An algebraic multigrid method for high order time-discretization of the div-grad and the curl-curl equations

Tim Boonen
Jan Van lent
Stefan Vandewalle

Report TW 483, December 2006

Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

An algebraic multigrid method for high order time-discretization of the div-grad and the curl-curl equations

Tim Boonen
Jan Van lent
Stefan Vandewalle

Report TW 483, December 2006

Department of Computer Science, K.U.Leuven

Abstract

We present an algebraic multigrid algorithm for fully coupled implicit Runge-Kutta and Boundary Value Method time discretizations of the div-grad and curl-curl equations. The algorithm uses a blocksmoother and a multigrid hierarchy derived from the hierarchy built by any algebraic multigrid algorithm for the stationary version of the problem. By a theoretical analysis and numerical experiments, we show that the convergence is similar to or better than the convergence of the scalar algebraic multigrid algorithm on which it is based. The algorithm benefits from several possibilities for implementation optimization. This results in a computational complexity which, for a modest number of stages, scales almost linearly as a function of the number of variables.

Keywords : div-grad equation, curl-curl equation, implicit Runge-Kutta methods, boundary value methods, algebraic multigrid
AMS(MOS) Classification : 65M55,65L06

An algebraic multigrid method for high order time-discretization of the div-grad and the curl-curl equations

Tim Boonen[*] Jan Van lent[†] Stefan Vandewalle[*]

Abstract

We present an algebraic multigrid algorithm for fully coupled implicit Runge-Kutta and Boundary Value Method time discretizations of the div-grad and curl-curl equations. The algorithm uses a blocksmoother and a multigrid hierarchy derived from the hierarchy built by any algebraic multigrid algorithm for the stationary version of the problem. By a theoretical analysis and numerical experiments, we show that the convergence is similar to or better than the convergence of the scalar algebraic multigrid algorithm on which it is based. The algorithm benefits from several possibilities for implementation optimization. This results in a computational complexity which, for a modest number of stages, scales almost linearly as a function of the number of variables.

1 Introduction

Implicit Runge-Kutta (IRK) and Boundary Value Method (BVM) time discretizations have many appealing properties, such as high order of accuracy and good stability [7, 9, 10, 6]. However, when applied to time-dependent partial differential equations, they give rise to linear systems which are difficult to solve. First of all, the size of these systems is a multiple of the size of the systems to be solved for the backward Euler (BE) discretization of the same problem. Also, the efficiency of existing direct and iterative solvers for BE discretizations usually significantly deteriorates when applied to IRK or BVM discretizations.

In the case of IRK methods, the computational cost can be reduced significantly by using *diagonally implicit* (DIRK) Runge-Kutta methods. The resulting system is then block-triangular, and can be solved as a sequence of smaller problems. These smaller problems have a similar structure as the one to be solved in the BE method, hence efficient solvers for that method can be reused. Unfortunately, DIRK methods have a lower order of accuracy and poorer stability properties than fully coupled IRK methods of the same size [7, 10]. This order and stability reduction can be avoided by using a DIRK method as a *preconditioner* for the fully coupled IRK system [15]. The algorithm developed in [15] allows the reuse of existing BE software components.

[*]Department of Computer Science, Katholieke Universiteit Leuven,small Celestijnenlaan 200A, B-3001 Leuven, Belgium; email: {Tim.Boonen,Stefan.Vandewalle}@cs.kuleuven.be. Tim Boonen is research assistant of the Fund for Scientifi c Research - Flanders (Belgium) (F.W.O.-Vlaanderen).

[†]Department of Mathematical Sciences, University of Bath, Bath BA2 7AY, UK; email: j.van.lent@maths.bath.ac.uk

Unfortunately, the number of iterations required for solving the preconditioned IRK-system increases rapidly with the number of stages in the IRK method.

In [20], it is shown how geometric multigrid techniques can be used to solve efficiently *fully* coupled IRK and BVM time discretizations of finite difference spatial discretizations of time-dependent parabolic problems

$$-\nabla \cdot (\alpha \cdot \nabla V) + \beta \frac{\partial V}{\partial t} = \rho, \qquad \alpha > 0,\ \beta \geq 0. \tag{1}$$

The key is the use of a blocksmoother updating all unknowns related to a grid point simultaneously. In this paper, the approach of [20] will be extended to *algebraic* multigrid (AMG) for (1) and for the curl-curl equation

$$\vec{\nabla} \times (\alpha \vec{\nabla} \times \vec{A}) + \beta \frac{\partial \vec{A}}{\partial t} = \vec{J}, \qquad \alpha > 0,\ \beta \geq 0. \tag{2}$$

Equation (2) arises in time-domain approaches for solving electromagnetic problems, such as the eddy current problem [16, 4].

A theoretical estimate of the complexity of the AMG algorithm developed in this paper would lead one to conclude that the number of floating point operations and the memory requirements scale cubically and quadratically respectively as a function of the number of stages in the IRK or BVM method. It will be shown, however, that the algorithm offers several opportunities for implementation optimization, yielding a significant increase of the floating point execution rate and a significant reduction of the memory requirements. The timing results of the optimized implementation will be shown to scale rather linearly as a function of the number of stages. Thanks to the favorable timings and memory requirements and to the high order of accuracy of the considered time discretization methods, the presented AMG algorithm is very useful in practice.

This paper is organized as follows. In Section 2, the discretizations in time and space of (1) and (2) used in this paper, are explained. In Section 3, the block AMG algorithm is discussed, and in Section 4 a convergence analysis is presented. Section 5 deals with important implementation issues. In Section 6, some numerical results are presented that illustrate the efficiency of the method.

2 Spatial and temporal discretization

For (1), we consider standard lowest order spatial discretizations using finite differences or finite elements conforming in $H_1(\Omega)$. For (2), we consider spatial discretizations using the finite integration technique (FIT) [8] or using lowest order edge finite elements conforming in $H(curl;\Omega)$ [11]. In that case, the degrees of freedom (DoF) are associated with the edges of the mesh. For both problem types, the considered spatial discretizations give rise to the following set of equations:

$$Ku + M\frac{du}{dt} = h. \tag{3}$$

The system size, denoted as N, equals the number of nodes or edges for the div-grad and the curl-curl problem respectively. The symmetric and positive definite matrices $K \in \mathbb{R}^{N\times N}$ and $M \in \mathbb{R}^{N\times N}$ denote the stiffness matrix and the mass matrix. The vector $h \in \mathbb{R}^N$ is the vector of excitations.

Several efficient AMG algorithms exist for solving the system arising from a BE discretization of (3). For the case of the div-grad equation, we refer to [18, 19, 17, 5],

and for the case of the curl-curl equation to [13, 1, 2]. In this paper, IRK and BVM time discretization (see [7, 9, 10, 6]) will be used. For the reader's convenience, and in order to introduce some essential notation, we shall recall some of the basic formulas and properties of these methods.

Consider the system of N ODEs

$$\frac{du}{dt} = f(t,u), \qquad \text{with } u(t_0) = u_0 \in \mathbb{R}^N. \tag{4}$$

An **implicit Runge-Kutta (IRK)** method (see [7, 9, 10]) approximates the solution $u(t_0+\Delta t)$ by u_1, computed from

$$u_1 \quad = \quad u_0 + \Delta t \sum_{j=1}^{s} b_j f(t_0 + c_j \Delta t, x_j). \tag{5}$$

Here, s denotes the number of stages. The s vectors $x_i \in \mathbb{R}^N$ satisfy the following system of equations

$$x_i \quad = \quad u_0 + \Delta t \sum_{j=1}^{s} a_{ij} f(t_0 + c_j \Delta t, x_j), \quad i = 1 \ldots s. \tag{6}$$

With $A_{irk} = [a_{ij}] \in \mathbb{R}^{N\times N}$, $b_{irk} = [b_1 \ \ldots \ b_s]^T \in \mathbb{R}^N$ and $f_j = f(t_0 + c_j\Delta t, x_j) \in \mathbb{R}^N$, and using the multivectors $X = [x_1 \ \ldots \ x_s] \in \mathbb{R}^{N\times s}$, $U_0 = [u_0 \ \ldots \ u_0] \in \mathbb{R}^{N\times s}$ and $F = [f_1 \ \ldots \ f_s] \in \mathbb{R}^{N\times s}$, (5) and (6) can be rewritten compactly as

$$u_1 \quad = \quad u_0 + \Delta t F b_{irk} \tag{7}$$

$$X \quad = \quad U_0 + \Delta t F A_{irk}^T. \tag{8}$$

Boundary value methods (BVM) can be considered a generalization of linear multi-step methods (LMM) (see [6]). In each step of a BVM, the solution values at a series of n points in time $t_1, t_2, \ldots, t_n$ are approximated. Here, for simplicity of notation, we shall only consider the equidistant case, where $t_j = t_0 + j\Delta t$. A k-step BVM for (4) has for each $i = k_1, \ \ldots, \ n-k_2$, with $k_1 + k_2 = k$, an equation of the form

$$\sum_{j=-k_1}^{k_2} \alpha_{k_1+j} u_{i+j} = \Delta t \sum_{j=-k_1}^{k_2} \beta_{k_1+j} f(t_{i+j}, y_{i+j}). \tag{9}$$

The main difference with traditional LMM time discretization is the use of *future* points in time. As such, a BVM discretization cannot be solved in a time-marching way. An additional $k_1 - 1$ initial and k_2 final equations are needed of the form

$$\begin{array}{rcll} \sum_{j=0}^{k} \alpha_j^{(i)} u_j & = & \Delta t \sum_{i=0}^{k} \beta_j^{(i)} f(t_j, u_j) & i = 1, \ \ldots, \ k_1 - 1 \\ \sum_{j=0}^{k} \alpha_j^{(i)} u_{n-k+j} & = & \Delta t \sum_{j=0}^{k} \beta_j^{(i)} f(t_{n-k+j}, u_{n-k+j}) & i = n-k_2+1, \ \ldots, \ n. \end{array} \tag{10}$$

The coefficients $\alpha_j^{(i)}$ and $\beta_j^{(i)}$ are chosen such that the truncation errors are of the same order as for the basic method (9). When applied to (3), the BVM problem reads in compact format:

$$[u_0 \ U] \ [a_0 \ A_{bvm}]^T = \Delta t [f_0 \ F] \ [b_0 \ B_{bvm}]^T. \tag{11}$$

Here, u_0 and f_0 represent the initial condition and the corresponding right-hand side value at time t_0. The matrices $A_{bvm},\ B_{bvm} \in \mathbb{R}^{n\times n}$ have a quasi-Toeplitz structure. For instance, $[a_0\ A_{bvm}]$ is given by

$$[a_0\ A_{bvm}] \quad = \quad \left[\begin{array}{c|ccccccc} \alpha_0^1 & \alpha_1^1 & \dots & \alpha_k^1 & & & & \\ \vdots & & & \vdots & & & & \\ \alpha_0^{k_1-1} & \alpha_1^{k_1-1} & \dots & \alpha_k^{k_1-1} & & & & \\ & \alpha_0 & & \dots & \alpha_k & & & \\ & & \ddots & & & \ddots & & \\ & & & \alpha_0 & & \dots & \alpha_k \\ & & & \alpha_0^{n-k_2+1} & & \dots & \alpha_k^{n-k_2+1} \\ & & & \vdots & & & \vdots \\ & & & \alpha_0^n & & \dots & \alpha_k^n \end{array}\right] \tag{12}$$

Assuming that A_{bvm} is invertible, (11) can be transformed into a system of the form (8):

$$U \quad = \quad \left(-u_0 a_0^T + \Delta t\, f_0 b_0^T\right) A_{bvm}^{-T} + \Delta t F B_{bvm}^T A_{bvm}^{-T} \quad := \quad \tilde{U}_0 + \Delta t F \tilde{A}^T. \tag{13}$$

The computational cost for IRK and BVM methods is dominated by the solution of (8) and (13) respectively. Thanks to the formal equivalence of these systems, the analysis in this paper can be restricted to IRK problems and the results will carry over automatically to BVM problems when A_{irk} is replaced by $A_{bvm}^{-1}B_{bvm}/n$. Note that the scaling factor $1/n$ takes into account the specific meaning of the time step Δt. For IRK methods, it corresponds to the *global* time step; for BVM methods, it is the time step *between the intermediate steps*.

We will now derive three formulations for the linear system that appears in every time-step when an IRK method is applied to the semi-discretized PDE. The application of (6) to (3) yields a system of the following form, with I_s the unit matrix of size s:

$$\left[\begin{array}{ccc} M+\Delta t K a_{11} & \dots & \Delta t K a_{1s} \\ \vdots & & \vdots \\ \Delta t K a_{s1} & \dots & M+\Delta t K a_{ss} \end{array}\right] \left[\begin{array}{c} \tilde{x}_1 \\ \vdots \\ \tilde{x}_s \end{array}\right] \quad = \quad \tilde{b}. \tag{14}$$

With $\otimes$ the Kronecker matrix product symbol, (14) can be written in tensor form:

$$(I_s \otimes M + \Delta t A_{irk} \otimes K)\,\tilde{x} \quad = \quad \tilde{b}. \tag{15}$$

Here, the unknowns are ordered block-wise; the vectors $\tilde{x}_i \in \mathbb{R}^N$ correspond to the unknowns of the ith IRK stage. If A_{irk} is triangular, as for DIRK methods, (14) can be solved efficiently with block back-substitution using solvers optimized for BE problems. This is because the diagonal blocks of (14) are formally equivalent to BE discretizations with a scaled time step $\Delta\tilde{t} = a_{ii}\Delta t$. In *fully* coupled IRK discretizations, however, it is not possible any more to solve for the $\tilde{x}_i$ values sequentially.

Reordering the unknowns in (14) per node for div-grad problems and per edge for curl-curl problems yields a system of the following form:

$$\left[\begin{array}{ccc} m_{11}I_s + \Delta t k_{11} A_{irk} & \cdots & m_{1N}I_s + \Delta t k_{1N} A_{irk} \\ \vdots & \ddots & \vdots \\ m_{N1}I_s + \Delta t k_{N1} A_{irk} & \cdots & m_{NN}I_s + \Delta t k_{NN} A_{irk} \end{array}\right] \left[\begin{array}{c} x_1 \\ \vdots \\ x_N \end{array}\right] \quad = \quad b, \tag{16}$$

which is equivalent to

$$(M \otimes I_s + \Delta t K \otimes A_{irk})\, x \quad = \quad b. \tag{17}$$

Here, $x_i \in \mathbb{R}^s$ denotes the vector of the stage values related to the node respectively edge i. Formulation (16) will be used to explain the block AMG method presented in this paper. The third formulation, which will prove useful for implementation purposes, stems from the application of (8) to (3). This yields:

$$MX + \Delta t K X A_{irk}^T \quad = \quad B. \tag{18}$$

Note that (15), (17) and (18) are mathematically equivalent. Once these systems have been solved, (5) or (7) can be used to propagate the solution values from t_0 to t_1.

3 Block AMG algorithm

In this section, an algebraic multigrid algorithm is developed for problems of the form (15)-(17)-(18), denoted in short as $Lx = b$. A standard multigrid cycle is considered, which is the recursive application of the following 2-level scheme:

1. smoothing on $Lx = b$
2. restriction: $b_c = R_{irk}(b - Lx)$
3. (approximate) solve of $L_c x_c = b_c$
4. prolongation: $x \leftarrow x + P_{irk} x_c$
5. smoothing on $Lx = b$

Here, the subscript c refers to the coarse level; the matrices P_{irk} and R_{irk} are the prolongation and restriction matrices respectively. In order to characterize the algorithm, a procedure to build P_{irk} and R_{irk} and the coarse system matrices K_c is needed. Also, the smoothing operation is to be specified. For the solution of the system at the coarsest level of the multigrid hierarchy, a direct solver or a preconditioned Krylov solver can be used.

3.1 Prolongation and restriction

We suggest to derive the prolongation matrix P_{irk} from the prolongation matrix P built for the stationary problem $Kx = b$, where K is the stiffness matrix from (3). The matrix P can be built using any of the classical AMG heuristics, for instance [18, 19, 17, 5] for the div-grad problem and [13, 1, 2] for the curl-curl problem. More precisely, for formulation (17), the following construction is suggested for P_{irk}:

$$P_{irk} \quad = \quad P \otimes I_s. \tag{19}$$

Note that this amounts to stage decoupling for the prolongation operator. Following the standard AMG practice, we set the restriction operator equal to the prolongator's transpose, that is

$$R_{irk} \quad := \quad P_{irk}^T = P^T \otimes I_s. \tag{20}$$

Using the compact formulation of (18), the restriction of the residual and the prolongation of the coarse grid correction read:

$$\begin{aligned} B_c &= P^T\left(B - MX - \Delta t K X A_{irk}^T\right) && (21)\\ X &\leftarrow X + PX_c && (22)\end{aligned}$$

3.2 Construction of the coarse system

As for any AMG method, the coarse system matrices are built as the Galerkin product of the fine system matrices and the prolongation matrices. For the formulation used in (17), we have

$$\begin{aligned} L_c &= P_{irk}^T\left(M \otimes I_s + \Delta t K \otimes A_{irk}\right) P_{irk} \\ &= \left(P^T M P\right) \otimes I_s + \Delta t \left(P^T K P\right) \otimes A_{irk}. && (23)\end{aligned}$$

Using the compact format of (18), the coarse system reads:

$$\left(P^T M P\right) X_c + \Delta t \left(P^T K P\right) X_c A_{irk}^T = B_c. \qquad (24)$$

Note that (24) corresponds to the classical IRK time-discretization of the coarse grid equivalent problem to (3), i.e.

$$K_c u + M_c \frac{du}{dt} = h_c, \qquad \text{with } K_c = P^T K P \text{ and } M_c = P^T M P.$$

3.3 Definition of the smoother

For the div-grad equation (1), the blocksmoother suggested in [20] for use with a geometric multigrid method, will be used. This smoother consists of a sequence of local solves, one corresponding to each node i, in which all stage values for that node are updated simultaneously:

$$\left(m_{ii} I_s + \Delta t\ k_{ii} A_{irk}\right) x_i = b_i - \sum_{j \neq i} \left(m_{ij} I_s + \Delta t\ k_{ij} A_{irk}\right) x_j. \qquad (25)$$

This blocksmoother is a block Gauss-Seidel type matrix splitting iteration applied to (16). Using the matrix splittings $M = M^+ + M^-$ and $K = K^+ + K^-$, with M^+ and K^+ denoting the lower triangular parts of M and K, (25) can be formulated in tensor notation as:

$$\left(M^+ \otimes I_s + \Delta t K^+ \otimes A_{irk}\right) x^{[\nu+1]} = b - \left(M^- \otimes I_s + \Delta t K^- \otimes A_{irk}\right) x^{[\nu]}. \qquad (26)$$

Here, superscript $[\nu]$ is used to denote the iteration index. The nonzero structure of the lefthand side matrix is indicated in Figure 1. Note that the same matrix splittings, applied to a backward Euler discretization of (3), correspond to a classical Gauss-Seidel iteration for the problem $(M + \Delta t K)x = b$,

$$\left(M^+ + \Delta t K^+\right) x^{[\nu+1]} = b - \left(M^- + \Delta t K^-\right) x^{[\nu]}. \qquad (27)$$

In multigrid for the curl-curl equation, more involved smoothers are needed. This is because the smoothing properties of classical smoothers, such as Gauss-Seidel, are insufficient in the gradient part of the discrete analogon of the Helmholtz splitting (see

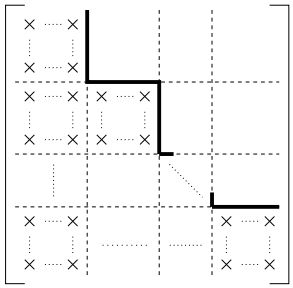

Figure 1: Nonzero structure of $M^+ \otimes I_s + \Delta t K^+ \otimes A_{irk}$.

[16]). The latter reads, with f denoting the number of mesh faces, and with $G \in \mathbb{R}^{e \times n}$ and $C \in \mathbb{R}^{f \times e}$ respectively the discrete gradient and curl operators

$$\mathbb{R}^e = ran(G) \oplus ran(C^T). \tag{28}$$

We refer the reader to [12] for an analysis of this phenomenon. In this paper, the so-called hybrid smoother, presented in [12], will be used. This smoother performs an additional smoothing step in $ran(G)$ on an auxiliary system. The system matrix L_{aux} of this auxiliary system is usually constructed as the Galerkin product of the original system matrix with the gradient matrix G.

For our purposes, a block version of the hybrid smoother is needed. The auxiliary system matrix is constructed as the Galerkin product of the original system matrix with the decoupled tensor version $G_{irk} = G \otimes I_s$ of the gradient:

$$\begin{aligned} L_{aux} &= G_{irk}^T \left(M \otimes I_s + \Delta t K \otimes A_{irk}\right) G_{irk} \\ &= \left(G^T M G\right) \otimes I_s + \Delta t \left(G^T K G\right) \otimes A_{irk}. \end{aligned} \tag{29}$$

Using the compact format of (18), the auxiliary system reads:

$$\left(G^T M G\right) X_{aux} + \Delta t \left(G^T K G\right) X_{aux} A_{irk}^T = B_{aux}. \tag{30}$$

Note that $G^T K G$ is zero away from the Dirichlet boundaries. One step of the block hybrid smoother is defined as follows:

1. smoothing on (18)
2. block restriction: $B_{aux} = G^T (B - MX - \Delta t K X A_{irk}^T)$
3. smoothing on (30)
4. block prolongation: $X \leftarrow X + G X_{aux}$
5. smoothing on (18)

Formally, this is equivalent to a 2-level multigrid cycle with prolongator G_{irk} and coarse system matrix (29). The smoother in steps (1), (3) and (5) is the block smoother of [20].

4 Convergence analysis

4.1 A formula for the asymptotic convergence factor

The convergence analysis for the AMG method presented in this paper is similar to the convergence analysis for the geometric multigrid method in [20]. First, we analyze the blocksmoother. To this end, we consider the matrix function

$$S(z) \quad = \quad -(zK^+ + M^+)^{-1}(zK^- + M^-). \tag{31}$$

Note that $S(\Delta t)$ is the iteration matrix for the Gauss-Seidel iteration defined by (27) for the BE discretization of (3) with a time step Δt. The iteration matrix for the blocksmoother (26) for the IRK discretization of the same problem can be written in terms of the matrix function (31) as well. It is given by $S(\Delta A_{irk})$, which equals

$$-\left(\Delta t A_{irk} \otimes K^+ + I_s \otimes M^+\right)^{-1} \left(\Delta t A_{irk} \otimes K^- + I_s \otimes M^-\right). \tag{32}$$

Using the eigenvalue decomposition $A_{irk} = V\Lambda V^{-1}$ and after applying the Butcher transformation, which is the similarity transformation corresponding to $I_N \otimes V$ (see [7]), (32) can be shown to be equivalent to

$$-\left(\Delta t \Lambda \otimes K^+ + I_s \otimes M^+\right)^{-1} \left(\Delta t \Lambda \otimes K^- + I_s \otimes M^-\right).$$

This shows that (32) is spectrally equivalent to the following block diagonal matrix:

$$S(\Delta t A_{irk}) \sim \begin{bmatrix} S(\Delta t \lambda_1) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & S(\Delta t \lambda_s) \end{bmatrix}, \quad \lambda_i \in \sigma(A_{irk}).$$

Here, $\sigma(A_{irk})$ denotes the spectrum of A_{irk}, which is complex for most fully coupled IRK and BVM methods. So, the asymptotic convergence behavior of the blocksmoother (26) for the IRK discretization can be analyzed using the convergence results for the corresponding Gauss-Seidel smoother (27) for the BE discretization of the same problem. With $\rho(A)$ denoting the spectral radius of A, the asymptotic convergence factor for the blocksmoother is given by:

$$\rho\left(S(\Delta t A_{irk})\right) = \max_{\lambda \in \sigma(A_{irk})} \rho\left(S(\Delta t \lambda)\right). \tag{33}$$

A similar analysis can be done for a two level multigrid cycle using the matrix function

$$T(z) = S(z)^{\nu_2} \left(I_e - P\left(P^T\left(zK + M\right)P\right)^{-1} P^T\left(zK + M\right)\right) S(z)^{\nu_1},$$

where ν_1 and ν_2 are the number of pre- and post-smoothing iterations. Here, the invariance of the prolongation matrix P_{irk} for the similarity transformation $I_N \otimes V$ is used:

$$\left(I_N \otimes V^{-1}\right) \cdot \left(P \otimes I_s\right) \left(I_N \otimes V\right) = \left(I_N \cdot P \cdot I_N\right) \otimes \left(V^{-1} \cdot I_s \cdot V\right) = P \otimes I_s.$$

The results are analogous to (33), i.e.

$$\rho\left(T(\Delta t A_{irk})\right) \quad = \quad \max_{\lambda \in \sigma(A_{irk})} \rho\left(T(\Delta t \lambda)\right). \tag{34}$$

By recursion, the results for the two level cycle immediately carry over to a multilevel cycle. The analysis for the block hybrid smoother is formally identical to the analysis for the two level cycle, with P replaced by G.

To summarize, the asymptotic convergence of the presented IRK AMG algorithm is related to the asymptotic convergence behavior of the corresponding AMG algorithm applied to the BE discretization of the same problem, but with a scaled time step. That is, the asymptotic multigrid convergence factor for the fully coupled IRK system (16) corresponds to the worst case of the asymptotic multigrid convergence for the problems

$$(\lambda_i \Delta t K + M)\, x = b, \quad \text{with } \lambda_i \in \sigma(A_{irk}). \tag{35}$$

The latter can be analyzed quantitatively by a so-called local Fourier mode analysis [18, 21, 14, 3]. Note that (33) and (34) continue to hold for BVM time discretizations if A_{irk} is replaced by $A_{bvm}^{-1} B_{bvm}/n$, as mentioned in Section 2.

4.2 Discussion

The convergence speed of multigrid for the BE problem $(\Delta t K + M)x = b$ depends on the value of Δt. Hence, in order to be able to interpret (35), knowledge of the location of the eigenvalues λ_i is of great importance. The location in the complex plane of the eigenvalues of A_{irk} and $A_{bvm}^{-1} B_{bvm}/n$ for some popular IRK and BVM methods is indicated in Figure 2. The picture is very similar for the classical IRK families of type RadauIA, RadauIIA, Gauss and LobattoIIIC and for the BVM family GAM. For the IRK families LobattoIIIA and LobattoIIIB, A_{irk} is singular and one eigenvalue is 0. For the BVM family GBDF with more then 5 intermediate steps, some eigenvalues have a (small) negative real part. Typically, the modulus of the eigenvalues decreases with increasing number of stages for IRK or intermediate steps for BVM, and the decrease becomes smaller with increasing number of stages or steps.

For a model problem on a regular grid, the asymptotic convergence factor of a scalar multigrid algorithm applied to $(zK + M)x = b$ for $z \in \mathbb{C}$ can be calculated by a so-called local Fourier mode analysis. We refer to [18, 21, 14] for the div-grad problem and to [3] for the curl-curl problem. Qualitatively, the results of the analysis for the model problem carry over to a more general problem setting. By using the Fourier analysis of [14, 3], we have computed $\rho(T(z))$. The results are depicted in Figure 3. The local Fourier mode analysis shows that typically, the asymptotic convergence factor improves with decreasing real part of z. Hence, taking into account the typical location of the eigenvalues of A_{irk}, the convergence analysis presented in this section allows to predict a faster multigrid convergence for IRK methods with more stages or BVM methods with more intermediate steps. Also, as expected, the use of a smaller time-step will typically lead to a faster multigrid convergence.

5 Implementation issues

The AMG algorithm presented in this paper offers several opportunities for optimization, which will yield a significant increase of the floating point execution rate. This is important, because the dense linear system to be solved in the basic operation (25) of the blocksmoother causes the number of floating point operations to scale *cubically* as a function of the number of stages s. In this section, several aspects of the implementation, which are crucial for its memory and CPU efficiency, will be explained.

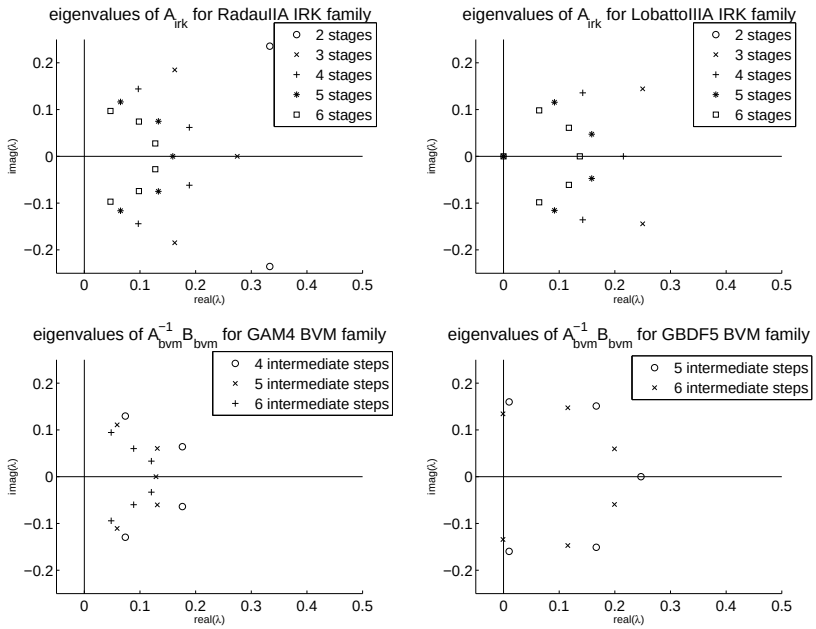

Figure 2: Eigenvalues of A_{irk} for the RadauIIA and LobattoIIIA IRK families and of $A_{bvm}^{-1}B_{bvm}/n$ for the GAM4 and GBDF5 BVM families. The 1-stage variant of the RadauIIA method corresponds to backward Euler; its eigenvalue is located at $(1,0)$. For the BVM methods, the smallest possible number of intermediate steps n equals the number of steps k of the LMMs it is composed of, which is 4 and 5 for the GAM4 and GBDF5 methods respectively.

5.1 Storage of the multivectors

A row-by-row storage of the multivectors offers opportunities for cache optimization for the product $Y = AX$ of a sparse matrix $A \in \mathbb{R}^{N\times N}$ and a multivector $X \in \mathbb{R}^{N\times s}$. In this case, optimal reuse of the cache is achieved if the inner of the three loops for the calculation of $Y = AX$ runs over the columns of the multivectors instead of over their rows:

```
FOR all rows r of Y,
    FOR all nonzeros locations i of A(r,:),
        FOR all columns c of Y,
            Y(r,c) = Y(r,c) + A(r,i)*X(i,c)
```

In this way, the data access patterns for X and Y match their storage layout. Note that in a classical column-by-column calculation, this loop is the *outer* loop. This procedure can be applied directly for the matrix-vector product (18), the block restriction (21) and the block prolongation (22). A similar cache optimization can be done for the blocksmoother in the calculation of the righthand sides of (25).

Note that a similar cache optimization is not possible if the multivectors are stored column-by-column. This is because the loop over the rows r of the multivectors cannot be moved to become the inner loop, as the values of i depend on r for sparse matrices.

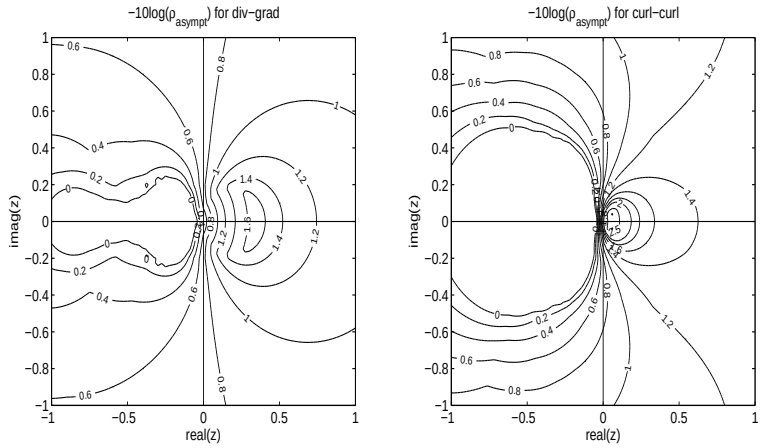

Figure 3: Asymptotic convergence factors of a V(1,1) multigrid cycle for the finite element discretization $(zK+M)x=b$ of the 2D div-grad and curl-curl equations on a quadrilateral grid as a function of z, calculated using local Fourier analysis.

5.2 Storage of the tensor system matrices

The compact format of (18) shows that *only* Δt, A_{irk} and the appropriate stiffness and mass matrices K and M need to be stored to represent the tensor system matrices. Correspondingly, the memory requirements are essentially independent of the number of stages and amount to $nnz(K)+nnz(M)$ matrix entries, with $nnz(X)$ denoting the number of nonzero elements of X. Obviously, the system matrices are never expanded to standard matrix form, as this would result in memory requirements scaling quadratically as $nnz(K+M)s^2$ as a function of the number of stages. Note that the compact format allows to change the time discretization scheme, which is represented by A_{irk}, without the need to rebuild the system matrix.

In finite element applications, Dirichlet boundary conditions are often introduced by eliminating the columns of Dirichlet unknowns and by replacing their equations by $x_i=v_i$, with v_i the required solution value for the unknown x_i. With d and $\bar{d}$ denoting the set of the indices of the Dirichlet unknowns and the non-Dirichlet unknowns respectively, this is achieved as follows in the compact format:

$$\begin{bmatrix} I & 0 \\ 0 & M_{\bar{d}\bar{d}} \end{bmatrix} \begin{bmatrix} X_d \\ X_{\bar{d}} \end{bmatrix} + \Delta t \begin{bmatrix} 0 & 0 \\ 0 & K_{\bar{d}\bar{d}} \end{bmatrix} \begin{bmatrix} X_d \\ X_{\bar{d}} \end{bmatrix} A_{irk}^T = \begin{bmatrix} V_d \\ B_{\bar{d}} - M_{\bar{d}\bar{d}} V_d - \Delta t K_{\bar{d}\bar{d}} V_d A_{irk}^T \end{bmatrix}.$$

5.3 Blocksmoother

The basic operation (25) of the blocksmoother consists of three steps. First, the local righthand side $b=b_i-\sum_{j\neq i}(m_{ij}I_s+\Delta t k_{ij}A_{irk})$ and the local system matrix $A=m_{ii}I_s+\Delta t k_{ii}A_{irk}$ are constructed. Next, the local $s\times s$ system denoted as $Ax=b$ is solved. In this procedure, several optimizations are possible to reduce the computational and memory complexity and to increase the floating point execution rate.

All local matrices A can be inverted and stored in a setup phase. This allows $Ax=b$ to be implemented as a matrix-vector product $x=A^{-1}b$, which causes the number of floating point operations to scale only quadratically, instead of cubically, as a function

of the number of stages. The price to be paid is an additional memory requirement, scaling quadratically as a function of the number of stages. This additional memory requirement can be reduced by a factor 2 by storing all A^{-1} in *single* precision.

In this case, the accuracy of the blocksmoother and, consequently, of the AMG cycle, is limited to single precision as well. This can be avoided by calculating the *increments* of the stage values instead of the stage values themselves:

$$A\Delta x_i = \left(b_i - \sum_j (m_{ij} I_s + \Delta t\ k_{ij} A_{irk})\, x_j \right) \tag{36}$$

$$x_i \leftarrow x_i + \Delta x_i. \tag{37}$$

For this purpose, x and b are to be stored in double precision and the calculation of the righthand side of (36) and the update formula (37) are to be carried out using double precision. Now, the precision of x_i is limited by the lowest possible *exponent* of the single precision floating point format instead of by the length of the single precision *mantissa*, which allows to achieve double precision. Note that this precision reduction does not occur if Krylov acceleration is used, since in this case, the preconditioner is applied to the error equation with zero initial guess, which is equivalent to (36)-(37).

For the dense system solves and the dense matrix-vector products used to solve (25) or (36), it is crucial *not* to use standard software components, such as LAPACK or BLAS routines. This is because the local system sizes are typically rather small, too small for the cache optimization strategies of BLAS and LAPACK to pay off. Instead, it is more appropriate to use *ad-hoc* routines. This allows to avoid parameter checking, which is typical for general purpose routines, and it allows to use cheap, highly tuned algorithms. For instance, the linear solve needed in the blocksmoother can be based on LU-factorization *without* pivoting.

5.4 Loop unrolling

Many important parts of AMG code contain small loops over the number of stages. Consider for instance the inner loop of the product of a sparse matrix and a multivector (see Section 5.1) and the routines for the linear solve and matrix-vector product used in the blocksmoother (see Section 5.3). If the body of the loop is small, the loop overhead, consisting of incrementing and checking the loop parameter, is not negligible. Template programming allows the compiler to optimize such loops for each number of stages *separately* by loop unrolling, resulting in the elimination of this overhead.

6 Numerical experiments

The presented algorithms have been implemented in C++. The underlying scalar AMG algorithms are smoothed aggregation nodal multigrid [19] for the div-grad problem and the algorithm of [2] for the curl-curl equation. All numerical results presented in this paper are generated on a Pentium IV 2.4GHz machine using the g++ 3.3.5 compiler. We will show two types of results. First, we illustrate the convergence of the AMG solver for a variety of IRK and BVM discretizations. Second, we discuss the effect of various implementation issues. We show, among other things, that for up to 6 stages, the actual cost per iteration scales almost linearly with the number of stages and not quadratically or cubically, as could be expected from a floating point operation count.

	div-grad			curl-curl		
nbStages	RadauIIA	Gauss	LobattoIIIA	RadauIIA	Gauss	LobattoIIIA
1	50 (22)	41 (19)	/	65 (29)	57 (27)	/
2	41 (22)	38 (19)	42 (19)	55 (26)	52 (26)	56 (26)
3	37 (22)	36 (21)	38 (21)	52 (26)	50 (24)	52 (25)
4	35 (22)	34 (20)	35 (21)	49 (26)	51 (25)	51 (25)
5	35 (22)	34 (20)	36 (21)	51 (25)	49 (25)	52 (24)
6	35 (22)	34 (21)	35 (21)	49 (25)	50 (25)	51 (27)

Table 1: Number of iterations as a function of the number of stages for some fully coupled IRK methods, needed for a V(2,2) AMG cycle as a standalone solver and with BiCGstab acceleration (between brackets) to reach an accuracy $\|residual\|/\|rhs\| < 10^{-14}$ for the finite element discretization of the 2D div-grad and curl-curl equations with $\alpha = \beta = 1$ and a time step of 0.01 on a triangular mesh containing 41624 nodes and 124075 edges on the unit square $[0\ 1]^2$ with homogeneous Dirichlet boundary conditions.

	div-grad		curl-curl	
nbSteps	gam4	gbdf5	gam4	gbdf5
5	36 (21)	37 (29)	50 (26)	54 (27)
10	34 (23)	38 (27)	50 (26)	53 (28)
15	35 (26)	36 (26)	48 (27)	52 (31)
20	34 (24)	36 (26)	49 (28)	55 (31)
25	34 (24)	36 (27)	48 (27)	53 (30)
30	34 (26)	36 (26)	50 (29)	54 (30)

Table 2: Number of iterations as a function of the number of intermediate steps of some BVM methods, needed for a V(2,2) AMG cycle as a standalone solver and with BiCGstab acceleration (between brackets) to reach an accuracy $\|residual\|/\|rhs\| < 10^{-14}$ for the finite element discretization of the 2D div-grad and curl-curl equations with $\alpha = \beta = 1$ and a time step of 0.01 on a triangular mesh containing 41624 nodes and 124075 edges on the unit square $[0\ 1]^2$ with homogeneous Dirichlet boundary conditions.

	div-grad			curl-curl		
nbStages $\setminus$ Δt	10^{-1}	10^{-2}	10^{-3}	10^{-1}	10^{-2}	10^{-3}
1	37 (16)	22 (12)	11 (7)	34 (17)	24 (13)	18 (11)
2	33 (14)	18 (12)	10 (7)	29 (15)	23 (13)	17 (10)
3	33 (16)	17 (11)	10 (7)	29 (17)	22 (12)	16 (10)
4	32 (17)	17 (11)	10 (7)	28 (15)	22 (13)	16 (10)
5	32 (17)	17 (10)	9 (7)	29 (15)	23 (13)	16 (10)
6	32 (19)	17 (11)	9 (7)	29 (16)	21 (12)	16 (10)

Table 3: Number of iterations as a function of the number of stages and the time step for a V(2,2) AMG cycle as a standalone solver and with BiCGstab acceleration (between brackets) to reach an accuracy $\|residual\|/\|rhs\| < 10^{-8}$ for the finite element discretization of the 2D div-grad and curl-curl equations with $\alpha = \beta = 1$ using the RadauIIA IRK family on a triangular mesh containing 41624 nodes and 124075 edges on the unit square $[0\ 1]^2$ with homogeneous Dirichlet boundary conditions.

Tables 1 and 2 show for some fully coupled IRK and BVM methods the number of iterations as a function of the number of stages, needed to solve a 2-dimensional div-grad and curl-curl problem to a high accuracy. The accuracy is chosen high to make the differences in the results more clear. For standalone AMG, the number of iterations typically decreases with increasing number of stages, and the decrease becomes smaller with increasing number of stages. Table 3 shows for the RadauIIA family of IRK methods that the AMG methods converge faster for a smaller time step. Both observations confirm the results of the convergence analysis of Section 4. Similar as for scalar AMG, Krylov acceleration results in a significant reduction of the number of iterations. Note that the convergence results are similar or slightly better than for scalar AMG for the BE problem, which is equivalent to the RadauIIA case with 1 stage.

Different possible implementations of the matrix-vector products or linear solves in (25) are compared in Tables 4 and 5. The use of ad-hoc templated routines instead of the corresponding BLAS or LAPACK routines will be responsible for the main part of the reduction of the time per AMG cycle. The corresponding time reduction is of the order of magnitude of 10. Additional time reductions, achieved by the optimizations explained in Section 5, are illustrated in Figure 4.

The time needed for a fully-optimized block AMG cycle scales only quasi-*linearly* as a function of the number of stages for up to 6 stages (see Figure 4). This is remarkable, as the corresponding number of floating point operations scales quadratically, due to the matrix-vector products $x = A^{-1}b$ used to solve (25). However, thanks to their dense character, these matrix-vector products cause very little cache misses, contrary to the construction of the righthand sides of (25), which is a sparse operation. Due to these cache effects, the cost of the blocksmoother is dominated by the construction of the local righthand sides (see the right picture of Figure 5). The cost of the local dense linear solve $Ax = b$ proves to be much higher then the cost of the corresponding matrix-vector product $x = A^{-1}b$, and has the same order of magnitude as the cost of the construction of the righthand sides. Correspondingly, if the inverses of the local system matrices are *not* constructed and stored in the setup phase, the cost of the blocksmoother scales supralinearly (see the left picture of Figure 5).

Figure 4 shows the cost of s scalar AMG cycles as a reference. This amounts essentially to the cost of a DIRK-based AMG solver. The figure shows that for the same number of stages, the solution by AMG of the linear system arising in a fully implicit Runge-Kutta discretization is not much more expensive (typically a factor of 2 or less) than solving the sequence of s linear systems appearing in a DIRK-based time-discretization. Obviously, for the same number of stages, the order of accuracy of DIRK time discretization is significantly lower. Hence, a DIRK method will require many more (smaller) time steps than a carefully chosen fully coupled IRK method to reach a similar accuracy. Note that a more precise and quantitative comparison of DIRK versus fully implicit RK-methods would require taking into account the detailed aspects of the time step control strategy, the cost of error estimation and many more implementation heuristics. This is outside the scope of the present paper.

7 Conclusion

We have shown by a theoretical analysis and by numerical experiments that it is possible to develop efficient algebraic multigrid methods for fully coupled implicit Runge-Kutta and Boundary Value Method time discretizations of the div-grad and curl-curl equations. Both the analysis and the experiments showed that the convergence behav-

size	templated	non-templated		BLAS	
2	7.00e-09	2.90e-08	(+ 314 %)	3.66e-07	(+ 5129 %)
3	1.80e-08	6.20e-08	(+ 244 %)	4.42e-07	(+ 2356 %)
4	3.00e-08	5.60e-08	(+ 87 %)	5.40e-07	(+ 1700%)
5	4.70e-08	1.04e-07	(+ 121 %)	5.49e-07	(+ 1068 %)
6	6.80e-08	1.62e-07	(+ 138 %)	5.80e-07	(+ 753 %)

Table 4: Average time in seconds of dense matrix-vector products using an ad-hoc templated implementation, an ad-hoc non-templated implementation and BLAS as a function of the matrix size. The compiler optimization flags -O2 and -funroll-loops are used.

size	templated	non-templated		LAPACK	
2	8.00e-08	1.00e-07	(+ 25 %)	7.60e-07	(+ 850 %)
3	1.80e-07	2.40e-07	(+ 33 %)	1.78e-06	(+ 889 %)
4	3.70e-07	4.20e-07	(+ 14 %)	2.06e-06	(+ 457 %)
5	5.10e-07	5.90e-07	(+ 16 %)	3.18e-06	(+ 524 %)
6	7.10e-07	8.70e-07	(+ 23 %)	4.52e-06	(+ 537 %)

Table 5: Average time in seconds of dense system solves using an ad-hoc templated implementation, an ad-hoc non-templated implementation and LAPACK as a function of the matrix size. The compiler optimization flags -O2 and -funroll-loops are used.

ior is similar to the convergence behavior of the underlying scalar multigrid method for the backward Euler time discretization of the same problem. Also, we demonstrated that these multigrid methods offer opportunities to achieve very high cache efficiency and floating point execution rates. Considering the high order of accuracy of the time discretization methods used and the favorable convergence and timings results, the presented algorithms show great promise.

References

[1] P. Bochev, C. Garasi, J. Hu, A. Robinson, and R. Tuminaro, "An improved algebraic multigrid method for solving Maxwell's equations," *SIAM Journal on Scientific Computing*, vol. 25, pp. 623–642, 2003.

[2] T. Boonen, G. Deliége, and Vandewalle, "On algebraic multigrid methods derived from partition of unity nodal prolongators," *Numerical Linear Algebra with Applications*, vol. 13(2-3), no. 2-3, pp. 105–131, 2006.

[3] T. Boonen, J. Van lent, and S. Vandewalle, "Local Fourier analysis of multigrid for the curl-curl equation," Technical Report TW484, Katholieke Universiteit Leuven, Department of Computerscience, December 2006.

[4] A. Bossavit, *Computational Electromagnetism*. Boston: Academic Press, 1998.

[5] M. Brezina, A. Cleary, R. Falgout, V. Henson, J. Jones, T. Manteuffel, S. McCormick, and J. Ruge, "Algebraic multigrid based on element interpolation (AMGe)," *SIAM Journal on Scientific Computing*, vol. 22, pp. 1570–1592, 2001.

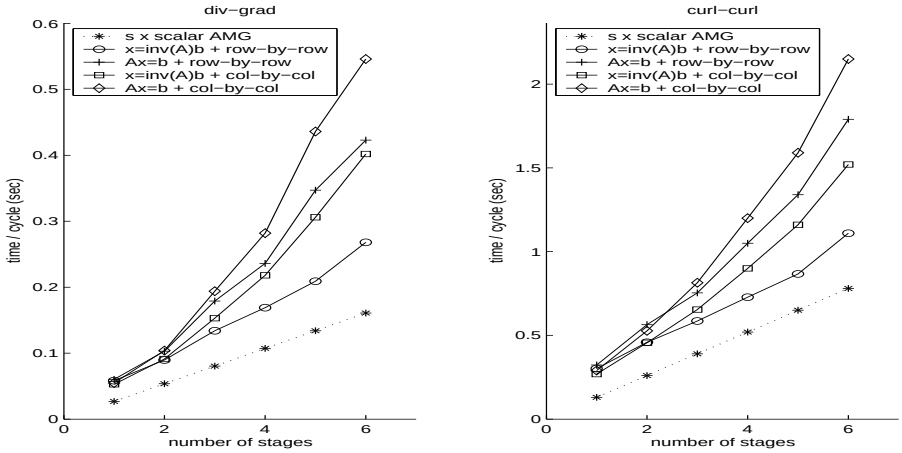

Figure 4: Average time of an AMG cycle for a 2D div-grad and curl-curl problem discretized on the same triangular mesh containing 41624 nodes and 124075 edges as a function of the number of stages, using different optimizations. "$Ax = b$" and "$x = inv(A)b$" indicate whether x in (25) is calculated by a linear solve or by a matrix-vector product (see Section 5.3). "row-by-row" and "col-by-col" refer to the storage layout of the multivectors (see Section 5.1). The time needed for s scalar AMG cycles is indicated for comparison.

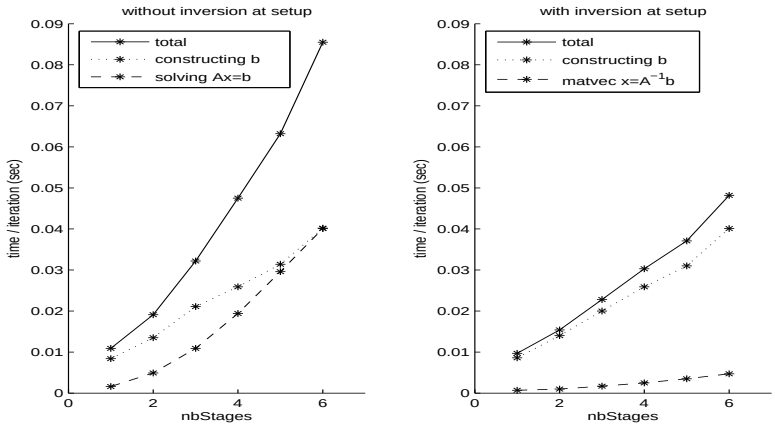

Figure 5: Average time in seconds needed for the basic operation (25) of the blocksmoother and for its dominating components with and without inversion and storage of the local system matrices A in the setup phase. The problem is the discretization of the 2D div-grad equation on a triangular mesh with 41624 nodes on the unit square $[0\ 1]^2$.

[6] L. Brugnano and D. Trigiante, *Solving Differential Problems by Multistep Initial and Boundary Value Methods*, ser. Stability and Control: Theory, Methods and Application. Amsterdam: Gordon & Breach, 1998, vol. 6.

[7] J. Butcher, *Numerical methods for ordinary differential equations*. Chichester: John Wiley & Sons Ltd., 2003.

[8] M. Clemens and T. Weiland, "Discrete electromagnetism with the finite integration technique," *Progress in Electromagnetics Research*, vol. 32, pp. 65–87, 2001.

[9] E. Hairer, S. Norsett, and G. Wanner, *Solving Ordinary Differential Equations I. Nonstiff Problems.*, ser. Springer Series in Comput. Mathematics. Springer-Verlag, 1993, vol. 8.

[10] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.*, ser. Springer Series in Comput. Mathematics. Springer-Verlag, 1996, vol. 14.

[11] R. Hiptmair, "Finite elements in computational electromagnetism," *Acta Numerica*, vol. 11, pp. 237–340, 2002.

[12] R. Hiptmair, "Multigrid method for Maxwell's equations," *SIAM Journal on Numerical Analysis*, vol. 36, no. 1, pp. 204–255, 1999.

[13] J. Hu, R. Tuminaro, P. Bochev, C. Garasi, and A. Robinson, "Toward an h-independent algebraic multigrid method for Maxwell's equations," *SIAM Journal on Scientific Computing*, vol. 27, no. 5, pp. 1669–1688, 2006.

[14] J. Janssen and S. Vandewalle, "Multigrid waveform relaxation on spatial finite element meshes: the discrete-time case," *SIAM J. Sci. Comput.*, vol. 17, no. 1, pp. 133–155, 1996.

[15] K.-A. Mardal, T. Nilssen, and G. Staff, "Order optimal preconditioners for implicit Runge-Kutta schemes applied to parabolic PDEs," *SIAM Journal on Scientific Computing*, to appear.

[16] P. Monk, *Finite element methods for Maxwell's equations*. Oxford: Clarendon Press, 2003.

[17] K. Stueben, "A review of algebraic multigrid," *Journal of Computational and Applied Mathematics*, vol. 128, pp. 281–309, 2001.

[18] U. Trottenberg, C. Oosterlee, and A. Schuller, *Multigrid*. London: Academic Press, 2001.

[19] P. Vanek, J. Mandel, and M. Brezina, "Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems," *Computing*, vol. 56, no. 3, pp. 179–196, 1996.

[20] J. Van lent and S. Vandewalle, "Multigrid methods for implicit Runge-Kutta and boundary value method discretizations of parabolic PDEs," *SIAM Journal on Scientific Computing*, vol. 27, no. 1, pp. 67–92, 2005.

[21] R. Wienands and W. Joppich, *Practical Fourier analysis for multigrid methods*. Chapman and Hall/CRC Press, 2005.