COMPUTATIONAL METHODS IN APPLIED MATHEMATICS, Vol. 2, No. 1(2002) 92 — 109

AMRVAC: a multidimensional grid-adaptive magnetofluid dynamics code

Margreet Nool

CWI

P.O.Box 94079, 1090 GB Amsterdam, The Netherlands

E-mail: Margreet.Nool@cwi.nl

Rony Keppens

FOM-Institute for Plasma Physics Rijnhuizen

P.O.Box 1207, 3430 BE Nieuwegein, The Netherlands

E-mail: keppens@rijnh.nl

Abstract. We present results obtained with AMRVAC, a software package designed for solution-adaptive time-accurate (magneto)hydrodynamic simulations. In any dimensionality, the grid adjusts to capture shocks and other sharp flow features accurately following an automated Adaptive Mesh Refinement [AMR] strategy. This grid adaptation algorithm is incorporated with the Versatile Advection Code [VAC], so that it can be used to time-advance sets of conservation laws with options for the spatial discretization employed. We demonstrate and evaluate the efficiency achievable by AMR for 1D, 2D, and 3D test problems and describe the employed data structures.

2000 Mathematics Subject Classification: 35L65; 65M50; 76W05.

Key words: Adaptive Mesh Refinement, magnetohydrodynamics, data structures.

1. Introduction

Adaptive Mesh Refinement (AMR) [1] is a proven means to automatically refine a base grid by adding finer level grids where a higher resolution is needed. AMR also operates to coarsen the grid structure by periodically checking whether finer level grids can be removed. This approach allows to dynamically capture shocks more accurate locally, which should be computationally more efficient than a global refinement on a single static grid. In this paper we evaluate and demonstrate that efficiency when using AMR in 1D, 2D or 3D (magneto)fluid simulations. To that end, we implemented AMR in a dimension-independent fashion, and incorporated it with the Versatile Advection Code (VAC, `http://www.phys.uu.nl/~toth/`) initiated by Tóth [12]. VAC [6, 9] is a general purpose software package, usable on several parallel platforms [8], for plasma physical and astrophysical numerical simulations. It offers a choice of spatial and temporal discretization schemes for hydrodynamic (HD) and magnetohydrodynamic (MHD) equation modules, easily combined with modules for gravitational, viscous, or resistive effects. The AMRVAC code presented here inherited a similar versatility in dimensionality, physical application, and spatial discretization. As a result, AMRVAC can be applied to multi-D HD and MHD simulations, using a variety of high resolution shock-capturing numerical schemes. The dimension and the particular system of conservation laws are selected in a preprocessing stage.

We describe the AMR strategy in Sec. 2. In Sec. 3 we discuss the VAC philosophy and specific implementation issues, such as the data structures currently employed in AMRVAC. These could be of general interest to similar coding efforts which try to augment existing computational fluid dynamics packages with an AMR functionality. 1D, 2D and 3D simulations and a study of AMR efficiency are presented in Sec. 4. This paper ends with conclusions in Sec. 5.

2. Adaptive Mesh Refinement Strategy

The AMR strategy minimizes the ratio between numerical cost and accuracy of the solution. It is described in detail by Berger [1], Berger and Colella [2] and Steiner *et al.* [11].

2.1. Grid Hierarchy

In the AMR approach, regions which need higher resolution are covered with a new grid that is r times finer than the underlying old grid, where r is an (even, typically $r = 2, 4$) integer. The algorithm is recursively formulated which allows for several levels of refined grids. On the resulting sequence of nested grids, the partial differential equations are discretized. Grids of different levels must be *properly nested*: a fine grid begins and ends at the corners of a cell in the next coarser grid and there must be at least one $l-1$ cell separating a grid cell at level l from a cell at level $l-2$, as is shown in Fig. 1.

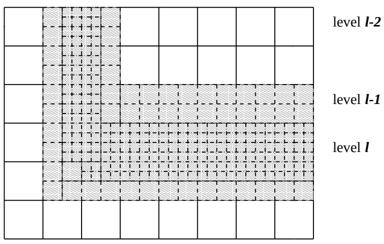

Figure 1. An example of a properly nested grid: at least one grid cell of level $l-1$ separates grids of level l and level $l-2$, except at the physical boundary.

These rules guarantee that a fine grid always overlays grids of only the next coarser level, but it is perfectly allowed that a l grid is contained in more than one $l-1$ grid, as demonstrated in Fig. 2, where grid $G(4,3)$ is contained in both $G(3,1)$ and $G(3,2)$. We restrict ourselves to Cartesian grids at all levels, while extensions to spherical and cylindrical grids are currently under consideration.

2.2. Time stepping and refinement criterion

The spatial grid refinement also results in a finer time step to provide numerical stability on all grid levels. If $\triangle x_{l,i}$ is the mesh size in the i-direction and $\triangle t_l$ the time step size, both for a level l grid, then we have

$$\frac{\Delta t_l}{\Delta x_{l,i}} \equiv \frac{\Delta t_{l-1}}{\Delta x_{l-1,i}} \equiv \ldots \equiv \frac{\Delta t_1}{\Delta x_{1,i}} \quad \text{for } i = 1, \cdots, \texttt{NDIM}, \tag{2.1}$$

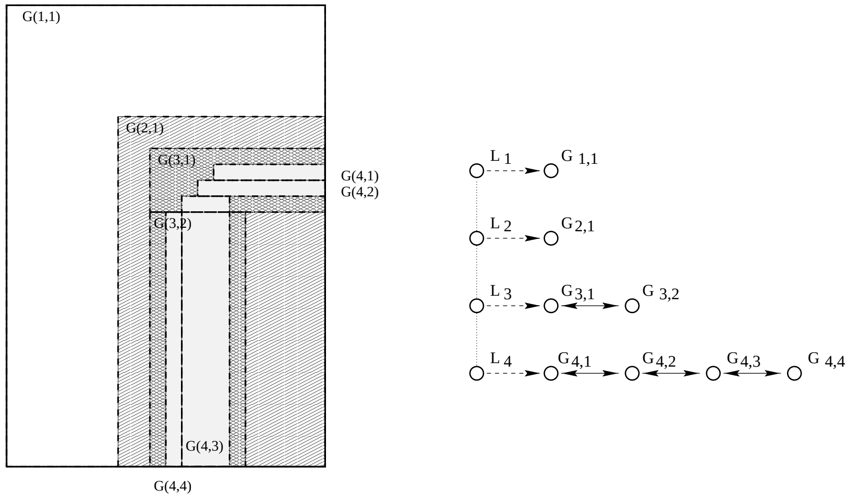

Figure 2. Example of a grid hierarchy and the corresponding linked list of level nodes. *Left*: The computational domain consists of 8 grids on 4 levels. The first index refers to the level, the second to the grid number on that level. *Right*: Structure of linked lists which keep track of the relationship between grids. Each grid node contains pointers to the next and previous sibling grids while the level node points to the first grid on the corresponding level.

where `NDIM` is the number of dimensions. The refinement ratio r_l between level $l-1$ and level l is then defined by $r_l \equiv \Delta t_{l-1}/\Delta t_l$. Due to this spatial and temporal refinement, two – related – issues for an AMR simulation are (1) how one time-advances on such a sequence of nested and thus overlapping grids, while (2) keeping the solutions consistent and conservative through a suitable 'update' process.

The approach taken is illustrated in Figure 3, showing a hypothetical sequence of three time steps in a case where $l_{\mathrm{MAX}} = 4$. The schema is traversed from left to right, bottom to top, and with horizontal 'update' arrows preceding vertical time 'advance' steps. Each time a level $l > 1$ has caught up in time with the underlying $l-1$ grids, all level $l-1$ cells that are covered by the finer grids are replaced by conservative averages. Also, those $l-1$ cells immediately adjacent to a level l grid boundary are modified to comply with the interface fluxes used on the finer level. For more complete discussions of these update and flux fixing steps, we refer to [2].

The time advancing is further complicated by the automated regridding operation, as indicated by the grey circles in Figure 3. This regrid action is explained in the following sections. It is controlled by a simple criterion: when at least k time steps are taken on a certain level, this level is evaluated for refinement ($k = 2$ in the figure). This affects all higher level grids which may either suddenly appear (in Figure 3 after the first and halfway in the second time step), disappear (after the second time step), or simply get rearranged or be left unchanged (halfway in the third time step). Of course, all level 1 grids remain unaltered, while the maximal allowed level l_{MAX} is never evaluated for further refinement.

A great variety of refinement criteria can be adopted. Powell *et al.* [10] use local measures of compressibility, rotationality, and current density in their 3D MHD simulations and a region is refined when chosen threshold values are exceeded. Ziegler [18] uses similarly motivated, though more abstract, 'gradient value based' criteria where the slopes of functionals of the fluid variables, like the sound speed c_s, control refinement. In cases where an (approximate) Riemann solver based method is exploited, one may flag cells in a 'wave-affected region'. The extent of this region can be computed from the wave strengths and speeds

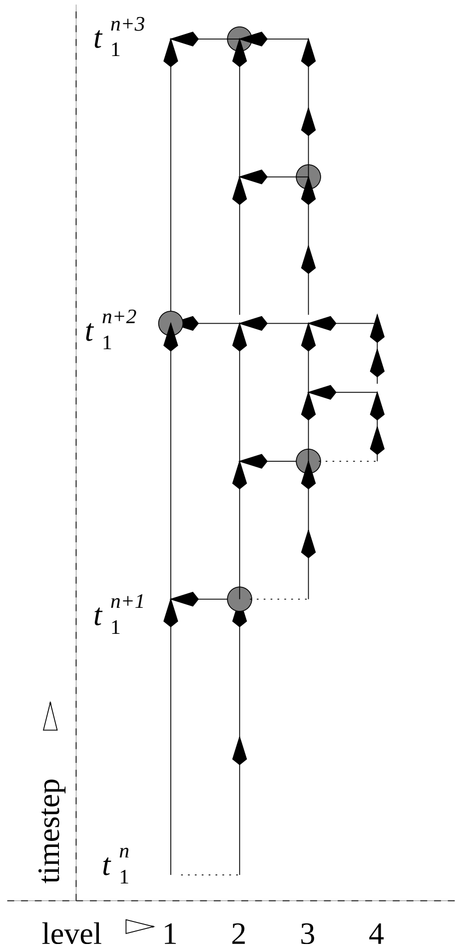

Figure 3. A hypothetical sequence of three time steps in an AMR simulation allowing for 4 nested refinement levels. Vertical arrows indicate a temporal advance, horizontal arrows correspond to update operations ensuring conservation and consistency, while the automated regridding operation is called at the position of the grey circles. Regridding is invoked when at least $k = 2$ time steps are taken.

as calculated in the Riemann solver. Such criterion was successfully exploited for unsteady Euler flows in [3].

Our refinement criterion is a variant of the error estimation procedure used in [2]. Needed are two solution vectors $\mathcal{U}_l^{n-1}$ and $\mathcal{U}_l^{n}$ on level l, separated in time by Δt_l^{n-1}. The regions which need refinement are identified as follows:

- coarsen $\mathcal{U}_l^{n-1}$ spatially by a factor 2 to obtain $\mathcal{V}_l^{n-1}$ with grid size $2\Delta x_i, i = 1, \cdots, \mathtt{NDIM}$.
- integrate $\mathcal{V}_l^{n-1}$ from time t_{n-1} to time t_{n+1} to obtain $\mathcal{V}_l^{n+1}$ using step size $2\Delta t_l^{n-1}$ with a cheap first order method.
- integrate $\mathcal{U}_l^{n}$ with step size Δt_l^{n-1} to $\mathcal{U}_l^{n+1}$ using the same scheme.
- coarsen $\mathcal{U}_l^{n+1}$ spatially by a factor 2 to obtain $\mathcal{W}_l^{n+1}$.
- flag for refinement where the relative error in a single pre-chosen component (or suitably weighted multiple components) of $\mathcal{V}$ and $\mathcal{W}$, e.g. the mass density, exceeds the

tolerance ϵ_{tol}:

$$|\frac{\mathcal{W}_l^{n+1} - \mathcal{V}_l^{n+1}}{\mathcal{W}_l^{n+1}}| > \epsilon_{\mathrm{tol}}. \tag{2.2}$$

As a result, $2^{\mathtt{NDIM}}$ level l cells are flagged.

2.3. Regridding operation

The AMR scheme then continues to arrange these $2^{\mathtt{NDIM}}$ flagged cells into properly nested *rectangular* grids. For our dimension-independent implementation, *rectangular* is to be interpreted according to dimensionality. The algorithm starts with creating grids such that the number of flagged points in each new grid divided by its total number of grid points N_{total} is larger than a preset efficiency $\mathtt{E}; 0 < \mathtt{E} \le 1$. It then enforces the proper nesting criterion by rearranging grids through bisections, projections and merging of grids. Since this may affect the grid efficiency, we introduce a minimal efficiency value $0 < \mathtt{E}_{min} < \mathtt{E} \le 1$ which is enforced. Hence, at least $\mathtt{E}_{min} \times N_{total}$ points in the newly created grids satisfy the refinement criterion. Obviously, the larger $\mathtt{E}_{min}$ and $\mathtt{E}$, the more grids will be created. The resulting new grid structure is initialized from the solutions available on levels l and up. For those cells which can not be copied directly from the original grid hierarchy, we use a conservative, *minmod* limited linear interpolation from the coarser level underneath. Due to the nesting property, this is always possible.

3. The philosophy of VAC–AMRVAC

3.1. The VAC philosophy and AMRVAC

The aim of the Versatile Advection Code, VAC (`http://www.phys.uu.nl/~toth/`), is to provide the plasma- and astrophysical community with a modern, versatile and user-friendly software, which can be adapted to the application and which runs efficiently on workstations, vector computers and parallel architectures as well. VAC uses various shock capturing numerical methods in a finite volume discretization. One of the schemes used in the numerical experiments (Sect. 4) is the Lax-Friedrich type Total Variation Diminishing (TVD) scheme called TVDLF. In this paper, we consider the second order accurate MUSCL (monotonic upwind schemes for conservation laws) TVDLF scheme

$$U_i^{n+1} = U_i^n - \frac{\Delta t}{\Delta x}(f_{i+1/2}^n - f_{i-1/2}^n) \tag{3.1}$$

where U_i^n is the discretized conservative variable defined on the mesh at the discrete time level t^n as a volume average within the i-th mesh cell centered at position x_i, and the numerical fluxes f are defined by

$$f_{i+1/2}^n = \frac{1}{2}\{F(U_{i+1/2}^L) + F(U_{i+1/2}^R) - |c^{max}(\frac{U^L + U^R}{2})|[U_{i+1/2}^R - U_{i+1/2}^L]\} \tag{3.2}$$

with left U^L and right U^R states from

$$U_{i+1/2}^L = U_i^n + \tfrac{1}{2}\overline{\Delta U_i^n} \tag{3.3}$$

and

$$U_{i+1/2}^R = U_{i+1}^n + \tfrac{1}{2}\overline{\Delta U_{i+1}^n}. \tag{3.4}$$

The *overbar* term $\overline{\Delta U_i^n}$ denotes the slope limiter, e.g. the *minmod* limiter can be used:

$$\overline{\Delta U_i^n} = sgn(U_i^n - U_{i-1}^n) max[0, min\{|U_i^n - U_{i-1}^n|, sgn(U_i^n - U_{i-1}^n)(U_{i+1}^n - U_i^n)\}] \tag{3.5}$$

In the original Lax-Friedrich scheme, c^{max} is replaced by $\Delta x/\Delta t$ which makes that scheme more diffusive. TVDLF does not use a Riemann solver and can therefore be applied to any system of conservation laws without knowledge of the characteristic waves. The TVD property ensures that the total variation does not increase with time

$$\sum_j |\Delta U_{j+1/2}^{n+1}| \leq \sum_j |\Delta U_{j+1/2}^n|, \tag{3.6}$$

and, a TVD scheme is monotonicity preserving. Detailed comparisons of Flux Corrected Transport (FCT) and TVD spatial discretization schemes can be found in [15]. Moreover, VAC offers choices for explicit, semi-implicit, or fully implicit time stepping on 1, 2 and 3 dimensional problems. For evaluations of semi-implicit and fully implicit time integration strategies against explicit time advancing, we refer to Tóth et al. [16]. For maintaining the $\nabla \cdot \mathbf{B} = 0$ constraint in multi-dimensional MHD calculations, a rigorous analysis of existing and newly developed strategies when using shock-capturing schemes is described in [14].

The dimension-independence of the different numerical schemes and equations in VAC, and of the AMR algorithm in AMRVAC, is realized by the use of the original Loop Annotation SYntax or LASY [13]. Both source codes can easily be configured to use a different set of equations and then preprocessed to 1, 2 or 3 dimensional configurations. The LASY syntax is combined with a subset of `Fortran 90/High Performance Fortran` language elements: data parallelism expressed by array syntax, `FORALL` statement, the `WHERE` and `CASE` construct, and the scalar valued `SUM, MAXVAL, MINVAL` functions. For VAC, other language elements, like dynamic allocation, and pointer structures are not used. VAC's performance and virtually linear scaling on distributed memory systems is reported in Keppens and Tóth [8].

In contrast, AMRVAC exploits more features of the `Fortran 90` language, including `MODULE`, derived types, dynamic allocation, and pointers. AMRVAC can be used on (virtually) shared memory platforms. This includes parallel execution on `ccNUMA`[1] architectures, e.g. the SGI Origin 3800, where we can make use of `OpenMP`. The code's present data structure is discussed below. For distributed memory execution, further developments of AMRVAC are under consideration.

AMRVAC: Data structure and implementation issues

AMRVAC uses linked lists to keep track of the levels and grids used at any particular time. Each grid and each level corresponds to a node, which is a list of a few informations about the level or grid. Each grid node points to a next grid on the same level, which makes it easy to find all grids on a given level. Levels are consecutively numbered from base level 1 to the finest level, l_{MAX}, and their nodes point to the first grid on the corresponding level (Fig. 2(*right*)).

Each grid `GRID` is associated with two lists – one `INTEGER` array `NODE` and one `REAL*8` array `RNODE` – containing information on the grid. Table 1 shows the fields of the `INTEGER` information array `NODE`, such as the number of grid points in each direction `NODE(pnx^D_,GRID)` and the level to which it belongs `NODE(plevel_,GRID)`. The array also has a pointer to the next `NODE(psibling_,GRID)`, and previous grid `NODE(psiblingrev_,GRID)` on the same

[1]Cache Coherent Non-Uniform Memory Access.

Table 1. The fields of the array NODE(nodehi,ngridshi) containing info per grid of dimensions and pointers.

		NDIM=1	NDIM=2	NDIM=3
integer,parameter:: nodehi	=^ND+7	8	9	10
integer,parameter:: pnx^D_	=^D	1	1,2	1,2,3
integer,parameter:: plevel_	=^ND+1	2	3	4
integer,parameter:: psibling_	=plevel_+1	3	4	5
integer,parameter:: psiblingrev_	=psibling_+1	4	5	6
integer,parameter:: psizenew_	=psiblingrev_+1	5	6	7
integer,parameter:: psizeold_	=psizenew_+1	6	7	8
integer,parameter:: psizeBflux_	=psizeold_+1	7	8	9
integer,parameter:: pinuse_	=psizeBflux_+1	8	9	10

Table 2. The fields of the array RNODE(rnodehi,ngridshi) containing info per grid of corner coordinates, Δx and times.

		NDIM=1	NDIM=2	NDIM=3
integer,parameter:: rnodehi	=3*^ND+2	5	8	11
integer,parameter:: rpxmin0_	=0			
integer,parameter:: rpxmin^D_	=rpxmin0_+^D	1	1,2	1,2,3
integer,parameter:: rpxmax0_	=^ND			
integer,parameter:: rpxmax^D_	=rpxmax0_+^D	2	3,4	4,5,6
integer,parameter:: rpdx^D_	=2*^ND+^D	3	5,6	7,8,9
integer,parameter:: rptold_	=rpdx^ND_+1	4	7	10
integer,parameter:: rptnew_	=rptold_+1	5	8	11

level. In accord with the LASY syntax, the pattern string **^D** is replaced by **1, 2** and **3** in case NDIM=3 and the string **^ND** translates to NDIM. The number of fields nodehi of NODE and rnodehi of RNODE depend on the dimension NDIM.

In the current implementation, solution vectors for the present t_n and previous t_{n-1} time are stored in a linear REAL*8 vector ALLOC starting at locations given by the entries NODE(psizenew_,GRID) and NODE(psizeold_,GRID). The latter is required for error estimation. In the same ALLOC vector, the pointer NODE(psizeBflux_,GRID) is used for storing the fluxes across the circumference of grid GRID counted in level NODE(plevel_,GRID)-1 cells. We allow for a maximal number of grids ngridshi, and active grid numbers are identified by the NODE(pinuse_,GRID) entry.

In Table 2, the entries for the RNODE array are listed. It contains the coordinates of the '*lower left*' RNODE(rpxmin**^D**_, GRID) and the '*upper right*' corners RNODE(rpxmax**^D**_,GRID) and the spatial increments RNODE(rpdx**^D**_,GRID). The times corresponding to the solution vectors mentioned above are stored in RNODE(rptold_,GRID) and RNODE(rptnew_,GRID).

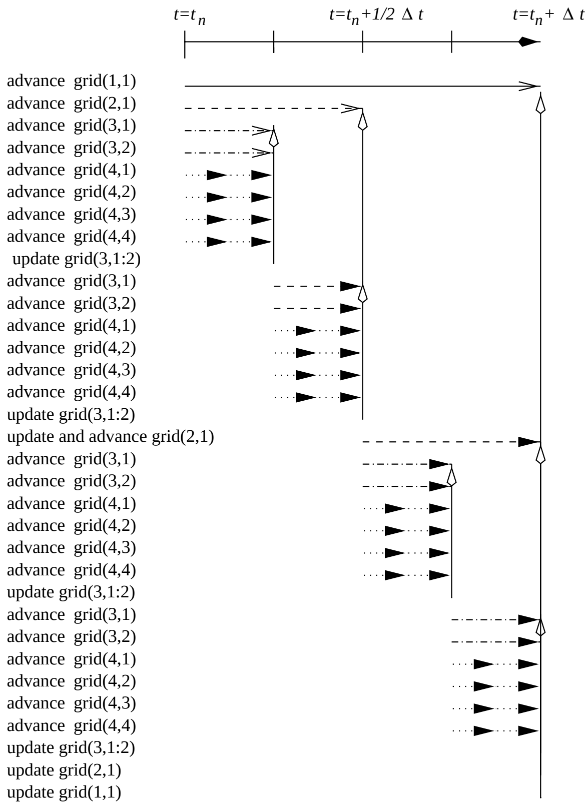

Figure 4. One time step in an AMR simulation for the grid shown in Fig. 2, with refinement ratio $r = 2$; first the coarsest grid is advanced in time with time step Δt; secondly, the level 2 grid with time step $1/2\,\Delta t$; then the level 3 grids with time step $1/4\,\Delta t$, and next twice all level 4 grids with time step $1/8\,\Delta t$. Vertical arrows correspond to update operations of grids on coarser levels ensuring conservation and consistency.

Table 3. Definition of the derived type `IALLOC` and the global array `psizeBlevp1`. One dimension of the pointer field corresponds to the next finer level circumference.

```
module modialloc
  integer,parameter:: ngridshi  = 800
  type ialloc
     integer, dimension(:,:), pointer:: i
  end type ialloc
  type (ialloc) psizeBlevp1(ngridshi)
end module modialloc
```

At level interfaces, a special flux fixing process is needed to ensure global conservation. Therefore, a `MODULE` (see Table 3) passes pointers to allocatable two-dimensional arrays defined for each grid `GRID` and accessible by `psizeBlevp1(GRID)%i`. The two dimensions correspond to the fact that it suffices to reserve `NDIM+3` integer values for the total circumference of the next finer level as measured in `NODE(plevel_,GRID)` cells.

The use of the linear `REAL*8 ALLOC` array involves a self-managing of the available storage space. If a new grid must be created, it is examined if there is enough concatenated room in the allocation array to store the new grid. *Pointers*, being `INTEGER` numbers, indicate the positions in the allocation vector where room is left for new grids or auxiliary arrays. When new storage space is needed, it is identified according to the first-fit algorithm as suggested in Berger [2]. We note that it is possible to abandon the usage of the linear `ALLOC` array all together by a similar construction as shown in Table 3.

As pointed out in Sec. 2, AMRVAC refines in space *and* in time. It is possible to use the same finest level time step on all levels, but this surely increases the computational cost. If this is not done, the temporal refinement requires that the time integration must proceed in a well defined order, such that the solutions of all grids at all levels agree with the physical solution, see Fig. 4. The needed `Update` steps, from level $l+1$ to level l also involve operations at boundaries between coarse and fine grids to ensure the global conservation. The coarse level l cell value is corrected to be consistent with fluxes as obtained by the addition of the adjacent fine cell fluxes.

The entire AMR algorithm – so far restricted to Cartesian grids – is implemented in the LASY-syntax, making it dimension-independent. It is possible to use different (conservative) discretizations on each level and AMRVAC can solve pure advection, full hydro- or magnetohydrodynamic problems. As a result, the code can be applied to a broad variety of plasma (astro-)physical studies. At present, Powell's eight-wave Riemann solver modification, including the source terms proportionate to the numerical magnetic field divergence [10], is one method to ensure a solenoidal magnetic field in AMRVAC. Another one is to essentially diffuse any field divergence by the maximal allowed rate which still satisfies the physics-based CFL criterion. Also relevant is the fact that AMRVAC can make use of the extended visualization macros developed for analyzing VAC data, although the different refinement levels provide some extra complications.

4. Results and efficiency evaluation

4.1. 1D Hydrodynamic shock tube

We start with an evaluation of the AMR efficiency on a pure Riemann problem for the 1D Euler system as found in Harten [4]. With a ratio of specific heats $\gamma = 1.4$, the left and right states are

$$\begin{aligned}(\rho, \rho v_x, e)_L &= (0.445 \quad , \quad 0.311 \quad , \quad 8.928 \quad)\\ (\rho, \rho v_x, e)_R &= (0.5 \quad , \quad 0 \quad , \quad 1.4275 \quad).\end{aligned}$$

With the domain $x \in [0, 14]$, the left state holds for $x < 8$ at $t = 0$, with open boundaries for $t \geq 0$. We fix the tolerance $\epsilon_{\mathrm{tol}} = 0.005$, select for the regrid parameters $\mathtt{E} = 0.6$ and $\mathtt{E}_{min} = 0.5$ and check for refinement every 4 time steps per level ($l < l_{\mathrm{MAX}}$). We then simulate using the second order TVD Lax-Friedrich (TVDLF) scheme [15] up to $t = 2$, using different spatial resolutions with and without AMR. The results are summarized in Table 4, where corresponding simulations are grouped. The first column reports on the number of levels used, the second column gives the mesh size of the finest level and the third column

Table 4. Efficiency table for 1D hydrodynamic shock tube problem. Timings for a single processor of SGI Origin 200, 270 MHz, MIPS, R12000.

# level	$dx_{\mathtt{MAX},1}$	# time steps	regrid in %	exec. time	max. space	# cells in time
1	1/20	469	-	0.720	2592	131320
2	1/20	235	13.31	0.572	1734	55332
1	1/40	939	-	2.278	5112	525840
2	1/40	469	14.46	1.656	3174	184280
3	1/40	235	17.29	1.398	2244	107876
1	1/80	1878	-	8.017	10152	2103360
2	1/80	939	14.41	4.523	5694	629504
3	1/80	470	17.94	3.605	3540	284528
4	1/80	235	19.33	3.370	2544	207348
1	1/160	3757	-	30.930	20232	8415680
2	1/160	1878	14.63	13.417	10734	2310992
3	1/160	939	18.10	8.888	6060	828296
4	1/160	470	20.41	7.913	3858	478616
5	1/160	235	21.04	7.700	2916	401300
1	1/320	7513	-	123.464	40392	33658240
2	1/320	3756	14.99	45.275	20814	8828064
3	1/320	1878	17.79	22.498	11100	2707456
4	1/320	939	20.45	17.896	6378	1215600
5	1/320	470	21.80	16.872	4212	863456

the number of time steps on the coarsest level. In all cases, the step size depends on the CFL-condition and we remark that decreasing the mesh size with a factor of 2 results in about twice as much integration steps. The fourth column reports on the percentage of the execution time required for regridding, including error estimation. The execution time in seconds is given in the fifth column. We also report on the maximum space usage in the linear `REAL*8 ALLOC` array, as an indication of the actual memory needs. The last column shows the number of cells advanced in time, from $l = 1$ up to the finest level. We note that this number is always dominated by the highest level(s) allowed: e.g. for the 401300 cells reported in the case with $l_{\mathrm{MAX}} = 5$, 51.6 % are $l = 5$ cells, with 23.7 % for $l = 4$, and 11 % on $l = 3$.

Clearly, as soon as the base resolution is high enough, it is computationally more efficient to use AMR. This is reflected both in the execution time and in the space usage. The accuracy of the solution at the location of the discontinuities (contact and shock) is nearly identical between the high resolution static and the different AMR solutions. Figure 6 demonstrates this clearly, by comparing the density profile at $t = 2$ for two resolutions included in Table 4. In each case, the high resolution solution is plotted on top of the AMR result. Since the continuous density variation in the rarefaction (near $x = 4$) is not triggering local refinement under the given tolerance, the local deviations between static and AMR solutions correspond to the difference in resolution. At the discontinuities, the solutions are identical as long as the AMR refinement criterion enforces the highest level grids to track these features. This is true at the location of the shock (near $x = 13$) for both cases shown in Figure 6. For the contact discontinuity, the AMR calculation with $dx_{\mathrm{MAX},1} = 1/320$ eventually coarsened the grid structure so that a slight difference is noticeable there.

By dividing the mesh size in the non-AMR case by a factor 2 the number of cell advances will be about 4 times more since the time step will be halved, too. However in the AMR

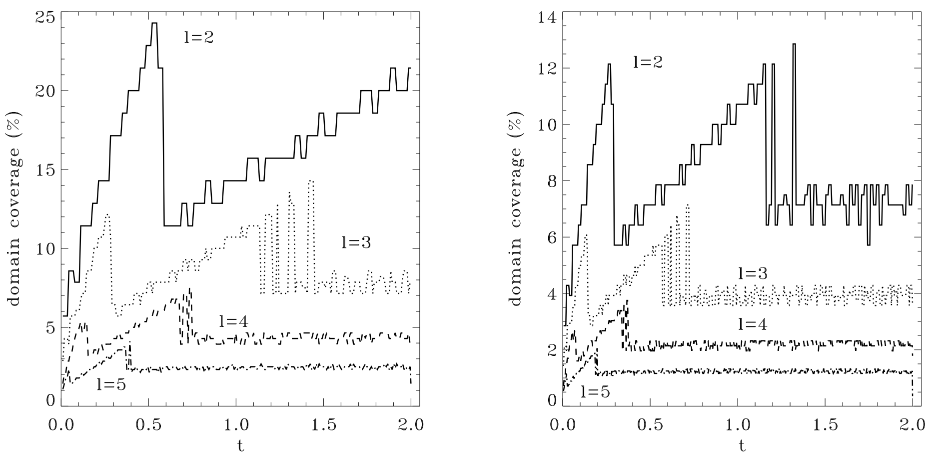

Figure 5. Domain coverage by $l > 1$ grids for 1D Harten problem with 5 levels and on the finest level $dx = 1/160$ (*left*) and $dx = 1/320$ (*right*). Note the different coverage ranges.

Table 5. Efficiency assessment for 2D hydrodynamic problem of Woodward and Colella. Timings for a single processor of SGI Octane2, 360 MHz, MIPS, R12000.

# level	$dx_{\mathtt{MAX},1} = dx_{\mathtt{MAX},2}$	# time steps	regrid in %	exec. time	max. space	# cells in time
1	1/48	258	-	19.7	122304	2377728
1	1/96	531	-	239.6	465600	19574784
2	1/96	266	14.26	68.2	320384	6124896
1	1/192	1086	-	2867.9	1815744	160137216
2	1/192	543	16.91	521.9	829600	36752432
3	1/192	272	15.22	309.6	736656	22015728
1	1/384	2219	-	17506.5	7170240	1308819456
2	1/384	1109	20.80	6098.1	2421896	238488352
3	1/384	555	15.78	1522.3	1705584	108836464
4	1/384	278	16.11	1240.1	1629096	92845680
5	1/768	285	19.20	6740.3	3649224	394630368

case, this factor will be much smaller, for instance for 5 levels this factor is 2.15 and this is also reflected in the execution time. The domain coverage by the different $l > 1$ levels is plotted as a function of time, for the AMR simulation with 5 levels and $dx_{\mathrm{MAX},1} = 1/160$ in Fig. 5(*left*) and $dx_{\mathrm{MAX},1} = 1/320$ in Fig. 5(*right*). Due to the nesting criterion the lines will never cross. Note that about 3% of the domain is covered by level 5 grids ($dx = 1/160$) and less than 2% by level 5 grids ($dx = 1/320$), whereas most of the operations are performed on those cells.

4.2. 2D Hydrodynamic simulations

A well-known shock dominated hydrodynamical test is the 2D reflected shock introduced by Woodward and Colella [17]. On a domain $[0, 4] \times [0, 1]$, a planar Mach 10 shock with post- and pre-shock states given by

$$\begin{array}{lcl}(\rho, \rho v_x, \rho v_y, e)_{\mathrm{post}} & = & (\ 8 \ ,\ 8 \times 8.25 \sin 60^\circ \ ,\ -8 \times 8.25 \cos 60^\circ \ ,\ \mathcal{E}\ ) \\ (\rho, \rho v_x, \rho v_y, e)_{\mathrm{pre}} & = & (\ 1.4 \ ,\ 0 \ ,\ 0 \ ,\ \frac{1}{\gamma-1}\ ).\end{array}$$

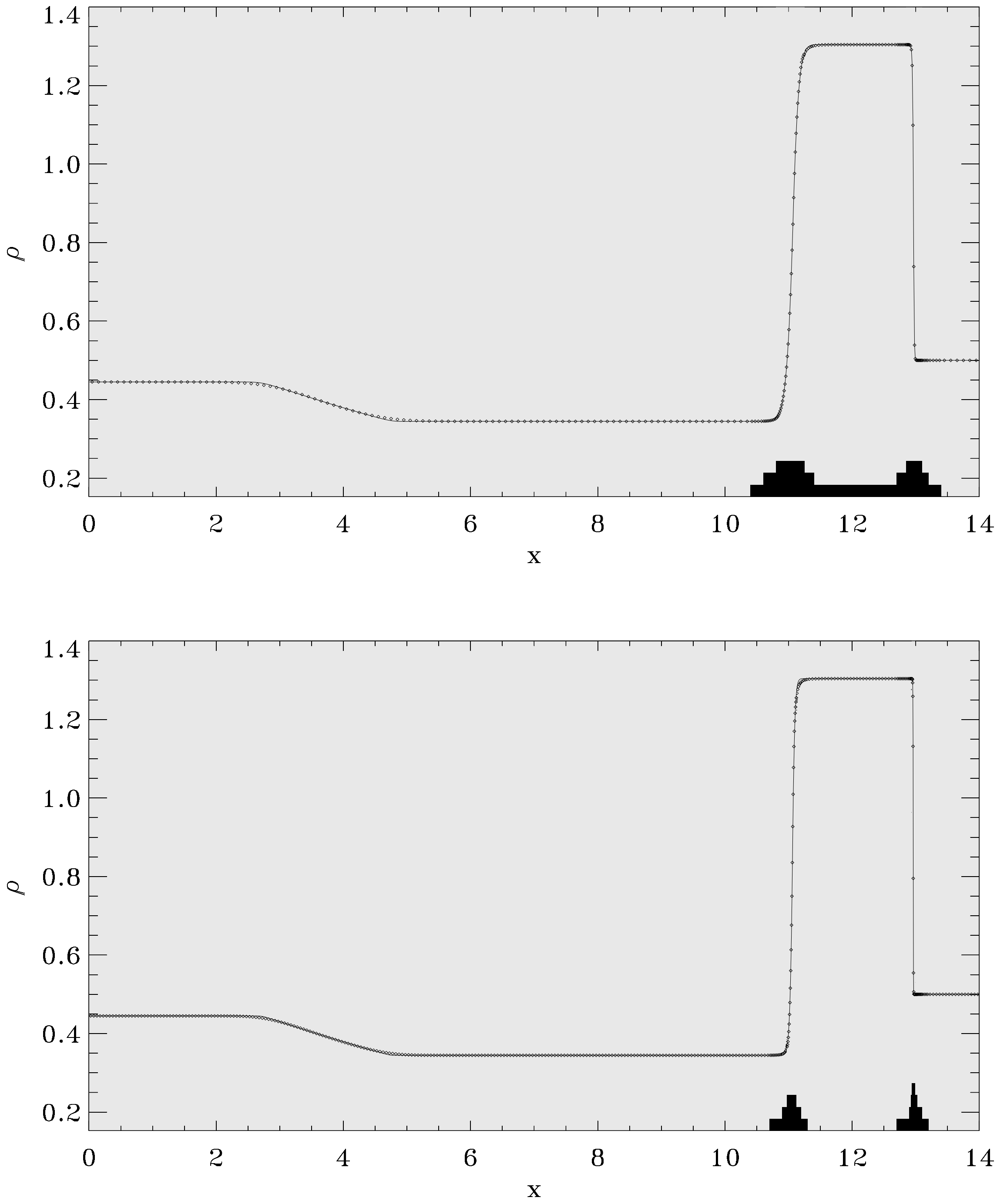

Figure 6. Comparison between high resolution static and locally refined AMR solutions for the 1D Hydrodynamic shock tube problem. Shown is the density profile at time $t = 2$, for runs with $dx_{\mathrm{MAX},1} = 1/80$ (top panel) and $dx_{\mathrm{MAX},1} = 1/320$ (bottom panel). In each panel, the AMR solution is plotted with diamonds, while the corresponding static solution is given by the solid line. The extent of the $l > 1$ AMR grids is indicated as well.

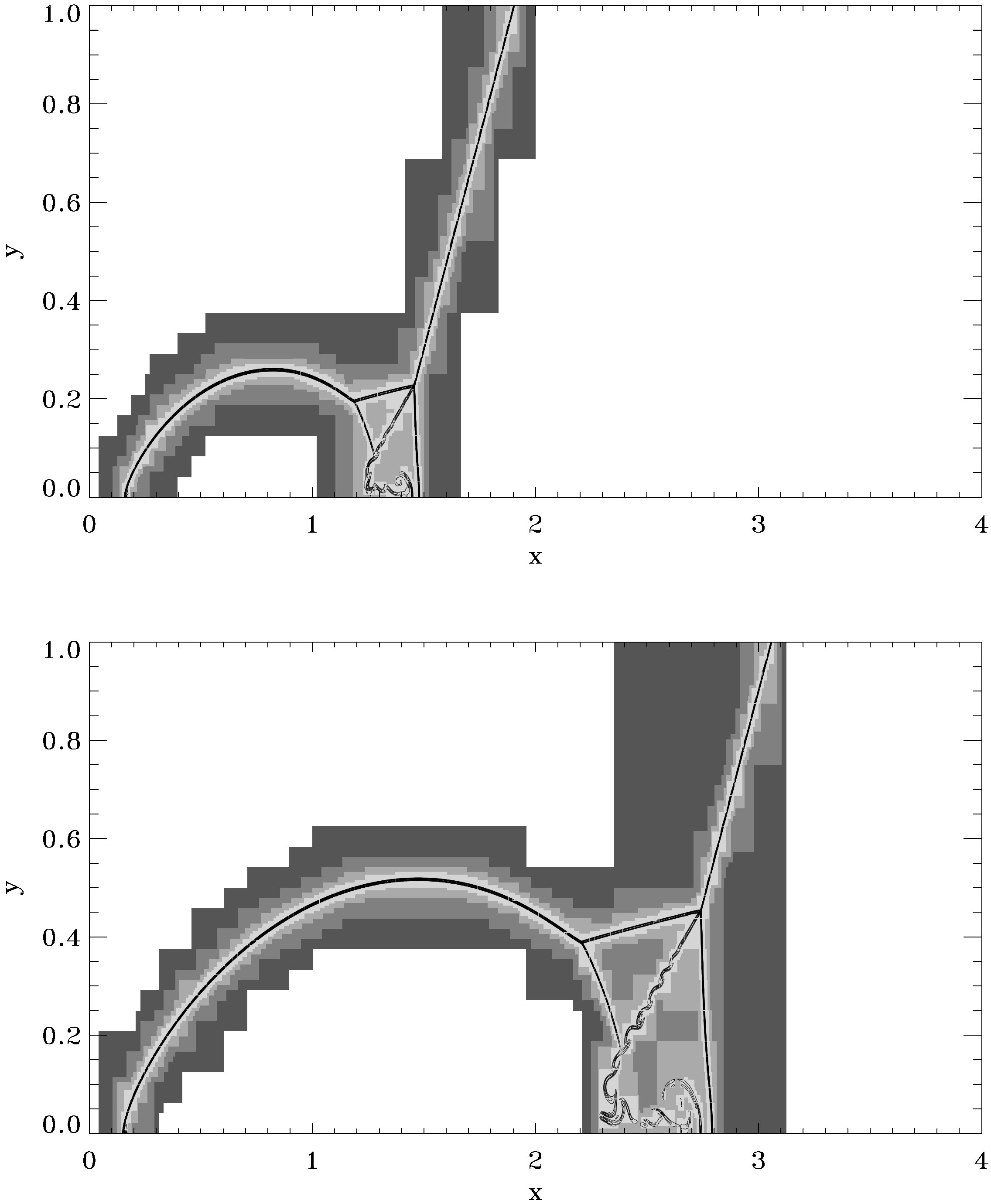

Figure 7. 2D hydrodynamic shock reflection with finest resolution $dx_{5,1} = dx_{5,2} = 1/768$. We show a Schlieren plot of the density field at times $t = 0.1$ and $t = 0.2$.

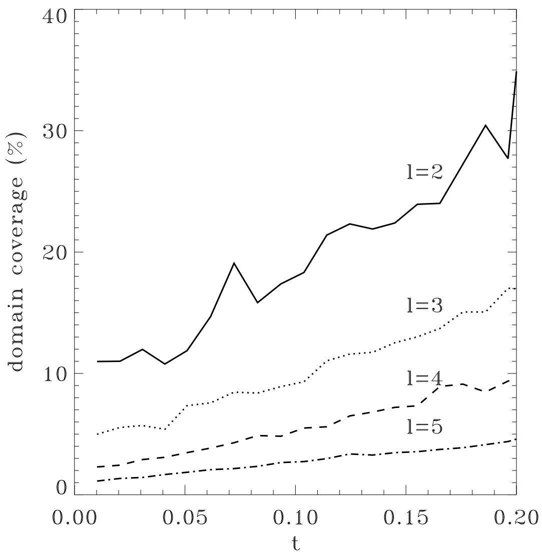

Figure 8. Domain coverage by $l > 1$ grids for 2D HD Woodward and Colella with 5 levels and on the finest level $dx_1 = dx_2 = 1/768$.

makes an angle of 60° with a reflective boundary for $x \in [1/6, 4]$ located at $y = 0$, where $\mathcal{E}$ is given by $\frac{1.165}{\gamma-1} + \frac{(\rho v_x)^2+(\rho v_y)^2}{2\rho}$. The adiabatic index is $\gamma = 1.4$. A self-similar pattern develops while the shock reflects of this bottom wall. Note that the problem setup involves fixing boundary values to the post-shock state at the left edge, open boundaries at right, a partly fixed, partly reflective bottom edge, and a time- and space-dependent boundary value prescription along the top. In the AMR simulation, we use the TVDLF discretization, and allow for up to 5 levels, with $r_l = 2, \mathtt{E}_{min} = 0.5; \mathtt{E} = 0.8$. We fix the tolerance $\epsilon_{\text{tol}} = 0.01$. Fig. 7 shows the density structure as a Schlieren plot at two time steps. An overall self-similar pattern emerges, with small-scale roll-up events in the jet head.

The gain (see Table 5) of adaptive mesh refinement in time (up to a factor of more than 14 for 4 levels with a smallest mesh size of $dx_{\mathtt{MAX},1} = dx_{\mathtt{MAX},2} = 1/384$ compared to a one level solution with the same mesh size) and memory is much larger than in the 1D case. Since at all times the AMR simulations successfully track the shock fronts with the highest level grids allowed, the solution locally coincides with the high resolution static simulation. For the highest resolution we did not execute the non-AMR case which would be a 3072×768 run, because these experiments would be too time- and memory-consuming. In all AMR runs, we regrid at each second time step at all levels except the finest level. As a consequence the time spent on regridding is still typically 14–20 % of the execution time.

We note that refinement causes growth of the number of flagged points in both directions, while particularly in 3D, it may be convenient to consider other refinement strategies where the ratio is direction-dependent. Fig. 8 shows that there is an increasing percentage of the domain covered by level 2 grids when using 5 levels and $dx_1 = dx_2 = 1/768$ (cf. Fig. 7). When the grid size is halved the coverage for all levels decreases proportionally.

4.3. 3D grid-adaptive simulations

To quantify obtainable efficiency with grid-adaptive simulations in three-dimensional problems, we first present a timing result of a pure advection problem. We simulated the diagonal advection of a sphere of radius 0.2 in the cube $[0, 1] \times [0, 1] \times [0, 1]$. Inside the sphere, the density at time $t = 0$ was set to 2, while it is 0.5 in the exterior. The advection velocity $\mathbf{v} = (1, 1, 1)$ brings the sphere at $t = 1$ to its original centered position using triple periodic boundary conditions. In Table 6, we compare execution time and memory requirements for a 320^3 static grid simulation with an AMR run exploiting 3 levels, with a 40^3 base grid, and consecutive refinement ratios 2 and 4. Since the coverage of the highest grid level now remains roughly at 10 % throughout the simulation, a significant reduction by a factor of 19.9 in computing time is realized. The time spent on the regridding process is fully negligible, and the AMR simulation benefits from much reduced memory requirements.

Table 6. Efficiency quantification for a 3D periodic advection problem. Timings for a single processor of SGI Origin 3800, 500 MHz, MIPS, R14000.

# level	$dx_{\mathtt{MAX},1-3}$	regrid in %	exec. time	efficiency	solution storage
1	1/320	-	172197	-	$\mathcal{O}(10^8)$ DP numbers
3	1/320	4.14	8653	19.9	1.68×10^7 DP numbers

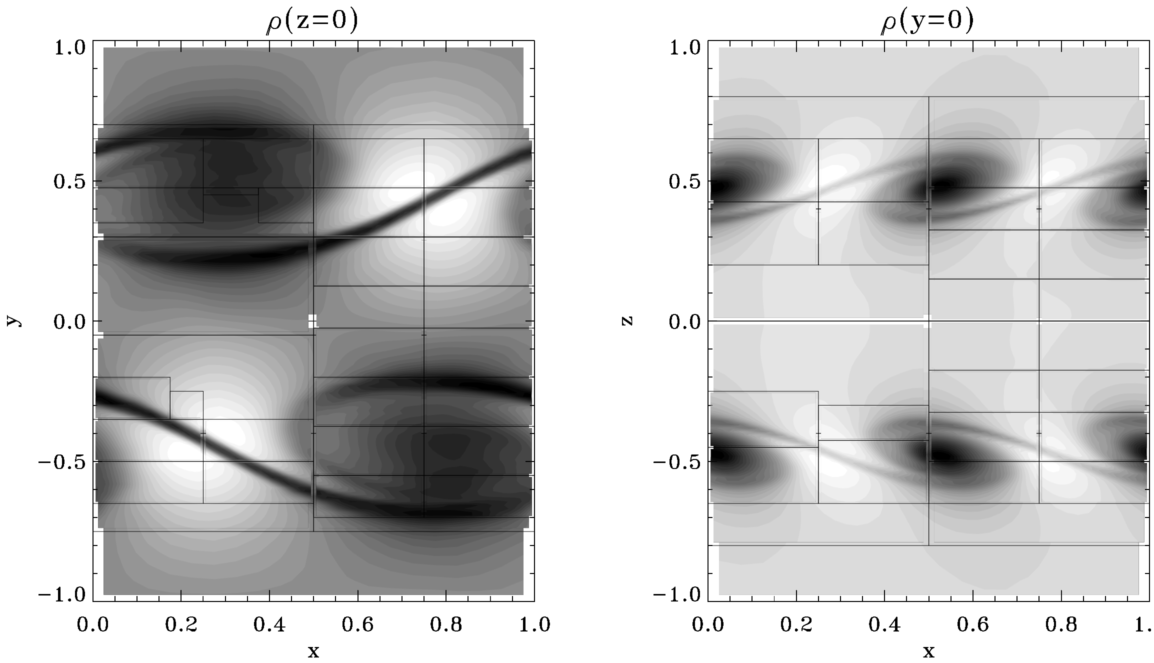

Figure 9. The density structure in a horizontal (left) and vertical (right) cutting plane at time $t = 4$ of a 3D Kelvin-Helmholtz unstable magnetized jet system.

As our final example, we present a demonstrative simulation of a 3D magnetohydrodynamic evolution which fully benefits from the AMR approach. The physical problem considered is taken from Keppens and Tóth [7]: we simulate the nonlinear development of a shear-flow driven Kelvin-Helmholtz instability in a periodic segment of a magnetized jet. In [7], the results were obtained with VAC using a static grid of size $50 \times 100 \times 100$ covering the Cartesian domain $[0, 1] \times [-1, 1] \times [-1, 1]$. The jet of radius $R_{\mathrm{jet}} = 0.5$ is aligned with the x-axis and is initially uniformly, but weakly magnetized parallel to its flow field $v_x = 0.645 \tanh[20(R - R_{\mathrm{jet}})]$. Using AMRVAC on 4 processors of the SGI Origin 3800, we recalculated its evolution in response to preferred wavenumber excitations. This evolution was shown to saturate at about four sound crossing times when locally dominant magnetic structures completely determine the jet deformation. This was most clearly visible in the density structure and Fig. 9 shows a horizontal and vertical cross-section of the density field at this time. They can directly be compared with the frames (A) and (D) in Fig. 3 from reference [7]. In the AMRVAC run, we allowed for three levels where the base grid is only $20 \times 40 \times 40$. Since the refinement ratios between consecutive levels were set to 2, we locally achieve a $80 \times 160 \times 160$ resolution. The refinement criterion exploited a weighted measure of the x-component of the momentum vector and the density field, with a tolerance $\epsilon_{\rm tol} = 0.05$. The grid structure on the higher levels is clearly seen in Fig. 10, a contour plot of the jet-aligned velocity component at saturation. In the AMR simulation, the level $l = 2$ grids start at a 48% coverage which rises to 58% at $t = 4$. The level 3 grids always trace the jet surface of radius $R_{\rm jet}$ with a coverage percentage rising from 24% to 37.5%. For this simulation where regridding occurred every second time step, the AMR overhead amounted to roughly 11% of the total execution time.

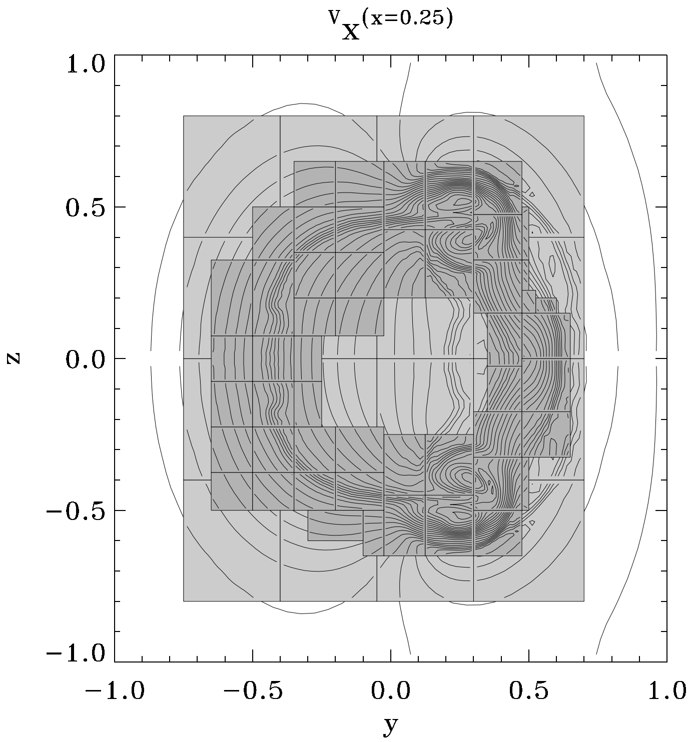

Figure 10. As in Fig. 9, the v_x component of the velocity field in a cross-section of the jet, showing the 3-level grid structure.

5. Conclusions

The efficiency of using AMR for HD and MHD problems in 1, 2 and 3 dimensions was demonstrated and analyzed for a variety of model problems, with execution times reduced by factors of up to 20. The overhead caused by regridding is very small. The data structures employed in the AMRVAC software package, together with its LASY implementation, can be of general interest and were therefore discussed in some detail.

In a forthcoming paper [5], we investigate the advantages of using different (conservative) discretizations on each level, for pure advection, full hydro- and magnetohydrodynamic problems. While we can now efficiently run 3D MHD problems thanks to the AMR strategy, more effort is needed on visualization of particularly 3D data. AMRVAC offers great potential for systematic multidimensional MHD studies where physical processes on vastly different length scales can be expected to play a role. Such large-scope 3D simulations can be run on virtually shared memory systems thanks to the parallelization strategy exploiting `OpenMP` directives.

Acknowledgments

This work was partly done in an NWO Priority Program on Parallel Computational Magneto-Fluid Dynamics coordinated by Prof. J.P. Goedbloed. MN was sponsored by the 'Stichting Nationale Computerfaciliteiten' (NCF, Grant # NRG 98.10). NCF sponsored the use of supercomputer facilities. RK performed this work as part of the association agreement of Euratom and Stichting FOM with financial support from NWO and Euratom.

References

[1] M.J. Berger. *Data structures for adaptive grid generation*, SIAM J. Sci. Statist. Comput., **7** (1986) pp. 904–916.

[2] M.J. Berger and P. Colella. *Local adaptive mesh refinement for shock hydrodynamics*, J. Comput. Phys., **82** (1989), pp. 64-84.

[3] Y.-L. Chiang, B. van Leer and K.G. Powell. *Simulation of unsteady inviscid flow on an adaptively refined Cartesian grid*, AIAA, **92-0443** (1992).

[4] A. Harten. *High resolution schemes for hyperbolic conservation laws*, J. Comput. Phys., **49** (1983), 357.

[5] R. Keppens, M. Nool, G. Tóth, and J.P. Goedbloed. *Adaptive Mesh Refinement for conservative systems: multi-dimensional efficiency evaluation*, submitted to J. Comput. Phys., 2001.

[6] R. Keppens and G. Tóth. *Simulating magnetized plasma with the Versatile Advection Code*, in: *High-Performance Computing and Networking*, *Lect. Notes Comput. Sci.*, **1573** 1999, 680.

[7] R. Keppens and G. Tóth. *Nonlinear dynamics of Kelvin-Helmholtz unstable magnetized jets: Three-dimensional effects*, Phys. Plasmas, **6** (1999), No.5, pp. 1461–1469.

[8] R. Keppens and G. Tóth. *Using high performance fortran for magnetohydrodynamic simulations*, Parallel Comput., **26** (2000), pp. 705–722.

[9] R. Keppens, G. Tóth, and J.P. Goedbloed. *Parallel computation of magneto-fluid dynamics*, in: *Parallel Comput.: Fundamentals & Applications*, Proceedings of ParCo99, (E.H. D'Hollander et al., eds.), 2000, 160.

[10] K.G. Powell, P.L. Roe, T.J. Linde, T.I. Gombosi, and D.L. De Zeeuw. *A solution-adaptive upwind scheme for ideal magnetohydrodynamics*, J. Comput. Phys., **154** (1999), pp. 284–309.

[11] O. Steiner, M. Knölker, and M. Schüssler. *Dynamic interaction of convection with magnetic flux sheets: first results of a new MHD code*, in: *Solar Surface Magnetism* (R.J. Rutten and C.J. Schrijver, eds.), Kluwer, Dordrecht, 1993.

[12] G. Tóth. *A general code for modeling MHD flows on parallel computers: Versatile Advection Code*, Astrophys. Lett. & Comm., **34** (1996), pp. 245.

[13] G. Tóth. *The LASY preprocessor and its application to general multi-dimensional codes*, J. Comput. Phys., **138** (1997), pp. 981–990.

[14] G. Tóth. *The* $\nabla \cdot \mathbf{B} = 0$ *constraint in shock-capturing magnetohydrodynamic codes*, J. Comput. Phys., **161** (2000), pp. 605–652.

[15] G. Tóth and D. Odstrčil. *Comparison of some Flux Corrected Transport and Total Variation Diminishing Numerical Schemes for Hydrodynamic and Magnetohydrodynamic Problems*, J. Comput. Phys., **128** (1996), pp. 82–100.

[16] G. Tóth, R. Keppens, and M.A. Botchev. *Implicit and semi-implicit schemes in the Versatile Advection Code: numerical tests*, Astron. & Astrophys., **332** (1998), pp. 1159–1170.

[17] P.R. Woodward and P. Colella. *The numerical simulation of two-dimensional fluid flow with strong shocks*, J. Comput. Phys., **54** (1984), pp. 115–173.

[18] U. Ziegler. *A three-dimensional Cartesian adaptive mesh code for compressible magnetohydrodynamics*, Comput. Phys. Commun., **116** (1999), pp. 65–77.