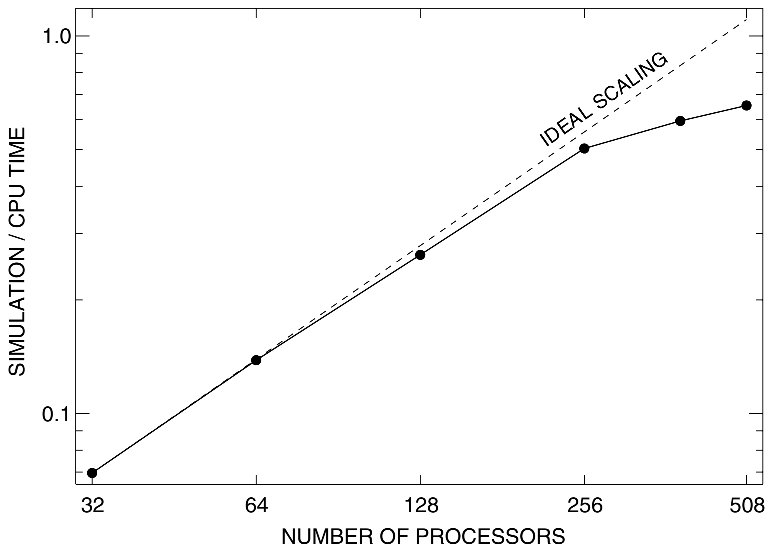

Fig. 20. Scaling of the partially implicit scheme for the high resolution magnetospheric simulation with a fixed problem size on an SGI Altix supercomputer. The execution time divided by the simulation time is shown as a function of the number of CPUs. Both axes are logarithmic.

Table 5
Number of blocks with various cells sizes on some typical grids

Grid name	Coarse	Low	Medium	High
Total blocks	512	2052	2332	5020
$\Delta x = 8$	52	32	24	32
$\Delta x = 4$	80	216	268	208
$\Delta x = 2$	100	260	336	316
$\Delta x = 1$	216	416	576	472
$\Delta x = 1/2$	64	424	424	400
$\Delta x = 1/4$	–	704	704	1096
$\Delta x = 1/8$	–	–	–	2496
Subcycling gain	41%	52%	48%	64%

Table 6
Performance of the explicit/implicit scheme on various grids

Grid name	Coarse	Low	Medium	High
No. total blocks	512	2052	2332	5020
No. implicit blocks	371	1327	1522	4062
CFL limited Δt (s)	0.145	0.028	0.028	0.014
Time step (s)	10.0	5.0	5.0	2.5
Wall clock time (s)	5.5	22.3	26.3	157.4
Speed up over explicit	2.9	11.2	11.3	8.6
Speed up over implicit	1.3	1.6	1.6	1.4
Average no. Krylov iter.	14.5	13.8	15.0	20.7
Load balance	0.9%	1.3%	1.9%	1.2%
Filling up ghost cells	11.1%	6.8%	6.4%	4.3%
Advancing explicit blocks	2.4%	2.4%	2.2%	0.8%
Advancing implicit blocks	71.6%	84.2%	84.3%	93.3%
Creating preconditioner	9.5%	22.9%	21.2%	20.1%
Applying preconditioner	6.2%	24.1%	24.6%	32.2%
Jacobian-free products	29.6%	19.1%	19.3%	21.9%

For sake of a direct comparison, we perform 12 short runs using the explicit, explicit/implicit and fully implicit time stepping algorithms on the four grids. The second-order TVD Lax–Friedrichs flux function is used with the minmod limiter. All simulations are carried out on 128 processors of an SGI Altix machine (columbia). The simulations are started from some restart files and the runs are stopped after one minute of simulation time. The time step is set proportional to the smallest cell size, and Δt is 10, 5, 5 and 2.5 s for the coarse, low, medium and high resolution grids, respectively. Table 6 compares the various aspects of the simulations. In these runs the fraction of implicit blocks varies between 64% and 81%, so the explicit/implicit algorithm gains only a factor of 1.3–1.6 over the fully implicit scheme. The gain over the explicit scheme varies from a factor of 3 to a factor of 11. We note that the CFL limit in the explicit runs is not linearly proportional to the smallest cell size, because the Alfvén speed increases as the cube of the radial distance, and the inner boundary is at $3.5R_E$ for the coarse grid, and $2.5R_E$ for the finer grids.

4. Conclusion

This paper presents and tests some implicit and explicit/implicit time integration schemes for block adaptive grids. The Newton–Krylov–Schwarz type implicit scheme is optimized for the efficient solution of time-accurate problems. The second order BDF2 implicit time discretization is linearized and solved with the GMRES algorithm. The matrix vector products are evaluated with the Jacobian-free algorithm using the spatially first-order variant of the second order scheme. The Schwarz preconditioner is based on the local data available in each block of the block-AMR grid. This choice for the preconditioner guarantees that the solution does not depend on the number of CPUs, and it has proven to be a simple but still quite efficient preconditioner. The MBILU preconditioner is based on the first-order local Lax–Friedrichs scheme. The implicit time integration scheme can achieve a factor of 3–10 or even more speed up relative to the explicit scheme.

The efficiency of the implicit method can be further improved if it is only applied for the blocks that would be unstable for the given time step. We have designed, implemented and successfully used the new block based explicit/implicit time stepping scheme. This scheme uses less memory than the fully implicit scheme, it typically runs faster by about a factor of 2–4, and it can provide more accurate solution in the explicitly handled region. Once the implicit scheme is implemented, it is relatively straightforward to extend it to the explicit/implicit algorithm. The new elements are the selection of the explicit and implicit blocks, the separate load balancing of the implicit and explicit blocks and the exclusion of a subset of the blocks from the explicit and implicit updates, respectively. It is possible but somewhat complicated to make the explicit/implicit scheme fully conservative across the interface between the explicit and implicit blocks. In our magnetospheric applications, however, applying such a flux correction step does not seem to be necessary, because there are no shock waves crossing the explicit/implicit boundary. Furthermore, the mass, momentum and energy fluxes enter and exit the boundaries of the computational domain at a rate which makes the magnetosphere an essentially driven system with no long term memory.

To improve the robustness of the implicit and explicit/implicit methods, we use a simple time-step adjusting algorithm based on the relative change in density and pressure. This algorithm has been found to greatly improve the robustness of the implicit time integration schemes.

We have also studied how the various $\nabla \cdot \mathbf{B}$ control methods can be combined with the implicit and explicit/implicit algorithms in multidimensional MHD problems. The 8-wave scheme seems to be crucial in making the iterative scheme robust in the implicit solver. The $\nabla \cdot \mathbf{B}$ error can be substantially reduced by the split application of the diffusive control scheme, or the error can be made arbitrarily small using the projection scheme after each time step. In our tests the cost of the $\nabla \cdot \mathbf{B}$ control methods remained very modest. While the diffusion and projection methods are efficient in reducing the numerical value of $\nabla \cdot \mathbf{B}$, the improvement in the accuracy of the overall solution in terms of the MHD variables has not been found to be significant in our tests and applications.

Based on our extensive experience with magnetospheric simulations, and on the numerical tests presented here, we can confidently claim that in a certain class of time accurate MHD problems, the implicit and explicit/implicit schemes can provide solutions that are as accurate as the explicit scheme, but can be obtained much more efficiently. Combined with the efficient block adaptive grid techniques, the explicit/implicit algorithm can solve problems that would be practically impossible to do with the explicit time stepping on a uniform grid.

For large enough problems, the implicit and explicit/implicit methods can scale to hundreds of CPUs of modern supercomputers.

Acknowledgments

This work has been supported by NASA AISRP Grant NNG04GP89G, NASA ESS CT cooperative agreement NCC5-614 and by DoD MURI Grant F49620-01-1-0359. G.T. has been partially supported by the Hungarian Science Foundation (OTKA, Grant Nos. T047042 and T037548). We thank David Keyes for some initial suggestions regarding the implicit scheme. We also thank the three reviewers for their numerous constructive comments and suggestions that helped improving the paper significantly.

References

[1] O. Axelsson, G. Lindskog, On the eigenvalue distribution of a class of preconditioning methods, Numer. Math. 48 (1986) 479.
[2] D.S. Balsara, Divergence-free adaptive mesh refinement for magnetohydrodynamics, J. Comput. Phys. 174 (2001) 614, doi:10.1006/jcph.2001.6917.
[3] D.S. Balsara, Second-order-accurate schemes for magnetohydrodynamics with divergence-free reconstruction, Astro. Phys. J. Suppl. 151 (2004) 149.
[4] J.U. Brackbill, D.C. Barnes, The effect of nonzero $\nabla \cdot \mathbf{B}$ on the numerical solution of the magnetohydrodynamic equations, J. Comput. Phys. 35 (1980) 426.
[5] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64.
[6] A. Dedner, F. Kemm, D. Kröner, C.-D. Munz, T. Schnitzer, M. Wesenberg, Hyperbolic divergence cleaning for the MHD equations, J. Comput. Phys. 175 (2002) 645, doi:10.1006/jcph.2001.6961.
[7] D.C. Dinge, P.R. Woodward, A parallel cell-by-cell AMR method for the PPM hydrodynamics code, Int. J. Mod. Phys. C 14 (2003) 857.
[8] C.R. Evans, J.F. Hawley, Simulation of magnetohydrodynamic flows: a constrained transport method, Astrophys. J. 332 (1988) 659.
[9] S. Gottlieb, C.-W. Shu, E. Tadmor, Strong stability-preserving high-order time discretization methods, SIAM Rev. 43 (2001) 89.
[10] R. Grauer, C. Marliani, Current-sheet formation in 3D ideal incompressible magnetohydrodynamics, Phys. Rev. Lett. 84 (2000) 4850.
[11] E. Hairer, S.P. Nørsett, G. Wanner, Solving Ordinary Differential Equations I. Nonstiff Problems Springer Series in Computational Mathematics, vol. 8, Springer, Berlin, 1987.
[12] K.C. Hansen, A.J. Ridley, G.B. Hospodarsky, M.K. Dougherty, T.I. Gombosi1, G. Toth, Global MHD simulations of saturn's magnetosphere at the time of Cassini approach, Geophys. Res. Lett. 32 (2005) L20S06, doi:10.1029/2005GL022835.
[13] D.W. Hughes, S.A.E.G. Falle, The rise of twisted magnetic flux tubes: a high Reynolds number adaptive grid calculation, Astro. Phys. J. 509 (1998) L57.
[14] R. Keppens, G. Tóth, M.A. Botchev, A. van der Ploeg, Implicit and semi-implicit schemes: algorithms, Int. J. Numer. Methods Fluids 30 (1999) 335.
[15] R. Keppens, M. Nool, G. Tóth, J.P. Goedbloed, Adaptive mesh refinement for conservative systems: multi-dimensional efficiency evaluation, Comput. Phys. Commun. 153 (2003) 317.
[16] J. Kleimann, A. Kopp, H. Fichtner, R. Grauer, K. Germaschewski, Three-dimensional MHD high-resolution computations with CWENO employing adaptive mesh refinement, Comput. Phys. Commun. 158 (2004) 47.
[17] R.I. Klein, R.T. Fisher, C.F. McKee, M.R. Krumholz, Recent advances in the collapse and fragmentation of turbulent molecular cloud cores, in: ASP Conference Series 323: Star Formation in the Interstellar Medium: In Honor of David Hollenbach, vol. 323, 2004, p. 227.
[18] D.A. Knoll, D.E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, J. Comput. Phys. 193 (2004) 357.
[19] P. Londrillo, L. Del Zanna, High-order upwind schemes for multidimensional magnetohydrodynamics, Astrophys. J. 530 (2000) 508.
[20] A. van der Ploeg, R. Keppens, G. Tóth, Block incomplete LU-preconditioners for implicit solution of advection dominated problems, in: B. Hertzberger, P. Sloot (Eds.), Proceedings of High Performance Computing and Networking Europe 1997, Lecture Notes in Computer Science, vol. 1225, Springer, Berlin, 1997, p. 421.
[21] K.G. Powell, An approximate Riemann solver for magnetohydrodynamics (that works in more than one dimension), ICASE Report No. 94-24, Langley, VA, 1994.
[22] K.G. Powell, P.L. Roe, T.J. Linde, T.I. Gombosi, D.L. De Zeeuw, A solution-adaptive upwind scheme for ideal magnetohydrodynamics, J. Comput. Phys. 154 (1999) 284, doi:10.1006/jcph.1999.6299.
[23] J.A. Rossmanith, A high-resolution constrained transport method with adaptive mesh refinement for ideal MHD, Comput. Phys. Commun. 164 (2004) 128.
[24] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput. 7 (3) (1986) 856.
[25] R. Samtaney, S.C. Jardin, P. Colella, D.F. Martin, 3D adaptive mesh refinement simulations of pellet injection in tokamaks, Comput. Phys. Commun. 164 (2004) 220.

[26] F. Shakib, Finite element analysis of the compressible Euler and Navier–Stokes equations, Ph.D. Thesis, 1988.
[27] P.K. Sweby, High resolution schemes using flux Limiters for hyperbolic conservation laws, SIAM J. Numer. Anal. 21 (1984) 995.
[28] G. Tóth, A general code for modeling MHD flows on parallel computers: versatile advection code, Astrophys. Lett. Commun. 34 (1996) 245.
[29] G. Tóth, R. Keppens, M.A. Botchev, Implicit and semi-implicit schemes in the versatile advection code: numerical tests, Astron. Astrophys. 332 (1998) 1159.
[30] G. Tóth, The $\nabla \cdot \mathbf{B} = 0$ constraint in shock-capturing magnetohydrodynamics codes, J. Comput. Phys. 161 (2000) 605, doi:10.1006/jcph.2000.6519.
[31] G. Tóth, P.L. Roe, Divergence- and curl-preserving prolongation and restriction formulas, J. Comput. Phys. 180 (2002) 736.
[32] G. Tóth, D. Kovács, K.C. Hansen, T.I. Gombosi, Three-dimensional MHD simulations of the magnetosphere of Uranus, J. Geophys. Res. (Space Physics) 109 (2004) 11210.
[33] G. Tóth, I.V. Sokolov, T.I. Gombosi, D.R. Chesney, C.R. Clauer, D.L. De Zeeuw, K.C. Hansen, K.J. Kane, W.B. Manchester, R.C. Oehmke, K.G. Powell, A.J. Ridley, I.I. Roussev, Q.F. Stout, O. Volberg, R.A. Wolf, S. Sazykin, A. Chan, B. Yu, J. Kóta, Space Weather Modeling Framework: a new tool for the space science community, J. Geophys. Res. (Space Physics) 110 (2005) A12226, doi:10.1029/2005JA011126.
[34] H.A. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 13 (1992) 631.
[35] H.C. Yee, Construction of explicit and implicit symmetric TVD schemes and their applications, J. Comput. Phys. 68 (1987) 151, doi:10.1016/0021-9991(87)90049-0.
[36] C. Zanni, A. Ferrari, S. Massaglia, G. Bodo, P. Rossi, On the MHD acceleration of astrophysical jets, Astrophys. J. Suppl. 293 (2004) 99.
[37] U. Ziegler, An ADI-based adaptive mesh Poisson solver for the MHD code NIRVANA, Comput. Phys. Commun. 157 (2004) 207.