 ICCES, vol.3, no.2, pp.113-120, 2007

$\mathscr{H}$-matrix preconditioners for saddle-point systems from meshfree discretization [1]

Suely Oliveira [2] and Fang Yang [2]

Summary

In this paper we describe and compare preconditioners for saddle-point systems obtained from meshfree discretizations, using the concepts of hierarchical (or $\mathscr{H}$-)matrices. Previous work by the authors using this approach did not use $\mathscr{H}$-matrix techniques throughout, as is done here. Comparison shows the method described here to be better than the author's previous method, an AMG method adapted to saddle point systems, and conventional iterative methods such as JOR.

keywords: Multilevel methods, hierarchical matrices, saddle-point systems, meshfree method, algebraic multigrid

Introduction

Meshfree methods construct approximate solutions to elliptic partial differential equations using basis functions constructed based on discrete particles. In this paper, Reproducing Kernel Particle Methods (RKPM) [5] are used. The Lagrange Multiplier Method is used to deal with the essential boundary conditions, but it gives an indefinite system of saddle-point type. In [9] a scheme based on smoothed Algebraic Multigrid (AMG) is proposed to solve the saddle-point systems from meshfree methods.

In this paper we present an $\mathscr{H}$-matrix based approach to solve the saddle-point system. Hierarchical-matrices ($\mathscr{H}$-matrices) were introduced in [7] and since then, much work has been done on the theory and applications of $\mathscr{H}$-matrices [1,3,4,6]. $\mathscr{H}$-matrices provide a cheap but approximate way of carrying out matrix computations. The basic idea of the $\mathscr{H}$-matrix representation is that instead of representing a matrix exactly, approximations are used to represent a matrix by a hierarchical block cluster tree. The leaves of the tree represent the matrix blocks that are not partitioned further, which are represented by low-rank matrices (Rk-matrix format) or by full matrices (full matrix format). $\mathscr{H}$-matrix arithmetic is developed by adapting conventional matrix operations to the $\mathscr{H}$-matrix format. This reduces the required storage for a matrix, and the computational complexity of operations, such as matrix-vector multiplication, matrix-matrix multiplication, matrix addition and inversion, are reduced to almost linear complexity ($O(n\log^{\alpha} n)$) [1,6].

[1]This work was supported by NSF ITR grant DMS-0213305.

[2]Department of Computer Science, University of Iowa, Iowa City 52242 (email for contact: oliveira@cs.uiowa.edu).

$\mathscr{H}$-matrices have solved finite element systems. However, $\mathscr{H}$-matrix construction [6] relies on the underlying geometric structure of the problem. Recently, algebraic $\mathscr{H}$-matrix construction methods have been developed for sparse matrices [10,3], which use matrix graphs.

In [10] we presented an approach to solve saddle-point systems which relied on sparse matrix computations. In this paper we develop a scheme such that all the subblocks of saddle-point system are represented using only $\mathscr{H}$-matrices, which speeds the process of building $\mathscr{H}$-matrix preconditioners. The $\mathscr{H}$-matrix preconditioners built by the new scheme also show good and stable convergence rates.

In this paper, $\#A$ denotes the number of elements in the set A, and $S(i)$ means the children of the node i.

The model problem and meshfree methods

The model problem is a second-order partial differential equation defined on a domain $\Omega \in \mathbf{R}^2$ [9]:

$$\begin{cases} -\nabla^2 u(x) = f(x), & x \in \Omega \\ u(x) = g(x), & x \in \Gamma_D \\ (\partial u/\partial n)(x) = h(x), & x \in \Gamma_N. \end{cases} \tag{1}$$

where $\Gamma_D \cup \Gamma_N$ is the boundary of Ω. For our numerical tests we let the domain Ω be $(0,1)\times(0,1)$. A Meshfree scheme, based on the Reproducing Kernel Partical Method (RKPM), is used to discretize the continuous problem (1). RKPM a function $u(x)$ by $u^h(x) = \sum_k^{NP} u_k \Psi_k(x)$, where NP stands for the number of particles, Ψ_I is the basis function for particle k, and u_k is its coefficient. Ψ_k is usually constructed by the product of a given kernel function $\Phi_a(x-x_k)$ and a correction function $C(x;x-x_k)$ to ensure that the discrete reproducing kernel condition is satisfied: $x^\alpha = \sum_{k=1}^{NP} \Psi_k(x) x_k^\alpha$ where $x^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_d^{\alpha_d}$, and the relationship holds for all non-negative integer vectors α where $|\alpha| := \sum_{i=1}^d \alpha_i \le p$ for some pre-determined p. The Φ_a functions are typically chosen to be tensor products of B-splines: $\Phi_a(x-x_k) = \prod_{i=1}^d \varphi((x_i-(x_k)_i)/a_i)$ where φ is a standard B-spline function.

To apply the above RKPM to the model problem (1), two sets of basis functions are generated separately on the domain Ω and the boundary Γ_D. The Lagrange Multiplier approach is used to handle the essential boundary conditions and the obtained meshfree linear system $Kx = F$ is of a saddle-point type:

$$Kx := \begin{bmatrix} A & B^T \\ -B & 0 \end{bmatrix} \begin{bmatrix} u \\ \lambda \end{bmatrix} = \begin{bmatrix} c \\ -d \end{bmatrix} \tag{2}$$

where $A_{ij} = \int_\Omega ((\nabla\Psi_i)^T \nabla\Psi_j + \Psi_i\Psi_j)\,dx$, $B_{ij} = \int_{\Gamma_D} \widetilde{\Psi}_i \Psi_j dS$, $c_i = \int_\Omega f\Psi_i dx + \int_{\Gamma_N} h\Psi_i dS$, and $d_i = \int_{\Gamma_D} g\widetilde{\Psi}_i dS$. Note that A is symmetric positive semi-definite, so that K is

non-symmetric but semi-definite. Even with this way of representing the problem, we need to ensure that the "inf-sup" or Ladyzhenskaya–Babuška–Brezzi (LBB) conditions are satisfied in order to ensure that the results are accurate:

$$\inf_w \sup_z \frac{\langle w, Bz\rangle}{\|w\|\,\|z\|_A} \geq \beta, \qquad \text{where } \|z\|_A = \sqrt{\langle z, Az\rangle}. \tag{3}$$

In order to expect convergence of the discrete approximations (2) we require that (3) holds *with the same* $\beta > 0$ regardless of how fine the discretization is.

With meshfree basis functions and $\widetilde{\Psi}_j = \Psi_j$, unfortunately, the LBB condition (3) is rarely satisfied. If the support of a basis function does not intersect the boundary then it causes no problems for the LBB condition. However, if the support of a basis function intersects the boundary *just a little*, there are severe problems for the LBB condition. Unlike standard finite element methods, there is no mesh to control the supports of the basis functions in meshfree methods, and so this is a likely occurrence for meshfree methods. Penalty methods in particular are likely to perform very badly with meshfree methods.

In order to overcome these problems, we use *an independently generated set of basis functions on the boundary* [9]. These can also be generated as meshfree functions, but using an different family of kernel functions $\widetilde{\Phi}_i$ on the boundary $\partial\Omega$. The size of the supports of the boundary basis functions $\widetilde{\Psi}_j$ should not be small in comparison with the size of the supports of the basis functions Ψ_i on Ω.

$\mathscr{H}$-matrices

The concept and properties of $\mathscr{H}$-matrices are induced by the index cluster tree T_I and the block cluster tree $T_{I\times I}$, which we now describe.

An index cluster tree T_I gives a hierarchy of partitions over an index set $I = \{0,\ldots,n-1\}$. It has the following properties: the root of T_I is I; any node $i \in T_I$ either is a leaf or has children $S(i)$; the parent node $i = \bigcup_{j\in S(i)} j$ and its children are disjoint (if $j_1,\, j_2 \in S(i)$, then either $j_1 = j_2$ or $j_1 \cap j_2 = \emptyset$).

A block cluster tree $T_{I\times I}$ is a hierarchical partition tree over the product index set $I \times I$. Given T_I and admissibility conditions (see [1,6,2]), $T_{I\times I}$ can be constructed as follows: the root of $T_{I\times I} = I \times I$; if $s \times t \in T_{I\times I}$ satisfies an admissibility condition, then it is Rk-matrix leaf; else if $\#s < N_s$ or $\#t < N_s$, then it is a full-matrix leaf; otherwise it will be partitioned further into subblocks on the next level and its children (subblocks) are defined as $S(s\times t) = \{\, i\times j \mid i, j \in T_I \text{ and } i \in S(s),\ j \in S(t)\,\}$.

Admissibility conditions are used to determine whether a block can be approximated by a Rk-matrix, and are important for maintaining the accuracy of the approximations. Classical $\mathscr{H}$-matrix admissibility conditions are geometric [1,6,2], and typically require the support sets to be sufficiently distant from each other. We will see later that we can develop combinatorial admissibility conditions that

also work well. Usually a block will not been partitioned into very small sub-blocks. In order to maintain the efficiency of the $\mathscr{H}$-matrix arithmetic, a constant $N_s \in [10, 100]$ is used to control the size of the smallest blocks.

Now we can define an $\mathscr{H}$-matrix H induced by $T_{I\times I}$ as follows: H shares the same tree structure with $T_{I\times I}$; the data are stored in the leaves; for each leaf $s \times t \in T_{I\times I}$, its corresponding block $H_{s\times t}$ is a Rk-matrix, or a full matrix with $\#s < N_s$ or $\#t < N_s$. Fig. 1 shows an example of T_I, $T_{I\times I}$ and corresponding $\mathscr{H}$-matrix. In this example each node in T_I has exact two children or none.

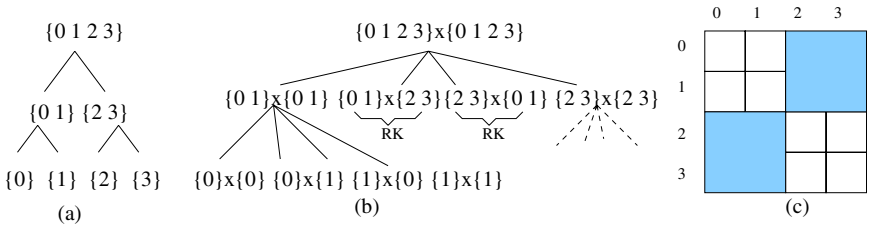


Figure 1: (a) is T_I, (b) is $T_{I\times I}$ and (c) is corresponding $\mathscr{H}$-matrix. The dark blocks in (c) are Rk-matrix blocks and the white blocks are full matrix blocks.

An algebraic approach for $\mathscr{H}$-matrix construction

In this section we will give a brief review of the algebraic approach to $\mathscr{H}$-matrix construction, which is based on multilevel clustering methods [10].

Multilevel clustering methods are widely used in graph partitioning, which generate a sequence of coarse graphs and corresponding clusters. Here we adapt multilevel clustering methods to build a sequence of coarse graphs over the matrix graph, which is built from a sparse matrix.

Given a symmetric sparse matrix M, its corresponding weighted undirected graph G_0 has the nodes $V(G_0) = I := \{0, 1, \ldots, n-1\}$, the matrix index set; there is an edge $e_{ij} \in E(G_0)$ if and only if the matrix entry $m_{ij} \neq 0$. An algorithm based on Heavy Edge Matching (HEM) [8] is used to build clusters over the nodes in $G_i = (V(G_i), E(G_i))$ and construct a coarser graph $G_{i+1} = (V(G_{i+1}), E(G_{i+1}))$. The HEM algorithm for the coarsening process is as follows: initially mark all the nodes as unmatched; randomly pick up an unmatched node s; among the edges that connect the node s to the other unmatched nodes, choose a node t with the maximum edge weight; mark both the node s and node t as matched, and create a cluster k at the current level i, i.e. $C_k^{(i)} = \{s, t\}$; if node s is isolated, then mark it as matched immediately and $C_k^{(i)} = \{s\}$. Repeat the above process until all the nodes are marked as matched. After building the clusters, a coarse graph G_{i+1} is constructed such that for each cluster $C_k^{(i)} \subset V(G_i)$ there is node $k \in V(G_{i+1})$. The

edge weight w_{kt} of an edge $e_{kt} \in E(G_{i+1})$ is set to the sum of the weights of all the edges, which connect the nodes in cluster $C_k^{(i)}$ to the nodes in cluster $C_t^{(i)}$ in the graph G_i. Recursively applying the above coarsening process gives a sequence of coarse graphs $G_1, G_2, \ldots, G_h$. We end this sequence with G_h, when $\#V(G_h)$ is sufficiently small.

To build the block cluster tree $T_{I\times I}$, we first define the admissibility condition based on the sequence of multilevel graphs $G_0, G_1, G_2, \ldots, G_h$: if the node $s, t \in V(G_i)$ are *not* connected in G_i then the block $r \times s$ is not partitioned further and represented by a Rk-matrix. If $\#s < N_s$ or $\#t < N_s$ where s and t are connected in G_i then we represent the block $s \times t$ by a full matrix. Otherwise, $s \times t$ is an interior node in $T_{I\times I}$ and its children are $u \times v$ where $u \in C_s^{(i-1)}$ and $v \in C_t^{(i-1)}$.

Note that if G_0 is the graph of M, then the $s \times t$ block will be represented by an Rk matrix if that block is a zero submatrix of M. In this case the $\mathscr{H}$-matrix approximation is exact.

$\mathscr{H}$-matrix preconditioners for saddle-point systems

The $\mathscr{H}$-matrix preconditioner for the meshfree system is built using $\mathscr{H}$-LU factorization. We can compute LU factors of K based on the following factorization process:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} L1 & 0 \\ L2 & -L3 \end{bmatrix} \begin{bmatrix} L1^T & L2^T \\ 0 & L3^T \end{bmatrix}, \tag{4}$$

where $L1, L1^T$ are the Cholesky factors of A. In [10], $L1$ is obtained by $\mathscr{H}$-Cholesky factorization, while $L2$ is obtained by sparse matrix operators. As the the size of the problem increases, the time to compute $L2$ and $L3$ consists a significant part of the factorization time. In this paper we present the following scheme to represent the subblock B in the $\mathscr{H}$-matrix representation to speed up the whole factorization process.

Let I denote the row and column index set of the submatrix A, and J denote the row index set of B. We build the index cluster trees over I and J separately.

To build T_J over the row index set J of B, we simply use bisection: the root of T_J is J; J is partitioned evenly into two subsets J_1 and J_2, and set them as the two children of the root node; continue this process to each node until the size of the set associated is less than N_s. In this way T_J is a binary tree.

Then the block cluster tree $T_{J\times I}$ is built using the graph of B as a graph joining nodes in J with nodes in I. Since $\#J \ll \#I$ in our model problems, so in the above process to build the block cluster tree, we stop partitioning the row index set J while continue partitioning the column index set I until it is small enough. In this way the leaves of $T_{J\times I}$ are almost square, which speeds up the computation.

Algebraic multigrid preconditioners

In this section, we review algebraic multigrid methods (AMG) and an adaption of AMG preconditioners [9] to saddle-point problems from meshfree methods, which is compared with the $\mathscr{H}$-matrices preconditioner in the experimental results section.

The basic idea of AMG is to use all levels in a hierarchy of fine to coarse grids to eliminate errors. Unlike the classical multigrid methods, the grids used in AMG are not geometric grids of a problem but the indexes of a matrix. Consider the following symmetric positive definite system: $Au = b$, and let c and f denote the coarse grid and the fine grid. Assume the interpolation operators I_c^f, the restriction operators I_f^c and the coarse operator A_c are defined. The basic structure of AMG algorithm of two level grids can be defined as follows: On the fine grid relax $A_f u_f = b_f$ using a standard iterative method, then restrict the residual r_f to get $r_c \leftarrow I_f^c r_f$; on the coarse grid (recursively) approximately solve $A_c e_c = r_c$, and obtain $e_f \leftarrow I_c^f e_c$; add e_f to the approximate solution u_f, and relax the original linear system again.

The key part of AMG is to define the interpolation operator I_c^f, since I_f^c can be defined as $I_f^c = (I_c^f)^T$ and $A_c = I_f^c A_c I_c^f$ by the Galerkin approach. To define I_c^f, first the coarse grid needs to be constructed. Basically there are two ways to construct the coarse grid from the fine grid: the coarse grid points are defined as the subset of the fine grid points; or the coarse grid points are constructed by aggregating the fine grid points and each coarse grid point represents a cluster over the fine grid points.

In [12] an aggregation method is introduced for coarse grid construction. Let $\{C_0, C_1, \ldots, C_k\}$ be the set of clusters built over the fine grid points by aggregation, an interpolation operator can be defined by $(E_c^f)_{ij} = 1$ if $i \in C_j$ and zero otherwise. This is a simple approach but gives slow convergence. To speedup the convergence, a variant of the smoothed aggregation method of [12] is implemented in [9]: to compute each column of the interpolation matrix a local linear system is solved. To adapt this approach to the saddle-point system (2), the interpolation operator I_c^f for the domain Ω and the the interpolation operator $\widetilde{I}_c^f$ for the boundary Γ_D are built separately by applying the above smoothed AMG algorithm. Then the coarse grid matrices are $A_c = I_f^c A_f I_c^f$, and $B_c = \widetilde{I}_f^c B_f I_c^f$ where $\widetilde{I}_c^f = (\widetilde{I}_f^c)^T$ is the interpolation matrix for the column nodes of B_f.

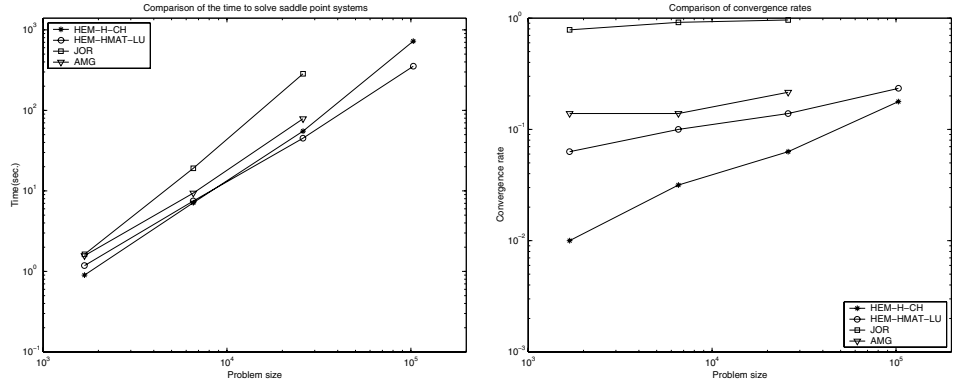
Experimental results

In this section, we will show the numerical results using the $\mathscr{H}$-matrix preconditioners obtained by the process in section 5 to solve the saddle-point system (2).

In our test, the number of the basis functions for the domain Ω is $N_\Omega = 1600$,

6400, 25600, 102400 and the corresponding number of the boundary basis functions is $N_\Gamma = 80, 160, 320, 640$. Thus the problem sizes are $n = 1680, 6560, 25920,$ and 103040 respectively.

We use $\mathscr{H}$-matrix arithmetic with adaptive ranks: the rank of each Rk-matrix block $M_{L_i \times L_j}$ approximating a matrix A in a $\mathscr{H}$-matrix satisfies $\text{rank}(M_{L_i \times L_j}) = \min\{\, k \mid \sigma_k \leq \alpha \sigma_1 \,\}$, where σ_k is kth largest singular value of A, and α is a parameter to control the accuracy which was set to $\alpha = 0.0625$. GMRES iteration stops where the original residuals were reduced by the factor of 10^{-12}. We set $N_s = 40$ in our test. In the experiments we compare the performance of four preconditioners in GMRES: JOR, AMG [9], the $\mathscr{H}$-matrix LU factors. The results are plotted in the log-log scale. All the experiments were performed on a Dell workstation with dual processor-Xeon 2.4GHz clock speed, and 1GB memory. The Meschach library [11] is used for the data structures and functions related to full matrices and vectors.



(a) Total time to solve the saddle-point systems with various preconditioners

(b) Convergence rates of various preconditioner per GMRES iteration

Figure 2: Comparison of preconditioners: total time and convergence rates

Fig. 2(a) shows the total time (including the time of building $\mathscr{H}$-matrices, building the preconditioners and GMRES iterations). 'HEM-H-CH' indicates the factorization method in [10] and 'HEM-HMAT-LU' indicates the factorization scheme described in this paper. Fig. 2(b) shows the average convergence rates of the various preconditioners. As respect to the total running time and the convergence rates, the $\mathscr{H}$-matrix based preconditioners give better performance than JOR and AMG. When the problem size is bigger than 10^4 the 'HEM-H-CH' time shows a sharp increase because of the sparse matrix blocks of the factors while 'HEM-HMAT-LU' needs the least time. As the problem size increases to around 10^5, the convergence rates of 'HEM-H-CH' and 'HEM-HMAT-LU' become very close.

Overall both $\mathscr{H}$-matrix based preconditioners outperform JOR and AMG. With

the increase of the problem size 'HEM-HMAT-LU' shows overall better performance than 'HEM-H-CH'.

References

1. S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. *EABE*, 27:403–564, 2003.
2. Sabine Le Borne and Lars Grasedyck. $\mathscr{H}$-matrix preconditioners in convection-dominated problems. *SIAM J. Matrix Anal. Appl.*, 27(4):1172–1183, 2006.
3. Sabine Le Borne and Lars Grasedyck. H preconditioners in convection-dominated problems. *SIAM J. Matrix Anal. Appl.*, 27(4):1172–1183, 2006.
4. Sabine Le Borne, Lars Grasedyck, and Ronald Kriemann. Domain-decomposition based H-LU preconditioners. *In Proceedings of the 16th International Conference on Domain Decomposition Methods (New York, 2005)*, LNCSE. Springer, 2006. To appear.
5. J.-S. Chen, C. Pan, C. T.Wu, andW. K. Liu. Reproducing kernel particle methods for large deformation analysis of non-linear structures. *Compt. Methods Appl. Engrg.*, 139:195–227, 1996.
6. L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathscr{H}$-matrices. *Computing*, 70(4):295–334, 2003.
7. W. Hackbusch. A sparse matrix arithmetic based on $\mathscr{H}$-matrices. part i: Introduction to $\mathscr{H}$-matrices. *Computing*, 62:89–108, 1999.
8. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM J. Sci. Comput.*, 20(1):359–392, 1999.
9. K. H. Leem, S. Oliveira, and D. Stewart. Algebraic multigrid (AMG) for saddle point systems from meshfree discretizations. *Numerical Linear Algebra and Applications*, 11(3):293–308, 2004.
10. Suely Oliveira and Fang Yang. An algebraic approach for H-matrix preconditioners. *Computing*, 2006. Submitted.
11. D. E. Stewart and Z. Leyk. *Meschach: Matrix Computations in C, volume 32 of Proceedings of the CMA*. The Australian National University, 1994.
12. P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing.*, pages 179–196, 1996.