Versatile Advection Code

Gábor Tóth

Astronomical Institute, Postbus 80000, 3508 TA Utrecht, The Netherlands
E-mail: toth@fys.ruu.nl, URL: http://www.fys.ruu.nl/~toth

Presenting author: Gábor Tóth

E-mail: toth@fys.ruu.nl
Tel.: +31 30 2535221
Fax: +31 30 2535201

Versatile Advection Code

Gábor Tóth

Astronomical Institute, Postbus 80000, 3508 TA Utrecht, The Netherlands
E-mail: toth@fys.ruu.nl, URL: http://www.fys.ruu.nl/~toth

Abstract. The Versatile Advection Code is a general tool for solving hydrodynamical and magnetohydrodynamical problems arising in astrophysics. The software package uses modern high-resolution shock-capturing numerical schemes to solve a hyperbolic system of partial differential equations with additional non-hyperbolic source terms. Due to its modular structure the code can be easily configured for different sets of equations. Simulations can be done on a general 1, 2 or 3D structured grid. In 1D and 2D both axial and slab symmetry can be assumed for the ignored dimension(s). The source code is written in a dimension independent notation using the new Loop Annotation Syntax. It can be translated to Fortran 90, especially suited for data-parallel computers, or Fortran 77. A user interface based on web browsers, online manuals, and macros for several visualisation softwares make the package complete and user-friendly.

1 Introduction

There is a strong need in the astrophysical community to use numerical simulations as a tool to investigate hydrodynamical and magnetohydrodynamical problems. Developing a suitable numerical code is a time-consuming task that requires expertise both in numerical methods and in programming. The aim of the Versatile Advection Code (VAC) is to provide the community with a modern, versatile yet efficient, and user-friendly software, which can be learnt easily, which can be adapted to the various applications, and which runs efficiently on work stations, vector super computers, and parallel architectures as well.

The general purpose VAC software is being developed by the author since the end of 1994 [10, 11]. More recently R. Keppens joined the software development efforts, and A. van der Ploeg and M. Botchev also contributed significantly to the source code [8, 5]. The philosophy behind VAC is using a single versatile software with options and switches for various problems, rather than developing a different method or version for each problem separately. The advantage of such a general approach is a reduction of overall time for software development, easier maintenance, compatibility of different parts, and automatic extension of new features to all existing applications. Therefore a preference is given to general methods over equation-, problem-, geometry-, or hardware-dependent solutions, still the efficiency and ease of usage are not compromised.

In Sect. 2 the software design, the structure of the source code, and the source language are discussed. Section 3 presents the class of equations that can be modeled. In Sect. 4 the different options for the geometry of the simulation and for the boundary conditions are described. The spatial and temporal discretisation methods are discussed in Sect. 5. Final conclusions are summarised in Sect. 6.

2 Software

2.1 Software Package

VAC is distributed as a coherent software package, with all parts using the same naming conventions, consistent data formats etc. The parts can be used together as well as separately. The visualisation macros, for example, can be used (and are used) for visualising results obtained from other software, or the source code can be preprocessed on one machine, and the translated source compiled and run on another machine, and so on. On the other hand, the user interface provides a uniform and user-friendly environment to do the configuration, to select and edit parameter files, to compile and to run the executables, and to start up the visualisation software.

The software package consists of the following major parts:

- the `vacpp` preprocessor using the Loop Annotation Syntax, and the translator programs `forall2do, f90tof77, f90tocmf` and `f90tohpf`
- the source code for the executables `vacini` and `vac`, which are responsible for the initial condition and time evolution, respectively
- library subroutines for source terms
- user interface used with a web browser via the Weblink web server
- online manuals in Hyper Text Markup Language (HTML) viewable through a web browser
- example parameter files and an example for the VACUSR module
- macros for different visualisation programs (IDL, Matlab, SM)
- various conversion programs for data file formats (Gnuplot, AVS, DX), for namelist formats, etc.

Figure 1 shows the information flow between the various parts. The most original part of this system is the user interface which is written in a mixture of HTML and Perl, for which I coined the word *HyPerlText*. The HTML part describes the visual appearance of the interface, while the Perl commands provide the access to the UNIX shell. The HyPerlText is interpreted by the Weblink web server (product of World-Wide Link Inc, 1996). The advantage of using a web browser as an interface is the high-quality machine-independent graphics, and the fact that it is available virtually everywhere, thus there is no need for installation at all. The Weblink server is written in Perl, which can also be run without any installation procedure.

The compiler/interpreter/software dependence is kept to the minimum. The executables `vac` and `vacini` can be compiled on any machine with a Fortran 90 or a Fortran 77 compiler. The preprocessor requires Perl or Fortran 77, while the

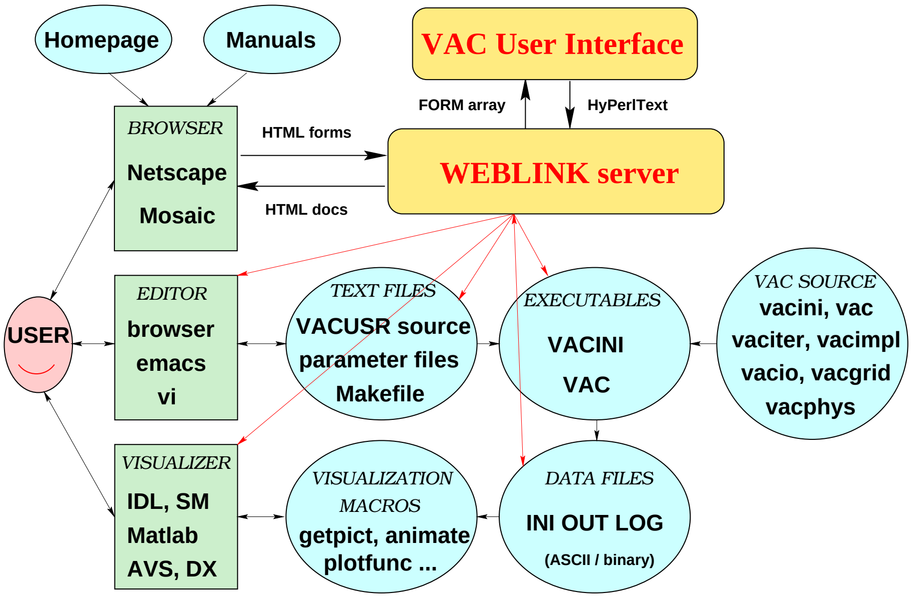

Fig. 1. Information flow-chart for the VAC software package

other translator programs are written in Perl. The manual, about hundred pages, can be viewed with any web browser, e.g. Netscape or Mosaic. The interface, which use is of course optional, needs a Perl interpreter and a web browser again.

VAC has been installed and is being used on a number of workstations (DEC, SGI, IBM, SUN), on the vector super computer Cray C90, and it has been tested for scaling properties on the parallel Connection Machine CM5. The HPF version will be tested on an IBM SP2.

User support is provided in form of up-to-date information on my homepage, (as indicated in Fig. 1), and the user can communicate me by E-mail.

2.2 Modules

The source code for `vac` and `vacini` is also highly structured (see Fig. 2). It is divided into smaller units, called *modules*, though they are not real Fortran 90 modules, since that would not allow an automatic translation to Fortran 77. Some modules have several versions, for example the VACPHYS module which contains the equation specific information: fluxes, sources, the eigenvalues and eigenvectors for the approximate Riemann solver, etc. Each system of equations are specified by its respective VACPHYS.EQUATION module.

The boundary conditions, the initial conditions, and the source terms are often problem specific. If the general subroutines provided in `vacini` and `vac` are not sufficient, the user can write one or more of the *special* subroutines which are collected in the VACUSR.PROBLEM modules. This is the only part of the

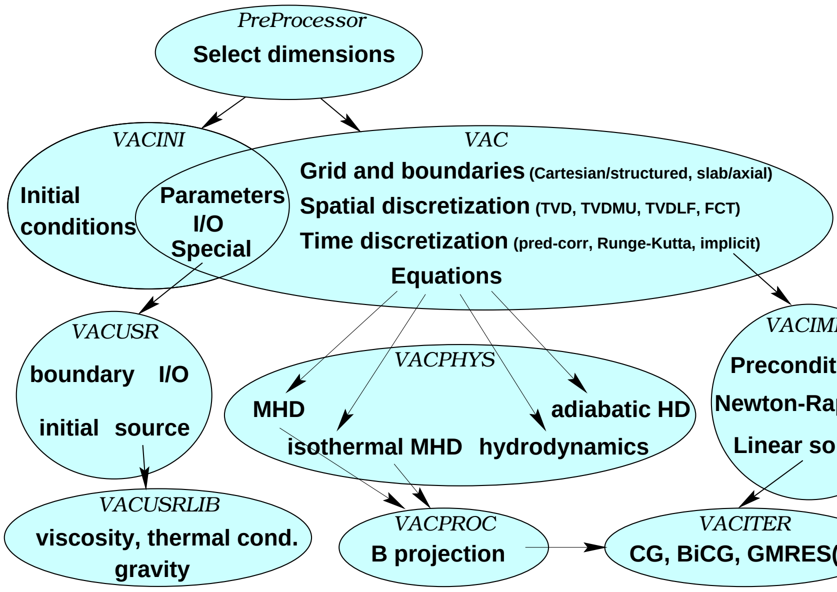

Fig. 2. The structure of the source code

source code which the user is expected to modify, and it can be written in Fortran 90, in Fortran 77, or in the dimension independent notation in which the rest of the code is implemented.

2.3 Source language

The dimension independent notation is realised through the Loop Annotation SYntax (LASY) [12]. LASY allows the programmer to define arrays with an arbitrary number of dimensions, and to use a compact notation for regularly repeated syntax with minor variations, such as indices, array sections, vector components etc.

The special-purpose preprocessor `vacpp` translates LASY to a data parallel Fortran 90 code extended with `forall` statements. The `f90tocmf` Perl script can translate this to CM Fortran and insert the data distribution directives into the source. A similar `f90tohpf` program, which will insert the High Performance Fortran (HPF) directives, is under development, and it is expected to be ready by the beginning of 1997. Parallelisation using the message passing paradigm is also considered. The simple `forall2do` Perl script can replace the `forall` constructs by `do` loops for Fortran 90 compilers that do not understand `forall`. The source can also be translated to Fortran 77 by the `f90tof77` program for machines that have no Fortran 90 compiler.

2.4 Executables and Data Files

The selected modules are translated by the preprocessor and compiled into the two executables `vacini` and `vac`. The advantage of separating the task of initialisation and time evolution is threefold: the same initial data file can be used several times by `vac` without recalculation, the result of a time evolution can be used as an initial condition, and finally the initial condition can be provided by another program than `vacini`. Both `vacini` and `vac` read their parameters from simple and flexible ASCII parameter files that can be easily edited by the user.

The simulation results can be stored either as machine independent ASCII files, or in binary format, which is more accurate, requires less storage, and can be written and read faster. The visualisation macros can read, process, and visualise the ASCII/binary data files for the visualisation softwares IDL, Matlab and SM (SuperMongo). Conversion for Gnuplot, and to AVS and DX file formats is also supported (the conversion program was written by A.J.C. Beliën).

3 Equations

Though the physical assumptions vary a lot among the applications, in most cases a conservative system of hyperbolic partial differential equations with additional non-hyperbolic source terms

$$\partial_t \mathbf{U} + \sum_i \partial_i \mathbf{F}_i(\mathbf{U}) = \mathbf{S}(\mathbf{U}, \partial_i \mathbf{U}, \partial_i \partial_j \mathbf{U}, \mathbf{x}, t) \tag{1}$$

has to be solved, where $\mathbf{U}$, $\mathbf{F}_i$ and $\mathbf{S}$ are the conservative variables, the fluxes and the source terms, respectively, and i represents the 1, 2 or 3 components of the spatial coordinate $\mathbf{x}$. In hydrodynamics, for example, $\mathbf{U}$ consists of mass, momentum and total energy densities, in MHD the magnetic field components are the additional variables. The source terms may be functions of $\mathbf{U}$, $\mathbf{x}$, and time t, e.g. for external gravitational forces, radiative cooling or heating terms etc. In case of viscosity, magnetic resistivity, thermal conduction (etc.), $\mathbf{S}$ may also depend on the spatial derivatives of $\mathbf{U}$.

Using the conservative form of the equations for numerical discretisation has two major advantages. First, the global conservation of mass, momentum, energy and magnetic induction is ensured, second, appropriate numerical schemes can find weak solutions with discontinuities. In astrophysical problems the flows are often supersonic or super Alfvénic, thus the proper representation of shock waves and other discontinuities is necessary.

At the moment five VACPHYS.EQUATION modules are implemented, these are independent of each other, but the equations they describe can be regarded as special cases of the resistive MHD equations

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\mathbf{v}\rho) = \mathrm{S}_\rho \tag{2}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} + \nabla \cdot (\mathbf{v}\rho\mathbf{v} - \mathbf{B}\mathbf{B}) + \nabla p_{\mathrm{tot}} = \mathbf{S}_{\rho v} \tag{3}$$

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

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

where the total pressure, thermal pressure, and current density are

$$p_{\mathrm{tot}} = p + \mathbf{B}^2/2 \tag{6}$$

$$p = (\gamma - 1)(e - \rho\mathbf{v}^2/2 - \mathbf{B}^2/2) \tag{7}$$

$$\mathbf{J} = \nabla \times \mathbf{B} \tag{8}$$

for an ideal gas with adiabatic index γ.

The full set of equations (2)-(8) is implemented in the VACPHYS.MHD module. If the thermal pressure is calculated from an isothermal assumption $p = \rho T$ instead of (7), the energy equation (4) can also be dropped, and we arrive at the isothermal MHD description of VACPHYS.MHDISO. The Euler equations of hydrodynamics are (2)-(4) and (7) with the magnetic field $\mathbf{B} = 0$, which is in the VACPHYS.HD module. In adiabatic hydrodynamics, VACPHYS.HDADIAB, (2)-(3) are used with $p = C\rho^\gamma$ and $\mathbf{B} = 0$. Finally the simple transport equation (2) with a fixed velocity field $\mathbf{v}$ is implemented in VACPHYS.RHO for test purposes.

The S_ρ, $\mathbf{S}_{\rho v}$, S_e, and $\mathbf{S}_B$ source terms can be defined by a user-written subroutine in the problem dependent VACUSR.PROBLEM module. External gravity, heat conduction, and viscosity are provided as modifiable library subroutines (written by R. Keppens). Note that using the viscosity source terms with the VACPHYS.HD module is identical with the Navier-Stokes equations.

All equation-modules and library source terms are written in the dimension independent notation, thus the number of spatial dimensions $1 \le D \le 3$ and the number of components $D \le C \le 3$ for the vector variables $\mathbf{v}$ and $\mathbf{B}$ may have 6 different combinations. The $C > D$ cases are usually referred to as 1.5D or 2.5D simulations in the literature. C and D are parameters for the preprocessor.

4 Grid and Boundary

The simulations can be done on a 1, 2, or 3D structured grid (Cartesian grid is handled as a special case for sake of efficiency). In the finite volume discretisation the average value $(1/V^k)\int_{V^k} U$ is stored for each cell where V^k is the volume of the k-th cell. Solving (1) requires the calculation of gradients and divergences, e.g. the divergence of velocity is discretised as

$$\nabla \cdot \mathbf{v} = \frac{1}{V^k} \sum_l A^l \sum_i \mathbf{n}_i^l \bar{\mathbf{v}}_i^l\,, \tag{9}$$

where the l index runs through all the interfaces of the k-th cell, A^l and $\mathbf{n}_i^l$ are the interface area and the components of its normal vector, respectively, and $\bar{\mathbf{v}}^l$ is the interface-averaged velocity.

In 1D or 2D one may assume axial symmetry in the ignored direction(s). In axial symmetry the cell volume and interface surfaces are multiplied by the radial distance r measured from the symmetry axis, and some extra terms appear, e.g. the pressure gradient is

$$\nabla_i\, p = \frac{1}{V^k}\sum_l A^l \mathbf{n}_i^l \bar{p}^l \; - \; \frac{p}{r}\delta_{1,i}\,, \tag{10}$$

where the radial direction is taken to be $i = 1$. Expressions similar to p/r appear in the discretisation of terms like $\nabla\cdot(\mathbf{v}\rho\mathbf{v})$ in (3), related to the Coriolis and centrifugal forces, these are defined in the VACPHYS.EQUATION modules.

The boundary conditions are implemented by two layers of ghost cells around the mesh. The values in the ghost cells are updated in every time step, thus the discretised equations can be applied to the cells inside the physical mesh without making special cases for the edges. This way the boundary conditions are fully specified by the subroutines updating the ghost cells, and the boundary conditions become independent of the choice of the numerical schemes.

On each side of the grid the boundary layer is divided into one or more boundary regions. For each boundary region and each conservative variable the type of the boundary is defined in the parameter file for `vac`. At present there are six predefined boundary types: *periodic, symmetric, antisymmetric, fixed, fixed-gradient* and *zero-gradient*. The latter two are approximate representations of an open boundary by keeping the gradients normal to the boundary fixed or zero, respectively. Symmetric and antisymmetric boundary types can be used, among other possibilities, to represent perfectly conducting solid walls. The periodic boundary regions on the opposite sides of the grid may be shifted sideways relative to each other which creates a shear in the periodicity. Time dependent or more complicated boundary conditions can be realised via the *special* boundary type which results in a call of a user defined subroutine in the VACUSR module.

5 Discretisation

5.1 Spatial Discretisation

Shock capturing hydrodynamical schemes have been developed and used for industrial and research purposes for many years, but their generalisation to the MHD equations has been started relatively recently [9, 3, 16].

In VAC four algorithms are available for solving the system of differential equations (1). The Flux Corrected Transport scheme (FCT) [1, 6], two Total Variation Diminishing schemes with a Roe-type [7] Riemann solver (TVD [4] and TVD-MUSCL [14]), and a Lax-Friedrich type TVD scheme (TVDLF) with no Riemann solver [15]. See [13] for a concise description of all these schemes and for the comparison of their performance on several test problems.

All these schemes are explicit, second order accurate in space, and they are able to simulate shocks and other discontinuities as well as smooth flows. Multi-dimensional equations can be solved either by Strang type dimensional splitting, or by adding the fluxes from all directions at the same time. In 2D, for example, the two discretisations of (1) are

$$\mathbf{U}^{n+1} = [\mathbf{I} + \Delta t(\partial_x \mathbf{F}_x + \mathbf{S}/2)]\ [\mathbf{I} + \Delta t(\partial_y \mathbf{F}_y + \mathbf{S}/2)]\ \mathbf{U}^n \tag{11}$$

$$\mathbf{U}^{n+1} = [\mathbf{I} + \Delta t(\partial_x \mathbf{F}_x + \partial_y \mathbf{F}_y + \mathbf{S})]\ \mathbf{U}^n\ , \tag{12}$$

respectively. Here $\mathbf{I}$ is the identity operator, n is the index of the discrete time step, Δt is its length, and $\mathbf{F}_x$, $\mathbf{F}_y$, and $\mathbf{S}$ are the discrete non-linear operators approximating the analytical fluxes and source terms, respectively. The order of the x and y sweeps in (11) is alternated in every time step. The three-dimensional splitting is implemented similarly.

The FCT schemes add artificial diffusion only where it is needed for numerical stability. They are good in resolving contact discontinuities and shock waves, but close to the discontinuities some numerical noise might be generated. The TVD schemes, on the other hand, are very efficient in maintaining monotone profiles by appropriately limiting the slopes of the variables. The TVD and TVD-MUSCL schemes with the Roe-type approximate Riemann solver can resolve discontinuities as well as or better than FCT without the spurious oscillations. The Riemann solvers are, however, specific to the equations. The TVDLF scheme does not need a Riemann solver at all, and though it is somewhat more diffusive than the others, it is also more robust.

For the VACPHYS.MHD and VACPHYS.MHDISO modules a projection scheme [2] is implemented to eliminate the numerically generated divergence $\mathbf{B}$. The $\nabla^2\phi = \nabla \cdot \mathbf{B}$ Poisson equation is solved very efficiently with a conjugate gradient type iterative solver (CG or BiCGSTAB), and then the magnetic field is corrected to $\mathbf{B}' = \mathbf{B} - \nabla\phi$ in every time step.

5.2 Temporal Discretisation

Second order time discretisation is achieved by a 2-step Runge-Kutta scheme for FCT, and a predictor-corrector method, using a Hancock predictor step, for the TVDLF and TVD-MUSCL methods. The single-step TVD scheme is second order in time by making use of the eigenvalues, however for this method source terms are difficult to include with second order accuracy. Higher order Runge-Kutta time integration schemes are also implemented, for certain problems they can be numerically more stable than the two-step schemes.

The latest development in the VAC project are the fully implicit and semi-implicit time integration schemes. A more detailed description of these is given in [8], and test results will be discussed in [5].

6 Conclusions

The code has been extensively tested [13] on standard numerical tests. It is currently used in several research projects, with applications ranging from accretion disks and atmospheres of magnetised stars to solar magnetic flux tubes, coronal holes, prominences, and reconnection. The VAC 2.3 version was distributed to 16 users in July 1996.

Performance was compared with several other codes employing similar FCT or TVD algorithms by solving identical problems on a DEC Alpha work station. In spite of its general design, VAC was faster than or as fast as the other less general programs.

In summary the Versatile Advection Code seems to fulfill its original design specifications by being versatile, efficient, and user-friendly. More detailed and up-to-date information on the code is available on the web at `http://www.fys.ruu.nl/~toth` and `http://www.fys.ruu.nl/~mpr`.

Acknowledgments

The Versatile Advection Code is being developed at the Astronomical Institute at Utrecht. The work has been funded by the Dutch Science Foundation (NWO) in a program of Massive Parallel Processing Applied to Magnetohydrodynamics. The author also receives partial support from the Hungarian Science Foundation, grant F 017313.

References

1. Boris, J. P., Book, D. L.: Flux-corrected transport. I. SHASTA, A fluid transport algorithm that works. J. Comput. Phys. **11** (1973) 38
2. Brackbill, J. U., Barnes, D. C.: The effect of nonzero $\nabla \cdot \mathbf{B}$ on the numerical solution of the magnetohydrodynamic equations. J. Comput. Phys. **35** (1980) 426
3. Dai, W., Woodward, P. R.: Extension of the piecewise parabolic method (PPM) to multidimensional magnetohydrodynamics. J. Comput. Phys. **111** (1994) 354
4. Harten, A.: High resolution schemes for hyperbolic conservation laws. J. Comput. Phys. **49** (1983) 357
5. Keppens, R., Tóth, G., van der Ploeg, A., Botchev, M.: Implicit and Semi-Implicit Schemes in the Versatile Advection Code. (work in progress)
6. Odstrčil, D.: A new optimized FCT algorithm for shock wave problems. J. Comput. Phys. **91** (1990) 71
7. Roe, P. L.: Approximate Riemann solvers, parameter vectors, and difference schemes. J. Comput. Phys. **43** (1981) 357
8. van der Ploeg, A., Keppens, R., Tóth, G.: Block Incomplete LU-preconditioners for Implicit Solution of Advection Dominated Problems. (in this volume)
9. Ryu, D., Jones, T. W., Frank, A.: Numerical magnetohydrodynamics in astrophysics: Algorithm and tests for multi-dimensional flow. Astrophys. J. **452** (1995) 785
10. Tóth, G.: A general code for modeling MHD flows on parallel computers: Versatile advection code. Proceedings of Int. Astron. Union Colloqium 153, Makuhari, Japan, (1995) 471–472
11. Tóth, G.: A general code for modeling MHD flows on parallel computers: Versatile advection code. Astrophys. Lett. & Comm. **34** (1996), 245–250
12. Tóth, G.: The LASY Preprocessor and its Application to General Multi-Dimensional Codes. J. Comput. Phys. (submitted)

13. Tóth, G., Odstrčil, D.: Comparison of some Flux Corrected Transport and Total Variation Diminishing Numerical Schemes for Hydrodynamic and Magnetohydrodynamic Problems. J. Comput. Phys. **128** (1996) 82–100
14. van Leer, B.: Towards the ultimate conservative difference scheme. V. A Second order sequel to Godunov's method. J. Comput. Phys. **32** (1979) 101
15. Yee, H. C.: A class of high-resolution explicit and implicit shock-capturing methods. NASA TM-101088 (1989)
16. Zachary, A. L., Malagoli, A., Colella, P.: A higher-order Godunov method for multidimensional ideal magnetohydrodynamics. SIAM J. Sci. Comput. **15** (1994) 263