A black–box generalized conjugate gradient minimum residual method based on variable preconditioners and local element approximations

Owe Axelsson[1], Radim Blaheta[2], Maya Neytcheva[3]

December 4, 2007

Abstract

In order to control the accuracy of a preconditioner for an outer iterative process one often involves variable preconditioners. The variability may for instance be due to the use of inner iterations in the construction of the preconditioner. Both the outer and inner iterations may be based on some conjugate gradient type of method, e.g. generalized minimum residual methods.

A background for such methods, including results about their computational complexity and rate of convergence, is given. It is then applied for a variable preconditioner arising for matrices partitioned in two-by-two block form. The matrices can be unsymmetric and also indefinite. The aim is to provide a black–box solver, applicable for all ranges of problem parameters such as coefficient jumps and anisotropy.

When applying this approach for elliptic boundary value problems, in order to achieve the latter aim, it turns out to be efficient to use local element approximations of arising block matrices as preconditioners for the inner iterations.

It is illustrated by numerical examples how the convergence rate of the inner-outer iteration method approaches that for the more expensive fixed preconditioner when the accuracies of the inner iterations increase.

1 Introduction.

This paper aims at presenting a new type of fully robust iterative solution method that can be applied for a variety of different types of problems, such as symmetric and positive definite, unsymmetric, indefinite and coupled system of different types of equations. Due to space limitations, the present paper discusses only the case of single elliptic equations.

[1]Department of Information Technology, Uppsala University, Sweden & Institute of Geonics AS CR, Ostrava, The Czech Republic, email owea@it.uu.se

[2]Institute of Geonics AS CR, Ostrava, The Czech Republic, email blaheta@ugn.cas.cz

[3]Department of Information Technology, Uppsala University, Sweden, email maya@it.uu.se

In applications for partial differential equations, the performance of the methods should be insensitive to practically all ranges of problem parameters involved, such as jumps of coefficients and anisotropy. A user is mostly interested in robustness rather than in fastest rate of convergence. Nevertheless, it turns out that the method to be presented converges also fast, normally with an optimal order convergence rate.

Preconditioning techniques have emerged as an essential part of successful and efficient iterative solutions of linear systems of equations. Preconditioning can be described as a transformation of the original system aiming at accelerating the convergence of iterative methods. Most successful techniques are based on some algebraic form of multilevel techniques and attempt to provide a "black–box" solver, similar to direct solution methods. At the same time, when applied to boundary value problems for partial differential equations, preconditioning aims to provide a mesh independent computational complexity, linearly proportional to the total degree of freedoms in the problem. Hence it aims at an optimal order of complexity which, in general, can not be achieved by direct solution methods which on the contrary can become very costly for large scale problems.

The preconditioning is applied for a parameter-free iterative method, such as a conjugate gradient method.

It is well-known , see e.g. [4], that if the preconditioner B to a matrix A is fixed and A, B are symmetric and positive definite, the preconditioned conjugate gradient method can be viewed as the corresponding unpreconditioned method applied to the preconditioned system $B^{-1}Ax = B^{-1}b$, which is symmetric in the inner product $(x, y)_B = (x, By)$. Therefore the convergence theory obtained for unpreconditioned methods holds also for preconditioned methods. For instance, for normal matrices, all recursions to update the solution and search vectors truncate automatically to short versions and the convergence rate can be determined from the best polynomial approximations (of zero) on the spectrum of $B^{-1}A$.

To fully control the accuracy of preconditioners in particular for ill–conditioned problems, it turns out that variable preconditioners must be used. When the preconditioner may vary from step to step, the above mentioned theory does not hold and more general convergence estimates, not related to eigenvalues, must be used.

An efficient technique to control the accuracy of the preconditioner is to include some inner iterations in its construction. If this is based on a conjugate gradient type of method, where the iterations depend on the initial vector and for which we use some fixed stopping criteria instead of a fixed number of iterations, then the global preconditioning becomes variable. In general, one can then only provide an upper bound of the convergence rate of the method, which in many cases can be overly pessimistic. On the other hand, as shown in [18] under certain, though special, cases these upper bounds may actually be sharp, showing that the method in these cases can not converge faster than for the corresponding steepest descent method. At any rate, we aim at satisfying some convergence bound which makes the method a black-box solver, i.e. fully robust.

For this purpose, we consider preconditioners with inner iterations for matrices partitioned in two-by-two block form. Such a matrix structure arises in a geometric context for finite element approximation of differential equations, when we partition the node set

in "coarse" (vertex) and "fine" (edge and interior) node sets, see e.g. [5]. It also arises in an algebraic context for corresponding partitioning of the matrix graph of a sparse matrix and for some aggregation, agglomeration techniques, see e.g. [14]. The technique can be applied also for systems of PDE's such as Navier-Stokes and Navier-Lamè equations.

In order to save computational labour, it turns out to be advantageous to make an initial transformation of the given matrix with a block (lower) triangular matrix with unit diagonal blocks. Such a transformation preserves the Schur complement matrix and allows a preconditioning on (upper) block triangular form.

The present paper, following [8], summarizes the basic properties of a black–box minimal residual method. A general framework based on local element approximations, to construct preconditioners with inner iterations for matrices partitioned in two-by-two block form is described. Preconditioning for inner iterations based on local element matrices are then discussed. This is partly based on earlier results in [12] and [15]. The paper ends with some numerical illustrations.

2 A black–box generalized minimum residual conjugate gradient method.

To solve a linear system $Ax = b$, where A is a nonsingular matrix of order n, we shall use an iterative method of preconditioned conjugate gradient type. Following [8], [10], the preconditioner will be defined by a, in general nonlinear, mapping $r \rightarrow B[r]$ such that $AB[r] \sim r$, i.e. $B[r]$ is an approximation of $A^{-1}r$ for a given vector r. The following presentation is modified to better cope with the actual preconditioner used here.

As outlined in [1], [3], the method of forming the approximate solutions will be based on certain linear combinations of an increasing set of linearly independent search direction vectors $\{d^{(j)}\}_{j=0}^{k}$. If the preconditioner is a fixed linear mapping, $B[r] = Br$ for some matrix B, this set spans a so called Krylov subspace,

$$\{r^{(0)}, ABr^{(0)}, \ldots, (AB)^k r^{(0)}\},$$

where $r^{(0)} = Ax^{(0)} - b$ is the initial residual, formed from some initial approximation $x^{(0)}$. However, for the purpose of making it possible to control the accuracy of the preconditioner and the convergence rate of the iterative method, in a way which depends on the current residual, we chose here a variable preconditioner by use of certain inner iterations.

The corresponding generalized conjugate gradient minimum residual method (GCG–MR) with a variable preconditioner takes the form:

Let $\widehat{r}^{(0)} = B[r^{(0)}]$ be the pseudo-residual for $x^{(0)}$ and let $d^{(0)} = -\widehat{r}^{(0)}$ be the initial search vector. Let (u, v) and $(u, v)_1$ be inner products and let $s \geq 1$ be a given integer, determining the maximum number of previous search vectors that will be used when forming a new approximation. Let $1 \leq s_k \leq \min(k, s)$.

For $k = 1, 2, \ldots,$ let $\{d^{(j)}\}_{j=k-s_k}^{k-1}$ be $(.,.)_1$ orthogonal search vectors. Then the coeffi-

cients α_j^{k-1} in the next approximation,

$$x^{(k)} = x^{(k-1)} + \sum_{j=k-s_k}^{k-1} \alpha_j^{(k-1)} d^{(j)} \tag{1}$$

will be determined such that the norm of the residual becomes minimal, i.e.

$$\underline{\alpha}^{(k)} = \text{argmin}(r^{(k)}, r^{(k)}),$$

is taken, where $\underline{\alpha}^{(k)} = (\alpha_{k-j}^{(k-1)}),\ 1 \le j \le s_k$.

It follows from (1) that the residuals $r^{(k)} \equiv Ax^{(k)} - b$ can alternatively be computed in a similar manner, i.e.

$$r^{(k)} = r^{(k-1)} + \sum_{j=k-s_k}^{k-1} \alpha_j^{(k-1)} Ad^{(j)}. \tag{2}$$

When $k \ge s_k + 1$, we delete the oldest vector from the recursion and the optimization takes place over a vector space of dimension s, when $k \ge s$.

The minimization property implies

$$\left(r^{(k)}, Ad^{(j)}\right) = 0,\ j = k - s_k, \ldots, k-1, \tag{3}$$

i. e., $r^{(k)}$ is the error in the projection of $r^{(k-1)}$ onto the subspace $\text{span}\{Ad^{(k-s_k)}, \ldots, Ad^{(k-1)}\}$. It follows from (3) that $\underline{\alpha}^{(k)}$ is the solution of

$$\Lambda^{(k)} \underline{\alpha}^{(k)} = \underline{\gamma}^{(k)}, \tag{4}$$

where

$$\Lambda_{ij}^{(k)} = (Ad^{(k-i)}, Ad^{(k-j)}), \quad 1 \le i, j \le s_k, \tag{5}$$

$$\gamma_1^{(k)} = -(r^{(k-1)}, Ad^{(k-1)}), \quad \gamma_j^{(k)} = 0, \quad j = 2, 3, , \ldots, s_k,$$

where $\Lambda^{(k)}$ is symmetric and also positive definite when the vectors $\{Ad^{(j)}\}_{k-s_k}^{k-1}$ are linearly independent. If they are linearly dependent at some step k, then $r^{(k)} = 0$ i.e., assuming exact arithmetic, the solution has already been found.

Further $\Lambda^{(k)}$ equals $\Lambda^{(k-1)}$ augmented with a row and a column. If, due to $k > s$, a truncation takes place, then the oldest row and column of $\Lambda^{(k)}$ is deleted. There is no need to compute any additional matrix – vector products when forming $\Lambda^{(k)}$, except to form $Ad^{(k-1)}$, which is anyway needed for the update of the recursion for $r^{(k)}$.

The new search direction at step k is defined by

$$d^{(k)} = -\widehat{r}^{(k)} + \sum_{j=k-s_k}^{k-1} \beta_j^{(k-1)} d^{(j)},$$

where $\widehat{r}^{(k)} = B[r^{(k)}]$. Here the coefficients are determined by a Gramm–Schmidt orthogonalization so that

$$(d^{(k)}, d^{(j)})_1 = 0, \quad j = k - s_k, \ldots, k-1,$$

that is

$$\beta_j^{(k-1)} = (\widehat{r}^{(k)}, d^{(j)})_1 / (d^{(j)}, d^{(j)})_1, \quad j = k - s_k, \ldots, k-1.$$

We consider two choices of inner products $(.\,,.)_1$:

(i) $(u, v)_1 = (Au, Av), \quad \forall u, v$

(ii) $(u, v)_1 = (u, v), \quad \forall u, v.$

Here $(.\,,.)$ can be the standard l_2-inner product, in which case $(u, v)_1 = (u, A^T A v)$.

In the case (i),

$$(Ad^{(k)},\, Ad^{(j)}) = 0, \quad j = k - s_k, \ldots, k-1$$

implying that $\Lambda^{(k)}$ in (4) becomes diagonal, so $d_{k-j}^{(k-1)} = 0, \quad j = 2, \ldots, s_k$ and the long recursions for $x^{(k)}$ and $r^{(k)}$ truncate automatically to short recursions. The GCG–MR method takes then the following form.

Algorithm 1.

Given $x^{(0)}$ and $s \geq 1$, let $r^{(0)} = Ax^{(0)} - b,\ d^{(0)} = -B[r^{(0)}]$.

For $k = 1, 2, \ldots$ **until** convergence:

Compute $Ad^{(k-1)}$

$$\begin{aligned} \text{Let } \alpha_{k-1}^{(k-1)} &= -\left(r^{(k-1)}, Ad^{(k-1)}\right) / \left(Ad^{(k-1)}, Ad^{(k-1)}\right) \\ x^{(k)} &= x^{(k-1)} + \alpha_{k-1}^{(k-1)} d^{(k-1)} \\ r^{(k)} &= r^{(k-1)} + \alpha_{k-1}^{(k-1)} Ad^{(k-1)} \end{aligned}$$

Compute $\widehat{r}^{(k)} = B[r^{(k)}]$

Compute $A\widehat{r}^{(k)}$

Let $s_k = \min(k, s); \quad d^{(k)} = d^{(k-1)}$

For $j = k - s_k, \ldots, k-1$ let

$\beta_j^{(k-1)} = \left(A\widehat{r}^{(k)}, Ad^{(j)}\right) / \left(Ad^{(j)}, Ad^{(j)}\right)$

$d^{(k)} = d^{(k)} + \beta_j^{(k-1)} d^{(j)}$

End

End

It is seen that the algorithm requires two matrix-vector multiplications with A. This algorithm is similar to the generalized conjugate residual (GCR) in [16]. In that algorithm, one normally replaces the computation of $A\widehat{r}^{(k)}$ by a long recursion, in this way avoiding the second matrix vector multiplication with A. This can, however, not be applied in our case when the preconditioner is variable.

Algorithm 1 achieves the same result as the well–known GMRES method [21], [24], but has a simpler form as the so called upper Hessenberg matrices need not be used. The GMRES is further based on an Arnoldi process to form the orthogonal vectors. Here also two matrix-vector multiplications with A are needed.

3 Reducing influence of rounding errors, computational complexity and rate of convergence of the GCG–MR method.

Rounding errors may cause some loss of orthogonality, and the vectors used in the recursion for x^k and r^k may not be exactly orthogonal, implying that the recursion for such inexact vectors may actually not truncate. To some extent, we may correct for this by computing the right hand side components $\gamma_0^{(k)} = -(r^{(k-1)}, Ad^{(k-j)}), j = 2, 3, \ldots s_k$ in (4) (which in exact arithmetic are equal to zero) and solve $\Lambda^{(k)}\underline{\alpha}^{(k)} = \underline{\gamma}^{(k)}$. To save computational labour, this computation should be performed only at every s step, say. As already pointed out, Algorithm 1 requires two matrix–vector multiplications with A, which may be costly in some applications where A is somewhat less sparse. It can therefore be advisable to apply case (ii), where the approximate solutions are computed by long recursions and the residual is minimized over a set of vectors:

Algorithm 2.

Given $x^{(0)}$, and s, let $r^{(0)} = Ax^{(0)} - b, d^{(0)} = -B[r^{(0)}]$.

For $k = 1, 2, \ldots$ **until** convergence:

Compute $Ad^{(k-1)}$

Let $s_k = \min(k, s)$

For $j = 1, \ldots, s_k$ compute:

$\left(Ad^{(k-1)}, Ad^{(k-j)}\right)$ (to form a new row of $\Lambda^{(k)}$)

End

Compute $\left(r^{(k-1)}, Ad^{(k-1)}\right)$

Solve $\Lambda^{(k)}\underline{\alpha}^{(k)} = \underline{\gamma}^{(k)}$

$$\text{Let} \qquad x^{(k)} = x^{(k-1)} + \sum_{j=1}^{s_k} \alpha^{(k-1)} d^{(k-j)}$$

$$r^{(k)} = r^{(k-1)} + \sum_{j=1}^{s_k} \alpha_j^{(k-1)} Ad^{(k-j)}$$

(or use $r^{(k)} = Ax^{(k)} - b$, if it is cheaper)

Compute $\widehat{r}^{(k)} = B[r^{(k)}]$. Let $d^{(k)} = -\widehat{r}^{(k)}$.

For $j = 1, \ldots, s_k$ let

$\beta_j^{(k-1)} = \left(\widehat{r}^{(k)}, d^{(j)}\right) / \left(d^{(j)}, d^{(j)}\right)$

$d^{(k)} = d^{(k)} + \beta_{k-j}^{(k-1)} d^{(k-j)}$

End

End

To avoid having three long recursions, for $x^{(k)}, r^{(k)}$ and $d^{(k)}$, we can postpone the update of the solutions $x^{(k)}$ until the residual at some step k_0 has become sufficiently small.

Assume then first that no truncation has taken place, i.e. $k_0 \leq s$ and let

$$x^{(k_0)} = x^{(0)} + \sum_{j=0}^{k_0-1} a_j^{(k_0)} d^{(j)}.$$

Then the corresponding residual satisfies

$$r^{(k_0)} = r^{(0)} + \sum_{j=0}^{k_0-1} a_j^{(k_0)} d^{(j)}$$

and the optimality property $\min(r^{(k_0)}, r^{(k_0)})$ implies

$$(r^{(k_0)}, Ad^{(j)}) = 0, \quad 0 \leq j \leq k_0,$$

or

$$\left[(Ad^{(j)}, Ad^{(i)})\right] \left[a_j^{(k_0)}\right] = - \begin{bmatrix} (r^{(0)}, & Bd^{(k_0-1)}) \\ \vdots & \\ (r^{(0)}, & Bd^{(0)}) \end{bmatrix} \tag{6}$$

where the vector $[a_j^{(k_0)}]$ has components $a_j^{(k_0)}, 0 \leq j \leq k_0 - 1$. The coefficient matrix is the same as $\Lambda^{(k)}$ in (4), so the computation of $x^{(k_0)}$ requires only the computation of the k_0 inner products in the right hand side in (6). This computational complexity is, however, less than that required when a long recursion has been used for $x^{(k)}, k = 0, 1, \ldots, k_0$, as in Algorithm 2.

If truncation takes place we must perform such an update of the solution each time is takes place. It is therefore advisable to truncate only after every $k_0 = s$ step, as which points we delete s of the oldest vectors after update of the solution and computes a "fresh" residual, $x^{(k_0)} = Ax^{(k_0)} - b$, not contaminated by rounding errors arising from the corresponding recursion. The first time truncation takes place is after 2s steps. In this way, the number of search directions used in the vector updates varies between s and $2s$.

The GCG–algorithms 1 and 2 require one action of $B[.]$. In addition, Algorithm 1 requires two matrix–vector multiplications with A at each iteration step, while Algorithm 2, based on the inner product $(u, v)_1 = (u, v)$, requires only one such matrix–vector multiplication.

This is the minimal number required for any method. Frequently, as in our application, the action of $B[\,.\,]$ is the most expensive part of the iteration method, and may dominate the efforts involved in the long recursions for the outer iterations.

The average number of arithmetic operations involved in the recursions for $r^{(k)}$ and $d^{(k)}$ is $2\frac{3}{2}s = 3s$, implying $3sn$ operations per iteration step. In addition, there occur $\frac{3}{2}s$ inner products in order to form a new row of $\Lambda^{(k)}$. The computation of $x^{(k_0)}$ requires $\frac{3}{2}k_0 = \frac{3}{2}s$ inner products and the same number of arithmetic operations involved in the recursion, i.e. in average $3n$ operations for every $k_0 = s$ steps.

If $x^{(k)}$ is updated at each step it would cost $\frac{3}{2}sn$ operations, that is, there is a saving for $s > 2$. In practice, one chooses $s \geq 6$, say. The solution of the linear system with matrix $\Lambda^{(k)}$, which has order $\frac{3}{2}s \times \frac{3}{2}s$ (in average), is negligible.

The computational cost of the corresponding inner products and vector updates in Algorithm 1 is $\frac{3}{2}s + 3$.

We conclude that if a matrix–vector multiplication with A requires more than $(4.5s + 3 - (\frac{3}{2}s + 3)) = 3sn$ in operations, which is the total for all occurring vector updates and inner products, then Algorithm 2 can be recommended over Algorithm 1 also from a computational labour point of view.

An upper bound of the rate of convergence of the methods can be determined by two parameters, the coercivity (i.e. positive definiteness) and boundedness parameters, see [4], [8]. For completeness, we present a proof similar but different from that in [1, 3].

Proposition 3.1 *Let the preconditioning mapping $B[\,.\,]$ satisfy the assumptions*

(i) $\quad (v, AB[v]) \geq \delta_1 (v, v) \quad \forall v \in R^n$

(ii) $\quad \| AB[v] \| \leq \delta_2 \| v \| \quad \forall v \in R^n$

for some positive constants $\delta_1,\ \delta_2$. Then the variable-step GCG method converges monotonically and the following convergence rate bound holds,

$$\| r^{(k)} \| \leq \sqrt{1 - (\delta_1/\delta_2)^2}\ \| r^{(k-1)} \|, \quad k = 1, 2, \ldots .$$

Proof Using the orthogonality property (3) we find

$$(r^{(k)}, r^{(k)}) = (r^{(k)}, r^{(k-1)}) = (r^{(k-1)}, r^{(k-1)}) + \alpha_{k-1}^{(k-1)}\, (r^{(k-1)}, Ad^{(k-1)})$$

Further Cramer's rule shows that

$$\alpha_{k-1}^{(k-1)} = \left(r^{(k-1)}, AB[r^{(k)}]\right) \det(\Lambda^{(k-1)}) / \det(\Lambda^{(k)}),$$

so

$$\| r^{(k)} \|^2 = \| r^{(k-1)} \|^2 - \left(r^{(k-1)}, AB[r^{(k)}]\right)^2 \det(\Lambda^{(k-1)}) / \det(\Lambda^{(k)}).$$

It has been shown in [4] that

$$\det(\Lambda^{(k)}) / \det(\Lambda^{(k-1)}) = \min_{g \in W_{k-2}} \| AB[r^{(k-1)}] - g \|^2 ,$$

where W_{k-2} is the vector space spanned by $\{Ad^{(j)}\}_{j=0}^{k-2}$. Therefore the upper bound

$$\| r^{(k)} \|^2 \leq \| r^{(k-1)} \|^2 - \left(r^{(k-1)}, AB[r^{(k-1)}]\right)^2 / \| AB[r^{(k-1)}] \|^2$$

holds, which applying (i) and (ii) completes the proof. ∎

Remark 3.1 *As is readily seen, the above upper bound holds also for the truncated versions of the method, including the steepest descent method which corresponds to* $s = 0$.

Remark 3.2 *There is a simple way to automatically determine if the preconditioner is sufficiently accurate, namely to check the sign of* $\left(r^{(k-1)}, AB[r^{(k-1)}]\right)$, *which equals to* $\left(-r^{(k-1)}, Ad^{(k-1)}\right)$. *Should this sign be negative, then* $\gamma_1^{(k)} < 0$ *and* $\alpha_{k-1}^{(k-1)} < 0$ *since* $\det(\Lambda^{(k)}) > 0$ *as* $\Lambda^{(k)}$ *is symmetric and positive definite. Then no convergence occurs and we must repeat the last step with a more accurate preconditioner, e.g. by performing more inner iterations.*

4 A variable preconditioning for matrices partitioned in two-by-two block form using inner iterations.

In an iterative solution method, preconditioning is the essential part to enhance its rate of convergence. It is normally applied in conjunction with a conjugate gradient type accelerator method.

To control the accuracy of a preconditioning method one can base it on use of inner iterations for instance for some subblocks of the given matrix. Inner-outer iteration methods have been presented in several publications, such as [17, 20, 22, 23, 7, 13]

We apply here these ideas for matrices which have been partitioned in two-by-two block form,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}. \tag{7}$$

Such a partitioning arises naturally for finite element matrices where we can split the node set in a finite element mesh into vertex and remaining nodes (i.e. edge and interior nodes in a 2D context). It also arises for algebraical two–level methods for a sparse general matrix, where we partition the matrix graph in a number of subdomains (elements) and corner nodes are separated from remaining nodes.

We assume that A_{11} and the Schur complement matrix, $S_2 = A_{22} - A_{21}A_{11}^{-1}A_{12}$ are nonsingular. As shown in the next Proposition, A can then be factored in block matrix form and its inverse can be written in a corresponding two-by-two block form.

Proposition 4.1 *Given the matrix A partitioned like in (7) where* A_{11} *and* $S_2 = A_{22} - A_{21}A_{11}^{-1}A_2$ *are assumed to be nonsingular, then A can be factorized in the form*

(a)
$$A = \begin{bmatrix} I_1 & 0 \\ A_{21}A_{11}^{-1} & I_2 \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ 0 & S_2 \end{bmatrix} \tag{8}$$

(b) A *is nonsingular and*

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} + A_{11}^{-1}A_{12}S_2^{-1}A_{21}A_{11}^{-1} & -A_{11}^{-1}A_{12}S_2^{-1} \\ -S_2^{-1}A_{21}A_{11}^{-1} & S_2^{-1} \end{bmatrix}$$

(c) If also A_{22} *is nonsingular, then* $S_1 = A_{11} - A_{12}A_{22}^{-1}A_{21}$ *exists and is nonsingular. Further;*

(d) $A_{11}^{-1}A_{12}S_2^{-1} = S_1^{-1}A_{12}A_{22}^{-1}$

(e) $A_{11}^{-1} + A_{11}^{-1}A_{12}S_2^{-1}A_{21}A_{11}^{-1} = S_1^{-1}$

(f) $A^{-1} = \begin{bmatrix} S_1^{-1} & -S_1^{-1}A_{12}A_{22}^{-1} \\ -S_2^{-1}A_{21}A_{11}^{-1} & S_2^{-1} \end{bmatrix}.$

Proof Part (a) is the block factorization of A, which leads to part (b) by direct computations.

If A_{22} is nonsingular, then S_1 is also nonsingular, which follows from a reordering $1 \leftrightarrow 2$ of A to $\widehat{A} = \begin{bmatrix} A_{22} & A_{21} \\ A_{21} & A_{11} \end{bmatrix}$ and a block matrix factorization of $\widehat{A}$, which shows part (c) since A, and hence $\widehat{A}$, are nonsingular.

Part (d) follows from

$$\begin{aligned} S_1A_{11}^{-1}A_{12}S_2^{-1} &= (I - A_{12}A_{22}^{-1}A_{21}A_{11}^{-1})A_{12}S_2^{-1} = \\ &= A_{12}(I - A_{22}^{-1}A_{21}A_{11}^{-1}A_{12})S_2^{-1} = A_{12}A_{22}^{-1}(A_{22} - A_{21}A_{11}^{-1}A_{12})S_2^{-1} = A_{12}A_{22}^{-1}, \end{aligned}$$

which implies

$$A_{11}^{-1} + A_{11}^{-1}A_{12}S_2^{-1}A_{21}A_{11}^{-1} = (I_1 + S_1^{-1}A_{12}A_{22}^{-1}A_{21})A_{11}^{-1} = S_1^{-1}A_{11}A_{11}^{-1} = S_1^{-1}$$

and part (f).

Although these and related results may be familiar to many readers, we include them here (with a short proof) for clarity. ∎

Normally, when constructing a preconditioner for A one bases it on (8) and replaces A_{11} and S_2 with some approximations. This requires two actions of the approximation of A_{11} and one for S_2, at each iterations step. In order to save computational labour, we make here instead an initial transformation of the matrix with a lower block triangular matrix with unit diagonal blocks. Here the lower, left matrix L_{21}, which approximates $A_{21}A_{11}^{-1}$, can be computed based on the inverses of local element matrices. This implies that L_{21} is sparse and the transformed matrix $\widetilde{A}$ is still as sparse as the original matrix A is.

This idea was first presented in [12]. The resulting matrix can then be efficiently preconditioned from right by an upper block triangular matrix, implying that only a single action of the approximation of matrix A_{11} is needed. An additional gain using the initial

transformation is that the lower block diagonal matrix $\widetilde{A}_{22}$, which is still sparse, is close to the Schur complement matrix S_2 and can therefore be used as an iteration matrix for inner iterations. On the other hand, note that when we use standard, as opposed to hierarchical, finite element basis functions the original matrix A_{22} in finite element contexts is normally diagonal, implying that it can not be a useful approximation of S_2.

The above follows from the next proposition.

Proposition 4.2 *The Schur complement of a matrix A of the 2×2 structure (7) is invariant under any transformation of the form*

$$\widetilde{A} = \begin{bmatrix} I_1 & 0 \\ -L_{21} & I_2 \end{bmatrix} A,$$

that is, $S_2(A) = S_2(\widetilde{A})$. Further, it holds

$$\widetilde{A} = \begin{bmatrix} A_{11} & A_{12} \\ \widetilde{A}_{21} & \widetilde{A}_{22} \end{bmatrix},$$

where $\widetilde{A}_{21} = A_{21} - L_{21}A_{11}$, $\widetilde{A}_{22} = A_{22} - L_{21}A_{12}$.

Proof A simple computation gives the above expression of $\widetilde{A}$. Further,

$$\begin{aligned} S_2(\widetilde{A}) &= \widetilde{A}_{22} - \widetilde{A}_{21}A_{11}^{-1}A_{12} = \widetilde{A}_{22} - (A_{21} - L_{21}A_{11})\,A_{11}^{-1}A_{12} \\ &= A_{22} - A_{21}A_{11}^{-1}A_{12} = S_2(A)\,. \end{aligned}$$

■

Note that $\widetilde{A}_{21} = E_{21}A_{11}$, where $E_{21} = A_{21}A_{11}^{-1} - L_{21}$ and

$$\widetilde{A}_{22} = S_2(A) + E_{21}A_{12}. \tag{9}$$

Here E_{21} is the error matrix arising from the initial transformation and approximation L_{21} of $A_{21}A_{11}^{-1}$.

Matrix $\widetilde{A}$ can now be preconditioned from right by the upper block triangular matrix

$$R = \begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}S^{-1} \\ 0 & S^{-1} \end{bmatrix} \tag{10}$$

where S is an approximation of $S_2(A)$.

Using (9), a direct computation shows that

$$\widetilde{A}R = \widetilde{A}\begin{bmatrix} A_{11}^{-1} & -A_{11}^{-1}A_{12}S^{-1} \\ 0 & S^{-1} \end{bmatrix} = \begin{bmatrix} I_1 & 0 \\ \widetilde{A}_{21}A_{11}^{-1} & S_2S^{-1} \end{bmatrix}.$$

As is seen from (10), each action of R involves one action of S^{-1} followed by one action of A_{11}^{-1}, namely

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = R \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \text{ is computed as } \begin{array}{ll} (i) & x_2 = S^{-1}y_2, \\ (ii) & x_1 = A_{11}^{-1}(y_1 - A_{12}x_2). \end{array} \tag{11}$$

In a variable preconditioning method we may perform these actions using inner iterations, preferably using a conjugate gradient method of GCG–MR type if the matrices are unsymmetric or of classical CG–type if they are symmetric and positive definite.

5 Local finite element approximations

While the approach taken in the paper up to this point has been general, applicable not only for finite element matrices, we now present a method which is more based on the finite element framework.

The outer iterations in the GCG–MR method involve matrix-vector actions with matrix $\widetilde{A}R$. Actions of $\widetilde{A}$ are straightforwardly implemented and involve only sparse matrices. To perform the actions with R, i.e. to compute $x_1,\ x_2$ in (11), we will involve inner iterations for A_{11}^{-1}, using some preconditioner B_{11} as an approximation of A_{11}^{-1}. The arising actions of S^{-1} to compute x_2 in (11,i) depend on the choice of S. We consider two variants,

$$\begin{array}{ll} (i) & S = \widetilde{A}_{22}, \\ (ii) & S = \widetilde{A}_{22} - \widetilde{A}_{21} B_{11} A_{12}. \end{array} \tag{12}$$

Here, S is still a sparse matrix with a favourable band matrix structure, such as the original matrix has for a proper ordering of the unknowns. Since the order of S is much smaller than that for the original matrix, it may be viable to use a direct solution method to compute x_2. However, if the order is still too big, we can partition S in a similar way as the original matrix and use again a preconditioned iterative method based on the two-by-two block partitioning to solve the system with S. This procedure can be repeated by using the multilevel framework, see e.g. [9]. Alternatively, one can use the coarse mesh finite element matrix as preconditioner and possibly solve this by such a multilevel preconditioned conjugate gradient iterative method.

To indicate that the latter can be an efficient approach, we recall from e.g. [4] that for symmetric positive definite problems there holds a spectral relation,

$$(1-\gamma^2) x^T \widehat{A}_{22} x \leq x^T S_2(A) x \leq x^T \widehat{A}_{22} x \qquad \forall\, x$$

where $\widehat{A}_{22}$ is the coarse mesh matrix and γ is the constant in the corresponding Cauchy-Bunyakowski-Schwarz inequality. Here γ takes values strictly less than unity, independent on the mesh size, anisotropy in the differential equation and element shape if we use piecewise linear finite element approximations.

For $S = \widetilde{A}_{22}$ we get

$$\widetilde{A}R = \begin{bmatrix} I_1 & 0 \\ \widetilde{A}_{21} A_{11}^{-1} & S_2(A) \widetilde{A}_{22}^{-1} \end{bmatrix}$$

where $S_2(A)\widetilde{A}_{22}^{-1} = I_2 - \widetilde{A}_{21} A_{11}^{-1} A_{12} \widetilde{A}_{22}^{-1}$. It follows that if we perform a sufficient number of inner iterations to solve the arising system with matrix A_{11} in (11), then the convergence of

the outer iteration method is essentially determined by the eigenvalues of $S_2(A)\widetilde{A}_{22}^{-1}$, which are close to unity if matrix $\widetilde{A}_{21}$ is small, i.e. if L_{12} is a sufficiently accurate approximation of $A_{21}A_{11}^{-1}$.

The way how to construct S, described in (12), can be extended to get increasingly more accurate approximations of the exact Schur complement matrix

$$S_2(\widetilde{A}) = \widetilde{A}_{22} - \widetilde{A}_{21}A_{11}^{-1}A_{12},$$

namely by adding more terms in the Neumann expansion of A_{11}^{-1}, i.e.

$$A_{11}^{-1} = B_{11} + B_{11}(I - A_{11}B_{11}) + B_{11}(I - A_{11}B_{11})^2 + \cdots .$$

The convergence assumes, however, that

$$\| I - A_{11}B_{11} \| < 1.$$

The next term in the approximation, following (12, i, ii) is then

$$(iii) \quad S = \widetilde{A}_{22} - \widetilde{A}_{21}(2B_{11} - B_{11}A_{11}B_{11})A_{12}. \tag{13}$$

A straightforward computation shows that $S_2(A)S^{-1}$ take the following forms for the three variants,

(i) if $S = \widetilde{A}_{22}$, then $S_2(A)S^{-1} = I_2 - E_{21}A_{12}\widetilde{A}_{22}^{-1}$, where $E_{21} = A_{21}A_{11}^{-1} - L_{21}$, a first order method,

(ii) if $S = \widetilde{A}_{22} - \widetilde{A}_{21}B_{11}A_{12}$, then $S_2(A)S^{-1} = (I_2 - E_{21}(A_{11}B_{11} - I_1)A_{12}S_2(A)^{-1})^{-1}$, a second order method,

(iii) if S is as in (13), then $S_2(A)S^{-1} = (I_2 + E_{21}(A_{11}B_{11} - I_1)^2A_{12}S_2(A)^{-1})^{-1}$, a third order method.

In the numerical tests to follow we have used as approximation of $S_2(A)$ the assembly of the exact local Schur complements.

It remains to discuss the choice of the preconditioner for the arising systems with A_{11} This will be based on elementwise approximations, see e.g. [6, 12, 15].

Since A_{11} is the assembly of local element matrices $A_{11,E}$, we can use

$$B_{11} = \sum_E R_E^T A_{11,E}^{-1} R_E \tag{14}$$

as an approximation to A_{11}^{-1}. Here, R_E is the restriction to element E and $R_E^T A_{11,E}^{-1} R_E$ denotes the extension of the element matrix inverse $A_{11,E}^{-1}$ to the global order. Note that R_E is a Boolean matrix.

It can be readily seen, that this choice does not work well for problems with coefficient jumps (between the elements E). For these problems, we assemble first the global matrix A_{11} and take its restrictions to elements E,

$$\widehat{A}_{11,E} = R_E A_{11} R_E^T.$$

The approximation B_{11} to A_{11}^{-1} is then formed as

$$B_{11} = \sum_E R_E^T \widehat{A}_{11,E}^{-1} R_E. \tag{15}$$

Due to space limitations, we will not present an analysis of the residual error $A_{11}B_{11} - I$ in this paper. We refer instead to [12].

6 Numerical tests

The major aim of the numerical tests is to show how the number of outer iterations depend on the number of inner iterations for the arising sub-block systems. All experiments are done using `MATLAB`.

For illustration purposes we implemented some variant of the described methods and show now the results obtained by solving two model type problems in 2D.

Problem 1 (Diffusion problem with discontinuous coefficients)

$$-\mathrm{div}(a\nabla u) = f \qquad \text{in } \Omega = [0,\,1]^2 \tag{16}$$

$$u = 0 \qquad \text{on } \partial\Omega \tag{17}$$

with a discontinuous coefficient a. The value of a is set to one in the whole domain Ω except in the subregion $0.5 \le x, y \le 0.75$, where a can vary. The discretization is done with right-angled triangles, which have varying size but are aligned with the discontinuity region.

Problem 2 (Model heat equation)

$$\begin{aligned} \frac{\partial u}{\partial t} - \mathrm{div}(\nabla u) &= f(x,y,t), \quad u = u(x,y,t), (x,y) \in \Omega = [0,\,1]^2, t > 0 \\ u &= 0 \qquad\qquad \text{on } \partial\Omega \\ u(x,y,0) &= g(x,y) \end{aligned} \tag{18}$$

The initial condition function $g(x,y)$ is discontinuous. It equals one in all points lying in a circle centered at $(0.5, 0.5)$ and radius equal to 0.2, and is zero elsewhere. The function $f(x,y,t)$ is taken to be the constant function 1.

To discretize in time, we use the so-called θ-method (see e.g., [2]), resulting in a sequence of linear equations

$$u(t+\delta_t) - \theta\delta_t\,\mathrm{div}(\nabla u(t+\delta_t)) = u(t) + (1-\theta)\delta_t\,\mathrm{div}(\nabla u(t)) + \delta_t$$

$t = 0, \delta_t, \cdots$, where $\delta_t > 0$ is the time-step and the parameter θ is chosen in the interval $[0, 1]$. Here $\theta = 1$ corresponds to the implicit Euler method and $\theta = 0.5$ corresponds to the trapezoidal method.

The problem is then discretized in space using triangular meshes and piecewise linear finite element basis functions. Figure 2 illustrates one triangulation and the initial condition $g(x, y)$. The fully discretized problem has the form

$$(M + \theta\delta_t K)U(t + \delta_t) = (M - (1 - \theta)\delta_t K)U(t) + \delta_t [\theta F(t + \delta_t) + (1 - \theta)F(t)] \qquad (19)$$

$t = 0, \delta_t, \cdots$. Here M is the corresponding mass matrix and K is the stiffness matrix for the operator $-\text{div}(\nabla u)$.

The time-step δ_t can be variable, however in these experiments it is constant and equals h_{max}, where h_{max} is the largest edge length in the corresponding spatial mesh. The theory recommends to choose the parameter θ as $0.5 + \xi$ for some small value of ξ in order to maintain second order of accuracy and at the same time avoid oscillations in the numerical solution (see for details [2]). Here, to better handle the discontinuity in the initial value, we choose θ to be equal to $1 - h_{max}^2$.

We first discuss the results in Tables 1 and 2, corresponding to Problem 1.

The matrix A of the corresponding algebraic system is first transformed by applying one-sided transformation of the form $\begin{bmatrix} I_1 & -Z_{12} \\ 0 & I_2 \end{bmatrix}$ and the resulting system is solved by a GCG–MR method, preconditioned by $\begin{bmatrix} A_{11} & 0 \\ A_{21} & S \end{bmatrix}$. The iterations are terminated after reducing the norm of the residual by a factor 10^{-6}. Here S is the approximation of the Schur complement of A, computed as an assembly of local Schur complements, $S = \sum_E R_E^T (A_{22,E} - A_{21,E} A_{11,E}^{-1} A_{12,E}) R_E$. where the matrices R_E are the same as in (15). Similarly, $Z_{12} = \sum_E R_E^T A_{11,E}^{-1} A_{12,E} R_E$. Systems with A_{11} are solved by an inner iteration method, preconditioned by B_{11}^{-1} computed as in (15). We point out that since we construct an approximate inverse of A_{11}, which is sparse, the preconditioning step in the inner iteration method is cheap - it involves one multiplication with a sparse matrix of the same structure as A_{11} itself.

Since in this case the matrix A_{11} is symmetric and positive definite, we can use the standard conjugate gradient method for the inner iterations with A_{11} which can demand a significantly lower arithmetic cost than the use of the GCG–MR method.

The primary goal is therefore to reduce the number of outer iterations. It is first tested how the number of the outer iterations depends on the number of inner iterations for A_{11}, see Table 1.

We see that the number of outer iterations decays with increasing number of inner iterations until a threshold (8 to 9) iterations is reached. Clearly one should try to find a proper number of inner iterations to achieve this threshold without performing too many iterations. This can be done by applying a proper relative stopping criteria for the inner iterations.

Number of inner iter.	jump 10^{-3}		jump 1		jump 10^{+3}	
	outer.it	total inner.it	outer.it	total inner.it	outer.it	total inner.it
1	22	22	22	23	25	25
2	13	26	13	26	16	32
3	9	27	9	27	11	33
4	8	32	8	32	10	40
6	8	48	8	64	9	54
8	8	64	8	64	9	72
10	8	80	8	80	9	90
rel.10^{-3}	8(6)	48	8(6)	48	9(5)	45
rel.10^{-2}	9(4)	36	9(4)	36	26(4)	104

Table 1: Problem 1: Number of outer iterations as a function of the number of inner iterations for problem size 37121. Outer iteration stopping criterion: rel. 10^{-6}
. Average numbers of inner iterations are in brackets

Size(A)	jump 10^{-3}		jump 1		jump 10^{+3}	
Size(A_{11})	3 in.it	rel.10^{-3}	3 in.it	rel.10^{-3}	3 in.it	rel.10^{-3}
609	8(3)	8(6)	8(3)	8(6)	10(3)	9(5)
448	24	48	24	48	30	45
2369	8(3)	8(6)	8(3)	8(6)	10(3)	9(6)
1760	24	48	24	48	30	54
9345	9(3)	8(6)	9(3)	8(6)	11(3)	9(6)
6976	27	48	27	48	33	54
37121	9(3)	8(6)	8(3)	8(6)	11(3)	9(5)
27776	27	48	24	48	33	45
147969	10(3)	8(6)	9(3)	8(6)	9(3)	8(6)
110848	30	48	27	48	27	48
590849	24(3)	8(9)	11(3)	8(6)	22(3)	18(6)
442880	72	72	33	48	66	108

Table 2: Problem 1: First line: number of outer iterations with inner iterations within brackets. Second line: total number of iterations with A_{11}

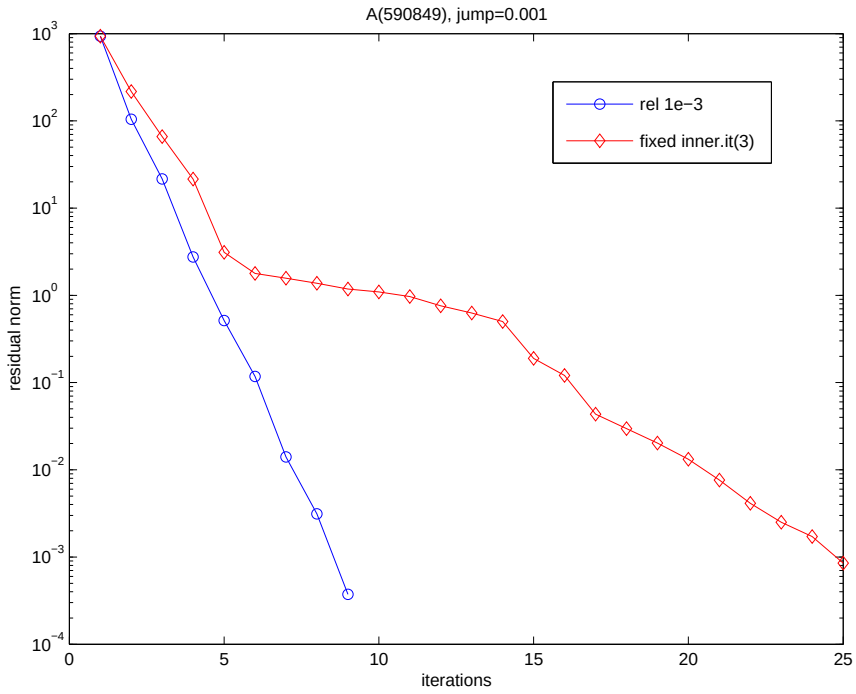

Figure 1: Problem 1: Decay of the outer iteration residual

In Table 1, for a jump in the coefficients of order 10^{+3}, a noticeable increase of the number of outer iterations is encountered. The reason for that is that the number of inner iterations per outer iteration varies. It turns out that in the beginning of the iterative process, the inner stopping criterion 10^{-2} is reached within two inner iterations, which does not provide the necessary quality of the outer preconditioner and entails more outer iterations. We see, that if we require three inner iterations for all outer iterations, the outer iteration count is twice smaller, 11 instead of 26.

Table 2 shows how the number of outer and inner iterations varies with problem size. It is seen that when a relative stopping criteria is used, the number of outer iterations does not vary with the problem size. The number of inner iterations is quite stable. Only for the largest problem size it did increase when a relative stopping criteria was used. A less accurate relative stopping criterion (10^{-2}) works also satisfactory, except for coefficient jumps from 1 to 10^{+3}.

Figure 1 illustrates clearly the advantage of using a relative stopping criteria ($\varepsilon = 10^{-3}$) as opposed to using a fixed number of inner iterations (3 in this case). It illustrates the decay of outer iteration residuals as a function of number of iterations. It is seen that for an inner iteration with a relative stopping criterion, the decay of the logarithm of the residual is linear while this does not hold when a fixed number of inner iteration is used.

Clearly we make substantial savings by having a low, constant number of outer iterations, independent of problem size. It saves in number of Schur complement matrix

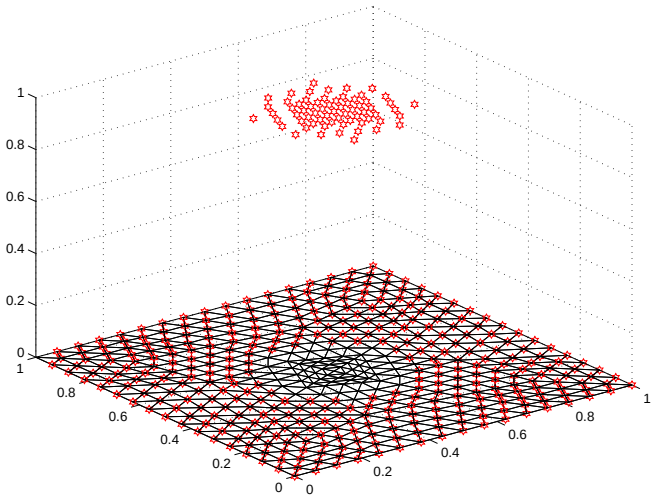

Figure 2: Problem 2: Example of a space discretization mesh and the initial condition (marked with hexagonals)

solutions, long recursions and number of inner products as well as in overhead in the form of data shuffling and message passing in a parallel computing environment, in particular for very large scale problems. These savings are very important and the benefits of the inner-outer iterative approach was already observed, see e.g. [11, 12] and the references therein.

Table 3 illustrates the behaviour of the block-factorized preconditioner

$$\begin{bmatrix} A_{11} & 0 \\ A_{21} & S \end{bmatrix} \begin{bmatrix} I_1 & Z_{12} \\ 0 & I_2 \end{bmatrix}$$

applied to the matrix $A = M + \theta\delta_t K$ in Problem 2. We perform 10 timesteps and report the largest number of outer iterations. The solution methods is as for Problem 1. The stopping criterion in this case is relative, of magnitude 10^{-4}. The approximation of the block A_{11} is as in (15). Here we do not perform any initial transformation of the system matrix.

It is seen that the number of outer iterations is very small, independent of the space discretization parameter and also essentially on the time-step. Only very large time-steps, (not used in practice) could possibly provoke an increase of the number of outer iterations.

Remark 6.1 *In the experiments, the system whith the approximate Schur complement in the preconditioner is solved exactly. Other experiments, not included here, show that it requires only two inner iterations per outer iteration when solved by an inner solver, preconditioned by an algebraic multilevel iteration method based on a recursive application of the local Schur approximation idea. The latter behaviour, illustrated on symmetric and nonsymmetric linear system originating from a visco-elasticity problem, is reported in [11] and [12].*

Problem size	Outer iter(Aver.inner iter)
417	7(5)
1601	7(5)
6272	8(6)
24833	8(6)
98817	8(6)
394241	8(6)

Table 3: Problem 2: Average outer iterations for ten timesteps and in brackets, the average number of inner iterations to solve systems with A_{11}

Remark 6.2 *All numerical experiments indicate that the total number of iterations to solve A_{11} is fairly constant, irrespective of the number of inner solves. Clearly, decreasing the number of outer iterations by having a sufficient, but not too big number of inner iterations, we save on fewer number of Schur complement solves and overhead (matrix–vector multiplications, data transfers etc.).*

7 Conclusions

It has been shown that an efficient approach to solve sparse matrix problems is to apply variable preconditioners, i.e., by solving block matrix equations by inner iterations, for certain subblocks which arise in the global matrix preconditioner for matrices partitioned in two-by-two block matrix form. The variability arises from using conjugate gradient methods also for the inner iterations. By using a sufficient number of inner iterations, the behaviour of the outer iterative method approaches that for a corresponding fixed but more expensive, optimal order preconditioner. This can result in few outer iterations, in this way making substantial gains in an overall lower computational complexity.

The number of inner iterations can be made small and can be performed cheaply and even in parallel by use of an efficient preconditioner, such as the assembly of certain local element approximations.

The combined method results in an efficient black-box solver applicable for a variety of different types of problems, and the behaviour of the method is insensitive to the actual values of problem parameters involved.

Acknowledgement

The work of the first two authors was supported by the Academy of Sciences of the Czech Republic through Research Plan AVOZ30860518 of the Institute of Geonics.

References

[1] O. Axelsson, Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations, *Linear Algebra and its Applications* 29(1980), 1–16.

[2] O. Axelsson, Error estimates over infinite intervals of some discretizations of evolutionary quations, *BIT*, 24 (1984), 413-424.

[3] O. Axelsson, A generalized conjugate gradient, least square method, *Numer. Math.* 51(1987), pp. 209–227.

[4] O. Axelsson, *Iterative Solution Methods*, Cambridge University Press, Cambridge, 1994.

[5] O. Axelsson and V.A. Barker, *Finite Element Solution of Boundary Value Problems. Theory and Computation*, Academic Press, Inc, 1984. Re-edition Classics in Applied Mathematics 35, SIAM Philadelphia 2001.

[6] O. Axelsson, R. Blaheta, and M. Neytcheva, Preconditioning of boundary value problems using elementwise Schur complements. Technical Report 2006-048, Department of Information Technology, Uppsala University, November 2006.

[7] O. Axelsson, M. Neytcheva, An iterative solution method for Schur complement systems with inexact inner solver. In O. Iliev, M. Kaschiev, S. Margenov, Bl. Sendov, P.S. Vassilevski eds., *Recent Advances in Numerical Methods and Applications II*, World Scientific, 1999, 795–803.

[8] O. Axelsson and P. S. Vassilevski, A black–box generalized conjugate gradient solver with inner iterations and variable–step preconditioning, *SIAM J. Matrix Anal. Appl.* 12(4)(1991) , 625–644.

[9] O. Axelsson and P.S. Vassilevski, Algebraic multilevel preconditioning methods, II, *SIAM J. Numer. Anal.*, 27 (1990), 1569-1590.

[10] O. Axelsson and P.S. Vassilevski, Variable–step multilevel preconditioning methods. I. Selfadjoint and positive definite elliptic problems. *Numer. Linear Algebra Appl.*, 1 (1994), 75-101.

[11] E. Bängtsson, B. Lund, A comparison between two solution techniques to solve the equations of glacially induced deformation of an elastic Earth, *International Journal for Numerical Methods in Engineering*, accepted, 2007.

[12] E. Bängtsson, M. Neytcheva, Finite element block-factorized preconditioners. Technical Report 2007-008, Department of Information Technology, Uppsala University, 2007.

[13] R. Blaheta, GPCG – generalized preconditioned CG method and its use with non-linear and non-symmetric displacement decomposition preconditioners. Numer. Linear Alg. Appl. 9(2002), 527-550

[14] R.Blaheta, Algebraic Multilevel Methods with Aggregations: An Overview. In *Large-Scale Scientific Computing*, Berlin : Springer, LNCS 3743/2006. pp. 3-14.

[15] R. Blaheta, P. Byczanski, Algebraic multilevel preconditioners: Elementwise constructed pivot block preconditioners II, Tech. Report DAMCS 2007/3, Institute of Geonics ASCR, Ostrava 2007.

[16] S.C. Eisenstat, H.C. Elman, M.H. Schultz, Variational iterative methods for nonsymmetric systems of linear equations, *SIAM J. Numer. Anal.*, 20(1983), 345–357.

[17] G.H. Golub and Q. Ye, Inexact preconditioned conjugate gradient method with inner–outer iteration. *SIAM J. Sci. Comput.*, 21(4): 1305–1320, 1999/00.

[18] A.V. Knyazev and I. Lashuk, Steepest descent and conjugate gradient methods with variable preconditioning. Electronic. Math. NA/0605767, arXiv. org., http://arxiv.org/abs/math/0605767, 2006-2007.

[19] J. Kraus, Algebraic multilevel preconditioning of finite element matrices using local Schur complements, *Numerical Linear Algebra with Applications*, 13(2006),49–70.

[20] Y. Notay, Flexible conjugate gradients, *SIAM J. Sci. Comput.*, 22(4): 1444–1460.

[21] Y. Saad and M.H. Schultz, GMRES: A generalized minimum residual algorithm for solving nonsymmetric linear systems, *SIAM J. Sci. Stat. Comput.* 7(1986), 856–869.

[22] V. Simoncini and D.B. Szyld, Flexible inner–outer Krylov subspace methods. *SIAM J. Numer. Anal.*, 40(2003), 2219–2239

[23] V. Simoncini and D.B. Szyld, Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM J. Sci. Comput.*, 25(2): 457–477, 2003.

[24] Homer F. Walker and Lu Zhou, A simpler GMRES, *Numerical Linear Algebra with Applications*, 1(1994), 571–581.