Using High Performance Fortran for Magnetohydrodynamic Simulations

R. Keppens, FOM-Institute for Plasma-Physics Rijnhuizen,
P.O. Box 1207, 3430 BE Nieuwegein, The Netherlands,
keppens@rijnh.nl

G. Tóth, Department of Atomic Physics, Eötvös University,
Pázmány Péter sétány 1, Budapest H-1117, Hungary,
gtoth@hermes.elte.hu

December 15, 1999

Keywords: High Performance Fortran, Magnetohydrodynamics

Abstract

Two scientific application programs, the Versatile Advection Code and the HEating by Resonant Absorption code are adapted to parallel computer platforms. Both programs can solve the time dependent nonlinear partial differential equations of magnetohydrodynamics with different numerical schemes. We describe our approach to parallelize the original Fortran 90 source codes using High Performance Fortran. Optimization strategies, scaling results, and comparison with performance on vector supercomputers are discussed. We find that the data parallel approach is quite successful for our applications both in terms of single node performance and parallel scaling. The combined use of VAC and HERA allows us to address complementary aspects of fully three-dimensional plasma dynamics. This is demonstrated for a Kelvin-Helmholtz unstable magnetized cylindrical jet.

1 Introduction

Matter in the universe mainly consists of ionized gas or 'plasma'. In experimental fusion machines, we also find plasma confined by magnetic fields. Consequently, a correct theoretical description of magnetized astrophysical and laboratory plasma has far-reaching fundamental and practical applications. One of the simplest but widely applicable descriptions is provided by magnetohydrodynamics (MHD), i.e. fluid dynamics under the influence of a magnetic field $\mathbf{B}$ (see, e.g. [2]). The latter is not just an additional variable, but a quantity that is completely involved in the dynamics, since magnetic fields can be responsible both for the global confinement as well as for the creation of small scale structure.

The MHD equations are nonlinear partial differential equations describing the temporal evolution of the eight primitive variables density ρ, the three components of the velocity vector $\mathbf{v}$, thermal pressure p, and the magnetic field vector $\mathbf{B}$. In ideal MHD, the magnetic flux enclosed by a closed curve moving with the plasma is conserved. If we incorporate a small but finite resistivity, magnetic field lines can reconnect. Still, the resistive MHD equations can be formulated as a set of conservation laws similarly to the Navier-Stokes equations of viscous hydrodynamics. The MHD formalism allows for three modes of wave propagation with different characteristics, and a variety of shock waves and discontinuities. This leads to a very rich dynamics of 3D time-dependent MHD structures. Particularly challenging for numerical modeling is the simultaneous presence of both very fine-scale and global spatial variations evolving on a number of vastly different time scales. Therefore, the required spatial and temporal resolutions in computational MHD necessitate an approach aimed at vector supercomputers and at massively parallel systems.

A lot of software used for solving the MHD equations is written in the Fortran language. Fortran 77 (F77) has evolved to the Fortran 90 (F90) standard. The advent of distributed memory architectures led to an orchestrated effort to standardize High Performance Fortran (HPF), which is an extension of F90 aimed at parallel computer architectures. A recent review of the HPF history and future can be found in [13]. Among other things, HPF extends the F90 syntax with user control on data distribution, and with parallel `forall` loops (the latter statement is in fact part of the Fortran 95 standard). HPF is a very attractive alternative to explicit message passing libraries like the Parallel Virtual Machine (PVM) or the Message Passing Interface (MPI). The main advantage is in the ease of programming: complicated packaging of data, sending and receiving procedures are not necessary, and synchronizing the processors is also taken care of by the HPF compiler. For scientists, a simple, high level, and portable programming language is strongly preferred over cumbersome and error-prone low level solutions. HPF compilers have the potential of becoming uniformly efficient on arbitrary computer platforms, as they can take advantage of the communication library which is best suited to the particular computer hardware. On the other hand, it is recognized that the current HPF standard does not provide sufficient support for dealing with unstructured mesh applications. Optimized versions of HPF have been pursued [1], which helped to identify which extensions to the language are desirable.

Here, we report on the program development strategy, the optimization steps, and the obtained parallel performance for two MHD application codes: the Versatile Advection Code (VAC) [16, 18] and the HEating by Resonant Absorption (HERA) code [7, 8]. Both software packages are briefly described in Section 2. The size of the softwares and their development history made data parallelism the obvious choice for distributed memory execution. Using only a limited subset of F90 and HPF language constructs, we can run both codes on various distributed memory architectures (work station cluster, IBM SP, Cray T3D and T3E), and compare their parallel performance against vector platforms (Cray J90 and C90).

We detail which F90 and HPF features we used in Section 3. Section 4 starts with an example of three-dimensional plasma dynamics that can be studied with both codes. It continues with a discussion of their scaling properties on a variety of architectures. Section 5 concludes with a comparison between distributed memory and vector machines.

2 Description of VAC and HERA

The VAC and HERA codes are independent computer programs serving different, although similar, purposes. VAC is the more general software in terms of equations, applications, and geometry, while HERA is more specialized and adapted to the solution of the 3D MHD equations in cylindrical geometry. The interaction between the developers led to certain similarities in the strategies taken towards platform independence and parallelization. The two applications together cover a substantial part of the numerical schemes generally used for solving the MHD equations.

2.1 Versatile Advection Code

There is a strong need in the astrophysical community to use numerical simulations as a tool to investigate hydrodynamical and magnetohydrodynamical problems. The aim of the Versatile Advection Code (VAC) [18, 16] is to provide a modern, versatile, and user-friendly software, which can be adapted to the application, and which runs efficiently on work stations, vector supercomputers, and parallel architectures as well. VAC uses various shock capturing numerical methods [17], explicit, semi-implicit, or fully implicit time stepping [11, 21] on 1, 2, or 3 dimensional finite volume grids. In two spatial dimensions both slab and cylindrical symmetry can be assumed in the ignored 3rd dimension. Cartesian and polar grids are handled as special cases.

For MHD applications, the divergence of the magnetic field can be kept (arbitrarily close) to zero with a projection scheme [3] which requires the solution of a Poisson equation. The Poisson equation is solved by the Conjugate Gradient (CG) and the stabilized Bi-Conjugate Gradient (BiCGSTAB) methods. These iterative algorithms work for arbitrary boundary conditions unlike direct solvers based on Fast Fourier Transforms. Also implemented in VAC are other methods for ensuring a divergenceless magnetic field, like Powell's eight-wave Riemann solver [14] and the constrained transport method [5].

The different numerical schemes and equations are implemented in modules. The original Loop Annotation Syntax (LASY) [19] provides a dimension independent notation, thus the same source code can be configured and preprocessed for different geometries with the VAC preprocessor. Further translation to F77, F90, and HPF is done by other Perl scripts, thus VAC can run on any machine with an F77, F90, or HPF compiler. The automatic translation steps involved are illustrated in Figure 1. The implicit time stepping module is not

HPF compatible. In the simulations presented in this paper, only explicit time integration and the Poisson solvers are used.

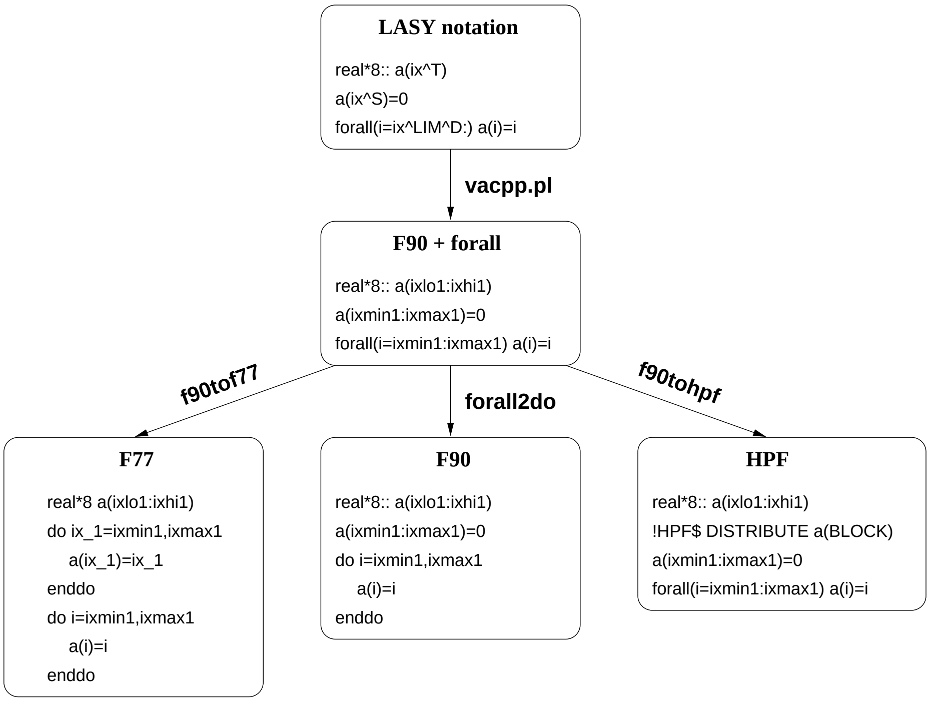

Figure 1: Translation steps of the VAC source code. The names of the translator programs are written next to the arrows, the boxes contain simple examples translated for one spatial dimension.

2.2 Heating by Resonant Absorption

The 3D MHD code HERA is especially suited for studying the nonlinear MHD behaviour of ideal and resistive instabilities and resonance heating layers in magnetic flux tubes [7, 8]. Such magnetized plasma loops can be seen in X-ray observations of the solar corona.

The code assumes a cylindrical coordinate system (R, φ, z) and employs a semi-implicit predictor-corrector time stepping algorithm [7]. The semi-implicit algorithm lifts the severe time step limitation posed by the fast magnetosonic waves. Spatially, the radial direction is discretized with a staggered finite difference mesh, while the periodic poloidal and toroidal directions use a pseudo-spectral discretization. The pseudo-spectral approach advances all variables using a 2D Fourier representation for the periodic directions, while non-linear terms in the MHD equations are evaluated in the real physical space using Fast

Fourier Transforms (FFT) to convert between the spectral and the physical representations. The divergence of the magnetic field can be controlled by determining the radial magnetic field component from the discretized $\nabla \cdot \mathbf{B} = 0$ in the combined real/2D Fourier representation.

This choice of discretization and the particular semi-implicit algorithm used in the time stepping introduces two computational bottlenecks in HERA. Efficient and scalable implementations are needed for the evaluation of 2D FFTs and for solving many (of the order of the number of Fourier modes) complex tridiagonal systems concurrently. It is advantageous to combine platform-specific implementations of these kernels with the full F90/HPF source code. Presently, we can run the code on any platform with a Fortran 90 compiler, and we tested its multi-processor execution on a cluster of workstations, an IBM SP, a Cray T3D and a Cray T3E [8].

3 Software Development Strategies

3.1 Source Language: VAC

VAC uses the following language elements of F90/HPF: data parallelism expressed by the array syntax, the `where` construct, the `forall` statement, and the scalar valued `sum, maxval, minval`, etc. functions. The programming style is improved with the `case` construct, F90 style variable declaration syntax, and free source format. The parameters are read as `namelist`-s, which are very convenient but not fully standardized in Fortran 77. Timing can be done by the F90 `system_clock` subroutine.

This choice of language elements is a good compromise. The selected data parallel constructs are sufficient for the algorithms used in this fluid dynamics code, while other language elements, like dynamic allocation, array valued functions, pointers, structures, etc. are not used, since they would pose difficulties for translation to F77, for the users accustomed to F77, and for many of the not-so-mature HPF compilers. Simplicity helps the users to understand the source code, the compilers to produce an efficient executable code, and the program developers to easily spot and avoid possible compiler bugs.

For explicit time integration on a finite volume grid, data needs to be read from neighbouring grid cells at every time step. We use the F90 array segment syntax to express this, e.g. in one dimension a finite difference formula would look something like `dx(1:n)=(x(2:n+1)-x(0:n-1))/2`. The inter-processor communication is dealt with by the HPF compiler. One could have used the `cshift` function too, but in VAC the compatibility requirement with F77 and the option of using part of the arrays for the computation made the array segments a preferred solution.

To help the HPF compilers to produce an efficient inter-processor communication, the general index shifts used in VAC need to be transformed. We used a semi-automatic translation of general shifts in arbitrary direction to shifts in a known direction combined with IF-ELSE-IF constructs. For example in 2D

```
dx(ixmin1:ixmax1,ixmin2:ixmax2,idim)=0.5D0*\&
     (x(jxmin1:jxmax1,jxmin2:jxmax2,idim)\&
     -x(hxmin1:hxmax1,hxmin2:hxmax2,idim))
```

is translated to

```
IF      (hxmin1==ixmin1-1.and.hxmin2==ixmin2.and.\&
        jxmin1==ixmin1+1.and.jxmin2==ixmin2,idim) THEN
    dx(ixmin1:ixmax1,ixmin2:ixmax2,idim)=0.5D0*\&
          (x(ixmin1+1:ixmax1+1,ixmin2:ixmax2,idim)\&
          -x(ixmin1-1:ixmax1-1,ixmin2:ixmax2,idim))
ELSE IF(hxmin1==ixmin1.and.hxmin2==ixmin2-1.and.\&
        jxmin1==ixmin1.and.jxmin2==ixmin2+1) THEN
    dx(ixmin1:ixmax1,ixmin2:ixmax2,idim)=0.5D0*\&
         (x(ixmin1:ixmax1,ixmin2+1:ixmax2+1,idim)\&
          -x(ixmin1:ixmax1,ixmin2-1:ixmax2-1,idim))
ELSE
   stop 'SHIFT did not optimize!'
ENDIF
```

The resulting code generates efficient local communication instead of global communication.

3.2 Source Language: HERA

For HERA, we did not require backwards compatibility with F77. HERA is written in free source format and uses array syntax, scalar valued and array valued functions, the `include, namelist, implicit none` features, the `where` construct, the `cshift` statement, and the `system_clock` subroutine. In contrast with VAC, HERA heavily uses `cshift` for expressing finite differencing formulae for the radial derivatives.

In HERA, all quantities are time advanced as individual Fourier modes $\hat{f}(m,n,R;t)$ with $f(R,\varphi,z,t) = \sum_{m,n} \hat{f}(m,n,R;t)\exp[i(m\varphi + nz2\pi/L)]$. We use a `(*,*,BLOCK)` distribution for $\hat{f}(m,n,R)$ which spreads the radial dimension across the processors and keeps the Fourier dimensions local. We then inherit perfect linear scaling for the most heavily used code kernel, the 2D FFTs. The pseudo-spectral evaluation of a non-linear term avoids calculating convolutions in Fourier space directly. Instead, we transform the convolution terms separately to real space, evaluate the non-linear term by direct multiplication, and convert this term back to the Fourier domain. The 2D FFTs are performed embarrassingly parallel using a `PURE` subroutine within an `INDEPENDENT` do-loop, and explicitly define `interface` blocks with `intent` declarations. The pseudo-code looks like:

```
     SUBROUTINE four_to_real(trm_real,trm)

     complex*16, dimension(1:mend, 1:nend, 0:nr+1) :: trm_dealias
!HPF$ DISTRIBUTE trm_dealias(*, *, BLOCK) ONTO PP

       INTERFACE
         PURE subroutine local_2dfft(n1, n2, x )
          integer, intent(in)                         :: n1, n2
          complex*16, dimension(n1,n2), intent(inout) :: x
         END subroutine local_2dfft
       END INTERFACE

! copy into larger array for dealiasing
   trm_dealias = .....(trm).....

!HPF$ INDEPENDENT, NEW(ix)
   do ix=0,nr+1
          Call local_2dfft(mend, nend, trm_dealias(:,:,ix))
   enddo

! used complex-to-complex for real quantity
   trm_real(:,:,:)=real(trm_dealias(:,:,:))

   end
```

The `PURE` subroutine *local-2dfft* can be used as a dummy interface to access, e.g. the optimized `f77-local` subroutine *ccfft2d* from Cray library.

We consistently used `cshift` for the finite difference formulae, ensuring that the compiler creates an efficient communication. Communication is also needed when enforcing the boundary conditions, and when solving tri-diagonal systems (TDS) involving the radial direction. Since many of these systems need to be solved concurrently, calculations and communications can overlap, e.g. when using a pipeline strategy. When simply indicating the independence of the TDS for each 2D Fourier mode by putting the tridiagonal system solution stage within two (one for each Fourier dimension) nested `INDEPENDENT` do-loops, the HPF compiler partially succeeds in overlapping calculations with communications.

3.3 Data Distribution and Load Balance

VAC distributes all arrays defined over the computational grid. A structured finite volume grid is used with ghost cells representing the boundary conditions. It is quite easy to make one of the grid dimensions an integer multiple of the number of processors, and use `BLOCK` distribution in that direction, while keeping the other direction local. If the whole array is used in the computation, the load balance will be almost ideal, except for the slight differences in the amount of computation required for ghost cells and physical cells.

Before VAC is compiled with an HPF compiler, the distribution directives are inserted into the source code by the `f90tohpf` Perl script. The arrays to be distributed over the processors are recognized from their declarations, which contain the grid dimensions. Automatic insertion has several advantages over writing down each directive manually. It is easy to forget about an array that should be distributed and to leave it replicated by accident. The resulting code will compile and run correctly, but the excessive communication will destroy the parallel scaling. Another advantage of the automatic insertion is that the distribution directives can easily be changed, e.g. from `(BLOCK,BLOCK)` to `(*,BLOCK)`, which is useful, since the optimal distribution is not necessarily the same for different grid geometries and/or computer platforms.

For HERA, the `(*,*,BLOCK)` distribution is optimal for the pseudo-spectral approach and is permanently present in the source code. As explained above, this distribution ensures that the most heavily used kernel executes in an embarrassingly parallel fashion. Distributing the radial dimension `BLOCK`-wise guarantees an almost perfect load balance as long as the number of radial grid points is an integer multiple of the number of processors.

Since both VAC and HERA distribute the arrays along dimensions in which a finite volume or finite difference discretization is used, `BLOCK` distribution is strongly preferred over `CYCLIC` distribution as it leads to a minimal amount of communication.

4 Performance Tests

4.1 Example application

The generality of VAC has led to a broad range of applications. In one of its earliest applications, VAC was used to evaluate and compare various explicit high resolution numerical schemes on a suite of hydrodynamic and magnetohydrodynamic test problems [17]. More recently, VAC demonstrated the advantages of a problem-specific semi-implicit or fully implicit time stepping approach over explicit schemes for certain steady-state and time-accurate problems [11, 21]. Its applicability to astrophysical fluid flow ranges from solar physics (flux tube dynamics, solar wind, prominence oscillations, reconnection, ...) to the study of accretion flows around black holes. A systematic study of the Kelvin-Helmholtz instability arising in a 2D shear flow configuration is found in [12]. An updated list of publications resulting from the use of VAC can be found at `http://hermes.elte.hu/~gtoth/Papers/vac.html`.

HERA is limited in its applications to the study of magnetized loop dynamics. It has been used to demonstrate the saturation behaviour of internal kink instabilities in closed perfectly conducting cylindrical geometry [7]. In [8], driven loops were investigated where the driving frequency excited a radially localized resonant layer.

As an example application where both codes can yield valuable, complementary results, we here report on a 3D time-dependent study of the Kelvin-

Helmholtz instability of a cylindrical plasma jet. This is a crude model for observed collimated jet-like flows in astrophysical configurations. We set up a flow field $v_z(R) = V_0 \tanh((R - R_{jet})/a)$ parallel to a cylinder of length L and radius R_{jet}, which is sheared in the radial direction. The strength of the velocity shear is $2V_0 = 1.29$, and the width of the shear layer is $2a = 0.2R_{jet}$. Further, the initial pressure p_0 and density ρ_0 are equal to unity everywhere, the ratio of specific heats is $\gamma = 5/3$, and we impose a uniform initial field parallel to the jet like $\mathbf{B} = B_0\hat{e}_z$. Taking $B_0 = 0.129$, we know from previous studies in two space dimensions that this configuration is Kelvin-Helmholtz unstable [12]. When we perturb the cylindrical jet surface, by imposing a radial velocity profile $v_R(R, \varphi, z; t = 0) = \delta v_R \exp -((R - R_{jet})/\sigma)^2 \cos(m\varphi) \sin(nz2\pi/L)$, the instability develops. A vortical flow pattern sets in which redistributes mass and amplifies magnetic fields where the plasma is being compressed. This problem can be studied easily with VAC, where we set up the jet configuration in a Cartesian (x, y, z) box of sizes $L \times 4R_{jet} \times 4R_{jet}$. The first, periodic x-dimension is along the jet, while we assume outflow boundary conditions on the sides parallel to the jet. We set $L = 1$ and use an aspect ratio of $L/R_{jet} = 2$. Similarly, the pseudo-spectral code HERA can simulate the unstable jet, when we put the jet within a co-axial cylinder of radius $a = 2R_{jet}$. In HERA, the equations were made dimensionless by taking $a = 1$, so the loop aspect ratio is then $L/a = 1$. HERA naturally assumes periodic boundary conditions along (z-direction) and about (φ-direction) the loop, while a regularity condition holds at $R = 0$. At a, we took a closed, perfectly conducting wall, which is the only difference in the problem setup between VAC and HERA.

Both codes then were used to study the evolution for an initial perturbation with $\delta v_R = 0.01$, $\sigma = 0.2$, $m = 1$ and $n = 1$. VAC used a $50 \times 100 \times 100$ Cartesian grid, while HERA used 64 radial grid points, and 32 modes both in the φ and z dimensions. When we monitor the evolution of the scaled volume averaged poloidal kinetic and magnetic energies, given by

$$E^{pol}_{mag}(t) \overset{\text{VAC}}{=} \frac{1}{B_0^2 R_{jet}^2 L} \int_V dV \, \frac{B_y^2 + B_z^2}{2} \overset{\text{HERA}}{=} \frac{1}{B_0^2 R_{jet}^2 L} \int_V dV \, \frac{B_R^2 + B_\varphi^2}{2},$$

and similarly

$$E^{pol}_{kin}(t) \overset{\text{VAC}}{=} \frac{1}{\rho_0 V_0^2 R_{jet}^2 L} \int_V dV \, \rho \frac{v_y^2 + v_z^2}{2} \overset{\text{HERA}}{=} \frac{1}{\rho_0 V_0^2 R_{jet}^2 L} \int_V dV \, \rho \frac{v_R^2 + v_\varphi^2}{2},$$

we can easily determine growth rates and the non-linear saturation phase. Growth rates are found from fitting the phase of exponential increase in these quantities, while saturation is reached at their first maximum. Figure 2 shows the evolution of these quantities for HERA and VAC. The agreement is very good and the saturation is reached at about $t = 4$ as judged from both simulations. Similarly, the poloidal magnetic energy agrees well. The deviations between the simulation results can safely be attributed to the difference in resolution, together with the major differences in the numerical schemes used. The distinction in the outer boundary condition (closed cylinder for HERA versus open box for VAC) did not (yet) influence these quantities. HERA also gives immediately the contributions from the individual mode numbers (m, n) to the poloidal magnetic energy, and these are indicated in the figure. In this way, both codes can give complementary information on the complicated 3D dynamics.

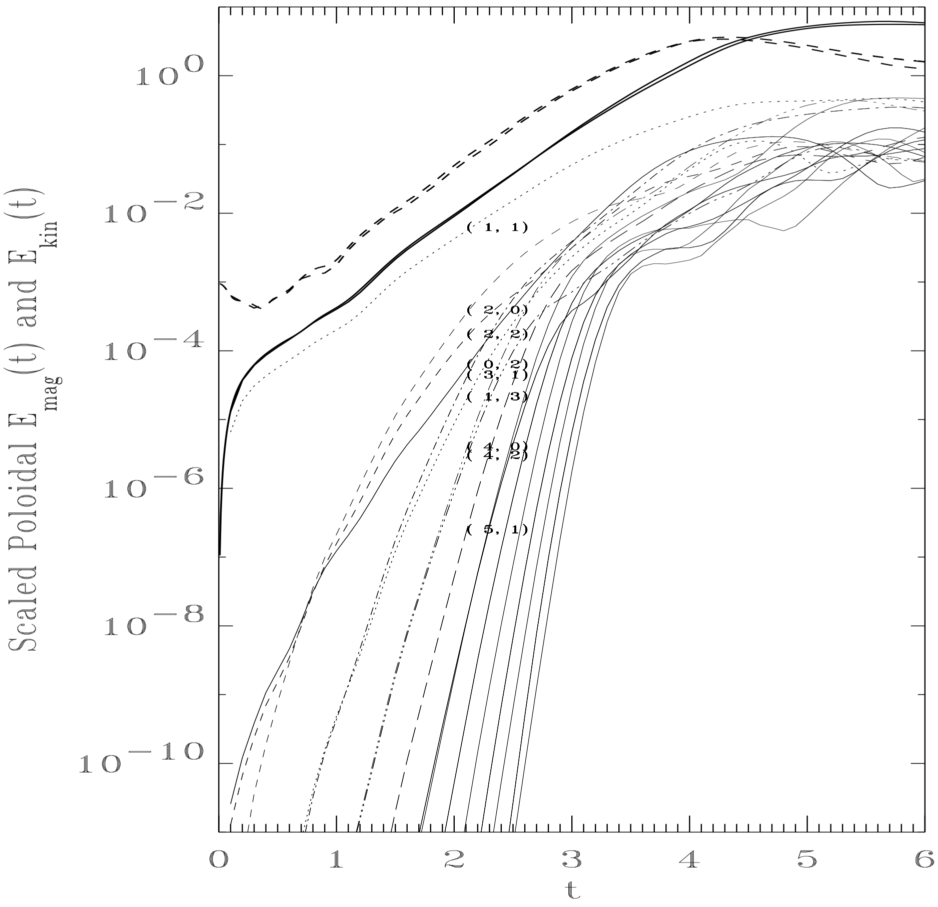

Figure 2: Poloidal kinetic energy (dashed) and magnetic energy (thick solid lines) for HERA and VAC simulating an $m = 1$ perturbation of a Kelvin-Helmholtz unstable magnetized jet. Results for HERA and VAC agree closely. The contributions (thin lines) of the different (m, n) modes are readily available from the HERA code.

The spectral approach is fairly expensive, especially when azimuthally and longitudinally localized structures need to be resolved, which is the case in this 3D Kelvin-Helmholtz evolution. Indeed, an impression of the 3D nature of the instability is shown in Figure 3. Several cuts through the simulation box used by VAC display contours of the initially uniform density at the time of saturation $t = 4$. The dark regions correspond to low density values, while the density is accumulated in the brighter areas. The agreement with the simulation from HERA is demonstrated in Figure 4, where we show the same horizontal cut as the one shown at left in Figure 3 at the same time. The structure is axially shifted (in fact, we used a $\sin(m\varphi)\cos(nz2\pi/L)$ perturbation here), but is further identical due to the periodicity.

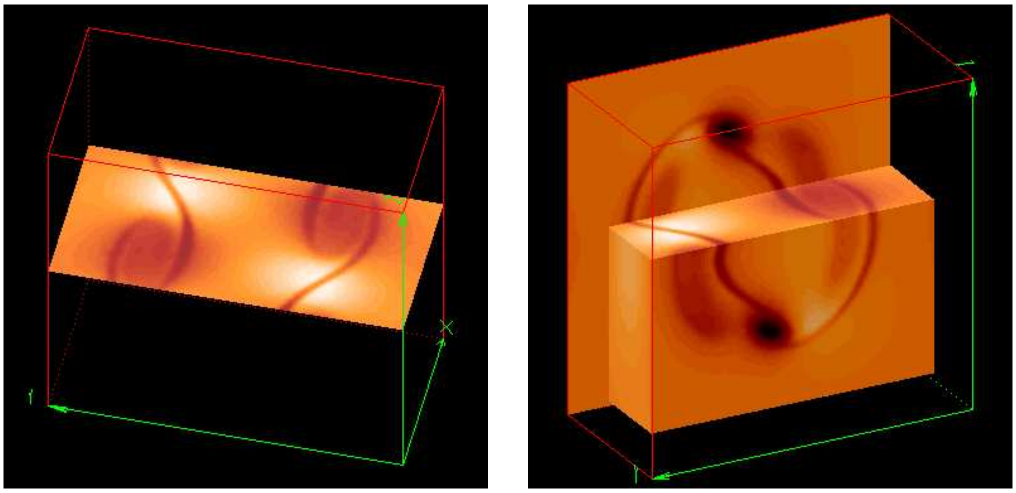

Figure 3: Various cuts in the 3D density structure at the time of saturation $t = 4$. The cuts are through VAC data on a Cartesian $50 \times 100 \times 100$ simulation box with the jet parallel to the x-axis. A horizontal cut slicing the jet midway containing the cylinder axis is shown at left, while a (rotated) 3D view is created at right.

A more detailed study of the intricate plasma dynamics in Kelvin-Helmholtz unstable jets is presented elsewhere [9]. We stress that both codes can be validated against one another for simulations of this kind, and used for quantitative studies. The simulations discussed here were run on 8 processors of both an IBM SP and a Cray T3E platform. In VAC, we used an explicit one-step Total Variation Diminishing scheme [6] with *Woodward* limiting [4] and applied the projection scheme at every time step. This scheme makes use of a Roe-type approximate Riemann solver [15]. In Table 1, we list execution times needed per time step for these 3D simulations with VAC and HERA. All results are for an IBM SP platform and a `(*,*,BLOCK)` distribution. Note how both codes show a good scaling when going from 4, 8 to 16 processors. For dealiasing purposes, the FFTs in HERA operate on doubled Fourier dimensions (64×64 instead of 32×32). For VAC, we also list the percentage used by the projection scheme to ensure a divergence-free magnetic field. The HPF implementation of the Conjugate Gradient (CG) iterative scheme scales very well for this problem size. An acceptable 12 % of the CPU time is spent on solving the Poisson equation independent of the number of processors. It should be noted that the

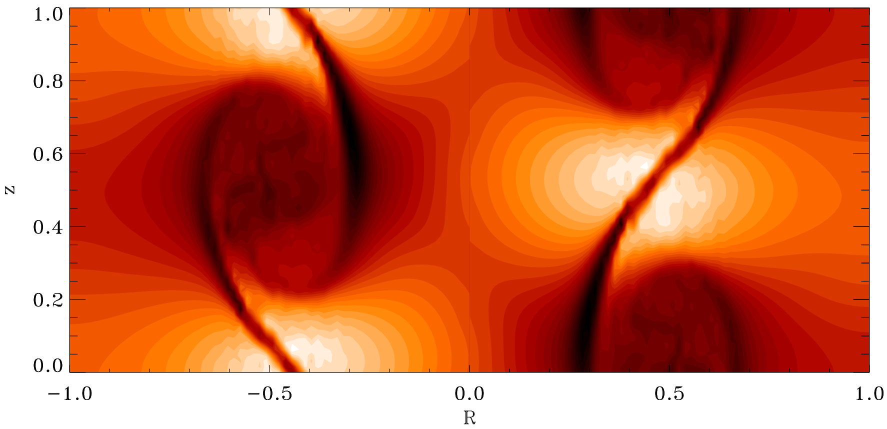

Figure 4: The same cut shown in Figure 3, but now taken from HERA data with resolution $64 \times 32 \times 32$. The agreement is excellent.

	Resolution	4PE	8PE	16PE
VAC	$(50 \times 100 \times 100)$	12.25	6.83	4.95
Poisson solver		11.5 %	11.6 %	11.9 %
HERA	$(64 \times 32 \times 32)$	5.59	3.08	1.63

Table 1: Timings on the IBM SP in seconds per time step for the example 3D application.

`(*,*,BLOCK)` distribution for the VAC run shows a loss in scaling when going from 8 to 16 processors. This is because the third grid dimension, which is 104 when the double layer of ghost cells on each side is taken into account, does no longer ensure a good load balance on 16 processors. In the next section, we demonstrate how we can achieve an almost linear scaling when taking the `(*,BLOCK,BLOCK)` distribution instead. This again emphasizes the advantage of the automatic insertion strategy discussed in Section 3.3.

4.2 Single Node Performance and Parallel Scaling

A detailed performance comparison for running VAC and HERA on several platforms is found in [20, 8]. Here, we summarize and update the main findings.

Figure 5 is an updated version of Fig. 3 in [20], and gives a complete picture of the performance and scaling properties of VAC. We use the Portland group *pghpf* compiler for HPF runs on a Cray T3D, a Cray T3E, and an IBM SP. The figure compares the number of grid cell updates per second obtained on a single processor of two vector (Cray J90 and C90) platforms with those from parallel executions of up to sixteen processors. The scaling improves when going to a larger problem size: the solid lines correspond to a 104×104 shallow water simulation, while the dashed lines are for a resolution of 208×208. For such small test problem, the nearly linear scaling is quite encouraging. For high resolution 3D simulations, we can demonstrate improved scaling properties up to many more processors. As an example, we repeated the timings [10] for the application problem discussed in Section 4.1, but now used the `(*,BLOCK,BLOCK)` distribution when running in HPF. Table 2 summarizes execution times for three parallel platforms: a shared memory multi-processor execution on the Cray C90, and both Cray T3E and IBM SP distributed memory runs. The difference in execution time for the 4 PE run on the IBM SP between Table 1 and Table 2 is partly due to the changed distribution, but also due to slightly changed circumstances (optimization level). One must be aware of the differences in single processor speeds between these three platforms, which can to some extent be deduced from Figure 5. It should also be noted that the $50 \times 100 \times 100$ problem size (alltogether 4 million unknowns), is still quite modest. The code requires roughly 400 MB memory on the Cray C90, where we reach 388.4 Mflops on a single CPU which is 40 % of the theoretical peak performance. When we exploit autotasking on the shared memory C90 (using `f90 -O scalar3,vector3,task3,inline3`), we reach an efficiency (Speedup/Number of PEs) of 97 % on 2 CPUs and 91 % on 4 CPUs. For the distributed memory platforms IBM SP and Cray T3E, we find similarly good speedup. On both platforms, the Portland Group `pghpf 2.4-4` compiler was used with optimization level `-O3`.

Platform	1PE	2PE	4PE	8PE	16PE
Cray C90	8.40	4.32 [$\times$1.9]	2.27 [$\times$3.7]	–	–
Cray T3E	–	–	–	24.3	12.9 [$\times$1.9]
IBM SP	–	–	14.3	7.7 [$\times$1.8]	4.2 [$\times$3.4]

Table 2: Timings for VAC on various platforms and number of PEs, in seconds per time step for the Kelvin-Helmholtz 3D ($50 \times 100 \times 100$) MHD simulation. Speedup factors are given between brackets.

Figure 6 gives a similar overview of the performance and scaling properties for HERA on four different platforms. We determined the number of updates per second as found from 3D ideal MHD calculations of spatial resolution $512 \times 16 \times 16$ and $256 \times 32 \times 32$. The scaling is linear as long as the number of radial grid points is an integer (≥ 2) multiple of the number of processors. This is clearly seen from the Cray T3D result where the largest problem with 256 radial grid points is executed on 8, 16, 32, 64, 128 and 256 processors. Linear scaling persists up to 64 processors, and degrades beyond 128 processors.

Note that we excluded I/O operations from the timing measurements. Experiments indicated that parallel execution combined with ASCII I/O for saving

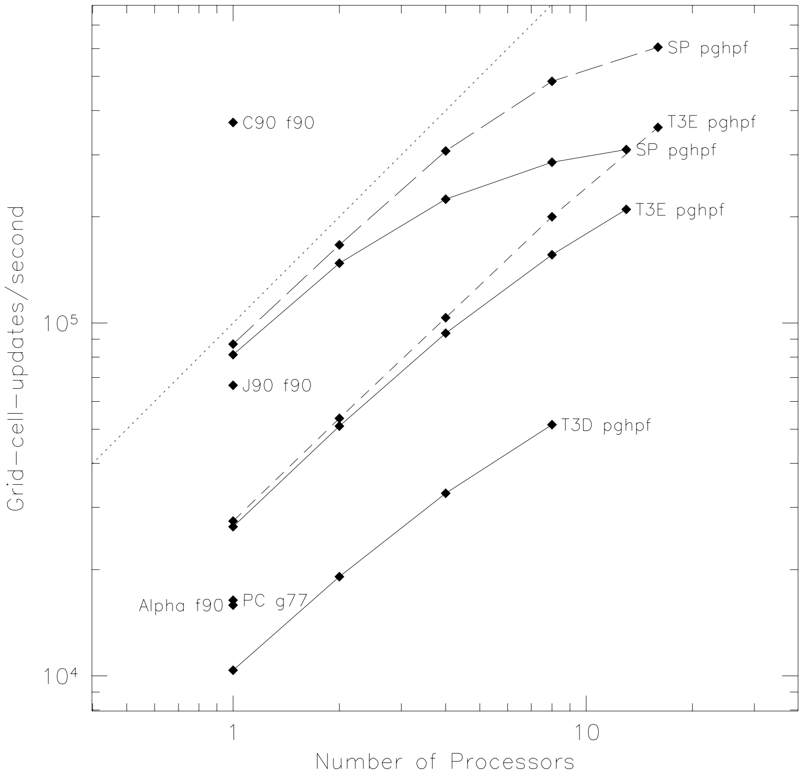

Figure 5: Timing results for VAC for a 104×104 test problem solving the shallow water equations. The number of physical grid cell updates per second is plotted against the number of processors for the different computer platforms and compilers as indicated by the labels. Execution times are measured with the best optimization on 1, 2, 4, 8, and 13 processors. The dotted line indicates the slope corresponding to the ideal linear speedup on the log-log plot. The dashed lines show measurements for a bigger 208×208 grid on 1, 2, 4, 8, and 16 processors. Note the improved scaling.

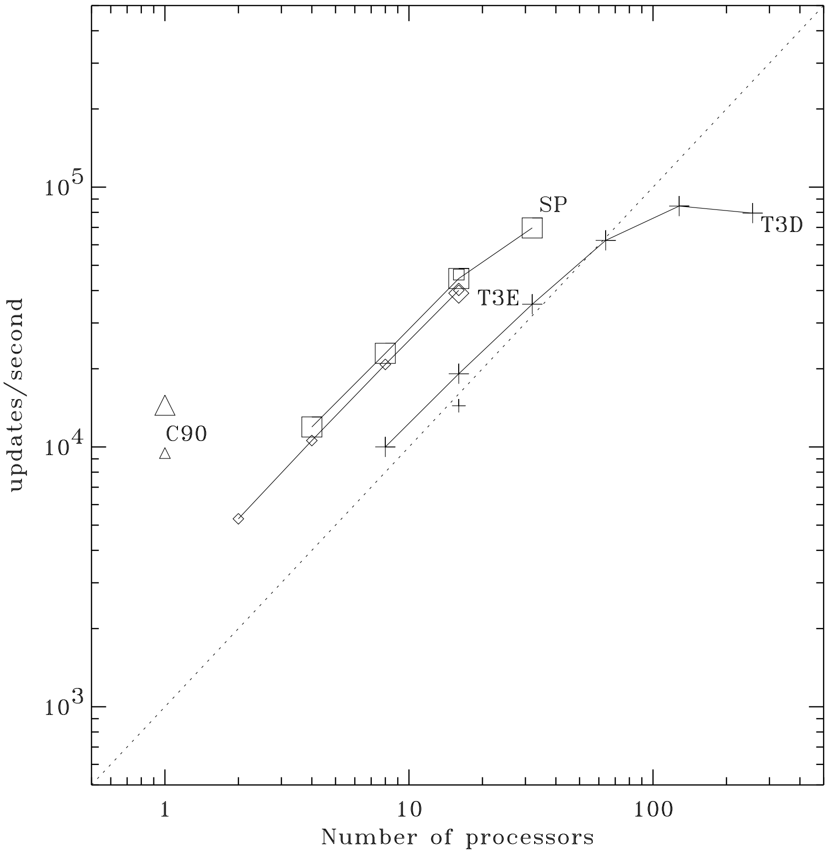

Figure 6: Performance and scaling of the full 3D MHD HERA code for two spatial resolutions, namely $512 \times 16 \times 16$ (small symbols) and $256 \times 32 \times 32$ (large symbols) on four different platforms. Triangles indicate F90 execution times obtained on a Cray C90. Parallel execution results are shown as diamonds for a Cray T3E, squares for the IBM SP, and crosses for a Cray T3D platform. Ideal speedup is indicated by the dotted line.

an individual snapshot of a high resolution 3D simulation can be prohibitively slow (especially on the IBM SP system). Therefore, binary I/O is used throughout for the large datasets. This is faster, and yields more accurate and compact datasets. It would be useful to have standardized parallel I/O possibilities, optimized to the specific platform and taken care of by the HPF compiler.

5 Conclusions

We demonstrated nearly linear scaling on several parallel architectures for both VAC and HERA. As a result, we can simulate complicated three-dimensional plasma dynamics at high resolution. It is of interest to compare how much can be gained when going from a shared memory vector supercomputer (Cray J90 or C90) to distributed memory platforms like the Cray T3E or the IBM SP. Since both codes are designed to run on vector architectures as well, we can compare their performance as judged from running realistic scientific applications. Figures 5 and 6 also contain timings for vector platforms, and it can be seen how at least 4, and typically 8 processors of the Cray T3E and IBM SP are needed to outperform a single node execution on a Cray C90. Obviously, one can only meaningfully exploit the scalability for sufficiently large problems by running these magnetofluiddynamics codes on at least 8 or 16 processors. Since the two codes cover a substantial part of the kind of algorithms used in computational fluid dynamics, we believe that these statements are applicable to a typical scientific CFD code using a structured grid. Of course, the need for sufficient parallelism is also motivated by the memory limitations of shared memory machines. However, for a simulation like the one presented in Section 3, the primary motivation for running in parallel is the faster execution when invoking enough processors, since this problem could still fit in the C90 memory. In fact, Table 2 shows that for the resolutions that can be tackled on the Cray C90, it is best to employ autotasking there.

We find that HPF is an excellent alternative to the message passing paradigm for both general-purpose, and application specific software packages. VAC and HERA are ideally suited for vector and data parallel simulations of magnetized plasmas. The parallelism expressed in High Performance Fortran works well for present day distributed memory supercomputers. This optimistic viewpoint may not be applicable for algorithms exploiting highly irregular data structures [1, 13].

We provided ample implementation details, which should be useful for HPF code developers. During our efforts to parallelize our F90 codes, compiler bugs were identified and resolved in successive releases of the still maturing HPF compilers. Our choice of using only a subset of the HPF language features was very useful in that respect. Most problems occured in the I/O related statements (namelists in particular), or when the HPF compiler is left to figure out the most efficient communication pattern when using indirect index shifts (Section 3). In general, the best solution is to simplify, wherever possible, complicated conditionals, index shifts, etc. This also leads to more streamlined and easily readable software.

The VAC software package is complete with manual pages written in hypertext, a user interface based on web browsers, and visualization macros for the most popular visualization softwares. These visualization macros are used for analyzing both VAC and HERA data, since we have built into HERA the option to output in the same format as VAC. The ever growing number of VAC users and applications proves that the concept of a single well designed general purpose scientific software package is a good alternative to the typical specialized scientific codes. Certain algorithms may warrant a specialized implementation that optimally exploits the parallelism. This is the case for the HERA code, where the pseudo-spectral approach suggests a specific parallelization strategy. We showed how these codes can be used in tandem to study 3D Kelvin-Helmholtz instabilities in astrophysical jets [9]. Plans for further development include the possibility for using Adaptive Mesh Refinement in VAC and the systematic study of 3D jet flows, solar and stellar wind configurations with both HERA and VAC.

Acknowledgements. This work is part of the research program of the 'Stichting voor Fundamenteel Onderzoek der Materie' (FOM) with financial support from the 'Nederlandse Organisatie voor Wetenschappelijk Onderzoek' (NWO). It was part of the project on 'Parallel Computational Magneto-Fluid Dynamics', funded by the NWO Priority Program on Massively Parallel Computing and coordinated by Prof. J.P. Goedbloed. It was sponsored by the 'Stichting Nationale Computerfaciliteiten' (NCF) for the use of supercomputer facilities (Cray C90, IBM SP, Cray T3E). The results quoted for the Cray J90 and the Cray T3D were obtained at the Edinburgh Parallel Computing Centre, under the TRACS program. G.T. currently receives a postdoctoral fellowship (D 25519) from the Hungarian Science Foundation (OTKA) and the Bolyai Fellowship from the Hungarian Academy of Sciences.

References

[1] S. Benkner, HPF+ High Performance Fortran for Advanced Industrial Applications, in: P. Sloot, M. Bubak, and B. Hertzberger, eds., Lecture Notes in Computer Science, Vol. 1401, Proceedings HPCN Europe 1998 (Springer-Verlag, Berlin, 1998) 797-808.

[2] D. Biskamp, Nonlinear Magnetohydrodynamics, Cambridge Monographs on Plasma Physics 1, (Cambridge University Press, Cambridge, 1993).

[3] J.U. Brackbill and D.C. Barnes, Note: The Effect of Nonzero $\nabla \cdot \mathbf{B}$ on the Numerical Solution of the Magnetohydrodynamic Equations, J. Comput. Phys. 35 (1980) 426-430.

[4] P. Collela and P.R. Woodward, The Piecewise Parabolic Method (PPM) for Gas-Dynamical simulations, J. Comput. Phys. 54 (1984) 174-201.

[5] C.R. Evans and J.F. Hawley, Simulation of Magnetohydrodynamic Flows: A Constrained Transport Method, Astrophys. Journ. 332 (1988) 659-667.

[6] A. Harten, High Resolution Schemes for Hyperbolic Conservation Laws, J. Comput. Phys. 49 (1983) 357-393.

[7] R. Keppens, S. Poedts, P.M. Meijer, and J.P. Goedbloed, A Data Parallel Pseudo-Spectral Semi-Implicit Magnetohydrodynamics Code, in: B. Hertzberger and P. Sloot, eds., Lecture Notes in Computer Science, Vol. 1225, Proceedings HPCN Europe 1997 (Springer-Verlag, Berlin, 1997) 190-199.

[8] R. Keppens, S. Poedts, and J.P. Goedbloed, Data parallel simulations of the magnetohydrodynamics of plasma loops, in: P. Sloot, M. Bubak, and B. Hertzberger, eds., Lecture Notes in Computer Science, Vol. 1401, Proceedings HPCN Europe 1998 (Springer-Verlag, Berlin, 1998) 233-241.

[9] R. Keppens and G. Tóth, Non-linear dynamics of Kelvin-Helmholtz unstable magnetized jets: three-dimensional effects, Phys. of Plasmas 6 (1999) 1461-1469.

[10] R. Keppens, G. Tóth, and J.P. Goedbloed, Parallel Computational Magneto-Fluid Dynamics, Accepted for publication in: Proceedings of Parallel Computing 99, August 17-20, Delft, The Netherlands.

[11] R. Keppens, G. Tóth, M.A. Botchev, and A. van der Ploeg, Implicit and Semi-Implicit Schemes: algorithms, Intern. Journ. for Numer. Meth. in Fluids 30 (1999) 335-352.

[12] R. Keppens, G. Tóth, R.H.J. Westermann, and J.P. Goedbloed, Growth and saturation of the Kelvin-Helmholtz instability with parallel and anti-parallel magnetic fields, J. Plasma Phys. 61 (1999) 1-19.

[13] P. Mehrotra, J. Van Rosendale, and H. Zima, High Performance Fortran: History, status and future, Parallel Comput. 24 (1998) 325-354.

[14] K.G. Powell, An approximate Riemann solver for magnetohydrodynamics (that works in more than one dimension), ICASE Report No 94-24, Langley, VA (1994).

[15] P.L. Roe, Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes, J. Comput. Phys. 43 (1981) 357-372.

[16] G. Tóth, A general code for modeling MHD flows on parallel computers: Versatile Advection Code, Astrophys. Lett. & Comm. 34 (1996) 245-250.

[17] 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) 82-100.

[18] G. Tóth, Versatile Advection Code, in: B. Hertzberger and P. Sloot, eds., Lecture Notes in Computer Science, Vol. 1225, Proceedings HPCN Europe 1997, (Springer-Verlag, Berlin, 1997) 253-262.

[19] G. Tóth, The LASY Preprocessor and its Application to General Multi-Dimensional Codes, J. Comput. Phys. 138 (1997) 981-990.

[20] G. Tóth and R. Keppens, Comparison of Different Computer Platforms for Running the Versatile Advection Code, in: P. Sloot, M. Bubak, and B. Hertzberger, eds., Lecture Notes in Computer Science, Vol. 1401, Proceedings HPCN Europe 1998, (Springer-Verlag, Berlin, 1998) 368-376.

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