NUMERICAL LINEAR ALGEBRA WITH APPLICATIONS
Numer. Linear Algebra Appl. 2002; **9**:141–162 (DOI: 10.1002/nla.260)

A parallel block overlap preconditioning with inexact submatrix inversion for linear elasticity problems

Igor E. Kaporin and Igor N. Konshin*,†

Computing Center RAS, Vavilov Str. 40, 117967 Moscow, Russia

SUMMARY

We present a parallel preconditioned iterative solver for large sparse symmetric positive definite linear systems. The preconditioner is constructed as a proper combination of advanced preconditioning strategies. It can be formally seen as being of domain decomposition type with algebraically constructed overlap. Similar to the classical domain decomposition technique, inexact subdomain solvers are used, based on incomplete Cholesky factorization. The proper preconditioner is shown to be near optimal in minimizing the so-called K-condition number of the preconditioned matrix. The efficiency of both serial and parallel versions of the solution method is illustrated on a set of benchmark problems in linear elasticity. Copyright © 2002 John Wiley & Sons, Ltd.

1. INTRODUCTION

In the present paper, we consider the preconditioned conjugate gradient (PCG) algorithm [1] for solving linear algebraic system

$$Ax = b \tag{1}$$

with a large sparse unstructured symmetric positive definite (SPD) matrix A of order n, such as arising in computational mechanics problems. Here we concentrate our efforts on development of parallel preconditioners that provide sufficiently fast convergence, at least not slower that the best known preconditioners of incomplete Cholesky type. As a basic technique, we use the block incomplete inverse Cholesky (BIIC) factorization first proposed in Reference [2] and considered in more details in Reference [3]. For the purpose of better parallelism, we use the additive form of this preconditioner given in Reference [3].

The BIIC preconditioner, as described in Reference [3], is based on some $s \times s$ block form of the matrix A, where *exact* Cholesky factorization is used to solve subsystems associated with the augmented diagonal blocks. When aiming at efficient parallel solution of (1) for large

*Correspondence to: Igor N. Konshin, Computing Center RAS, Vavilov Str. 40, 117967 Moscow, Russia
†E-mail: konshin@sci.kun.nl, horse@ccas.ru

Contract/grant sponsor: The Netherlands Technology Foundation (STW); contract/grant number: NNS. 4683
Contract/grant sponsor: The National Science Foundation; contract/grant number: NWO 047.008.007

Received 12 January 2001
Revised 30 July 2001

matrices, the block size will be approximately n/s, where s is usually taken to be equal to the number of processors available. It is obvious that performing exact factorizations of still large enough blocks is too cost expansive for large problems. However, a viable alternative is to use an incomplete factorization and for this we choose the so-called robust incomplete Cholesky second order factorization (IC2) method described in Reference [4]. Practice has shown that using IC2 reduces significantly the preconditioning costs almost without impairing the quality of the resulting preconditioner.

As a consequence, most eigenvalues of the preconditioned matrix become highly clustered around unity while the remaining ones may lie close to zero or slightly above unity. Such eigenvalue distribution ensures a low number of the CG iterations needed for the convergence even if the spectral condition number of the preconditioned matrix is still rather large [5].

That is, inserting some sufficiently *inexact* factorization within BIIC, one can obtain very promising preconditioning by properly balancing the following three criteria: parallel efficiency, arithmetic costs per iteration, and rate of the PCG convergence.

Our construction results in a purely algebraic procedure which is applicable to the preconditioning of any sparse SPD matrix independent of the nature of the underlying physical problem and the type of discretization method used.

The paper is organized as follows: in Section 2 the alternative theory of PCG convergence is presented which supports the basic BIIC construction described in Section 3, in Section 4 we outline the technique for the block partitioning used, in Section 5 we describe the overall parallel algorithm of the solution method, Section 6 presents the numerical results obtained for a set of real-life large-scale computational problems, Section 7 contains some conclusive remarks, and in Appendix we consider in more detail the estimates for the K-condition number for the BIIC, IC2, and BIIC-IC2 preconditionings.

2. AN ALTERNATIVE THEORY OF THE PCG METHOD CONVERGENCE

Consider the PCG method [1] for solving a linear algebraic system (1) with an SPD sparse $n \times n$ matrix A

$$\begin{aligned} & r_0 = b - Ax_0, \quad p_0 = Hr_0; \quad \text{for } i = 0, 1, \dots \\ & \alpha_i = \frac{r_i^{\mathrm{T}} H r_i}{p_i^{\mathrm{T}} A p_i}, \quad x_{i+1} = x_i + p_i \alpha_i, \quad r_{i+1} = r_i - A p_i \alpha_i \\ & \beta_i = \frac{r_{i+1}^{\mathrm{T}} H r_{i+1}}{r_i^{\mathrm{T}} H r_i}, \quad p_{i+1} = H r_{i+1} + p_i \beta_i \end{aligned} \tag{2}$$

The standard upper bound for the number of PCG iterations i needed to satisfy the convergence criterion

$$r_i^{\mathrm{T}} A^{-1} r_i \leqslant \varepsilon^2 r_0^{\mathrm{T}} A^{-1} r_0$$

with $r_i = b - Ax_i$, is

$$i = i_C(\varepsilon) = \frac{1}{2}\sqrt{C} \log \frac{2}{\varepsilon} \tag{3}$$

where

$$C = C(HA) = \lambda_{\max}(HA)/\lambda_{\min}(HA) \tag{4}$$

is the spectral condition number of the preconditioned matrix HA.

It is well known that (3) provides only an upper bound for i_C which can be very pessimistic if $C(HA)$ is large but the smallest eigenvalues of HA are well isolated and their number is not very large. In References [3, 6] alternative estimates of the number of iterations has been derived, based on the so-called K-condition number. Namely, the number of PCG iterations i needed to satisfy the convergence criterion

$$r_i^{\mathrm{T}} H r_i \leqslant \varepsilon^2 r_0^{\mathrm{T}} H r_0$$

can be bounded from above as

$$i = i_K(\varepsilon) = \log_2 K + \log_2 \frac{1}{\varepsilon} \tag{5}$$

where

$$K = K(HA) = \left(\frac{1}{n}\ \mathrm{trace}(HA)\right)^n \Big/ \det(HA) \tag{6}$$

is the so-called K-condition number of the preconditioned matrix HA, see Reference [7]. The K-condition number is not very sensitive with respect to the smallest eigenvalues of HA and therefore often gives a more relevant preconditioning quality criterion as compared to the spectral condition number. A number of preconditioning strategies were analysed from the viewpoint of the K-condition number reduction, see References [2–4, 6]. The preconditioning discussed below is based on a proper combination of the block incomplete inverse Cholesky (BIIC) preconditioning [2, 3, 8] and the robust incomplete Cholesky second order (IC2) preconditioning [4].

3. BLOCK EXPLICIT PRECONDITIONER

Let us briefly describe an approximate version of the block incomplete inverse Cholesky (BIIC) preconditioning algorithm [2, 3, 8]. Let A be reordered and split in the same way as for the block Jacobi preconditioning, i.e. the tth diagonal block of the symmetrically reordered matrix has the dimension n_t and $n_1 + \cdots + n_s = n$. Here $t = 1, \ldots, s$, and s is the block dimension of A. For the tth diagonal block, let us define a 'basic' index set as $\{k_{t-1}+1, \ldots, k_t\}$, where $k_{t-1} = n_1 + \cdots + n_{t-1}$, $k_0 = 0$, $k_s = n$, and introduce 'overlapping' index sets as $\{j_t(1), \ldots, j_t(m_t - n_t)\}$, $j_t(p) \leqslant k_{t-1}$. For each t, the latter index set typically includes those indices not greater than k_t that are the most 'essentially' connected to the basic index set, e.g. in the sense of the sparse matrix graph adjacency relations, see Section 4. Here $m_t \geqslant n_t$ and, obviously, $m_1 = n_1$, i.e. at least the first overlapping set is empty. The BIIC-IC2 preconditioner H (which will be further referred to as the BIIC2 preconditioner) can be represented in the following additive form:

$$H(\tau) = \sum_{t=1}^{s} V_t U_t^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_t} \end{bmatrix} U_t^{-\mathrm{T}} V_t^{\mathrm{T}} \tag{7}$$

where V_t are rectangular matrices composed of unit n-vectors e_j as follows:

$$V_t = [e_{j_t(1)}|\dots|e_{j_t(m_t-n_t)}|e_{k_{t-1}+1}|\dots|e_{k_t}], \quad t=1,\dots,s$$

and each upper triangular matrix U_t is the (in general, approximate) right Cholesky factor constructed for the tth 'backward extended' principal $m_t \times m_t$ submatrix $V_t^{\mathrm{T}} A V_t$, that is,

$$V_t^{\mathrm{T}} A V_t = U_t^{\mathrm{T}} U_t + U_t^{\mathrm{T}} R_t(\tau) + R_t(\tau)^{\mathrm{T}} U_t \tag{8}$$

where $R_t(\tau)$ is a strictly upper triangular error matrix (i.e., containing only relatively small entries). The recurrences for the calculation of IC2 factorization can easily be obtained from (8), especially in the case when the sparsity patterns of U and R do not have coinciding non-zero positions and their non-zero elements are subjected to the conditions $|U_{i,j}| \geqslant \tau$ and $|R_{i,j}| < \tau$, respectively, $i<j$. Here $0<\tau\ll 1$ is some chosen drop tolerance parameter which determines the quality of the incomplete factorization. Note that with $\tau=0$ we get $R_t(\tau)=0$, so that the exact Cholesky factors are used in the construction of the preconditioner $H(0)$, and we obtain exactly the BIIC preconditioner from [2, 3, 8]. The existence and correctness of these IC2 decompositions are guaranteed for any SPD matrix. Some modifications of the IC2 decomposition involving less computational effort can be found in References [4, 9].

It is shown in References [2, 3, 8] that the BIIC preconditioner $H(0)$ possesses, in a certain sense, the K-optimality property. Moreover, using the properties of the IC2 and BIIC decompositions one can prove (see Appendix below) that

$$\log K(H(\tau)A) \leqslant \log K(H(0)A) + c_0\tau^2 \tag{9}$$

where c_0 is not very large even for ill-conditioned matrices. The latter estimate together with (5) shows that for some reasonably small dropping tolerance τ such BIIC2 hybrid construction will give nearly the same rate of the PCG convergence as for the K-optimal BIIC preconditioner $H(0)$. At the same time BIIC2 involves essentially smaller costs to construct, store and apply the preconditioner as compared to the 'exact' BIIC one.

Remark 3.1.
Unlike the classical overlapping DD preconditioners, the proposed one gives the exact representation of A^{-1} as the overlap increases to its natural limit (and, of course, with $\tau=0$). This is easily seen from the formula, cf. Reference [3],

$$H(0) = \sum_{t=1}^{s} V_t (V_t^{\mathrm{T}} A V_t)^{-1} V_t^{\mathrm{T}} - \sum_{t=2}^{s} W_t (W_t^{\mathrm{T}} A W_t)^{-1} W_t^{\mathrm{T}}$$

where

$$W_t = [e_{j_t(1)}|\dots|e_{j_t(m_t-n_t)}], \quad t=1,\dots,s$$

are the matrices determining the overlap structure. Thus, the proposed method is actually based on a completely new approach to the construction of block overlap preconditionings: instead of any sort of overlap weighting we add the *negative* semidefinite correction term.

For this reason, we would like to refrain from detailed comparison of the BIIC preconditioning with widely known techniques such as multisplitting methods, see Reference [10] for a review.

For instance, the unweighted overlapped block Jacobi preconditioner $\sum_{t=1}^{s} V_t(V_t^{\mathrm{T}} A V_t)^{-1} V_t^{\mathrm{T}}$ was tested by the authors earlier and its performance was found rather unsatisfactory (even less effective than the non-overlapped block Jacobi).

Therefore, when we refer our preconditioning to as the 'overlapping DD' one, we only want to indicate the formal structural similarity of (7) to known approaches.

4. DESCRIPTION OF THE MATRIX SPLITTING STRATEGY

The problem of matrix graph splitting is discussed in this section. We do not use any knowledge of the real topology of the domain, so the decomposition of the domain is performed using the sparse matrix graph only. Most of the ill-conditioned problems which are investigated in the present report are defined on very stretched domains, so because of the narrow bandwidth the simplest method for block splitting turns out to be quite successful.

The computational procedure is the following. First, a global bandwidth reduction is performed using the reversed Cuthill-McKee algorithm (RCM), then the node set is divided into s approximately equal 'subdomains' (or blocks), and finally the bandwidth inside each block is again reduced by RCM. The overlap is obtained by using sparsity structure of the qth degree of the coefficient matrix, i.e. the sparsity structure of A^q (see overlap size parameter q in Tables II and III as well as Figure 4).

More complicated matrix graph splitting techniques (as in the public-domain graph partitioning package METIS [11]) could provide a somewhat better splitting [12, 13]. The experiments with medium size benchmark problems demonstrate that with this technique it is possible to obtain 5–10% reduction of the total solution time. However, on the computer where all tests were performed, the use of METIS for the largest problems was not possible due to its excessive memory requirements.

5. PARALLEL IMPLEMENTATION

The above-described mathematical technique is implemented in a portable software with the use of message passing interface (MPI) library for communications between processes.

Let us assume that the linear system (1) is solved on a parallel distributed memory computer having N_{PE} processor elements (PEs). Let us also assume that $s = N_{\mathrm{PE}}$, i.e. the number of processors coincides with the number of blocks into which the original matrix A is split, and the tth block is mapped to the tth PE, $t = 1, \ldots, s$.

The following distribution of data is used. The tth processor stores the 'basic' vector components $k_{t-1} + 1, \ldots, k_t$ and the rows of the matrix A with the same numbers as well as the corresponding block of preconditioner together with the overlap.

The proposed algorithm can be implemented as follows. Let us assume that the linear system is already permuted in accordance with the matrix splitting described in Section 4. Perform the IC2 factorization [4] of the local submatrix $V_t^{\mathrm{T}} A V_t$ at the local tth PE. No data exchanges are required at this stage.

The PCG iterations stage involves the following types of operations:

1. multiplication of the coefficient matrix A by a vector,
2. multiplication of the preconditioner $\tilde{H}$ by a vector,
3. vector updates, and
4. inner products.

Matrix-by-vector product. The multiplication of a sparse matrix A by a vector is a well investigated problem. It can be presented as the following three-stage algorithm:

1. for any PE requiring some data which are local for the PE, place the corresponding components of the vector to a local data buffer and send them to PEs which require them;
2. receive the needed data from the other PEs;
3. multiply the local matrix coefficient data by the vector gathered.

The resulting components of the vector will be located at the PE which computes them.

Preconditioner by a vector product. The parallel implementation of the multiplication of the preconditioner by a vector is qualitatively similar to that of A and can be described analogously. The differences are the following:

1. the data exchange topology is based on the overlap geometry;
2. the type of operation with the local data is different (local triangular system solutions instead of the local matrix by vector product);
3. after the first global synchronization, two successive triangular system solutions with U_t^{T} and U_t are performed, and after these local computations a second global synchronization operation is required.

Vector updates. Three vector update operations do not use interprocessor communications, because all the vector components needed for a vector update are already placed at the local PE.

Inner products. Two inner products are required for each PCG iteration. To perform this operation, we call to MPI_AllReduce function from the MPI library.

Properties of the parallel algorithm.

1. The total number of global exchange operations does not depend on the number of blocks and the size of the linear system and is equal to five for each PCG iteration.
2. After completion of the global data exchange, all the computations can be performed on each PE without any additional synchronizations.
3. The computations are well balanced if the sizes of the local submatrices are approximately equal.
4. The communication costs are not large as compared to the arithmetic costs.

6. RESULTS OF THE NUMERICAL EXPERIMENTS

In this section, we present numerical results obtained on an eight-processor SUN Enterprise 3000 computer with UltraSparc CPU at 170 MHz. The size of core memory available was only 1 Gbyte, however, our implementation fitted well this memory size even for the largest problems.

The test problems come from structural mechanics and linear elasticity [14, 15]. There are two two-dimensional ('Dam' and 'Bridge') and four three-dimensional ('Brick', 'Soil', 'Detail', and 'Sluice') problems. For each of these real life problems a number of discrete linear systems has been provided, corresponding to few consecutive regular mesh refinement of the initial mesh. This gives us the opportunity to examine the scalability of the parallel solution method with respect to enlarging the problem size.

Table I. Matrix properties, tuned IC2 parameters, and PCG results.

Name/#	n	nz(A)	$C(A_S)$	density	t_{iter}	t_{total}	true res.
Bridge ($\tau = 3 \times 10^{-4}$, MD)							
1	1722	20937	3.99+6	2.42	0.07	0.21	3.34–10
2	6270	80727	2.74+7	3.16	0.71	1.53	4.31–09
3	23838	316755	1.38+8	3.93	5.70	12.1	9.75–09
4	92862	1254552	5.96+8	4.64	54.0	110.5	1.24–08
5	366462	4993117	2.46+9	5.02	413.7	852.7	5.43–08
Bridge ($\tau = 3 \times 10^{-4}$, PR)							
1	1722	20937	3.99+6	3.01	0.01	0.31	1.71–09
2	6270	80727	2.74+7	4.13	0.98	2.70	4.46–09
3	23838	316755	1.38+8	5.72	10.27	25.4	2.65–09
4	92862	1254552	5.96+8	7.18	105.1	227.7	1.41–08
5	366462	4993117	2.46+9	7.75	1620.6	2765.6	8.42–08
Dam ($\tau = 3 \times 10^{-3}$, PR)							
0	3474	45862	1.30+5	2.70	0.38	1.42	8.59–09
1	13474	182502	5.79+5	2.75	3.19	9.83	7.74–09
2	53058	729582	2.41+6	2.72	28.9	59.8	8.03–09
3	210562	2917328	9.79+6	2.71	243.9	374.2	6.69–09
4	838914	11667444	3.94+7	2.70	1991.2	2482.9	8.96–09
Brick ($\tau = 3 \times 10^{-2}$, PR)							
2	675	32785	6.06+2	0.53	0.10	0.47	8.80–09
3	4131	255415	2.45+3	0.44	2.04	4.74	6.15–09
4	28611	2016264	9.82+3	0.41	32.5	53.1	7.00–09
5	212355	16024772	3.93+4	0.40	514.6	656.5	9.90–09
Soil ($\tau = 3 \times 10^{-2}$, PR)							
0	375	15017	2.12+3	0.60	0.04	0.18	9.09–09
1	2187	122450	9.80+3	0.47	0.92	2.13	3.46–09
2	14739	986644	4.11+4	0.42	15.9	25.2	9.89–09
3	107811	7925485	1.66+5	0.40	279.0	351.1	9.79–09
Detail ($\tau = 2 \times 10^{-2}$, PR)							
0	11484	749446	1.09+8	0.57	64.1	73.6	7.57–09
1	79065	5726430	5.07+8	0.51	1260.2	1352.3	1.09–08
2	584127	44668305	2.15+9	0.48	21801.7	22566.3	3.09–08
Sluice ($\tau = 2 \times 10^{-2}$, PR)							
0	2172	102114	7.52+3	0.69	0.89	1.78	7.79–09
1	12957	775743	4.06+4	0.54	19.1	27.4	5.83–09
2	87159	6007266	1.92+5	0.47	372.4	442.6	8.33–09
3	633579	47193500	8.56+5	0.43	6719.7	7295.5	9.62–09

In Table I we present the numerical results for serial IC2 preconditioning, which are the best obtained ones with respect to the total solution time for a single right-hand side. In the first three columns we show the order of the system, the total number of non-zero elements in the matrix A, and the spectral condition number of A_S. Here, $A_S = D_A^{-1/2} A D_A^{-1/2}$ represents the original matrix symmetrically scaled by its main diagonal. The minimal and maximal eigenvalues of A_S were computed from the coefficients α_i and β_i obtained in the CG method applied to the unpreconditioned system with the matrix A_S. Namely, a tridiagonal matrix was constructed in a standard way, cf. Section 11.3.4 in Reference [7], and its extremum eigenvalues were used as the approximations to the desired ones. The next column 'density' shows the quantity $\mathrm{nz}(U)/\mathrm{nz}(D_A + U_A)$ which indicates the (relative) memory volume needed for the preconditioning, and in the next column the PCG iteration time and the total solution time in seconds are given (the latter includes reordering, IC2 factorization, and PCG iterations). The iterations were started with zero initial guess $x_0 = 0$, so that $r_0 = b$, and performed until the stopping criterion

$$\|r_i\| \leqslant \varepsilon \|r_0\|, \quad \varepsilon = 10^{-8}$$

is satisfied. The actual value of the relative residual norm $\|b - Ax_k\|/\|b\|$ is given in the last column as 'true res.'. The IC2 tuning options are given in brackets next to the name of the problem set, where the value of the threshold parameter is given, as well as the type of preordering is indicated: 'MD' for the minimum degree preordering [16], and 'PR' for the profile/bandwidth reduction reordering applied to the coefficient matrix A and performed with the use of some Cuthill–McKee type procedure.

The results of numerical experiments for the parallel BIIC2 solver for 8 PEs used are presented in Table II. The BIIC2 tuning options are given in brackets next to the name of the problem set, where the value of the threshold parameter τ as well as the overlap size parameter q are given. Note that the value of τ is taken the same as for the sequential IC2 algorithm. The column 'iter.' indicates the number of PCG iterations performed. In the next columns the wall clock time in seconds for both the PCG iterations and the total solution are given (the latter includes the construction of the BIIC2 preconditioner and the PCG iterations, but does not include the additional time for scaling, reordering, and computation of the final true unscaled residual that is usually not greater than 1 per cent of the total solution time). It should be noted that the wall clock time on one processor is always greater than the pure central processor time usually reported for the sequential version of a solver. The column S_{eff} presents the efficient speedup calculated by the formula $S_{\text{eff}} = N_{\text{PE}}(t_{\text{total}} - t_{\text{mpi}})/t_{\text{total}}$, where t_{mpi} is the time spent for mpi data exchanges, global synchronizations and all other overheads (delays) in computations. To compute t_{mpi} we sum the astronomical time spent for all calls to mpi functions, including initialization of data exchanges (MPI_IRecv and MPI_ISend), waiting of data receiving (MPI_WaitAll), operations of global sum (MPI_AllReduce) and global synchronization (MPI_Barrier). The columns S_1 presents the so-called relative speedup, which is the ratio t_1/t_{PE}, thus the time for the parallel execution on N_{PE} processors, divided by the time for the same program executed on one processor. The column S_{IC2} is a measure of the amount of redundancy caused by the parallel implementation and is computed as $t_{\text{IC2}}/t_{\text{PE}}$, where t_{IC2} is the central processor serial solution time of the IC2 solver (especially tuned for one PE) with the same strategy of profile reduction ordering (PR), which is taken from Table I. It should be noted that the efficient speedup is the only one which can be estimated directly during the parallel run with no information on the run on one PE. The value of the efficient speedup is usually greater than that of the relative and the absolute one.

The total wall clock solution time for each set of benchmark problems is also presented in Figure 1 in a logarithmic scale. From this figure, we can see that the total solution time

Table II. Tuned BIIC2 parameters and PCG results for $N_{\mathrm{PE}} = 8$.

Name/#	n	density	iter.	t_{iter}	t_{total}	S_{eff}	S_1	S_{IC2}	true res.
Bridge ($\tau = 3\times10^{-4}$, $q = 40$)									
1	1722	10.84	14	0.10	0.11	4.53	2.09	2.81	6.69–9
2	6270	9.98	27	0.65	0.72	5.27	2.86	3.75	4.31–9
3	23838	9.43	62	5.73	6.48	5.82	3.17	3.91	9.75–9
4	92862	8.54	152	45.93	53.93	6.46	3.92	4.22	1.24–8
5	366462	7.92	378	421.90	534.33	6.76	6.04	5.17	5.43–8
Dam ($\tau = 3\times10^{-3}$, $q = 10$)									
0	3474	5.19	32	0.25	0.27	4.67	4.22	5.25	8.59–9
1	13474	4.13	56	1.39	1.59	5.91	5.49	6.18	7.74–9
2	53058	3.40	109	9.66	11.45	6.54	5.21	5.22	8.03–9
3	210562	3.05	214	64.70	76.01	7.20	5.52	4.92	6.69–9
4	838914	2.86	425	495.38	577.27	7.41	5.96	4.30	8.96–9
Brick ($\tau = 3\times10^{-2}$, $q = 1$)									
2	675	1.04	35	0.08	0.08	3.25	3.62	5.87	8.80–9
3	4131	0.68	73	0.79	0.80	4.62	5.11	5.92	6.15–9
4	28611	0.53	153	10.74	10.93	6.44	5.01	4.85	7.00–9
5	212355	0.46	290	148.40	151.15	7.08	5.26	4.34	9.90–9
Soil ($\tau = 3\times10^{-2}$, $q = 3$)									
0	375	2.16	22	0.05	0.05	3.05	2.40	3.60	9.09–9
1	2187	1.54	50	0.40	0.41	4.19	4.34	5.19	3.46–9
2	14739	1.03	118	5.83	5.90	5.91	4.50	4.27	9.89–9
3	107811	0.71	258	80.42	81.71	6.77	4.99	4.29	9.79–9
Detail ($\tau = 2\times10^{-2}$, $q = 1$)									
0	11484	0.60	514	10.93	11.31	6.98	7.76	6.50	7.57–9
1	79065	0.52	1415	234.49	240.02	7.35	6.76	5.63	1.09–8
2	584127	0.48	3286	4355.20	4482.18	7.46	6.65	5.03	2.86–8
Sluice ($\tau = 2\times10^{-2}$, $q = 1$)									
0	2172	0.72	81	0.30	0.30	4.35	4.70	5.93	7.79–9
1	12957	0.57	183	4.59	4.65	6.12	5.84	5.89	5.83–9
2	87159	0.48	551	98.60	100.92	7.12	5.33	4.38	8.33–9
3	633579	0.44	1137	1666.90	1782.42	7.42	5.61	3.76	9.55–9

reduces when the number of processors increases even for the smallest problems with $n \approx 1000$ and when the total solution time is less than 1 s. As a rule the larger the problem is, the better the parallel properties of the BIIC2 solver become.

Table III presents results for BIIC2 parameters tuning for the largest problem ‘Dam’ #4 ($n = 838914$). The column ‘overlap’ shows the value $\sum_{t=1}^{s} m_t/n$ specifying the total size of the overlap. The table is split into four parts. The first part shows the behaviour of the BIIC2 method for increasing number of blocks. It is seen that the preconditioning quality is dependent only slightly on the number of blocks (equal to the number of PEs) which provides a very good scalability of the method. The second part of the table shows the dependence of the

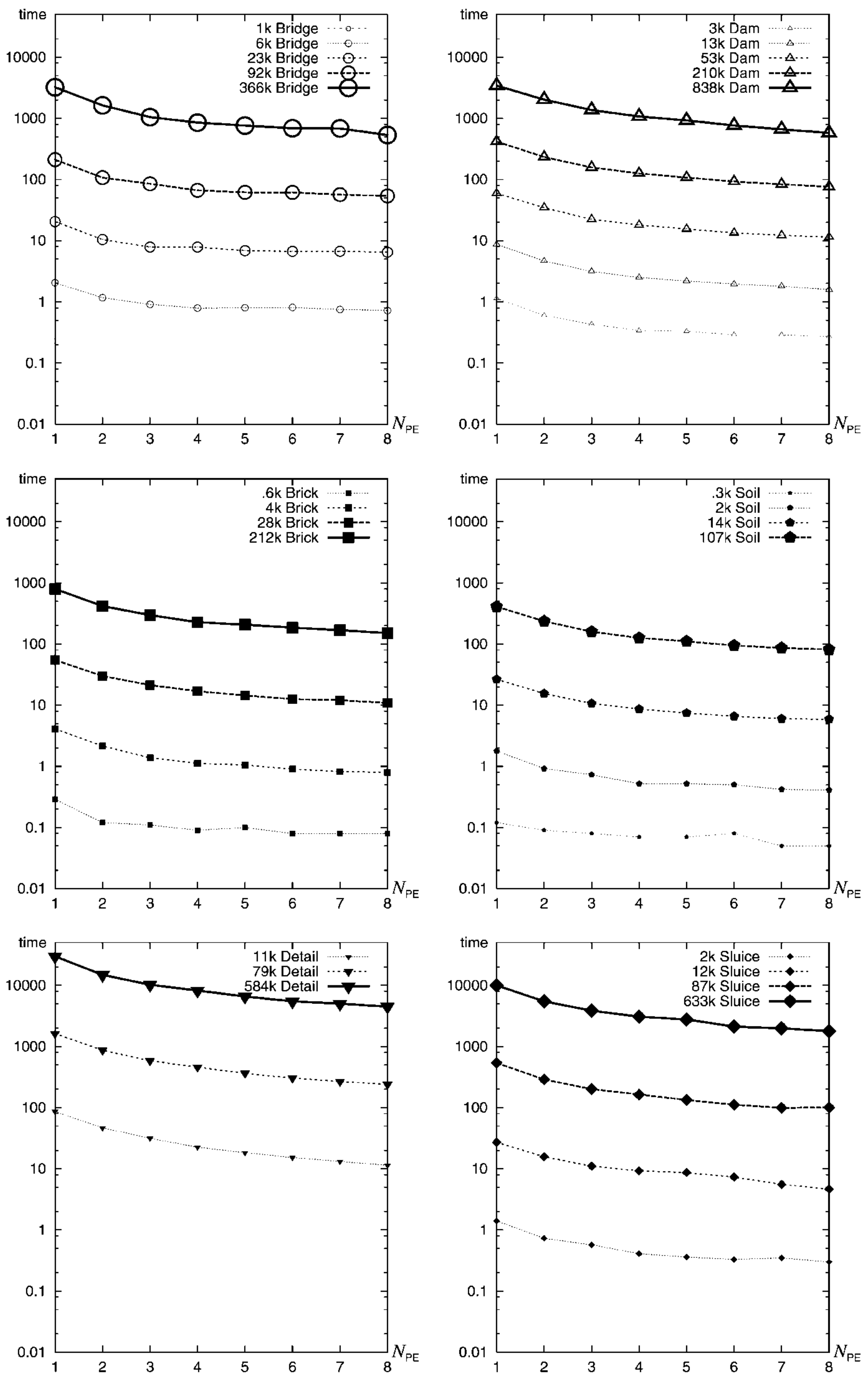


Figure 1. Total wall clock solution time in seconds for each set of benchmark problems.

Table III. The results of BIIC2 parameters tuning for problem Dam #4 ($n = 838914$), $s = N_{PE}$.

s	q	τ	density	overlap	$C(\tilde{H}A)$	iter.	t_{total}		Remarks
1	10	3.0-3	2.70	0.00	5.38 + 4	357	3441.36		
2	10	3.0-3	2.76	0.03	5.04 + 4	390	2023.69		
3	10	3.0-3	2.72	0.03	4.43 + 4	398	1371.41		
4	10	3.0-3	2.78	0.05	5.43 + 4	406	1075.14		
5	10	3.0-3	2.80	0.06	5.62 + 4	424	928.49		
6	10	3.0-3	2.82	0.07	5.52 + 4	421	766.40		
7	10	3.0-3	2.84	0.08	5.56 + 4	425	663.27		
8	10	3.0-3	2.86	0.10	5.86 + 4	425	577.27	$^{(*)}$	
8	0	3.0-3	2.56	0.00	1.00 + 5	771	903.87		
8	2	3.0-3	2.61	0.02	7.16 + 4	552	696.82		
8	4	3.0-3	2.68	0.04	6.56 + 4	491	634.88		
8	6	3.0-3	2.74	0.06	6.24 + 4	464	624.95		
8	8	3.0-3	2.80	0.08	6.01 + 4	441	613.44		
8	10	3.0-3	2.86	0.10	5.86 + 4	425	577.27	$^{(*)}$	
8	12	3.0-3	2.91	0.12	5.79 + 4	415	626.14		
8	14	3.0-3	2.96	0.14	5.70 + 4	406	612.71		
8	16	3.0-3	3.01	0.16	5.64 + 4	402	631.98		
8	18	3.0-3	3.07	0.17	5.59 + 4	400	629.71		
8	20	3.0-3	3.12	0.19	5.56 + 4	398	647.33		
8	10	1.0-3	4.50	0.10	2.16 + 4	285	690.29		
8	10	2.0-3	3.38	0.10	4.02 + 4	364	661.85		
8	10	3.0-3	2.86	0.10	5.86 + 4	425	577.27	$^{(*)}$	
8	10	4.0-3	2.54	0.10	7.62 + 4	482	690.66		
8	10	5.0-3	2.33	0.10	9.26 + 4	529	713.73		
8	0	3.0-3	0.00	0.00	3.94 + 7	11198	4182.84		PJ
8	0	3.0-3	2.56	0.00	1.00 + 5	771	956.42		BJ
8	10	3.0-3	3.18	0.19	1.45 + 5	642	1091.38		OBJ
8	10	3.0-3	2.86	0.10	5.86 + 4	425	577.27	$^{(*)}$	BIIC2

BIIC2 performance on the overlap size parameter q. It is seen that this parameter must only be not too small (say, $q \geqslant 6$) in order to assure good performance, while its further increase does not change the total solution time essentially. The results of tuning the overlap parameter q for some other problems are also presented in Figure 2.

The third part of the table shows that the dependence of the total solution time on the dropping threshold parameter τ is not very crucial: over 200 per cent variations change t_{total} only by less than 20%. Additional numerical results for some large problems are presented in Figure 3. The fourth part of the table demonstrates the comparison results with some other methods. Here PJ stands for point Jacobi preconditioning, BJ and OBJ stand for the block Jacobi and the overlapped block Jacobi preconditioning, respectively, with IC2 subdomain solvers. The best result in each series is marked by '$^{(*)}$'. These results demonstrate the advantages of the BIIC2 preconditioning as compared with the other methods having approximately the same parallel properties. For comparison of different methods for the same problem 'Dam' #4 see also Figure 4.

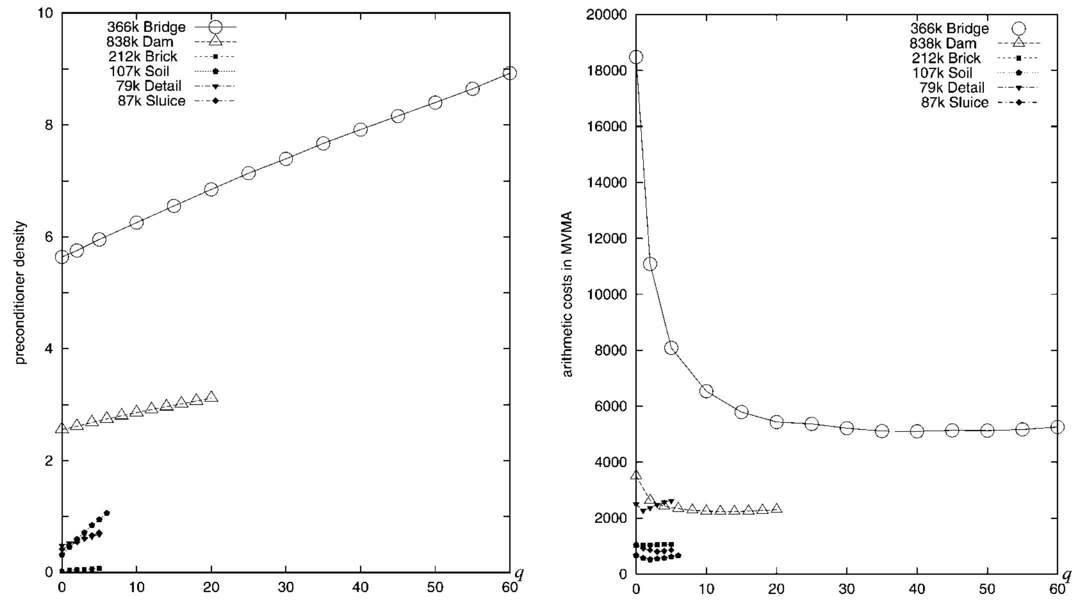


Figure 2. Tuning overlap parameter q for some large problems ($s = 8$).

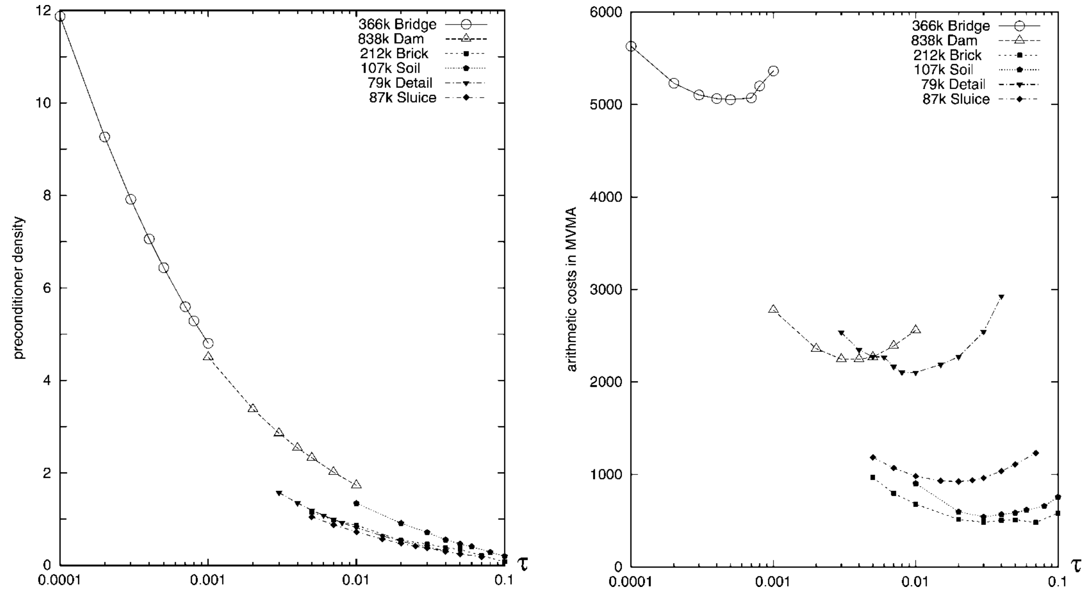


Figure 3. Tuning threshold parameter τ for some large problems ($s = 8$).

An important feature of the above-described algorithm observed in the course of numerical testing is that its total arithmetic cost grows quite slowly with the increase of the number of subdomains (equal to the number of PEs). This is not the case for the (approximate) block

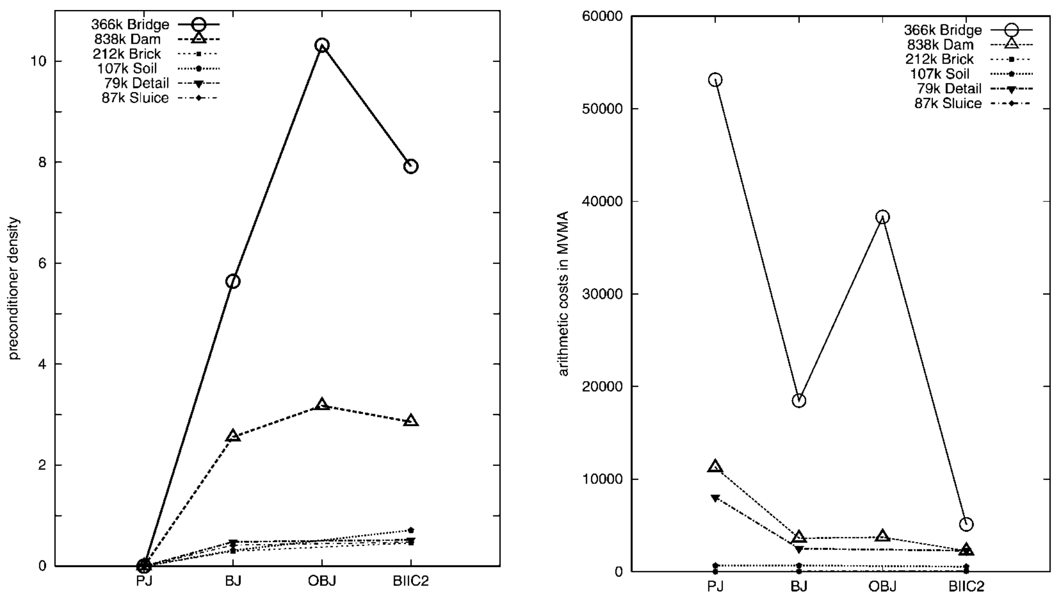


Figure 4. Comparison of different methods for $s = N_{\mathrm{PE}} = 8$.

Jacobi preconditioned CG method, where the number of iterations grows rapidly with the number of subdomains (see, for example, Figure 2a for 'Dam' #4 and 'Bridge' #5 problems with $q = 0$, which correspond to the above-mentioned BJ method).

It is very important that the number of processors used can be further increased up to 64 or more, because the total arithmetic costs increase not too much when the number of blocks increases (see Figure 5).

Communication costs for BIIC2 solver for the largest problem in each set and for all 'Dam' problems are presented in Figure 6. These data demonstrate that the communication costs are not large as compared to the arithmetic costs.

The value of relative speedup for the largest problem in each set and for all problems in the Dam set are depicted in Figure 7. The presented results show that larger the problem is, the greater the achieved speedup becomes. The data presented in Figure 7a shows that the proposed method achieve almost ideal speedup for the largest problem in each set independently of the origin and the properties of the problem solved.

It should be noted that for problem 'Bridge' #5 on 2 and 3 PEs we obtain even superlinear speedup. This is due to purely algebraic reason of reduction of the total arithmetic costs for these number of blocks $s = 2, 3$ (see Figure 5b). This effect was already observed for the test problem BIHAR255 [12, 13].

7. CONCLUSIONS

In this paper, the parallel implementation of the BIIC2 preconditioned CG method is described and tested. Both theoretical and experimental results show a remarkable efficiency of the

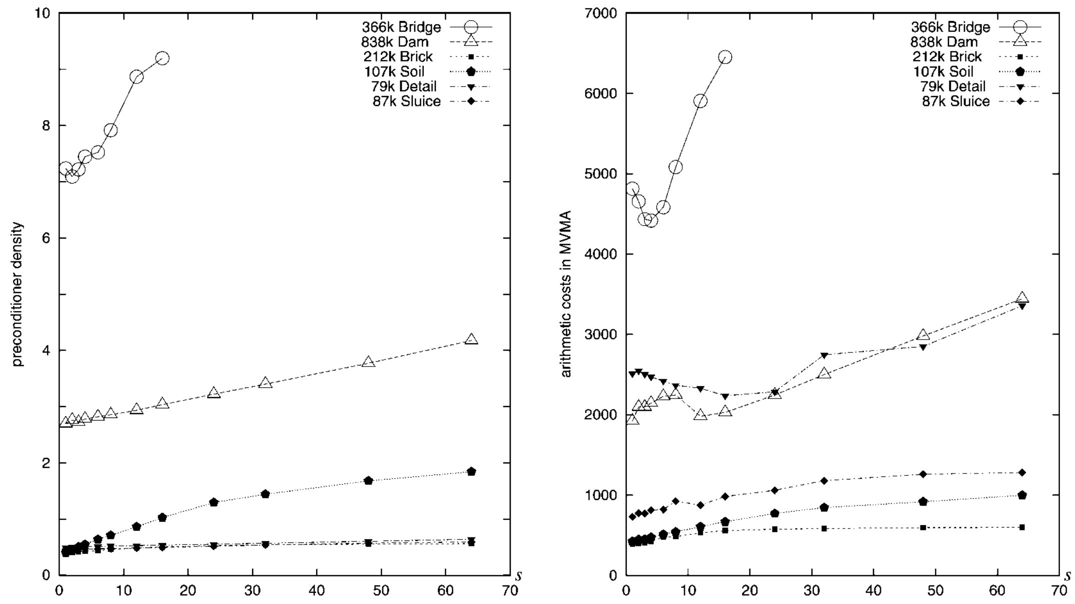


Figure 5. Tuning number of blocks s for some large problems.

proposed preconditioning for the cases when the coefficient matrix of the linear system is SPD but not an M-matrix and has very large condition number. Due to the relatively small portion of parallel overhead in BIIC2-CG algorithm, the increase of speedup is clearly observed when increasing the problem size. Despite a rather high computational cost needed by this preconditioning, the gain obtained from the considerable reduction in the total costs of PCG iterations far outweighs this drawback, in particular when solving a set of linear equations with the same matrix but with many different right-hand sides. The parallel efficiency of the solver for multiple right-hand sides will also increase due to the relative reduction of the number of global synchronizations.

The numerical results presented in Table II demonstrate a reasonable speedup in comparison with the single processor runs of both BIIC2 and IC2 algorithms. The larger the size of the problem is, the greater is the speedup obtained. The absence of slowdown even for the smallest problems shows the very efficient usage of the computer resources in parallel mode.

The experience accumulated by the authors (more than 1000 test runs, only a small portion of which is presented above) shows that the harder the problem is, the greater the gain in performance of the proposed method is as compared to other commonly used parallel iterative solvers. The parallel properties of the solver appear to be as good as expected from our theoretical considerations.

APPENDIX

Here we consider in more details the estimates for the K-condition number for the BIIC, IC2, and BIIC-IC2 preconditionings.

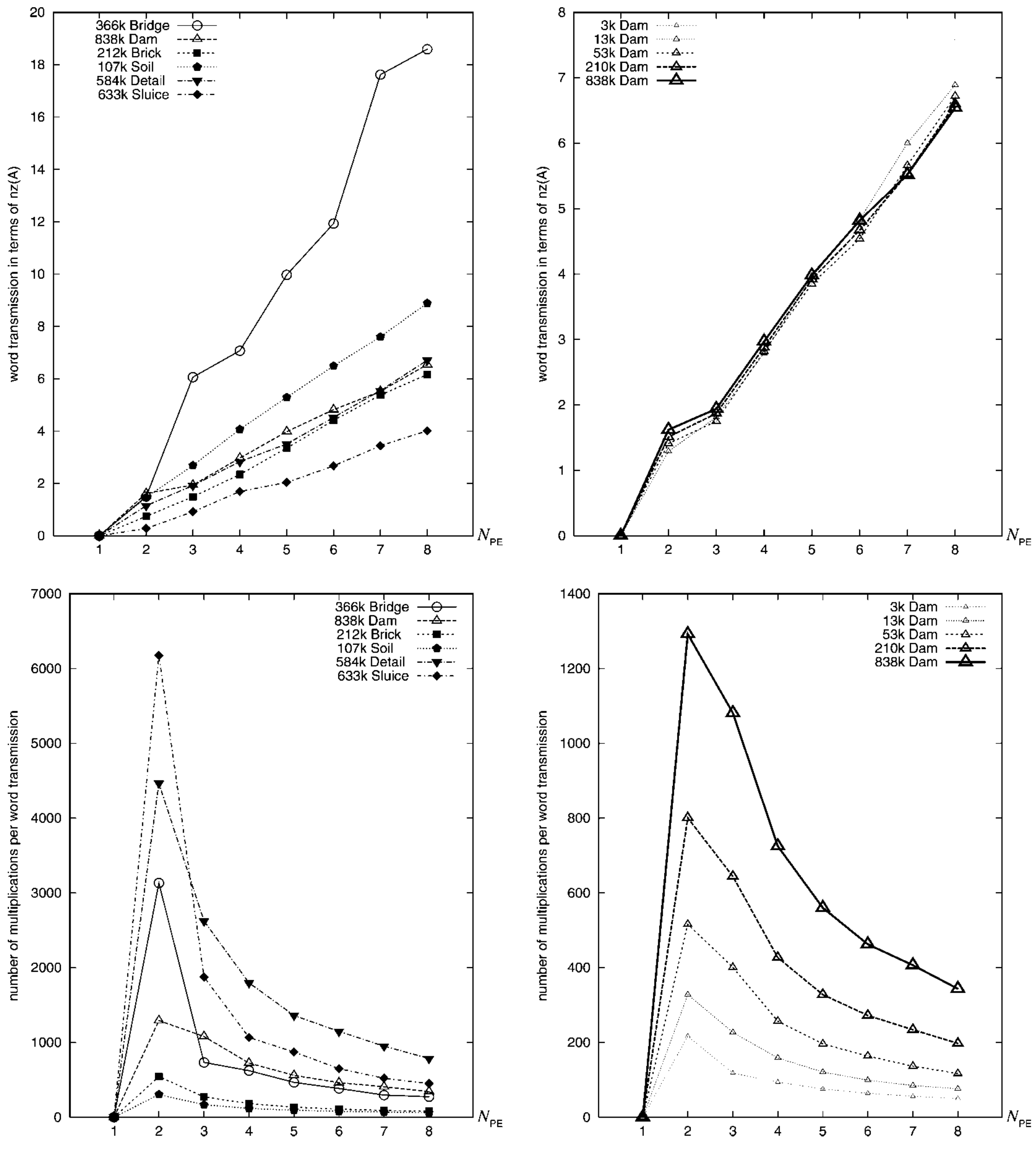
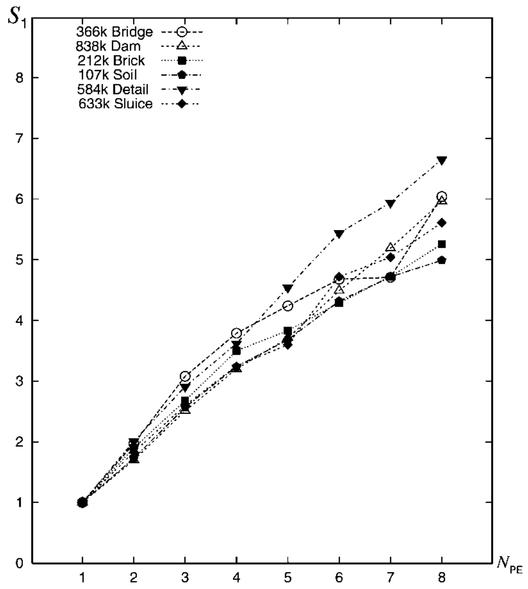


Figure 6. Communication costs of BIIC2 solver for the largest problem in each set (left) and for all problems in the Dam set (right).

1. *Pointwise incomplete inverse Cholesky factorization.* Let us recall the preconditioning techniques first introduced in Reference [2]. Let the preconditioner H be presented in the factorized form

$$H = G^{\mathrm{T}} G$$



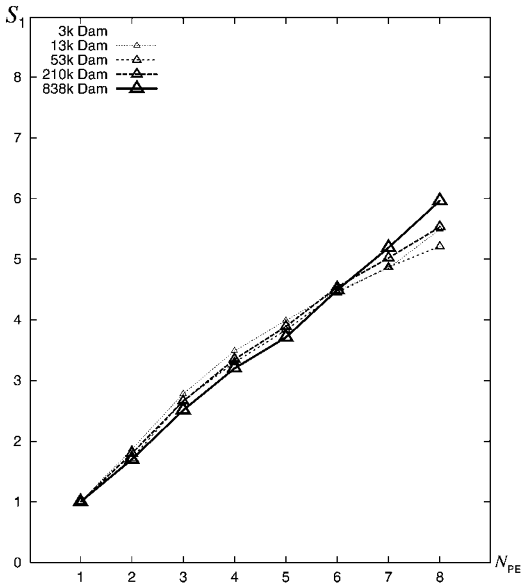


Figure 7. Relative speedup of BIIC2 solver for the largest problem in each set (left) and for all problems in the Dam set (right).

where G is a non-singular lower triangular matrix with prescribed sparsity pattern. Let the structurally non-zero elements of the ith row of G be

$$(G)_{i,j_i(1)}, \ldots, (G)_{i,j_i(m_i)}, \quad 1 \leqslant i \leqslant n$$

where

$$1 \leqslant j_i(1) < \cdots < j_i(m_i) = i$$

Then the minimum of $K(HA)$ is attained with

$$(G)_{i,j_i(p)} = \frac{(S_i^{-1})_{m_i,p}}{\sqrt{(S_i^{-1})_{m_i,m_i}}}, \quad 1 \leqslant p \leqslant m_i$$

where S_i is the $m_i \times m_i$ submatrix of A associated with ith subset of row and column indices, that is,

$$(S_i)_{p,q} = (A)_{j_i(p),j_i(q)}, \quad 1 \leqslant p \leqslant m_i, \quad 1 \leqslant q \leqslant m_i$$

Such preconditioning is referred to as K-optimum one (over the set of matrices H with prescribed structure of triangular factors).

An important reformulation of this result is as follows. Let

$$S_i = U_i^{\mathrm{T}} U_i$$

be the Cholesky factorization of S_i, then

$$(G)_{i,j_i(p)} = (U_i^{-\mathrm{T}})_{m_i,p}, \quad 1 \leqslant p \leqslant m_i$$

2. *Block incomplete inverse Cholesky factorization*. The block incomplete inverse Cholesky (BIIC) preconditioning was described in References [2, 3].

The idea underlying BIIC can be explained as follows. Suppose that the sparsity structure of the $(i+1)$th row of G is *the same* as that of ith one (with the $(i+1)$th diagonal non-zero position added, of course). In this case one obviously has

$$S_{i+1} = \begin{bmatrix} S_i & * \\ * & * \end{bmatrix}$$

Hence, if the Cholesky decomposition $S_i = U_i^{\mathrm{T}} U_i$ is available, then

$$S_{i+1} = U_{i+1}^{\mathrm{T}} U_{i+1} = \begin{bmatrix} U_i^{\mathrm{T}} & 0 \\ * & * \end{bmatrix} \begin{bmatrix} U_i & * \\ 0^{\mathrm{T}} & * \end{bmatrix}$$

and, finally,

$$U_{i+1}^{-\mathrm{T}} = \begin{bmatrix} U_i^{-\mathrm{T}} & 0 \\ * & * \end{bmatrix}$$

This means that it is not necessary to perform any separate calculations in order to obtain the non-zero elements of the ith row of G, it only suffices to evaluate the inverse for the left Cholesky factor for S_{i+1} and then use its two last rows as the non-zero elements of the ith and $(i+1)$th rows of G.

In the general case, the sparsity pattern for the lower triangular block incomplete inverse Cholesky factor G is determined blockwise. Let G have dense lower triangular blocks on the main diagonal and let the block off-diagonal non-zero positions be the same for each row of a block. Therefore, if the matrix G is divided into block rows as

$$G = \begin{bmatrix} G_1^{\mathrm{T}} \\ \cdots \\ G_s^{\mathrm{T}} \end{bmatrix}$$

where every $n_t \times n$ block G_t^{T} contains rows of G from the $(k_{t-1}+1)$th to the k_tth, and $0 = k_0 < k_1 < \cdots < k_s = n$, then the non-zero column indices of the non-zero elements of G for these rows are

$$\begin{array}{lll} j_t(1), \ldots, j_t(m_t - n_t), & k_{t-1}+1 & \\ \cdots & \cdots & \cdots \\ j_t(1), \ldots, j_t(m_t - n_t), & k_{t-1}+1, \ldots, k_t & \end{array}$$

Here $n_t = k_t - k_{t-1}$ is the size of the tth block and $m_t - n_t$ is the overlap size, which is equal to the number of the off-blockdiagonal non-zero columns in the tth block, $t = 1, \ldots, s$. Within this sparsity structure it is possible to use *the same Cholesky factorization data for each row of a block*. In this case, one can verify that the above basic representation of non-zero elements of the matrix G can be rewritten as

$$(G_t^{\mathrm{T}})_{i, j_t(p)} = ([0 \ldots 0 \;\; I_{n_t}] U_t^{-\mathrm{T}})_{i,p}, \quad 1 \leqslant i \leqslant n_t, \quad p = 1, \ldots, m_t$$

where

$$V_t^{\mathrm{T}} A V_t = U_t^{\mathrm{T}} U_t$$

is the Cholesky factorization of the corresponding principal submatrix of A. Now the key observation leading to the BIIC algorithm is that every leading submatrix of U_t is the right Cholesky factor for the corresponding leading submatrix of $V_t^{\mathrm{T}} A V_t$. Introducing the $n \times n_t$ matrix

$$E_t = [e_{k_{t-1}+1} | \cdots | e_{k_t}], \quad t = 1, \ldots, s$$

one can readily show that

$$G_t^{\mathrm{T}} \equiv E_t^{\mathrm{T}} G = E_t^{\mathrm{T}} V_t U_t^{-\mathrm{T}} V_t^{\mathrm{T}}$$

and therefore

$$H = G^{\mathrm{T}} G = \sum_{t=1}^{s} G_t G_t^{\mathrm{T}} = \sum_{t=1}^{s} V_t U_t^{-1} (V_t^{\mathrm{T}} E_t E_t^{\mathrm{T}} V_t) U_t^{-\mathrm{T}} V_t^{\mathrm{T}}$$

Noting that $E_t^{\mathrm{T}} V_t = [O_t | I_{n_t}]$, where O_t is an $n_t \times (m_t - n_t)$ zero matrix, and I_{n_t} is an $n_t \times n_t$ identity matrix, we obtain

$$V_t^{\mathrm{T}} E_t E_t^{\mathrm{T}} V_t = \begin{bmatrix} O_{n_t - m_t} & O_t^{\mathrm{T}} \\ O_t & I_{n_t} \end{bmatrix}$$

Substituting this representation into the above formula for H one readily obtains the additive form of the BIIC preconditioning similar to the given above in Section 3.

3. *K-condition number estimates for BIIC-IC2 preconditioned matrices*. The important properties of the IIC preconditioning are

$$(GAG^{\mathrm{T}})_{ii} = 1$$

and, by the definition of the K-condition number,

$$K(HA) \equiv K(GAG^{\mathrm{T}}) = (\det A)^{-1} \prod_{i=1}^{n} (G)_{ii}^{-2}$$

In the case of BIIC preconditioning these properties take the form

$$E_t^{\mathrm{T}} GAG^{\mathrm{T}} E_t = I_{n_t}$$

and

$$K(HA) \equiv K(GAG^{\mathrm{T}}) = (\det A)^{-1} \prod_{t=1}^{s} \left(\prod_{i=k_{t-1}+1}^{k_t} (G)_{ii}^{-2} \right)$$

$$= (\det A)^{-1} \prod_{t=1}^{s} \left(\prod_{i=m_t - n_t + 1}^{m_t} (U_t)_{ii}^{2} \right)$$

respectively.

The next step in constricting the BIIC-IC2 preconditioning is to replace the exact Cholesky factors U_t by their inexact counterparts $\tilde{U}_t$ constructed as was proposed in Reference [4],

$$V_t^{\mathrm{T}} A V_t = \tilde{U}_t^{\mathrm{T}} \tilde{U}_t + \tilde{U}_t^{\mathrm{T}} R_t + R_t^{\mathrm{T}} \tilde{U}_t$$

where R_t is an error matrix with a relatively small norm. (In order to clarify the exposition, here we use the simplest possible version of the IC2 decomposition.) The corresponding BIIC-IC2 preconditioner takes the form

$$\tilde{H} = \sum_{t=1}^{s} V_t \tilde{U}_t^{-1} \begin{bmatrix} 0 & 0 \\ 0 & I_{n_t} \end{bmatrix} \tilde{U}_t^{-\mathrm{T}} V_t^{\mathrm{T}}$$

and, as above, one can readily check up that

$$\tilde{H} = \tilde{G}^{\mathrm{T}} \tilde{G}$$

where

$$\tilde{G} = \begin{bmatrix} \tilde{G}_1^{\mathrm{T}} \\ \cdots \\ \tilde{G}_s^{\mathrm{T}} \end{bmatrix}$$

with

$$\tilde{G}_t^{\mathrm{T}} \equiv E_t^{\mathrm{T}} \tilde{G} = E_t^{\mathrm{T}} V_t \tilde{U}_t^{-\mathrm{T}} V_t^{\mathrm{T}}, \quad t = 1, \ldots, s$$

First of all, one should recall one of the main properties of the IC2 decomposition

$$\mathrm{Diag}(\tilde{U}_t^{-\mathrm{T}} (V_t^{\mathrm{T}} A V_t) \tilde{U}_t^{-1}) = I_{m_t}$$

In view of the above definition of $\tilde{G}$, the latter readily yields that

$$\mathrm{Diag}(E_t^{\mathrm{T}} (\tilde{G} A \tilde{G}^{\mathrm{T}}) E_t) = I_{n_t}$$

Hence, one has trace $\tilde{H}A = n$, and therefore

$$K(\tilde{H}A) \equiv K(\tilde{G} A \tilde{G}^{\mathrm{T}}) = (\det A)^{-1} \prod_{t=1}^{s} \left(\prod_{i=k_{t-1}+1}^{k_t} (\tilde{G})_{ii}^{-2} \right)$$

$$= (\det A)^{-1} \prod_{t=1}^{s} \left(\prod_{i=m_t-n_t+1}^{m_t} (\tilde{U}_t)_{ii}^{2} \right)$$

The second important property of the IC2 preconditioning is

$$\det(V_t^{\mathrm{T}} A V_t + R_t^{\mathrm{T}} R_t) = \det(\tilde{U}_t^{\mathrm{T}} \tilde{U}_t)$$

which readily yields

$$\det(I_{m_t} + R_t (V_t^{\mathrm{T}} A V_t)^{-1} R_t^{\mathrm{T}}) = (\det \tilde{U}_t)^2 (\det U_t)^{-2} = \prod_{i=1}^{m_t} \frac{(\tilde{U}_t)_{i,i}^2}{(U_t)_{i,i}^2}$$

Using now the above expressions for $K(\tilde{H}A)$ and $K(HA)$, we obtain

$$\frac{K(\tilde{H}A)}{K(HA)} = \prod_{t=1}^{s} \left(\prod_{i=m_t-n_t+1}^{m_t} \frac{(\tilde{U}_t)_{i,i}^2}{(U_t)_{i,i}^2} \right)$$

$$= \prod_{t=1}^{s} \left(\prod_{i=1}^{m_t-n_t} \frac{(U_t)_{i,i}^2}{(\tilde{U}_t)_{i,i}^2} \right) \det(I_{m_t} + R_t(V_t^{\mathrm{T}} A V_t)^{-1} R_t^{\mathrm{T}})$$

$$\leqslant \prod_{t=1}^{s} \det(I_{m_t} + R_t(V_t^{\mathrm{T}} A V_t)^{-1} R_t^{\mathrm{T}})$$

where the latter inequality readily follows from the properties of IC2 factorization of the leading $(m_t - n_t) \times (m_t - n_t)$ submatrix of $V_t^{\mathrm{T}} A V_t$ (in view that it is obviously expressed via the leading $(m_t - n_t) \times (m_t - n_t)$ submatrices of $\tilde{U}_t$ and R_t).

Finally, if each non-zero element of R_t is not larger than τ in absolute value, then the latter equality readily yield the estimate

$$\log K(\tilde{H}A) \leqslant \log K(HA) + c_0 \tau^2$$

with

$$c_0 = \sum_{t=1}^{s} \frac{\mathrm{nz}(R_t)}{\lambda_{\min}(V_t^{\mathrm{T}} A V_t)}$$

Our analysis can be completed using the (rather rough) bound

$$\mathrm{nz}(R_t) \leqslant \mathrm{nz}(U_t)$$

and therefore the desired estimate $\log K(\tilde{H}A) = \log K(HA) + O(\tau^2)$ is obtained.

4. *On the choice of overlap in the BIIC preconditioning*. Let us consider here, for simplicity, the case $s = 2$ (the conclusions in the general case appear to be essentially the same)

$$A = \begin{bmatrix} A_{00} & A_{01} & A_{02} \\ A_{10} & A_{11} & A_{12} \\ A_{20} & A_{21} & A_{22} \end{bmatrix}$$

where the 'basic' diagonal blocks are taken as

$$\begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \quad \text{and} \quad A_{22}$$

while the second block including the overlap is

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

Here we supposed that the matrix is ordered in such a way that the overlapping positions are placed the last in the first block. The corresponding BIIC preconditioner has the form

$$H = \begin{bmatrix} \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}^{-1} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & A_{11}^{-1} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

and the corresponding K-condition number of the preconditioned matrix can be expressed as

$$K(HA) = (\det A)^{-1} \det \begin{bmatrix} A_{00} & A_{01} \\ A_{10} & A_{11} \end{bmatrix} \det(A_{22} - A_{21}A_{11}^{-1}A_{12})$$

Since only the term $\det(A_{22} - A_{21}A_{11}^{-1}A_{12})$ depends on the choice of the overlap, we conclude that, at least, the norm of $A_{12} = A_{21}^{\mathrm{T}}$ should be as large as possible, which reassures our claim that the overlapped positions should be 'strongly connected' to the corresponding diagonal block A_{22}.

ACKNOWLEDGEMENTS

The authors are very grateful to Prof. Owe Axelsson for his kind attention to this research during several last years. The authors would also like to acknowledge the positive input from numerous discussions of the related algorithms and test problems with Dr. Maya Neytcheva who actually provided the generation of the test problem data.

This work has been supported by The Netherlands Technology Foundation (STW) grant NNS.4683 and by The National Science Foundation grant NWO 047.008.007.

REFERENCES

1. Axelsson O. A class of iterative methods for finite element equations. *Computer Methods in Applied Mechanics and Engineering* 1976; **9**:123–137.
2. Kaporin IE. On preconditioned conjugate-gradient method for solving discrete analogs of differential problems. *Differential Equations* 1990; **26**(7):897–906.
3. Kaporin IE. New convergence results and preconditioning strategies for the conjugate gradient method. *Numerical Linear Algebra with Applications* 1994; **1**(2):179–210.
4. Kaporin IE. High quality preconditioning of a general symmetric positive definite matrix based on its $U^{\mathrm{T}}U + U^{\mathrm{T}}R + R^{\mathrm{T}}U$-decomposition. *Numerical Linear Algebra with Applications* 1998; **5**(5):483–509.
5. Axelsson O, Lindskog G. On the rate of convergence of the preconditioned gradient methods. *Numerische Mathematik* 1986; **48**(5):499–523.
6. Kaporin IE. Explicitly preconditioned conjugate gradient method for the solution of unsymmetric linear systems. *International Journal of Computer Mathematics* 1992; **40**:169–187.
7. Axelsson O. *Iterative Solution Methods*. Cambridge University Press: Cambridge, 1994.
8. Kaporin IE. Spectrum boundary estimation for two-side explicit preconditioning. *Vestnik Moskovskogo Universiteta*, *serija 15*, *Vychislitel'naja Matematika i Kibernetika* 1993; **2**:28–42 (in Russian).
9. Tismenetsky M. A new preconditioning technique for solving large sparse linear systems. *Linear Algebra and its Applications* 1991; **154–156**:331–353.
10. Migallon V, Penades J, Szyld DB. Non-stationary multisplitting with general weighting matrices. *SIAM Journal on Matrix Analysis and Applications* 2001; **22**(4):1089–1094.
11. Karypis G, Kumar V. Multilevel k-way hypergraph partitioning, *Technical Report 98-036*, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, 1998.
12. Kaporin IE, Konshin IN. Parallel solution of large sparse SPD linear systems based on overlapping domain decomposition. In *Parallel Computing Technologies*, Malyshkin V (ed.), *Lecture Notes in Computer Science*, *Vol. 1662*, Springer: Berlin-Heidelberg-New York, 1999; 436–445.

13. Kaporin IE, Konshin IN. Parallel solution of symmetric positive definite systems based on decomposition into overlapping blocks. *Computational Mathematics and Mathematical Physics* 2001; **41**(4):481–493.
14. Axelsson O, Kaporin I, Konshin I, Kucherov A, Neytcheva M, Polman B, Yeremin A. Comparison of algebraic solution methods on a set of benchmark problems in linear elasticity. *Technical Report*, Department of Mathematics, University of Nijmegen, The Netherlands, 2000, 89p.
15. Axelsson O, Kaporin I, Kucherov A, Neytcheva M. Benchmark problems in linear elasticity. Part I: Direct and robust second order incomplete factorization methods. *International Journal for Numerical Methods in Engineering* 2000, submitted for publication.
16. Eisenstat SC, Gursky MC, Schultz MH, Sherman AH. Yale sparse matrix package I: The symmetric codes. *International Journal for Numerical Methods in Engineering* 1982; **18**:1145–1151.