Parallel Solution of Large Sparse SPD Linear Systems Based on Overlapping Domain Decomposition

Igor E. Kaporin and Igor N. Konshin

Computing Center RAS
Vavilov str. 40, 117967 Moscow, Russia
{kaporin,horse}@ccas.ru

Abstract. We present a parallel iterative solver for large sparse symmetric positive definite (SPD) linear systems based on a new theory describing the convergence of the Preconditioned Conjugate Gradient (PCG) method and a proper combination of advanced preconditioning strategies. Formally, the preconditioning can be interpreted as a special (nearly optimum from the viewpoint of the new PCG theory) version of overlapping domain decomposition with incomplete Cholesky solutions over subdomains. The estimates of parallel efficiency are given as well as the results of numerical experiments for the serial and parallel versions of the solver.

1 The New Theory of the PCG Method Convergence

Consider the PCG method [1] for solving linear algebraic system

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

with SPD sparse $n \times n$ matrix A:

$$r_0 = b - Ax_0, \quad p_0 = Hr_0; \qquad \text{for } i = 0, 1, \ldots :$$

$$\alpha_i = \frac{r_i^T H r_i}{p_i^T A p_i}, \quad x_{i+1} = x_i + p_i\alpha_i, \quad r_{i+1} = r_i - Ap_i\alpha_i,$$

$$\beta_i = \frac{r_{i+1}^T H r_{i+1}}{r_i^T H r_i}, \quad p_{i+1} = Hr_{i+1} + p_i\beta_i. \tag{2}$$

The standard upper bound for the iteration number needed for the ε times reduction of the error norm $\sqrt{r_i^T A^{-1} r_i}$ is

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

where

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

V. Malyshkin (Ed.): PaCT-99, LNCS 1662, pp. 436–446, 1999.

is the spectral condition number of the preconditioned matrix HA.

Motivated by the need of a more feasible preconditioning quality criterion, the following theory was developed. It was shown in [5,7] that the number of iterations needed for the ε times reduction of the error norm $\sqrt{r_i^T H r_i}$ can be bounded from above as

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

where

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

is the so-called K-condition number of the preconditioned matrix HA, see [2]. A number of preconditioning strategies were analyzed with respect to a decrease of the K-condition number, see [4,5,7,8]. The preconditioning discussed below is actually based on a proper combination of the Block Incomplete Inverse Cholesky (BIIC) preconditionings [4,6,7] and the robust Incomplete Cholesky 2nd order (IC2) preconditionings [8].

2 A Simple Version of IC2 Factorization

For the sake of convenience, let us assume further that the matrix A is symmetrically scaled to the unit diagonal. The basic relationship determining the IC2 type preconditioning has the form

$$A = U^T U + U^T R + R^T U, \tag{7}$$

where U is a nonsingular upper triangular matrix (an approximation to the exact right Cholesky factor of A), and R is a strictly upper triangular error matrix with "small" elements. The existence and correctness of such decomposition is guaranteed for any SPD matrix A. Some modifications of this approach requiring less computational effort can be found in [8,10].

Very important properties of the IC2 two-side preconditioned matrix

$$M = U^{-T} A U^{-1} \equiv I + RU^{-1} + U^{-T} R^T \tag{8}$$

are as follows:

$$\mathrm{diag} M = I, \qquad K(M) = \det(I + RA^{-1}R^T). \tag{9}$$

The recurrences for the calculation of IC2 factorization can easily be obtained from (7), e.g., when the sparsity patterns of U and R do not have coinciding nonzero positions and their nonzero elements are subjected to the conditions $|U_{i,j}| \geq \tau$ and $|R_{i,j}| < \tau$, respectively, $i < j$. Here $0 < \tau \ll 1$ is the drop tolerance parameter determining the preconditioning quality. In particular, the second equation of (9) easily yields $\log K(M) = O(\|A^{-1}\|\tau^2)$ which should be related to (5). The latter remark explains why this decomposition is referred to as the second order one.

However, the calculation and application of the incomplete Cholesky preconditioning cannot be efficiently parallelized for message-passing architectures having relatively large communication overheads. Henceforth, next we consider some special parallel preconditionings which effectively exploit IC2 factorizations of certain set of submatrices of the original coefficient matrix A.

3 Block Explicit Preconditioner

Let us briefly describe the Block Incomplete Inverse Cholesky (BIIC) preconditioning algorithm [4,6,7]. Let A be reordered and split in the same way as for the Block Jacobi preconditioning, i.e. the t-th diagonal block of the symmetrically reordered matrix has the dimension n_t and $n_1 + \ldots + n_s = n$. Here $t = 1, ..., s$, and s is the block dimension of A. For the t-th diagonal block, let us define the "basic" index set as

$$\{k_{t-1} + 1, ..., k_t\},$$

where $k_{t-1} = n_1 + ... + n_{t-1}$, $k_0 = 0$, $k_s = n$, and introduce the "overlapping" index sets as

$$\{j_t(1), ..., j_t(m_t - n_t)), \quad j_t(p) \le k_{t-1}.$$

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. Here $m_t \ge n_t$ and, obviously, $m_1 = n_1$, i.e. at least the first overlapping set is empty. The BIIC preconditioner H can be represented in the following additive form:

$$H = \sum_{t=1}^{s} V_t U_t^{-1} (V_t^T E_t E_t^T V_t) U_t^{-T} V_t^T, \tag{10}$$

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

$$V_t = [Q_t | E_t], \quad t = 1, ..., s,$$

$$Q_t = [e_{j_t(1)} | ... | e_{j_t(m_t - n_t)}], \qquad E_t = [e_{k_{t-1}+1} | ... | e_{k_t}],$$

(Q_1 is set to an empty matrix), and each upper triangular matrix U_t is the right Cholesky factor of the t-th "extended" diagonal $m_t \times m_t$ submatrix $V_t^T A V_t$, that is,

$$V_t^T A V_t = U_t^T U_t. \tag{11}$$

Remark 1. It was shown in [4,6,7] that the BIIC preconditioner H possesses the K-optimality property in the following sense. Let $\mathcal{L}$ be a set of sparse lower triangular matrices which may have nonzero elements only in positions (i, j),

$$j \in \{j_t(1), \ldots, j_t(m_t - n_t), k_{t-1} + 1, \ldots, i\}, \quad k_{t-1} + 1 \le i \le k_t.$$

Then

$$H = \arg \min_{H = L^T L, L \in \mathcal{L}} K(HA).$$

Another useful property is $\mathrm{trace}(HA) = n$.

Remark 2. Note that the Block Jacobi preconditioner (where $m_t = n_t$, $t = 1, \ldots, s$, and therefore all Q_t are empty matrices) can be represented using the above notations as

$$H = \sum_{t=1}^{s} E_t (E_t^T A E_t)^{-1} E_t^T,$$

where $E_t^T A E_t$ is exactly the t-th diagonal block of A.

4 Using IC2 Factorizations within the BIIC Preconditioning

For each $t = 1, \ldots, s$, let us replace the exact Cholesky factorizations (11) by the corresponding IC2 decompositions of the type (7)

$$V_t^T A V_t = \tilde{U}_t^T \tilde{U}_t + \tilde{U}_t^T R_t + R_t^T \tilde{U}_t. \tag{12}$$

For the sake of simplicity, let also assume that the small element dropping strategy is modified in such a way that the first $m_t - n_t$ diagonal elements of $\tilde{U}_t$ coincide with those of U_t. Using the properties of the IC2 and BIIC decompositions mentioned above, one can see that the ratio of the K-condition numbers for the matrices $\tilde{H}A$ and HA, where

$$\tilde{H} = \sum_{t=1}^{s} V_t \tilde{U}_t^{-1} (V_t^T E_t E_t^T V_t) \tilde{U}_t^{-T} V_t^T, \tag{13}$$

can be estimated as

$$\frac{K(\tilde{H}A)}{K(HA)} = \prod_{t=1}^{s} \det(I + R_t (V_t^T A V_t)^{-1} R_t^T) \le \exp(c_0 \tau^2).$$

The latter inequality follows since R_t are the local IC2 error matrices with $O(\tau)$ elements. Moreover, we expect c_0 to be not very large even for ill-conditioned matrices since usually $\|(V_t^T A V_t)^{-1}\| \ll \|A^{-1}\|$. This yields

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

and the estimate (5),(6) shows that for some reasonably small IC2 dropping tolerance τ such BIIC-IC2 hybrid construction will give nearly the same rate of the PCG convergence as for the K-optimum BIIC preconditioner. At the same time BIIC-IC2 involves essentially smaller costs for the evaluation, storage, and the use of the preconditioner.

5 Parallel Implementation

The above mathematical technologies were implemented in a portable software written in simplified message-passing style using the MPI-like interface to the low-level communication library.

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

The proposed algorithm can be implemented as follows. Perform the IC2 factorization (12) of the local submatrix $V_i^T A V_i$ at the local i-th PE. No data exchanges are required at this stage.

The PCG iterations stage (2) involves the following types of operations:

(a) multiplication of the coefficient matrix A by a vector,

(b) multiplication of the preconditioner $\tilde{H}$ by a vector, and

(c) inner product.

Three vector update operations are also needed on each iteration, but they do not require interprocessor communications with our distribution of data, where t-th processor stores the 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.

Matrix by a vector product. The multiplication of matrix 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 required components of the vector to a local data buffer and send them to PEs which require them;

2. receive the required 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 multiplication of preconditioner by a vector can be presented as an analogous algorithm. 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 (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_i^T and U_i are performed, and the second global synchronization operation is required after these local computations.

Inner product. Two inner products are required to perform at each PCG iteration (2). This is one global exchange MPI-like operation consisting from the following local ones:

1. the partial inner product for the local part of the vector is computed at each PE;

2. the scalars obtained are sent to the root PE;

3. the final scalar product is computed on the root PE;

4. the final scalar product is sent from the root PE to the other PEs.

Communication costs. Let us consider a model 3-D problem with standard 7-point stencil matrix operator on a cube, which is split into p^3 cubic subdomains each of the size $m \times m \times m$, i.e. $n = m^3 p^3$ and $N_{\mathrm{PE}} = s = p^3$. Let the overlap be of the width of q grid points and the corresponding preconditioner $\tilde{H}$ is two times more dense than the original matrix A (this is a typical preconditioner density used in practice).

The arithmetic and communication costs per one PCG iteration are given in Table 1. It can be easily seen from the Table 1 that the ratio of the total arithmetic costs to the communication costs is approximately equal to $c = 4m/(q+1)$, i.e. it is required to perform c arithmetic operations per one float word exchange between PEs. It should be noted that for 2-D decomposition this ratio is expressed by the same formula, where $m = (n/s)^{1/2}$.

Table 1. The costs per one PCG iteration for the uniform $p \times p \times p$ DD.

Stage	Arithmetic costs	Communication costs
(a) Ax	$7n$	$6m^2 s$
(b) $\tilde{H}x$	$14n$	$6mq(m+2)s$
(c) $y^T x$	$2n$	$4log_2 s$

Properties of the parallel realization. (1) The total number of global exchange initialization operations does not depend on the number of blocks and the size of the linear system and is equal to 5 per each PCG iteration. (2) After completion of the global data exchange, all the computations can be performed at each PE without any additional synchronizations. (3) There is no serial part in the code implementing the proposed algorithm. (4) The computations are well balanced if the sizes of the local submatrices are approximately equal. (5) The communication costs are not large as compared to the arithmetic costs.

6 Numerical Experiments

We present numerical results obtained on eight-processor SUN 10000 Starfire computer and a cluster of four Pentium II workstations. We have used certain hard-to-solve test matrices arising in finite element analysis of thin shells which were examined in [3] and are available from *MatrixMarket* collection (CYLSHELL set available at URL http://math.nist.gov/MatrixMarket/data/misc/cylshell/cylshell.html). We also present the results obtained for the matrix BIHAR255 resulting from discrete biharmonic operator with 13-point stencil on a 255×255 square grid with Dirichlet type boundary conditions [4,6,7].

Some data on these test matrices are given in Table 2.

Zero initial guess and the relative stopping criterion by the Jacobi scaled residual norm with $\varepsilon = 10^{-9}$ were used for all test problems. For the CYLSHELL

Table 2. Test matrix properties.

name	mesh	n	nz(A)	$C(A)$
S1RMQ4M1	30×30	5489	281111	$1.8 \cdot 10^6$
S3RMQ4M1	30×30	5489	281111	$1.8 \cdot 10^{10}$
S3DKQ4M2	150×100	90499	4820891	$1.9 \cdot 10^{11}$
S3DKT3M2	150×100	90499	3753461	$3.6 \cdot 10^{11}$
BIHAR255	255×255	65025	840229	$2.2 \cdot 10^8$

set, the right hand side was computed from the test solution $x = [1 \dots 1]^T$ as in [3], while for BIHAR255 the test solution was obtained from the function $x \sin(\pi x) \sin(\pi y) \exp(xy)$ over the unit square. Block splittings of the matrices were obtained with the use of the public-domain graph partitioning package METIS [9] with the default parameters. The overlap was obtained using sparsity structure of the q-th degree of the coefficient matrix. The preconditioning was constructed using overlap parameter $q = 6$ and IC2 drop tolerance parameter $\tau = 0.003$.

The results on parallel performance are illustrated for BIHAR255 test. The speedups and wall clock times (in seconds) obtained on a SUN 10000 Starfire computer are given in Table 3. The case of 8 PEs is not presented because one node of the computer system was permanently occupied by another process running for a week or even more.

Table 3. Parallel efficiency for BIHAR255 test.

N_{PE}	time	Mflops/N_{PE}	Mults/Sends	Efficiency	Speedup	Actual speedup
1	766.90	16.24	–	100.00	1.00	1.00
2	239.46	16.14	1394.44	97.95	1.96	3.20
3	133.14	15.75	905.54	96.85	2.91	5.76
4	93.67	15.92	612.59	97.51	3.90	8.18
5	69.88	15.92	415.98	95.67	4.78	10.97
6	65.36	15.36	345.35	89.32	5.36	11.73
7	52.55	14.97	294.22	88.39	6.19	14.59

The observed superlinear actual speedup is due to the sharp decrease of the preconditioning costs when passing from the incomplete factorization of the whole matrix to that of its submatrices, which have significantly smaller bandwidth (see Table 4).

The parallel efficiency obtained on a cluster of four Pentium II 266 MHz workstations connected via an 100 Mbit Ethernet switch was within the 45 to 85% range for the problems of larger sizes.

In Tables 4–6, AC_{prec}, AC_{iter}, and AC_{tot} denote the number of scalar multiplications needed to construct the preconditioner, perform N_{iter} PCG iterations, and the total number of multiplications divided by $\text{nz}(A)$, respectively. The "Fill-in" given in percents means the ratio of the space occupied by the preconditioner to the space occupied by the upper triangle of the coefficient matrix.

Table 4. Dependence of operation count and iteration number on the number of blocks for BIHAR255 test.

s	$C(\tilde{H}A)$	N_{iter}	AC_{prec}	AC_{iter}	AC_{tot}	Fill-in,%
1	0.215E+05	408	4934.51	1824.44	6758.95	278.20
2	0.111E+05	313	2250.61	1509.73	3760.35	310.47
3	0.122E+05	328	2143.84	1596.56	3740.40	314.61
4	0.990E+04	314	1547.47	1549.70	3097.17	320.83
5	0.114E+05	315	1246.48	1568.12	2814.59	324.79
6	0.119E+05	336	1165.37	1696.17	2861.54	331.35
7	0.115E+05	328	974.74	1664.06	2638.80	333.66

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 Jacobi preconditioned CG method, where the number of iterations grows rapidly with the number of subdomains.

In order to compare some other parallel preconditioning to our method, we present iteration data for the diagonal (Jacobi) preconditioning, (approximate) Block Jacobi and the simple "uniform" overlapping with weighting. The same IC2 approximate inversion of blocks were used for the latter two methods. The results obtained with the number of blocks $s = 4$ for the test problem S3DKT3M2 are given in Table 5.

Table 5. Comparison of parallel iterative methods for S3DKQ4M2 test.

Preconditioner	$C(\tilde{H}A)$	N_{iter}	AC_{prec}	AC_{iter}	AC_{tot}	Fill-in,%
Jacobi	0.312E+11	41930	0.00	42272.13	42272.13	0.00
Block Jacobi	0.434E+09	1251	391.59	3530.51	3922.10	158.89
Simple overlap	0.356E+09	1267	755.91	4175.87	4931.78	205.12
BIIC overlap	0.812E+08	643	429.93	1932.18	2362.11	181.22

Further results obtained for the test problems S3DKQ4M2 and S3DKT3M2 are presented in Tables 6 and 7, respectively.

Table 6. Dependence of operation count and iteration number on the number of blocks for S3DKQ4M2 test.

s	$C(\tilde{H}A)$	$N_{\rm iter}$	$AC_{\rm prec}$	$AC_{\rm iter}$	$AC_{\rm tot}$	Fill-in,%
1	0.323E+08	553	456.48	1446.36	1902.84	147.08
2	0.467E+08	584	474.92	1571.24	2046.16	154.45
3	0.513E+08	579	464.80	1603.54	2068.34	162.19
4	0.532E+08	573	414.62	1582.98	1997.60	161.51
5	0.561E+08	579	422.40	1636.23	2058.63	167.72
6	0.583E+08	569	425.58	1644.81	2070.39	174.06
7	0.597E+08	578	442.12	1671.39	2113.51	174.16
8	0.650E+08	568	403.83	1664.51	2068.34	177.95
10	0.643E+08	557	390.80	1667.49	2058.29	184.14
12	0.704E+08	574	416.56	1749.91	2166.47	189.53
16	0.681E+08	532	365.79	1686.47	2052.26	201.39
20	0.789E+08	564	375.07	1839.39	2214.45	210.37
24	0.876E+08	555	357.88	1844.29	2202.17	216.41
32	0.862E+08	546	363.74	1895.77	2259.51	231.00

Table 7. Dependence of operation count and iteration number on the number of blocks for S3DKT3M2 test.

s	$C(\tilde{H}A)$	$N_{\rm iter}$	$AC_{\rm prec}$	$AC_{\rm iter}$	$AC_{\rm tot}$	Fill-in,%
1	0.398E+08	614	528.23	1738.48	2266.71	164.28
2	0.620E+08	667	518.44	1930.44	2448.88	170.44
3	0.692E+08	666	481.57	1953.73	2435.30	174.27
4	0.776E+08	658	445.01	1967.11	2412.13	179.73
5	0.847E+08	673	445.58	2026.98	2472.56	181.92
6	0.841E+08	682	463.73	2117.29	2581.02	190.96
7	0.906E+08	651	420.54	2000.73	2421.27	187.89
8	0.869E+08	637	424.42	2012.10	2436.52	196.21
10	0.982E+08	666	387.29	2188.58	2575.88	208.65
12	0.107E+09	642	348.71	2082.22	2430.93	204.46
16	0.112E+09	628	326.98	2098.51	2425.48	214.03
20	0.131E+09	658	310.09	2251.29	2561.38	221.84
24	0.137E+09	650	312.75	2278.20	2590.95	229.97
32	0.143E+09	661	288.25	2410.93	2699.18	243.87

As is seen, with the increase of the number of blocks s, the iteration number of the proposed method stays nearly the same, which is important for attaining high speedups on computers with large communication latency. Also, the increase in the storage occupied by the preconditioner is not very substantial as long as the number of subdomains is not large. The reason is that the increase of the overlap is partially compensated by more precise and more sparse incomplete factorizations corresponding to smaller blocks.

7 Conclusions

The experience accumulated by the authors (more than 500 test runs, only small portion of which is presented above) shows that the harder the problem is, the greater the gain in the performance of the proposed method 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.

Acknowledgements

The authors would like to acknowledge several useful discussions with V.A.Garanzha and V.N.Konshin concerning algorithmic implementation issues and models of parallel computations.

References

1. Axelsson, O.: A class of iterative methods for finite element equations. Computer Meth. Appl. Mech. Engrg. **9** (1976) 123–137 436
2. Axelsson, O.: Iterative solution methods. Cambidge University Press, Cambridge (1994) 437
3. Benzi, M., Kouhia, R., Tuma, M.: An assessment of some preconditioning techniques in shell problems. Technical Report LA-UR-97-3892, Los Alamos National Laboratory, Los Alamos, NM (1992) 441, 442
4. Kaporin, I. E.: On preconditioning for the conjugate gradient method when solving discrete analogues of differential problems. Differ. Uravn. **7** (1990) 1225–1236 (in Russian) 437, 438, 441
5. Kaporin, I. E.: Explicitly preconditioned conjugate gradient method for the solution of unsymmetric linear systems. Int. J. Computer Math. **40** (1992) 169–187 437
6. Kaporin, I. E.: Spectrum boundary estimation for two-side explicit preconditioning. Vestnik Mosk. Univ., ser. 15, Vychisl. Matem. Kibern. **2** (1993) 28–42 (in Russian) 437, 438, 441
7. Kaporin, I. E.: New convergence results and preconditioning strategies for the conjugate gradient method. Numer. Linear Algebra Appls., **1** (1994) 179–210 437, 438, 441
8. Kaporin, I. E.: High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition. Numer. Linear Algebra Appl., **6** no.2 (1999) (to appear) 437

9. Karypis, G., Kumar, V.: Multilevel k-way hypergraph partitioning, Technical Report 98-036, Dept. Comp. Sci. Engrg., Army HPC Research Center, Univ. of Minnesota, MN (1998) 442
10. Tismenetsky, M.: A new preconditioning technique for solving large sparse linear systems. Linear Algebra Appls. 154-156 (1991) 331–353 437