Block Incomplete LU-preconditioners for Implicit Solution of Advection Dominated Problems

Auke van der Ploeg[1], Rony Keppens[2], and Gábor Tóth[3]

[1] Centrum voor Wiskunde en Informatica, Postbus 94079, 1090 GB Amsterdam, The Netherlands
[2] FOM Institute for Plasma-Physics Rijnhuizen, Postbus 1207, 3430 BE Nieuwegein, The Netherlands
[3] Astronomical Institute, Postbus 80000, 3508 TA Utrecht, The Netherlands

Presenting author: Auke van der Ploeg

E-mail: Auke.van.der.Ploeg@cwi.nl
Tel.: +31 20 592 4115
Fax: +31 20 592 4199

Block Incomplete LU-preconditioners for Implicit Solution of Advection Dominated Problems

Auke van der Ploeg[1], Rony Keppens[2], and Gábor Tóth[3]

[1] Centrum voor Wiskunde en Informatica, Postbus 94079, 1090 GB Amsterdam, The Netherlands
[2] FOM Institute for Plasma-Physics Rijnhuizen, Postbus 1207, 3430 BE Nieuwegein, The Netherlands
[3] Astronomical Institute, Postbus 80000, 3508 TA Utrecht, The Netherlands

Abstract. In many astrophysical hydrodynamical and magnetohydrodynamical problems explicit time integration is inefficient due to the disparate time scales, or because we are looking for a steady-state. Implicit schemes can offer a viable alternative, which requires efficient solution of large linear systems. In multi-dimensional flows these linear systems cannot be solved with a direct method, so that iterative methods must be used. Since the physical problems are often advection dominated, the matrix is approximately anti-symmetric and non-diagonally dominant, thus preconditioning is necessary.
We describe a new approach, in which a GMRES type iterative method is combined with an Eisenstat implementation of a relaxed form of a modified Block Incomplete LU-decomposition as preconditioner. Parallelisation aspects of the algorithm are also discussed.

1 Introduction

In many astrophysical problems a conservative system of hyperbolic partial differential equations with additional non-hyperbolic source terms

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

has to be solved, where $\mathbf{W}$, $\mathbf{F}_i$ and $\mathbf{S}$ are the conservative variables, the fluxes and the source terms, respectively, and i represents the 1, 2 or 3 spatial coordinates. In hydrodynamics, for example, $\mathbf{W}$ consists of mass, momentum and total energy densities, in MHD the magnetic field components are the additional variables. The sources may include elliptic type viscous and resistive terms.

When we are interested in finding a steady-state solution to (1), the explicit integration schemes limited by the Courant-Friedrichs-Levy (CFL) condition may converge very slowly, whereas implicit schemes can use arbitrarily large time-steps. Another important application of implicit schemes arises in time-accurate calculations when the flow evolves much slower than as determined by the fastest wave of the hyperbolic equation or by a stiff and/or strongly elliptic source term.

For this reason, we implemented a general (semi-)implicit time integration scheme [8] in the Versatile Advection Code (VAC) [10, 11] which uses modern shock-capturing numerical schemes to solve equations of type (1). In keeping with the general outset of the code, we allow for both a pseudo-timestepping approach towards steady-state, and for time-accurate calculations. The semi-implicit scheme also provides the option to treat only some of the equations or only the source terms implicitly. In all cases the solution of large linear problems is required.

VAC uses structured grids, which is combined with a first-order spatial discretisation of the implicitly treated terms, but the solution itself can still be second-order accurate. In one dimension this leads to a block tridiagonal matrix, which can be solved efficiently in parallel with a direct block tridiagonal solver [15]. In 2D a block pentadiagonal matrix results, and in 3D a block heptadiagonal matrix. In practice, these linear systems cannot be solved directly, therefore iterative methods must be exploited. For elliptic type problems the iterative schemes (e.g. the (Bi-)Conjugate Gradient method) perform very well, but for advection dominated problems the matrix will be approximately anti-symmetric and non-diagonally dominant, thus suitable preconditioners are needed.

We find that a pointwise ILU-preconditioner does not work for the typical application shown in Sect. 4. Therefore, in this paper we use a new and efficient implementation of a Block Incomplete LU (BILU)-decomposition as a preconditioner for the restarted version of GMRES [9]. It will be shown that modification of the blocks on the main diagonal [7, 3] can improve the convergence behaviour.

Section 2 describes the time discretisation that results in the linear systems, in Sect. 3 the new preconditioner is introduced, and in Sect. 4 we demonstrate the performance of the method on a two-dimensional test problem. In the last section we draw some conclusions and discuss our plans for further developments.

2 Fully and Semi-Implicit Time Discretisation

We discretise the system of equations (1) using a one parameter (β) semi-implicit scheme

$$\frac{\mathbf{W}^{n+1}-\mathbf{W}^{n}}{\Delta t}=\mathbf{T}_{\mathrm{tot}}(\mathbf{W}^{n})+\beta\mathbf{T}_{\mathrm{impl}}(\mathbf{W}^{n+1})-\beta\mathbf{T}_{\mathrm{impl}}(\mathbf{W}^{n})\,, \tag{2}$$

where $\mathbf{T}_{\mathrm{tot}}$ is a spatially high (second-)order discretisation of $-\sum_i \partial_i \mathbf{F}_i + \mathbf{S}$, while $\mathbf{T}_{\mathrm{impl}}$ is a spatially low (first-)order discretisation of all the implicitly treated terms. For time-accurate solutions $\beta = 1/2$ can be used, while, when a steady-state is sought for, $\beta = 1$ is preferred.

We may linearise $\mathbf{T}_{\mathrm{impl}}$ around $\mathbf{W}^n$ to obtain

$$\left[\frac{1}{\Delta t}-\beta\frac{\partial\mathbf{T}_{\mathrm{impl}}}{\partial\mathbf{W}}\right]\left(\mathbf{W}^{n+1}-\mathbf{W}^{n}\right)=\mathbf{T}_{\mathrm{tot}}(\mathbf{W}^{n}). \tag{3}$$

In the limit where the time step $\Delta t \to \infty$, this is effectively an inexact Newton method for the steady-state solution.

We note that using a low-order spatial discretisation for $\mathbf{T}_{\text{impl}}$ has the advantage that the Jacobian matrix contains fewer elements, thus the linear system can be solved more efficiently. Still the solution $\mathbf{W}^{n+1}$ will be high-order accurate if we use a high-order discretisation for $\mathbf{T}_{\text{tot}}$. This holds both for time-accurate and steady-state solutions.

3 BILU-preconditioners

If we denote $(\mathbf{W}^{n+1} - \mathbf{W}^n)$ by x and $\mathbf{T}_{\text{tot}}(\mathbf{W}^n)$ by b we can write the system of linear equations (3) simply as

$$Ax = b \tag{4}$$

in which the vectors x and b are partitioned as

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ \vdots \\ b_N \end{bmatrix}. \tag{5}$$

Herein N denotes the number of grid points. Each subvector x_i, $i = 1, \ldots, N$, contains m unknowns corresponding to the i-th grid point. Because the system of linear equations comes from an advective problem, A is strongly non-symmetric and from the partitioning of the vectors it inherits a block structure corresponding to a five-point stencil in two dimensions and a seven-point stencil in three dimensions. Due to periodic boundary conditions there can be some subblocks that do not belong to this block structure.

In Fig. 1 we show a small characteristic part of the coefficient matrix A for a 2D hydrodynamical problem with $m = 4$ unknowns in each cell of a small 30×50 Cartesian grid. The dot sizes indicate the absolute values of the non-zero entries. For the test problem in Sect. 4, the block size is $m = 6$, and the grid size is 60×60, which makes the distance from the outer-most blocks to the blocks on the main diagonal much larger than for the matrix shown in Fig. 1.

One may attempt to solve (4) by constructing a complete LU-decomposition. Unfortunately the lower-triangular factor L and the upper-triangular factor U are not sparse due to fill-in during the factorisation process, thus computer storage demands are very high. Moreover, the computational work for constructing L and U increases strongly with N and m. We therefore want to solve (4) with an iterative method.

From numerical experiments it appears that we need a preconditioner in order to get convergence. In this paper, we use GMRES [9] as the iterative method, but the preconditioners to be discussed can be combined with other Conjugate Gradient-like methods as well. We focus on Incomplete LU-decompositions (ILU), i.e. we construct a splitting $A = LU - R$, where R is the residual matrix, which has to be small in some sense. This splitting is constructed in such

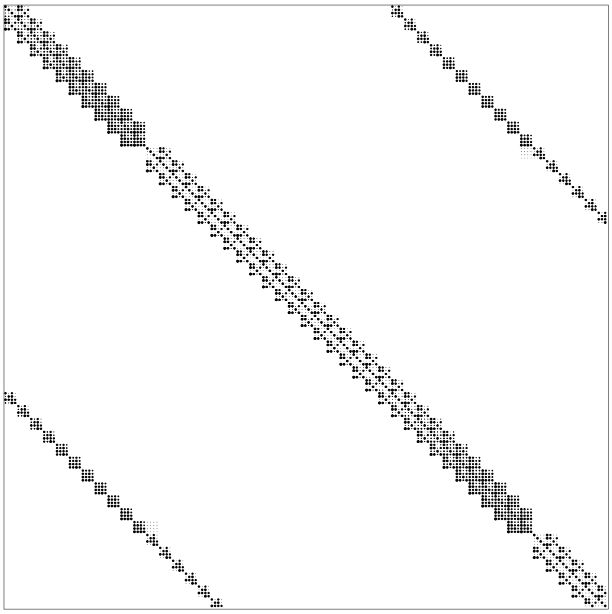

Fig. 1. Sparsity pattern of a characteristic part of A coming from a 2D hydrodynamical problem.

a way that L and U are sparse matrices, hence the action of L^{-1} and U^{-1} can be computed without too much computational effort. The iterative method is applied to the preconditioned system

$$L^{-1}AU^{-1}\tilde{x} = L^{-1}b \tag{6}$$

and we obtain x from $x = U^{-1}\tilde{x}$.

3.1 Pointwise versus Blockwise ILU-decompositions

First we have tried a pointwise ILU-decomposition in which the sparsity pattern of L and U is based on a threshold parameter [14]. In this approach, we scale the matrix A in such a way that

$$\sum_{j=1}^{Nm} |a_{ij}| = 1, \text{ for } 1 \leq i \leq Nm \tag{7}$$

and then a splitting $A = LU - R$ is constructed in which the entries of R satisfy $|r_{ij}| < \varepsilon$ for $1 \leq i, j \leq Nm$. Our numerical experiments showed that for our applications this approach is not successful. For small values of ε the factors L and U contain too many non-zero elements, and when ε increases, the number of GMRES iteration steps required to solve (6) increases strongly.

We therefore use a Block Incomplete LU (BILU-)decomposition, which exploits the block structure of the system matrix (see, e.g., [1]). In this approach, we again construct a splitting $A = LU - R$, but now the elimination process is performed on a block level. Hence we demand that the matrix $L + U$ also has a block structure, with subblocks having the same size as the subblocks in A. The block structure can be the same as that of A, or we can allow some additional fill-in blocks in $L + U$. During the construction and application of the BILU-preconditioner, we have to compute the action of the inverse of the subblocks on the main diagonal. This is done by constructing a *complete* LU-decomposition of these subblocks, which can be combined with partial pivoting. An advantage of this approach compared to pointwise ILU-decomposition without pivoting is that we can deal with diagonal entries that are zero. However, we still have the requirement that each subblock on the main diagonal has to be non-singular.

Note that this approach is different from the BILU-preconditioners described in [2], in which the block structure of the system matrix corresponds to a partitioning of the unknowns belonging to lines (or planes in the 3-dimensional case) of a Cartesian product grid. In this approach, the subblocks can be very large, and therefore *incomplete* LU-decompositions of the blocks on the main diagonal are used.

3.2 Eisenstat Implementation of BILU-preconditioners

We choose the block sparsity pattern of $L + U$ to be the same as that of A. For our applications, both in two and in three dimensions, this implies that during the construction of L and U only corrections on the main diagonal blocks are performed. For this kind of block LU-decompositions, computer storage demands are low, because the preconditioner only requires one extra block diagonal. Note that our approach is different from using a block Jacobi preconditioner, since this would only consist of scaling the blocks on the main diagonal of A to unity.

We will show that it is possible to implement the matrix-vector multiplication with the preconditioned matrix efficiently by using a block form of the so-called Eisenstat implementation [6]. Suppose that we construct L and U in such a way that the diagonal blocks of U are unity. Hence the preconditioner LU can be written as

$$(D + L_A)(I + U_A) \tag{8}$$

where L_A and U_A are the strictly lower- and upper-triangular part of A respectively, and D is a block diagonal matrix with the same diagonal blocks as L. Without loss of generality we may assume that D only has unity blocks on the main diagonal, because we can scale the original system with D. Note that in general this scaling does not imply that the diagonal blocks of A are all unity blocks. Suppose that after scaling the block diagonal of A is given by D_A. Now the preconditioned matrix $L^{-1}AU^{-1}$ can be represented by

$$\begin{aligned}
&(I + L_A)^{-1}(D_A + L_A + U_A)(I + U_A)^{-1} = \\
&(I + L_A)^{-1}(I + L_A + D_A - 2I + I + U_A)(I + U_A)^{-1} = \\
&(I + U_A)^{-1} + (I + L_A)^{-1}(D_A - 2I)(I + U_A)^{-1} + (I + L_A)^{-1}
\end{aligned}$$

which implies that $x = L^{-1}AU^{-1}y$ for given y can be computed as follows:

$$v := (I + U_A)^{-1}y;$$
$$x := (I + L_A)^{-1}[y + (D_A - 2I)v] + v.$$

This computation requires only solving two triangular systems plus the multiplication of the block diagonal matrix $(D_A - 2I)$ with v plus two vector additions. Hence the multiplication with the preconditioned matrix $L^{-1}AU^{-1}$ costs about the same number of flops as the multiplication with A.

3.3 Relaxed Modified BILU

In the so-called Modified Block Incomplete LU-decomposition (MBILU), which has been discussed by Gustafsson for pointwise ILU-decompositions [7], a splitting $A = LU - R$ is made in which fill-in blocks outside the sparsity pattern of L and U are added to the main diagonal blocks. As a result R has block row sums zero. Hence if the subblocks of R are denoted by R_{ij}, for $1 \leq i, j \leq N$, this yields

$$\sum_{j=1}^{N} R_{ij} = 0, \text{ for } 1 \leq i \leq N. \tag{9}$$

In many applications, especially for M-matrices, these modified incomplete LU-decompositions are known to be powerful for reducing the number of iteration steps. Unfortunately, the performance of such methods is sometimes spoiled by the largest eigenvalues of the preconditioned matrix. Therefore, we use a block form of the so-called relaxed modified incomplete decomposition [3]. The fill-in blocks that are not allowed in the factors L and U are first multiplied by a constant factor α, before they are added to the main diagonal blocks. This constant α is chosen in the interval [0,1]. For $\alpha = 0$ we obtain the standard BILU-decomposition without modification of the main diagonal blocks, and for $\alpha = 1$ the block row sum criterion (9) is satisfied exactly. A drawback of this approach is that the optimal value for α is not known in advance. For our applications, choosing α somewhere in the interval $[0.3, 0.7]$ appears to be a proper choice.

3.4 Implementation on Parallel Computers

Algorithms to solve the lower- and upper-triangular systems are highly sequential by nature. However, for our applications, both in two and three dimensions it is possible to improve the performance on parallel computers by changing the order of computation. Suppose that x is solved from $Lx = y$. The unknowns corresponding to the grid point (i, j) only depend on the previously computed unknowns corresponding to the grid points $(i - 1, j)$ and $(i, j - 1)$. Hence the unknowns corresponding to the grid points along a diagonal of the grid depend only on the unknowns corresponding to the previous diagonal. Therefore, if the unknowns are computed in the order corresponding to these diagonals, all components of x corresponding to the same diagonal can be computed independently. This technique is a special case of the more general technique known as level-scheduling (see, e.g., [4]).

Another way to obtain an efficient implementation on parallel computers is to use a block form of a red-black ordering, or of other reordering strategies. A drawback of such reordering techniques is that in some cases the number of iteration steps increases [5].

4 Test Case and Numerical Results

As an illustrative example we model super-Alfvénic flow around a perfectly conducting cylinder (see Fig. 2).

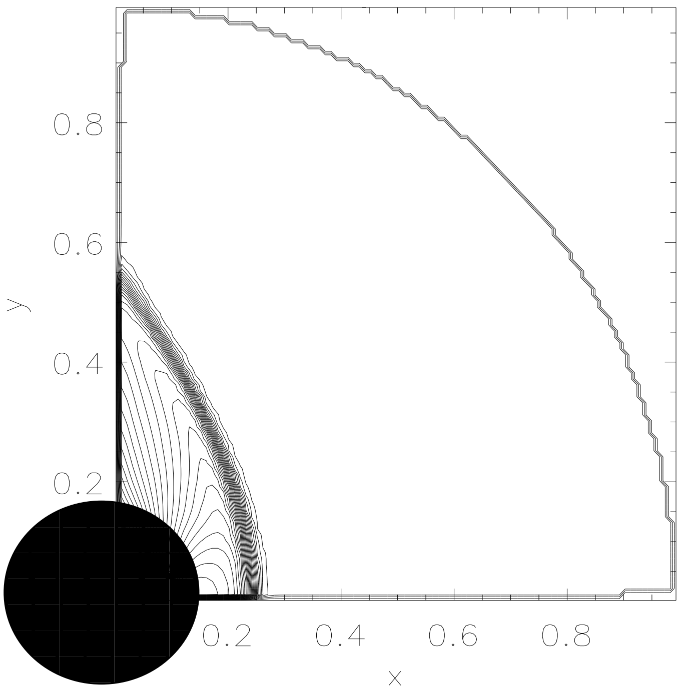

Fig. 2. Contourplot of the density for the steady-state solution. The black circle indicates the position of the perfectly conducting cylinder. The other boundaries are at $x = 0$, $y = 0$, and $r = 1$.

The 2D ideal MHD equations of $m = 6$ variables are solved on a 60×60 polar grid. The surface of the perfectly conducting cylinder is at $r = 0.125$, the outer boundary is at $r = 1$. Due to the symmetry of the initial condition, we may use a symmetric boundary condition at $\phi = 0$, thus the $0 < \phi < \pi/2$ quarter represents the $-\pi/2 < \phi < \pi/2$ half of the full polar grid. Initially both the velocity $v = 4$ and the magnetic field $B = 1$ are parallel to the $\phi = 0$ direction everywhere, and they point to the left. The density and thermal pressure are $\rho = 1$ and $p = 1$, and the adiabatic index is $\gamma = 5/3$, thus the Mach number with respect to the sound speed is approximately 3.1. The units for B are chosen such that the magnetic permeability is 1, thus the Alfvén speed is 1, and the Alfvénic Mach number is 4.

The system evolves to a steady-state as the supersonic flow hits the cylinder and a shock front forms around it. The initially imposed velocity field causes a rather violent transient in the first time steps as it hits the cylinder instantaneously. Therefore, we first integrate the system using an explicit scheme up to a point where the shock front has already taken shape, and then we switch to a fully implicit pseudo-timestepping scheme (3) to speed up the convergence towards the final steady-state.

We use a first-order Lax-Friedrich spatial discretisation to calculate the Jacobian $\partial\mathbf{T}_{\mathrm{impl}}/\partial\mathbf{W}$ in (3), while the right-hand side $\mathbf{T}_{\mathrm{tot}}$ is evaluated by a second-order total variation diminishing scheme (TVD Lax-Friedrich [13, 12]). We thus end up with a block pentadiagonal Jacobian matrix, with 3600 diagonal blocks of size 6. Hence the total number of unknowns is 21600.

Figure 3 shows the convergence behaviour of GMRES(20) combined with the relaxed form of the modified BILU-decomposition for different values of the parameter α. The relative 2-norm of the residual is plotted against the number of matrix-vector multiplications.

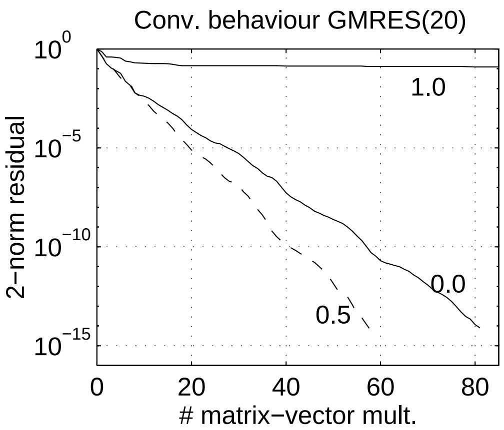

Fig. 3. Convergence behaviour of GMRES(20) combined with a relaxed, modified BILU-decomposition for different values of α.

It is clear that the choice $\alpha = 1$ which satisfies (9) exactly, is not a good one for our application. However, choosing α in the neighbourhood of 0.5 gives a better convergence behaviour of GMRES(20) than with unmodified BILU ($\alpha = 0$). The number of matrix-vector multiplications required to satisfy the stopping criterion

$$\|L^{-1}(b - AU^{-1}\tilde{x}^{(n)})\|_2 < \varepsilon \|L^{-1}(b - AU^{-1}\tilde{x}^{(0)})\|_2, \qquad \varepsilon = 10^{-14} \qquad (10)$$

is 81, 64, 58, and 60 for $\alpha = 0$, 0.4, 0.5, and 0.6, respectively. On a Silicon Graphics workstation (100 Mhz, MIPS R4010) the construction of the preconditioner takes 2.5 seconds, and for $\alpha = 0.5$ the iteration takes 20 seconds.

Note that for the actual simulation, we do not need a stopping criterion that is as strict as (10). For a solution which is accurate enough given the spatial resolution, $\varepsilon = 10^{-5}$ is sufficient, thus typically only 20 iterations are needed per time step. The steady-state solution of the same accuracy can be found in 47 (pseudo-) time steps (plus 100 initial explicit time steps) in contrast with the 3000 time steps required by the explicit time integration scheme. Of course, an implicit time step is more expensive than an explicit one, still the total speedup is approximately a factor of 8.

5 Conclusions and Plans for Future Developments

The systems of linear equations arising in VAC can be solved efficiently with an iterative method combined with a BILU-decomposition as preconditioner. The memory requirements for the preconditioner are low, since it only needs one extra block diagonal matrix. We have shown that, by using a block form of Eisenstat's trick, the matrix-vector multiplication with the preconditioned matrix $L^{-1}AU^{-1}$ costs about the same number of floating point operations as the matrix-vector multiplication with A. Our numerical results show that a relaxed form of the modified BILU-decomposition performs better than the unmodified BILU-decomposition.

In three dimensions, the matrices have, apart from some blocks coming from periodic boundary conditions, a block heptadiagonal structure. Both the number of grid points and the size of the subblocks will increase in three dimensions. We intend to use the same preconditioner as for the 2D case, so we can use the efficient Eisenstat implementation again.

For realistic modelling, especially in three dimensions, the coefficient matrices can become so large that it is not possible to store the whole matrix on one processor, and in that case we have to distribute the matrix and the vectors over several processors. Therefore, we intend to make a parallel implementation of the algorithm. As mentioned in Sect. 3.4, the incomplete decomposition described in this paper can be combined with a reordering technique for parallel processing.

For the test problem used in this paper, the number of GMRES iteration steps is not very high. However, we expect that the number of iteration steps increases with refinement of the mesh. Therefore, it is desirable to allow extra fill-in blocks in the incomplete LU-decomposition. This has to be done in such a way that we do not loose possibilities for an efficient implementation on parallel machines.

Acknowledgements

This work has been funded by the Dutch Science Foundation (NWO) in a program of Massive Parallel Processing Applied to Magnetohydrodynamics (95MPR04).

References

1. Aarden, J., Karlsson, K.-E.: Preconditioned cg-type methods for solving the coupled systems of fundamental semiconductor equations. BIT **29** (1989) 916–937
2. Axelsson, O., Gustafsson, I.: Iterative solution for the solution of the Navier equations of elasticity. Comput. Methods Appl. Mech. Eng. **15** (1978) 241–258
3. Axelsson, O., G. Lindskog, G.: On the eigenvalue distribution of a class of preconditioning methods. Numer. Math. 48 (1986) 479–498
4. Dongarra J. J., Duff I. S., Sorensen, D. C., van der Vorst, H. A.: Solving linear systems on vector and shared memory computers. (SIAM, 1991) 174–179
5. Duff, I. S., Meurant, G. A.: The effect of ordering on preconditioned conjugate gradient. BIT **29** (1989) 635–657
6. Eisenstat, S. C.: Efficient implementation of a class of preconditioned conjugate gradient methods. SIAM J. Sci. Statist. Comput. **2** (1981) 1–4
7. Gustafsson, I.: A class of 1:st order factorization methods. BIT **18** (1978) 142–156
8. Keppens, R., Tóth, G., van der Ploeg, A., Botchev, M.: Implicit and Semi-Implicit Schemes in the Versatile Advection Code. (work in progress)
9. Saad, Y., Schultz, M. H.: A generalized minimal residual algorithm for solving non-symmetric linear systems. SIAM J. Sci. Statist. Comput. **7** (1986) 856–869
10. Tóth, G.: A general code for modeling MHD flows on parallel computers: Versatile advection code. Astrophys. Lett. & Comm. **34** (1996), 245–250
11. Tóth, G.: Versatile Advection Code. (in this volume)
12. 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
13. Yee, H. C.: A class of high-resolution explicit and implicit shock-capturing methods. NASA TM-101088 (1989)
14. van der Ploeg, A.: Preconditioning for sparse matrices with applications. Ph.D. thesis, University of Groningen (1994)
15. van der Ploeg, A.: Reordering strategies and LU-decomposition of block tridiagonal matrices for parallel processing. Technical Report NM-R9618, Centrum voor Wiskunde en Informatica, October 1996