A Multipole Approach for Preconditioners

Ph. Guillaume, A. Huard and C. Le Calvez

MIP, UMR 5640
INSA, Département de Mathématiques
135 Avenue de Rangueil, 31077 Toulouse Cedex 4, France
guillaum,huard,lecalvez@gmm.insa-tlse.fr

Abstract. A new class of approximate inverse preconditioners is presented for solving large linear systems with an iterative method. It is at the intersection of multipole, multigrid and SPAI methods. The method consists in approximating the inverse of a matrix by a block constant matrix, instead of approximating it by a sparse matrix as in SPAI methods. It does not require more storage, or even less, and it is well suited for parallelization, both for the construction of the preconditioner and for the matrix-vector product at each iteration.

1 Introduction

Recently, multipole methods [9, 4, 2, 19] have dramatically improved the solution of scattering problems in Electromagnetism. The basic idea behind multipole methods consists in a low rank approximation of far field interactions. The matrix A obtained when using integral equations can at the first glance be considered as an approximation of the Green function $G(x, y)$ of the problem, that is $A_{ij} \simeq G(x_i, x_j)$ if the x_i's are the discretization points. The matrix A is dense, and for large problems it becomes impossible to store the whole matrix. When two points x^* and y^* are distant from each other, the starting point of the multipole approach is a separate variables approximation

$$G(x, y) \simeq \sum_{n=1}^{r} u_n(x)v_n(y)$$

which is valid for x close to x^* and y close to y^*. It leads to a low-rank approximation of the associated block of the matrix A :

$$A_{IJ} \simeq UV^T.$$

Here I and J are sets of indices of points respectively close to x^* and y^*. The sizes of A_{IJ}, U and V are respectively $|I| \times |J|$, $|I| \times r$ and $|J| \times r$. Hence both memory and computational time are saved if $r << |I|$ and $r << |J|$. When x^* and y^* are close to each other, the above approximation is not valid anymore, and $G(x^*, y^*)$ must be computed more carefully. This approach is general, and relies on the fact that the Green function associated to a pseudo-differential operator is singular on the diagonal, but regular outside.

The context looks quite different when considering finite element methods: the matrix A issued from the discretization of a partial differential equation is usually sparse, and there is a priori no interest for using a low-rank approximation of almost zero blocks (see however [3]). But its inverse A^{-1} is a dense matrix and, like the matrix issued from an integral equation, it is associated to a Green function $G(x, y)$ which is singular on the diagonal $x = y$ but smooth outside the diagonal. In the context of an approximate inverse used as a preconditioner in an iterative method, one can think of approximating off-diagonal blocks of A^{-1} by low-rank matrices. Since it leads to a number of unknowns significantly larger than in the original problem and because we do not need such a good approximation as in the case of integral equations, we can go even further in the approximation: off-diagonal blocks $(A^{-1})_{IJ}$ can be simply approximated by constant blocks. The size of the constant blocks can vary: smaller when they are close to the diagonal, and getting larger away from it. This approach makes sense in the case of non oscillatory Green functions associated to an elliptic equation like Poisson's equation or elasticity equations. This relies on the fact that piecewise constant functions can well approximate the Green function of the problem. It would not be well suited for Green functions arising for example from Helmholtz equations.

Like in the multipole method (see *e.g.* [7] [15]), a crucial point is the ordering of the unknowns. They need to be sorted by proximity, that is, in such a way that unknowns associated to neighboring points must be grouped together, and vice-versa. When the nodal table which has been used for assembling the matrix is available, a simple way to achieve this is to use a recursive coordinates bisection, but more sophisticated methods are available like recursive graph bisections or recursive spectral bisection [16] [18] [17] which do not require the nodal table.

In this paper, we focus our attention on the solution of systems issued from the discretization of elliptic partial differential equations, leading to a sparse symmetric and positive definite (SPD) matrix A. Our goal is to obtain a block constant approximate inverse C of the matrix A, used for preconditioning an iterative method.

Section 2 describes how to determine such a Block Constant Preconditioner (BCP) together with it's relation to multipole, multigrid and sparse approximate inverse (SPAI) methods. Section 3 reports some numerical experiments: the BCP is compared to a basic SPAI using the sparsity pattern of the original matrix A [6] and to the incomplete Cholesky factorization with no fill-in IC(0).

2 Description of the Block Constant Preconditioner

Consider a linear system

$$Ax = b, \quad x, b \in \mathbb{R}^n$$

where $A \in \mathcal{M}_n(\mathbb{R})$ is an SPD matrix. As mentioned in the introduction, the unknowns must be ordered by proximity, like in multipole methods. In the sequel, we suppose that this reordering has been done.

When the dimension n is large, an iterative method like the preconditioned conjugate gradient (PCG) is often used for solving such a system. It consists in applying the conjugate gradient algorithm to a system of the form $MAx = Mb$. Here M is an explicit left preconditioner. It should also be SPD. Right preconditioners can be used in a similar way, and will not be discussed here. The BCP preconditioner M is of the form

$$M = C + \omega I, \quad \omega > 0 \tag{1}$$

where I is the identity matrix and C is a block constant matrix (BCM), which consists in rectangular blocks of variable size whose elements are constant. The steps for computing the BCP of the matrix A are the following:

- determine the pattern of the BCM, i.e., location and size of the different constant blocks,
- compute the constants associated to this pattern by minimizing some Frobenius norm of $CA - I$ over the set of matrices having the same pattern,
- choose the parameter ω.

The way of computing C resembles SPAI methods [13] [11] [5]. The computation of C as well as the matrix-vector preconditioning operation is highly parallelizable. The difference lies in the fact that the approximation of the Green function $G(x, y)$ by a sum of discrete Dirac functions (SPAI methods) is replaced by a piecewise constant function, which is likely to offer a better approximation outside the diagonal $x = y$. For some particular cases of pattern of the BCM, the method becomes very close to a two-level multigrid method [10] [14]. In the general case, the difference lies in an attempt to simulate a cycle over several grids in a single operation. Hence, the BCP method is at the intersection of multipole, multigrid and SPAI methods.

2.1 Determination of the pattern of the BCM

A BCM pattern is obtained by a recursive splitting of the initial matrix A. The depth of recursiveness (the level of refinement) is determined by three parameters l_c, l_d and l_o:

- l_c is the coarsest level of refinement, and fixes the size of the largest blocks of the BCM,
- l_o is an intermediate level of refinement and determines the size of the smallest off-diagonal blocks of the BCM,
- l_d is the finest level of refinement and determines the size of the blocks of the BCM containing the diagonal elements.

Let d be the integer defined by $2^{d-2} < n \leq 2^{d-1}$. It is supposed here that $l_c \leq l_o \leq l_d \leq d$. The two extreme situations correspond to $l_d = 1$, in which case the whole matrix is constant, and to $l_c = d$, in which case each block of the BCM is of size 1×1.

Different values of these parameters lead to different sequences of refinement, and consequently to different patterns. Each refinement consists in splitting a block K into four blocks of equal size if possible. The algorithm is the following:

```
REFINEMENT ALGORITHM FOR DETERMINING THE PATTERN
$lev = 1$ ; $K$ is a $n \times n$ block of level 1;
for $lev = 2 : l_c$
    split each $k \times l$ block $K$ of level $lev - 1$ into 4 blocks of size $k_i \times l_j$
    where $k_1 \geq k_2$, $k_1 + k_2 = k$, $k_1 - k_2 \leq 1$, and $l_1 \geq l_2$, $l_1 + l_2 = l$, $l_1 - l_2 \leq 1$;
endfor
for $lev = l_c + 1 : \max(l_o, l_d)$
    for each block $K$ of level $lev - 1$
      if
        $K$ is a diagonal block and $lev \leq l_d$
        or
        $K$ is not a diagonal block and $lev \leq l_o$
      and
        the corresponding block of $A$ has nonzero elements
      then
        split the block $K$ into 4 blocks following the previous rule as long as
        possible: if $\min(k, l) = 1$, just split it into 2 blocks when $k \neq l$ and of
        course do not split it if $k = l = 1$;
      endif
    endfor
 endfor
```

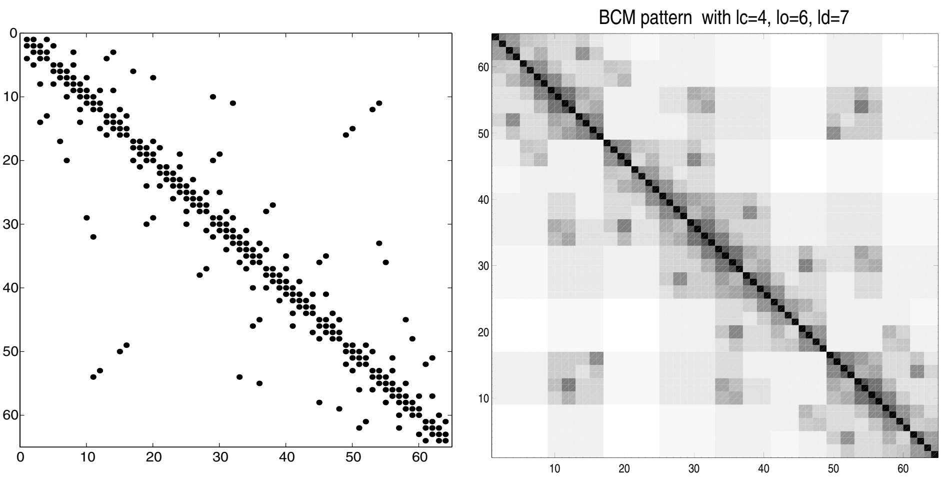

Fig. 1. Reordered A and example of BCM pattern

Note that the obtained pattern is also symmetric if A is symmetric. Following this algorithm, the BCM in Fig. 1 (right) has been obtained from the classical Poisson's matrix of size 64×64 reordered by a recursive coordinates bisection method (Fig. 1, left). With $l_c = 4$, $l_o = 6$ and $l_d = 7$, the largest blocks are 8×8, the smallest off-diagonal blocks are 2×2 and the smallest diagonal blocks are 1×1 (as all A_{ii} are nonzero, all diagonal blocks are in fact of size 1×1). The parameter l_d makes it possible to have a finer refinement on the diagonal where the Green function is singular. Each color in Fig. 1 (right) corresponds to a constant block. One can see that the blocks are smaller at the locations where the matrix A has nonzero elements. As in SPAI methods, many other algorithms can be proposed. For example, instead of using the pattern of A, one could use the pattern of A^s for a certain integer s.

2.2 Definition of the BCP

Once the pattern of the BCM has been determined, it remains to compute the values c_i of the constants for each block. Let $\mathcal{C} \subset \mathcal{M}_n(\mathrm{I\!R})$ denote the linear space of matrices which satisfy a given pattern. A basis of this space consists in matrices E_i which have a single nonzero block (corresponding to the given pattern) with value 1. In this basis, a matrix $C \in \mathcal{C}$ can be written

$$C = \sum_{i=1}^{n_b} c_i E_i$$

where n_b is the dimension of $\mathcal{C}$ (the number of blocks). The space $\mathcal{M}_n(\mathrm{I\!R})$ is equipped with the Frobenius norm defined by its square

$$\|A\|_F^2 = \sum_{i,j=1}^{n} A_{ij}^2, \quad A \in \mathcal{M}_n(\mathrm{I\!R}).$$

This norm is associated to the scalar product

$$A : B = \sum_{i,j=1}^{n} A_{ij} B_{ij} = \mathrm{trace}(AB^T), \quad A,\, B \in \mathcal{M}_n(\mathrm{I\!R}).$$

The subspace $\mathcal{C}$ is equipped with the same norm.

Definition 1. *The block constant preconditioner for solving the linear system $Ax = b$ is defined by*

$$M = \frac{1}{2}(C + C^T) + \omega I$$

where $C \in \mathcal{C}$ is the solution to the residual norm minimization problem

$$\min_{C \in \mathcal{C}} \left\| (CA - I) A^{-1/2} \right\|_F^2 \tag{2}$$

and $\omega > 0$ is chosen in such a way that M is a symmetric and positive definite matrix.

If $l_c = l_d$ and if A is symmetric, then it can be shown that C is also symmetric, thus $M = C + \omega I$. When $l_d = d$, it may not be necessary to add a diagonal term ωI, because in that case $\mathcal{C}$ contains already the diagonal matrices. The larger is l_d, the larger is the number n_b of unknowns involved in the BCP, however the numerical results of Section 3 show that taking $l_d < d$ performs well for a reasonable number $n_b < n$ of constant blocks. When $l_d < d$, adding some diagonal terms becomes necessary because, as in multigrid methods, CA is not invertible and some smoother must complete the preconditioning operation CA. More general definitions are possible, as for example $M = C + D$ with D diagonal, and M minimizing $\|(MA - I)A^s\|_F$. The value $s = -1/2$ has a precise meaning only for SPD matrices, although the optimality condition (3) can be used for any kind of matrices, without the guaranty of a minimum. A simple choice for non SPD matrices is to take $s = 0$.

2.3 Computation of the BCM

The computation of the constants c_i follows from the next proposition.

Proposition 1. *If the matrix A is symmetric and positive definite, Problem (2) has a unique solution, which is also the solution to the linear system of equations*

$$(CA - I) : H = 0, \quad \forall H \in \mathcal{C}. \tag{3}$$

This linear system has a unique solution, and for $C = \sum_{i=1}^{n_b} c_i E_i$, it reads

$$Nc = g, \;\; N \in \mathcal{M}_{n_b}(\mathbb{R}),\; c, g \in \mathbb{R}^{n_b} \tag{4}$$

where for i, $j = 1, \ldots, n_b$

$$N_{ij} = A : E_j^T E_i, \quad g_i = \mathrm{trace}(E_i).$$

Proof. Equ. (3) is the optimality condition associated to Problem (2). Furthermore, the associated homogeneous system $CA : H = 0$ for all $H \in \mathcal{C}$ has the unique solution $C = 0$ (taking $H = C$ yields $\|CA^{1/2}\|_F^2 = 0$). Hence (3) has a unique solution, which is given by (4) when using the basis $(E_i)_{i=1}^{n_b}$ defined in Section 2.2.

The matrix N is block diagonal with $p = 2^{l_c - 1}$ sparse blocks on the diagonal. Hence its computation as well as solving Equ. (4) are parallelizable. Depending on the distribution of the matrix A among the processors, its parallelization may be complicated but still remains possible. Similarly, the matrix-vector preconditioning operation is parallelizable.

3 Numerical Results

We present some numerical experiments with the Poisson equation on a square with an homogeneous Dirichlet boundary condition, solved by using the five-point finite difference stencil reordered by the recursive coordinate bisection algorithm [16].

The BCP is compared to the SPAI(1) (same pattern as A for the sparse approximate inverse) and IC(0) preconditioners. The incomplete Cholesky factorization is of the form $A = U^T U + R$. The three preconditioners are used with the conjugate gradient algorithm. The experiments presented here were performed

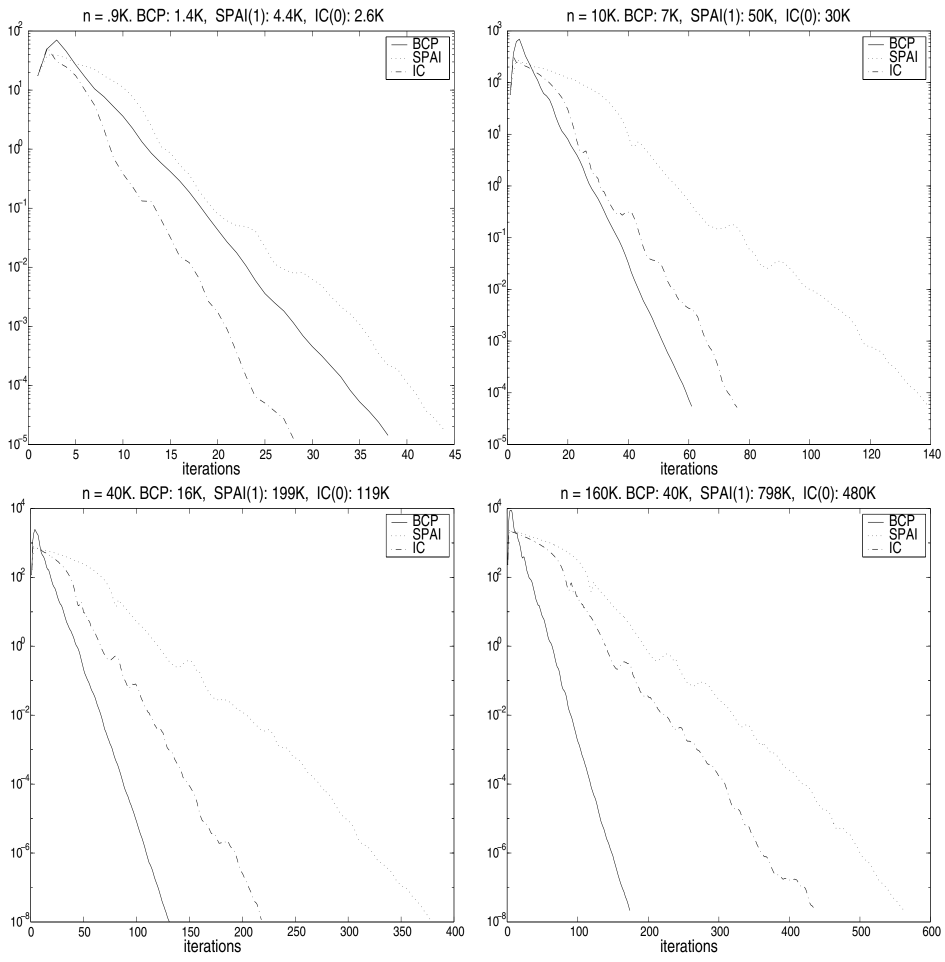

Fig. 2. Comparison BCP - SPAI - IC(0), $n = 900$, 10000, 40000, 160000.

with MATLAB and they all used the following parameters for the BCP:

```
omega = 1.5/normest(A);
d = ceil(log2(n))+1;
lc = ceil(d/2)-2;
lo = lc+3; ld = lo+2;
```

The function normest(A) computes an approximation of $||A||_2$ and ceil(x) is the smallest integer $i \geq x$. The letters omega, d, lc, ld and lo denote respectively ω, d, l_c, l_d and l_o (see Section 2 for the notation).

A first series of experiments is reported in Fig. 2 which shows the residual 2-norm history for four $n \times n$ matrices with increasing size $n = 900$, 10000, 40000 and 160000. At the top of each graphic is indicated the amount of storage (number of blocks or of nonzero elements) which was used by each preconditioner. One can observe that the BCP takes a clear advantage when n becomes large, and moreover it uses ten times less storage than IC(0) in the last case $n = 160000$. For an adequate coding of the BCP matrix-vector multiplication which will not be discussed here, the number of elementary operations per iteration is comparable.

A second series of experiments is reported in Fig. 3, which plots the number of iterations needed for obtaining a given precision (relative residual norm $\leq 10^{-6}$) versus the size of the system. One can see that the number of iterations with BCP is of order

$$O(h^{-1/2}) = O(n^{1/4}).$$

The results for BCP match the line $y = 6\,n^{1/4}$, and the results with IC(0) match the line $y = 2.5\,n^{0.35}$. For these experiments, the chosen values for n_b (number

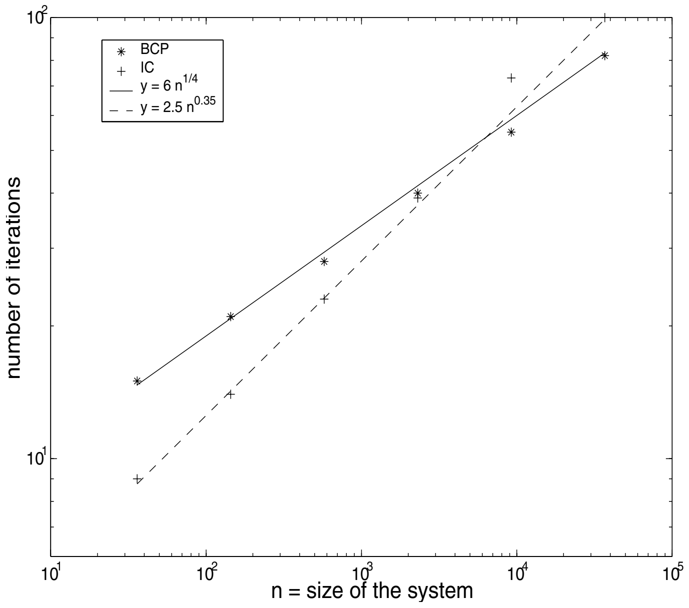

Fig. 3. Number of iterations with respect to the size n.

of blocks) and nnz(U) (number of nonzero elements in the matrix U) are given in Table 1. They indicate the amount of storage used by the preconditioners and they satisfy asymptotically

$$n_b \simeq \frac{n}{4}, \quad \texttt{nnz(U)} \simeq 3\,n.$$

Table 1. Storage used by the preconditioners.

n	36	144	576	2300	9200	36900
n_b	208	616	1300	2900	6300	15000
`nnz(U)`	90	408	1700	6900	27600	111000

A third series of experiments is reported in Table 2, which shows the spectral condition numbers $\lambda_{\max}/\lambda_{\min}$ of the matrices A, MA and $U^{-T}AU^{-1}$, with $M = C+\omega I$ and U defined above. We used the MATLAB function `eigs` fore computing them. One can observe that the condition number of MA is of order $\sqrt{n}$.

Table 2. Condition numbers.

n	n_b	cond(A)	cond(MA)	$\sqrt{n}$	cond$(U^{-T}AU^{-1})$
625	1480	273	23	25	25
2500	3220	1053	38	50	93
10000	7024	4133	65	100	366
40000	16048	16373	119	200	1448

Conclusion

The new BCP preconditioner is at the intersection of multipole, multigrid and SPAI methods. Its computation as well as its application on a vector is highly parallelizable. In our experiments the results showed that the BCP takes the advantage when the size of the system increases, both in terms of memory requirements and number of iterations, the costs per iterations being comparable.

References

1. O. Axelsson. Iterative Solution Methods. Cambridge University Press, 1994.
2. S. S. Bindiganavale and J. L. Volakis. Comparison of three fmm techniques for solving hybrid fe-bi systems. *IEEE Trans. Antennas Propagat. Magazine*, vol. 4 No. 4 (1997), 47-60.

3. R. Bramley and V. Meñkov. Low rand off-diagonal block preconditioners for solving sparse linear systems on parallel computers. Tech. Rep. 446, Department of Computer Science, Indiana University, Bloomington,1996.
4. R. Coifman, V. Rokhlin, and S.W Wandzura. The fast multipole method for the wave equation: A pedestrian description. *IEEE Trans. Antennas Propagat. Mag.*, vol. AP-35 No 3 (1993), 7-12.
5. E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations, SIAM Journal on Scientific Computing, 19 (1998), 995-1023.
6. J.D.F. Cosgrove, J.C. Dias. Fully parallel preconditionings for sparse systems of equations, Proceedings of the Second Workshop on Applied Computing, ed. S. Uselton, Tulsa, Oklahoma, The University of Tulsa, 1988, 29-34.
7. E. Darve. Méthodes multipôles rapides : résolution des équations de Maxwell par formulations intégrales. Thèse de doctorat de l'Université Paris 6 (France), 1999.
8. T.F. Dupont, R.P. Kendall, H.H. Rachford. An approximate factorization procedure for solving self-adjoint elliptic difference equations, SIAM J. Numer. Anal., vol 5 (1968), 559-573.
9. N. Engheta, W. D. Murphy, V. Rokhlin and M. S. Vassiliou. The fast multipole methode (fmm) for electromagnetic scattering problems. IEEE Trans. Antennas Propagat., 40 No. 6 (1992) 634–641
10. A. Greenbaum. Iterative methods for solving linear systems. SIAM, 1997.
11. M. Grote, H.S. Simon. Parallel preconditioning and approximate inverse on the Connection Machine, Proceedings of the Sixth SIAM conference on parallel processing for scientific computing, SINCOVEC, R. Ed. SIAM, 519-523.
12. I. Gustafsson. A class of 1st order factorization methods, BIT, 18 (1978), 142-156.
13. L. Yu. Kolotina, A. Yu. Yeremin. On a family of two-level preconditionings of the incomplete block factorization type. Sov. J. Numer. Anal. Math. Modelling 1 (1986), 292-320.
14. G. Meurant. Computer solution of large linear systems. North Holland, 1999.
15. J.R. Poirier. Modélisation électromagnétique des effets de rugosité surfacique. Thèse de doctorat de l'INSA Toulouse (France), 2000.
16. A. Pothen, Alex, Horst D. Simon, Kang-Pu Liou. Partitioning sparse matrices with eigenvectors of graphs. Sparse matrices (Gleneden Beach, OR, 1989). SIAM J. Matrix Anal. Appl. 11 (1990), no. 3, 430-452.
17. Y. Saad. Iterative Methods for Sparse Linear Systems, PWS publishing, New York, 1996.
18. Horst D. Simon, Shang-Hua Teng. How good is recursive bisection ? SIAM J. Sci. Comput. 18 (1997), no. 5, 1436-1445.
19. J. Song, C.-C. Lu, and W. C. Chew. Multilevel fast multipole algorithm for electromagnetic scattering by large complex objects. *IEEE Trans. Antennas Propagat.*, vol. 45 no. 10 (1997), 1488-149.